Journal of Ambient Intelligence and Humanized Computing (2019) 10:163–173
https://doi.org/10.1007/s12652-017-0628-8

ORIGINAL RESEARCH

Provably leakage-resilient three-party password-based authenticated key exchange

Ou Ruan[1] · **Qingping Wang**[1] · **Zihao Wang**[1]

Received: 13 September 2017 / Accepted: 10 November 2017 / Published online: 22 November 2017

Abstract
Three-party password-based authenticated key exchange (3PAKE) protocol is an important practical cryptographic primitive in the client-client communication environments, where two clients could generate a shared secure session key using their human-memorable passwords with a server's help. Many 3PAKE protocols were proposed, but these protocols were only secure in the traditional model where no leakage attacks exist. In Mobile Internet, Wireless Networks and Sensor Networks environments, 3PAKE systems are very vulnerable to side-channel attacks. Therefore, it is very necessary to design 3PAKE protocols that are secure in the leakage environments. However, there is no previous works for formalizing the security model for leakage-resilient (LR) 3PAKE and designing the LR 3PAKE protocols. In the paper, we first define a continuous after-the-fact LR eCK-security model for 3PAKE and propose a LR 3PAKE protocol, then present a formal security proof in the standard model.

Keywords Leakage-resilience · Password-based authenticated key exchange · Three-party setting · Provable security

1 Introduction

With the development of Intelligent Information Systems, people gradually move into intelligent life. Mobile Internet, Wireless Networks and Sensor Networks are the important infrastructures of Intelligent Information Systems. In these environments, there are many intelligent terminals such as smart phones. For example, in China, there were more than 656 million Mobile Internet users from the 38th Internet Statistics Report of China Internet Network Information Center. There are many kinds of applications for Intelligent Information Systems, including Wechat, Intelligent Transportation, Mobile payment and so on. The 2016 Internet Trends Report of KPCB showed that the number of Wechat monthly active users was more than 700 million. Lots of sensitive and private data are handled and transferred in these applications. When Wechating you hope that others besides your friends cannot learn your chatting messages, and you wish your bank accounts and passwords could be kept in secret when paying by mobile phones. Therefore, these applications must take measures to ensure communication security and data privacy. For example, Chen et al. (2016a) designed a secure Mobile Chat APP. In general, each user of these applications has a secret private password that is shared with a trusted server. When user A wants to communicate with user B securely, they first must generate a common cryptographically strong key for the session with the server's help. This model is called three-party password-based authenticated key exchange (3PAKE) protocol. There are lots of works for modelling and designing 3PAKE protocols. For a brief reviews, please refer to related works of traditional 3PAKE.

In Mobile Internet, Wireless Networks and Sensor Networks environments, because almost all the intelligent terminals are running in the public environment, there have many side-channel attacks (Hu et al. 2016; Chasaki and Mansour 2015) such as measuring electromagnetic emissions, calculating power consumption, observing keystroke gesture and so on. Thus, in order to ensure communication security and data privacy of the intelligent terminals, applications of these terminals not only should employ the secure protocols such as 3PAKE, but also must take measures to resist side-channel attacks. For 3PAKE, it's very vulnerable to side-channel attacks, because a very small leakage may completely expose the whole password. Therefore, it is very necessary to model and construct the leakage-resilient (LR)

✉ Ou Ruan
ruanou@163.com

[1] School of Computer Science and Technology, Hubei University of Technology, Wuhan, China

3PAKE protocols. However, there is no previous works for standardizing the security models and designing the LR 3PAKE protocols.

In the paper, we first define a continuous after-the-fact (AF) LR (CAFLR) eCK security model for 3PAKE, where the leakages are continuous and are permitted after the adversary asks the test challenger. Then, we propose a LR 3PAKE protocol based on key derivation function (KDF) (Krawczyk 2008), leakage-resilient storage (LRS) (Davì et al. 2010) and leakage-resilient refreshing of LRS. At last, we show a formal security proof in the standard model based on the new CAFLR eCK security model.

The rest of the paper is arranged by the following. Section 2 reviews related works. Section 3 presents the used cryptography tools. Section 4 introduces the λ-CAFLR eCK security model for 3PAKE protocol. Section 5 describes the new protocol and its provable security. Finally, Sect. 6 shows the conclusions and the future works.

2 Related works

2.1 Traditional 3PAKE

The first password-based authenticated key exchange (PAKE) protocol was introduced by Bellovin and Merritt (1992). Bellare et al. (2000) and Mackenzie et al. (2000) first proposed the provably secure PAKE protocols and formally proved the security in the random oracle (RO) model. Then, Jin et al. (2007) and Farash and Attari (2014b) gave some improvements and generalizations to provably PAKE protocols in the RO model. In 2006, Goldreich and Lindell (2006) first showed a PAKE protocol that was secure in the standard model. Then, some efficient constructions for PAKE protocols in the standard model were presented (Katz et al. 2009, 2012; Ran et al. 2012; Goyal 2012). Ou et al. (2015) proposed a mutual authenticated PAKE protocol that is called explicit PAKE.

More precisely, the above PAKEs are called two-party PAKE protocols (2PAKE) that are fit for the client–server environment. However, in the large-scale client–client communication environment, 2PAKE protocols are impractical because it is infeasible for each two clients having a shared password and for each client remembering all passwords of all his communicating partners. Thus, three-party PAKE protocol (3PAKE) is proposed where a trusted server helps two communicating clients to generate a secure session key and each client shares his secret password with the trusted server in some form. A lot of 3PAKE protocols have been developed in which three main approaches were used, that were (1) Utilizing the public key of the server (Xie et al. 2013; Xiong et al. 2013), (2) Utilizing symmetric keys shared by the sever and clients (Zhao and Gu 2012; Yang and Cao 2012) and (3) without public key or symmetric algorithms (Wu et al. 2012; Tso 2013; Pu et al. 2013; Farash and Attari 2014a; Wang et al. 2018).

2.2 LR authenticated key exchange (AKE)

The study of LR protocols has attracted many researchers, for example, Li and Zhang (2013) designed a LR identity-based encryption scheme. The first formal LR security model for AKE protocol was defined by Moriyama and Okamoto (2011) who named it λ-LR eCK security model. According to the proposed security model, they showed the first LR AKE protocol and its formal security proof. The eCK security model (Lamacchia et al. 2007) and the CK security model (Canetti and Krawczyk 2001) are two main used security model for AKE. In the CK model the adversary could corrupt users to get the secret private keys, and the adversary of the eCK security model is much stronger, he not only could do any things that the adversary of the CK security model can do, but also he could corrupt users to get the ephemeral random secret keys. One central limitation of Moriyama D et al.'s construction is that the leakage attacks are only permissible before the adversary receives the test challenge. Then, After-the-fact (AF) LR security model was presented by Alawatugoda et al. (2014a), where leakages could happen after the adversary receives the test challenge. In 2014, a CAFLR CK-secure AKE protocol was gave by Alawatugoda et al. (2014a), and a bounded AFLR (BAFLR) eCK-secure AKE protocol was shown by Alawatugoda et al. (2014b). In 2015, a CAFLR eCK-secure AKE protocol was proposed by Alawatugoda et al. (2015). In 2016, Chen et al. (2016b) defined a strong AFLR eCK security model and showed a one-round CAFLR eCK-secure AKE protocol. In the new security model, they first considered the leakage attacks to ephemeral random secret keys. In 2017, Ou et al. (2017) first proposed an AFLR eCK-secure identity-based AKE protocol. However, it is very surprising that there has no previous work for LR 3PAKE protocols.

3 Preliminaries

In this section, we describe the used primitives, such as Decision Diffie-Hellman (DDH) assumption, KDF, LRS and leakage-resilient refreshing of LRS.

Notation: Let $s \xleftarrow{\$} \Omega$ denote that s is picked uniformly from a finite set Ω at random.

Definition 1 (*Negligible function*) A negligible function $\varepsilon(k)$ means for each positive integer $c \geq 0$ there exists an integer k_c that $\varepsilon(k) < k^{-c}$ holds for each $k \geq k_c$.

Definition 2 (*Decision Diffie-Hellman (DDH) Assumption*) A distinguishing game is used to formally define DDH assumption:

1. A challenger $\boldsymbol{C}$ generates (G, g) and send them to an adversary $\boldsymbol{A}$, where G is a cyclic multiplicative group with a large prime order p and g is a random generator of G.
2. $\boldsymbol{C}$ randomly chooses $x, y, z \xleftarrow{\$} Z_p^*$ and $b \xleftarrow{\$} (0,1)$. If $b = 1, \boldsymbol{C}$ sends (g^x, g^y, g^{xy}) to $\boldsymbol{A}$, else $\boldsymbol{A}$ is given (g^x, g^y, g^z).
3. $\boldsymbol{A}$ outputs his guessed bit b', and $\boldsymbol{A}$ wins if $b' = b$.

DDH assumption means that

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

where $Adv_{DDH}(A)$ represents the advantage that $\boldsymbol{A}$ wins the above game and $\varepsilon(\cdot)$ is a negligible function.

Definition 3 ($\lambda - Leakage - ResilientStorage$) A λ-LRS includes two probabilistic polynomial time (PPT) algorithms (***Encode***, ***Decode***) and a bounded leakage parameter $\lambda = (\lambda_1, \lambda_2)$.

Encode: $Encode(s) = s_L \times s_R$, where s is an element chosen from the message space M, $s_L \times s_R$ is the encoded output element in the encoding space $L \times R$.

Decode: $Decode(s_L \times s_R) = s$.

A LRS must satisfy the following two properties:

I. Correctness of LRS. For each $s \xleftarrow{\$} M$, there has

$$Decode(Encode(s)) = s.$$

II. Security of LRS. A distinguishing game is shown as follows:

1. An adversary $\boldsymbol{A}$ picks two elements $(s_0, s_1) \xleftarrow{\$} M$ at random and sends (s_0, s_1) to a challenger $\boldsymbol{C}$.
2. $\boldsymbol{C}$ randomly selects a bit $b \xleftarrow{\$} (0,1)$ and generates $Encode(s_b) = s_b^L \times s_b^R$.
3. For each round $i = 1, \ldots, t$, $\boldsymbol{A}$ selects leakage functions $f = (f_i^L, f_i^R)$ and get the leakage $(f_i^L(s_b^L), f_i^R(s_b^R))$ back from $\boldsymbol{C}$, where the total leakage size should be bounded by (λ_1, λ_2),i.e., $\sum_1^t f_i^L(s_b^L) \le \lambda_1 \wedge \sum_1^t f_i^R(s_b^R) \le \lambda_2$.
4. $\boldsymbol{A}$ outputs his guessed bit b', and $\boldsymbol{A}$ wins if $b' = b$. The security of LRS means that

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

where $Adv_{LRS}(A)$ denotes the advantage of $\boldsymbol{A}$ in winning the above game and $\varepsilon(\cdot)$ is a negligible function.

Definition 4 ($(\lambda_{Refresh}, \lambda) - Leakage - Resilient\ RefreshingofLRS$) A leakage-resilient refreshing is a PPT algorithm ***Refresh*** with 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 LRS should satisfy the following two properties:

I. Correctness of leakage-resilient refreshing. For each $s \xleftarrow{\$}, M$ there has

$$Decode(s_L' \times s_R') = D \text{ e } code(s_L \times s_R).$$

II. $(\lambda_{Refresh}, \lambda)$-Security of leakage-resilient refreshing. A distinguishing game is shown as follows:

1. An adversary $\boldsymbol{A}$ picks two elements $(s_0, s_1) \xleftarrow{\$} M$ at random and sends (s_0, s_1) to a challenger $\boldsymbol{C}$.
2. $\boldsymbol{C}$ randomly selects a bit $b \xleftarrow{\$} (0,1)$ and generates $Encode(s_b) = s_{bL}^0 \times s_{bR}^0$.
3. For each $i = 1, \ldots, \ell$. $\boldsymbol{A}$ selects the iith round leakage functions $f_{Refresh\text{-}i} = (f_{Refresh\text{-}i}^L, f_{Refresh\text{-}i}^R)$ and gets back the leakages $(f_{Refresh\text{-}i}^L(s_{bL}^i), f_{Refresh\text{-}i}^R(s_{bR}^i))$ from $\boldsymbol{C}$, where $f_{Refresh\text{-}i}^L(s_{bL}^i) \le \lambda_{Refresh1} \wedge f_{Refresh\text{-}i}^R(s_{bR}^i) \le \lambda_{Refresh2}$; then, $\boldsymbol{C}$ refreshes the encodings,

$$Refresh(s_{bL}^{i-1} \times s_{bR}^{i-1}) = s_{bL}^i \times s_{bR}^i,$$

4. $\boldsymbol{A}$ outputs his guessed bit b', and $\boldsymbol{A}$ wins if $b' = b$. The $(\lambda_{Refresh}, \lambda)$-security of leakage-resilient refreshing means that

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

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

Definition 5 *(Dziembowski-Faust (DF) LRS Scheme)* Suppose s $\in (Z_p^*)^m$ is a secret value with any m $\in$ N.

Encode: Choose a random $s_L \xleftarrow{\$} (Z_p^*)^n \backslash \{(0^n)\}$ and generate $s_R \in (Z_p^*)^{n \times m}$ such that $s_L \times s_R = s$, where n $\in$ N. Output (s_L, s_R).

Decode: $Decode(s_L \times s_R) = s$.

Lemma 1 (Dziembowski and Faust 2011) *If* $m < n/20$, *Definition 5 is a λ-secure LRS scheme with* λ=(0.3 $\cdot n \cdot$ log p, 0.3 $\cdot n \cdot \log p$), *named* $\Phi_{Z_p^*}^{n,m}$.

Lemma 2 (Dziembowski and Faust 2011) *If* $m/3 \leq n \wedge n \geq 16$, *there has a* $(\lambda/2, \lambda)$*-secure leakage-resilient refreshing* $Refresh_{Z_p^*}^{n,m}$ *for* $\Phi_{Z_p^*}^{n,m}$, *where* $\Phi_{Z_p^*}^{n,m}$ *is a* λ*-secure DF-LRS.*

Definition 6 (*key derivation function*) KDF is a PPT algorithm that is used to compute a secret key with inputs (σ, ℓ, r, c), i.e., $k = \text{KDF}(\sigma, \ell, r, c)$, where σ denotes the source material of k, ℓ is some public knowledge about σ such as its length, r is a salt value and c represents a context variable.

Security of KDF. A distinguishing game is defined as follows:

1. The challenger $\boldsymbol{C}$ chooses (σ, ℓ) and sends them to an adversary $\boldsymbol{A}$.
2. $\boldsymbol{A}$ randomly selects a value c and a salt value r.
3. $\boldsymbol{C}$ picks a random bit $b \xleftarrow{\$} (0, 1)$. If $b = 1$, $\boldsymbol{C}$ calculates k=KDF(σ, ℓ, r, c), else $\boldsymbol{C}$ picks a string s at random, and then give it to $\boldsymbol{A}$, where the length of s and k is equal.
4. $\boldsymbol{A}$ outputs his guessed bit b', and $\boldsymbol{A}$ wins if $b' = b$.

The security of KDF means that

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

where $Adv_{KDF}(A)$ denotes the advantage of $\boldsymbol{A}$ in winning the above game and $\varepsilon(\cdot)$ is a negligible function.

4 The λ-CAFLR security model for 3PAKE

This section formally defines the λ-CAFLR security model for 3PAKE. The new model follows the only computation leakage (OCL) model, which assumes that leakage only occurs in the calculations associated with the secret password. In theλ-CAFLR security model an adversary $\boldsymbol{A}$ could continuously get arbitrarily leakages of the secret password, but for each instantiation of the protocol the amount of leakage is bounded by λ. In each instantiation, $\boldsymbol{A}$ could adaptively select any PPT leakage functions $f = (f_1, \ldots, f_n)$ to obtain leakage of the password, and the overall amount of leakages is bounded by λ, i.e., $\sum |f_i(pw)| \leq \lambda$. After receiving a leakage function f_i chosen by $\boldsymbol{A}$, $\boldsymbol{A}$ will be given the leakage $f_i(pw)$ of the long-term secret password.

4.1 System framework

The typical system model of 3PAKE protocols is shown in Fig. 1, in which has a server S and two clients A, B who seek to generate a shared session key k with the help of S. In this model, k is only learnt by two clients, and is not known by others including the server. S is semi-honest but

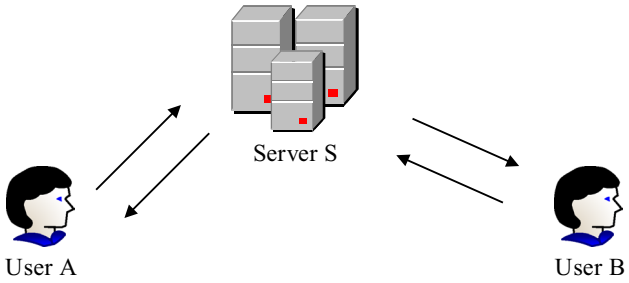

Fig. 1 System model

curious, who will do exactly based on the protocol specifications but will collect and analyze any message he can get. Thus, the adversary $\boldsymbol{A}$ cannot get the users' passwords from S, but he can activate leakage attacks.

Notations in the system framework:

Principal: is a party involved into a protocol instance. Each principal is either a client $u \in U$ or a semi-honest server $s \in S$. We assume there has only a single server in the set S. The client who activates the protocol is called the initiator principal. Each client principal has a password pw, computes $H(pw)$ and gives it to S. Thus, S holds a table $H(pw_s) =< H(pw_s[u]) >_{u \in U}$ with a record for each client.

Session: represent a protocol instance with principals.

Oracle $\Pi_{U,(S,V)}^t$: is the client principal U interacting with the server S and the intended principal V in the tth session.

Oracle $\Pi_{S,(U,V)}^t$: is the server principal S in the tth session who helps the client U and V to generate a shared session key.

4.2 Adversarial powers

Adversarial powers are modelled by the following queries:

SendClient$(\Pi_{U,(S,V)}^t, m, f)$ query: Upon receiving ***SendClient*** query with a message m and a leakage function f, $\Pi_{U,(S,V)}^t$ of tth the session will generate a normal protocol message produced based on the protocol specifications and the leakage $f(pw_U)$ of the long-term password, and send them to the adversary $\boldsymbol{A}$. $\boldsymbol{A}$ can activate a new protocol instance by asking ***SendClient***$(\Pi_{U,(S,V)}^t, (\text{start}),( ))$ to the initiator principal.

SendServer$(\Pi_{S,(U,V)}^t, m, f_1, f_2)$ query: Upon receiving ***SendServer*** query with a message m and the leakage functions f_1 and f_2, $\Pi_{S,(U,V)}^t$ of the tth session will generate the normal protocol messages produced based on the protocol specifications and the leakage $sf_1(pw_U)$ and $f_2(pw_V)$, and send them to $\boldsymbol{A}$.

RevealSessionKey($\Pi^t_{U,(S,V)}$) query: $\Pi^t_{U,(S,V)}$ gives the session key of the tth session to $\boldsymbol{A}$.

RevealClientEphemeralKey($\Pi^t_{U,(S,V)}$) query: $\Pi^t_{U,(S,V)}$ gives his random ephemeral key of the tth session to $\boldsymbol{A}$.

RevealServerEphemeralKey($\Pi^t_{S,(U,V)}$) query: $\Pi^t_{S,(U,V)}$ gives his random ephemeral key of the tth session to $\boldsymbol{A}$.

Corrupt($\Pi^t_{U,(S,V)}$) query: $\Pi^t_{U,(S,V)}$ gives his secret password pw_U to $\boldsymbol{A}$.

Test($\Pi^t_{U,(S,V)}$) query: Upon receiving a ***Test*** query, the challenger randomly chooses a bit $b \xleftarrow{\$} (0,1)$, if $b = 1$ then $\boldsymbol{A}$ is given the actual session key, while a random key is given to $\boldsymbol{A}$.

4.3 λ-CAFLR security model

In the λ-CAFLR security model, the total leakage amount of the secret password are bounded by the parameter λ, i.e., $\sum |f_i(pw)| \leq \lambda$.

Definition 7 (*Partners in CAFLR eCK security model*) Two oracles $\Pi^t_{U,(S,V)}$ and $\Pi^{t'}_{U',(S,V')}$ are called partners if the followings satisfy:

1. Two oracles $\Pi^t_{U,(S,V)}$ and $\Pi^{t'}_{U',(S,V')}$ have produced a same session key;
2. $U = V'$ and $V = U'$;
3. $t = t'$.

Definition 8 (λ-*CAFLR-freshness*) Assume $f = (f_1, \ldots, f_n)$ be n arbitrary PPT leakage functions for an instantiation of the protocol selected by the adversary $\boldsymbol{A}$. An oracle $\Pi^t_{U,(S,V)}$ is λ-CAFLR-fresh if the followings satisfy:

1. The oracle $\Pi^t_{U,(S,V)}$ or its partner, $\Pi^t_{V,(S,U)}$ (if it exists) has not been queried a ***RevealSessionKey***.
2. If the partner $\Pi^t_{V,(S,U)}$ exists, none of the followings has been queried:
   (a) ***Corrupt***($\Pi^t_{U,(S,V)}$) and ***RevealClientEphemeralKey***($\Pi^t_{U,(S,V)}$).
   (b) ***Corrupt*** ($\Pi^t_{V,(S,U)}$) and ***RevealClientEphemeralKey***($\Pi^t_{V,(S,U)}$).
   (c) ***RevealClientEphemeralKey***($\Pi^t_{U,(S,V)}$), ***RevealClientEphemeralKey***($\Pi^t_{V,(S,U)}$) and ***RevealServerEphemeralKey***($\Pi^t_{S,(U,V)}$).
3. If the partner $\Pi^t_{V,(S,U)}$ does not exist, none of the followings has been queried:
   (a) ***Corrupt*** ($\Pi^t_{U,(S,V)}$) and ***RevealClientEphemeralKey***($\Pi^t_{U,(S,V)}$).
   (b) ***Corrupt*** ($\Pi^t_{V,(S,U)}$) .
   (c) ***RevealClientEphemeralKey***($\Pi^t_{U,(S,V)}$) and ***RevealServerEphemeralKey***($\Pi^t_{S,(U,V)}$).
4. For all ***SendClient*** (.,U, ., .,f_i) queries, $\sum |f_i(pw_U)| \leq \lambda$.
5. For all ***SendClient*** (.,V, ., .,f_i) queries, $\sum |f_i(pw_V)| \leq \lambda$.
6. For all ***SendServer*** (.,S, ., .,f_{1i}, f_{2i}) queries, $\sum |f_{1i}(pw_U)| \leq \lambda \wedge \sum |f_{2i}(pw_V)| \leq \lambda$.

Definition 9 (λ-*CAFLR security game*) λ–CAFLR security game is as follows:

1. An adversary $\boldsymbol{A}$ asks any of ***SendClient***, ***SendServer***, ***RevealSessionKey***, ***RevealClientEphemeralKey***, ***RevealServerEphemeralKey*** and ***Corrupt*** to any oracle as he wants.
2. $\boldsymbol{A}$ chooses a λ-CAFLR-fresh oracle and asks a ***Test*** query. Upon getting a ***Test*** query, the challenger $\boldsymbol{C}$ randomly selects a bit $b \xleftarrow{\$} (0,1)$, if $b = 1$ then $\boldsymbol{A}$ is given the actual session key, while a random key is given to $\boldsymbol{A}$.
3. $\boldsymbol{A}$ continues asking ***SendClient***, ***SendServer***, ***RevealSessionKey***, ***RevealClientEphemeralKey***, ***RevealServerEphemeralKey*** and ***Corrupt***. All these queries should not violate the λ-CAFLR -freshness of the test oracle.
4. $\boldsymbol{A}$ outputs his guessed bit b', and $\boldsymbol{A}$ wins if $b' = b$.

Definition 10 (λ -*CAFLR security*) λ–CAFLR security means that

$$Adv^{\lambda-CAFLR}_{3\text{PAKE}} = |\Pr[b' = b] - 1/2| = N_S/N + \varepsilon(\cdot),$$

where $Adv^{\lambda-CAFLR}_{3PAKE}$ represents the advantage that $\boldsymbol{A}$ wins λ -CAFLR security game in Definition 9, N_S is the number of sessions on a client principal, N denotes the size of the password dictionary that is shared by all clients, and $\varepsilon(\cdot)$ is a negligible function.

In 3PAKE protocols, the on-line dictionary attack is unavoidable, and N_S/N is the success probability of the on-line dictionary attack. Thus, a λ-CAFLR secure 3PAKE protocol means that there hasn't any PPT adversary that could win the above game with an advantage more than N_S/N. There are many ways to limit the on-line dictionary attack, one of the most common methods is using a policy that blocks using a password if failed attempts have happened several times.

5 A new λ-CAFLR 3PAKE secure protocol

5.1 The proposed protocol

Figure 2 shows the proposed protocol, which includes the following two stages:

The initial setup stage:

Users U and V respectively compute s_U=$H(pw_U)$, $s_V = H(pw_V)$, and send s_U/s_V to the server S, in which s_U/s_V is an element of the group G. We suppose that these calculations are secretly computed and there hasn't any leakage attack. Then, User U , V and the server S run a λ-secure DF-LRS scheme $\Phi^{n,1}_{Z^*_p}$ respectively, pick $u^0_L \xleftarrow{\$} (Z^*_p)^n \backslash \{(0^n)\}$, $v^0_L \xleftarrow{\$} (Z^*_p)^n \backslash \{(0^n)\}$, and $su^0_L \xleftarrow{\$} (Z^*_p)^n \backslash \{(0^n)\}$, $sv^0_L \xleftarrow{\$} (Z^*_p)^n \backslash \{(0^n)\}$ at random and generates $a^0_R \in (Z^*_p)^{n\times 1}$, $b^0_R \in (Z^*_p)^{n\times 1}$ and $su^0_R \in (Z^*_p)^{n\times 1}$, $sv^0_R \in (Z^*_p)^{n\times 1}$such that $u^0_L \cdot u^0_R = s_U, v^0_L \cdot v^0_R = s_V$ and $su^0_L \cdot su^0_R = s_U, sv^0_L \cdot sv^0_R = s_V$.

The protocol execution stage:

Step 1. User U picks a number $x \xleftarrow{\$} Z^*_p$ at random, calculates $X = g^x, T_{U1} = g^{u^i_L}, T_{U2} = (T_{U1})^{u^i_R}$ and $T_U = X \cdot T_{U2}$, then sends (U, V, T_U) to S.

Step 2. Upon receiving (U, V, T_U), the server chooses a number $z \xleftarrow{\$} Z^*_p$ at random, computes

$$\begin{aligned} z \xleftarrow{\$} Z^*_p, T_{SU1} &= g^{su^i_L}, T_{SU2} = (T_{SU1})^{su^i_R}, \\ T_{SV1} &= g^{sv^i_L}, T_{SV2} = (T_{SV1})^{sv^i_R}, \\ X_S &= T_U/T_{SU2}, T_{SV} = (X_S)^z \cdot T_{SV2}, \end{aligned}$$

then sends (U, V, T_{SV}) to V.

Step 3. Upon receiving (U, V, T_{SV}), user V randomly picks a number $y \xleftarrow{\$} Z^*_p$, computes $Y = g^y, T_{V1} = g^{v^i_L}, T_{V2} = (T_{V1})^{v^i_R}$ and $T_V = Y \cdot T_{V2}$, then sends (U, V, T_V) to S.

Step 4. Upon receiving (U, V, T_V), S computes $Y_S = T_V/T_{SV2}, T_{SU} = (Y_S)^z \cdot T_{SU2}$, then sends (U, V, T_{SU}) to U.

Step 5. Two users generate the session key and refresh the store encodings. U computes $K_U = (T_{SU}/T_{U2})^x$ and $k_{uv} = KDF(U, V, K_U)$, then refresh the store pieces with

$$(u^{i+1}_L, u^{i+1}_R) \leftarrow \text{Refresh}^{n,1}_{Z^*_p}(u^i_L, u^i_R).$$

And, V also computes $K_V = (T_{SV}/T_{V2})^y$ and $k_{uv} = KDF(U, V, K_V)$, then refresh the store pieces with

$$(v^{i+1}_L, v^{i+1}_R) \leftarrow \text{Refresh}^{n,1}_{Z^*_p}(v^i_L, v^i_R).$$

Correctness of the proposed protocol

Since,

$$\begin{aligned} &T_{U2} = (g)^{su^i_L \cdot su^i_R} = g^{s_U} = T_{SU2} = (g)^{su^i_L \cdot su^i_R} = g^{s_U} \\ &T_{V2} == g^{s_V} = T_{SV2} \\ &Y_S = T_V/T_{SV2} = Y \cdot T_{V2}/T_{SV2} = Y \\ &K_U = (T_{SU}/T_{U2})^x = ((Y_S)^z \cdot T_{SU2}/T_{U2})^x = (Y_S)^{xz} = Y^{xz} = g^{xyz} \end{aligned}$$

By the same way, we get

$K_V = g^{xyz}$.

Thus, $KDF(U, V, K_U) = KDF(U, V, K_V)$, i.e., the proposed protocol is correct.

5.2 Security proof

Theorem 1 If DDH assumption is hard, the leakage-resilient refreshing of LRS is $(2\lambda, \lambda)$-secure and KDF is secure, the new 3PAKE protocol is λ-CAFLR eCK-secure, i.e.,

$$\begin{aligned} Adv^{\lambda-CAFLR}_{3PAKE} \leq & N_S/N + N^2_P N^2_S (Adv_{DDH} + Adv_{Refresh-LRS} \\ & + Adv_{KDF}), \end{aligned}$$

where $Adv^{\lambda-CAFLR}_{3PAKE}$ denotes the advantage of an adversary $\boldsymbol{A}$ in winning the λ CAFLR security game of the proposed protocol Adv_{DDH}, Adv_{KDF}, $Adv_{Refresh-LRS}$ represent advantages of $\boldsymbol{A}$ in winning the security game of DDH, KDF and leakage-resilient refreshing of LRS, respectively, and N_P is the number of protocol client principals, N_S denotes the number of sessions on a client principal, N is the password dictionary's size.

Our formal proof is based on the game hopping technique. First, we give a sequence of games in which Game 1 is the original λ-CAFLR security game and the advantages of the last Game is negligible; Second, we show that each game is not distinguished from its previous game. Thus, we get that the advantages of the original λ-CAFLR security game is negligible.

Proof . The proof could be divided into two main cases: (1) a partner to the test oracle exists, and (2) it does not exist.

Case 1. A partner to the test oracle exists

In this case, the adversary $\boldsymbol{A}$ is a passive adversary who only collects the protocol messages. We split its proofs into three sub cases as follows:

1. $\boldsymbol{A}$ asks ***corrupt***$(\Pi^t_{U,(S,V)})$ and ***corrupt***$(\Pi^t_{V,(S,U)})$ queries. In this case, $\boldsymbol{A}$ could get the long-term secret password pw_U and pw_V.
2. $\boldsymbol{A}$ asks one of ***corrupt***$(\Pi^t_{U,(S,V)})$ and ***corrupt***$(\Pi^t_{V,(S,U)})$ queries. In this case, $\boldsymbol{A}$ could not get one long-term secret password of two principles.
3. $\boldsymbol{A}$ asks none of ***corrupt***$(\Pi^t_{U,(S,V)})$ and ***corrupt***$(\Pi^t_{V,(S,U)})$ queries. In this case, $\boldsymbol{A}$ could not get any long-term secret passwords of two principles.

Case 1.1 $\boldsymbol{A}$ **asks** ***corrupt***$(\Pi^t_{U,(S,V)})$ **and** ***corrupt***$(\Pi^t_{V,(S,U)})$ **queries**

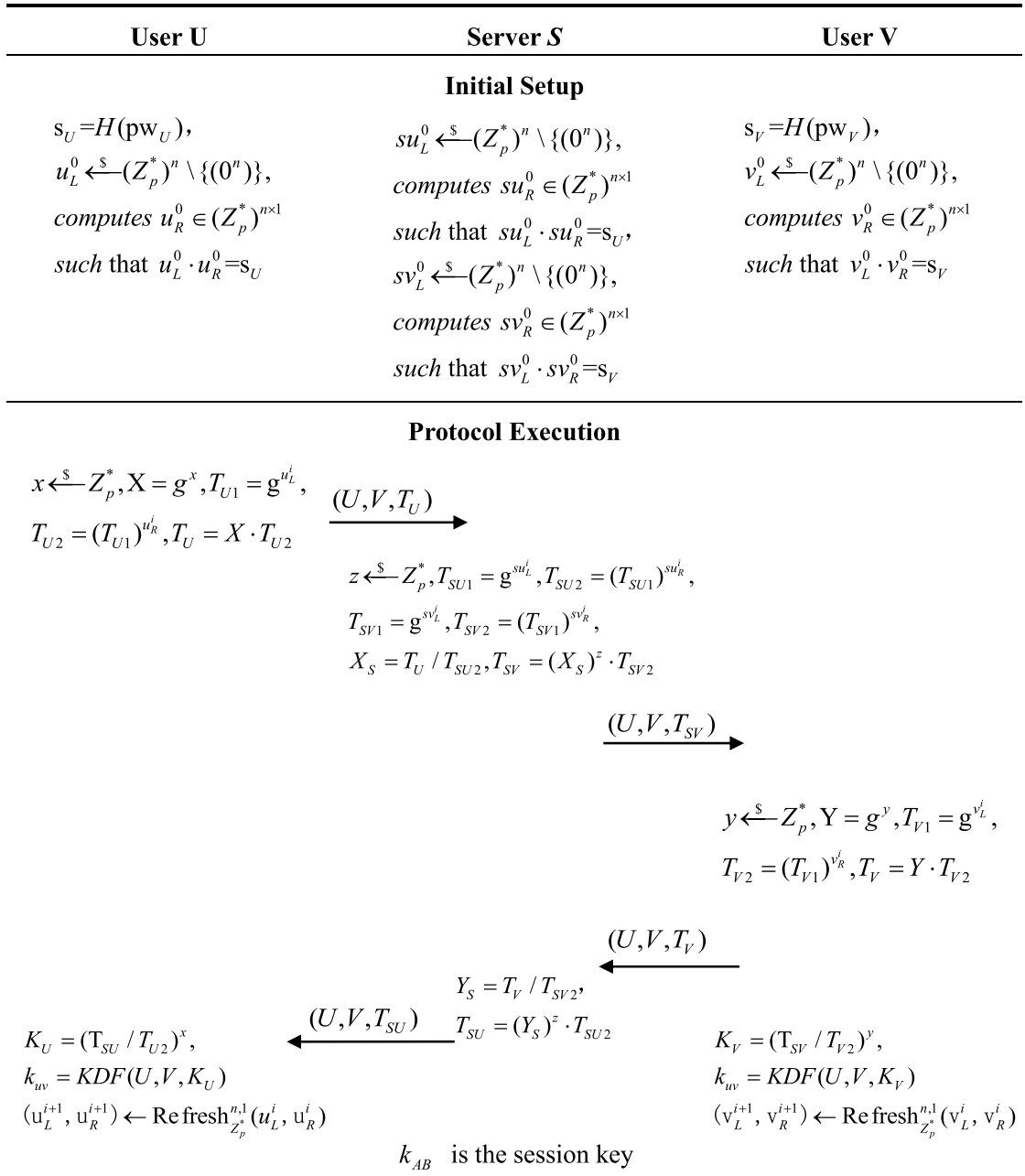

Fig. 2 The LR 3PAKE protocol

In this case, ***A*** could get the long-term secret passwords pw_U and pw_V by ***corrupt***$(\Pi^t_{U,(S,V)})$ and ***corrupt***$(\Pi^t_{V,(S,U)})$ queries, thus, leakage attacks don't need to consider. And ***A*** could learn the server's random key z by ***RevealServerEphe- meralKey***$(\Pi^t_{S,(U,V)})$, but ***A*** could not ask queries ***RevealClientEphemeralKey***$(\Pi^t_{U,(S,V)})$ and ***RevealClientEphemeralKey***$(\Pi^t_{V,(S,U)})$ in order not to violate λ -CAFLR-freshness of ***Test*** oracle.

Game 1: This is the original λ-CAFLR security game.

Game 2: Game 2 and Game 1 only have the following differences: ***A*** selects two different client principals $U^*, V^* \xleftarrow{\$} \{u_1, \ldots, u_{N_p}\}$ and two numbers $t^*, r^* \xleftarrow{\$} \{1, \ldots, N_s\}$ at random. Then, ***A*** begin to activate Game 2 and chooses the oracle $\Pi^{t^*}_{U^*,(S,V^*)}$ as the target oracle and $\Pi^{r^*}_{V^*,(S,U^*)}$ as the partner oracle. If the test oracle is not $\Pi^{t^*}_{U^*,(S,V^*)}$ or the partner

oracle is not $\Pi^{r^*}_{V^*,(S,U^*)}$, Game 2 challenger $\boldsymbol{C}$ exists and terminates Game 2.

Game 3: Game 3 and Game 2 only have the following differences: $\boldsymbol{C}$ calculates $k_{U^*V^*} = KDF(U^*, V^*, (g^z)^r)$ where $r \xleftarrow{\$} Z_p^*$. Then, upon receiving a $\boldsymbol{Test}(\Pi^{t^*}_{U^*,(S,V^*)})$ or $\boldsymbol{Test}(\Pi^{r^*}_{V^*,(S,U^*)})$ query, $\boldsymbol{C}$ gives k_{U^*V*} to $\boldsymbol{A}$.

Game 4: Game 4 and Game 3 only have the following differences: $\boldsymbol{C}$ selects a random key $k_{U^*V^*} \xleftarrow{\$} \{0,1\}^k$. Then, upon getting a $\boldsymbol{Test}(\Pi^{t^*}_{U^*,(S,V^*)})$ or $\boldsymbol{Test}(\Pi^{r^*}_{V^*,(S,U^*)})$ query, $\boldsymbol{C}$ gives $k_{U^*V^*}$ to $\boldsymbol{A}$.

Differences between games: The followings show that each game t is not distinguished from its previous game $t-1$. Let $Adv_{Game t}(A)$ be the advantage that $\boldsymbol{A}$ wins Game t.

Game 1: In the original game, there has

$$Adv_{Game1}(A) = Adv^{\lambda-CAFLR}_{3PAKE} \quad \text{(I)}$$

Game 1 and Game 2: If the test oracle is $\Pi^{t^*}_{U^*,(S,V^*)}$ and the partner oracle is $\Pi^{r^*}_{V^*,(S,U^*)}$, Game 2 is consistent with Game 1. The probability that $\boldsymbol{A}$ correctly selects a test session and a partner is $1/N_P^2N_S^2$. Therefore

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

Game 2 and Game 3: In Game 2 $k_{U^*V^*} = KDF(U^*, V^*, (g^z)^{xy})$, while in Game 3 $k_{U^*V^*} = KDF(U^*, V^*, (g^z)^r)$. From DDH assumption there has

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

Game 3 and Game 4: In Game 3 $k_{U^*V^*} = KDF(U^*, V^*, (g^z)^r)$, while $k_{U^*V^*} \xleftarrow{\$} \{0,1\}^k$ in Game 4. Because KDF is secure, there has

$$|Adv_{Game3}(A) - Adv_{Game4}(A)| \le Adv_{KDF} \quad \text{(IV)}$$

Game 4: In Game 4 the session key $k_{U^*V^*}$ is a random string that doesn't depends on any information. Therefore

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

Using equations (I)–(V) we get

$$Adv^{\lambda-CAFLR}_{3PAKE} \le N_P^2N_S^2(Adv_{KDF} + Adv_{DDH}).$$

Case 1.2 $\boldsymbol{A}$ ask one of $\boldsymbol{corrupt}(\Pi^t_{U,(S,V)})$ and $\boldsymbol{corrupt}(\Pi^t_{V,(S,U)})$ queries

For simplify, suppose $\boldsymbol{A}$ asks $\boldsymbol{corrupt}(\Pi^t_{U,(S,V)})$ query.

In this case, $\boldsymbol{A}$ can learn the long-term secret password pw_U by $\boldsymbol{corrupt}(\Pi^t_{U,(S,V)})$ query, and learn the random key y and z by $\boldsymbol{RevealClientEphemeralKey}(\Pi^t_{V,(S,U)})$ and $\boldsymbol{RevealServerEphemeralKey}(\Pi^t_{S,(U,V)})$, but $\boldsymbol{A}$ could not ask queries $\boldsymbol{RevealClientEphemeralKey}(\Pi^t_{U,(S,V)})$ in order not to violate λ -CAFLR-freshness of $\boldsymbol{Test}$ oracle.

Game 1: It is the original game.

Game 2: Consistent with Game 2 in Case 1.1.

Game 3: Game 3 and Game 2 only have the following differences: $\boldsymbol{C}$ picks a random $s_{V^*} \xleftarrow{\$} Z_p^*$, encodes $(v_L^0, v_R^0) = Encode(s_{V^*})$ and $(sv_L^0, sv_R^0) = Encode(s_{V^*})$, and continues refreshing the two encodings, then uses them to simulate the answers to $\boldsymbol{A}$'s leakage function of the client principal V* and the server S.

Game 4: Game 4 and Game 3 only have the following differences: $\boldsymbol{C}$ calculates $k_{U^*V^*} = KDF(U^*, V^*, (g^{yz})^{x'})$ where $x' \xleftarrow{\$} Z_p^*$. Upon receiving a $\boldsymbol{Test}(\Pi^{t^*}_{U^*,(S,V^*)})$ or

$\boldsymbol{Test}(\Pi^{r^*}_{V^*,(S,U^*)})$ query, $\boldsymbol{C}$ gives $k_{U^*V^*}$ to $\boldsymbol{A}$.

Game 5: Consistent with Game 4 in Case 1.1.

Differences between games:

Game 1:

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

Game 1 and Game 2: Consistent with Game 1 and Game 2 in Case 1.1.

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

Game 2 and Game 3: In Game 2 the leakage of the password pw_V is the real leakage of $s_V = H(pw_V)$, while the leakage is a leakage of a random value s_{V^*}in Game 3. Because the leakage-resilient refreshing of LRS is secure, there has

$$|Adv_{Game2}(A) - Adv_{Game3}(A)| \le Adv_{Refresh-LRS} \quad \text{(III)}$$

Game 3 and Game 4: In Game 2 $k_{U^*V^*} = KDF(U^*, V^*, (g^{yz})^x)$, while in Game 3 $k_{U^*V^*} = KDF(U^*, V^*, (g^{yz})^{x'})$. Because x' is chosen at random and independent on x, $(g^{yz})^x$ and $(g^{yz})^{x'}$ are perfectly indistinguishable. Therefore,

$$|Adv_{Game3}(A) - Adv_{Game4}(A)| = 0 \quad \text{(IV)}$$

Game 4 and Game 5: Consistent with Game 3 and Game 4 in Case 1.1.

$$|Adv_{Game4}(A) - Adv_{Game5}(A)| \le Adv_{KDF} \quad \text{(V)}$$

Game 5: Consistent with Game 4 in Case 1.1.

$$Adv_{Game5}(A) = 0 \quad \text{(VI)}$$

Using equations (I)–(VI) we get

$$Adv^{\lambda-CAFLR}_{3PAKE} \le N_P^2N_S^2(Adv_{Refresh-LRS} + Adv_{KDF}).$$

Case 1.3 $\boldsymbol{A}$ ask none of $\boldsymbol{corrupt}(\Pi^t_{U,(S,V)})$ and $\boldsymbol{corrupt}(\Pi^t_{V,(S,U)})$ queries

In this case, $\boldsymbol{A}$ could get the random keys x and y by $\boldsymbol{RevealClientEphemeralKey}(\Pi^t_{U,(S,V)})$ and $\boldsymbol{RevealCli-}$

entEphe-meralKey($\Pi^t_{V,(S,U)}$), but $\boldsymbol{A}$ could not ask ***Reveal-ServerEph-emeralKey*** ($\Pi^t_{S,(U,V)}$) query in order not to violate λ-CAFLR-freshness of ***Test*** oracle.

Game 1: It is the original game.

Game 2: Consistent with Game 2 in Case 1.1.

Game 3: Game 3 and Game 2 only have the following differences: $\boldsymbol{C}$ picks $s_{V^*} \xleftarrow{\$} Z_p^*$ encodes (v_L^0, v_R^0)=$Encode(s_{V^*})$ and (sv_L^0, sv_R^0)=$Encode(s_{V^*})$, and continues refreshing the two encodings, then uses them to simulate the answers to $\boldsymbol{A}$'s leakage function of the principal V* and the server S. By the same way, $\boldsymbol{C}$ picks $s_{U^*} \xleftarrow{\$} Z_p^*$at random, encodes it and continues refreshing them, then simulate the answers to $\boldsymbol{A}$'s leakage function of the principal U* and the server S.

Game 4: Game 4 and Game 3 only have the following differences: $\boldsymbol{C}$ generates $k_{U^*V^*} = KDF(U^*, V^*, (g^{xy})^{z'})$ where$z' \xleftarrow{\$} Z_p^*$. Upon receiving a ***Test***($\Pi^{t^*}_{U^*,(S,V^*)}$) or ***Test***($\Pi^{r^*}_{V^*,(S,U^*)}$) query, $\boldsymbol{C}$ gives $k_{U^*V^*}$ to $\boldsymbol{A}$.

Game 5: Consistent with Game 4 in Case 1.1.

Differences between games:

The analyze is almost consistent with Case 1.2, and there has

$$Adv^{\lambda-CAFLR}_{3PAKE} \leq N_P^2 N_S^2 (Adv_{Refresh-LRS} + Adv_{KDF}).$$

Case 2 A partner oracle to the test oracle does not exist.

In this case, $\boldsymbol{A}$ is an active adversary. He may masquerade as the intended partner principal V and run the protocol with the test oracle $\Pi^t_{U,(S,V)}$. Therefore, $\boldsymbol{A}$ could not ask a ***corrupt*** ($\Pi^t_{V,(S,U)}$) query to get the password of V. But, $\boldsymbol{A}$ can ask the ***corrupt***($\Pi^t_{U,(S,V)}$) query to get the password of U. We spilt its proof into two sub cases as follows:

1. $\boldsymbol{A}$ asks ***corrupt***($\Pi^t_{U,(S,V)}$) query. In this case, $\boldsymbol{A}$ could get the secret password pw_U.
2. $\boldsymbol{A}$ doesn't ask ***corrupt***($\Pi^t_{U,(S,V)}$). In this case, $\boldsymbol{A}$ could not get the secret password pw_U.

Case 2.1 $\boldsymbol{A}$ asks a *corrupt*($\Pi^t_{U,(S,V)}$) query

In this case, $\boldsymbol{A}$ can learn the long-term secret password pw_U by ***corrupt***($\Pi^t_{U,(S,V)}$) query, and learn the random key y and z by ***RevealClientEphemeralKey***($\Pi^t_{V,(S,U)}$) and ***Reveal-ServerEphemeralKey***($\Pi^t_{S,(U,V)}$), but $\boldsymbol{A}$ could not ask queries ***RevealClientEphemeralKey***($\Pi^t_{U,(S,V)}$) in order not to violate λ-CAFLR-freshness of ***Test*** oracle.

Game 1: It is the original game.

Game 2: Game 2 and Game 1 only have the following differences: $\boldsymbol{A}$ selects a password pw'_V, computes s'_V=$H(pw'_V)$, encodes it, then uses the encodings of s'_V to generate the message based on the protocol specifications.

Game 3: Game 3 and Game 2 only have the following differences: $\boldsymbol{A}$ first chooses two different principals $U^*, V^* \xleftarrow{\$} \{u_1, \ldots, u_{N_p}\}$ and two numbers $t^*, r^* \xleftarrow{\$} \{1, \ldots, N_s\}$ at random. Then, $\boldsymbol{A}$ begin to activate Game 3 and chooses the oracle $\Pi^{t^*}_{U^*,(S,V^*)}$ as the target oracle. If the test oracle is not $\Pi^{t^*}_{U^*,(S,V^*)}$ $\boldsymbol{C}$ exits and terminates Game 3.

Game 4: Consistent with Game 3 in Case 1.2.

Game 5: Consistent with Game 4 in Case 1.2.

Game 6: Consistent with Game 4 in Case 1.1.

Differences between games

Game 1: $Adv_{Game1}(A) = Adv^{\lambda-CAFLR}_{3PAKE}$ (I)

Game 1 and Game 2: if $pw'_V = pw_V$, Game 2 is consistent with Game 1, otherwise Game 2 is independent on Game 1. The probability that $pw'_V = pw_V$ is N_s/N. Therefore

$$|Adv_{Game2}(A) - Adv_{Game1}(A)| = \frac{N_s}{N} \quad \text{(II)}$$

Game 2 and Game 3: The analysis is consistent with Game 1 and Game 2 in Case 1.1.

$$Adv_{Game3}(A) = \frac{1}{N_P^2 N_S^2} Adv_{Game2}(A) \quad \text{(III)}$$

Game 3 and Game 4: The analysis is consistent with Game 2 and Game 3 in Case 1.2

$$|Adv_{Game2}(A) - Adv_{Game3}(A)| \leq Adv_{Refresh-LRS} \quad \text{(IV)}$$

Game 4 and Game 5: The analysis is consistent with Game 3 and Game 4 in Case 1.2.

$$|Adv_{Game4}(A) - Adv_{Game5}(A)| = 0 \quad \text{(V)}$$

Game 5 and Game 6: The analysis is consistent with Game 4 and Game 5 in Case 1.2.

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

Game 6: The analysis is consistent with Game 4 in Case 1.1

$$Adv_{Game6}(A) = 0 \quad \text{(VII)}$$

Using equations (I)–(VI) we get we get

$$Adv^{\lambda-CAFLR}_{3PAKE} \leq N_S/N + N_P^2 N_S^2 (Adv_{Refresh-LRS} + Adv_{KDF}).$$

Case 2.2 $\boldsymbol{A}$ cannot ask a *corrupt*($\Pi^t_{U,(S,V)}$) query

In this case, $\boldsymbol{A}$ could get the random keys x and y by ***RevealClientEphemeralKey***($\Pi^t_{U,(S,V)}$) and ***RevealCli-entEphe-meralKey*** ($\Pi^t_{V,(S,U)}$), but $\boldsymbol{A}$ could not ask ***Reveal-ServerEphe-meralKey*** ($\Pi^t_{S,(U,V)}$) query in order not to violate λ-CAFLR-freshness of ***Test*** oracle.

Table 1 Security comparison of AKE protocols

Scheme	(Moriyama et al. 2011)	(Alawatugoda et al. 2014a)	(Alawatugoda et al. 2014b)	(Alawatugoda et al. 2015)	(Chen et al. 2016b)	(Ruan et al. 2017)	Ours
Security model	eCK	CK	eCK	eCK	eCK	eCK	eCK
Leakage Feature	RLM	CLM	RLM	CLM	RLM	BRM	CLM
After-the-fact	No	Yes	Yes	Yes	Yes	Yes	Yes
Proof model	Standard	Standard	RO	RO	Standard	Standard	Standard
Key Infrastructure	PKI	PKI	PKI	PKI	PKI	ID-based	PW-based
Parties	2	2	2	2	2	2	3

Game 1: It is the original game.
Game 2: Consistent with Game 2 in Case 2.1.
Game 3: Consistent with Game 3 in Case 2.1.
Game 4: Consistent with Game 3 in Case 1.3.
Game 5: Consistent with Game 4 in Case 1.3.
Game 6: Consistent with Game 4 in Case 1.1.

Differences between games:

The analysis is almost consistent with Case 2.1, and there has

$$Adv_{3PAKE}^{\lambda-CAFLR} \leq \frac{N_S}{N} + N_P^2 N_S^2 (Adv_{Refresh-LRS} + Adv_{KDF}).$$

From **Case 1** and **Case 2** we get

$$Adv_{3PAKE}^{\lambda-CAFLR} \leq \frac{N_S}{N} + N_P^2 N_S^2 (Adv_{DDH} + Adv_{Refresh-LRS} + Adv_{KDF}).$$

5.3 Protocol analysis

We compare our proposed protocol with all other LR AKE protocols. The result is given in Table 1, From which we could learn that: (1) the new protocol is the first LR 3PAKE protocol; (2) it is secure in the CAFLR eCK-security model, while (Moriyama and Okamoto 2011) only captured the leakage attacks that happen before the test session was chosen, (Alawatugoda et al. 2014a) just showed the security proof in the CK model; (3) it is proven in the standard model, while (Alawatugoda et al. 2014b) and (Alawatugoda et al. 2015) only proved the security in the RO mode.

6 Conclusions and future works

By combining the OCL model and eCK-security 3PAKE model appropriately, we first give a CAFLR eCK security model for 3PAKE. Then, we propose a LR 3PAKE protocol and show its detailed formal security proof in the standard model. The new protocol could be used to provide secure communications and identity authentications for the leakage environments such as Mobile Internet, Wireless Networks and Sensor Networks. Our future work includes at least the following two directions. Firstly, the multi-user and group settings are very common application environments, for example, Yamamoto (2016) proposed a secure group discussion system for active learning using smartphone. We will extend our approaches to the multi-user and group settings. Secondly, efficiency is very important in some specific environments such as time-critical and low end-to-end delay pay-TV and the power grid scenarios (Wang et al. 2016). We will look for optimization methods to improve efficiency of our construction because mobile devices and IoT devices in general are very limited in computational resources.

Acknowledgements The work was supported by the Natural Science Foundation of Hubei Province of China (No. 2017CFB596) and the Green Industry Technology Leading Project of Hubei University of Technology (No. ZZTS2017006).

References

Alawatugoda J, Boyd C, Stebila D (2014a) Continuous after-the-fact leakage-resilient key exchange. In: australasian conference on information security and privacy, pp 258–273

Alawatugoda J, Stebila D, Boyd C (2014b) Modelling after-the-fact leakage for key exchange. In: ACM symposium on information, computer and communications security, pp 207–216

Alawatugoda J, Stebila D, Boyd C (2015) Continuous after-the-fact leakage-resilient eck-secure key exchange. In: IMA international conference on cryptography and coding, pp 277–294

Bellare M, Pointcheval D, Rogaway P (2000) Authenticated key exchange secure against dictionary attacks. In: international conference on the theory and applications of cryptographic techniques, pp 139–155

Bellovin SM, Merritt M (1992) Encrypted key exchange: Password-based protocols secureagainst dictionary attacks. In: IEEE symposium on research in security and privacy, pp 72–84

Canetti R, Krawczyk H (2001) Analysis of key-exchange protocols and their use for building secure channels. Adva Cryptol EUROCRYPT 2045:453–474

Chasaki D, Mansour C (2015) Security challenges in the internet of things. Int J Space Based Situat Comput 5(3):141–149

Chen HC, Mao CH, Lin YT, Kung TL, Weng CE (2016a) A secure group-based mobile chat protocol. J Ambient Intell Hum Comput 7(5):693–703

Chen R, Mu Y, Yang G, Susilo W, Guo F (2016b) Strongly leakage-resilient authenticated key exchange. In: Cryptographers track at the RSA conference, pp 19–36

Davì F, Dziembowski S, Venturi D (2010) Leakage-resilient storage. SCN, vol 6280. Lecture Notes in Computer Science. Springer, Berlin, pp 121–137

Dziembowski S, Faust S (2011) Leakage-resilient cryptography from the inner-product extractor. In: Advances in cryptology - ASIACRYPT 2011 - international conference on the theory and application of cryptology and information security, Seoul, Proceedings, pp 702–721

Farash MS, Attari MA (2014a) An efficient and provably secure three-party password-based authenticated key exchange protocol based on chebyshev chaotic maps. Nonlinear Dyn 77(1–2):399–411

Farash MS, Attari MA (2014b) An efficient client–client password-based authentication scheme with provable security. J Supercomput 70(2):1002–1022

Goldreich O, Lindell Y (2006) Session-key generation using human passwords only. J Cryptol 19(3):241–340

Goyal V (2012) Positive results for concurrently secure computation in the plain model. In: foundations of computer science, pp 41–50

Hu C, Liu P, Guo S (2016) Public key encryption secure against related-key attacks and key-leakage attacks from extractable hash proofs. J Ambient Intell Hum Comput 7(5):1–12

Jin WB, Dong HL, Lim JI (2007) Ec2c-paka: An efficient client-to-client password-authenticated key agreement. Inf Sci 177(19):3995–4013

Katz J, Ostrovsky R, Yung M (2009) Efficient and secure authenticated key exchange using weak passwords. J ACM 57(1):78–116

Katz J, Mackenzie P, Taban G, Gligor V (2012) Two-server password-only authenticated key exchange. J Comput Syst Sci 78(2):651–669

Krawczyk H (2008) On extract-then-expand key derivation functions and an hmac-based kdf. http://webee.technion.ac.il/~hugo/kdf/kdf.pdf

Lamacchia B, Lauter K, Mityagin A (2007) Stronger security of authenticated key exchange. In: International conference on provable security, pp 1–16

Li S, Zhang F (2013) Leakage-resilient identity-based encryption scheme. Int J Grid Utility Comput 4(2/3):187–196

Mackenzie PD, Patel S, Swaminathan R (2000) Password-authenticated key exchange based on RSA. In: International conference on the theory and application of cryptology and information security, pp 599–613

Moriyama D, Okamoto T (2011) Leakage resilient ECK-secure key exchange protocol without random oracles. In: ACM symposium on information, computer and communications security, pp 441–447

Ou R, Kumar N, He D, Lee JH (2015) Efficient provably secure password-based explicit authenticated key agreement. Pervas Mob Comput 24(12):50–60

Ou R, Zhang Y, Zhang M, Zhou J, Harn L (2017) After-the-fact leakage-resilient identity-based authenticated key exchange. IEEE Syst J (99):1–10

Pu Q, Wang J, Wu S, Fu J (2013) Secure verifier-based three-party password-authenticated key exchange. Peer–Peer Netw Appl 6(1):15–25

Ran C, Dachman-Soled D, Vaikuntanathan V, Wee H (2012) Efficient password authenticated key exchange via oblivious transfer. Int Conf Pract Theory Public Key Cryptogr 7293:449–466

Tso R (2013) Security analysis and improvements of a communication-efficient three-party password authenticated key exchange protocol. J Supercomput 66(2):863–874

Wang Q, Ou R, Wang Z (2018) Security analysis and improvements of three-party password-based authenticated key exchange protocol. Springer, Cham, pp 497–508

Wang Y, Ma J, Lu X, Lu D, Zhang L (2016) Efficiency optimisation signature scheme for time-critical multicast data origin authentication. Int J Grid Utility Comput 7(1):1–11

Wu S, Pu Q, Wang S, He D (2012) Cryptanalysis of a communication-efficient three-party password authenticated key exchange protocol. Inf Sci 215(1):83–96

Xie Q, Dong N, Tan X, Wong DS, Wang G (2013) Improvement of a three-party password-based key exchange protocol with formal verification. Inf Technol Control 42(3):231–237

Xiong H, Chen Y, Guan Z, Chen Z (2013) Finding and fixing vulnerabilities in several three-party password authenticated key exchange protocols without server public keys. Inf Sci 235(1):329–340

Yamamoto N (2016) An improved group discussion system for active learning using smartphone and its experimental evaluation. Int J Space Based Situat Comput 6(4):221–227

Yang JH, Cao TJ (2012) Provably secure three-party password authenticated key exchange protocol in the standard model. J Syst Softw 85(2):340–350

Zhao J, Gu D (2012) Provably secure three-party password-based authenticated key exchange protocol. Inf Sci 184(1):310–323