Efficient and Strong Symmetric Password Authenticated Key Exchange With Identity Privacy for IoT

Huanhuan Lian, Yafang Yang, and Yunlei Zhao

***Abstract*—Password authenticated key exchange (PAKE) allows two parties with a shared password to establish a session key. In order to provide secure and private communication between devices in an Internet of Things (IoT) environment, the PAKE protocol is considered one of the most common and promising security methods. However, many existing PAKE proposals still face challenges in security and efficiency. First, both the terminals of participants may suffer from the compromise attack and precomputation attack, which will lead to the leakage of password. Second, the majority of the existing schemes cannot guarantee participants' identity privacy. Third, most PAKE protocols are not suitable for IoT devices with limited computing capability because of a large number of exponential and pairing operations. To address these issues, we propose a strong symmetric PAKE protocol for IoT devices, which only requires 3 exponentiations per party. The proposed scheme can protect both parties from compromise and precomputation attacks, in which the password file relies on the identity and random salt. What's more, our protocol guarantees the identity privacy, that is, the transmitted record and password file will not reveal the identity information. We further present a new security model for our protocol, and prove that the proposed scheme is secure under this model. Finally, we show the practicality of our PAKE via experiments and efficiency analysis.**

***Index Terms*—Authenticated key exchange, identity privacy, Internet of Things (IoT) devices, precomputation, preshared password.**

I. Introduction

THE ADVANCEMENT of the Internet technology has provided the support for rapid development in Internet of Things (IoT) environments. The IoT is a self-structured network in which IoT devices, such as digital devices and computing devices, can collect data from the IoT environment and transmit data to a server through the Internet [1], [2]. According to the prediction [3], there will be 25 billion IoT devices connected by 2025. With the rapid increase in the number of IoT devices, the data of communication interaction has also grown [4], [5], [6], [7]. However, the vast majority of data is transmitted over public channels. To protect personal data and privacy for two parties, it is particularly significant to realize the security of communication between IoT devices [8], [9]. Authentication key exchange providing confidentiality, data integrity and authentication services for communication, is considered as one of the forms of protocol to ensure the security of the IoT [10], [11], [12].

Presently, many authenticated key agreement protocols for IoT environments are based on public key infrastructure (PKI), which suffer from a burden certificate management problem. Password authenticated key exchange (PAKE) protocol is a core cryptographic primitive that allows two parties with a preshared low-entropy password to agree on a common session key. It is widely used in IoT because it eliminates complicated certificate management and distribution, which makes the protocol well implementable. PAKE protocol is divided into two categories: 1) symmetric PAKE and 2) asymmetric PAKE (aPAKE). In symmetric setting, the client and server share the same knowledge related with a password to authenticate each other [13], [14], [15], [16], [17]. However, there is a common problem in such protocols. If the server is compromised, an adversary who gains access to the server's local database of passwords can immediately use any of these passwords to impersonate other users. In order to reduce the risk caused by a compromised password, the notion of aPAKE was put forward. It allows the server to store verifiers instead of the plaintext-format passwords in its database [18], [19], [20], [21], [22]. Unfortunately, these aPAKE protocols suffer from precomputation attacks because the verifier in these protocols is transformed from a password using a deterministic one way function and the salt is transmitted publicly. That is to say, an attacker can instantly find a user's password once the server is compromised by precomputing a list of function values for all possible passwords. To prevent precomputation attacks, Jarecki et al. [23] proposed a new notion of strong aPAKE (saPAKE). The scheme gives a solution to increase the offline computation overhead by user-side interacted hashing without transmitting the salt from server

Manuscript received 8 May 2022; revised 18 August 2022; accepted 26 October 2022. Date of publication 4 November 2022; date of current version 7 March 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802000 and Grant 2022YFB2701600; in part by the National Natural Science Foundation of China under Grant 1536205 and Grant 61472084; in part by the Shanghai Innovation Action Project under Grant 16DZ1100200; in part by the Shanghai Science and Technology Development Funds under Grant 16JC1400801; in part by the Technical Standard Project of Shanghai Scientific and Technological Committee under Grant 21DZ2200500; and in part by the Shandong Provincial Key Research and Development Program of China under Grant 2017CXG0701 and Grant 2018CXGC0701. *(Corresponding author: Yunlei Zhao.)*

Huanhuan Lian and Yafang Yang are with the College of Computer Science and Technology, Fudan University, Shanghai 200433, China (e-mail: 19110240032@fudan.edu.cn; 18110240046@fudan.edu.cn).

Yunlei Zhao is with the College of Computer Science and Technology, Fudan University, Shanghai 200433, China, also with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China, and also with the State Key Laboratory of Cryptology, Beijing 100036, China (e-mail: ylzhao@fudan.edu.cn).

Digital Object Identifier 10.1109/JIOT.2022.3219524

to user, and has no instaneous password retrieval upon server compromise. In such saPAKE protocol, the server can be protected from these attacks. However, in the IoT environment the information stored on the client device is at risk of compromise, which leaves the client-side vulnerable. Consequently, achieving the privacy of both parties' passwords is a critical issue which can not be ignored.

With the continuous development of the communication technology, in many IoT applications, the importance of participants' identity and data privacy protection in communication has become clear. Hence, identity privacy is also an indispensable security property in the PAKE protocol. Moreover, it is well known that IoT devices are composed of several resource-limited components, such as battery, storage, and low-power processors. Therefore, we should take low computational overhead into consideration as well when designing the PAKE scheme for IoT devices.

A. *Related Work*

In recent years, various of key agreement schemes for IoT environment have been proposed [24], [25], [26], [27]. Farash et al. [24] designed a new and improved user authenticated and key agreement scheme for heterogeneous wireless sensor network. They dealt with the man-in-the-middle attack and provided forward secrecy. Srinivas et al. [26] built an anonymous user-authenticated key agreement protocol for the IoT environment, in which only authorized uses have access to the services.

Most of the above protocols are developed based on public-key infrastructure, which leads to many problems, such as certificate storage and certificate management. To solve the problems brought by certificates, the password authentication key exchange is developed and widely used. Starting from the work of Bellovin and Merritt [28], the PAKE protocol was first studied and formalized in the game-based indistinguishability method by Bellare et al. [14]. There later followed plenty of PAKE proposals [13], [17], [29], [30], [31] with varied security and complexity.

However, in such protocols the passwords are directly stored on server device, which will bring about exposure of the password upon device compromise. To deal with this issue, Bellovin and Merritt introduced an augmented EKE protocol [32], where the server possesses a verifier instead of a password itself. To strengthen its security and efficiency, different types of verifier-based PAKE (VPAKE) protocols [20], [33], [34], [35] from different perspectives have been proposed. Benhamouda and Pointcheval [20] proposed a new VPAKE (or called aPAKE) protocol. They defined the notions of password hashing and tight one wayness, and prove the protocol in the standard model. Subsequently, Kiefer and Manulis [33] devised a concrete construction of zero-knowledge password policy checks (ZKPPCs) protocol and a general VPAKE protocol for ASCII-based password and policies. Pointcheval and Wang [34] revisited the protocol proposed by Jablon [35] and used ecliptic curves to construct a verifier-based two-basis PAKE, which is efficient at both computation and communication levels. Hoang et al. [25] proposed a password-based signcryption key exchange, in which the server stores verifiers. The protocol is constructed by adopting a signcryption scheme. Shin et al. [36] claimed that most of VPAKE protocols involve strong assumptions and presented a single VAPKE protocol by using tramper-proof hardware without random oracles.

Unfortunately, Jarecki et al. [23] pointed out that these aPAKE protocols are all vulnerable to precomputation attacks. Namely, the attacker $\mathcal{A}$ can set up a table of $(H(pw), pw)$ pairs for all passwords pw, and it finds a user's password once the server is compromised. They presented two constructions for saPAKE by utilizing an oblivious pseudorandom function. Bradley et al. [37] proposed an alternative UC saPAKE protocol which employs a new form of conditional key encapsulation mechanism and a general smooth projective hash function. It uses only two flows and avoids hashing onto a group. Although saPAKE supports protection to the server, the user is still vulnerable to the above-mentioned attacks. Besides, these protocols contain many exponentiations, and efficiency is a handicap. Naor et al. [38] introduced a secure PAKE protocol that resists against compromise for all parties. Nevertheless, it performs too many pairing operations and hashing onto group, which requires high computational expenditure. An additional disadvantage of most of schemes mentioned above is the lack of identity privacy. Therefore, it is significant to investigate a symmetric PAKE scheme to satisfy the strong security requirements and support high computational efficiency.

B. *Our Contributions*

Considering the above problems, an open question arises naturally: can we put forward a secure and efficient symmetric PAKE protocol for IoT devices that simultaneously enjoys: 1) compromise resilience for both parties and precomputation attack resistance; 2) no pairing operation by each other; and 3) identity privacy protection? We investigate the question and propose a novel symmetric PAKE protocol with identity privacy for IoT devices. The main contributions of our proposed protocol are summarized as follows.

1) We design a strong symmetric password-authenticated key agreement between IoT devices in a client–client setting, which protects both parties from compromise and resists precomputation attacks. The devices store the password files instead of clear text passwords. The password file is derived from the password, a salt and identity. It is comprised of two elements, one of which is computed by the password hash and a random salt, and another is computed by the salt hash and the identity of the participant. The passwords are protected by the password files stored by all participants. Besides, the precomputation attack can be prevented even if the password file is compromised, because the random salt is not stored in the terminal and will never be revealed.
2) Our protocol is quite efficient: each party establishes a same session key without performing pairing operations. The proposed scheme mainly relies on hash, exclusive-or and few exponentiation operations.
3) Our symmetric PAKE protocol provides identity privacy protection. Specifically, the transmission does not leak the identity information of the participants. And even if the password file is compromised, there is no way

to get the identity information from it. In addition, our protocol also has the following advantages: resilience to impersonation attack, replay attack, perfect forward security, and mutual authentication.

We present a new security model that departs from the previous model for PAKE protocol by capturing forward secrecy and identity privacy. Finally, we give the formal security analysis for our proposed scheme under the new security model and compare the proposed scheme with state-of-the-art by means of implementation, which demonstrate our scheme possesses satisfactory security and efficiency.

II. Preliminaries

Definition 1 [Discrete Logarithm (DL)]: Given random point $(P, Q) \in \mathbb{G}$, where $Q = P^a, a \in \mathbb{Z}_q^*$, the advantage of any probabilistic polynomial-time (PPT) adversary to compute a from Q is negligible.

Definition 2 [Computational Diffie–Hellman (CDH)]: Given a tuple $(g, g^x, g^y) \in \mathbb{G}$, where $x, y \in \mathbb{Z}_q^*$, the advantage of any PPT adversary to calculate $g^{xy} \in \mathbb{G}$ without the knowledge of x and y is negligible.

Definition 3 (One-Way Collision-Resistant Hash Function): A $(m, n)-$family H of hash functions is a collection of functions that take as input the set of binary strings of length m and return as output the set of binary strings of length n [39], [40]. The advantage $\mathrm{Adv}^h_{\mathcal{A}}(t)$ of adversary $\mathcal{A}$ in finding a hash collision in execution time t is defined as $\mathrm{Adv}^h_{\mathcal{A}}(t) = Pr[(M_1, M_2) \in_R \mathcal{A} : M_1 \neq M_2, h(M_1) = h(M_2)]$, where $h \in_R H$ is a one-way collision-resistant hash function. The advantage of adversary attacking the collision resistance at most t times is negligible.

A property of some hash functions that simplifies their analysis is being linear relative to the bitwise exclusive-or operation [40]. This property is found in many natural constructions.

Definition 4: A family of functions H is $\oplus$-linear if for all M, M', we have $h(M \oplus M') = h(M) \oplus h(M')$.

Authentication Encryption [41], [42]: Briefly speaking, an authenticated encryption scheme transforms a message and a public header information in such a way which the ciphertext provides both integrity and privacy. Let AEnc=(KG,Enc,Dec) be a symmetric encryption scheme, where KG, Enc and Dec are the key generation algorithm, encryption algorithm, and decryption algorithm, respectively. The polynomial-time encryption algorithm $\mathsf{Enc} : \mathcal{K} \times \{0, 1\}^* \to \{0, 1\}^* \cup \{\perp\}$, and the polynomial-time decryption algorithm $\mathsf{Dec} : \mathcal{K} \times \{0, 1\}^* \to \{0, 1\}^* \cup \{\perp\}$ satisfy: for any $K \leftarrow \mathcal{K}$, any associated data H$\in \{0, 1\}^*$ and any message $m \in \{0, 1\}^*$, if $\mathsf{Enc}_K(H, m)$ outputs $c \neq \perp$, $\mathsf{Dec}_K(c)$ always outputs m.

III. System Model and Security Model

In this section, we describe the system model and security model to be used in the proposed symmetric PAKE for IoT.

A. System Model

As presented in Fig. 1, the system architecture in the strong symmetric PAKE for IoT is comprised of users and IoT device.

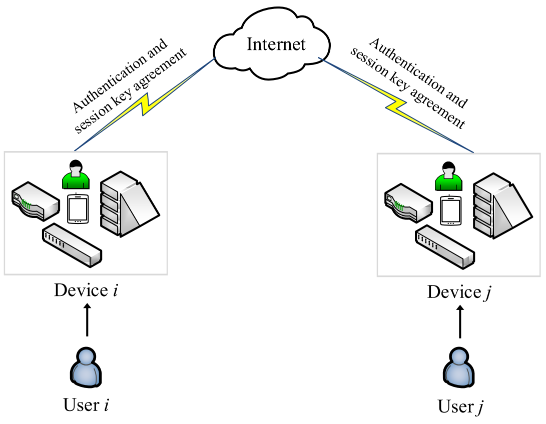
Internet
Authentication and session key agreement
Authentication and session key agreement
Device i
Device j
User i
User j

Fig. 1. System model of the proposed protocol.

We note that the protocol does not rely on a trusted third party. Specifically, users enter their own password and identity into the corresponding device. The device can be a personal computer, a smartphone or an access point, and so on. The device computes and stores the password file by using the password and identity. Then, the device engages in subsession with the intended partner to derive a shared session key while authenticating each other. The exchange messages between two parties over a public network do not reveal the participants' identity information.

B. Security Model

In this section, we introduce a new security model for our strong symmetric PAKE protocol, in which both parties are allowed to be corrupted. Our starting point is the security model proposed by Bellare, Pointcheval, and Rogaway [14], referred to as the BPR model. The BPR model is the most popular model for PAKE till now, which has been used in numerous protocols [17], [30], [43]. In most existing security model for PAKE [14], [44], they focus on key secrecy but without considering identity secrecy. Also, no previous PAKE security definition allows the adversary to corrupt honest parties.[1] Several models used in aPAKE [14], [45] only model the server corruption, thus do not deal with full forward secrecy.

Participants: In this model, we consider a list of pairs of matching client–client. We denote a client with an IoT device by U_A (resp., U_B) and the identity of U_A (resp., U_B) by ID_A (resp., ID_B). U_A (resp., U_B) holds a password file derived from its password pw_A (resp., pw_B), identity ID_A (resp., ID_B), and a secret random salt. For each matching pair (U_A, U_B), their passwords are equal, i.e., $pw_A = pw_B$. We assume that each password is chosen independently and uniformly from the set $\mathcal{D}$ for simplicity, where $\mathcal{D}$ denotes a general dictionary containing all strings.

Adversarial Abilities: The adversary denoted by $\mathcal{A}$ is a PPT with full control over all communication channels between participants. Specifically, $\mathcal{A}$ can read, intercept and modify all

[1]Corruption of honest parties is usually introduced in AKE protocols to model forward secrecy.

exchanged messages, and can forge new messages. The adversary is allowed to query the following oracles concurrently and in any order.

1) *Execute*(U_A^i, U_B^j)*:* This oracle models the passive eavesdropping attack. It executes the protocol between unused instances U_A^i and U_B^j, and returns the transcript of all messages sent over the network to the adversary.
2) *Send*(U_A^i, M)*:* This oracle models the active attack. It sends message M to instance U_A^i and returns the output of U_A^i to the adversary. If the adversary prompts an unused instance U_A^i to run the protocol with partner U_B by querying $\mathsf{Send}(U_A^i, U_B)$, the first protocol message of U_A^i is returned.
3) *Reveal*(U_A^i, U_B^j)*:* This oracle models the ability of adversary to obtain session keys. The adversary is given the session key established between U_A^i and its partner U_B^j.
4) *Corrupt*(U_A^i)*:* This oracle models the client U_A^i corruption. It returns the password file of the instance U_A^i to the adversary. The password file contains two values, where one is relative to the password and another is relative to the identity. The adversary can obtain the password file upon querying $\mathsf{Corrupt}(U_A^i)$, but cannot get any other internal state of U_A^i.
5) *Corrupt*(U_B^j)*:* This oracle models the client U_B corruption. The adversary can get the password file of the instance U_B^j from the oracle, but not any other internal data.
6) *Test*(U_A^i)*:* This oracle is allowed only to query fresh instances. It chooses a random bit $b \leftarrow_r \{0, 1\}$ at the beginning of the experiment. If the session key has not been defined or the instance U_A^i (or its partner) is not fresh, then the oracle returns the undefined symbol $\perp$. Otherwise, if $b = 0$, it returns a key chosen uniformly at random, whereas if $b = 1$, it returns the real session key. It is restricted that the adversary $\mathcal{A}$ is allowed to made only a single query to this oracle.

Partnering: Let session identifier (*sid*) be the partial transcript of the conversation between both instances before acceptance. We say that two instances U_A and U_B^j are partnered if they have the same session identifier which is not empty and they are in the accepted state.

Freshness: We say that an instance U_A^i is fresh if its state is accepted and the following conditions hold.

1) The adversary has not asked the query $\mathsf{Reveal}(U_A^i)$ to instance U_A^i.
2) The adversary has not asked the query $\mathsf{Reveal}(U_B^j)$ to instance U_B^j who is the partner of U_A^i.

Remark 1: It is not restricted that the adversary cannot make the Corrupt query in this model, which implies our model captures forward secrecy. In particular, even though the password file is compromised, the adversary cannot learn any information about the previously established session key by using that password file.

Definition 5 (PAKE Security): The adversary $\mathcal{A}$ is allowed to make any sequence of queries to Execute, Send, Corrupt, and Reveal oracles, but only to make Test query to fresh instances. The Send, Corrupt, Reveal, and Test queries are online attacks. We say that the adversary $\mathcal{A}$ wins if $\mathcal{A}$ outputs a guess b' for b and guesses correctly, i.e., $b' = b$. The advantage $\mathrm{Adv}_{\mathcal{A}}(\kappa)$ of adversary $\mathcal{A}$ in breaking the semantic security of a PAKE protocol is defined as $\mathrm{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot Pr[b' = b] - 1|$.

A PAKE protocol is secure if every PPT adversary making online attacks at most t times, it holds that $\mathrm{Adv}_{\mathcal{A}}(\kappa) \leq \varepsilon(\kappa, t) + \mathrm{negl}(\kappa)$ for some negligible function negl, where κ is the security parameter and ε is a function which is at most linear in t.

Definition 6 (Identity Privacy): The adversary makes Execute and Send queries, we say that a PAKE protocol achieves identity privacy if the transmission and password file cannot leak participants' identity information.

IV. Our Proposed Strong Symmetric PAKE

In this section, we present the construction of our symmetric PAKE protocol with identity privacy. We adopt the idea of the recent work [23] about disallowing the precomputation to construct a strong PAKE. Our construction is designed in the symmetric settings which protects both parties from compromise and precomputation attacks. And, the proposed protocol also provides identity privacy protection.

The protocol has two phases, which are the offline phase and interactive online phase between IoT devices. In the proposed scheme, the identities of participants are protected since the transmission does not leak any information of that. The detailed description of our protocol is as follows.

Let κ be a security parameter, the user identity $ID \in \{0, 1\}^\kappa$, $\mathbb{G}$ be a group of prime order q, and g be a generator of $\mathbb{G}$. Denote by $1_\mathbb{G}$ the identity element of $\mathbb{G}$, by $\mathbb{G}/1_\mathbb{G}$ the set of element of $\mathbb{G}$ except $1_\mathbb{G}$. Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^{p(\kappa)}$, $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, and $H_3 : \mathbb{G} \rightarrow \mathbb{Z}_q^*$ be cryptographic hash functions modeled as random oracles, where $p(\kappa)$ is a polynomial in κ and is the length of session key. And, let $H_2 : \mathbb{Z}_q^* \rightarrow \{0, 1\}^\kappa$ be $\oplus$-linear hash function.

A. Password File Derivation Phase

The IoT devices are allowed to take precomputation to obtain a password file. And, the password file generation is in offline phase. The process of password registration of IoT devices is given in Fig. 2.

Device U_A Registration: In this phase, the user enters a password pw_A and an identity ID_A into the device U_A. U_A picks a salt $s_A \leftarrow \mathbb{Z}_q^*$ randomly to randomize the password hash and identity. It then computes $hw_A = H_1(pw_A) \oplus s_A$ and $D_A = H_2(s_A) \oplus ID_A$. Then, U_A records a password file which includes hw_A and D_A.

Device U_B Registration: In this phase, U_B performs the same operations as device U_A. Finally, U_B records the password file <File $= (hw_B, D_B)$>.

B. Authenticated Key Exchange Phase

In this phase, the interactions between IoT devices are performed online. The protocol runs the following operations to realize mutual authentication and generate a common session

Password Registration (offline phase)

U_A on $(\text{StorePwdFile}, ID_A, S_B, pw_A)$	U_B on $(\text{StorePwdFile}, ID_B, U_A, pw_B)$
Pick a salt $s_A \leftarrow \mathbb{Z}_q^*$	Pick a salt $s_B \leftarrow \mathbb{Z}_q^*$
Compute $hw_A = H_1(pw_A) \oplus s_A$	Compute $hw_B = H_1(pw_B) \oplus s_B$
$D_A = H_2(s_A) \oplus ID_A$	$D_B = H_2(s_B) \oplus ID_B$
Record $< \text{File} = (hw_A, D_A) >$	Record $< \text{File} = (hw_B, D_B) >$

Fig. 2. Password registration of the proposed PAKE.

Key Exchange (online phase)

U_A		U_B
On input $(\text{Newsession}, T_A, S_B)$		On input $(\text{Newsession}, T_B, U_A)$
$r_A \leftarrow \mathbb{Z}_q^*$		$r_B \leftarrow \mathbb{Z}_q^*$
Retrieve $< \text{File} = (hw_A, D_A) >$		Retrieve $< \text{File} = (hw_B, D_B) >$
$X_A = H_2(hw_A) \oplus H_2(r_A)$		$X_B = H_2(hw_B) \oplus H_2(r_B)$
$Y_A = g^{r_A}$		$Y_B = g^{r_B}$
	$\xrightarrow{X_A, Y_A, T_A}$	If T_A is fresh, compute
		$tk_B = X_A \oplus D_B \oplus ID_B \oplus H_2(r_B) \oplus H_2(H_3(Y_A^{r_B}))$
		$C_B = Enc_{tk_B}(T_B, hw_B, r_B)$
	$\xleftarrow{X_B, Y_B, T_B, C_B}$	
If T_B is fresh, compute		
$tk_A = X_B \oplus D_A \oplus ID_A \oplus H_2(r_A) \oplus H_2(H_3(Y_B^{r_A}))$		
$Dec_{tk_A}(C_B) = (T_B, hw_B, r_B)$		
If T_B is valid,		
verify if $Y_B = g^{r_B} \wedge X_B = H_2(hw_B) \oplus H_2(r_B)$		
$C_A = Enc_{tk_A}(T_A, hw_A, r_A)$	$\xrightarrow{C_A}$	
$sid_A = X_A \parallel Y_A \parallel X_B \parallel Y_B$		$Dec_{tk_B}(C_A) = (T_A, hw_A, r_A)$
$K := H(tk_A \parallel sid_A)$		If T_A is valid,
		verify if $Y_A = g^{r_A} \wedge X_A = H_2(hw_A) \oplus H_2(r_A)$
		$sid_B = X_A \parallel Y_A \parallel X_B \parallel Y_B$
		$K := H(tk_B \parallel sid_B)$

Fig. 3. Construction of the proposed PAKE protocol.

key. The high-level overview of our symmetric PAKE is shown in Fig. 3.

1) *(NewSession,T_A, U_B):* The device U_A selects $r_A \leftarrow \mathbb{Z}_q^*$ and retrieves <File $= (hw_A, D_A)$>. It sets $X_A = H_2(hw_A) \oplus H_2(r_A)$ and $Y_A = g^{r_A}$. Then U_A sends (X_A, Y_A, T_A) to U_B, where T_A denotes the current timestamp.
2) *(NewSession,T_B, U_A):* After receiving (X_A, Y_A, T_A) from U_A, the device U_B selects $r_B \leftarrow \mathbb{Z}_q^*$ and retrieves <File $= (hw_B, D_B)$>. It computes $X_B = H_2(hw_B) \oplus H_2(r_B)$ and $Y_B = g^{r_B}$. S_B checks if the timestamp T_A is fresh, and aborts if not. It then sets $tk_B = X_A \oplus D_B \oplus ID_B \oplus H_2(r_B) \oplus H_2(H_3(Y_A^{r_B}))$ and computes $C_B = \text{Enc}_{tk_B}(T_B, hw_B, r_B)$. Finally, S_B sends (X_B, Y_B, T_B, C_B) to U_A.
3) Upon receiving (X_B, Y_B, T_B, C_B) from U_B, U_A first checks if the timestamp T_B is fresh. Next, it computes $tk_A = X_B \oplus D_A \oplus ID_A \oplus H_2(r_A) \oplus H_2(H_3(Y_B^{r_A}))$ and decrypts the ciphertext C_B. Then U_A verifies if T_B is valid, $Y_B = g^{r_B}$ and $X_B = H_2(hw_B) \oplus H_2(r_B)$. If it fails, then aborts; else, U_A computes $C_A = \text{Enc}_{tk_A}(T_A, hw_A, r_A,)$ and sends C_A to U_B. It sets $\text{sid}_A = X_A \| Y_A \| X_B \| Y_B$ and computes session key $K := H(tk_A \| \text{sid}_A)$. Then, it destroys the session state and outputs K.
4) After receiving C_A from U_A, U_B obtains (T_A, hw_A, r_A) by decrypting the ciphertext C_A. It verifies if T_A is valid, $Y_A = g^{r_A}$ and $X_A = H_2(hw_A) \oplus H_2(r_A)$. If it fails, then aborts; else, it sets $\text{sid}_B = X_A \| Y_A \| X_B \| Y_B$ and

computes session key $K := H(tk_B \| \text{sid}_B)$. Finally, U_B destroys the session state and outputs K. This completes the description of our protocol.

Correctness: It is easy to see that both parties can computes the same session key with overwhelming probability. From the property of hash function H_2, we have that $tk_A = X_B \oplus D_A \oplus ID_A \oplus H_2(r_A) \oplus H_2(H_3(Y_B^{r_A})) = H_2(H_1(pw_B)) \oplus H_2(s_B) \oplus H_2(s_A) \oplus H_2(r_B) \oplus H_2(r_A) \oplus H_2(H_3(g^{r_A r_B}))$ and $tk_B = X_A \oplus D_B \oplus ID_B \oplus H_2(r_B) \oplus H_2(H_3(Y_A^{r_B})) = H_2(H_1(pw_A)) \oplus H_2(s_A) \oplus H_2(s_B) \oplus H_2(r_A) \oplus H_2(r_B) \oplus H_2(H_3(g^{r_A r_B}))$. Thus, the correctness of our proposed protocol is trivially achieved.

V. Security Analysis

In this section, we analyze the security for symmetric PAKE with identity privacy. We first present the formal security proof of the scheme based on the security model described in Section III-B. What's more, we discuss other security properties.

A. Formal Security Proof

Theorem 1: Assume that the authenticated encryption scheme $\mathsf{AEnc} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is semantically secure and DLP, CDH assumptions hold, then the above-proposed scheme is a secure PAKE protocol.

Proof: We now give the formal proof composed of a sequence of games from the real security game G_0 to the random game G_9. The advantage of the adversary breaking the protocol's semantic security in game G_0 and G_9 will differ at most $\varepsilon(\kappa, t) + \text{negl}(\kappa)$. $Adv_{\mathcal{A},i}(\kappa)$ is used to indicate the adversary $\mathcal{A}$'s advantage in G_i.

Game G_0: This game corresponds to the real attack, where all oracle queries are answered honestly in accordance with protocol specifications. According to the security definition, we have $\text{Adv}_{\mathcal{A},0}(\kappa) = |2Pr[b' = b] - 1|$.

Game G_1: This game modifies the way $\mathsf{Execute}$ queries are answered. We replace hw_A with the value hw'_A chosen uniformly at random. The protocol runs honestly to generate X'_A. By the random oracle assumption, the value X_A and X'_A are indistinguishable. Then, we have $|\text{Adv}_{\mathcal{A},1}(\kappa) - \text{Adv}_{\mathcal{A},0}(\kappa)| \leq \text{negl}(\kappa)$.

Game G_2: This game modifies the way $\mathsf{Execute}$ queries are answered. We replace Y_A with the value r'_A chosen uniformly at random. The protocol runs honestly to generate Y'_A. By the DLP assumption, the value Y_A and Y'_A are indistinguishable. Then, we have $|\text{Adv}_{\mathcal{A},2}(\kappa) - \text{Adv}_{\mathcal{A},1}(\kappa)| \leq \text{negl}(\kappa)$.

Game G_3: This game further changes the way $\mathsf{Execute}$ queries are answered. The ciphertext C_B is replaced with an encryption of hw'_B. If the PPT adversary can distinguish game G_3 from game G_2 with nonnegligible adversary, then the semantic security of authenticated encryption will be broken with the same advantage. Thus, we have $|\text{Adv}_{\mathcal{A},3}(\kappa) - \text{Adv}_{\mathcal{A},2}(\kappa)| \leq \text{negl}(\kappa)$.

According to the message that may be sent as part of the protocol, we separate Send queries into four types.

1) $\mathsf{Send}_0(U_A^i, U_B^j)$ query enables the adversary to prompt unused instance U_A^i to run the protocol with U_B^j, and returns the first message $\text{msg}_1 = (X_A, Y_A, T_A)$ to $\mathcal{A}$.
2) $\mathsf{Send}_1(U_B^j, \text{msg}_1)$ query enables the adversary $\mathcal{A}$ to send the first flow $\text{msg}_1 = (X_A, Y_A, T_A)$ to an unused instance U_B^j, and returns the second flow to $\mathcal{A}$ if U_B^j accepts msg_1 as a valid message.
3) $\mathsf{Send}_2(U_A^i, \text{msg}_2)$ query enables the adversary $\mathcal{A}$ to send the second flow $\text{msg}_2 = (X_B, Y_B, T_B, C_B)$ to instance U_A^i, and returns the third flow to $\mathcal{A}$ if U_A^i accepts msg_2 as a valid message.
4) $\mathsf{Send}_3(U_B^j, \text{msg}_3)$ query enables the adversary $\mathcal{A}$ to send the last flow $\text{msg}_3 = (C_A)$ to instance U_B^j, and returns nothing but sets session key K.

Game G_4: This game modifies the way Send_1 queries are answered. If msg_1 was output by a previous Send_0 query, the game G_4 is observed to be indistinguishable from G_3; otherwise, we replace tk_B with a random value chosen from the uniform distribution. Thus, the modification in this game can only introduce negligible statistical difference. We have $\text{Adv}_{\mathcal{A},3}(\kappa) \leq \text{Adv}_{\mathcal{A},4}(\kappa) + \text{negl}(\kappa)$.

Game G_5: This game modifies the way Send_2 queries are answered. If msg_2 was output by a previous Send_1 query, the game G_5 is observed to be indistinguishable from G_4; otherwise, we force the instance U_A^i to hold the same value tk_A as the value tk_B of the instance U_B^j. Since the secret tk_A is hidden from the adversary, this modification does not change the adversary's view. We have $\text{Adv}_{\mathcal{A},5} = \text{Adv}_{\mathcal{A},4}$.

Game G_6: This game modifies the way Send_0 queries are answered. We replace X_A with X'_A which is derived from hw'_A chosen uniformly at random. By the random oracle assumption, the adversary can distinguish game G_6 from G_5 with negligible advantage. We have $|\text{Adv}_{\mathcal{A},6}(\kappa) - \text{Adv}_{\mathcal{A},5}(\kappa)| \leq \text{negl}(\kappa)$.

Game G_7: This game modifies the way Send_3 queries are answered. The ciphertext C_A is replaced with an encryption of hw_A. If C_A is a valid ciphertext, sets the session key identical to that of U_B^j; else, sets the session key as an uniformly chosen element from $\{0, 1\}^{p(\kappa)}$. Under the semantic security of authenticated encryption, we have $|\text{Adv}_{\mathcal{A},7}(\kappa) - \text{Adv}_{\mathcal{A},6}(\kappa)| \leq \text{negl}(\kappa)$.

Game G_8: This game modifies the way $\mathsf{Corrupt}(U_A^i)$ are answered. We show that the adversary can distinguish a real session key from a random number if the following situations occurs.

1) If the query $\mathsf{Corrupt}(U_A^i)$ is made but subsequently no $\mathsf{Send}(U_B^j)$ query is made, the adversary successfully forges a valid message C_A. The adversary obtains the password file which contains the value hw_A and D_A after asking $\mathsf{Corrupt}(U_A^i)$ query. By the DLP, CDH assumptions, and collision resistance of hash functions, the adversary cannot get identity information and any data about r_A from X_A, hw_A, D_A, and Y_A. Thus, the adversary who correctly calculate the value tk_A with negligible advantage can forge a valid ciphertext with the same advantage.
2) If the query $\mathsf{Corrupt}(U_A^i)$ is made and subsequently $\mathsf{Send}(U_B^j)$ query is made, the adversary successfully forges a valid message C_A. When the adversary obtains the password file of U_A^i after making $\mathsf{Corrupt}$ query, it

sets r'_A and subsequently asks $\mathsf{Send}(U^j_B, \text{msg}'_1)$ query to get a valid message msg'_2. To forge a valid C_A, $\mathcal{A}$ must correctly calculate the value $tk'_A = X_B \oplus D_A \oplus ID_A \oplus H_2(r'_A) \oplus H_2(H_3(Y_B^{r'_A}))$. If the adversary $\mathcal{A}$ executes $t-1$ times Send queries for guessing ID_A, the probability that $\mathcal{A}$ outputs a valid C_A is bounded by $(t-1)/2^\kappa$.

Therefore, we have $|\text{Adv}_{\mathcal{A},8}(\kappa) - \text{Adv}_{\mathcal{A},7}(\kappa)| \le (t-1)/2^\kappa + \text{negl}(\kappa)$.

Game G_9: This game modifies the way $\mathsf{Corrupt}(U^j_B)$ queries are answered. The query $\mathsf{Corrupt}(U^j_B)$ is asked and subsequently the $\mathsf{Send}(U^i_A)$ query is made. We show that the adversary can distinguish a real session key and a random number if $\mathcal{A}$ successfully forges a valid ciphertext C_B. When the adversary obtains the password file of U^j_B which includes hw_B and D_B after querying $\mathsf{Corrupt}(U^j_B)$, it subsequently asks $\mathsf{Send}(U^i_A, U^j_B)$ query to get a valid message $\text{msg}_1 = (X_A, Y_A, T_A)$. Then, $\mathcal{A}$ sets r'_B and compute $tk'_B = X_A \oplus D_B \oplus ID_B \oplus H_2(r'_B) \oplus H_2(H_3(Y_A^{r'_B}))$. To forge a valid C_B, $\mathcal{A}$ must correctly calculate the value tk'_B, and then it needs to know the identity ID_B. If the adversary $\mathcal{A}$ executes $t-1$ times Send queries for guessing ID_B, the probability that $\mathcal{A}$ outputs a valid C_B is bounded by $(t-1)/2^\kappa$. Thus, we have $|\text{Adv}_{\mathcal{A},9}(\kappa) - \text{Adv}_{\mathcal{A},8}(\kappa)| \le \text{negl}(\kappa)$.

If the adversary $\mathcal{A}$ only makes online attacks at most t times, we have $\text{Adv}_{\mathcal{A},9}(\kappa) \le \varepsilon(\kappa, t) + \text{negl}(\kappa)$. In all, we obtain that $\text{Adv}_{\mathcal{A},0}(\kappa) \le \varepsilon(\kappa, t) + \text{negl}(\kappa)$ from game G_1 to game G_9. This ends the proof of Theorem 1.

Theorem 2: In the proposed scheme, the transmitted data contains no identity information of participants, and the password file cannot leak the identity information.

Proof: First, the adversary makes $\mathsf{Execute}$ queries, the transcripts of all sent messages are not related to the identity information. Second, the adversary can obtain the value $D_A = H_2(s_A) \oplus ID_A$ or $D_B = H_2(s_B) \oplus ID_B$ by asking the $\mathsf{Corrupt}$ query. However, $\mathcal{A}$ can not get any identity information without the knowledge of the salt s_A or s_B. Therefore, the transmission contains no participants' identity information, and the password file cannot leak that information. In conclusion, the proposed symmetric PAKE achieves identity privacy protection.

B. Other Security Analysis

The symmetric PAKE protocol also meets other security properties which are discussed nonmathematically. The following list descriptions of these security properties.

Off-Line Password Guessing Attacks: The adversary $\mathcal{A}$ can get the transcripts during previous protocol executions, such as X_A, in which one of the password file values hw_A involved. However, because of unawareness of the ephemeral secret r_A and the property of the one-way collision-resistance hash function, the value hw_A is preserved. Thus, it is computationally infeasible to guess the password pw_A for the adversary.

Replay Attack: There is a timestamp with a short valid period in the message. Due to freshness of timestamps (T_A, T_B), the proposed protocol is immune to a replay attack through the verification of message validity.

Precomputation Attack: Namely, the adversary can precompute a table of values for a dictionary of the most probable passwords, so it can learn a user's password instantly upon device compromise. In our protocol, the value hw in the password file is derived from the password hash and salt. Since the random private salt s is never revealed, precomputing values for a dictionary of the passwords is disallowed.

Compromise Impersonation Attack: Although the password file of one entity is leaked, the adversary cannot disguise himself as other entities to participant in the key exchange. Therefore, our proposed scheme can defend against compromise impersonation.

Perfect Forward Secrecy: Even though the password files of all participants are exposed, the security of the established session key in the previous protocol will not be impacted because of without knowledge of ephemeral secret r and identity, or without knowledge of a true password and r. Therefore, we conclude that our protocol enjoys full forward secrecy.

Mutual Authentication: Two parties authenticate each other by verifying the decryption validity of ciphertexts (C_A, C_B) they received. The plaintext corresponding to the ciphertext contains partial data of the password file and ephemeral random secret. If the verification is successful, it indicates that the instance sending the message is honest. Therefore, this proposed scheme realizes mutual authentication between the two parties.

VI. Performance Evaluation

In this section, we first present a comparison about security features between our proposed scheme and several related AKE schemes [23], [25], [27], [37], [38], where the schemes in [23], [25], [37], and [38] are password-based AKEs. Moreover, we analyze communication and storage cost and computational efficiency of our PAKE scheme and the schemes [23], [25], [27], [37], [38], and compare with the password file derivation time and key exchange time of these schemes through experiments.

A. Comparison of Functionality

We compare the proposed PAKE protocol with other PAKE protocols [23], [25], [27], [37], [38] considering the set of security and functionality features in Table I. Obviously, only our PAKE protocol satisfies all features listing in the table. Except for the protocol in [27], the rest of protocols can defend against off-line password guessing attack. The protocols in [23], [37], and [38] and our protocol are immune to replay attack. All protocols in the table meet compromise resilience. However, only the proposed scheme and the schemes in [27] and [38] are resilient to compromise for both parties, and the schemes in [23], [25], and [37] are only immune to server compromise. In those protocols [23], [25], [27], [37], [38] which are resilient to compromise, the aPAKE protocol in [25] does not resist precomputation attack, and the protocol in [27] is not related to precomputation attack. In addition, only our protocol and the protocol in [27] satisfy full forward secrecy. From the comparison result, other protocols [23], [25], [27], [37], [38] except for our proposed protocol cannot provide identity privacy protection. Besides,

TABLE I
COMPARISON OF SECURITY AND FUNCTIONALITY WITH OTHER PAKE PROPOSALS

Security and Functionality features	Protocol [23]	Protocol [27]	Protocol [37]	Protocol [25]	Protocol [38]	Ours
Off-line password guessing attack	✓	-	✓	✓	✓	✓
Replay attack	✓	×	✓	×	✓	✓
Compromise resilience for server	✓	✓	✓	✓	✓	✓
Compromise resilience for both parties	×	✓	×	×	✓	✓
Pre-computation attack	✓	-	✓	×	✓	✓
Mutual authentication	✓	✓	×	✓	×	✓
Full forward secrecy	×	✓	×	×	×	✓
Identity privacy	×	×	×	×	×	✓

*Note:"✓": Feature is satisfied by the protocol; " × ": Feature is not satisfied by the protocol; "-": Feature is not related to the protocol.

TABLE II
COMPARISON OF COMMUNICATION AND STORAGE COST

Protocol	Communication cost (bit)	Storage cost (bit)
Protocol [23]	2240	1216
Protocol [25]	2176	640
Protocol [27]	2112	1280
Protocol [37]	2752	384
Protocol [38]	1600	1024
Our protocol	1728	416

the protocols in [37] and [38] fail to meet explicit mutual authentication. In conclusion, our symmetric PAKE protocol is superior to other related protocol in terms of security.

B. Communication and Storage Cost Comparison and Analysis

We compare the communication and storage costs of our PAKE scheme with those of the related schemes [23], [25], [27], [37], [38]. The comparison results are shown in Table II. We assume that the random number/nonce is 64 bits, the identity is 160 bits, and the timestamp is 32 bits. The output of hash functions H_1 and H_3 is 160 bits each. In addition, the outputs of hash functions H_2 and H are 256 and 512 bits, respectively. Let $G_1 = <g>$ and G_2 denote two groups of the prime order p. $g_T = e(g, g) \in G_2$ is a generator of G_2, where $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map. The elements in group G_1 and G_2 are 320 and 512 bits, respectively.

In protocol [23], the exchange message is $\{\alpha, X_u, \beta, X_s, c, f_k(1), f_k(2)\}$. The stored information contains $\{k_s, p_s, P_s, P_u, c\}$. Therefore, the total communication cost and the storage cost of protocol in [23] are $320 + 320 + 320 + 320 + 448 + 256 + 256 = 2240$ bits and $64 + 64 + 320 + 320 + 448 = 1216$ bits, respectively. In protocol [25], the exchange message is $\{ID_U, ID_S, Z, \text{salt}, ID_U, T = (s, r, c, g_1^r), ID_S, AC\}$. The stored information contains $\{D, \text{salt}, P\}$. Therefore, the total communication cost and the storage cost of protocol in [25] are $160 + 160 + 320 + 64 + 160 + (64 + 256 + 256 + 320) + 160 + 256 = 2176$ bits and $256 + 64 + 320 = 640$ bits, respectively. In protocol [27], the exchange message is $\{X, N_C, ID_C, \text{Auth}_S, N_S, Y_1, Y_2, ID_S, \text{Auth}_C\}$. The stored information contains $\{U_{S,j,1}, U_{S,j,2}, X_{S,j,1}, X_{S,j,2}\}$. Therefore, the total communication cost and the storage cost of protocol in [27] are $320 + 64 + 160 + 256 + 64 + 320 + 512 + 160 + 256 = 2112$ bits and $320 + 320 + 320 + 320 = 1280$ bits, respectively. In protocol [37], the exchange message is $\{c, d, hp, e, \tau\}$. The stored information contains $\{s, z\}$. Therefore, the total communication cost and the storage cost of protocol in [37] are $320 + 320 + (320 + 320) + (64 \times 4 + 320 \times 3) + 256 = 2752$ bits and $64 + 320 = 384$ bits, respectively. In protocol [38], the exchange message is $\{id_i, A_i, C_i, id_j, A_j, C_j\}$. The stored information contains $\{id, A, B, C\}$. Therefore, the total communication cost and the storage cost of protocol in [38] are $160 + 320 + 320 + 160 + 320 + 320 = 1600$ bits and $64 + 320 \times 3 = 1024$ bits, respectively. In our protocol, the exchange message is $\{X_A, Y_A, T_A, X_B, Y_B, T_B, C_B, C_A\}$. The stored information contains $\{hw, D\}$. Therefore, the total communication cost and the storage cost of our protocol are $256 + 320 + 32 + 256 + 320 + 32 + 256 + 256 = 1728$ bits and $160 + 256 = 416$ bits, respectively.

As the results of Table II imply, the communication cost of scheme [38] and the storage cost of scheme [37] are better than ours, respectively. However, the storage cost of scheme [38] and the communication cost of scheme [37] are much more expensive than ours, respectively. In addition, they cannot fully assuage the required security properties.

C. Computation Cost Comparison and Analysis

This section presents the computation comparison and analysis of the proposed scheme and other schemes [23], [25], [27], [37], [38], where the schemes in [23], [37], and [38] enjoy precomputation resistance and compromise resilience, and the schemes in [25] and [27] are designed for the IoT environment. We also give the time consumption for those protocols through experiments.

Table III lists the computation cost in the password file derivation phase and the authenticated key exchange phase for sender and receiver, and shows the comparison between our scheme and the related schemes in [23], [25], [27], [37], and [38].

In the computational cost analysis, the execution cost of the exclusive-or operation is ignored as it incurred very small costs compared to other operations. $\oplus$-linear hash operation cost is the same as a general hash operation, so we denote it as a general hash. For the password file generation phase of our protocol, it needs to perform two hash operations. For the

TABLE III
COMPARISON OF COMPUTATION COST

Protocol	Password file/Key derivation	Authenticated key exchange		
		Sender	Receiver	Total
Protocol [23]	$3Exp + 1MTP + 1Hash$	$5.16Exp + 1MTP + 4Hash + 1Inv$	$4.16Exp + 7Hash$	$9.32Exp + 1MTP + 11Hash + 1Inv$
Protocol [25]	$1Exp + 1Hash$	$3Exp + 6Hash + 2Inv$	$4.5Exp + 3Hash$	$7.5Exp + 9Hash + 2Inv$
Protocol [27]	$2Exp$	$7Exp + 2Hash$	$6Exp + 2Hash + 2P$	$13Exp + 4Hash + 2P$
Protocol [37]	$1Exp + 1Hash + 1Inv$	$14Exp + 3Hash + 1Inv$	$9Exp + 2Hash$	$23Exp + 5Hash + 1Inv$
Protocol [38]	$3Exp + 2MTP$	$3Exp + 1MTP + 3P$	$3Exp + 1MTP + 3P$	$6Exp + 2MTP + 6P$
Ours	$2Hash$	$3Exp + 7Hash$	$3Exp + 7Hash$	$6Exp + 14Hash$

*Note: Exp denotes modular exponentiation in $\mathbb{G}$; MTP denotes map-to-point hash in $\mathbb{G}$; $Hash$ denotes general hash operation; P denotes bilinear pairing operation in $\mathbb{G} \times \mathbb{G}_1$; Inv denotes modular inverse operation.

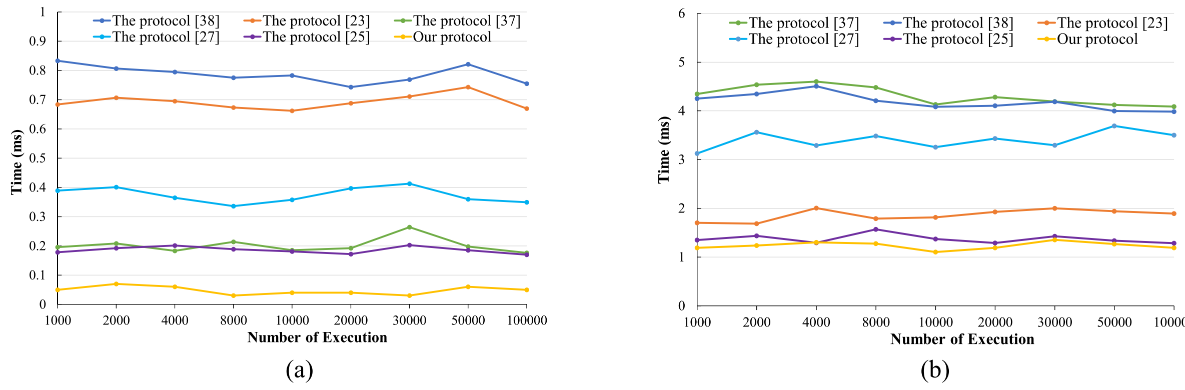

Fig. 4. Time consumption in offline phase and online phase. (a) Password registration. (b) Session key agreement.

key exchange phase, both parties need six modular exponentiation and fourteen general hash operations totally. Compared with other protocols in [23], [25], [27], [37], and [38], the computational overhead of our proposed scheme in the key exchange phase is just a little lower than [23], [25]. However, the schemes in [23] and [25] do not provide compromise resilience for both parties, and the scheme in [25] is vulnerable to the precomputation attack. On the whole, our protocol's computation cost in total is the best one.

We make some experiments for the proposed protocol and other protocols [23], [25], [27], [37], [38] to show the efficiency of our protocol. The experiments are done on the platform of AMD Ryzen 7 4800H with Radeon Graphics 2.90-GHz CPU at 8-GB RAM running on the Ubuntu 20.04.2 LTS operation system. To better compare those schemes, we implemented the schemes based on the same pairing specified in the MCL library. Our program uses C++ language, and uses the MCL library and OPENSSL library.

The computation time of our scheme and other schemes [23], [25], [27], [37], [38] is presented in Fig. 4. Fig. 4(a) and (b), respectively, show the running time under different execution times in the password file generation phase and the authenticated exchange phase. Table IV gives the average time consumption of password file derivation and key exchange by executing the protocol 10 000 iterations. As shown in Fig. 4 and Table IV, our PAKE protocol takes less time than other five protocols whether in the password registration phase or in the authenticated key exchange phase.

To sum up, the proposed protocol is more secure and efficient than the existing PAKE protocols.

TABLE IV
TIME CONSUMPTION

Protocol	Password file derivation (us)	Session key agreement (us)
Protocol [23]	662	1817
Protocol [25]	181	1369
Protocol [27]	358	3257
Protocol [37]	185	4131
Protocol [38]	783	4086
Our protocol	4	1102

VII. CONCLUSION

This article proposed an efficient and strong PAKE between IoT devices in the IoT environment. The proposed protocol is suitable for symmetric settings. It protects both parties from compromise attack and precomputation attack simultaneously. In this scheme, it only requires a few exponentiations and no pairing operation. Furthermore, we give a new security model for our scheme which captures forward secrecy and identity privacy protection. Finally, through performance evaluation, we demonstrate that our proposed PAKE scheme has great advantages in terms of the total computation cost and security properties.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, 2010.
[2] S. Mumtaz, A. Alsohaily, Z. Pang, A. Rayes, K. F. Tsang, and J. Rodriguez, "Massive Internet of Things for industrial applications: Addressing wireless IIoT connectivity challenges and ecosystem fragmentation," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 28–33, Mar. 2017.

[3] "Global Mobile Trends 2021." 2020. [Online]. Available: https://www.gsma.com/globalmobiletrends/

[4] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, Sep. 2014.

[5] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. S. Wong, "New algorithms for secure outsourcing of large-scale systems of linear equations," *IEEE Trans. Inf. Forensics Security*, vol. 10, pp. 69–78, 2015.

[6] Y. Yu et al., "Identitybased remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 767–778, 2017.

[7] L. Zhang, "OTIBAAGKA: A new security tool for cryptographic mix-zone establishment in vehicular ad hoc networks," *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 2998–3010, 2017.

[8] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *J. Inf. Security Appl.*, vol. 38, pp. 8–27. Feb. 2018.

[9] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of IoT systems: Design challenges and opportunities," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design*, 2014, pp. 417–423.

[10] X. Li, J. Niu, M. Z. A. Bhuiyan, F. Wu, M. Karuppiah, and S. Kumari, "A robust ECC based provable secure authentication protocol with privacy protection for Industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3599–3609, Aug. 2018.

[11] D. H. Lee and I. Y. Lee, "A lightweight authentication and key agreement schemes for IoT environments," *Sensors*, vol. 20, no. 18, p. 5350. 2020.

[12] Q. Zhang et al., "A group key agreement protocol for intelligent Internet of Things system," *Int. J. Intell. Syst.*, vol. 37, no. 1, pp. 699–722, 2022.

[13] M. Abdalla and D. Pointcheval, "Simple password-based encrypted key exchange protocols," in *Proc. CT-RSA*, 2005, pp. 191–208.

[14] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attack," in *Proc. EUROCRYPT*, 2000, pp. 139–155.

[15] E. Bresson, O. Chevassut, and D. Pointcheval, "Security proofs for an efficient password-based key exchange," in *Proc. CCS*, 2003, pp. 241–250.

[16] E. Bresson, O. Chevassut, and D. Pointcheval, "New security results on encrypted key exchange," in *Proc. PKC*, 2004, pp. 145–158.

[17] J. Katz, R. Ostrovsky, and M. Yung, "Efficient password-authenticated key exchange using human-memorable passwords," in *Proc. EUROCRYPT*, 2001, pp. 475–494.

[18] M. I. G. Vasco, A. L. P. D. Pozo, and C. Soriente, "A Key for John Doe: Modeling and designing anonymous password-authenticated key exchange protocols," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1336–1353, May/Jun. 2021.

[19] P. D. MacKenzie, S. Patel, and R. Swaminathan, "Password-authenticated key exchange based on RSA," in *Proc. ASIACRYPT*, 2000, pp. 599–613.

[20] F. Benhamouda and D. Pointcheval, "Verifier-based password-authenticated key exchange: New models and constructions," IACR Cryptol. ePrint Archive, Lyon, France, Rep. 2013/833, 2013.

[21] J. Y. Hwang, S. Jarecki, T. Kwon, J. Lee, J. S. Shin, and J. Xu, "Round-reduced modular construction of asymmetric password-authenticated key exchange," in *Proc. SCN*, 2018, pp. 485–504.

[22] C. S. Jutla and A. Roy, "Smooth NIZK arguments," in *Proc. TCC*, 2018, pp. 235–262.

[23] S. Jarecki, H. Krawczyk, and J. Xu, "OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks," in *Proc. EUROCRYPT*, 2018, pp. 1–61.

[24] M. S. Farash, M. Turkanović, S. Kumari, and M. Hölbl, "An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the Internet of Things environment," *Ad Hoc Netw.*, vol. 36, pp. 152–176, Jan. 2016.

[25] V. H. Hoang, E. Lehtihet and Y. Ghamri-Doudane, "Password-based authenticated key exchange based on signcryption for the Internet of Things," in *Proc. Wireless Days (WD)*, 2019, pp. 1–8.

[26] J. Srinivas, A. K. Das, M. Wazid, and N. Kumar, "Anonymous lightweight chaotic map-based authenticated key agreement protocol for Industrial Internet of Things," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 6, pp. 1133–1146, Nov./Dec. 2020.

[27] A. L. Peng, Y. M. Tseng, and S. S. Huang, "An efficient leakage-resilient authenticated key exchange protocol suitable for IoT devices," *IEEE Syst. J.*, vol. 15, no. 4, pp. 5343–5354, Dec. 2021.

[28] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *Proc. IEEE Comput. Soc. Symp. Res. Security Privacy*, 1992, pp. 72–84.

[29] S. Jiang, G. Gong, J. He, K. Nguyen, and H.Wang, "PAKEs: New framework, new techniques and more efficient lattice-based constructions in the standard model," in *Proc. PKC*, 2020, pp. 396–427.

[30] J. Zhang and Y. Yu, "Two-round PAKE from approximate SPH and instantiations from lattices," in *Proc. ASIACRYPT*, 2017, pp. 37–67.

[31] Z. Li and D. Wang, "Achieving one-round password-based authenticated key exchange over lattices," *IEEE Trans. Services Comput.*, vol. 15, no. 1, pp. 308–321, Jan./Feb. 2022, doi: 10.1109/TSC.2019.2939836.

[32] S. M. Bellovin and M. Merritt, "Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise," in *Proc. CCS*, 1993, pp. 244–250.

[33] F. Kiefer and M. Manulis, "Zero-knowledge password policy checks and verifier-based PAKE," in *Proc. ESORICS*, 2014, pp. 295–312.

[34] D. Pointcheval and G. Wang, "VTBPEKE: Verifier-based two-basis password exponential key exchange," in *Proc. AsiaCCS*, 2017, pp. 1–18.

[35] D. P. Jablon, "Strong password-only authenticated key exchange," *SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 5, pp. 5–26, Oct. 1996.

[36] S. J. Sun, J. Minjae, H. J. Yeon, and L. Jaehwan, "A verifier-based password-authenticated key exchange using tamper-proof hardware," *Comput. J.*, vol. 64, no. 8, pp. 1293–1302, 2021.

[37] T. Bradley, S. Jarecki, and J. Xu, "Strong asymmetric PAKE based on trapdoor CKEM," in *Proc. CRYPTO*, 2019, pp. 798–825.

[38] M. Naor, S. Paz, and E. Ronen, "CRISP: Compromise resilient identity-based symmetric PAKE," IACR Cryptol. ePrint Arch., Lyon, France, Rep. 2020/529, 2020.

[39] D. R. Stinson, "Some observations on the theory of cryptographic hash functions," *Designs, Codes Cryptogr.*, vol. 38, no. 2, pp. 259–277, Feb. 2006.

[40] H. Krawczyk, "LFSR-based hashing and authentication," in *Proc. CRYPTO*, 1994, pp. 129–139.

[41] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *Proc. ASIACRYPT*, 2000, pp. 531–545.

[42] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," *J. Cryptol.*, vol. 21, no. 4, pp. 469–491, 2008.

[43] J. Katz and V. Vaikuntanathan, "Round-optimal password-based authenticated key exchange," in *Proc. TCC*, 2011, pp. 293–310.

[44] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Proc. CRYPTO*, vol. 773, 1993, pp. 232–249.

[45] M. Abdalla, P.-A. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *Proc. PKC*, 2005, pp. 65–84.

Huanhuan Lian is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Fudan University, Shanghai, China.

Her main research interests include cryptography and information security, in particular, key exchange and lattice cryptography.

Yafang Yang received the M.S. degree from the College of Mathematics and Statistics, Chongqing University, Chongqing, China, in 2018. She is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Fudan University, Shanghai, China.

Her research interests include information security, (certificateless) public key cryptography, applied cryptography, and VANET security.

Yunlei Zhao received the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2004.

He joined Hewlett-Packard European Research Center, Bristol, U.K., as a Postdoctoral Researcher, in 2004. Since 2005, he has been with Fudan University, where he is currently a Distinguished Professor with the School of Computer Science. His research interests include the theory and applications of cryptography.