IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 24, 2017, accepted November 17, 2017, date of publication November 21, 2017,
date of current version December 22, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2776160

Provably Leakage-Resilient Password-Based Authenticated Key Exchange in the Standard Model

OU RUAN[1], JING CHEN[2], AND MINGWU ZHANG[1]
[1]School of Computer Science, Hubei University of Technology, Wuhan 430068, China
[2]School of Computer Science, Wuhan University, Wuhan 430072, China

Corresponding author: Ou Ruan (ruanou@163.com)

This work was supported in part by the Educational Commission of Hubei Province of China under Grant D20151401, in part by the Natural Science Foundation of Hubei Province of China under Grant 2017CFB596, and in part by the Green Industry Technology Leading Project of Hubei University of Technology under Grant ZZTS2017006.

ABSTRACT The password-based authenticated key exchange (PAKE) protocol is one of most practical cryptographic primitives for trusted computing, which is used to securely authenticate devices' identities and generate shared session keys among devices in insecure environments by using a short, human-memorable password. With the fast development of the Internet of Things (IoT), new challenges regarding PAKE have emerged. The traditional PAKE protocols are completely insecure in IoT environments, since there are many kinds of side-channel attacks. Therefore, it is very important to model and design leakage-resilient (LR) PAKE protocols. However, there has been no prior work on modeling and constructing LR PAKE protocols. In this paper, we first formalize an LR eCK security model for PAKE based on the eCK-secure PAKE model and the only computation leakage model. Then, we propose the first LR PAKE protocol by using Diffie–Hellman key exchange, LR storage (LRS) and LR refreshing of LRS appropriately and formally present a security proof in the standard model.

INDEX TERMS Leakage-resilience, password-based authenticated key exchange, side-channel attacks, trusted computing, internet of things.

I. INTRODUCTION

Trusted computing (TC) is a very important technology, which aims to enhance the overall security, privacy and trustworthiness of a variety of computing devices. Trustworthy software assurance, trusted execution environment and trusted collaboration are hot research fields of TC. Establishing and protecting device identity by trustworthy software assurance make up one of the fundamental security requirements of TC. Authenticated key exchange (AKE) protocols [1] provide a good solution for identity authentication, which can securely authenticate device identity and generate a shared session key among devices in an insecure environment. With the fast development of the Internet of Things (IoT), new challenges regarding TC have emerged. There are many IoT devices that vary widely in their cost, usage, and capabilities. For the trusted execution environment, IoT devices should have the ability to perform mutual authentication with IoT services or with other IoT devices [2]. AKE is one of main technologies for performing mutual authentication and session key generation for IoT devices. For example, Hsu *et al.* [3] designed an AKE protocol for wearable devices in IoT environments. Among AKEs, the password-based AKE (PAKE) protocols are most widely used since no additional device is required, just a human-memorable password for authenticating the parties. The concept of PAKE was introduced by Bellovin and Merritt [4]. The first provably secure PAKE protocols were proposed by Bellare *et al.* [5] and MacKenzie *et al.* [6]; they formally proved the security in the random oracle (RO) model. Then, Byun *et al.* [7] and Mohammad and Mahmoud [8] improved and generalized the constructions of PAKE in the RO model. In 2006, the first provably secure PAKE protocol in the standard model was shown by Goldreich and Lindell [9], and then [10]–[13] showed efficient constructions for PAKE protocols in the standard model. In 2012, Wen *et al.* [14] introduced a three-party password-authenticated multiple key exchange protocol. Subsequently, Tsai *et al.* [15] made some improvements to [14]. However, Luo *et al.* [16] demonstrated off-line password guessing attacks against both protocols. Recently, Ruan *et al.* [17] designed an explicit PAKE protocol with mutual key confirmation and gave a formal security proof; Yi [18] et al. presented a two-server PAKE protocol, in which two servers know only partial information about the client's password, but can cooperate to authenticate the client's identity; Islam [19], Amin and Biswas [20] and Lu [21] designed three-party/multi-party PAKE protocols with formal security proofs; Nam *et al.* [22] and Guo *et al.* [23] proposed provably secure group PAKE protocols.

Computations or communications of IoT devices emit signals known as "side channels", such as electromagnetic emissions and power consumption. Most IoT devices are exposed to the public outside, and an attacker can overcome the security protections by measuring these signals, which are called side-channel attacks [24]. Traditional PAKEs are completely insecure in leakage environments. Moreover, the technologies of side channel attacks are developing, and new attack methods may appear at any moment. Thus, it is impossible to consider all types of side-channel attacks in the hardware design of IoT devices. Furthermore, we know that current TC technologies focus on establishing trust, but how to maintain trust in dynamically changing environments has not been deeply studied. Thus, to resist side-channel attacks and provide trustworthy software assurance, it is very important to model and design leakage-resilient (LR) AKE protocols.

The first LR security model for AKEs was introduced by Moriyama and Okamoto [25] and is called the MO model. The MO model was based on the eCK security model [26], which is an extension of the CK security model [27]. The adversary of the eCK security model has more power than the CK model and can access both the long-term secret key and the ephemeral secret randomness of the test session. The central limitation of the MO model is that the leakages are only allowed until the adversary learns the challenge. Leakage that occurs after the adversary learns the challenge is called after-the-fact (AF) leakage. The first AFLR CK security model and the first continuous AFLR (CAFLR) AKE protocol were introduced by Alawatugoda *et al.* [28]. Then, the first AFLR eCK security model and the first bounded AFLR (BAFLR) AKE protocol were proposed by Alawatugoda *et al.* [29], and the first CAFLR eCK-secure AKE protocol was introduced by Alawatugoda *et al.* [30]. In 2016, Chen *et al.* [31] first considered leakage attacks on both the long-term secret private key and the ephemeral secret randomness, and proposed a one-round AFLR AKE protocol under this strong security model. In 2017, the first ID-based BAFLR AKE protocol was introduced by Ruan *et al.* [32]. Recently, Toorani [33] demonstrated an ephemeral key compromise impersonation (KCI) attack on the construction of [28]; Yang and Li [34] also showed that the construction of [29] was insecure against KCI attacks and the proofs of Case 2 (the adversary is active) were incorrectly reduced to the Decision Diffie-Hellman (DDH) assumption, and then they improved the construction and formally showed the security proof under the Gap Diffie-Hellman (GDH) assumption in the RO model.

In this paper, we formalize the LR eCK security model for PAKE and propose an LR PAKE protocol that is based on the key derivation function (KDF) [35], leakage-resilient storage (LRS) [36] and leakage-resilient refreshing of LRS. Then, we give the detailed formal security proof. The main contributions are as follows:

- First, we first formalize an LR eCK security model for PAKE by combining the eCK security PAKE model and the only computation leakage (OCL) model appropriately.
- Second, we propose the first LR PAKE protocol by using Diffie-Hellman key exchange and the Dziembowski-Faust (DF) LRS (DF-LRS) scheme [37] properly. Our protocol is more efficient than other LR AKE protocols.
- Third, based on game simulation techniques, we show a formal security proof in the standard model under a stronger security model, namely, the λ-CAFLR eCK security model, in which the leakages are continuous and are allowed after the adversary selects the test session. In the model, the total leakage size may be infinitely large, and for each protocol instance, the amount of leakage is bounded by λ.

The remainder of this paper is organized as follows. In Section 2, we review the primitives that are used. In Section 3, we describe the CAFLR eCK security model of PAKE. In Section 4, we present the proposed protocol and analyse the provable security, performance comparison and leakage tolerance. Finally, in Section 5, we conclude the paper.

Compared with the conference version [38], there are four significant improvements in this paper. First, we formally give the detailed security proof in the standard model. Second, we analyse the leakage tolerance of our proposed protocol. Third, we complement the LR eCK security model for PAKE with a graphical framework of the security game. Finally, we present the primitives that are used and analyse why they are needed and how they are used.

II. PRELIMINARIES

In this section, we address the primitives that are used, such as the DDH assumption, KDF, LRS and leakage-resilient refreshing of LRS.

Notation: Let $s \xleftarrow{\$} S$ denote that s is a uniform value that is selected from a finite set S at random and let κ and λ denote the system security parameter and the leakage parameter, respectively.

Definition 1 (Negligible Function): A negligible function $\varepsilon(\kappa)$ is a function $N \to R$ such that for each positive integer $c \geq 0$, there exists an integer k_c such that $\varepsilon(\kappa) < k^{-c}$ for all $k \geq k_c$.

Definition 2 (DDH Assumption): We define a distinguishing game as follows:

(1) A challenger $\boldsymbol{C}$ generates a cyclic multiplicative group G with a large prime order p, picks a generator g at random, and then sends (G, g) to an adversary $\boldsymbol{A}$.
(2) $\boldsymbol{C}$ picks a bit $b \xleftarrow{\$} (0, 1)$ and three elements $x, y, z \xleftarrow{\$} Z_p^*$ at random. If $b = 0$, $\boldsymbol{C}$ sends (g^x, g^y, g^{xy}) to $\boldsymbol{A}$; otherwise, $\boldsymbol{A}$ is given (g^x, g^y, g^z).
(3) $\boldsymbol{A}$ outputs his guessed bit b'. $\boldsymbol{A}$ wins if $b' = b$.

DDH assumption is satisfied if

$$Adv_{DDH}(A) = |\Pr[b' = b] - 1/2| = \varepsilon(\kappa),$$

where $Adv_{DDH}(\mathrm{A})$ denotes the advantage of $\boldsymbol{A}$ in the distinguishing game and $\varepsilon(\kappa)$ is a negligible function.

Definition 3 (λ-Leakage-Resilient Storage): An λ-LRS consists of a pair of algorithms (***Encode***, ***Decode***) and a bounded leakage parameter $\lambda = (\lambda_1, \lambda_2)$.

Encode: $Encode(s) = s_L \times s_R$ is a randomized and efficient probabilistic polynomial time (PPT) algorithm, where s is an element that is chosen from the message space M and $s_L \times s_R$ is the encoded output element in the encoding space $L \times R$.

Decode: $Decode(s_L \times s_R) = s$ is a deterministic and efficient PPT algorithm.

An λ-LRS should satisfy the following two properties:

I. Correctness of the LRS. For every $s \xleftarrow{\$} M$, $Decode(Encode(s)) = s$.
II. Security of the LRS.

We define a distinguishing game as follows:

(1) $\boldsymbol{A}$ chooses two random messages $(s_0, s_1) \xleftarrow{\$} M$ and sends (s_0, s_1) to $\boldsymbol{C}$.
(2) $\boldsymbol{C}$ picks a bit $b \xleftarrow{\$} (0, 1)$ at random and calculates

$$Encode(s_b) = s_b^L \times s_b^R.$$

(3) For $i = 1, \cdots, t$, $\boldsymbol{A}$ selects leakage functions $f = (f_i^L, f_i^R)$ and sends it to $\boldsymbol{C}$ and $\boldsymbol{C}$ returns the leakages $(f_i^L(s_b^L), f_i^R(s_b^R))$ back to $\boldsymbol{A}$. The total leakage size should be bounded by (λ_1, λ_2), *i.e.*, $\sum_{i=1}^{t} f_i^L(s_b^L) \le \lambda_1 \wedge \sum_{i=1}^{t} f_i^R(s_b^R) \le \lambda_2$.
(4) $\boldsymbol{A}$ outputs his guessed bit b'. $\boldsymbol{A}$ wins if $b' = b$.

The λ-LRS is secure if the following holds:

$$Adv_{LRS}(A) = \varepsilon(\kappa),$$

where $Adv_{LRS}(A)$ represents the advantage of $\boldsymbol{A}$ in the distinguishing security game and $\varepsilon(\kappa)$ is a negligible function.

Definition 4 ($(\lambda_{Refresh}, \lambda)$-Leakage-Resilient Refreshing of the LRS): A leakage-resilient refreshing is a PPT algorithm ***Refresh*** with an λ-LRS (***Encode***,***Decode***), a secret s and a bounded leakage amount $\lambda_{Refresh} = (\lambda_{Refresh1}, \lambda_{Refresh2})$.

Refresh: $Refresh(s_L \times s_R) = s'_L \times s'_R$ where $s_L \times s_R$ is the encoding value of the secret s.

A leakage-resilient refreshing of an λ-LRS should satisfy the following two properties:

I. Correctness of the leakage-resilient refreshing. For every $s \xleftarrow{\$} M$,

$$Decode(s'_L \times s'_R) = Decode(s_L \times s_R).$$

II. $(\lambda_{Refresh}, \lambda)$-security of the leakage-resilient refreshing.

We define a distinguishing game as follows:

(1) $\boldsymbol{A}$ chooses two random messages $(s_0, s_1) \xleftarrow{\$} M$ and sends (s_0, s_1) to $\boldsymbol{C}$.
(2) $\boldsymbol{C}$ picks a bit $b \xleftarrow{\$} (0, 1)$ at random and calculates

$$Encode(s_b) = s_b^L \times s_b^R.$$

(3) For $i = 1, \cdots, t$, $\boldsymbol{C}$ runs the i^{th} round refreshing protocol, $Refresh(s_{bL}^{i-1} \times s_{bR}^{i-1}) = s_{bL}^i \times s_{bR}^i$, $\boldsymbol{A}$ selects the i^{th} round leakage functions $f_{Refresh\text{-}i} = (f_{Refresh\text{-}i}^L, f_{Refresh\text{-}i}^R)$ and sends it to $\boldsymbol{C}$, and $\boldsymbol{C}$ returns the leakages $(f_{Refresh\text{-}i}^L(s_{bL}^i), f_{Refresh\text{-}i}^R(s_{bR}^i))$ to $\boldsymbol{A}$, where $f_{Refresh\text{-}i}^L(s_{bL}^i) \le \lambda_{Refresh1} \wedge f_{Refresh\text{-}i}^R(s_{bR}^i) \le \lambda_{Refresh2}$ should hold.
(4) $\boldsymbol{A}$ outputs his guessed bit b'. $\boldsymbol{A}$ wins if $b' = b$.

An $(\lambda_{Refresh}, \lambda)$ leakage-resilient refreshing is secure if the following holds:

$$Adv_{Refresh-LRS}(A) = \varepsilon(\kappa),$$

where $Adv_{Refresh-LRS}(A)$ denotes the advantage of $\boldsymbol{A}$ in distinguishing the above security game and $\varepsilon(\kappa)$ is a negligible function.

Definition 5 (DF-LRS Scheme): The DF-LRS Scheme [37] is an LRS that efficiently stores a secret value s $\in (Z_p^*)^m$ with any m $\in N$, where p is a large prime. We denote it as $\Phi_{Z_p^*}^{n,m}$.

Encode: Pick $s_L \xleftarrow{\$} (Z_p^*)^n \backslash \{(0^n)\}$ at random, compute $s_R \in (Z_p^*)^{n \times m}$ such that $s_L \times s_R = s$, where n $\in N$, and then output (s_L, s_R).

Decode: Output s such that $s = s_L \times s_R$.

Lemma 6 [37]: The scheme that is defined in Definition 5 is an λ-secure LRS scheme if $20m < n$, where $\lambda = (0.3n\log p, 0.3n\log p)$.

Lemma 7 [37]: If $\Phi_{Z_p^*}^{n,m}$ is an λ-secure DF-LRS scheme and $m/3 \le n \wedge n \ge 16$, there is an $(\lambda/2, \lambda)$-secure leakage-resilient refreshing $Refresh_{Z_p^*}^{n,m}$ for $\Phi_{Z_p^*}^{n,m}$.

Definition 8 (Key Derivation Function): A KDF is a function $key \leftarrow KDF(\sigma, \ell, r, c)$ that can generate a cryptographically strong secret key efficiently, where σ is the source material of the secret key and ℓ denotes some public knowledge about σ such as its length, r represents a salt value and c denotes a context variable.

Definition 9 (Security of KDF): We define a distinguishing game as follows:

(1) $\boldsymbol{C}$ picks (σ, ℓ) and a salt value r at random, and then gives (ℓ, r) to $\boldsymbol{A}$.
(2) $\boldsymbol{A}$ chooses a value c at random, and then gives it to $\boldsymbol{C}$.
(3) $\boldsymbol{C}$ selects a random bit $b \xleftarrow{\$} (0, 1)$. If $b = 0$, $\boldsymbol{C}$ calculates $key \leftarrow KDF(\sigma, \ell, r, c)$ and gives it to $\boldsymbol{A}$; otherwise, $\boldsymbol{C}$ picks a string s at random and gives it to

A, where the length of s is the same as the length of $key \leftarrow KDF(\sigma, \ell, r, c)$.

(4) ***A*** outputs his guessed bit b'. ***A*** wins if $b' = b$.

A KDF is secure if the following holds:

$$Adv_{KDF}(A) = \varepsilon(\kappa),$$

where Adv_{KDF}(A) denotes the advantage of ***A*** in the distinguishing security game and $\varepsilon(\kappa)$ is a negligible function.

III. THE λ-CAFLR eCK SECURITY MODEL FOR PAKE

Based on the eCK security PAKE model and the OCL model, we define the λ-CAFLR eCK security model for PAKE in this section, where leakage attacks are modelled as leakage functions that are defined in ***Send*** queries. ***A*** can learn the leakages of the long-term secret password by asking ***Send*** queries with leakage functions that are chosen by him. The new model has three main properties: First, we suppose that only the calculations will lead to leakages of the long-term shared secret password pw. Second, in each instance of the protocol, the total leakage size of the secret password is limited to λ. ***A*** can perform leakage attacks by asking ***Send*** queries with the leakage functions $f = (f_1, \ldots, f_n)$, which are chosen adaptively by him, and get back the leakages of pw. However, we require that the total leakage amount is limited to λ for each instance, *i.e.*, $\sum_{i=1}^{n} |f_i(pw)| \leq \lambda$. Third, ***A*** can continuously carry out the leakage attacks instance by instance and learn an infinitely large amount of leakage information about the secret password.

A. ADVERSARIAL POWERS

In our model, two parties, who are denoted as U and V, run the PAKE protocol together to obtain a secure shared key. We define the following notations.

Session is used to represent a protocol instance.

Principal is used to denote a party of a session. A principal may be involved in multiple different sessions that may be executed concurrently.

Oracle($\Pi^s_{U,V}$) is used to represent the s^{th} session with principals U and V, of which U is the owner principal and V is the intended partner principal.

Initiator is used to represent the principal who activates a session.

Responder is used to represent the principal who responds to the initiator.

In our model, the adversary ***A*** is active, adaptive and malicious, interacts with any oracles and performs attacks. We model the adversarial capabilities by the following queries.

Send(U, V, s, m, f) query: this query models ***A***'s abilities to execute the protocol and carry out the leakage attacks. The adversary ***A*** sends a ***Send***(U, V, s, m, f) query to the oracle $\Pi^s_{U,V}$ in the s^{th} session, where m is a protocol message and f is a leakage function. Then, ***A*** gets back a normal protocol message and the leakage $f(pw)$ of the long-term password, which are produced by the oracle $\Pi^s_{U,V}$ based on the protocol specifications and f. ***A*** can use this query to run a protocol or activate a new protocol instance as an initiator with blank m and f.

RevealSessionKey(U, V, s) query: this query models ***A***'s capability to learn the s^{th} session key. The adversary ***A*** sends this query to the oracle $\Pi^s_{U,V}$ in the s^{th} session. Then, ***A*** gets back the s^{th} session key from $\Pi^s_{U,V}$.

RevealEphemeralKey(U, V, s) query: this query models ***A***'s capability to learn ephemeral keys of the s^{th} session. The adversary ***A*** sends this query to the oracle $\Pi^s_{U,V}$ in the s^{th} session. Then, ***A*** gets back the s^{th} ephemeral keys of $\Pi^s_{U,V}$.

RevealPassword() query: this query models ***A***'s capability to learn the principals' shared password. The adversary ***A*** sends this query to any oracle in any session. Then, ***A*** gets back the long-term shared secret password pw.

Test(U, s) query: this query is different from all of the above queries, as it is only used to specify the security definition of our model. Upon receiving this query from the adversary ***A***, the challenger ***C*** chooses a bit $b \xleftarrow{\$} (0, 1)$ at random. If $b = 1$, then ***C*** sends the actual session key to ***A***, while a random string is given to ***A***. ***A*** can issue this query only once across all sessions.

B. λ-CAFLR eCK SECURITY MODEL

In our security model, the total leakage size of the secret password for each instance is bounded by the leakage parameter λ, *i.e.*, $\sum_{i=1}^{n} |f_i(pw)| \leq \lambda$.

Definition 10 (Partner in λ-CAFLR eCK Security Model): Two oracles $\Pi^s_{U,V}$ and $\Pi^{s'}_{U',V'}$ are partners if they have the following properties:

(1) $\Pi^s_{U,V}$ and $\Pi^{s'}_{U',V'}$ have received the same session keys;

(2) The messages that are sent from $\Pi^s_{U,V}$ are the same as the messages that are received by $\Pi^{s'}_{U',V'}$;

(3) The messages that are sent from $\Pi^{s'}_{U',V'}$ are the same as the messages that are received by $\Pi^s_{U,V}$;

(4) $U = V'$ and $V = U'$;

(5) There are an initiator and a responder of two principals U and V.

Definition 11 (λ-C AFLR-eCK-Freshness): Assume $f = (f_1, \ldots, f_n)$ denotes n PPT leakage functions for a certain protocol instance that is chosen by the adversary ***A*** arbitrarily. An oracle $\Pi^s_{U,V}$ is λ-CAFLR-eCK-fresh if the followings hold:

(1) ***RevealSessionKey*** queries have not been asked by the oracle $\Pi^s_{U,V}$ or its partner, $\Pi^{s'}_{V,U}$ (if it exists).

(2) If the partner $\Pi^{s'}_{V,U}$ exists, neither of the following combinations has been queried:

   (a) ***RevealPassword***() and ***RevealEphemeralKey***(U, V, s);

   (b) ***RevealPassword***() and ***RevealEphemeralKey***(V, U,s').

(3) If the partner $\Pi^{s'}_{V,U}$ does not exist, ***A*** cannot ask the ***RevealPassword***() query.

(4) For all ***Send***(., U, ., ., f_i) queries, $\sum_{i=1}^{n} |f_i(pw)| \leq \lambda$.

(5) For all ***Send***(., V, ., ., f_i) queries, $\sum_{i=1}^{n} |f_i(pw)| \leq \lambda$.

C. SECURITY DEFINITION

This section formalizes the security definition of the λ-CAFLR eCK model.

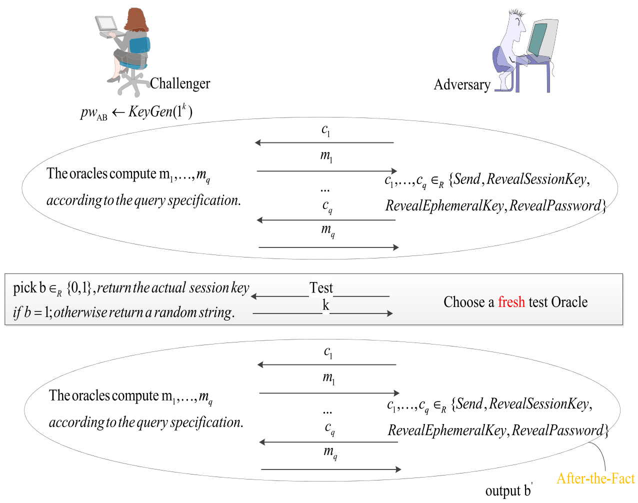


FIGURE 1. The Security Game of λ-CAFLR eCK-Secure PAKE.

Definition 12 (λ -CAFLR eCK Security Game): The λ -CAFLR eCK security game is shown in Fig. 1, which is run by the protocol challenger $\boldsymbol{C}$ with a PPT adversary $\boldsymbol{A}$:

(1) $\boldsymbol{A}$ asks any of the ***Send***, ***RevealSessionKey***, ***RevealEphemeralKey*** and ***RevealPassword*** queries to any oracle.

(2) $\boldsymbol{A}$ chooses an λ-CAFLR-eCK-fresh oracle and asks a ***Test*** query. Upon receiving a ***Test*** query, $\boldsymbol{C}$ selects a random bit $b \xleftarrow{\$} (0, 1)$. If $b = 1$, then sends the actual session key to $\boldsymbol{A}$, while a random string is given to $\boldsymbol{A}$.

(3) $\boldsymbol{A}$ continues asking ***Send***, ***RevealSessionKey***, ***RevealEphemeralKey*** and ***RevealPassword*** queries. All these queries should satisfy the λ-CAFLR-eCK-freshness of the chosen oracle.

(4) At last, $\boldsymbol{A}$ outputs his guessed bit b'. $\boldsymbol{A}$ wins if $b' = b$.

Definition 13 (λ-CAFLR eCK Security): Let $Adv_{\mathrm{PAKE}}^{\lambda-CAFLReCK}$ be the advantage of $\boldsymbol{A}$ in the λ-CAFLR eCK security game that is defined in Definition 12. λ-CAFLR eCK security means that

$$Adv_{\mathrm{PAKE}}^{\lambda-CAFLReCK} = |\Pr[b' = b] - 1/2| = N_S/N + \varepsilon(\kappa),$$

where N_S is the number of sessions, N denotes the size of the password dictionary, and $\varepsilon(\kappa)$ represents a negligible function.

In other words, a PAKE protocol is λ-CAFLR eCK-secure if there is no PPT adversary who can win the above security game with an advantage of more than N_S/N. In PAKE protocols, on-line dictionary attacks are unavoidable, and N_S/N represents the success probability of on-line dictionary attacks. However, this attacks can be limited by some kind of strategy, for example, by disallowing further attempts after a certain number of failed attempts to the correct password.

IV. A NEW λ-CAFLR ECK-SECURE PAKE PROTOCOL

In this section, we formally present our λ-CAFLR eCK-secure PAKE protocol and its detailed security proof in the standard model.

A. THE PROPOSED PROTOCOL

1) OVERVIEW OF THE PROPOSED PROTOCOL

There are three main stages:

- Initially, we map the password pw to a random element s of a group G using a one-way collision-free hash function H, and then encode s using an LRS scheme. This approach can resist leakage attacks on the shared secret password. However, determining how to use the encodings of the shared password to achieve authentication and obtain a secure session key becomes a big challenge.
- Then, we use the encodings of the shared password to achieve authentication and obtain a secure session key by combining Diffie-Hellman key exchange and the DF-LRS scheme appropriately. This method gives a good solution to the above challenge. The important observations are as follows: (1) Two primitives can share a common group G with a big prime order p; (2) In the DF-LRS scheme, $(g^{s_L})^{s_R} = g^{s_L \cdot s_R} = g^s$ since $s = s_L \cdot s_R$, where g is a generator of G, s denotes the secret mapping element, and (s_L, s_R) represents two encodings of the DF-LRS scheme.
- Finally, since there is an efficient leakage-resilient refreshing protocol for the DF-LRS scheme, we can refresh two encodings of s after using them in the end of the protocol. Thus, our construction is secure against continuous leakage attacks.

2) DETAILS OF THE PROPOSED PROTOCOL

Fig. 2 illustrates the proposed protocol, which includes the following two stages.

a: THE INITIAL SETUP STAGE

Users U and V first map the shared secret password to a random element of the group G, $\mathrm{s}_{UV} = H(\mathrm{pw}_{UV})$. We assume that this computation is executed in secret and leakage attacks are not allowed. Then, U runs an λ-secure DF-LRS scheme $\Phi_{Z_p^*}^{n,1}$, picks $a_L^0 \xleftarrow{\$} (Z_p^*)^n \backslash \{(0^n)\}$ at random and generates $a_{\mathrm{R}}^0 \in (Z_p^*)^{n\times 1}$such that $a_L^0 \cdot a_{\mathrm{R}}^0 = \mathrm{s}_{UV}$; V also chooses $b_L^0 \xleftarrow{\$} (Z_p^*)^n \backslash \{(0^n)\}$ at random and computes $b_R^0 \in (Z_p^*)^{n\times 1}$ such that $b_L^0 \cdot b_{\mathrm{R}}^0 = \mathrm{s}_{UV}$.

b: THE PROTOCOL EXECUTION STAGE

Step 1. User U calculates $Y_U = g^{x_U}$ and $T_{U1} = (Y_U)^{a^j_L}$, where $x_U \xleftarrow{\$} Z^*_p$ is a random number, and then sends (U, T_{U1}) to user V.

Step 2. Upon receiving (U, T_{U1}), V calculates $Y_V = g^{x_V}$, $T_{U2} = (T_{U1})^{x_V}$ and $T_{V1} = (Y_V)^{b^j_L}$, where $x_V \xleftarrow{\$} Z^*_p$ is a random number, and then sends (V, T_{U2}, T_{V1}) to U.

Step 3. Upon receiving (V, T_{U2}, T_{V1}), U calculates $T_{V2} = (T_{V1})^{x_U}$ and sends it to V. Finally, U calculates $T_U = (T_{U2})^{a^j_R}$ and $k_{UV} = KDF(U, V, T_U)$, and refreshes the stored pieces with

$$(a^{j+1}_L, a^{j+1}_R) \leftarrow Refresh^{n,1}_{Z^*_p}(a^j_L, a^j_R).$$

Step 4. Upon receiving (U, T_{V2}), V calculates $T_V = (T_{V2})^{b^j_R}$ and $k_{UV} = KDF(U, V, T_V)$, and refreshes the stored pieces with
$(b^{j+1}_L, b^{j+1}_R) \leftarrow Refresh^{n,1}_{Z^*_p}(b^j_L, b^j_R)$.

Correctness of the proposed protocol.

From

$$\begin{aligned} T_U &= (T_{U2})^{a^j_R} = (((g^{x_U})^{a^j_L})^{x_V})^{a^j_R} \\ &= (((g^{x_U})^{x_V})^{a^j_L})^{a^j_R} = ((g^{x_U})^{x_V})^{a^j_L \cdot a^j_R} \\ &= ((g^{x_U})^{x_V})^{s_{UV}} = (((g^{x_U})^{x_V})^{b^j_L})^{b^j_R} \\ &= (((g^{x_V})^{b^j_L})^{x_U})^{b^j_R} \\ &= (((Y_V)^{b^j_L})^{x_U})^{b^j_R} \\ &= (T_{V2})^{b^j_R} = T_V \end{aligned}$$

it follows that

$$\Rightarrow KDF(U, V, T_U) = KDF(U, V, T_V).$$

Therefore, the proposed protocol is correct.

B. MUTUAL AUTHENTICATION

In our protocol, we can add mutual authentication conveniently. To do this, we can introduce an authenticator structure $KDF(U, V, k_{UV})$, where $k_{UV} = KDF(U, V, T_V)$. At the end of the protocol, users U and V each calculate the authenticator Auth $= KDF(U, V, k_{UV})$ and send Auth to the other user. Then, each user can identify the other's identity by checking whether the received authenticator Auth is equal to $KDF(U, V, k_{UV})$. When we give our security proof, we will not consider the security of mutual authentication because this authenticator transformation preserves the indistinguishability security of the original protocol.

C. SECURITY PROOF

Our CAFLR PAKE protocol is *eCK*-secure if the DDH problem is hard and the leakage-resilient refreshing of LRS and KDF is secure.

*Theorem 14: Let $Adv^{\lambda-CAFLReCK}_{PAKE}$ represent the advantage of a PPT adversary **A** against the λ -CAFLR eCK-security of the proposed protocol, and let Adv_{DDH}, Adv_{KDF}*

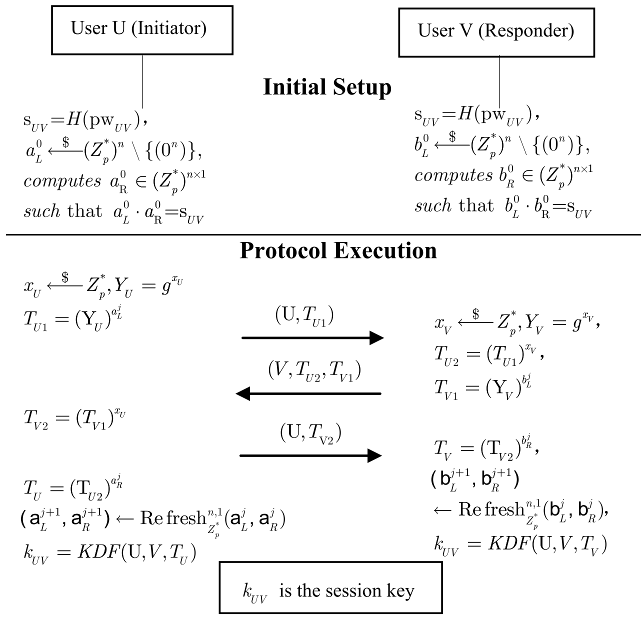


FIGURE 2. The λ-CAFLR eCK-Secure PAKE Protocol.

*and $Adv_{Refresh-LRS}$ be advantages of **A** against the security of the DDH assumption, KDF and leakage-resilient refreshing of LRS, respectively. Then*

$$Adv^{\lambda-CAFLReCK}_{PAKE} \le N_S/N + N^2_P N^2_S (Adv_{DDH} + Adv_{Refresh-LRS} + Adv_{KDF}),$$

where N_P is the number of protocol principals, N_S denotes the number of sessions on a principal, and N represents the size of the password dictionary.

Proof: The proof can be divided into the following two main cases.

Case 1 (A Partner Session to the Test Session Exists):

In this case, the adversary **A** may obtain the principals' long-term shared secret password pw_{UV} by the ***RevealPassword*** query. Let $Adv^{\lambda-CAFLReCK}_{PAKE}$ be the advantage of **A** in the λ-CAFLR eCK security game. We split this case into two subcases as follows:

(1) **A** asks a ***RevealPassword*** query. In this case, **A** can learn the principals' long-term shared secret password. To not violate the λ-CAFLR-eCK-freshness of the chosen session, **A** cannot ask ***RevealEphemeralKey*** query to learn random ephemeral keys of the owner or partner principals to the chosen session.

(2) **A** doesn't ask a ***RevealPassword*** query. In this case, **A** cannot obtain the long-term shared secret password, but can learn random ephemeral keys of the owner and his partner to the chosen session.

Case 1.1 (A Asks a RevealPassword Query):

In this case, the adversary **A** can obtain the principals' long-term shared secret password pw_{UV} by the ***RevealPassword*** query and learn $s_{UV} = H(pw_{UV})$. Thus, leakage attacks do not need to be considered. To not violate the λ-CAFLR-eCK-freshness of the chosen session, **A** cannot obtain any of the principals' ephemeral keys to the chosen session.

Game 1: This game is the original λ-CAFLR eCK security game that is defined in Definition 12.

Game 2: Game 2 has the following differences from Game 1: First, the adversary A picks two random distinct principals $U^*, V^* \xleftarrow{\$} \{U_1, \ldots, U_{N_p}\}$ and two random numbers $s^*, t^* \xleftarrow{\$} \{1, \ldots, N_s\}$, where N_P denotes the number of principals and N_S represents the number of sessions on a principal. Second, A activates the security game and chooses $\Pi^{s^*}_{U^*,V^*}$ as the target oracle and $\Pi^{t^*}_{V^*,U^*}$ as the partner oracle. If the test oracle is not $\Pi^{s^*}_{U^*,V^*}$ or the partner oracle is not $\Pi^{t^*}_{V^*,U^*}$, the Game 2 challenger stops and exits the game.

Game 3: Game 3 has the following differences from Game 2: The Game 3 challenger C picks $z \xleftarrow{\$} Z_p^*$ at random and calculates $s_{U*V^*} = H(pw_{U^*V^*})$ and $k_{U^*V^*} = KDF(U_{U^*}, U_{V^*}, (g^{s_{U^*V^*}})^z)$. After receiving a $Test(U^*, V^*, s^*)$ query from A, C gives $k_{U^*V^*}$ to A. In addition, after receiving a $Test(V^*, U^*, t^*)$ query, C sends the same $k_{U^*V^*}$ to A, since there is a partner session $\Pi^{t^*}_{V^*,U^*}$.

Game 4: Game 4 has the following differences from Game 3: The Game 4 challenger C selects a random value $k_{U^*V^*} \xleftarrow{\$} \{0, 1\}^k$. Then, after receiving a $Test(U^*, V^*, s^*)$ query or $Test(V^*, U^*, t^*)$ query from A, C gives $k_{U^*V^*}$ to A.

Differences Between Games: We analyse the indistinguishability of each game t from its previous game t-1. Let $Adv_{Gamet}(A)$ be the advantage of A in Game t.

Game 1:

$$Adv_{Game1}(A) = Adv^{\lambda-CAFLReCK}_{\text{PAKE}} \quad \text{(I)}$$

Game 1 and Game 2: Game 1 and Game 2 are the same if the target oracle and the partner oracle are chosen by A correctly. The probability of A choosing a correct test oracle and its correct partner is $1/N_P^2 N_S^2$. Therefore,

$$Adv_{Game2}(A) = 1/N_P^2 N_S^2 Adv_{Game1}(A) \quad \text{(II)}$$

Game 2 and Game 3: In Game 2, $k_{U^*V^*} = KDF(U_{U^*}, U_{V^*}, (g^{s_{U^*V^*}})^{x_{U^*} \cdot x_{V^*}})$, while $k_{U^*V^*} = KDF(U_{U^*}, U_{V^*}, (g^{s_{U^*V^*}})^z)$ in Game 3. Assume A outputs a bit b to distinguish between Game 2 and Game 3: $b = 1$ if Game 2 is running; otherwise, $b = 0$. We design an algorithm B against the DDH distinguishing game, which uses A as a subroutine and runs as follows: (1) Upon receiving a message $((g^{s_{U^*V^*}})^{x_{U^*}}, (g^{s_{U^*V^*}})^{x_{V^*}}, (g^{s_{U^*V^*}})^{x_{U^*} \cdot x_{V^*}})$ or $((g^{s_{U^*V^*}})^{x_{U^*}}, (g^{s_{U^*V^*}})^{x_{V^*}}, (g^{s_{U^*V^*}})^z)$ from the DDH challenger, B transfers it to A'schallenger, who uses it to generate the response message to A's challenge. If the received message is $((g^{s_{U^*V^*}})^{x_{U^*}}, (g^{s_{U^*V^*}})^{x_{V^*}}, (g^{s_{U^*V^*}})^{x_{U^*} \cdot x_{V^*}})$, the simulation is the same as Game 2; otherwise, it's the same as Game 3. (2) B outputs the bit that A outputs.

If A can distinguish between Game 2 and Game 3, B wins the DDH distinguishing game. Therefore,

$$|Adv_{Game2}(A) - Adv_{Game3}(A)| \leq Adv_{DDH} \quad \text{(III)}$$

Game 3 and Game 4: In Game 3, $k_{U^*V^*} = KDF(U_{U^*}, U_{V^*}, (g^{s_{U^*V^*}})^z)$, while in Game 4, $k_{U^*V^*} \xleftarrow{\$} \{0, 1\}^k$. Assume A outputs a bit b to distinguish between Game 3 and Game 4: $b = 1$ if Game 3 is running; otherwise, $b = 0$. We design an algorithm B against the KDF distinguishing game, which uses A as a subroutine and runs as follows: (1) Upon receiving a message $k_{U^*V^*} = KDF(U_{U^*}, U_{V^*}, (g^{s_{U^*V^*}})^z)$ or $k_{U^*V^*} \xleftarrow{\$} \{0, 1\}^k$ from the KDF challenger, B transfers it to A's challenger, who uses it to generate the answer message to A's challenge. If the received message is $KDF(U_{U^*}, U_{V^*}, (g^{s_{U^*V^*}})^z)$, the simulation is the same as Game 3; otherwise, it's the same as Game 4. (2) B outputs the bit that A outputs.

If A can distinguish between Game 3 and Game 4, B wins the KDF distinguishing game. Therefore,

$$|Adv_{Game\ 3}(A) - Adv_{Game\ 4}(A)| \leq Adv_{KDF} \quad \text{(IV)}$$

Game 4: A has no advantage in Game 4 because the session key $k_{U^*V^*}$ of $\Pi^{s^*}_{U^*,V^*}$ is picked at random and doesn't depend on any other values. Therefore,

$$Adv_{Game4}(A) = 0 \quad \text{(V)}$$

Using equations (I)-(V), we obtain

$$Adv^{\lambda-CAFLReCK}_{\text{PAKE}} \leq N_P^2 N_S^2 (Adv_{DDH} + Adv_{KDF}).$$

Case 1.2 (A Does Not Ask a RevealPassword Query):

For simplicity, we assume that the test oracle is an initiator.

Game 1: Same as Game 1 in Case 1.1.

Game 2: Same as Game 2 in Case 1.1.

Game 3: Game 3 has the following differences from Game 2: Game 3 challenger C picks $s \xleftarrow{\$} Z_p^*$ at random, encodes $(s_L, s_R) = Encode(s)$, continues refreshing the two encodings, and then uses the refreshed encodings of s to simulate the answers to A's leakage query function $f_{Refresh} = (f^L_{Refresh}, f^R_{Refresh})$ of the principal U^*.

Game 4: Game 4 has the following differences from Game 3: Game 4 challenger C chooses a random element $s' \xleftarrow{\$} Z_p^*$ and calculates $k_{U^*V^*} = KDF(U_{U^*}, U_{V^*}, (g^{x_{U^*} \cdot x_{V^*}})^{s'})$. After receiving a $Test(U^*, V^*, s^*)$ query from the adversary A, C gives $k_{U^*V^*}$ to A. In addition, after receiving a $Test(V^*, U^*, t^*)$ query, C also sends the same $k_{U^*V^*}$ to A, since there is a partner oracle $\Pi^{t^*}_{V^*,U^*}$.

Game 5: Same as Game 4 in Case 1.1.

Differences Between Games:

Game 1:

$$Adv_{Game1}(A) = Adv^{\lambda-CAFLReCK}_{\text{PAKE}} \quad \text{(I)}$$

Game 1 and Game 2: From Game 1 and Game 2 in Case 1.1., we get

$$Adv_{Game2}(A) = 1/N_P^2 N_S^2 Adv_{Game1}(A) \quad \text{(II)}$$

Game 2 and Game 3: In Game 2, the leakage of the shared password is the real leakage of $s_{U*V^*} = H(pw_{U^*V^*})$, while the leakage in Game 3 is a leakage of a random value s. Assume A outputs a bit b to distinguish between Game 2 and Game 3: $b = 1$ if Game 2 is running; otherwise, $b = 0$. We design an algorithm B against the leakage-resilient refreshing security distinguishing game, which uses $\boldsymbol{A}$ as a subroutine and runs as follows: (1) Upon receiving $\mathrm{s}_{\mathrm{U}*V^*}$ or $\mathrm{s} \xleftarrow{\$} Z_p^*$ from the leakage-resilient refreshing challenger, $\boldsymbol{B}$ transfers it to $\boldsymbol{A}$'s challenger $\boldsymbol{C}$. $\boldsymbol{C}$ uses it as the mapping group element of the shared secret password, encodes it, continues refreshing two encodings, and then uses these encodings to simulate the answers to $\boldsymbol{A}$'s ***Send*** queries with $f_{Refresh} = (f^L_{Refresh}, f^R_{Refresh})$ of the principal U^*. If the received message is $\mathrm{s}_{\mathrm{U}*V^*}$ in the first step, the simulation is the same as Game 2; otherwise, it's the same as Game 3. (2) $\boldsymbol{B}$ outputs the same bit that $\boldsymbol{A}$ outputs.

If $\boldsymbol{A}$ can distinguish between Game 2 and Game 3, $\boldsymbol{B}$ wins the leakage-resilient refreshing security distinguishing game. Therefore,

$$|\mathrm{A}dv_{Game2}(A) - \mathrm{A}dv_{Game3}(A)| \leq Adv_{Refresh-LRS} \qquad \text{(III)}$$

Game 3 and Game 4: In Game 3, $k_{U^*V^*} = KDF(U_{U^*}, U_{V^*}, (g^{x_{\mathrm{U}*}\cdot x_{V^*}})^{s_{U^*V^*}})$, while $k_{U^*V^*} = KDF(U_{U^*}, U_{V^*}, (g^{x_{U^*}\cdot x_{V^*}})^{s'})$ in Game 4. Because s' is chosen at random and is independent of $s_{U^*V^*}$, $(g^{x_{U^*}\cdot x_{V^*}})^{s_{U^*V^*}}$ and $(g^{x_{\mathrm{U}*}\cdot x_{V^*}})^{s'}$ are perfectly indistinguishable. Therefore,

$$|\mathrm{A}dv_{Game3}(A) - \mathrm{A}dv_{Game4}(A)| = 0 \qquad \text{(IV)}$$

Game 4 and Game 5: From Game 3 and Game 4 in Case 1.1., we obtain

$$|\mathrm{A}dv_{Game4}(A) - \mathrm{A}dv_{Game5}(A)| \leq Adv_{KDF} \qquad \text{(V)}$$

Game 5: In Game 5, the leakage is computed using a random value s, and the session key $k_{U^*V^*}$ of $\Pi^{s^*}_{U^*,V^*}$ is picked at random. Therefore,

$$\mathrm{A}dv_{Game5}(A) = 0 \qquad \text{(VI)}$$

Using equations (I)-(VI), we obtain

$$Adv^{\lambda-CAFLReCK}_{\mathrm{PAKE}} \leq N_P^2 N_S^2 (Adv_{Refresh-LRS} + Adv_{KDF}).$$

Case 2 (A Partner Session to the Test Session Does Not Exist):

In this case, $\boldsymbol{A}$ is an active adversary who masquerades as the intended partner principal of the owner principal. Therefore, $\boldsymbol{A}$ is not permitted to obtain the principals' long-term shared password by asking a ***RevealPassword*** query. However, $\boldsymbol{A}$ can learn the two parties' ephemeral session keys by asking ***RevealEphemeralKey*** queries.

Game 1: Same as Game 1 in Case 1.2.

Game 2: Game 2 has the following differences from Game 1: $\boldsymbol{A}$ picks a random password pw'_{UV}, computes $\mathrm{s}'_{\mathrm{UV}} = H(\mathrm{pw}'_{UV})$, encodes it, and then uses the encodings of $\mathrm{s}'_{\mathrm{UV}}$ to generate the protocol message based on the protocol specifications.

Game 3: Game 3 has the following differences from Game 2: First, $\boldsymbol{A}$ chooses two random distinct principals $U^*, V^* \xleftarrow{\$} \{U_1, \ldots, U_{N_p}\}$ and a random number $s^* \xleftarrow{\$} \{1, \ldots, N_s\}$. Second, $\boldsymbol{A}$ begins to run the game and chooses $\Pi^{s^*}_{U^*,V^*}$ as the target oracle. If the test oracle is not $\Pi^{s^*}_{U^*,V^*}$, the Game 3 challenger stops and exits the game.

Game 4: Same as Game 3 in Case 1.2.

Game 5: Same as Game 4 in Case 1.2.

Game 6: Same as Game 5 in Case 1.2.

Differences Between Games:

Game 1:

$$\mathrm{A}dv_{Game1}(A) = Adv^{\lambda-CAFLReCK}_{\mathrm{PAKE}} \qquad \text{(I)}$$

Game 1 and Game 2: If pw'_{UV} selected by $\boldsymbol{A}$ is equal to pw_{UV}, Game 2 is the same as Game 1; otherwise, Game 2 is independent of Game 1. The probability that $\mathrm{pw}'_{UV} = \mathrm{pw}_{UV}$ is N_S/N, where N_S denotes the number of sessions on a principal and N is the size of the password dictionary. Therefore,

$$|\mathrm{A}dv_{Game2}(A) - \mathrm{A}dv_{Game1}(A)| = N_S/N \qquad \text{(II)}$$

Game 2 and Game 3: The analysis is the same as that for Game 1 and Game 2 in Case 1.1.

$$\mathrm{A}dv_{Game3}(A) = 1/N_P^2 N_S \mathrm{A}dv_{Game2}(A) \qquad \text{(III)}$$

Game 3 and Game 4: The analysis is the same as that for Game 2 and Game 3 in Case 1.2.

$$|\mathrm{A}dv_{Game3}(A) - \mathrm{A}dv_{Game4}(A)| \leq Adv_{Refresh-LRS} \qquad \text{(IV)}$$

Game 4 and Game 5: The analysis is the same as that for Game 3 and Game 4 in Case 1.2.

$$|\mathrm{A}dv_{Game4}(A) - \mathrm{A}dv_{Game5}(A)| = 0 \qquad \text{(V)}$$

Game 5 and Game 6: The analysis is the same as that for Game 4 and Game 5 in Case 1.2.

$$|\mathrm{A}dv_{Game5}(A) - \mathrm{A}dv_{Game6}(A)| \leq Adv_{KDF} \qquad \text{(VI)}$$

Game 6: The analysis is the same as that for Game 5 in Case 1.2.

$$\mathrm{A}dv_{Game6}(A) = 0 \qquad \text{(VII)}$$

Using equations (I)-(VII), we obtain

$$Adv^{\lambda-CAFLReCK}_{\mathrm{PAKE}} \leq N_S/N + N_P^2 N_S (Adv_{Refresh-LRS} + Adv_{KDF}).$$

From **Case 1** and **Case 2**, we obtain

$$Adv^{\lambda-CAFLReCK}_{\mathrm{PAKE}} \leq N_S/N + N_P^2 N_S^2 (Adv_{DDH} + Adv_{Refresh-LRS} + Adv_{KDF}).$$

■

D. SECURITY AND PERFORMANCE COMPARISON

We summarize the security and performance comparison of our protocol with other protocols in Table 1, where Exp denotes modular exponentiation.

From Table 1, our protocol enjoys three advantages: (1) it's the first LR PAKE protocol; (2) it's an AFLR eCK-secure AKE protocol in the standard model, while leakage attacks are not allowed after the adversary chooses the test session in [25], and its AFLR eCK-security has just been proven in the CK security model in [28] and in the RO model in [29] and [30]; (3) our protocol is more efficient than other LR AKE protocols [25], [28]–[31].

TABLE 1. Security and efficiency comparison of AKE protocols.

Scheme	[25]	[28]	[29]	[30]	[31]	Ours
Security model	eCK	CK	eCK	eCK	eCK	eCK
Leakage Feature	RLM	CLM	RLM	CLM	RLM	**CLM**
After-the-fact	No	Yes	Yes	Yes	Yes	**Yes**
Proof model	Standard	Standard	RO	RO	Standard	**Standard**
Key Infrastructure	PKI	PKI	PKI	PKI	PKI	**PW-based**
Rounds	2	1	2	1	1	**2**
Computations	16Exp	20Exp	24Exp	12Exp	16Exp	**8Exp**

E. LEAKAGE TOLERANCE OF THE PROPOSED PROTOCOL

First, the overall leakage amount is arbitrarily large since the encodings are refreshed in each instance of the proposed protocol and continuous leakage is allowed.

Second, for each instance of the proposed protocol, the leakage size is bounded by $\lambda_{Refresh} = (\lambda_{Refresh1}, \lambda_{Refresh2})$. Based on Lemma 6, an LRS scheme $\Phi^{n,1}_{Z^*_p}$ is λ-secure with $\lambda = (0.3n\log \mathrm{p}, 0.3n\log \mathrm{p})$ if $20 < n$. Moreover, based on Lemma 7, a leakage-resilient refreshing $Refresh^{n,1}_{Z^*_p}$ for $\Phi^{n,1}_{Z^*_p}$ is $(\lambda/2, \lambda)$- secure if $1/3 \leq n \wedge n \geq 16$. Therefore, the leakage size for each occurrence is bounded by $(0.15n\log \mathrm{p}, 0.15n\log \mathrm{p})$. In the protocol, the shared secret password is mapped to a group element $\mathrm{s}_{UV} = H(\mathrm{pw}_{UV})$ and encoded into two parts, namely, $a_L \in (Z^*_p)^n$ and $a_R \in (Z^*_p)^{n\times 1}$, of size $n \cdot \log \mathrm{p}$. Thus, the leakage tolerance for each occurrence is

$$(0.15n\log \mathrm{p}/n\log \mathrm{p}, 0.15n\log \mathrm{p}/n\log \mathrm{p}) = 15\%.$$

V. CONCLUSION

By combining Diffie-Hellman key exchange and the DF-LRS scheme appropriately, we first design an λ-CAFLR eCK security PAKE protocol. Our protocol is one of most practical cryptographic primitives for trusted computing, which could be used to securely authenticate devices' identities and generate shared session keys among devices in insecure leakage environments such as IoT.

REFERENCES

[1] A. G. Reddy, E.-J. Yoon, A. K. Das, V. Odelu, and K.-Y. Yoo, "Design of mutually authenticated key agreement protocol resistant to impersonation attacks for multi-server environment," *IEEE Access*, vol. 5, pp. 3622–3639, 2017.
[2] Trusted Computing Group. (Sep. 2015). *Guidance for Securing IoT Using TCG Technology Reference Document*. [Online]. Available: https://www.trustedcomputinggroup.org/wp-tontent/uploads/TCG_Guidance_for_Securing_IoT_1_0r21.pdf
[3] C.-L. Hsu, T.-H. Chuang, and T.-W. Lin, "End-to-end authenticated key exchange agreement for wearable devices in IoT environments," in *Proc. IEEE Great Lakes Biomed. Conf. (GLBC)*, Milwaukee, WI, USA, Apr. 2017, p. 1, doi: 10.1109/GLBC.2017.7928891.
[4] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 1992, pp. 72–84.
[5] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proc. EUROCRYPT*, Bruges, Belgium, 2000, pp. 139–155.
[6] P. MacKenzie, S. Patel, and R. Swaminathan, "Password-authenticated key exchange based on RSA," in *Proc. ASIACRYPT*, Kyoto, Japan, 2000, pp. 599–613.
[7] J. W. Byun, D. H. Lee, and J. I. Lim, "EC2C-PAKA: An efficient client-to-client password-authenticated key agreement," *Inf. Sci.*, vol. 177, no. 19, pp. 3995–4013, 2007.
[8] M. S. Farash and M. A. Attari, "An efficient client–client password-based authentication scheme with provable security," *J. Supercomput.*, vol. 70, no. 2, pp. 1002–1022, 2014.
[9] O. Goldreich and Y. Lindell, "Session-key generation using human passwords only," *J. Cryptol.*, vol. 19, no. 3, pp. 241–340, 2006.
[10] J. Katz, R. Ostrovsky, and M. Yung, "Efficient and secure authenticated key exchange using weak passwords," *J. ACM*, vol. 57, no. 1, 2009, Art. no. 3.
[11] J. Katz, P. MacKenzie, G. Taban, and V. Gligor, "Two-server password-only authenticated key exchange," *J. Comput. Syst. Sci.*, vol. 78, no. 2, pp. 651–669, 2012.
[12] R. Canetti, D. Dachman-Soled, V. Vaikuntanathan, and H. Wee, "Efficient password authenticated key exchange via oblivious transfer," in *Proc. PKC*, Darmstadt, Germany, 2012, pp. 449–466.
[13] V. Goyal, "Positive results for concurrently secure computation in the plain model," in *Proc. 53rd Annu. Symp. Found. Comput. Sci. (FOCS)*, New Brunswick, NJ, USA, 2012, pp. 41–50.
[14] W. M. Li, Q. Y. Wen, Q. Su, H. Zhang, and Z. P. Jin,, "Password-authenticated multiple key exchange protocol for mobile applications," *China Commun.*, vol. 9, no. 1, pp. 64–72, 2012.
[15] K.-L. Tsai, Y.-L. Huang, F.-Y. Leu, and I. You, "TTP based high-efficient multi-key exchange protocol," *IEEE Access*, vol. 4, pp. 6261–6271, 2016.
[16] M. Luo, X. Zhou, L. Li, K.-K. R. Choo, and D. He, "Security analysis of two password-authenticated multi-key exchange protocols," *IEEE Access*, vol. 5, pp. 8017–8024, 2017.
[17] O. Ruan, N. Kumar, D. He, and J.-H. Lee, "Efficient provably secure password-based explicit authenticated key agreement," *Pervasive Mobile Comput.*, vol. 24, no. 12, pp. 50–60, 2015.
[18] X. Yi *et al.*, "ID2S password-authenticated key exchange protocols," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3687–3701, Dec. 2016.
[19] S. H. Islam, "Design and analysis of a three party password-based authenticated key exchange protocol using extended chaotic maps," *Inf. Sci.*, vol. 312, pp. 104–130, Aug. 2015.
[20] R. Amin and G. P. Biswas, "Cryptanalysis and design of a three-party authenticated key exchange protocol using smart card," *Arabian J. Sci. Eng.*, vol. 40, no. 11, pp. 3135–3149, 2015.
[21] C. F. Lu, "Multi-party password-authenticated key exchange scheme with privacy preservation for mobile environment," *KSII Trans. Internet Inf. Syst.*, vol. 9, no. 12, pp. 5135–5149, 2015.
[22] J. Nam, J. Paik, J. Kim, Y. Lee, and D. Won, "Server-aided password-authenticated key exchange: From 3-party to group," in *Proc. Int. Conf. Human Interface Manage. Inf.*, Orlando, FL, USA, 2011, pp. 339–348.
[23] C. Guo, Z. Zhang, L. Zhu, Y.-A. Tan, and Z. Yang, "Scalable protocol for cross-domain group password-based authenticated key exchange," *Frontiers Comput. Sci.*, vol. 9, no. 1, pp. 157–169, 2014.
[24] A. Ometov, A. Levina, P. Borisenko, R. Mostovoy, A. Orsino, and S. Andreev, "Mobile social networking under side-channel attacks: Practical security challenges," *IEEE Access*, vol. 5, pp. 2591–2601, 2017.
[25] D. Moriyama and T. Okamoto, "Leakage resilient eCK-secure key exchange protocol without random oracles," in *Proc. ASIACCS*, Hong Kong, 2011, pp. 441–447.
[26] B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger security of authenticated key exchange," in *Proc. ProvSec*, Wollongong, NSW, Australia, 2007, pp. 1–16.
[27] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Proc. EUROCRYPT*, Innsbruck, Austria, 2001, pp. 453–474.
[28] J. Alawatugoda, C. Boyd, and D. Stebila, "Continuous after-the-fact leakage-resilient key exchange," in *Proc. ACISP*, Wollongong, NSW, Australia, 2014, pp. 258–273.
[29] J. Alawatugoda, D. Stebila, and C. Boyd, "Modelling after-the-fact leakage for key exchange," in *Proc. ASIA CCS*, Kyoto, Japan, 2014, pp. 207–216.
[30] J. Alawatugoda, D. Stebila, and C. Boyd, "Continuous after-the-fact leakage-resilient eCK-secure key exchange," in *Proc. IMA Int. Conf. Cryptogr. Coding*, Oxford, U.K., 2015, pp. 277–294.
[31] R. Chen, Y. Mu, G. Yang, W. Susilo, and F. Guo, "Strongly leakage-resilient authenticated key exchange," in *Proc. CT-RSA*, San Francisco, CA, USA, 2016, pp. 19–36.

[32] O. Ruan, Y. Zhang, M. Zhang, J. Zhou, and L. Harn, ''After-the-fact leakage-resilient identity-based authenticated key exchange,'' *IEEE Syst. J.*, to be published, doi: 10.1109/JSYST.2017.2685524.

[33] M. Toorani, ''On continuous after-the-fact leakage-resilient key exchange,'' in *Proc. 2nd Workshop Cryptogr. Secur. Comput. Syst.*, Amsterdam, The Netherlands, 2015, pp. 31–35.

[34] Z. Yang and S. Li, ''On security analysis of an after-the-fact leakage resilient key exchange protocol,'' *Inf. Process. Lett.*, vol. 116, no. 1, pp. 33–40, 2016.

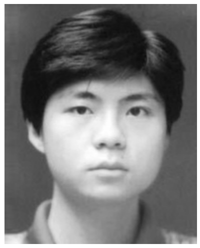
[35] H. Krawczyk. (Apr. 2008). *On Extract-Then-Expand Key Derivation Functions and an HMAC Based KDF*. [Online]. Available: http://webee.technion.ac.il/~hugo/kdf/kdf.pdf

[36] F. Davì, S. Dziembowski, and D. Venturi, ''Leakage-resilient storage,'' in *Proc. SCN*, Amalfi, Italy, 2010, pp. 121–137.

[37] S. Dziembowski and S. Faust, ''Leakage-resilient cryptography from the inner-product extractor,'' in *Proc. ASIACRYPT*, Seoul, South Korea, 2011, pp. 702–721.

[38] O. Ruan, M. Zhang, and J. Chen, ''Leakage-resilient password-based authenticated key exchange,'' in *Proc. Algorithms Archit. Parallel Process.*, Helsinki, Finland, 2017, pp. 285–296.

OU RUAN received the Ph.D. degree from the College of Information Security, School of Computer Science and Technology, Huazhong University of Science and Technology of China, in 2013. He is currently an Associate Professor with the School of Computer Sciences, Hubei University of Technology. His research interests include leakage-resilient cryptography, secure computations, and network security.



JING CHEN received the M.S. and Ph.D. degrees from the Huazhong University of Science and Technology, China. He is currently a Professor with the School of Computer Science, Wuhan University. His work focuses on wireless networks, network security, routing, and distributed resource management.

MINGWU ZHANG was a JSPS Post-Doctoral Fellow of the Japan Society of Promotion Sciences, Institute of Mathematics for Industry, Kyushu University, from 2010 to 2012. He is currently a Professor at with the School of Computer Sciences, Hubei University of Technology, where he is also the Director of Institute of Data Security and Privacy Preservation. His research interests include cryptography technology for networks, secure computations, and privacy preservations.