PUF-based Mutual Authentication and Key Exchange Protocol for Peer-to-Peer IoT Applications

Yue Zheng, Member, IEEE, Wenye Liu, Member, IEEE, Chongyan Gu, Senior Member, IEEE, and Chip-Hong Chang, Fellow, IEEE

Abstract—Peer to Peer (P2P) or direct connection IoT has become increasingly popular owing to its lower latency and higher privacy compared to database-driven or server-based IoT. However, wireless vulnerabilities raise severe concerns on IoT device-to-device communication. This is further aggravated by the challenge to achieve lightweight direct mutual authentication and secure key exchange between IoT peer nodes in P2P IoT applications. Physical unclonable function (PUF) is a key enabler to lightweight, low-power and secure authentication of resource-constrained devices in IoT. Nevertheless, current PUF-enabled authentication protocols, with or without the challenge-response pairs (CRPs) of each of its interlocutors stored in the verifier’s side, are incompatible for P2P IoT scenarios due to the security, storage and computing power limitations of IoT devices. To solve this problem, a new lightweight PUF-based mutual authentication and key exchange protocol is proposed. It allows two resource-constrained PUF embedded endpoint devices to authenticate each other directly without the need for local storage of CRPs or any private secrets, and simultaneously establish the session key for secure data exchange without resorting to the public-key algorithm. The proposed protocol is evaluated using the game-based formal security analysis method as well as the automatic security analysis tool ProVerif to corroborate its mutual authenticity, secrecy, and resistance against replay and man-in-the-middle (MITM) attacks. Using two Avnet Ultra96-V2 boards to emulate the two IoT endpoint devices, a physical prototype system is also constructed to demonstrate and validate the feasibility of the proposed secure P2P connection scheme. A comparative analysis shows that the proposed protocol outperforms related protocols in terms of security features, computational complexity as well as communication and storage costs.

Index Terms—Peer-to-peer Internet of Things, IoT Security, Physical Unclonable Functions, Peer-entity Authentication Protocol, Authenticated Key Exchange Protocol, Man-in-the-Middle Attacks

1 INTRODUCTION

THE Internet of Things (IoT) regime can be roughly classified into two categories, i.e., database-driven IoT and peer-to-peer (P2P) IoT [1]. For database-driven IoT, the connections between two IoT endpoint devices are established indirectly through a server, which is a common approach for IoT connections. Data collected from IoT devices will be transmitted to the server and stored in its database, and the retrieval of data can be achieved by sending a request to the server. By contrast, P2P IoT enables a direct connection between two IoT endpoints, which provides lower-latency performance and higher-privacy level compared to server-centric database-driven IoT.

As data collectors, IoT endpoints have access to high-value assets and confidential information. Since they are easily accessible and resource-constrained, they are also alluring targets of attack, especially when the intelligent autonomous endpoint devices are allowed to interact or exchange sensitive data directly with each other in a smart environment. Authentication and secure key exchange between two endpoints are the crucial first line of defense to prevent fake endpoints from exfiltrating data and holding information hostage. Unfortunately, traditional security solutions, despite sufficiently mature, do not transfer convincingly to the P2P IoT scenarios. Traditional mutual authentication (MA) and key exchange (KE) protocols rely on either symmetric or asymmetric key cryptographic primitives for security. Secret keys are required to be stored in the target device, which are difficult to protect due to the resource and memory constraints of IoT devices. Symmetric cryptography requires the secret keys to be securely pre-shared among each pair of devices. As the number of IoT nodes increases, key distribution and management become an issue for P2P device authentication without falling back to the server-centric IoT. Asymmetric cryptography avoids key sharing problem at a high price of computational cost, and the individual device still has to safekeep its own private key locally. Expensive asymmetric cryptography is a compelling option when a very high level of security is required but not a preferred primitive to be relied on for P2P device authentication.

To this end, Physical Unclonable Function (PUF) stands out as a promising embodiment to facilitate secure authentication and key exchange of IoT devices [2], [3], [4], [5], [6].
A PUF can be viewed as a physical circuit realization of a random oracle by harnessing the minute variance in modern semiconductor manufacturing processes [7]. The lynchpin of PUF is the irreversible random mapping of a digital input (known as a challenge) to a digital output (known as a response). The challenge-response mapping is very similar to a cryptographic hash function except that the hardness to invert the function is originated from the physical disorder instead of the computational complexity theory. The chip-to-chip variances of a manufacturing process can be harvested by the PUF circuit. With sufficient basic PUF cells, many unique challenge-response pairs (CRPs) of arbitrary length can be generated from each chip for a huge number of manufactured chips. Although the PUF circuit itself is easy to make and the responses can be readily measured, it is practically infeasible to physically clone a PUF instance to reproduce the same CRPs. More importantly, no secret key needs to be stored locally on a device as the PUF can generate the device-specific secret (response) only upon request (by applying a challenge). Once the required number of CRPs have been successfully measured and enrolled into a secure server database, the external measurement interface of the PUF responses can be permanently disabled. Since the entire set of CRPs is intricately embodied in the nano-structure of the PUF, any active manipulation of the PUF circuit internals will cause dysfunction of the challenge-response mapping mechanism and destroy the secret. This tamper-evident property of PUF and its ability to securely identify a device by interrogation without the need for a permanent secret residence in anti-temper memory largely reduce the risks of a number of powerful hardware attack vectors such as reverse engineering, probing and fault injection attacks on seizure devices [8], [9], [10], [11].

Using a PUF for device authentication requires the verifier to keep a copy of its prover’s CRPs. This is a common practice in PUF-based authentication protocols in database-driven IoT as the server has the computing power and memory resources to securely store the database of the CRPs of all its interlocutors. However, this requirement becomes impractical when mutual authentication has to be performed directly between two end devices for secure communication in P2P IoT applications. IoT end devices are usually small, low cost and resource-constrained, which renders classical recipes [2], [5], [6] of PUF-based mutual authentication between device and server inapplicable, especially when one end-device needs to communicate with several other end devices. Besides, storing the prover’s CRPs in the verifier device allows the latter to impersonate the prover, which rules out the applicability of traditional PUF-based authentication schemes in the P2P IoT scenarios.

This paper presents a lightweight device-to-device PUF-based authentication protocol for P2P IoT without requiring the devices to store each other’s CRPs. The concept was first proposed without implementation in our preliminary work [12]. That work still requires the server’s participation during the authentication phase and is vulnerable to the two-time pad attack. The shortcomings and overly simplistic security analysis in [12] are addressed in this paper. Our major contributions are:

1. The proposed mutual authentication scheme enables secure direct communication between any two IoT end-points. This enables the IoT data to be shared directly between a pair of endpoints. The server only helps to create the connection in the enrollment and update phase, and is then completely released from the authentication and communication phase of the system.
2. The proposed PUF-based scheme requires no keys or CRPs to be stored in the IoT endpoints. No extra intermediate “verifier” node is required, which is unlike the scheme in [12]. This makes it possible for a resource-constrained PUF-embedded device to directly authenticate any other PUF-embedded devices provided that they are enrolled by the same server.
3. Two secure and efficient key exchange options are offered by the proposed scheme. The proposed basic key exchange protocol can be augmented with perfect forward secrecy (PFS) by adding two elliptic curve point multiplications on each device. The shared keys are automatically established upon successful authentication, which offloads the complexity of managing and storing the keys for IoT devices.
4. The security of the protocol is formally verified via sequences-of-games [14] and automatically verified via the professional security verification tool ProVerif [15]. Mutual authentication and key exchange processes of the proposed protocol and its PFS-augmented version are proven to be secure, and possess robust reachability and secrecy, mutual authenticity, and high resilience against multiple attacks such as replay and MITM attacks.
5. The proposed scheme has also been evaluated physically by a prototype system using Avnet Ultra96-V2 boards to emulate the IoT endpoints equipped with the required hardware primitive. The proposed protocol and its PFS augmented version only require three handshakes each to achieve the desired high security and computational efficiency. Security features, computational complexity, communication and storage costs are also analyzed and compared with related existing works.

The rest of the paper is organized as follows. Sec. 2 introduces the related works; Sec. 3 described the system model, threat models and assumptions. Sec. 4 presented the proposed protocol design, which includes an enrollment and update phase, and a mutual authentication and key exchange phase. In addition, a PFS protocol enhancement is proposed in Sec. 5. The security of the proposed protocol is analyzed in Sec. 6 while its implementation and comparison against related existing protocols are presented in Sec. 7. Finally, Sec. 8 concludes the paper.

2 RELATED WORKS

In this section, two pitfalls in using traditional authentication protocols for securing P2P IoT applications are discussed. In the same light, the limitations of existing PUF-based device authentication schemes in IoT setting are also identified.

2.1 P2P IoT

Two necessary steps and the potential issues of adopting typical authentication protocols for securing P2P IoT con-
ections are identified as follows. Firstly, the authenticity of the two IoT endpoints has to be assured before a secure direct connection between them can be established. A common approach is to involve a trusted server to authenticate the two IoT devices individually. Once both devices are authenticated, the communication is then handed over to the two devices without involving the server any more. This approach requires two mutual authentications to be performed independently between the server and the devices. The extra authentication through a proxy server incurs a transmission bandwidth overhead and authentication delay. Besides, the establishment and relay of shared secrets through the server for each authentication request also increase the vulnerability of MITM and chosen-plaintext or chosen-ciphertext attacks. Secondly, end-to-end encryption is required to ensure secure data transmission between the two interlocutors. Either a symmetric key or a public key cryptographic (PKC) algorithm can be utilized for data encryption. Both cases require a secret key to be safely stored in both endpoints. If the secret keys are stored in the non-volatile memory (NVM) of inexpensive endpoint devices with a limited room for sophisticated key protection, they can be relatively easily extracted by powerful commercial data extraction tools like Graykey [16] of Grayshift and UFED [17] of Cellebrite.

The above-mentioned issues of P2P IoT communication can be avoided if the two IoT endpoints can directly authenticate each other without the need to store the secret key persistently and locally on the endpoint devices to achieve end-to-end encryption.

### 2.2 Existing PUF-based Authentication Schemes

PUF as “device fingerprint” has been widely used for individual device authentication [2], [5], [6]. A typical PUF-based authentication process is shown in Fig. 1. An enrollment phase has to be carried out securely in a controlled environment once before the PUF can be deployed in the field for device authentication. During the enrollment phase, an adequate number of CRPs of the PUF has to be collected and stored in the verifier’s database. In the authentication phase, the verifier randomly selects a CRP from its database and then transmits the challenge to the prover. The prover obtains a response by applying this challenge to its PUF and sends the response back to the verifier. By comparing the received PUF response with the enrolled one, the verifier can authenticate the prover’s device.

As shown in Fig. 1, a typical PUF-based authentication scheme requires the verifier to safely store a CRP database of the prover’s PUF. This is easier to achieve if the verifier is a powerful server. If direct authentication of P2P IoT devices is to be achieved without a proxy server as discussed in Sec. 2.1, this scheme is infeasible as the verifier, an IoT endpoint, has limited resources to ensure the secure storage of the CRPs. As a result, PUF-based authentication schemes without the CRP storage in both endpoint devices of the verifier and the prover are desperately needed in P2P IoT applications.

Several relevant solutions have been proposed in the literature [13], [18], [19], [20]. They are briefly reviewed as follows.

The protocol of [13] combines the PUF with a certificateless identity based encryption. It requires a verifier node to act as the proxy to authenticate the two IoT endpoints. It also requires the presence of a Security Association Provider (SAP) during the authentication phase to manage the authentication helper data. Besides, the protocol requires the verifier to store a secret hash key in an NVM. This requirement has completely defeated the fundamental tenet of using PUF to avoid keeping the secret in a local memory.

On the other hand, the Babel-chain PUF-based mutual authentication protocol (BC-PHEMAP) [18] applies the PUF response iteratively to the PUF to create PUF chains for authentication. Since multiple PUF operations (8 in the text example) are required and each PUF operation has to be 100% reliable in order to preserve the enrolled chain result, this scheme is not practical due to the inevitable unreliability of PUF responses in the field. What is more, the BC-PHEMAP scheme reveals raw PUF responses during protocol execution. The scheme requires strong PUF to ensure the chain or its subset is never reused to pair different devices but this also provides an opportunity for the adversary to collect a large number of used raw responses for modeling attack.

The scheme in [19] also requires a trusted third-party server during the mutual authentication of two IoT endpoints. The server is responsible for providing authentication messages and generating session keys. The protocol trades message exchange time (delay) for zero-storage on the device side and reduced storage requirements on the server side. Two rounds of mutual authentication between the server and the device and one round of mutual authentication between devices are required for a complete protocol run. Similar to [18], it does not take into account practical PUF reliability issues.

The work reported in [20] relies on the combination of PUF with a PKC on elliptic curve to achieve mutual authentication between IoT nodes without the need for a local CRP storage. Each IoT node will obtain its public and private key pair after an enrollment phase with the server in a secure environment. The public keys are stored in the device’s memory while the private keys are recovered with the help of the PUF during the authentication phase. Sixteen rounds of elliptic curve point multiplications are required to complete the protocol runs, which increases the computational cost of the protocol significantly.

---

**Fig. 1.** A typical PUF-based device authentication scheme. Here the PUF is a strong PUF with an exponential number of CRPs. Each CRP is used only once for secure authentication.
In short, existing PUF-based authentication schemes, either with or without requiring a CRP storage in the verifier, cannot be used to instantiate a low-cost, key-less, directly authenticated and directly connected P2P IoT network.

3 Threat Models and Assumptions

Fig. 2 shows the system model. The objective of the system is to establish direct mutual authentication and direct communications between IoT peer nodes by utilizing the on-the-fly key generation property of PUF while avoiding the shortfalls of existing PUF-based authentication schemes. The proposed protocol consists of two phases, i.e., an enrollment and update phase, and a mutual authentication and key exchange phase.

![System model](image)

Fig. 2. System model. The server is only present in the enrollment and update phase to create/update the authentication mask. Once this is completed, the IoT endpoints can freely and directly authenticate and exchange data among themselves.

Dolev-Yao (DY) model [21] and Canetti-Krawczyk (CK) adversary model [22] are two popular threat models. They are briefly introduced as follows:

DY model assumes that the protocol is run in the presence of a probabilistic polynomial-time (PPT) adversary, who fully controls the network and is able to eavesdrop, intercept, modify, delete, inject, and replay any messages transmitted between two honest principals, except that he/she cannot perform any cryptanalysis. The attacker can only decrypt/sign a message if he/she has the correct key. The attacker can only form new messages from keys and messages in his/her possession [23]. Besides, it is also assumed that the attacker can initiate an unbound number of authentication requests.

A CK adversary is more powerful than a DY adversary and is widely adopted in the authenticated key exchange protocols. In addition to the basic adversaries’ capabilities like actively manipulating the transmitted messages over the communication channel, the attacker is assumed to be able to obtain any secrets stored in the honest parties’ memory via explicit attacks. Besides, the adversary can compromise the established session keys and session states. The adversary can also use oracle queries to interact with the entities [24]. According to the type of information leaked, security attacks in the CK-adversary model can be categorized into three types: session-state reveal, session-key query and party corruption.

In this paper, both the DY model and the CK-adversary model are considered for security analysis. For a formal security analysis, we adopt the more powerful CK-adversary model with the following security assumptions. We assume that the non-transmitted ephemeral (erased or overridden upon used) intermediate variables involved in session state computation during the brief device authentication process are secure. In addition, we assume that the server can afford expensive anti-tamper storage to secure any secret and is powerful enough to perform complex computations. By contrast, the end devices have limited storage capacity and are incapable of protecting the secrets stored in their NVM. Hence, they are vulnerable to session-key query and party corruption attacks under the CK-adversary model. We also conduct an automatic security analysis of the proposed protocol using ProVerif, which is operated under the DY model.

In addition, some assumptions about the properties of the security primitives are also discussed below:

Firstly, the PUF is assumed to be secure against micro-probing and physical attacks. This is because the key is not permanently present in digital form like the NVM key. It is derived from within the hardware only upon queried by a challenge. The CRP generation mechanism cannot be replicated by an invasive attack or probing into the internal cell structure of the PUF. Doing so will not only destroy the PUF functionality, but also leave a permanent tamper evidence that can be detected easily. As a natural byproduct of the semiconductor manufacturing process variations, the CRPs of the PUF are assumed to have a very high entropy. Although its generation mechanism is inexpensive, such physical disorder is very hard to be fully characterized. In short, it is practically infeasible for the adversary to physically clone a PUF, nor guess an unseen response of a PUF by a brute-force approach if the bit length of the PUF response is sufficiently long. Besides, we assume that the PUF can also be made resilient against side-channel attacks using several countermeasures proposed in the literature [25], [26].

Decisional Uniqueness Problem (DUP) Assumption [13]: Given a challenge $C$, an arbitrary PUF instance $PUF_{\text{adv}}$ and an $l$-bit string $z \in \{0,1\}^l$, it is hard to decide if $z = PUF_t(C)$ or a random $l$-bit string, where $PUF_t$ is the target PUF instance to corrupt. For any PPT algorithm $A$, there exists a negligible $\epsilon_{\text{puf}}$ such that

$$\left| Pr[A(C, PUF_{\text{adv}}, z \leftarrow \{0,1\}^l) = 1] - Pr[A(C, PUF_{\text{adv}}, z = PUF_t(C)) = 1] \right| \leq \epsilon_{\text{puf}}$$

Secondly, the hash function used is assumed to be a one-way hash function that possesses perfect pre-image and collision resistances. Pre-image resistance refers to the difficulty of finding any correct input given the output of the hash function, whereas collision resistance refers to the computational complexity of finding two different inputs that hash to the same output value. It should be noted that collision resistance also implies second pre-image resistance. The latter is a property of a hash function to ensure that given an input, it is difficult to find another input that has the same hash output.

Entropy Smoothing (ES) of Hash Functions [14]: Let $H : \{H_k\}_{k \in K}$ denote a family of keyed hash functions, where each $H_k$ is a function that maps $G$, a finite cyclic group of order $n$, to $\{0,1\}^l$. Let $A$ be a PPT algorithm whose inputs are an element of $K$ and an element of $\{0,1\}^l$ and whose
output is a bit. Then, for hash functions that fulfill the above assumptions, there exists a negligible $\epsilon_{es}$ such that

$$\Pr[k \overset{R}{\leftarrow} K, s \overset{R}{\leftarrow} G : \mathcal{A}(k, H_k(s)) = 1] - \Pr[k \overset{R}{\leftarrow} K, h \overset{R}{\leftarrow} \{0, 1\}^l : \mathcal{A}(k, h) = 1] \leq \epsilon_{es} \tag{2}$$

Thirdly, a trapdoor function is required to achieve PFS of the secret key generated in the proposed protocol. Since elliptic curve cryptography is used in related papers [13], [20] for PUF-based mutual authentication and key exchange protocol, elliptic curve cryptography is assumed for ease of comparison with these protocols.

**Elliptic Curve Diffie-Hellman (ECDH) Assumption** [27]: Let $E$ be an elliptic curve defined over a finite field $\mathbb{F}_p$, where $p$ is a prime. Further, let $G$ denote a finite cyclic group of order $n$ and $P \in E(\mathbb{F}_p)$ is a generator point of $G$. The ECDH problem is the problem of finding $S = [ab \mod(n)]P$ given $(E, n, P, aP, bP)$, where $P$, $aP$, $bP \in G$. As recommended by [28], “Transitioning to ECDH key exchange with appropriate parameters avoids all known feasible cryptanalytic attacks”. It is therefore reasonable to assume that with proper choices of the curve $E$, solving the ECDH problem is computationally hard. This intractability assumption can be formally expressed as [20]:

$$\Pr[\mathcal{A}(E, n, P, aP, bP)] = [ab \mod(n)]P \leq \epsilon_{edh} \tag{3}$$

where $\epsilon_{edh}$ is a negligibly small value.

## 4 PROPOSED PUF-BASED P2P AUTHENTICATION SCHEME

The notations used in our proposed protocol are described in Table 1.

### 4.1 Enrollment and Update Phase

A general enrollment is assumed to be conducted in the design house prior to device deployment, where the CRPs of each PUF-embedded chip are collected after fabrication. After which, the direct CRP access path is permanently removed. Once the customer identity is authenticated, the CRPs will be transferred to the customer’s local server for the intended applications. Note that the CRP collection is assumed to be conducted in a secure environment and the transfer of CRPs is also securely completed with the help of various supply chain authentication protocols.

The main idea of the proposed work is to utilize the One-Time Pad (OTP) to generate a key mask for a dedicated device pair. The mask can be publicly stored in the device without compromising the security. Its purpose is to enable secure retrieval of the secrets during the authentication process. As shown in Fig. 3, for each intended pair of edge devices, $A$ and $B$, the server randomly selects a challenge $C_a$ for $A$ and a challenge $C_b$ for $B$ from their respective challenge sets, $C_A$ and $C_B$. The corresponding responses, $R_a$ and $R_b$, are also retrieved from the server’s CRP database. To enable $R_a$ and $R_b$ to be recovered from the noisy responses reproduced by the respective device PUFs in the field, the server also applies ECC [29] to $R_a$ and $R_b$ to generate the corresponding helper data, $h_a$ and $h_b$. To ensure the freshness of the session, the server generates two random nonces $m$ and $n$ for device $A$ and device $B$. The update request $<\text{ID}_{\text{self}}, \text{challenge}_{\text{self}}, \text{ID}_{\text{target}}, \text{challenge}_{\text{target}}, \text{helper}_{\text{data}}, \text{nonce}>$ is then sent individually to each designated recipient’s devices, where subscripts ‘self’ and ‘target’ refer to the designated device and its pairing device, respectively. An honest device $A$ (or $B$) can generate a valid $A_a$ (or $A_b$) to prove its identity to the server. Once the device is authenticated by the server, the server calculates a pair of random intermediate secrets, $P_1$ and $P_2$, from the response $R$ of each designated device and the identity of its pairing device. A lightweight cryptographic hash function $H$ is used to boost the entropy of PUF responses and prevent the plain responses from being intercepted. Then, an OTP mask $\phi$ is generated by the XOR operation. $\phi$ needs not to be kept private. It can be stored in the device memory. Due to the OTP, the knowledge of $\phi$ does not help to reveal $P_1$ and $P_2$. Hence, both $P_1$ and $P_2$ can serve as the keys to decrypt a pair of puzzles ($Q_a$ and $Q_b$ in Fig. 4) designated for a device with the help of its PUF. Specifically, for device $A$, $P_{1a}$ is first calculated by hashing the concatenated $R_a$ and $ID_B$, $P_{2a}$ is then obtained by hashing $P_{1a}$, $P_{1b}$ and $P_{2b}$ are similarly obtained from hashing the concatenated $R_b$ and $ID_A$ consecutively. These private data are concealed in $\phi_1$.

### Table 1

| Notation | Description |
|----------|-------------|
| PUFA($C$) (PUFB($C$)) | Apply the challenge $C$ to the PUF of device $A$ ($B$) |
| $H(X)$ | Hash of $X$ |
| $a \oplus b$ | Exclusive OR |
| $a, b$ | Concatenation |
| $x \overset{R}{\leftarrow} X$ | $x$ is randomly chosen from the set $X$ |
| $\phi$ | A key mask |
| $h = \text{ECC}.\text{gen}(X)$ | Error correction code (ECC) generation phase. It generates the helper data $h$ for the recovery of $X$ |
| $X = \text{ECC}.\text{rep}(X, h)$ | ECC reproduction phase. It corrects the noisy $X$ with the helper data $h$ |
| $M = \text{CCM}.\text{enc}(f, k, m)$ | CCM encryption of plaintext $m$ with $k$ as the key and $f$ as the underlying block cipher, which could either be a pseudo-random function (PRF) or a pseudo-random permutation (PRP). The output $M$ includes the ciphertext and a tag |
| $V = \text{CCM}.\text{dec}(f, k, M)$ | CCM decryption of $M$ with $k$ as the key and $f$ as the underlying block cipher. If key $k$ is correct, $V = m$, where $m$ is the plaintext. Otherwise, an error is returned in $V$. |
| $\{0,1\}^l$ | The set of bit strings of length $l$ |
and $\phi_2$ by the OTP construction, $P_{1a} \oplus P_{1b}$ and $P_{2a} \oplus P_{2b}$, respectively.

Note that an OTP can achieve perfect secrecy if the length of the random key is at least as long as the plaintext. Besides, the random key must be kept completely secret and never be reused by different device pairs. All these requirements can be fulfilled by the proposed method. Use $\phi_1 = P_{1a} \oplus P_{1b}$ for illustration. $P_{1a}$ and $P_{1b}$ are the outputs of the same hash function, which makes the length of both parameters the same. The “completely secret” requirement is also met by our proposed scheme as $P_{1a}$ and $P_{1b}$ are defined by the PUF responses. They can only be generated by the device when needed and are never stored or transmitted in plaintext. As for the one-time use policy of the OTP, even if the same challenge is picked for a device $A$ to pair with different devices, e.g., $B$ and $C$, the generated $P_{1a}$ is different for different pairing devices as the hash function used for the generation of $P_{1a}$ is dependent on both $R_a$ and its pairing device’s unique identity, i.e., ID$_B$ or ID$_C$. Even though an authorized device can retrieve the corresponding $P_{1a}$ or $P_{2a}$ to the allocated challenge for any of its interlocutor, it still cannot impersonate as its interlocutor to communicate with other devices, neither does the attacker.

A device should only accept a fresh authentication mask from an authorized server that possesses the correct CRPs of the device’s PUF. To ensure this, the server packs the mask $\phi_2$ with the fresh $x$ sent by $A$. The message package is encrypted using counter with cipher block chaining message authentication code (CCM) mode [30] with $H(R_a)$ as the key and AES as the underlying PRP. CCM combines cipher block chaining message authentication code (CBC-MAC) and counter (CTR) mode encryption to provide authenticated encryption, which assures both the confidentiality and authenticity of data. Besides, CCM has smaller code size as the decryption block of the underlying block cipher is not required [30]. To decrypt the received message, $A$ applies $C_a$ to its PUF to recover the key $H(R_a)$, and performs a decryption upon receiving the message. If the message can be successfully decrypted, it means that the message sender possesses the correct $R_a$. Therefore, device $A$ can believe that the enrolled or updated message (ID$_A$, $C_a$, ID$_B$, $C_b$, $h_a$, $\phi_2$) is indeed from a trusted local server. If the decrypted $x$ is the same as the fresh $x$ that it sent, device $A$ can confirm the freshness of the updated message, thus the replay of a compromised $X_a$ is prevented. Once the authenticity and freshness are confirmed, $A$ stores the information in its local database $L_A$. Similar process is carried out by device $B$. The same process can be performed whenever the authentication mask on the device’s side needs to be updated.

### 4.2 Mutual Authentication and Key Exchange Phase

If an IoT end-device $A$ wants to communicate with another end-device $B$, it will send a request message MSG$_1$ containing the identities (ID$_A$ and ID$_B$), the challenges used for the current session ($C_a$ and $C_b$), and a nonce ($T_a$), to device $B$, as shown in Fig. 4. $B$ checks the correctness of the message receiver and the existence of requestor and pairing challenges in its local database $L_B$. If yes, it fetches the enrolled challenge $C_b$, helper data $h_b$, and key mask $\phi_1$ corresponding to (ID$_A$, $C_a$) from $L_B$. $B$ applies the challenge $C_b$ to its embedded PUF to generate a reliable PUF response $R_b$ with the helper data $h_b$. Using the key mask $\phi_1$, $B$ is able to recover the paired $P_{1b}$ of $A$ by XORing $\phi_1$ with $P_{1b}$. The latter is the hash of $R_b$ and ID$_A$. $B$ then generates a random nonce $n_b$. Using the recovered $P_{1b}$ as the encryption key, a puzzle $Q_b$ is created by device $B$ for device $A$ by encrypting the concatenated message of $n_b$ and the received $T_a$ using CCM encryption. Next, $B$ generates another nonce $T_b$ and then sends MSG$_2$ = $\langle Q_b, T_b \rangle$ to $A$ for authentication. Upon receiving MSG$_2$, $A$ calculates the decryption key $P_{1a}$ from ID$_B$ and its clean PUF response $R_a$ reproduced with the help of $h_a$, and then uses it to decrypt the received $Q_b$.

![Fig. 3. Enrollment and update phase. Each entity has a list stored in their corresponding database, denoted as $L_A$, $L_S$ and $L_B$. A pair of devices can have multiple records by selecting different challenges. The number of updates is bounded by $|C_A| \times |C_B|$ for each pair, where $|S|$ denotes the cardinality of $S$. The regenerated response is possibly noisy, which is indicated by a hat on the response. Similar notations are used for Fig. 4 and Fig. 5.](image-url)
If decryption is successful and the decrypted $\tilde{T}_a$ is equal to $T_a$, $A$ accepts $B$ as the authenticated interlocutor.

By reversing the roles of the prover and the verifier, a similar flow can be reciprocated by $B$ to authenticate $A$. Device $A$ retrieves $\tilde{P}_b$ with the help of $\tilde{Q}_b$. Using $\tilde{P}_b$, as a key, $A$ encrypts a random nonce $n_b$ and the received $T_b$ into a puzzle $Q_b$ for device $B$ to solve. Upon receiving $MSG_3 = < Q_b >$, $B$ can authenticate $A$ by following the same procedure. If $Q_b$ can be successfully decrypted by $P_b$ retrieved from hashing $\tilde{P}_b$, $B$ can further check the freshness of the recovered $\tilde{T}_b$. If both are valid, $B$ accepts $A$ as its authenticated interlocutor.

Upon the successful mutual authentication, both parties can calculate $sk = H(n_a, n_b) = H(\tilde{n}_a, n_b) = H(n_a, \tilde{n}_b)$ as the session key and use it directly for encrypted communication. Note that the session secrets, $n_a$ and $n_b$, are independent and ephemeral. Therefore, the calculated session key $sk$ is also fresh and independent. Besides, $\tilde{Q}_b$ and $Q_b$ do not have to be stored by both devices. With the current setting, the protocol can also execute successfully if $B$ initiates the protocol first. The only difference is that $B$ will authenticate $A$ first before $A$ authenticates $B$.

5 Proposed PFS-enhanced PUF-based P2P Authentication Scheme

The proposed protocol shown in Sec. 4 is efficient but cannot achieve PFS if the long-term secrets of the entities are compromised (See Sec. 6). A small tweak on the protocol can be made to achieve PFS. The idea is to use a trapdoor function. The choice of a trapdoor function can be determined by the application. In this section, an augmented mutual authentication and key exchange protocol based on ECDH problem is introduced.

5.1 Enrollment Phase

To achieve PFS, the enrollment phase of the proposed protocol requires several additional registration steps. The server chooses a safe elliptic curve $E$ over a finite field $\mathcal{F}_p$, where $p$ is a prime, and then selects $P \in E(\mathcal{F}_p)$ of order $n$ as a generator. The local server publishes $\{ E, n, P \}$ as public parameters, which are available to all entities including the adversary. All the other procedures remain the same.

5.2 Mutual Authentication and Key Exchange Phase

For the augmented protocol, the plaintexts of the puzzles are slightly different. Device $B$ encrypts $(n_a P, T_a)$ instead of $(n_b, T_b)$; Similarly, device $A$ encrypts $(n_b P, T_b)$ to create the puzzle $Q_b$, where $n_a P$ and $n_b P$ are point multiplications over the elliptic curve. In this way, the leakage of long-term secrets, i.e., $R_a$, $R_b$ (or $P_{1a}$, $P_{1b}$), will only provide the adversary with $n_a P$ and $n_b P$. Thanks to the ECDH problem, the adversary is not able to calculate $sk = n_a n_b P$ with those information. The PFS-augmented mutual authentication and key exchange protocol is delineated in Fig. 5.

6 Security Analysis

The proposed protocols are analyzed by a game-based formal security analysis under the CK-adversary model in Sec. 6.1 and an automatic security analysis under the DY model using the security verification tool ProVerif [15] in Sec. 6.2.

6.1 Game-based Security Analysis

This section presents a rigorous formal security analysis of the proposed protocols using the sequence-of-game approach [14] under the CK-adversary model with the assumptions delineated in Sec. 3. Considering concurrent executions, we use $A_i$ and $B_j$ to denote the $i^{th}$ session of “Device $A$” and the $j^{th}$ session of “Device $B$”, respectively. The instances $A_i$ and $B_j$ are partners if they have the same session identifier, play different roles (i.e., initiator and responder) and agree on the same session key. Let $U$

2. https://safecurves.cr.yp.to/
be an instance, which can be either $A_i$ or $B_j$. Based on the CK-adversary model in Sec. 3, the adversary can issue the queries defined below:

1) \textbf{Send}(MSG, U) query: Send queries model the active attacks such as modify/delete/inject/replay the transmitted messages [31]. This query allows $A$ to send an arbitrary message MSG to the instance $U$. \textbf{Send}(Start, U) initializes the protocol. This query is answered with the corresponding messages that $U$ outputs according to the proposed protocol.

2) \textbf{Execute}(A_i, B_j) query: This query simulates the passive eavesdropping attack and is answered with all the transmitted messages during an honest protocol execution between $A_i$ and $B_j$.

3) \textbf{Corrupt}(U) query: This query simulates the physical attacks on the entity $U$ and is answered with the corrupted entity’s stored parameters.

4) \textbf{SKReveal}(U) query: This query allows $A$ to learn the session key held by $U$.

5) \textbf{Test}(U) query: This query can only be issued to a fresh session if the session key has been accepted by $U$. \textbf{Fresh}$ implies that SKReveal(U) and Corrupt(U) have not been requested by both this session and its partner session (if any). If a partner session exists, it further requires that this partner session is not also a test session [32]. The \textbf{Test} query is answered as follows: $U$ flips a random bit $b$. If $b = 1$, return the session key; otherwise, return a random key of the same size.

Game-based security proof is a proof constructed by a sequence of games (Game $i$, $i = 0, 1, \cdots, n$) to be played between a hypothetical challenger (a probabilistic “checking” algorithm) and an efficient adversary (a PPT algorithm $A$). Game 0 is defined as the real attack game. In each Game $i$, we additionally define the winning event $S_i$, which denotes $A$ can successfully: 1) launch an impersonation attack, i.e., $A$ is able to make any honest session accept an authenticator that has not actually been sent by the expected partner session; or 2) distinguish a session key from a random key, i.e., $b = b'$, where $b$ is involved in the $\textbf{Test}(U)$ query while $b'$ is the guessed output of $A$.

Let $Pr[\cdot]$ denotes the probability of occurrence of the underlying event. For a secure session key exchange proof, the objective is to show that the aggregate difference between $Pr[S_i]$ and $Pr[S_{i+1}]$, $\forall i = 0, 1, \cdots, n – 1$, is negligible and $Pr[S_n]$ is negligibly close to certain “target probability”. Since the number of games ($n$) is finite and small, $Pr[S_0]$ for the real attack can be proved to be negligibly close to the “target probability” as well based on the triangular inequality.

The following lemma and theorem are useful in establishing the advantage of a PPT adversary in the games.

\textbf{Difference Lemma} [14]: Let $A, B, F$ be the events defined in some probability distribution. If $A$ and $B$ proceed identically unless a “failure event” $F$ occurs, i.e., $A \land \neg F \leftrightarrow B \land \neg F$, then:

$$|Pr[A] - Pr[B]| \leq Pr[F]$$  \hfill (4)

\textbf{CCM-mode Theorem using PRP} [30]: Suppose $F$ is a secure PRP from $\{0,1\}^l$ to $\{0,1\}^b$. Let $q_F$ be the number of forgery attempts, $n_E$ be the total number of block cipher calls in response to all encryption queries and $n_F$ be the total number of block cipher calls to decrypt and check the validity of an attempt ciphertext $c^*$. Then, the advantage of a PPT adversary $A$ against the authenticity of CCM is:

$$Adv_{\text{CCM,F}}^{auth}(A) \leq \epsilon_{PRP} + \frac{q_F}{2^b} + \frac{(n_E + n_F)^2}{2^{2b}}$$  \hfill (5)

where $l_z$ is the bit length of the tag and $l_b$ denotes the block size. For any secure PRP, $\epsilon_{PRP}$ is negligible [14]. Therefore, CCM provides up to $2^{b+1/2}$ block cipher calls of authenticity.

\textbf{Theorem (SK-secure)}: Let $A$ be a PPT adversary, with $q_h$ Hash queries, $q_s$ Send queries and $q_e$ Execute queries. Furthermore, let the proposed MA and KE protocol shown

3. Advantage is defined in [33] to capture the capability of the adversary in distinguishing the cipher text from random text.
in Fig. 4 and Fig. 5 be denoted as $\pi_1$ and $\pi_2$, respectively. Let $l$ be the bit length of PUF challenge/response, hash output and CCM block cipher. Then, the advantage of $A$ on $\pi_1$’s session key is:

$$Adv^{SS}_{\pi_1}(A) \leq 6\epsilon_{ca} + 2\epsilon_{pf} + \epsilon_{prp} + \frac{10q_a + 37(2q_a + q_e)^2 + q_h^2}{2^l}$$

(6)

and the advantage of $A$ on $\pi_2$’s session key is:

$$Adv^{SS}_{\pi_2}(A) \leq 4\epsilon_{ca} + 2\epsilon_{pf} + \epsilon_{prp} + \frac{10q_a + 65(2q_a + q_e)^2 + q_h^2}{2^l}$$

(7)

Since $\epsilon_{ca}$, $\epsilon_{pf}$ and $\epsilon_{prp}$ are negligible, $Adv^{SS}_{\pi_1}(A)$ and $Adv^{SS}_{\pi_2}(A)$ are negligible up to $2^{l/2}$ calls of all queries. Therefore, $l$ can be made sufficiently long for the security of the proposed protocol. Update session can be called to change the underlying authentication keys ($P_{ia}$, $P_{ib}$) after certain number ($\approx 2^{l/2}$) of total queries to ensure the attack probability is negligible. Alternatively, other secure PRF or PRP with longer ($\geq 256$ bits) block length can be used in place of AES.

**Proof:** This section incrementally defined a sequence of games from Game 0 to Game 6 to formally prove the session key security of the proposed protocols.

**Game 0:** This is defined as the real attack game against our protocol by a PPT adversary $A$. By definition, the adversary’s advantage of breaking the semantic security (SS) of the proposed authenticated key exchange scheme is:

$$Adv^{SS}_{\pi_1} = |Pr[S_0] - 1/2|$$

(8)

**Game 1:** In this game, we simulate the hash oracles and PUF oracles under the random oracle model. Besides, we also simulate all the instances, as the real players would do, for the Send, Execute, Corrupt, SKReveal and Test queries as shown in Table 2.

1) **Hash oracle query:** Hash function is modeled as a random oracle ($RO_H$), i.e., it is modeled as a truly random function. The challenger and the adversary have blackbox access to the random oracle. A list $L_H$ is maintained to store the $query, answer$ of $H$. For each new hash query, $L_H$ will be checked first to find out if the query has been recorded before. If yes, the corresponding answer will be replied. Otherwise, a randomly chosen value is returned, and the new random is added to $L_H$.

2) **PUF oracle query:** A list $L_{PUF}$ similar to $L_H$ is also maintained for the PUF oracle. The only difference is the PUF oracle cannot be directly accessed by the adversary $A$. Note that the simulation of the fuzzy extractor is no longer necessary as the output of the PUF oracle is assumed to be stable.

From this simulation, it is easy to observe that Game 1 proceeds indistinguishably with the real attack Game 0. The transition is to compute the hash outputs and the PUF responses by uniformly choosing them at random, rather than by hash or the real PUF instances. With entropy smoothing assumption of hash function and the DUP of PUF, we claim that

$$|Pr[S_1] - Pr[S_0]| \leq 6\epsilon_{ca} + 2\epsilon_{pf}$$

(9)

**Game 2:** This game is defined identically as Game 1 except that collisions are avoided. Collisions could occur if two honest sessions pick the same nonce or an honest responder session picks a nonce that has already been submitted by the adversary to an initiator session [34]. If at most $q$ queries are made to a random function of $l$ bits, the probability of collision is within $q^2/2^l$ according to the birthday paradox bound. Since there are at most $(q_a + q_e)$ pairs of nonces chosen uniformly at random in all the calls to SessionA1 and SessionB1, the collision happens with a probability bounded by $(q_a + q_e)^2/2^{l+1}$. Besides, $A$ can submit at most $q_a$ pairs to the initiator sessions, so the probability that a responder session randomly chooses one of these pairs is at most $q_a/2^l$. In this game, we also abort collisions that come from the output of hash queries. Game 2 and Game 1 are identical unless the “abort” event happens due to the above collisions. According to the **Difference Lemma**, we have

$$|Pr[S_2] - Pr[S_1]| \leq \frac{(q_a + q_e)^2}{2^{l+1}} + \frac{q_a^2}{2^l} + \frac{q_a^2}{2^{l+1}}$$

(10)

In all subsequent games, we are now sure that unique nonces are chosen at each honest session.

**Game 3:** This game is defined by excluding the event that $A$ is lucky in obtaining the authentication keys $\tilde{P}_{ia}$ ($\tilde{P}_{ib}$) and $\tilde{P}_{ib}$ ($\tilde{P}_{ib}$). A correct guess will allow $A$ to impersonate as an honest device by forging a valid MSG2 or MSG3 for authentication, or decrypt $Q_a$ or $Q_b$ to obtain $n_a$ or $n_b$ for $sk$ calculation. When this happens, we set a bad flag $bad_{P_{ia}}$ or $bad_{P_{ib}}$, and abort the session. Since $\tilde{P}_{ia} = P_{ia} = P_{ib} \oplus P_{2a}$, $\tilde{P}_{ib} = P_{ib} = P_{ib} \oplus P_{2a}$, $\tilde{P}_{2a} = H(P_{ia})$ and $P_{2b} = H(P_{ib})$, $A$ can obtain $\tilde{P}_{ib}$ ($\tilde{P}_{ib}$) and $P_{2b}$ ($P_{ib}$) by the following ways:

1) $A$ can get $\phi_1$ by Corrupt(B) and then tries to obtain $P_{ib}$.

Since $\tilde{P}_{ib} = H(R_b, ID_A)$, $A$ can obtain $\tilde{P}_{ib}$ by randomly guessing $R_b$ or directly guessing $P_{ib}$. The probability is not more than $\frac{2q_b}{2^l}$.

2) **Corrupt(B) does not happen.** $A$ directly guesses $\tilde{P}_{ib}$. The probability of a correct guess is not more than $\frac{q_b}{2^l}$.

3) $A$ can get $\phi_2$ by **Corrupt(A)**, and then tries to obtain $P_{2a}$ directly by random guess, or $A$ can guess $R_a$ or $P_{ia}$ directly and then calculates $P_{2a} = H(H(R_{ia}, ID_B))$ or $P_{2a} = H(P_{ia})$, respectively. The probability is not more than $\frac{3q_b}{2^l}$.

4) **Corrupt(A) does not happen.** $A$ directly guesses $\tilde{P}_{ib}$, or directly guesses $R_b$ or $P_{ib}$ then calculates $P_{2b} = H(H(R_{ib}, ID_A))$ or $P_{2b} = H(P_{ib})$, respectively. The probability of correctly guessing $\tilde{P}_{ib}$ is not more than $\frac{3q_b}{2^l}$.

Since the probabilities of a successful guess on $P_{ia}$ and on $R_b$ or $P_{ib}$ have been repeatedly accounted in 2) and 3), and 1) and 4), respectively, the repeated probability, i.e., $\frac{3q_b}{2^l}$, has to be excluded from the sum of these probabilities. Therefore, we have

$$|Pr[S_3] - Pr[S_2]| \leq \frac{6q_b}{2^l}$$

(11)

**Game 4:** We now transfer Game 3 into Game 4, which aborts the executions wherein the authenticity of the CCM


### Table 2

Game 1 (changes highlighted in gray) and Game 2 (changes highlighted in frames) for \( \pi_1 \). \( A \) has access to oracles

\[ O := \{ \text{Send, Execute, RO}_H, \text{Corrupt}, \text{SKReveal}, \text{Test} \}.

| Games 1-2: | SessionB1(MSG1, B1): |
|-----------|------------------------|
| \( b \xleftarrow{R} \{0, 1\} \) | \( (ID_A, C_A, ID_B, C_B, T_u) \leftarrow \text{MSG}_1 \) |
| \( b' \leftarrow A^O(A_1, B_1) \) | if \( ID_B \neq ID_B' \): abort \|
| // \( A \) can obtain all the public information of \( A_i \) and \( B_j \) for \( sid \in S \): | if \( (ID_A, C_A, C_B') \notin L_B: \) abort |
| if Impersonation(sid) is true: return 1 \|
| if Fresh(sid) is false: \( b' \leftarrow 0 \) | \( R_b := RO_{PUFB}(C_B) \)
| return \( b' = b \) | \( P_{1b} := RO_H(R_b, ID_B) \) |

| SessionA1(Start, A1) | SessionB2(MSG2, B2): |
|----------------------|-----------------------|
| \( T_a \xleftarrow{R} \{0, 1\} \) \|
| if \( T_a \in N:\ abort \|
| \( N := N \cup \{T_a\} \) \|
| return MSG1 : \( (ID_A, C_A, ID_B, C_B, T_u) \) \|

| SessionA2(MSG2, A1) | SessionB2(MSG2, B2): |
|----------------------|-----------------------|
| \( (Q_b, T_u) \leftarrow \text{MSG}_2 \) \|
| \( \text{Recv} := \text{Recv} \cup \{T_b\} \) \|
| \( sid := \langle T_a, T_b \rangle \) \|
| \( \text{init}(sid) := A_1 \) // initiator of the session \|
| \( R_a := RO_{PUFB}(C_A) \) \|
| \( P_{1a} := RO_H(R_a, ID_A) \) \|
| \( V_a := CCM.dec(AES, P_{1a}, Q_a) \) \|
| if \( V_a \neq \varnothing\): // \( \varnothing \) = error \|
| \( (n_a, T_a) \leftarrow V_a \) \|
| if \( T_a = T_u\): \|
| \( \text{statusA}[sid] := \text{accept} \) // Authenticated session \|
| \( P_{2a} := RO_H(P_{1a}) \) \|
| \( P_{2b} := \phi_2 \oplus P_{2a} \) \|
| \( n_b \xleftarrow{R} \{0, 1\} \) \|
| if \( n_b \in N:\ abort \|
| \( N := N \cup \{n_b\} \) \|
| \( Q_b := CCM.enc(AES, P_{2b}, (n_b, T_b)) \) \|
| \( \text{sk} := RO_H(n_a, n_b) \) \|
| \( \text{sk}eY[sid] := \text{sk} \) // session key \|
| \( l[sid] := Q_b \) // messages outputted by the initiator \|
| return MSG3 : \( Q_b \) \|

| Send(MSG, U): | SKReveal(sid, U): |
|-------------|----------------|
| If MSG = Start and \( U = A_1\): | RevealedU[sid] := true \|
| Run SessionA1(Start, A1) \|
| If MSG = MSG2 and \( U = A_1\): | return \( \text{sk}eY[U][sid] \) \|
| Run SessionA2(MSG2, A1) \|
| If MSG = MSG1 and \( U = B_1\): | \|
| Run SessionB1(MSG1, B1) \|
| If MSG = MSG3 and \( U = B_1\): | \|
| Run SessionB2(MSG3, B1) \|

| Execute(Ai, Bj): | Corrupt(U): |
|-------------|---------|
| Honestly run SessionA1(Start, Ai) \|
| Honestly run SessionB1(MSG1, Bj) \|
| Honestly run SessionA2(MSG2, Ai) \|
| Honestly run SessionB2(MSG3, Bj) \|

**Test(sid, U):**

if \( sid \in S \): return \( \perp \) // already tested
if statusU[sid] \neq \text{accept}: return \( \perp \)
if \( \text{sk}eY[U][sid] \neq \text{sk} \):
\( S := S \cup \{\text{sid}\} \)
\( K_0 := \text{sk}eY[U][sid] \)
\( K_1 \xleftarrow{R} \{0, 1\} \)
return \( K_b \)
is broken, i.e., the attacker may have luck in generating $Q_a$ or $Q_b$ without having the access to the authentication keys. When sessions SessionA2 and SessionB2 want to derive a session key, we set a bad flag $badQ$ and abort the session if 1) the received message does not correspond to a previously sent message, 2) the corresponding $Q_a$ and $Q_b$ are valid, and 3) the authentication key is not corrupted before the session finishes. The only way to distinguish Game 3 and Game 4 is to trigger the abort event $badQ$. To bound the probability of this event, we construct an adversary $B$ against the authenticity of the $CCM$-mode Theorem using PRP in Table 4 [32, 35]. When $badQ$ is triggered, $B$ submits $(ID_B, Q_a)$ or $(ID_A, Q_b)$ as its forgery. As the triggering messages $Q_a$ and $Q_b$ have not been produced before under the corresponding authentication keys, the forgery is valid and $B$ wins if $badQ$ is set.

In the proposed protocol, $CCM$-encryption is realized using AES, which is a secure PRP. Since $(n_a, T_a)$ or $(n_b, T_b)$ are 2-block plaintexts, each encryption query will call the block cipher six times\(^4\). Since forgery attempts can only be realized using $Send$ queries while encryption calls can be realized in both $Send$ and $Execute$ queries, there are at most $q_s$ forgery attempts and at most $(q_s + q_e)$ encryption calls. If the tag size $l_t$ is the same as the block size $l_b$, according to (5), we have

\[
Pr[S_3] - Pr[S_4] \leq Adv^{auth}_{CCM,AES}(A) \\
\leq \epsilon_{prp} + q_s + (12q_s + 6q_e)^2/2^l_t
\]

**Game 5:** This game is defined by excluding the event that $A$ is lucky in obtaining $n_a$ and $n_b$. Game 4 has excluded the probability of obtaining $n_a$ and $n_b$ from decrypting $Q_a$ and $Q_b$ or forging $n_a$ and $n_b$ directly with $P_{2a}$ and $P_{2b}$. The only way left to obtain $n_a$ and $n_b$ is by guessing. Since $n_a$ and $n_b$ are ephemeral and fresh random numbers of each session, the probability of a successful guess is not more than $2q_s/2^l_t$.

From the above analysis, Game 4 and Game 5 proceed identically unless $A$ successfully guesses $n_a$ and $n_b$. Therefore:

\[
Pr[S_5] - Pr[S_4] \leq 2q_s/2^l_t
\]

**Game 6:** We define this game by considering that $A$ may compute $sk$ by utilizing $sk'$ in the previous or subsequent sessions between $A$ and $B$. According to the protocol, $sk = H(n_a, n_b) = H(n_a, n_a) = H(n_a, \tilde{n}_b)$ are totally unrelated to $sk'$ in any other session as the two arguments, $n_a$ and $n_b$ or $n_a$ and $\tilde{n}_b$, of the hash function are random nonces. Since this event does not increase $A$’s winning probability,

\[
Pr[S_6] - Pr[S_5] = 0
\]

\(^4\) Including two initial blocks of $CBC$-MAC and $CTR$ [36].
Table 4: The adversary against the authenticity of the CCM. The CCM game provides oracles $CCM_{enc}$ and $CCM_{dec}$ to $B$ [30]. The adversary $A$ has access to oracles $O := \{Send, Execute, RO_H, Corrupt, SK_{Reveal}, Test, RandomGuess\}$. The most relevant codes are highlighted in gray.

```plaintext
| SessionA2($MSG_2, A$) |
|------------------------|
| $(Q_a, T_b) \leftarrow MSG_1$ |
| Recv := Recv $\cup \{T_b\}$ |
| $sid := <T_a, T_b>$ // session identifier |
| init[$sid$] := $A_1$ // initiator of the session |
| $R_{a} := RO_{H}(C_{b})$ |
| $P_{a} := RO_{H}(R_{a}, I_{a})$ |
| $V_a := CCM_{dec}(AES, P_{a}, Q_a)$ |
| If $V_a \neq 0$: // $\emptyset = error$ |
| $\forall [corruptedB=true \& (guessR_b || guessP_{1b}=true)]$ |
| $\forall [corruptedA=false \& (guessP_{2a} || guessR_a || guessP_{1a}=true)]$: |
| $bad_{P_{2a}} := true$; abort |
| $\forall [sid'=\{resP[sid'],I[sid']\}=(A_i, Q_b)$ |
| $bad_{P_{1b}} := false$: |
| $\forall [sid'=\{resR[sid'],R[sid']\}=(B_j, (Q_a, T_b))]:$ |
| $bad_{R_{a}} := false$: |
| $\forall (n_a, T_a) \leftarrow V_a$: |
| $\forall T_{a} = T_{b}$: |
| statusA[$sid$] := accept // Authenticated session |
| $P_{2a} := RO_{H}(P_{1a})$ |
| $P_{2b} := \phi_{2} \oplus P_{ba}$ |
| $n_{b} \leftarrow_{\{0, 1\}^{t}}$: |
| $\forall n_{a} \in N$: abort |
| $N := N \cup \{n_{b}\}$ |
| $Q_{b} := CCM_{enc}(AES, P_{2b}, (n_a, T_b))$ |
| $sk := RO_{H}(n_{a}, n_{b})$ |
| $sk_{b}$[sid] := $sk$ // session key |
| $\forall$[sid] := $Q_a$ // messages outputted by the initiator |
| return $MSG_3 := Q_b$ |
```

We now consider the final advantage of $A$ playing Game 6. $A$ has a non-zero advantage if:

1) Impersonation is true, or
2) Fresh is true and $[b' = b]$.

Impersonation attack is true if $A$ is able to forge a valid $Q_a$ or $Q_b$. Due to Game 3, $A$’s possibility of forging $Q_a$ or $Q_b$ by luckily obtaining $P_{i,a}$ or $P_{i,b}$ have been excluded. Game 4 aborts the event whenever $A$ breaks the authenticity of the CCM. Therefore, in Game 6, Impersonation is always false.

Note that if Fresh is false, $A$’s guessed bit $b'$ would be set to 0. Accordingly, the probability of $b' = b$ would be 1/2, which leads to the advantage being 0 by definition. As all the parameters that $A$ may exploit to compute $sk$ have been excluded, $A$ can only win this game by guessing $sk$. Due to Game 1, the output of the hash function, i.e., the secret key $sk$, is drawn uniformly at random under the random oracle model. Therefore, the probability for $A$ to win Game 6 (i.e., $b' = b$ in this game) is $Pr[S_6] = 1/2$.

From the differences in probabilities of success from (8) to (15) for Game 0 to Game 6, we have:

$$Adv_{\pi_1}^{SS}(A) = |Pr[S_0] - 1/2| = |Pr[S_0] - Pr[S_6]|$$
$$\leq 6e_{na} + 2\epsilon_{puf} + \epsilon_{prf} + \frac{10q_{a} + 37(2q_{a} + q_{r})^2 + q_{a}^2}{2!}$$

The SK-secure proof of $\pi_2$ in Fig. 5 is similar to the above analysis of $\pi_1$. It is omitted due to page limit. Note that the plaintext of CCM mode in this case occupies a 3-block length

which leads to eight block cipher calls, the probability of triggering $bad_{Q}$ in Game 4 needs to be modified accordingly.

Remarks on Perfect Forward Secrecy (PFS): PFS ensures that even if the long-term secrets are compromised, the session keys in the previous sessions will not be compromised. The long-term secrets are $R_{a}$ and $R_{b}$, and $P_{a}$ and $P_{b}$ in the proposed protocol $\pi_1$. If these information are leaked, the adversary can recover $n_{a}$ and $n_{b}$ from the public puzzles, $Q_{a}$ and $Q_{b}$, and uses them to calculate the session key $sk = H(n_{a}, n_{b})$. Therefore, strictly speaking, the proposed MA and KE protocol shown in Fig. 5 cannot achieve PFS. However, due to the generate-upon-request and tamper-proof properties of PUF, PUF response related long-term secrets do not have to be stored and cannot be obtained via intrusive attacks. The $Corrupt(U)$ query can only obtain the stored authentication mask, the leakage of which will not compromise the system’s security. The augmented protocol shown in Fig. 6 can therefore achieve PFS. Even if the long-term secrets are leaked, the adversary $A$ cannot calculate $sk = n_{a}n_{b}P$ with the decrypted $n_{a}P$ and $n_{b}P$ from the past sessions owing to the intractability of ECDH problem. The cost for PFS enhancement is two rounds of elliptic curve point multiplications in each device.

6.2 Automatic Security Analysis Via ProVerif

ProVerif [15] uses the symbolic, Dolev-Yao model [21] of cryptography for analysis. ProVerif is sound in providing correspondence, reachability and observational equivalence proofs [37]. The proposed protocol introduced in Sec. 4 and Sec. 5 are translated into ProVerif scripts and arbitrarily
many number of sessions and message space are simulated between protocol entities.

6.2.1 Mutual Authentication and Various Attacks

One objective of the proposed protocol is to ensure that the mutual authentication between two IoT peer nodes is valid and secure. The provenance proof of the mutual authentication also indicates the resistance of the protocol against replay and MITM attacks. To test this security property, the following events are declared:

1) \texttt{event beginAparam(hostA, hostB)}
2) \texttt{event endAparam(hostA, hostB)}
3) \texttt{event beginBparam(hostB, hostA)}
4) \texttt{event endBparam(hostB, hostA)}
5) \texttt{event beginAfull(hostA, hostB, n_a, P_{la}).}
6) \texttt{event endAfull(hostA, hostB, n_a, P_{la}).}
7) \texttt{event beginBfull(hostB, hostA, n_b, P_{lb}).}
8) \texttt{event endBfull(hostB, hostA, n_b, P_{lb}).}

Two different levels of authentication are used. The first four events prove the agreement on the host names only. “\texttt{event beginAparam(hostA, hostB)}” means hostB has commenced a run of the protocol with hostA and “\texttt{event endAparam(hostA, hostB)}” means hostA believes she has successfully completed the protocol with hostB, and vice versa for “\texttt{event beginBparam(hostB, hostA)}” and “\texttt{event endBparam(hostB, hostA)}”. The events that end with “full” are used for proving the agreement on all parameters of the protocol. For example, “\texttt{event beginAfull(hostA, hostB, n_a, P_{la})}” denotes the intention of hostB to launch the protocol with hostA with the secret nonce \(n_a\) and authentication key \(P_{la}\), and “\texttt{event endAfull(hostA, hostB, n_a, P_{la})}” means hostA believes she has successfully completed the protocol with hostB with the declared authentication messages.

If mutual authentication is established, the following correspondence assertions should hold:

1) \texttt{inj-event(endAparam(hostA,hostB))}
\hspace{1cm} \Rightarrow \texttt{inj-event(beginAparam(hostA,hostB))}
which means if hostA reaches the end of the protocol and she believes she has completed the protocol with hostB, then hostB should have started a session with hostA. The relationship is injective. Similar implication holds for the following assertions.
2) \texttt{inj-event(endBparam(hostB,hostA))}
\hspace{1cm} \Rightarrow \texttt{inj-event(beginBparam(hostB,hostA))}
3) \texttt{inj-event(endAfull(hostA,hostB, n_a, P_{la}))}
\hspace{1cm} \Rightarrow \texttt{inj-event(beginAfull(hostA,hostB, n_a, P_{la}))}
4) \texttt{inj-event(endBfull(hostB,hostA, n_b, P_{lb}))}
\hspace{1cm} \Rightarrow \texttt{inj-event(beginBfull(hostB,hostA, n_b, P_{lb}))}

6.2.2 Reachability and Secrecy

The other objective of the proposed protocol is to evaluate the reachability and secrecy of the shared secrets, i.e., the session key \(sk\), between the two IoT peer nodes. Reachability evaluation is a basic capability of ProVerif, which tests whether the term under test is reachable to an attacker. In our experiments, the following queries are included to test this property:

\texttt{query attacker(messageA); attacker(messageB)}.

where “messageA” (“messageB”) is a free name sent by device \(A\) (\(B\) in encrypted form with the encryption key being \(sk\) over a public channel after the mutual authentication.

6.2.3 Observational Equivalence

In addition to the above correspondence and reachability proofs, four typical classes of observational equivalence (\(OE\)) are also evaluated using ProVerif:

1) Strong secrecy:
This query evaluates if an attacker is able to distinguish when the secrets change. By denoting
\texttt{noninterf messageA; messageB}
the secrecy of the transmitted “messageA”, “messageB” encrypted by the established session key can be proved.

2) Off-line guessing attacks:
If the secrets are weak with low entropy, it may be successfully guessed by the attackers without further interaction with the protocol. The absence of off-line guessing attacks against “messageA” and “messageB” can be evaluated by the query:

\texttt{weaksecret messageA; messageB}

3) \(OE\) between processes that differ only by terms:
This query tests the equivalence between two processes that have the same structure but different terms, which can be encoded by two single “biprocesses”:
\texttt{choice[ENC(messageA, sk), ENC(messageA, randomA)]}
\texttt{choice[ENC(messageB, sk), ENC(messageB, randomB)]}
where the construct “\texttt{choice[term1,term2]}” is used to represent the two processes with different terms while \texttt{randomA} and \texttt{randomB} are freshly generated random values.

4) Real-or-random secrecy:
This query evaluates whether the secrets are indistinguishable from the fresh random values. For example,
\texttt{choice[sk, randomA]}
\texttt{choice[sk, randomB]}

As shown in Fig. 6, the proposed protocols \((\pi_1 \text{ and } \pi_2)\) have successfully achieved mutual authentication (correspondence assertion) and proved the secrecy of the shared keys (reachability) as well as the above-mentioned four classes of observational equivalence.

7 Prototype and Comparison

7.1 Hardware Implementation

Fig. 7 shows the prototype implementation of our proposed system. Two Avnet Ultra96-V2 boards are used as the IoT endpoints. Ultra96-V2 is an ARM-based, Xilinx Zynq UltraScale+TM MPSoC ZU3EG A484 development board. It supports PYNQ for Xilinx FPGA device, which means that Python language and libraries are provided for the application development to benefit from both the FPGA programmable logic and ARM microprocessor. A personal computer is used as the local server. The involvement of the local server is only in the enrollment and update phase as described in Sec. 4.

Any PUF with good uniqueness and randomness to minimize the probability of response collision or correlation
Fig. 6. ProVerif outputs. (a-b): Reachability proof; (c-d): Strong secrecy proof; (e-f): The absence of off-line guessing attacks proof; (g-j): Mutual authentication proof; (k-l): $OE$ between processes that differ only by terms and real-or-random secrecy proof.

that may be exploited by the attackers can be embedded into the IoT devices. The PUF can be flexibly selected to meet the device resource or any specific application requirements. A highly reliable PUF will relax the error correction requirement. Though PUF with near-perfect reliability has been proposed in the literature [38], without loss of generality, we assume that the PUFs embedded in the devices have high but not perfect reliability. Therefore, ECC is required as the PUF responses are used for key generation in the proposed scheme. In our experiment, the reliable and modeling resistant Arbiter PUF from [39] is chosen for the implementation on the FPGA. The PUF overlay is then loaded through the ARM core for system configuration. The remaining cryptographic modules such as ECC and hash are directly imported from Python libraries. For demo purpose, the proposed protocol is designed on top of the network socket. The outputs of devices $A$ and $B$ in the authentication process are indicated in Fig. 7. It shows that the proposed mutual authentication and key exchange can be successfully conducted. The elapsed time from device $A$ initiating the protocol till the completion of mutual authentication and session key generation is less than 1.5 seconds. As only public information, i.e., device identity, challenge, ECC helper data and key mask, are stored in the device, the storage and security requirements in each device are also low.

7.2 Comparison With Existing Works

There are only limited existing works [13], [18], [19], [20] that are able to achieve PUF-based device to device authentication without the need to store the CRPs at the verifier’s side. In this section, only the related protocols in the existing works are compared, i.e., protocols in Fig. 3 of [13], [19] and [20], the BC-PHEMAP protocol in [18], and our proposed mutual authentication and key exchange protocols presented in Fig. 4 and Fig. 5. The comparison is conducted from three perspectives: i) security features; ii) computational complexity; and iii) communication and storage cost.

Table 5 compares the protocols against a list of security features, including mutual authentication, secure session key, the requirement of server participation, NVM for storage of secret locally and PFS. Mutual authentica-
TABLE 5
Comparison of Security Features

| Protocols          | Mutual Authentication | Secure Session Key | No Server Participation | NVM Secret Free | PFS |
|--------------------|-----------------------|--------------------|-------------------------|-----------------|-----|
| JIoT2017 [19]      | ✓                     |                    |                         | ✓               | x   |
| TDSC2018 [13]      | ✓                     | ✓                  |                         | x               | x   |
| Elsevier2019 [18]  | ✓                     | x                  |                         | x               | ✓   |
| JSEN2021 [20]      | ✓                     |                    |                         | ✓               | x   |
| This Work (Fig. 4) | ✓                     | ✓                  |                         | ✓               | x   |
| This Work (Fig. 5) | ✓                     | ✓                  |                         | ✓               | ✓   |

✓: the security feature is provided; ×: the security feature is not provided.

TABLE 6
Comparison of Computational Complexity

| Protocols          | Computational Complexity                  | Node A                | Node B                | Third Party1 | Total                          |
|--------------------|------------------------------------------|-----------------------|-----------------------|--------------|--------------------------------|
| JIoT20172 [19]     | 1\(N_H + 6N_{MAC} + 2N_{PUF}\) + 3\(N_{Enc}\) + 2\(N_{ECC}\) | 1\(N_H + 4N_{MAC} + 2N_{PUF}\) + 2\(N_{Enc}\) + 2\(N_{ECC}\) | 2\(N_H + 4N_{MAC} + 4N_{Enc}\) + 4\(N_{ECC}\) (Server) | 4\(N_H + 14N_{MAC} + 4N_{PUF}\) + 4\(N_{ECC}\) |
| TDSC2018 [13]      | 9\(N_H + 1N_{PUF} + 1N_{ECC}\) + 3\(N_{Pa}\) + 2\(N_{PM}\) + 1\(N_{BP}\) | 9\(N_H + 1N_{PUF} + 1N_{ECC}\) + 3\(N_{Pa}\) + 2\(N_{PM}\) + 1\(N_{BP}\) | 12\(N_H + 10N_{Pa} + 4N_{PM}\) + 2\(N_{BP}\) (Verifier), 0 (SAP) | 30\(N_H + 2N_{PUF} + 2N_{ECC}\) + 16\(N_{Pa}\) + 8\(N_{PM}\) + 4\(N_{BP}\) |
| Elsevier20192 [18] | 4\(N_{PUF} + 4N_{ECC}\) | 4\(N_{PUF} + 4N_{ECC}\) | 0 (Authentication service) | 8\(N_{PUF} + 8N_{ECC}\) |
| JSEN2021 [20]      | 6\(N_H + 1N_{PUF} + 1N_{ECC}\) + 3\(N_{Pa}\) + 8\(N_{PM}\) | 6\(N_H + 1N_{PUF} + 1N_{ECC}\) + 3\(N_{Pa}\) + 8\(N_{PM}\) | 0 | 12\(N_H + 2N_{PUF} + 2N_{ECC}\) + 16\(N_{Pa}\) + 16\(N_{PM}\) |
| This Work3 (Fig. 4) | 3\(N_H + 1N_{PUF} + 1N_{ECC}\) + 2\(N_{Enc}\) + 2\(N_{MAC}\) | 3\(N_H + 1N_{PUF} + 1N_{ECC}\) + 2\(N_{Enc}\) + 2\(N_{MAC}\) | 0 | 6\(N_H + 2N_{PUF} + 2N_{ECC}\) + 4\(N_{MAC}\) |
| This Work3 (Fig. 5) | 2\(N_H + 1N_{PUF} + 1N_{ECC}\) + 2\(N_{Enc}\) + 2\(N_{MAC}\) + 2\(N_{PM}\) | 2\(N_H + 1N_{PUF} + 1N_{ECC}\) + 2\(N_{Enc}\) + 2\(N_{MAC}\) + 2\(N_{PM}\) | 0 | 4\(N_H + 2N_{PUF} + 2N_{ECC}\) + 4\(N_{MAC}\) + 4\(N_{PM}\) |

1 In this column, “\(N(M)\)” means \(N\) operations conducted in the third party \(M\). 0(\(M\)) means that the third party \(M\) does not have to perform computation but serves as an authentication helper data provider.
2 Without loss of generality, PUFs used in the protocols are assumed to have non-ideal reliability performance. Therefore, the omitted \(N_{ECC}\) is included in the computational complexity analysis of [18], [19] for a fair comparison.
3 Since CCM mode is a combination of CBC-MAC and CTR encryption, one CCM operation is considered to be composed of one \(N_{MAC}\) and one \(N_{Enc}\).

TABLE 7
Comparison of Communication and Storage Costs

| Protocols          | Protocol Rounds | Communication Cost | Storage Cost (per pair) |
|--------------------|-----------------|--------------------|-------------------------|
|                    |                 | Node A             | Node B                  | Node A | Node B |
| JIoT2017 [19]      | 7               | 2880 bits          | 1664 bits               | 2432 bits | 6976 bits | 512 bits1 | 512 bits1 |
| TDSC2018 [13]      | 12              | 2112 bits          | 2112 bits               | 11456 bits | 15680 bits | 0 | 0 |
| Elsevier20192 [18] | 6               | 608 bits           | 512 bits               | 1024 bits | 2144 bits | 3072 bits | 2048 bits |
| JSEN2021 [20]      | 3               | 2336 bits          | 2336 bits               | 0 bits | 4672 bits | 2048 bits | 2048 bits |
| This Work (Fig. 4) | 3               | 1472 bits          | 896 bits               | 0 bits | 2368 bits | 1056 bits | 1056 bits |
| This Work (Fig. 5) | 3               | 1782 bits          | 1152 bits              | 0 bits | 2934 bits | 1056 bits | 1056 bits |

1 The storage cost of JIoT2017 [19] is resulted from two ECC helper data of PUF responses.
2 In the example BC PHEMAP protocol of [18], the Babel-chain carnet contains four expected interactions. The underlying communication and storage costs are calculated and shown in this table. Note that the corresponding four ECC helper data are also accounted in the calculation of storage cost.
and our proposed augmented protocol in Fig. 5 stand out as the only two candidates that meet all the desirable security features for securing MA and KE in P2P IoT setting.

Table 6 compares the computational complexities of these protocols. The computational complexity listed for each protocol includes computations performed during the MA and KE phase at both the device node and the participating third-party node (if any). Let \(N_{H}, N_{MAC}, N_{PUF}, N_{ECC}, N_{Enc}, N_{PM}, N_{FM} \) and \(N_{RP}\) denote the numbers of rounds of hash function, message authentication code (MAC), PUF, error correction code for PUF, encryption and decryption, point addition over elliptic curve, point multiplication over elliptic curve and bilinear pairing, respectively. Since the proposed protocol does not dictate the type of cryptographic modules such as PUF and hash as long as they meet the required functionality with acceptable implementation cost, Table 7 only lists the names of the cryptographic modules for a fair comparison. Similarly, for an implementation-platform-independent comparison of computational complexity, the number of rounds for each required operation is listed and compared. Trivial operations like concatenation or bitwise exclusive OR are omitted without impacting the relative accuracy of the comparison.

As shown in Table 6, protocols, TDSC2018 [13], Elsevier2019 [18] and JIoT2017 [19], all require a third party to complete the mutual authentication between devices. Among the existing protocols, TDSC2018 [13] and JSEN2021 [20] have the highest computational complexity as they both require compute-intensive PKC over elliptic curve. The protocol TDSC2018 [13] requires 16 point additions, 8 point multiplications and 4 bilinear pairing operations to be performed in total while the protocol JSEN2021 [20] requires 6 rounds of point addition and 16 rounds of point multiplication. A point addition and a point multiplication over elliptic curve secp256r1 consume around 2.56 ms and 112.34 ms, respectively, on TI’s CC2538 SoC (in comparison, SHA256 only consumes around 0.061 ms in the same platform) [20]. Bilinear pairing has been widely regarded as an expensive operation for terminal devices [40]. Though the proposed augmented protocol in Fig. 5 also involves elliptic curve cryptography, only two point multiplications are needed on each device to achieve PFS. The protocol Elsevier2019 [18] is lightweight but its practicability is severely constrained by the PUF reliability problem. Besides, as analyzed in Sec. 2.2, it is also vulnerable to modeling attacks. The number of cost-effective hash operation rounds of JIoT2017 [19] is similar to ours, but JIoT2017 [19] requires more MAC, encryption/decryption, PUF and the associated ECC operations. Therefore, its overall complexity is estimated to be higher than our proposed protocol in Fig. 4 besides its need for a trust server. In addition to the high performance, privacy and security achievable without the need for a trusted third party, our proposed protocol outperforms other existing schemes in terms of its low component cost for direct P2P authentication and the avoidance of storing a secret key locally.

Another important factor for protocol comparison is the communication bandwidth and storage costs, which can be calculated for each device pair. Note that global public parameters like the elliptic curve to be used are not accounted. Following the same generalization setting in [20], the device identities and timestamps are assumed to be 32 bits, the outputs of hash, private keys, random nonces, and challenges and responses of PUFs are all recommended to be 256 bits. The tag produced by MAC operations is assumed to be 128 bits. As for elliptic curve points, we assume each point \((P_x, P_y)\) is 512 bits, where \(P_x\) and \(P_y\) denote the 256-bit \(x\) and \(y\) coordinates, respectively. Besides, we assume the encryption method used in [19] is also the CTR-mode encryption and the ciphertext and plaintext are assumed to have the same length. As shown in Table 7, those protocols that involve a third party in the mutual authentication process, i.e., JIoT2017 [19], TDSC2018 [13] and Elsevier2019 [18] require a higher number of message exchanges. Though the first two protocols [13], [19] have minimum storage cost per device pair, their respective total communication cost is much higher than other protocols in comparison. Each of our two proposed protocols can be completed with only three message passes, and consumes very low bandwidth (around 3,000 bits) and storage (around 1,000 bits).

In summary, based on the results of the comparison in Table 5 to Table 7, our proposed PUF-based MA and KE protocols provide a very high-level of security and require comparatively low computational complexity, communication bandwidth and storage cost.

8 Conclusion

This paper presents a PUF-based protocol for direct mutual authentication with an automatic key exchange for encrypted data transmission between IoT end devices. The proposed protocol overcomes the PUF’s bottleneck in providing direct mutual authentication between resource-constrained devices without going through a proxy server. The requirement to store and safekeep a database of all provers’ CRPs in the verifier’s local memory is not only difficult to be satisfied by the resource-limited IoT devices, but the locally stored secret is also vulnerable to hardware attacks due to the accessibility of IoT endpoint devices. This security problem is solved by creating and storing the authentication mask data that do not need to be kept private in the device. A new and important side benefit of the proposed protocol is that secure key exchange between the two endpoints is directly established upon successful mutual authentication with neither additional message transfer nor complex public-key algorithm. Game-based formal security analysis and ProVerif simulation have been performed to prove the security soundness of the proposed protocol. The hardware cost and computational complexity are reasonably low in comparison with the existing PUF-based CRP-storage-free authentication protocols.

References

[1] I. HÜBSCHMANN, “P2P IoT vs. Database-driven IoT - Which to choose?” Nov. 27 2018. [Online]. Available: https://www.nabto.com/p2p-vs-database-iot/

[2] Y. Zheng, Y. Cao, and C. H. Chang, “UDhashing: Physical unclonable function based user-device hash for endpoint authentication,” IEEE Trans. Industrial Electron., vol. 66, no. 12, pp. 9559–9570, Dec. 2019.
[3] Y. Zheng, Y. Cao, and C. H. Chang, “A PUF-based data-device hash for tampered image detection and source camera identification,” *IEEE Trans. Inf. Forensics, Security*, vol. 15, pp. 620–634, Jul. 2019.

[4] J. R. Wallrabenstein, “Practical and secure IoT device authentication using physical unclonable functions,” in *Proc. 2016 IEEE 4th Int. Conf. Future Internet Things Cloud (FiCloud)*. Vienna, Austria: IEEE, Aug. 2016, pp. 99–106.

[5] T. Idrissi and M. Bayouni, “Lightweight highly secure PUF protocol for mutual authentication and secret message exchange,” in *Proc. 2017 IEEE Int. Conf. RFID Technol. Appl. (RFID-TA)*. Warsaw, Poland: IEEE, 2017, pp. 214–219.

[6] P. Gope and B. Sikdar, “Lightweight and privacy-preserving two-factor authentication scheme for IoT devices,” *IEEE Internet Things J.*, vol. 6, no. 1, pp. 580–589, Feb. 2018.

[7] C. H. Chang, Y. Zheng, and L. Zhang, “A retrospective and a look forward: Fifteen years of physical unclonable function advancement,” *IEEE Circuits Syst. Mag.*, vol. 17, no. 3, pp. 32–62, Aug. 2017.

[8] S. P. Skorobogatov, “Semi-invasive attacks - A new approach to hardware security analysis,” University of Cambridge, Tech. Rep. 630, Apr. 2005.

[9] P. Courbon, S. Skorobogatov, and C. Woods, “Reverse engineering flash EEPROM memories using scanning electron microscopy,” in *Proc. Int. Conf. Smart Card Research Adv. Appl.* Cannes, France: Springer, Nov. 2016, pp. 57–72.

[10] C. Hellmeier, D. Nedesposav, C. Tarnovsky, J. S. Krissler, C. Boit et al., “Breaking and entering through the silicon,” in *Proc. 2013 ACM SIGSAC Conf. Comput. Commun. Security*. Berlin, Germany: ACM, Nov. 2013, pp. 733–744.

[11] N. F. Ghalaty, B. Yuce, M. Taha, and P. Schaumont, “Diffential fault intensity analysis,” in *2014 Workshop Fault Diagnosis Tolerance in Cryptography*. Busan, South Korea: IEEE, Sept. 2014, pp. 49–58.

[12] Y. Zheng and C. H. Chang, “Secure mutual authentication and key-exchange protocol between PUF-embedded IoT endpoints,” in *Proc. 2021 IEEE Int. Symp. Circuits Syst. (ISCAS)*. Daegu, Korea, May 2021, pp. 1–4.

[13] U. Chatterjee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, R. S. Chakraborty et al., “Building PUF based authentication and key exchange protocol for IoT without explicit CRPs in verifier database,” *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 3, pp. 424–437, May–Jun. 2018.

[14] V. Shoup, “Sequences of games: a tool for taming complexity in security proofs.” *LACK Cryptol. ePrint Arch.*, vol. 2004, p. 302, 2004.

[15] B. Blanchet, B. Smyth, V. Cheval, and M. Sylvestre, “Proverif 2.02: Automatic cryptographic protocol verifier, user manual and tutorial,” pp. 01–129, Jul. 2020.

[16] Grayshift, “GrayKey: Extract evidence from encrypted mobile devices.” [Online]. Available: https://www.grayshift.com/

[17] Cellebrite, “Cellebrite UFED: The industry standard for accessing digital device data.” [Online]. Available: https://www.cellebrite.com/en/ufed/

[18] M. Barbareschi, A. De Benedictis, E. La Montagna, A. Mazzocca, and N. Mazzocca, “A PUF-based mutual authentication scheme for cloud-edges IoT systems,” *Future Generation Comput. Syst.*, vol. 101, pp. 246–261, Dec. 2019.

[19] M. N. Aman, K. C. Chua, and B. Sikdar, “Mutual authentication in IoT systems using physical unclonable functions,” *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1327–1340, Oct. 2017.

[20] S. Li, T. Zhang, B. Yu, and K. He, “A provably secure and practical PUF-based end-to-end mutual authentication and key exchange protocol for IoT,” *IEEE Sensors J.*, vol. 21, no. 4, Feb. 2021.

[21] D. Dolev and A. Yao, “On the security of public key protocols,” *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 198–208, Mar. 1983.

[22] R. Canetti and H. Krawczyk, “Analysis of key-exchange protocols and their use for building secure channels,” in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.* Springer, Apr. 2001, pp. 453–474.

[23] Q. Chen, K. Su, C. Liu, and Y. Xiao, “Automatic verification of web service protocols for epistemic specifications under Dolev-Yao model,” in *Proc. 2010 Int. Conf. Service Sciences*. Hangzhou: IEEE, May 2010, pp. 49–54.

[24] B. Narwal and A. K. Mohapatra, “A survey on security and authentication in wireless body area networks,” *J. Systems Architecture*, p. 101883, Feb. 2021.

[25] A. Aghaie and A. Moradi, “TI-PUF: Toward side-channel resistant physical unclonable functions,” *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3470–3481, Apr. 2020.

[26] Y. Nozaki and M. Yoshikawa, “Countermeasure of lightweight physical unclonable function against side-channel attack,” in *2019 Cybersecurity and Cyberforensics Conference (CCC).* IEEE, May 2019, pp. 30–34.

[27] R. Haakegaard and J. Lang, “The elliptic curve diffie-hellman (ecdh),” Dec. 2015. [Online]. Available: http://koclab.cs.ucsb.edu/teaching/ecc/project/2015Projects/Haakegaard-Lang.pdf

[28] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thome, L. Valenta et al., “Imperfect forward secrecy: How diffie-hellman fails in practice,” in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, Oct. 2015, pp. 5–17.

[29] A. Houghton, *Error coding for engineers*. Springer Science & Business Media, 2001, vol. 641.

[30] J. Jonsson, “On the security of ctr+ cbc-mac,” in *Proc. Int. Selected Areas in Cryptography*. St. John’s, NF, Canada: Springer, Aug. 2002, pp. 76–93.

[31] M. Skrobot and J. Lancrenon, “On composability of game-based password authenticated key exchange,” in *2018 IEEE European Symp. Security Privacy (EuroS&P)*. London, UK: IEEE, Apr. 2018, pp. 443–457.

[32] J. Pan, C. Qian, and M. Ringerud, “Signed (group) diffie-hellman key exchange with tight security,” in *Cryptographers’ Track at the RSA Conference*. Springer, 2021, pp. 201–226.

[33] D. A. McGrew, “Counter mode security: Analysis and recommendations,” *Cisco Systems*, vol. 2, no. 4, Nov. 2002.

[34] H. Davis and F. Günther, “Tighter proofs for the sigma and tls 1.3 key exchange protocols,” in *Proc. Int. Conf. Applied Cryptography and Network Security*. Springer, 2021, pp. 448–479.

[35] F. Guo, W. Susilo, and Y. Mu, “Introduction to security reduction,” *Springer*, Jun. 2018.

[36] P.-A. Fouque, G. Martinet, F. Valette, and S. Zimmer, “On the security of the cm encryption mode and of a slight variant,” in *Proc. Int. Conf. Applied Cryptography and Network Security*. Springer, 2008, pp. 411–428.

[37] S. Malladi and B. Blanchet, “ProVerif web interface.” [Online]. Available: http://proverif16.paris.inria.fr/index.php

[38] D. Jeon et al., “A Physical Unclonable Function With Bit Error Rate < 2.3 ×10−8 Based on Contact Formation Probability Without Error Correction Code,” *IEEE J. Solid-State Circuits*, vol. 55, no. 3, pp. 805–816, Nov. 2019.

[39] S. S. Zalivaka, A. A. Ivaniuk, and C. H. Chang, “Reliable and modeling attack resistant authentication of arbiter PUF in FPGA implementation with trinary quadruple response,” *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 1109–1123, Apr. 2018.

[40] K. Sovjany, A. Dasgupta, S. Ray, and M. S. Obaidat, “An efficient elliptic curve cryptography-based without pairing KPABE for Internet of Things,” *IEEE J. Internet Things J.*, vol. 14, pp. 2154–2163, Jun. 2019.
Yue Zheng (S’15-M’20) received the B. Eng. Degree in School of Communication and Information Engineering from Shanghai University (SHU, China), and Ph.D. degree in School of Electrical and Electronic Engineering, Nanyang Technological University (NTU, Singapore), in 2015 and 2020, respectively. She was a visiting scholar at Kyoto University between March to June 2019 and she is currently a research fellow at NTU. Her area of research includes hardware security, physical unclonable function, security and privacy of Artificial Intelligence of Things. Dr. Zheng currently serves as the Associate Editor of Transactions on Circuits and Systems II: Express Briefs, the PhD Forum Chair of 2022 Asian Hardware-Oriented Security and Trust (AsianHOST 2022) conference, the Special Session Chair of the 15th International SoC Design Conference (ISOCC 2022). She also served as the Special Session organizer of AICAS 2022, the review committee member and the special session organizer of 2021 IEEE International Symposium on Circuits and Systems (ISCAS 2021), and the track chair of the 29th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC 2021). Dr. Zheng is also a member of VLSI Systems and Applications Technical Committee (VSA-TC).

Wenyu Liu (S’17-M’21) received the B.S. degree in microelectronics from Shenzhen University, China, in 2014, the B.S. degree in physics from Umea University, Sweden, in 2014, the M.S. degree in IC design engineering from Hong Kong University of Science and Technology, and the Ph.D. degree from Nanyang Technological University (NTU), Singapore in 2021. He is currently a research fellow at NTU. His current research interests include hardware security, machine learning accelerator and fault injection attack.

Chongyan Gu (S’14-M’16-SM’21) received the Ph.D. degree from Queen’s University Belfast, Belfast, U.K., in 2016. She is currently a Lecturer in the School of EEECS at NTU, Singapore. She was an assistant professor in the School of Electrical and Electronic Engineering (EEE) of NTU. Associate Professor of the School of Electrical and Electronic Engineering (EEE) of NTU. He held joint appointments with the university as Assistant Chair of Alumni from 2008 to 2014, Deputy Director of the Center for High Performance Embedded Systems from 2000 to 2011, and Program Director of the Center for Integrated Circuits and Systems from 2003 to 2009. He has coedited 4 books, published 13 book chapters, more than 100 international journal papers (more than 70 are in IEEE) and more than 180 refereed international conference papers (mostly in IEEE), and delivered more than 50 keynotes, tutorial and invited seminars. His current research interests include hardware security, unconventional number systems, and low-power and fault-tolerant digital signal processing algorithms and architectures.

Dr. Chang currently serves as the Senior Area Editor of IEEE Transactions on Information Forensic and Security, and Associate Editor of the IEEE Transactions on Circuits and Systems-I and IEEE Transactions on Very Large Scale Integration (VLSI) Systems. He also served as the Associate Editor of the IEEE Transactions on Information Forensic and Security and IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems from 2016 to 2019, IEEE Access from 2013 to 2019, IEEE Transactions on Circuits and Systems-I from 2010 to 2013, Integration, the VLSI Journal from 2013 to 2015, Springer Journal of Hardware and System Security from 2016 to 2020 and Microelectronics Journal from 2014 to 2020. He guest edited around 10 special issues and served in the organizing and technical program committee of more than 70 international conferences (mostly IEEE). He is an IEEE Fellow, IET Fellow, and 2018-2019 Distinguished Lecturer of IEEE Circuits and Systems Society.

Chi-Ho Wong (S’92-M’98-SM’03-F’18) received the B.Eng. (Hons.) degree from the National University of Singapore in 1988, and the M. Eng. and Ph.D. degrees from Nanyang Technological University (NTU) of Singapore in 1993 and 1998, respectively. He is an Associate Professor of the School of Electrical and Electronic Engineering (EEE) of NTU. He held joint appointments with the university as Assistant Chair of Alumni from 2008 to 2014, Deputy Director of the Center for High Performance Embedded Systems from 2000 to 2011, and Program Director of the Center for Integrated Circuits and Systems from 2003 to 2009. He has coedited 4 books, published 13 book chapters, more than 100 international journal papers (more than 70 are in IEEE) and more than 180 refereed international conference papers (mostly in IEEE), and delivered more than 50 keynotes, tutorial and invited seminars. His current research interests include hardware security, unconventional number systems, and low-power and fault-tolerant digital signal processing algorithms and architectures.

Dr. Chang currently serves as the Senior Area Editor of IEEE Transactions on Information Forensic and Security, and Associate Editor of the IEEE Transactions on Circuits and Systems-I and IEEE Transactions on Very Large Scale Integration (VLSI) Systems. He also served as the Associate Editor of the IEEE Transactions on Information Forensic and Security and IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems from 2016 to 2019, IEEE Access from 2013 to 2019, IEEE Transactions on Circuits and Systems-I from 2010 to 2013, Integration, the VLSI Journal from 2013 to 2015, Springer Journal of Hardware and System Security from 2016 to 2020 and Microelectronics Journal from 2014 to 2020. He guest edited around 10 special issues and served in the organizing and technical program committee of more than 70 international conferences (mostly IEEE). He is an IEEE Fellow, IET Fellow, and 2018-2019 Distinguished Lecturer of IEEE Circuits and Systems Society.

Chip-Hong Chang (S’92-M’98-SM’03-F’18) received the B.Eng. (Hons.) degree from the National University of Singapore in 1988, and the M. Eng. and Ph.D. degrees from Nanyang Technological University (NTU) of Singapore in 1993 and 1998, respectively. He is an Associate Professor of the School of Electrical and Electronic Engineering (EEE) of NTU. He held joint appointments with the university as Assistant Chair of Alumni from 2008 to 2014, Deputy Director of the Center for High Performance Embedded Systems from 2000 to 2011, and Program Director of the Center for Integrated Circuits and Systems from 2003 to 2009. He has coedited 4 books, published 13 book chapters, more than 100 international journal papers (more than 70 are in IEEE) and more than 180 refereed international conference papers (mostly in IEEE), and delivered more than 50 keynotes, tutorial and invited seminars. His current research interests include hardware security, unconventional number systems, and low-power and fault-tolerant digital signal processing algorithms and architectures.

Dr. Chang currently serves as the Senior Area Editor of IEEE Transactions on Information Forensic and Security, and Associate Editor of the IEEE Transactions on Circuits and Systems-I and IEEE Transactions on Very Large Scale Integration (VLSI) Systems. He also served as the Associate Editor of the IEEE Transactions on Information Forensic and Security and IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems from 2016 to 2019, IEEE Access from 2013 to 2019, IEEE Transactions on Circuits and Systems-I from 2010 to 2013, Integration, the VLSI Journal from 2013 to 2015, Springer Journal of Hardware and System Security from 2016 to 2020 and Microelectronics Journal from 2014 to 2020. He guest edited around 10 special issues and served in the organizing and technical program committee of more than 70 international conferences (mostly IEEE). He is an IEEE Fellow, IET Fellow, and 2018-2019 Distinguished Lecturer of IEEE Circuits and Systems Society.