Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.078873

ARTICLE

Peer-to-Peer IoT Authentication Protocol Based on PUF and Multiple Reference Fuzzy Extractor

Qingyao Gu[1,2,#], Mengqi Hu[2,#], Zerui Zhao[2], Liquan Chen[2,*] and Huiyu Fang[2]

[1]School of Integrated Circuits, Southeast University, Nanjing, China
[2]School of Cyber Science and Engineering, Southeast University, Nanjing, China
*Corresponding Author: Liquan Chen. Email: lqchen@seu.edu.cn
[#]These authors contributed equally to this work

Received: 09 January 2026; Accepted: 06 March 2026; Published: 08 May 2026

ABSTRACT: With the rapid development of the Internet of Things (IoT), the widespread adoption of applications such as smart homes and industrial IoT has raised the demand for secure authentication and key agreement among resource-constrained devices over open communication channels. Traditional authentication protocols often rely on centralized servers for key distribution, which results in high communication overhead and exposes systems to single-point-of-failure risks. Moreover, IoT devices are typically constrained in computational resources and are vulnerable to hardware cloning. These limitations necessitate lightweight yet robust security mechanisms. To address these challenges, we propose a lightweight peer-to-peer authentication protocol based on Physically Unclonable Function (PUF) and Multiple Reference Fuzzy Extractor (MRFE). The proposed protocol enables direct mutual authentication and key agreement between IoT devices without the participation of a trusted third-party server. Formal security analysis, along with evaluations of computation and communication costs, demonstrates that the protocol achieves strong security guarantees while maintaining high efficiency. Therefore, the proposed protocol is well-suited for lightweight peer-to-peer authentication scenarios in IoT environments.

KEYWORDS: Internet of Things; physically unclonable function; multiple reference fuzzy extractor; peer-to-peer authentication

1 Introduction

The rapid development of the Internet of Things (IoT) has converted a massive number of terminal devices from isolated entities into collaborative network nodes that interact via cloud and edge infrastructures. Under this circumstance, identity authentication and the establishment of secure communication channels among devices are fundamental to ensuring system security. However, conventional protocols based on complex public key infrastructure (PKI) [1,2] or cryptographic primitives [3–5], while offering strong security guarantees, generally incur significant computation and communication costs. This cost renders them unsuitable for resource-constrained IoT devices.

Physically Unclonable Function (PUF) [6], as an emerging hardware security primitive, exploits inherent manufacturing variations to provide uniqueness and unclonability. These properties make PUFs promising candidates for lightweight authentication mechanisms [7]. Nevertheless, existing PUF-based authentication protocols often depend on third-party servers for authentication and key distribution [8–10]. Such server-assisted approaches typically involve multiple communication rounds, impose heavy

computational loads on resource-constrained IoT devices, and introduce risks of single points of failure. More critically, they fail to satisfy the demand for direct peer-to-peer authentication among IoT devices. Additionally, fuzzy extractors are commonly employed to mitigate noise in PUF responses, but their computational complexity remains a considerable burden for resource-constrained terminals [11,12].

To address these challenges, this paper proposes a lightweight peer-to-peer authentication and key agreement protocol that integrates PUF with Multiple Reference Fuzzy Extractor (MRFE). The proposed protocol supports direct mutual authentication between IoT devices after initial registration with the server, significantly reducing communication rounds and eliminating server dependency, thereby improving robustness and communication efficiency. Furthermore, by introducing MRFE into the peer-to-peer authentication context, we reduce the computational load on resource-constrained IoT devices while enhancing the reliability of authentication, as MRFE achieves high fault tolerance without requiring larger error correction blocks that cause overhead explosion in traditional fuzzy extractors.

From a security perspective, we formally prove the security of our protocol under the Real-or-Random (ROR) model and further validate it using the AVISPA tool. Comparative evaluations against existing protocols demonstrate that our protocol achieves high-level security while outperforming baseline protocols in terms of computational efficiency and communication overhead.

The main contributions of this paper are summarized as follows:

- A novel PUF-based peer-to-peer IoT authentication protocol is proposed, eliminating the requirement for pre-shared keys, PKI, or trusted third-party servers.
- MRFE is innovatively integrated into peer-to-peer authentication, reducing PUF computational overhead while enhancing its robustness against noise.
- Formal security verification is conducted using the ROR model and AVISPA tool, demonstrating that the proposed protocol resists common cryptographic and physical attacks.

The remainder of this paper is organized as follows. Section 2 reviews related works, while Section 3 introduces the necessary preliminaries. We present the system model and detail the proposed protocol in Section 4. A comprehensive security analysis is provided in Section 5, followed by a performance evaluation and comparison with existing solutions in Section 6. Finally, Section 7 concludes the paper.

2 Related Works

With the rapid evolution of the Internet of Things (IoT), designing secure, efficient, and decentralized authentication mechanisms for resource-constrained IoT devices has emerged as a pivotal research challenge. Existing works can be broadly classified into three categories:

2.1 PKI and Cryptography-Based Protocols

Early IoT authentication protocols were primarily based on PKI [1] and complex cryptographic algorithms such as RSA [3] and elliptic curve cryptography (ECC) [13]. Although these approaches still provide strong security guarantees, their high computational complexity and heavy communication overhead have rendered them unsuitable for IoT devices with limited computational and storage resources. Moreover, PKI-based systems introduce additional challenges such as complex certificate management and key distribution [2,4]. Although identity-based encryption (IBE) [14] partially alleviates the burden of public key management, large-scale IoT deployments still face bottlenecks in private key distribution and updates [15]. To address these challenges, Höglund et al. [16] proposed AutoPKI, which automates credential updates and trust transfer using compact C509 certificates to reduce management overhead. Nevertheless,

PKI-based schemes still depend on centralized CAs, resulting in single points of failure and increased latency in revocation and verification.

2.2 PUF-Based Authentication Protocols

To address the demand for lightweight authentication in IoT scenarios, Physically Unclonable Functions (PUFs) have emerged as promising hardware primitives due to their ability to generate unique and unclonable device fingerprints. Traditional PUF-based protocols, however, commonly adopt a server-centric model in which a central server is responsible for authentication and key distribution. For example, in Alladi et al. [8], a base station acts as a trusted third party to allocate session keys for UAVs; in Wang et al. [17], a command center is required to establish keys for terminal devices; and Fan et al. [9] propose an industrial IoT authentication protocol that similarly depends on a domain server. Protocols such as T2T-MAP [10], PUF-RAKE [18], and Chatterjee et al. [19] demonstrate basic authentication capabilities but suffer from excessive communication rounds and vulnerabilities to single points of failure at the server. In addition, PUF-RAKE is insecure against continuous eavesdropping attacks, while Chatterjee et al. [19] requires storing secrets in non-volatile memory, exposing devices to physical extraction attacks. Recently, Nyangaresi et al. [20] proposed a cost-effective PUF- and ECC-based authentication protocol for secure Internet of Drones communications. However, the scheme still depends on a trusted third party for device-to-device authentication and incurs considerable computational overhead due to frequent ECC point multiplications.

2.3 Peer-to-Peer and Decentralized Protocols

To adapt to edge computing and decentralized environments, recent research has shifted toward peer-to-peer authentication protocols without third-party involvement. Clupek and Zeman [21] proposed an ultra-lightweight end-to-end authentication protocol, but it required devices to pre-store challenge-response pairs (CRPs), which increased storage overhead and risked impersonation attacks. Li et al. [22] designed a peer-to-peer authentication protocol integrating ECC, enabling two-way authentication without server involvement, though computationally expensive point multiplication operations imposed significant computational burdens on constrained IoT devices. Zheng et al. [23] proposed an optimized protocol that avoids private key or CRP storage while achieving perfect forward secrecy using ECDH, with physical prototypes demonstrating strong performance. Furthermore, Li et al. [24] introduced a flexible end-to-end protocol that achieves mutual authentication without real-time third-party intervention while providing enhanced anonymity against both verifiers and registration servers. Despite these advancements, most existing protocols still rely heavily on fuzzy extractors to ensure PUF response stability, which imposes non-negligible computational loads on devices [11,15,22–24]. Consequently, reducing the computational and storage burden of error-correction mechanisms while maintaining strong security remains an open research problem.

Existing authentication solutions either rely on heavyweight cryptography with high overhead or server-centric architectures that introduce single points of failure concerns. To address these limitations, this paper proposes a lightweight peer-to-peer authentication protocol based on PUF and Multiple Reference Fuzzy Extractor. The proposed design removes server dependency, reduces computational load on devices, and improves authentication stability, making it suitable for resource-constrained IoT environments.

3 Preliminaries

3.1 Physically Unclonable Functions (PUFs)

A Physically Unclonable Function (PUF) [6] is, in essence, a hardware function rooted in microscopic disparities in the physical structure of a device. Minute random variations inherent in hardware manufacturing processes are capitalized on to generate unique and unpredictable responses (Response, R) corresponding to distinct challenges (Challenge, C), i.e., $\mathrm{PUF}(\mathrm{C}) = \mathrm{R}$.

As shown in Fig. 1, PUF responses are mutually independent across different instances or challenges, yet remain consistent for the same challenge-instance pair. Consequently, PUFs can be regarded as the "hardware fingerprint" of devices.

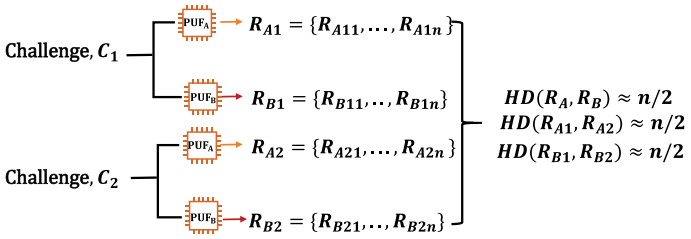

Figure 1: The challenge-response behavior of PUFs.

3.2 Fuzzy Extractor and Multiple Reference Fuzzy Extractor (MRFE)

Although PUFs exhibit strong uniqueness, their responses are affected by environmental noise such as temperature and voltage variations. To obtain stable and reliable keys, error-correction mechanisms like Fuzzy Extractors (FE) are employed. Introduced by Dodis et al. [25], an FE extracts uniformly random keys from noisy data through two core algorithms:

$\mathrm{FE.Gen}(\mathrm{R}) \rightarrow (\mathrm{K}, \mathrm{H})$: generates a key K and helper data H from a response R;

$\mathrm{FE.Rep}(\mathrm{R}', \mathrm{H}) \rightarrow \mathrm{K}$: reconstructs K from a noisy response R' and H if the error is within tolerance.

To enhance reliability in dynamic environments, the Multiple Reference Fuzzy Extractor (MRFE) [26] extends FE by maintaining several reference responses $\{\mathrm{R1}, \mathrm{R2}, \mathrm{R3}\}$ collected under different conditions.

In the generation phase, multiple key-helper pairs $\{(\mathrm{K1}, \mathrm{H1}), (\mathrm{K2}, \mathrm{H2}), (\mathrm{K3}, \mathrm{H3})\}$ are created; during the reproduction phase, the device iteratively applies $\mathrm{FE.Rep}(\mathrm{R}', \mathrm{H_i})$ until the correct key is recovered. This design slightly increases storage cost but significantly improves recovery accuracy, making MRFE ideal for resource-constrained IoT devices.

4 Proposed Protocol

In this section, we describe the system model, adversarial assumptions, and the proposed peer-to-peer authentication and key agreement protocol. The protocol consists of three phases: Setup, Registration, and Peer-to-Peer Authentication and Key Agreement.

4.1 System Model

As shown in Fig. 2, the IoT system comprises three main entities:

- End Devices (D): Resource-constrained IoT nodes (e.g., sensors, mobile devices) integrated with PUF modules. After initialization, these devices can mutually authenticate with each other and establish secure communication links.

- Cloud Server (CS): A trusted entity responsible for managing devices and securely storing PUF challenge–response pairs (CRPs). It assists only in the registration phase and is not involved in peer-to-peer authentication.
- Router (R): A communication relay between End Devices and the Cloud Server, forwarding messages without executing security-related operations.

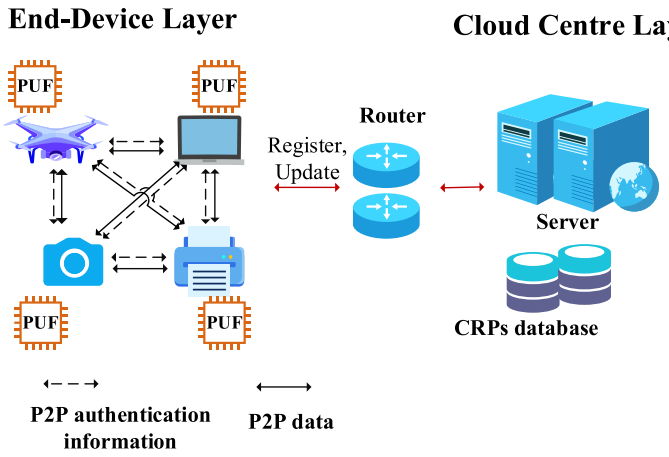

Figure 2: Communication system model.

We adopt a strong adversarial model that combines the Dolev-Yao (DY) and Canetti-Krawczyk (CK) frameworks. The adversary is capable of fully controlling the open communication channel (i.e., eavesdropping, modification, replay, and injection of messages) and may also acquire sensitive information through physical compromise of device memory.

Security relies on the following assumptions:

- The cloud server is trusted and securely stores CRPs of devices.
- The PUF hardware is immune to cloning or physical tampering by the adversary.
- Intermediate secrets generated locally during protocol execution but not transmitted remain secure.

Table 1 lists the primary notations used in our protocol.

Table 1: List of primary notations.

Symbol	Definition
C	PUF challenge
R	PUF response
i, j	Index
R_A/R_B	multiple reference responses set of a challenge of device A/B
R_{ai}/R_{bi}	The *i*-th reference response of R_A/R_B
ID_A/ID_B	Identity of device A/B
h_A/h_B	Helper data set of multiple reference responses
H_A/H_B	Helper data set of response and ID mask
RNG	Random number generator

(Continued)

Table 1 (continued)

Symbol	Definition
FE.Gen()/FE.Rep()	Generation/Reproduction function of fuzzy extractor
H ( )	Hash function
TRS	Random seed
ID	Unique device identifier
m, n	Nonce
D, Q	Ciphertext
sk	Session key
ϕ	Secret mask
p	Message hash
k	Secret key
δ	Secret digest
T, t	Timestamp

4.2 Setup Phase

Prior to deployment, each device is provisioned with a PUF module and undergoes initialization within a trusted environment:

- Global cryptographic primitives (e.g., hash functions, fuzzy extractor algorithms) are agreed upon by all entities.
- Each device generates a unique identifier ID_i.
- The server queries the device with a predefined set of challenges under different environmental conditions (e.g., −25°C, 25°C, 80°C). For each challenge, the device utilizes the MRFE to produce multiple reference responses R_A: $\{R_{a1}, R_{a2}, R_{a3}\}$and the corresponding helper data $h_A = \{h_{a1}, h_{a2}, h_{a3}\}$. These pairs are then securely stored at the server.

4.3 Registration Phase

During registration, the server establishes mutual trust with each device and distributes the necessary parameters for future peer-to-peer authentication. The process (illustrated in Fig. 3) is summarized as follows:

Step R1: For device A, the server selects a challenge C_a from the challenge set C_A and retrieves $R_A = \{R_{a1}, R_{a2}, R_{a3}\}$ from the CRP database. The server then generates a random value m, which, together with ID_A, C_a, ID_B, C_b, is sent to device A. A uses its PUF circuit to generate a response R_a and compute $(k_a, h_a) = \text{FE.Gen}(R_a)$. It then returns to the server $p_a = h(k \oplus m)$ along with a random number t (serving as a timestamp).

Step R2: The server attempts to reconstruct the secret using MRFE.Rep( ) with stored helper data $h_A = \{h_{a1}, h_{a2}, h_{a3}\}$ and device A's reference responses $R_A = \{R_{a1}, R_{a2}, R_{a3}\}$. Specifically, the server iterates over the reference set $(\text{for i} = 1 \text{ to } 3)$ to recover the candidate secret key $k_{ai} = \text{FE.Rep}(R_{ai}, h_{ai})$. It then verifies the hash by checking if $H(k_{ai} \oplus m)$ matches p_a. If the verification succeeds, the server successfully authenticates device A, and the corresponding k_{ai} and R_{ai} are selected. The same process is repeated for device B.

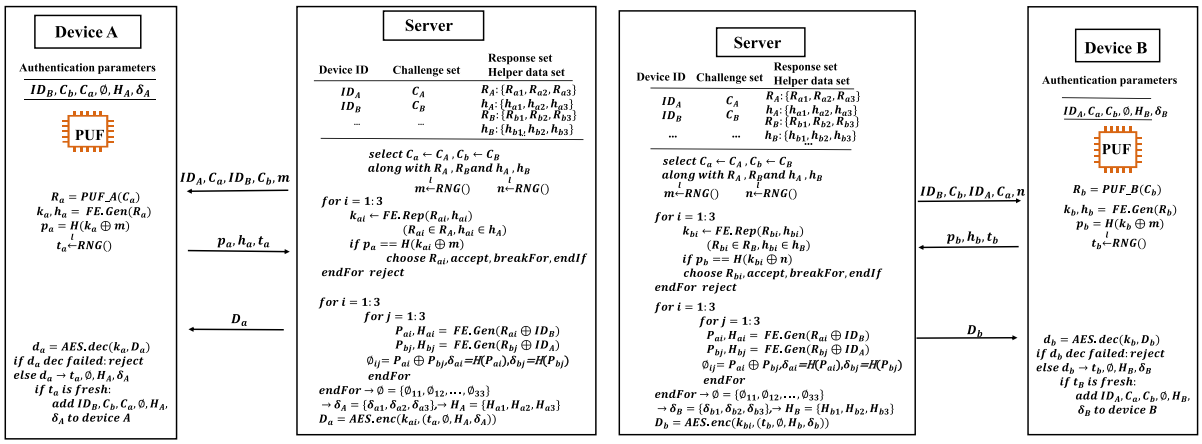

Figure 3: A/B-Server registration.

Step R3: A nested loop is then executed $(i, j \in \{1, 2, 3\})$ by the server, where fuzzy extractor instances FE.Gen $(R_{ai} \oplus ID_B)$ and FE.Gen$(R_{bj} \oplus ID_A)$ are applied to generate the corresponding helper data H_{ai}, H_{bj} and secrets P_{ai}, P_{bj}. The secret digest δ_{ai}, δ_{bj} is computed as $\delta_{ai} = H(P_{ai}), \delta_{bj} = H(P_{bj})$ and the secret mask $\varnothing_{ij}$ is derived as $\varnothing_{ij} = P_{ai} \oplus P_{bj}$ (this redundancy ensures that, in the authentication phase, the appropriate secret mask $\varnothing_{ij}$ can always be used to correctly recover the required secrets P_{ai} andP_{bj}, regardless of which reference responses are successfully reconstructed by devices A and B). Upon completion of the loop, the server obtains the final secret mask $\varnothing = \{\varnothing_{11}, \varnothing_{12}, \ldots, \varnothing_{33}\}$, the helper data set $H_A = \{H_{a1}, \ldots, H_{a3}\}, H_B = \{H_{b1}, \ldots, H_{b3}\}$, and the digest set $\delta_A = \{\delta_{a1}, \ldots, \delta_{a3}\}, \delta_B = \{\delta_{b1}, \ldots, \delta_{b3}\}$. Finally, the server sends $D_a = AES.enc(k_{ai}, (t_a, \varnothing, H_A, \delta_A))$ to device A (k_{ai} is previously selected from Step R2).

Step R4: Upon receiving D_a, device A uses the secret key k_a (generated in Step R1) to decrypt the ciphertext and obtain the data $d_a = \{t_a, \varnothing, H_A, \delta_A\}$. If the decryption is successful and t_a is fresh, device A successfully authenticates the server. Finally, A stores $\{ID_B, C_b, C_a, \varnothing, H_A, \delta_A\}$ for future peer-to-peer authentication. If a session key agreement between the terminal device and the server is required, both parties can independently compute the session key $ss_a = H(k_a, t_a)$ and $ss_s = H(k_{ai}, t_a)$, respectively.

At the end of this phase, devices A and B possess the necessary public parameters and auxiliary data to perform direct mutual authentication without further server involvement.

4.4 Peer-to-Peer Authentication and Key Agreement Phase

Once registered and deployed in the IoT environment, devices can mutually authenticate and negotiate a session key without server participation (see Fig. 4). The procedure is as follows:

Step A1: To initiate communication with device B, device A first generates a one-time random nonce T_a. Subsequently, A proceeds to reconstruct the secret information P_{ai} using the MRFE. Specifically, A applies its PUF circuit PUF_A(C_a) to obtain the response R_a, and iterates over the reference set (i = 1 to 3) with the corresponding helper data H_{ai}. In each iteration, A executes the FE's reproduction function FE.Rep (R_a, H_{ai}) to reconstruct the secret P_{ai}, and verifies whether $H(P_{ai})$ matches the stored digest δ_{ai}. If the verification is successful, the corresponding index i and the reconstructed secret P_{ai} are selected, and the loop immediately terminates. Otherwise, the process continues to the next iteration. If all attempts are exhausted without successful reconstruction, the authentication process is aborted. Finally, A transmits $(ID_A, C_a, ID_B, C_b, T_a, i)$ to B.

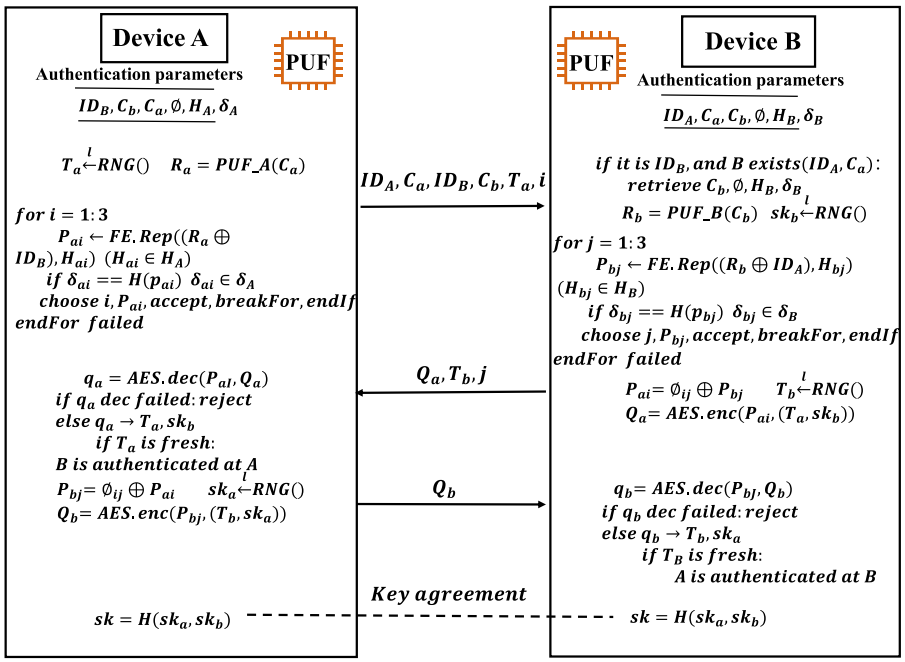

Figure 4: A–B mutual authentication and key agreement.

Step A2: Upon receiving the message, device B first checks whether the target identifier corresponds to its own identity. and verifies the validity of the included parameters (ID_A, C_a). Subsequently, B executes the reconstruction process with its PUF and helper data to retrieve P_{bj}. If the secret reconstruction is successful, B retrieves the secret j and P_{bj}. Otherwise, it rejects the request. Finally, device B generates a fresh random session secret sk_b, which will be used for subsequent session key negotiation.

Step A3: Device B generates a fresh random nonce T_b, and selects the secret mask $\varnothing_{ij}$ according to the parametersi andj. Then it computes the secret $P_{ai} = \varnothing_{ij} \bigoplus P_{bj}$. This reconstructed secret P_{ai} is used as an encryption key to obtain $Q_a \leftarrow AES.enc\,(P_{ai}, (T_a,\ sk_b))$, which is transmitted together with (T_b, j) to device A. Upon reception, device A attempts to decrypt Q_a using its previously reconstructed secret P_{ai} (obtained in Step A1) as the key. If decryption fails, the request is rejected. If successful, A confirms that device B holds the correct PUF response. After further verifying the freshness of T_a, device A successfully authenticates device B.

Step A4: Device A selects the secret mask $\varnothing_{ij}$ based on the received i andj, and computes $P_{bj} = \varnothing_{ij} \bigoplus P_{ai}$. A generates a fresh, random session secret sk_a for session key establishment. A constructs the ciphertext: $Q_b \leftarrow AES.enc\,(P_{bj}, (T_b,\ sk_a))$, and transmits Q_b to B. Device B attempts to decrypt Q_b using its previously reconstructed secret P_{bj} (obtained in Step A2) as the key. If decryption is successful, it confirms that device A holds the correct PUF response and validates the freshness of T_b, thereby authenticating device A. With both sk_a and sk_b successfully exchanged, the final session key sk is derived as: $sk = H(sk_a, sk_b)$.

Consequently, the mutual authentication is completed, and a secure session key sk is established between device A and device B.

5 Security Analysis

In this section, we analyze the security properties of the proposed protocol.

5.1 Security Assumptions

Before proceeding with the analysis, we first establish several security assumptions.

Definition 1: *Decisional Uniqueness Problem (DUP)*

The security of the proposed protocol relies on the DUP of PUFs [19]. This problem states that it is computationally infeasible for an adversary $\mathcal{A}$ to distinguish a PUF response from a truly random value, thereby ensuring unpredictability and security.

Consider a target device with PUF_t and an arbitrary PUF instance PUF_{adv} (which may be owned by the adversary). Given a random challenge C, the adversary attempts to determine whether a response z originates from $PUF_t(C)$ or from a uniform random distribution. The adversary's advantage is bounded by a negligible probability ε_{puf}:

$$\left| \begin{matrix} PR\left[\mathcal{A}, PUF_{adv}, z \overset{R}{\leftarrow} \{0,1\}^l = 1\right] - \\ PR\left[C,\ PUF_{adv}, z = PUF_t(C)\right] \end{matrix} \right| \leq \varepsilon_{puf} \tag{1}$$

Since PUF responses exhibit randomness-like behavior, even if the adversary possesses another PUF instance PUF_{adv}, it cannot reliably predict the behavior of the target PUF. Moreover, the unclonability of PUFs prevents the adversary from fabricating a device with the same response characteristics. According to the DUP definition, the adversary's maximum success probability of distinguishing between the PUF output and a random value is negligible, i.e., bounded by ε_{puf}. Similarly, when treating PUF outputs as inputs to the fuzzy extractor, the resulting secret key P is indistinguishable from a random number.

Definition 2: *(Entropy Smoothing Property of Hash Functions* [27])

The entropy smoothing property of a hash function can be intuitively understood as its resistance to collision. Even if the input data is non-random, the output $H_k(s)$ should be computationally indistinguishable from a uniformly random value h. As long as the output length is sufficient, the probability of successful distinction by an adversary is negligible.

Formally, the adversary's maximum advantage in distinguishing between the hash output and a random value is defined as:

$$\left| \begin{matrix} PR[k \overset{R}{\leftarrow} K, s \overset{R}{\leftarrow} G: A(k, H_k(s)) = 1] - \\ PR[Ck \overset{R}{\leftarrow} K, h \overset{R}{\leftarrow} \{0,1\}^l, A(k,h)] \end{matrix} \right| \leq \varepsilon_{es} \tag{2}$$

where A is the adversary's algorithm attempting to distinguish between $H_k(s)$ and a random value h. Here, k is the randomly chosen key (e.g., the hash family parameter), s is the input (such as PUF responses or keying material), and $H_k(s)$ is the hash output. The random value h is uniformly distributed with the same length as $H_k(s)$. The term ε_{es} denotes a negligible security bound.

5.2 Formal Security Analysis Using ROR Model

According to the system model, each participant device in the IoT network is denoted as D^i. In the ROR security model, an adversary can interact with oracles by sending queries to simulate various types of protocol attacks, and multiple concurrent instances may exist simultaneously [28]. Based on the CK model, the adversary can access the oracle set Oracles(), including {Execute, Send, Corrupt, Test, RO_{Hash}, RO_{puf}, RO_{sv}}, for each entity. These queries are defined as follows:

Execute (D^i, D^j): Simulates a passive eavesdropping attack, where the adversary obtains the exchanged messages between two honest parties D^i and D^j.

Send (E, msg): Simulates an active attack, where the adversary sends a message msg to entity E. The oracle then returns the response generated by the protocol execution, covering replay, modification, injection, or deletion.

Corrupt (E): Simulates a physical compromise attack, where the adversary tampers with entity E and retrieves all internal storage information.

Test (E): Used to evaluate the confidentiality of the session key, i.e., whether an attacker can distinguish between the real session key and a random value. Before the first query, Entity E generates a random bit b. if b = 1, the oracle returns the real session key sk; if b = 0, it returns a random value of the same length. Otherwise, an invalid symbol $\perp$ is returned [29].

If the adversary cannot distinguish sk from a random value with probability significantly greater than 1/2, then the protocol is considered secure. The adversary's advantage in breaking session key semantic security (SS) is defined as:

$$\mathrm{Adv}_{\pi}^{SS}(\mathcal{A}) = \left|\Pr\left[b' = b\right] - \frac{1}{2}\right| \tag{3}$$

Counter with Cipher Block Chaining-Message Authentication Code (CCM) is a block cipher mode of operation that combines CBC-MAC for authentication with CTR mode for encryption, typically applied to the 128-bit AES algorithm. CBC-MAC first authenticates the transmitted data, followed by CTR mode to perform encryption. For an n-block plaintext, CCM requires 2n + 2 AES operations [27]. In our protocol, AES is employed in the CCM mode, modeled as a Pseudorandom Permutation (PRP), mapping $\{0,1\}^{l_b} \rightarrow \{0,1\}^{l_b}$.

Theorem 1: *The adversary's advantage against CCM security is bounded as* [30]:

$$\mathrm{Adv}_{\mathrm{CCM_F}}^{\mathrm{auth}} \leq \varepsilon_{\mathrm{prp}} + \frac{q_F}{2^{l_t}} + \frac{(n_E + n_F)^2}{2^{l_b}} \tag{4}$$

where q_F, l_b, l_t, n_E, n_F denote the number of forgery queries, the block length, the tag length, the numbers of encryption, and decryption queries, respectively. For a secure PRP, ε_{prp} is negligible. In our protocol, AES with a 256-bit key and 128-bit block length is selected, requiring 6 AES operations when $n = 2$.

To rigorously quantify the security of the proposed protocol, we employ a game-based proof strategy consisting of a sequence of games Game $G_{0\sim5}$. This approach systematically bounds the adversary's advantage by transitioning from the real protocol execution to an idealized scenario where the session key is indistinguishable from a random value. The sequence accounts for various attack vectors, including the adversary's ability to distinguish random oracle outputs (for hash functions and PUFs), exploit potential collisions in random nonces or PUF responses, guess long-term secrets, and forge ciphertexts against the AES-CCM encryption scheme. The cumulative advantage derived across these games establishes the upper bound presented in the following Theorem 2.

Theorem 2: *When a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ attempts to break the semantic security of the session key* via *oracle queries, its advantage is bounded by:*

$$\mathrm{Adv}_{\pi}^{SS}(\mathcal{A}) \leq \frac{8q_s + {q_h}^2 + {q_p}^2 + 37(2q_s + q_e)^2}{2^l} + 4\varepsilon_{es} + 2\varepsilon_{puf} + \varepsilon_{prp} \tag{5}$$

where q_s, q_e, q_h, q_p denote the number of $Send$, $Execute$, RO_{Hash}, and RO_{puf} queries, respectively, and l denotes the output length of the hash function, PUF-derived key, or CCM block cipher.

Proof: To prove the session key security of the protocol, we define a series of games Game $G_{0\sim5}$ against adversary $\mathcal{A}$. Let G_i denote the event that the adversary correctly guesses the bit b in the Test query during

Game i. The success probability in Game i is $\Pr[G_i](\Pr\left[b' = b|\text{Game i}\right])$, and the probability difference between two consecutive games is given as: $|\Pr[G_i] - \Pr[G_{i-1}]|(i > 1)$.

Game 0: This is the initial attack game, where the bit b is chosen randomly. In the ROR model, Game 0 and the original protocol are identical. We have:

$$\mathrm{Adv}_{\pi}^{SS}(\mathcal{A}) = \left|\Pr[G_0] - \frac{1}{2}\right| \tag{6}$$

Game 1: Game 1 is essentially identical to Game 0, except that the hash function and the PUF are replaced by their corresponding random oracles, denoted as RO_{Hash} and RO_{PUF}, respectively. This game models an adversary attempting to distinguish the real hash function and PUF used in the protocol from ideal random functions through observing or interacting with protocol messages. During the peer-to-peer authentication and key agreement process, the protocol invokes the hash function four times for identity binding, authentication verification, and session key derivation, and invokes the PUF twice, once at each device. Therefore, by the union bound and the assumptions in Section 5.1, the adversary's distinguishing advantage between is bounded as:

$$|\Pr[G_1] - \Pr[G_0]| \leq 4\varepsilon_{es} + 2\varepsilon_{puf} \tag{7}$$

where ε_{es} and ε_{puf} denote negligible advantages for distinguishing hash and PUF outputs, respectively. Here, ε_{es} and ε_{puf} emphasize the security of hash function outputs, ensuring they do not leak valid information. Furthermore, since the output of RO_{puf} is used as the input for the FE.Gen(), its resulting secret value (P) can be treated as a random number and output via the Random Oracle for Secret Value (RO_{sv}). Consequently, in this protocol, the number of queries to both the RO_{puf} oracle and the RO_{sv} oracle is q_p.

Game 2: In this game, we analyze the impact of collision events that may occur during repeated protocol executions. The adversary attempts to distinguish the session key by exploiting potential collisions in hash outputs, random nonces, or PUF responses. Hash collisions may arise when the adversary observes multiple authentication messages generated using the hash function through Send or Execute queries. Random number collisions correspond to repeated nonces or challenges generated across different protocol sessions. By the Birthday Paradox, the probability of a hash collision occurring is less than $\frac{q_h^2}{2^{l+1}}$. And the total number of non-repeated random number selections is at most $(q_s + q_e)$, thus the probability of a random number collision occurring is $\frac{(q_s+q_e)^2}{2^{l+1}}$. Considering the q_p PUF queries, the PUF collision probability is $\frac{q_p^2}{2^{l+1}}$. Thus, the adversary's advantage in this game is bounded as:

$$|P[G_2] - P[G_1]| \leq \frac{(q_s + q_e)^2}{2^{l+1}} + \frac{q_h^2}{2^{l+1}} + \frac{q_p^2}{2^{l+1}} \tag{8}$$

where q_s, q_e, q_p, q_h denote the number of Send, Execute, PUF, and Hash queries, respectively.

Game 3: Game 3 excludes the adversary's advantage from randomly guessing session secrets. The adversary may attempt to guess P_{ai}, P_{bj}, sk. Once guessed, the adversary could break Q_a, Q_b and derive sk. The adversary can use corrupt(A) or corrupt(B) and obtain the corresponding mask $\varnothing_{ij}$. They can then randomly guess R_a and calculate $P_{ai} = \mathrm{FE.Rep}(R_a, H_{ai})$, or directly guess P_{ai} to obtain the secret P_{ai}. The probability of correctly guessing P_{ai} or P_{bj} is at most $\frac{2q_s}{2^l}$, while the probability of guessing sk is at most $\frac{3q_s}{2^l}$. Hence, the adversary's advantage in this game is bounded as:

$$|P[G_3] - P[G_2]| \leq \frac{7q_s}{2^l} \tag{9}$$

Game 4: In this game, we consider the adversary's ability to forge valid ciphertexts protected by AES-CCM. The adversary observes legitimate encryptions Q_a and Q_b via Execute queries and attempts these ciphertext forgery through Send queries. According to Theorem 1, in the CCM mode the adversary can attempt to forge valid ciphertexts through Send queries. Since encryption can also be triggered by Execute queries, the adversary may perform at most q_s forgery attempts and $(q_s + q_e)$ encryption attempts (i.e., $q_F = q_s, n_E = (q_s + q_e)$), with each encryption attempt requiring 6 AES operations. And the Send attack will cause q_s decryption operations on the other end, we have $n_F = q_s$.

By Theorem 1, the adversary's advantage in this game is therefore bounded as:

$$|P[G_4] - P[G_3]| \leq \varepsilon_{prp} + \frac{q_s + (12q_s + 2q_e)^2}{2^l} \tag{10}$$

Game 5: Since each session key $sk = H(sk_a, sk_b)$ is derived from fresh randomness sk_a and sk_b, session key exposure does not affect other sessions, and thus:

$$|P[G_5] - P[G_4]| = 0 \tag{11}$$

In the final game, the adversary is assumed to perform all possible oracle queries. However, even with session key exposure, the adversary gains no advantage in distinguishing the hidden bit in the Test query. Hence, the adversary's success probability is equivalent to random guessing, i.e.,$P[G_5] = 1/2$.

Therefore, the adversary's overall advantage from Game 0 to Game 5 is expressed as:

$$Adv_\pi^{SS}(\mathcal{A}) = \left|P[G_0] - \frac{1}{2}\right| = |P[G_0] - P[G_5]| \leq |P[G_0] - P[G_1]| + \ldots + |P[G_4] - P[G_5]| \tag{12}$$

By combining inequalities Eqs. (6)–(10) into Eq. (11), we obtain the final bound:

$$\begin{aligned} Adv_\pi^{SS}(A) &\leq \frac{8q_s + 36(2q_s + q_e)^2}{2^l} + \frac{{q_p}^2 + {q_h}^2 + (q_s + q_e)^2}{2^{l+1}} + 4\varepsilon_{es} + 2\varepsilon_{puf} + \varepsilon_{prp} \leq \\ &\frac{8q_s + {q_h}^2 + {q_p}^2 + 37(2q_s + q_e)^2}{2^l} + 4\varepsilon_{es} + 2\varepsilon_{puf} + \varepsilon_{prp} \end{aligned} \tag{13}$$

The adversary's advantage in breaking the semantic security of the session key is negligible. □

5.3 Security Properties

5.3.1 Perfect Forward Secrecy (PFS)

Since the long-term secrets P_{ai} and P_{bj} are generated instantaneously by PUF responses, they possess an erasure property, making it infeasible for an adversary to extract or reuse them. Even if, in theory, P_{ai} and P_{bj} were compromised, the adversary still cannot derive the session key. While an enhanced scheme could employ elliptic curve cryptography (ECC) to achieve PFS based on the Elliptic Curve Discrete Logarithm Problem (ECDLP), this approach incurs high computational cost. Considering that PUFs inherently provide erasure functionality, our protocol achieves PFS without relying on ECC.

5.3.2 Session Key Management Security

The generation of session keys in the proposed protocol does not rely on a centralized server, thereby avoiding the dependency and communication delays associated with traditional third-party key distribution.

End devices directly negotiate session keys with each other, improving both efficiency and the confidentiality of session keys.

5.3.3 Mutual Authentication

During the registration phase, the server authenticates end devices using helper data and reconstructed PUF responses. In the mutual authentication phase, devices A and B generate secrets P_{ai} and P_{bj} from their PUFs and verify each other using $\varnothing_{ij}$ to generate the encryption key. This ensures mutual authentication between devices without server intervention, thereby guaranteeing the legitimacy of both communication parties.

5.3.4 Resistance to Physical Invasion and Cloning Attacks

It is acknowledged that, despite the physical unclonability of PUFs, recent studies have shown that modeling attacks based on machine learning techniques may attempt to predict PUF behavior given sufficient challenge–response pairs [31,32]. While such attacks are outside the formal threat model considered in this work, the proposed protocol incorporates several design choices to mitigate their practical risk. First, the protocol avoids exposing a large number of challenge–response pairs and restricts CRP usage to the registration phase with a trusted server. Second, hash functions and AES-CCM encryption are employed to prevent direct leakage of PUF-related information through public protocol messages. Finally, for practical deployments, lightweight physical protection mechanisms (e.g., shielding, random delays, or noise injection) can be combined with the proposed protocol to further increase the difficulty of modeling and invasive attacks.

5.3.5 Resistance to Man-in-the-Middle Attacks

Even if an adversary intercepts a message Q_a from device A and attempts to replay or modify it as Q_a', the forged message will fail verification because it does not correspond to a valid PUF-derived secret. Similarly, forged responses such as Q_b' cannot be accepted by device A. Therefore, man-in-the-middle attacks are effectively prevented.

5.3.6 Resistance to Replay Attacks

The protocol employs fresh random numbers and timestamps (m, t_a, T_a, T_b) for each authentication session. Any replayed message will fail freshness verification and be rejected. Furthermore, each session key is bound to ephemeral random values, ensuring that every session is unique and immune to replay attacks.

Compared with representative PUF-based authentication schemes, including both server-assisted and decentralized designs, the proposed protocol enables peer-to-peer authentication without third-party participation during the authentication phase. Moreover, it avoids storing sensitive secrets in non-volatile memory and achieves perfect forward secrecy without relying on ECC operations, thereby reducing computational overhead while maintaining strong security guarantees.

A qualitative comparison of security features and design characteristics is presented in Table 2.

Table 2: Features comparison.

Protocol	Secure session key	No third party	No NVM required	PFS	High-efficiency FE
[33]	√	×	√	×	×
[34]	√	×	√	√	×
[35]	×	×	√	×	×
[22]	√	√	√	√	×
[23]	√	√	√	√	×
ours	√	√	√	√	√

5.4 AVISPA Automated Tool Verification

We further verify the protocol using the AVISPA (Automated Validation of Internet Security Protocols and Applications) tool. As shown in Fig. 5, the verification results from both the On-the-Fly Model-Checker (OFMC) and the Constraint-Logic-based Attack Searcher (CL-AtSe) backends report a "SAFE" status. In the context of AVISPA, this status signifies that the protocol has been exhaustively analyzed under the Dolev-Yao intruder model, and no execution states were found where the specified security goals could be compromised. Specifically, the tool simulates a wide range of active attack traces, including Man-in-the-Middle (MitM) and replay attacks, by allowing the intruder full control over the communication channel to intercept, modify, or resend messages. The "SAFE" conclusion confirms that the protocol successfully maintains session key secrecy and ensures mutual authentication between peers, effectively resisting impersonation and unauthorized access even in the presence of a malicious adversary.

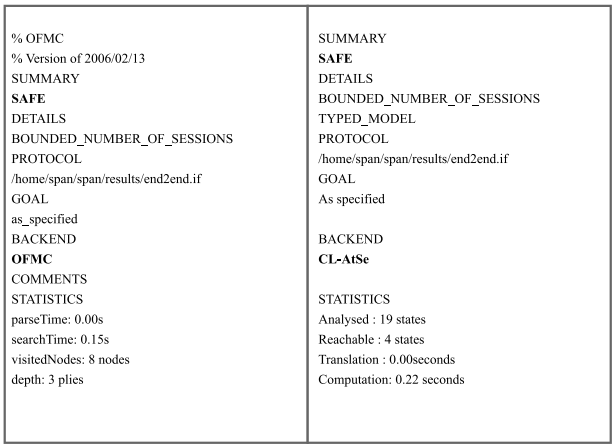

Figure 5: AVISPA simulation results.

6 Performance Evaluation

This section evaluates our protocol in terms of communication, storage, and computational cost, and compares it with existing protocols [22,23,33–35]. In addition, we compare the security features of the protocols.

6.1 Security Features

Table 2 summarizes the main functional features, including secure session key establishment, third-party dependence, confidential data storage in non-volatile memory (NVM), and perfect forward secrecy (PFS). All compared protocols provide basic mutual authentication.

Several protocols rely on servers for authentication, incurring higher latency and overhead. The protocol in [35] lacks session key generation. Moreover, the protocols in fail to ensure PFS once long-term secrets are compromised and require the participation of a third-party server. In contrast, the decentralized designs in [22,23] eliminate server dependency, reducing authentication latency and improving system scalability. Our protocol further introduces an efficient fuzzy extractor, eliminating server dependency and reducing computational load on constrained devices, thus enhancing both robustness and efficiency in decentralized IoT environments.

6.2 Computational Cost

Table 3 lists the computational parameters, where N_H, N_{PUF}, N_{PUF}, N_{ENC}, N_{MAC}, N_{PA}, N_{PM}, and N_{MRFE} represent the cost of hash, PUF, fuzzy extractor, encryption/decryption, MAC, elliptic curve addition/multiplication, and MRFE operations, respectively. Lightweight operations such as concatenation and XOR are ignored. Our protocol avoids third-party computation and removes expensive elliptic curve multiplication. Compared with [23], our scheme eliminates two hash function operations and, through MRFE-based error correction, enhances robustness while simultaneously reducing potential time overhead.

Table 3: Computational cost comparison.

Protocol	A, B	Third Party	Total Cost
[33]	$10N_H + 4N_{PUF} + 2N_{FE} + 2N_{ENC}$	$6N_H + N_{FE} + 2N_{ENC}$	$16N_H + 4N_{PUF} + 3N_{FE} + 4N_{ENC}$
[34]	$6N_H + 4N_{PUF} + 2N_{FE} + 4N_{PM} + 2N_{ENC}$	$6N_H + 2N_{ENC} + 2N_{FE}$	$12N_H + 4N_{PUF} + 4N_{FE} + 4N_{PM} + 4N_{ENC}$
[35]	$8N_{PUF} + 8N_{FE}$	0	$8N_{PUF} + 8N_{FE}$
[22]	$12N_H + 2N_{PUF} + 2N_{FE} + 6N_{PA} + 16N_{PM}$	0	$12N_H + 2N_{PUF} + 2N_{FE} + 6N_{PA} + 16N_{PM}$
[23]	$6N_H + 2N_{PUF} + 2N_{FE} + 4N_{ENC} + 4N_{MAC}$	0	$6N_H + 2N_{PUF} + 2N_{FE} + 4N_{ENC} + 4N_{MAC}$
ours	$4N_H + 2N_{PUF} + 2N_{MRFE} + 4N_{ENC} + 4N_{MAC}$	0	$4N_H + 2N_{PUF} + 2N_{MRFE} + 4N_{ENC} + 4N_{MAC}$

6.3 Communication and Storage Cost

Table 4 compares the communication and storage cost of our protocol with traditional protocols [22,23,33–35], assuming consistent parameter bit lengths across all protocols. Our protocol achieves lower communication overhead than some existing protocols, requiring only three rounds, which minimizes bandwidth use and latency.

Table 4: Communication and storage cost comparison.

Protocol	Communication Cost/Bit	Storage Cost/Bit	Communication Cycle
[33]	2944	896	4
[34]	2048	64	7
[35]	2144	5120	6
[22]	4672	4096	3
[23]	2934	2112	3
ours	2938	4672	3

Although compared with [23], our protocol incurs additional storage cost due to the MRFE's helper data (18 × 128 bits), which is negligible for modern IoT devices. In return, MRFE substantially lowers computational cost and enhances reliability under noisy conditions.

6.4 Robustness and Time Overhead Trend Analysis

In real-world IoT deployments, environmental fluctuations such as temperature, voltage, and electromagnetic interference can destabilize PUF responses, making robustness (the ability of a PUF to generate stable responses under noise) and fault tolerance key metrics.

Our protocol leverages multiple reference responses for redundant error correction, mitigating performance degradation under high bit error rates. Let P_{FE} denote the probability of correctly recovering the secret key using a single response in a traditional fuzzy extractor. In our MRFE scheme, a successful recovery is achieved if at least one of the m reference responses (m = 3 in our case) is successfully decoded. Consequently, the overall recovery probability is improved to $P_{MRFE} = 1 - (1 - P_{FE})^m$. This multi-reference reconstruction approach offers a stronger capability to retrieve stable keys even under significant noise. As shown in Fig. 6, our simulation results indicate that the recovery probability of the MRFE scheme declines more gradually with increasing BER compared to the traditional fuzzy extractor. This clearly demonstrates the superior robustness and environmental adaptability of our MRFE approach.

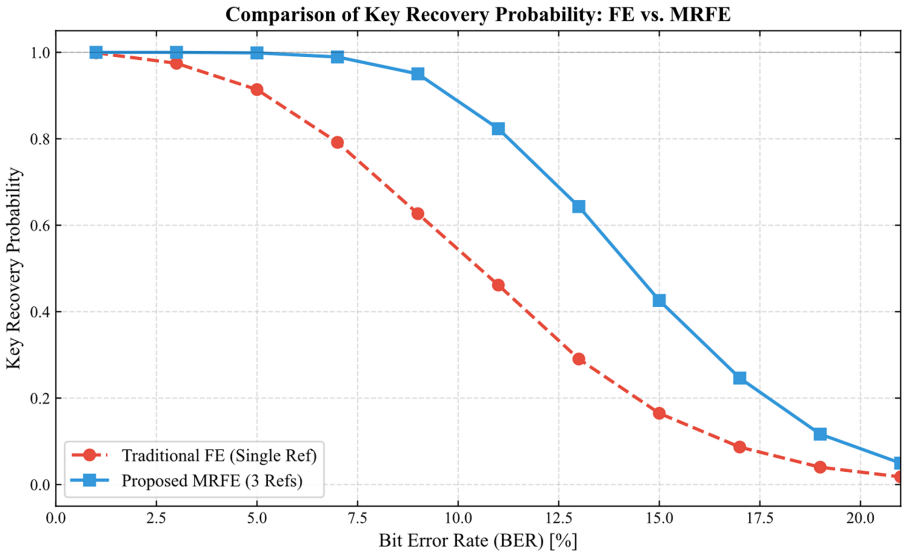

Figure 6: Key recovery probability vs. BER for FE and MRFE.

In addition, although the recovery phase of the Multiple Reference Fuzzy Extractor (MRFE) in this scheme needs to try multiple reference responses, resulting in a relatively high time overhead due to a large number of computation rounds. However, this overhead is converted into a core advantage in high fault tolerance scenarios. To meet high fault tolerance requirements, traditional schemes often need to improve error correction strength, which inevitably relies on extremely large error correction code blocks and ultimately triggers the problem of overhead explosion [36]. In contrast, MRFE's "one-out-of-multiple" strategy avoids this drawback. According to [26], MRFE can save over 40% time overhead compared with traditional FE under low failure rates. Through multi-reference response matching, it greatly reduces the dependence on ultra-large error correction code blocks, thereby cutting down the time overhead and further ensuring the efficient operation of the protocol on resource-constrained terminals.

6.5 Scalability Discussion

In terms of scalability, the proposed protocol relies on pairwise registration, which introduces specific constraints in large-scale deployments. To achieve full mesh connectivity in a network of N devices, the trusted server must generate unique secret masks for every distinct pair during the registration phase, resulting in a computational complexity of $O(N^2)$. Furthermore, each device is required to store authentication parameters for every potential peer, leading to a linear storage overhead of $O(N)$. Despite this initialization overhead, the online peer-to-peer authentication and key agreement phase remains highly efficient, requiring a constant number of communication rounds and no server involvement, regardless of network size. To further mitigate these scalability limitations, future work will explore group authentication extensions. These mechanisms aim to integrate threshold cryptography or broadcast-based key agreement with the MRFE framework, thereby decoupling device storage costs from the total number of network nodes.

Consequently, the proposed protocol achieves higher security and robustness with lower computational and communication overhead, making it well-suited for peer-to-peer IoT applications in resource-constrained and variable environments.

7 Conclusion

To address the lightweight authentication requirements of peer-to-peer IoT scenarios, this paper designs a novel authentication and key agreement (AKA) protocol based on PUF and MRFE. The proposed protocol eliminates the reliance on traditional pre-shared keys or PKI systems, thereby reducing storage and computational overhead. Furthermore, the protocol introduces an enhanced MRFE mechanism to effectively reduce computational cost under a given bit error rate, while its decentralized design avoids dependence on third-party servers, thereby mitigating risks of single points of failure and high latency inherent in centralized architectures. The security of the protocol was rigorously validated through formal analysis under the ROR model, proving the semantic security of the session key, and was further confirmed by the AVISPA automated tool against common threats like man-in-the-middle and replay attacks. Overall, the proposed protocol provides a secure, highly efficient, and lightweight solution that is superior to existing peer-to-peer IoT authentication protocols, offering significant potential for practical deployment in next-generation decentralized IoT systems.

Acknowledgement: This research work is supported by the Big Data Computing Center of Southeast University.

Funding Statement: This research work was funded by the National Natural Science Foundation of China (62572121, U22B2026), Natural Science Foundation of Xizang (XZ202501ZY0094), Frontier Technology R&D Program of Jiangsu (BF2025067), and Open Foundation of Key Laboratory of Cyberspace Security, Ministry of Education of China and Henan Key Laboratory of Network Cryptography (No. KLCS20240301).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Liquan Chen, Qingyao Gu, and Mengqi Hu; methodology, Qingyao Gu and Mengqi Hu; writing—original draft preparation, Qingyao Gu, Mengqi Hu, and Zerui Zhao; writing—review and editing, Liquan Chen and Huiyu Fang. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wazid M, Das AK, Shetty S. TACAS-IoT: trust aggregation certificate-based authentication scheme for edge-enabled IoT systems. IEEE Internet Things J. 2022;9(22):22643–56. doi:10.1109/JIOT.2022.3181610.
2. Khurshid A, Raza S. AutoCert: automated TOCTOU-secure digital certification for IoT with combined authentication and assurance. Comput Secur. 2023;124(4):102952. doi:10.1016/j.cose.2022.102952.
3. Mumtaz M, Akram J, Ping L. An RSA based authentication system for smart IoT environment. In: Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS); 2019 Aug 10–12; Zhangjiajie, China. p. 758–65. doi:10.1109/hpcc/smartcity/dss.2019.00112.
4. Singh K, Singh N. Multi-level authentication protocol for enabling secure communication in IoT. Preprint. 2021. doi:10.21203/rs.3.rs-382412/v1.
5. Zou S, Cao Q, Wang C, Huang Z, Xu G. A robust two-factor user authentication scheme-based ECC for smart home in IoT. IEEE Syst J. 2022;16(3):4938–49. doi:10.1109/JSYST.2021.3127438.
6. Pappu R, Recht B, Taylor J, Gershenfeld N. Physical one-way functions. Science. 2002;297(5589):2026–30. doi:10.1126/science.1074376.
7. Aman MN, Basheer MH, Sikdar B. Data provenance for IoT with light weight authentication and privacy preservation. IEEE Internet Things J. 2019;6(6):10441–57. doi:10.1109/JIOT.2019.2939286.
8. Alladi T, Naren, Bansal G, Chamola V, Guizani M. SecAuthUAV: a novel authentication scheme for UAV-ground station and UAV-UAV communication. IEEE Trans Veh Technol. 2020;69(12):15068–77. doi:10.1109/TVT.2020.3033060.
9. Fan CI, Lai CI, Vishwasrao Medhane D. CAKE-PUF: a collaborative authentication and key exchange protocol based on physically unclonable functions for industrial Internet of Things. IEEE Internet Things J. 2024;11(24):39709–20. doi:10.1109/JIOT.2024.3450959.
10. Lounis K, Zulkernine M. T2T-MAP: a PUF-based thing-to-thing mutual authentication protocol for IoT. IEEE Access. 2021;9:137384–405. doi:10.1109/ACCESS.2021.3117444.
11. Tian C, Jiang Q, Li T, Zhang J, Xi N, Ma J. Reliable PUF-based mutual authentication protocol for UAVs towards multi-domain environment. Comput Netw. 2022;218(2):109421. doi:10.1016/j.comnet.2022.109421.
12. Zhang S, Yan Z, Liang W, Li KC, Dobre C. BAKA: biometric authentication and key agreement scheme based on fuzzy extractor for wireless body area networks. IEEE Internet Things J. 2024;11(3):5118–28. doi:10.1109/JIOT.2023.3302620.
13. Li T, Ma J, Feng P, Meng Y, Ma X, Zhang J, et al. Lightweight security authentication mechanism towards UAV networks. In: Proceedings of the 2019 International Conference on Networking and Network Applications (NaNA); 2019 Oct 10–13; Daegu, Repubilc of Korea. p. 379–84. doi:10.1109/nana.2019.00072.
14. Boneh D, Franklin M. Identity-based encryption from the Weil pairing. In: Advances in cryptology—CRYPTO 2001. Berlin/Heidelberg, Germany: Springer; 2001. p. 213–29. doi:10.1007/3-540-44647-8_13.
15. Qureshi MA, Munir A. PUF-RAKE: a PUF-based robust and lightweight authentication and key establishment protocol. IEEE Trans Dependable Secur Comput. 2022;19(4):2457–75. doi:10.1109/TDSC.2021.3059454.
16. Höglund J, Bouget S, Furuhed M, Preuß Mattsson J, Selander G, Raza S. AutoPKI: public key infrastructure for IoT with automated trust transfer. Int J Inf Secur. 2024;23(3):1859–75. doi:10.1007/s10207-024-00825-z.
17. Wang D, Cao Y, Lam KY, Hu Y, Kaiwartya O. Authentication and key agreement based on three factors and PUF for UAV-assisted post-disaster emergency communication. IEEE Internet Things J. 2024;11(11):20457–72. doi:10.1109/JIOT.2024.3371101.
18. Roy S, Das D, Mondal A, Mahalat MH, Sen B, Sikdar B. PLAKE: PUF-based secure lightweight authentication and key exchange protocol for IoT. IEEE Internet Things J. 2023;10(10):8547–59. doi:10.1109/JIOT.2022.3202265.
19. Chatterjee U, Govindan V, Sadhukhan R, Mukhopadhyay D, Chakraborty RS, Mahata D, et al. Building PUF based authentication and key exchange protocol for IoT without explicit CRPs in verifier database. IEEE Trans Dependable Secur Comput. 2019;16(3):424–37. doi:10.1109/TDSC.2018.2832201.

20. Nyangaresi VO, Ahmad M, Maghrabi LA, Althaqafi T. Cost-effective PUF and ECC-based authentication protocol for secure Internet of drones communication. IEEE Internet Things J. 2025;12(16):33844–57. doi:10.1109/JIOT.2025.3576313.
21. Clupek V, Zeman V. Robust mutual authentication and secure transmission of information on low-cost devices using Physical unclonable functions and Hash functions. In: Proceedings of the 2016 39th International Conference on Telecommunications and Signal Processing (TSP); 2016 Jun 27–29; Vienna, Austria. p. 100–3. doi:10.1109/TSP.2016.7760837.
22. Li S, Zhang T, Yu B, He K. A provably secure and practical PUF-based end-to-end mutual authentication and key exchange protocol for IoT. IEEE Sens J. 2021;21(4):5487–501. doi:10.1109/JSEN.2020.3028872.
23. Zheng Y, Liu W, Gu C, Chang CH. PUF-based mutual authentication and key exchange protocol for peer-to-peer IoT applications. IEEE Trans Dependable Secur Comput. 2023;20(4):3299–316. doi:10.1109/TDSC.2022.3193570.
24. Li S, Huang Y, Yu B. A practical and flexible PUF-based end-to-end anonymous authentication protocol for IoT. Comput Netw. 2024;247(4):110426. doi:10.1016/j.comnet.2024.110426.
25. Dodis Y, Reyzin L, Smith A. Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In: Advances in cryptology—EUROCRYPT 2004. Berlin/Heidelberg, Germany: Springer; 2004. p. 523–40. doi:10.1007/978-3-540-24676-3_31.
26. Gao Y, Su Y, Xu L, Ranasinghe DC. Lightweight (reverse) fuzzy extractor with multiple reference PUF responses. IEEE Trans Inf Forensics Secur. 2019;14(7):1887–901. doi:10.1109/TIFS.2018.2886624.
27. Fouque PA, Martinet G, Valette F, Zimmer S. On the security of the CCM encryption mode and of a slight variant. In: Applied cryptography and network security. Berlin/Heidelberg, Germany: Springer; 2008. p. 411–28. doi:10.1007/978-3-540-68914-0_25.
28. Shoup V. Sequences of games: a tool for taming complexity in security proofs. IACR Cryptol Eprint Arch. 2004;2004:332.
29. Abdalla M, Fouque PA, Pointcheval D. Password-based authenticated key exchange in the three-party setting. In: Public key cryptography—PKC 2005. Berlin/Heidelberg, Germany: Springer; 2005. p. 65–84. doi:10.1007/978-3-540-30580-4_6.
30. Jonsson J. On the security of CTR + CBC-MAC. In: Selected areas in cryptography. Berlin/Heidelberg, Germany: Springer; 2003. p. 76–93. doi:10.1007/3-540-36492-7_7.
31. Alhamarneh RA, Mahinderjit Singh M. Strengthening Internet of Things security: surveying physical unclonable functions for authentication, communication protocols, challenges, and applications. Appl Sci. 2024;14(5):1700. doi:10.3390/app14051700.
32. Wang H, Hao W, Tang Y, Zhu B, Dong W, Liu W. Deep neural network modeling attacks on arbiter-PUF-based designs. Cybersecurity. 2025;8(1):11. doi:10.1186/s42400-024-00308-7.
33. Karmakar R, Kaddoum G, Akhrif O. A PUF and fuzzy extractor-based UAV-ground station and UAV-UAV authentication mechanism with intelligent adaptation of secure sessions. IEEE Trans Mob Comput. 2024;23(5):3858–75. doi:10.1109/TMC.2023.3284216.
34. Roy S, Mahalat MH, Sen B. Design and implementation of cost-effective end-to-end authentication protocol for PUF-enabled IoT devices. IEEE Trans Emerg Top Comput. 2025;13(3):1055–67. doi:10.1109/TETC.2025.3563064.
35. Barbareschi M, De Benedictis A, La Montagna E, Mazzeo A, Mazzocca N. A PUF-based mutual authentication scheme for Cloud-Edges IoT systems. Future Gener Comput Syst. 2019;101:246–61. doi:10.1016/j.future.2019.06.012.
36. Yu MD, Devadas S. Secure and robust error correction for physical unclonable functions. IEEE Des Test Comput. 2010;27(1):48–65. doi:10.1109/MDT.2010.25.