Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 16(5): 3623–3640.
DOI: 10.3934/mbe.2019180
Received: 21 February 2019
Accepted: 10 April 2019
Published: 23 April 2019

Research article

Verifier-based anonymous password-authenticated key exchange protocol in the standard model

Qihui Zhang[1], Pradeep Chaudhary[2], Saru Kumari[3], Zhiyin Kong[4] and Wenfen Liu[5,*]

[1] State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, Henan 450001, China
[2] Department of Statistics, Chaudhary Charan Singh University, Meerut 250004, India
[3] Department of Mathematics, Chaudhary Charan Singh University, Meerut 250004, India
[4] Science and Technology on Information Assurance Laboratory, Beijing 100072, China
[5] Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China

* **Correspondence:** Email: liuwenfen@guet.edu.cn; Tel: +8617307735812.

Abstract: Anonymous password-authenticated key exchange (APAKE) allows a client to authenticate herself and to establish a secure session key with a remote server via only a low-entropy password, while keeping her actual identity anonymous to the third party as well as to the server. Since that APAKE protocol enjoys both the convenience of password authentication and the advantage of privacy protection, researchers have paid much attention to them. However, most of the existing APAKE protocols are designed in the symmetric setting which does not take into consideration the threat of password file leakage. To mitigate the damage of server compromise, we propose a verifier-based anonymous password-authenticated key exchange protocol, in which the server holds a verifier corresponding to each client instead of the clear password. The construction of our protocol is built on standard cryptographic primitives such public key encryption, smooth projective hash functions and password hashing schemes. The resulting protocol is proved secure in the standard model, i.e., without resorting to random oracles. Comparisons with other similar schemes show that our protocol guarantees stronger security while enjoys considerable efficiency in terms of computational cost.

Keywords: password authentication; anonymous protocol; key exchange; server compromise; standard model

1. Introduction

Password authenticated key exchange (PAKE) is an important cryptographic primitive that allows a client to authenticate herself and to establish a high-entropy session key with a remote server via only a common, low-entropy password. It is very convenient since that no complicated public key infrastructure (PKI) or dedicated hardware such as a token is needed in advance. Because of its convenience and significance, password authentication schemes are regarded as, and are likely to continue to be, the dominant authentication mechanism in the foreseeable future [1].

In 1992, the most famous pioneering work of PAKE, called the Encrypted Key Exchange (EKE) protocol, was proposed by Bellovin and Merritt [2], in which the Diffie-Hellman flows are symmetrically encrypted under the common password to offer resistance to dictionary attack. Since then, a multitude of studies have been published on the construction of more efficient and more secure PAKE protocols in random oracle model [3,4], in the standard model [5–9], in multi-user setting [10,11], as well as on the security models suitable for the analysis of PAKE protocols [12,13]. Organizations like the International Organization for Standardization (ISO) have further issued a lot of password-related standards, such as ISO/IEC 11770-4, IEEE Std 1363.2, RFC 6124, etc., which further popularizes the widespread application of the PAKE protocols.

Along with the growing concern about privacy protection, many people want to strengthen the widely used PAKE protocols with the additional property of anonymity, which keeps a client's identity secret not only to all outsiders but also to the server. To address this need, anonymous password- authenticated key exchange (APAKE) protocols were first presented by Viet et al. in 2005 [14] and then improved by Yang et al. [15], Shin et al. [16] and Hu et al. [17] in the password-only setting. While the client has access to some extra storage other than the password, the computational performance of APAKE protocols, especially on the server side, could be further improved. The storage extra approach was first proposed by Yang et al. [18] in ACSAC'09, and further developed by Shin et al. [19], Zhang et al. [20]. Furthermore, ISO/IEC has recently issued an APAKE standard denoted as ISO/IEC 20009-4 [21].

Almost all the existing APAKE protocols, and most of the PAKE protocols, are designed in the symmetric setting, where the server holds all the clients' password in clear. This would be dramatic in case of server compromise. At the same time, the number of password leakage incidents and the cost to those experiencing them continue to increase in the last few years. Typical password data breaches include 167 million from LinkedIn [22], 3 billion from Yahoo [23] and 50 million from Facebook [24]. In order to mitigate the damage of server compromise, it is desirable to adapt traditional APAKE protocols to the verifier-based setting, in which the server holds a verifier corresponding to each client instead of the plain password. Although server verifier-based solutions have been put forward with respect to traditional (non-anonymous) PAKE protocols [25,26], only few verifier-based APAKE protocols [27,28] were proposed and, moreover, they are vulnerable to impersonation attacks (see details in Section 4).

In this paper, we propose a verifier-based anonymous password-authenticated key exchange protocol, which provides additional security guarantees in case of server compromise. The new protocol, which is built on standard cryptographic primitives such as ElGamal public key encryption, password hashing scheme and smooth projective hash functions, is proven secure in the standard model rather than the random oracle model. Comparisons show that our protocol guarantees stronger security while pays very little in terms of computational efficiency.

2. Security model

In this section, we recall the security model for APAKE protocols [17], which is extended from the security models for PAKE protocols [12] and for APAKE protocols [14–16].

2.1. Participants and passwords

The participants of an APAKE protocol P consist of a set of clients $\mathcal{C}$ and a set of servers $\mathcal{S}$. Without loss of generality, we assume that the set $\mathcal{S}$ contains only one server $\mathcal{S} = \{S\}$. The client set $\mathcal{C}$ consists of a set of honest clients $\mathcal{G} = \{C_j\}_{1 \le j \le n}$ and a set of malicious clients $\mathcal{E}$. Each client $C \in \mathcal{C}$ holds a password p_C chosen according to some given distribution $\mathcal{D}$ as its authentication credential; the server holds a list of transformed password $pw_S = \{H(p_C)\}_{C \in \mathcal{C}}$ corresponding to all the clients. The notion $H(p_C)$ denotes the value by using a password transformation function on p_C. It might be, for example, $H(p_C) = p_C$ for the symmetric setting and $H(p_C) = s^{p_C}$ for the asymmetric setting.

2.2. Protocol execution

The adversary $\mathcal{A}$, who has full control of the communication network and is modeled as a probabilistic polynomial time (PPT) Turing machine, can interact with various concurrent sessions of each user $U \in \mathcal{C} \cup \mathcal{S}$ through the following oracle queries:

- $Execute(C^r, S^d)$: This query models those attacks that could be mounted by a passive adversary eavesdropping on communication between two honest sessions, where C^r denotes the r-th session of client C and S^d denotes the d-th session of server S. The output of the query contains all the messages outputted by C^r and S^d during an honest execution.
- $Send(U^r, M)$: This query is used to model active attacks against a session U^r of a user $U \in \mathcal{C} \cup \mathcal{S}$. The adversary sends an arbitrary message M to the session U^r, and gets the outputting message that would be generated by the U^r upon receiving the message M.
- $Reveal(U^r)$: This query models the leakage or misuse of the session keys that have been used in those sessions other than the session being tested. In other words, this query is used to model known session key attacks.
- $Corrupt(U)$: The query models the leakage of passwords of some clients or the server. If this query is asked to a client $C \in \mathcal{C}$, the password p_C of this client is returned. If this query is asked to the server S, the list of transformed passwords $pw_S = \{H(p_C)\}_{C \in \mathcal{C}}$ is returned.

2.3. AKE security

The AKE security would guarantee that, when a session key is generated freshly by honest participants and has not been leaked or misused, it would look like uniformly random to an adversary. For this purpose, the notions of partnering sessions and fresh sessions need to be defined first.

Partnering sessions. A client session C^r and a server session S^d are called partnering sessions, if (1) they are the intended partner of each other; (2) they both accept; (3) their communication transcripts are matched with each other.

Freshness. We say that a session U^r satisfies the definition of freshness, if no $Reveal$ query has been asked to this session or its partnering session, and no $Corrupt$ query has been asked to the participants involved in this session.

In addition, we need a new query to model the attack that the adversary distinguishes a real session key from a random one.

- $Test(U^r)$: This query can only be asked to a fresh session. When this query is received, a random bit $b \in \{0,1\}$ is chosen and the query is answered as follows. If $b = 1$, the real session key of U^r is returned; if $b = 0$, a random session key is returned.

The adversary is allowed to ask as many as queries defined in Section 2.2 and the above $Test$ query, with the restriction that the $Test$ query can only be asked once and the target session of the $Test$ query should be kept fresh. At the end, the adversary outputs a bit b' as his guess to the hidden random bit b. The advantage of the adversary $\mathcal{A}$ is defined by $Adv(\mathcal{A}) = 2\Pr\{b = b'\} - 1$. An APAKE protocol P is AKE secure if for every PPT adversary $\mathcal{A}$ the advantage satisfies $Adv_P^{AKE}(\mathcal{A}) \leq C' q_{send}^{s'} + negl(k)$, where C' and s' are the Zipf parameters [1,29,30] corresponding to the password dictionary D, and q_{send} denotes the number of $Send$ queries asked by the adversary.

2.4. Client anonymity

The client anonymity notion states that even the trusted server, which behaves in an honest-but-curious way, should not be capable of getting the actual identity of the client in communication. What the server can only learn about the client is that she is a legitimate member of the group $\Gamma = \{C_j\}_{1 \leq j \leq n}$.

Here we consider the server as a passive adversary $\mathcal{M}$, who knows all the transformed passwords $pw_S = \{H(p_C)\}_{C \in \Gamma}$, interacting with the client in an honest way and wants to figure out the identity of the client. Denote by $\mathcal{P}_i$ the distribution of transcripts of the protocol P executed between the client $C_i \in \Gamma$ and the server S. If for any two clients $C_i, C_j \in \Gamma$, the two distributions $\mathcal{P}_i, \mathcal{P}_j$ are computational indistinguishable, i.e., $|\Pr\{M(\mathcal{P}_i) = 1\} - \Pr\{M(\mathcal{P}_j) = 1\}| \leq negl(k)$, we say that the protocol P satisfies client anonymity.

3. Cryptographic primitives

3.1. Public key encryption scheme

In the construction of our verifier-based APAKE protocol, we use both CPA secure (i.e., secure under chosen-plaintext attacks) and labeled PCA secure (standing for secure under plaintext-checking attacks) public key encryption schemes. Formally speaking, a public key encryption scheme is defined by three algorithms $\mathcal{E} = (KGen, Enc, Dec)$, where $KGen$ on input 1^k outputs a public/secret key pair (pk, sk), Enc on inputs pk, message m and the random string r outputs a ciphertext c, and Dec on inputs sk and c outputs m or $\perp$. A label public key encryption scheme is a public key encryption that additionally admits a label to its encryption and decryption algorithms as $c \leftarrow Enc(pk, m; label; r)$ and m or $\perp \leftarrow Dec(sk, c; label)$.

We say that a public key encryption scheme is CPA secure, if when the public key pk is already published, every probabilistic polynomial time (PPT) adversary would have only negligible advantage in distinguishing the challenge ciphertexts corresponding to two messages m_0, m_1 chosen by the adversary. In order to address active attacks, a stronger notion called CCA security (standing for security under chosen-ciphertext attacks) was proposed, in which the adversary has additional access to a decryption oracle, which could decrypt any ciphertext other than the challenge ciphertext. Between CPA and CCA security, a new security notion termed PCA security, short for security under plaintext-checkable attacks, was recently proposed by Abdalla et al. [31]. The adversary in this notion does not have access to any decryption oracle, but has only access to an oracle indicating whether a given ciphertext/message pair is consistent, in the sense that the given ciphertext encrypts the given message.

3.2. Password hashing scheme

In order to formalize the way how verifiers are generated, Benhamouda et al. [25] presented a new cryptographic primitive called password hashing scheme. As an added benefit, the password hashing scheme could easily suit to smooth projective hash functions in an algebraic way. Concretely, a password hashing scheme consists of a tuple of algorithms $PHS = (PSetup, PPreHash, PSalt, Phash)$. Let n denote the bit-length of possible passwords, i.e., $\pi \in D \subset \{0,1\}^n$. The algorithm $PSetup$ on inputs the security parameter 1^k and the bit-length of the password 1^n outputs the parameter $param$. The algorithm $PPreHash$ on inputs $param$ and a password p outputs a pre-hash value P. The algorithm $PSalt$ on input $param$ outputs a salt s. The algorithm $Phash$ takes as inputs the parameter $param$, a salt s and a password p, and outputs a hash value H.

For the security of verifier-based APAKE protocols, a password hashing scheme should satisfy several security notions, such as the correctness, salt indistinguishability, second pre-image resistance, entropy preservation, pre-hash entropy preservation, and tight one-wayness [25]. The tight one-wayness is the most vital notion, as it ensures that recovering a valid password from a leaked server's file requires a time approximately equals computing 2^b times of $Phash$, where b denotes the min-entropy of the password distribution.

3.3. Smooth projective hash functions

Smooth projective hash functions (SPHF) were introduced by Cramer and Shoup [32] in order to construct CCA secure public key encryption schemes. In 2003, Gennaro et al. [33] tailored the initial definition to the construction of PAKE protocols. Later, the SPHF definition was further developed by Katz et al. [34], Benhamouda et al. [6] and Abdalla et al. [35] to fit more complex language obtained by disjunctions or conjunctions of simpler languages.

More specifically, given a language L Ì X for some set X, a SPHF system for the language L consists of the following 4 algorithms $\mathrm{H} = (HashKG, ProjKG, Hash, ProjH)$. The hash key generation algorithm $HashKG$ takes as input the language L and outputs a hash key hk. The algorithm $ProjKG$ takes as inputs the language L, a hash key hk and possibly a word c Î L and outputs a projection key hp. The algorithm $Hash$ takes as inputs the language L, a hash key hk and an element c Î X, and outputs the hash value corresponding to the word c. The algorithm $ProjH$ takes as inputs the language L, a projection key hp, a word c Î L as well as the witness

w with respect to the fact $c \in L$, and outputs the hash value corresponding to the word c.

The security requirement of a SPHF system contains correctness and smoothness. The correctness property ensures that for any word $c \in L$ and its related witness w, the following equation $Hash(L,hk,c) = ProjH(L,hp,c,w)$ always holds. The smoothness property guarantees that, for any element $c \in X \setminus L$, even the corresponding projection key hp is published, the statistical distance between the distribution of $Hash(L,hk,c)$ and uniform random is negligible.

3.4. One-time signature scheme

A one-time signature [36] is a weak notion of cryptographic signature that could be instantiated in the standard model, yet enjoys much more efficient computational complexity. Similar to classical signatures, a one-time signature consists of 3 algorithms $\Sigma = (SignKG, Sign, Verify)$. The algorithm $SignKG$ takes as input the security parameter 1^k and outputs a pair of signing/verification keys (SK,VK). The algorithm $Sign$ takes as inputs SK and the message m and outputs a signature σ. The algorithm $Verify$ takes as inputs VK, a message m and a signature σ, and outputs 1 iff the σ is a valid signature of m. With respect to the security definition of existential unforgeability, we allow the adversary to get access to only one message/signature pair. If for any PPT adversary, the possible advantage is negligible, we say that the one-time signature scheme is secure.

4. Verifier-based APAKE protocol

4.1. Attacks on the existed schemes

Although APAKE protocols have been researched for more than a decade, as far as we know, only two recent studies have focused on verifier-based setting [27,28]. However, we find that these protocols are not as secure as claimed, both of them are vulnerable to impersonation attacks.

4.1.1. An impersonation attack on the protocol in [27]

In 2016, Yang et al. [27] put forward the first verifier-based APAKE protocol and proved its security in the standard model. The proposed protocol, which achieves mutual authentication in only two rounds, is very efficient in terms of communication complexity.

The construction of Yang et al.'s protocol is illustrated in Figure 1, which utilizes a CCA-secure labeled public key encryption scheme (KG_1, Enc_1, Dec_1), a CPA-secure public key encryption scheme (KG_2, Enc_2, Dec_2), and a password hashing scheme $PHS = (PSetup, PPHSalt, PPreHash, PHSalt, PHash)$. Note that the definition of password hashing scheme used here follows the one provided in [37], which employs two salt-generating algorithms. Given two salts $s_P, s_H \in \mathbf{Z}_p^*$, the pre-hash and hash values are generated as $P = g^{s_P \cdot p}$ and $H = (H_1, H_2) = (g^{s_P}, P ? h^{s_H})$.

Their protocol also uses two KV-SPHFs [34] for the following two languages,

$$L_C = \{(l, c_C) \mid \exists p, \exists r_C : c = Enc_1^l(pk_1, g^{p_i - i}; r_C) ? H_{2i} \quad H_1^{p_i} h^{s_{H_i}}\},$$

$$L_S = \{c_{S_i} \mid \exists r_S : c_{S_i} = Enc_2(pk_2, H_{2i} h^{s_{H_i}}; r_S)\}.$$

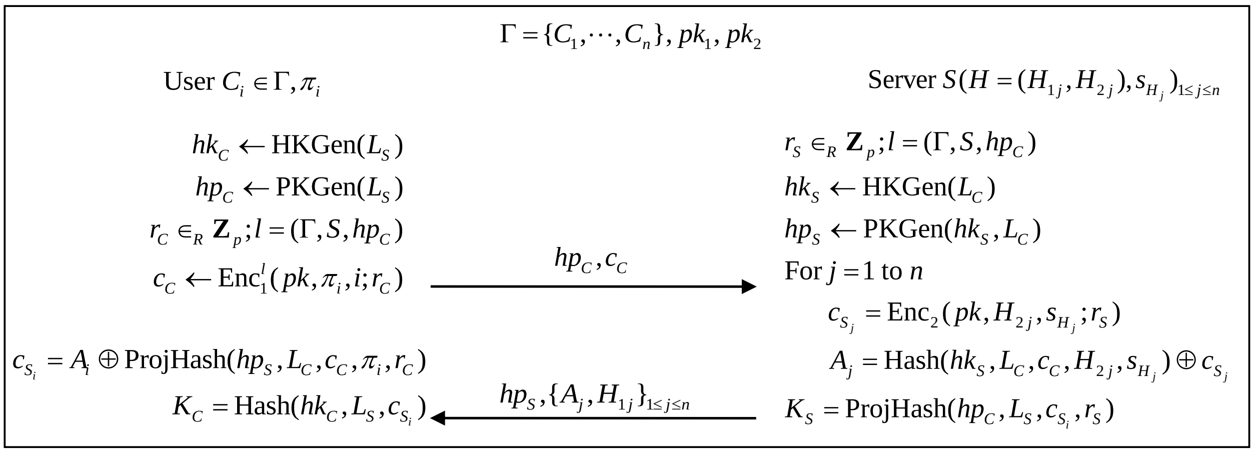
$\Gamma = \{C_1, \cdots, C_n\}, pk_1, pk_2$
User $C_i \in \Gamma, \pi_i$
Server $S(H = (H_{1j}, H_{2j}), s_{H_j})_{1 \le j \le n}$
$hk_C \leftarrow \mathrm{HKGen}(L_S)$
$hp_C \leftarrow \mathrm{PKGen}(L_S)$
$r_C \in_R \mathbf{Z}_p; l = (\Gamma, S, hp_C)$
$c_C \leftarrow \mathrm{Enc}_1^l(pk, \pi_i, i; r_C)$
hp_C, c_C
$r_S \in_R \mathbf{Z}_p; l = (\Gamma, S, hp_C)$
$hk_S \leftarrow \mathrm{HKGen}(L_C)$
$hp_S \leftarrow \mathrm{PKGen}(hk_S, L_C)$
For $j = 1$ to n
$c_{S_j} = \mathrm{Enc}_2(pk, H_{2j}, s_{H_j}; r_S)$
$A_j = \mathrm{Hash}(hk_S, L_C, c_C, H_{2j}, s_{H_j}) \oplus c_{S_j}$
$c_{S_i} = A_i \oplus \mathrm{ProjHash}(hp_S, L_C, c_C, \pi_i, r_C)$
$K_C = \mathrm{Hash}(hk_C, L_S, c_{S_i})$
$hp_S, \{A_j, H_{1j}\}_{1 \le j \le n}$
$K_S = \mathrm{ProjHash}(hp_C, L_S, c_{S_i}, r_S)$

Figure 1. The VAPAKE protocol presented in [23].

We point out that Yang et al.'s protocol is vulnerable to an impersonation attack. Recall that the projection key hp_C generated and sent by the client is protected by containing it in the label of the CCA-secure ciphertext c_C. However, the projection key hp_S sent by the server to the client is not in protected. This will lead to an impersonation attack in which an adversary impersonates the legitimate server by sending well-crafted message. More specifically, if the protocol is instantiated by Cramer-Shoup-like encryption schemes as that was presented in Section D of [27], the adversary could generate the message $< hp_S, \{A_j, H_{1j}\}_{1 \# j \quad n} >$ as follows. The projection key hp_S is chosen to be $hp_S = (hp_{S1}, hp_{S2}, hp_{S3})$ where $hp_{S1} = f^{-x+1}, hp_{S2} = f, hp_{S3} = g$. Note that the adversary can eavesdrop the message sent by the client and get $c_C = Enc_1^l(pk_1, g^{p_i - i}; r_C) = (u, v, e, w)$ where $e = f^{r_C} ? g^{p_i - i}$. Then the adversary chooses a random r_S, computes $c_{S_j} = (u_j, e_j) = (g^{r_S}, f^{r_S})$, $A_j = (e 肴 g^j)$ $c_{S_j} = f^{r_C} g^{p_i - i + j}$? c_{S_j}, and sets $H_{1j} = 1$, for all $1 \# j \quad n$.

Upon receiving the above message, the client C_i will compute according to the protocol specification as follows. She first computes a projection hash value as $(hp_{S1} \cdot hp_{S2}^{\xi})^{r_C} \cdot hp_{S3}^{\pi_i} = g^{\pi_i} f^{r_C}$. Then she computes $c_{S_i} = A_i \oplus (hp_{S1} \cdot hp_{S2}^{\xi})^{r_C} \cdot hp_{S3}^{\pi_i} = A_i \oplus g^{\pi_i} f^{r_C} = (g^{r_S}, f^{r_S})$ and generates her session key as $K_C = (g^{r_S})^a (f^{r_S})^b$. However, as the adversary has already obtained the projection key $hp_C = g^a f^b$ from the first message, he can also compute the session key as $K_S = hp_C^{r_S} = (g^a f^b)^{r_S}$. This results in a successful impersonation attack.

4.1.2. An impersonation attack on the protocol in [28]

In 2018, Chen et al. [28] gave out a new verifier-based APAKE protocol by taking Zhang et al.'s algebraic MAC [20] as the server's verifier. With the help of hash functions, which would usually be recognized as random oracles (RO), this protocol performs much better than Yang et al.'s protocol [27], and is more efficient than most of the existed APAKE protocols in terms of computation and communication cost. It is claimed that this protocol is also secure against various known attacks, such as mutual authentication and forward secrecy.

The concrete steps of Chen et al.'s protocol [28] is depicted in Figure 2. In the registration phase, each user U_j enrolls to the server S by sending ID_j and $m_j = H(ID_j \| PW_j)$, then the server S uses its secret key s to compute an algebraic MAC $V_j = g^{1/(m_j+s)}$ as the corresponding verifier.

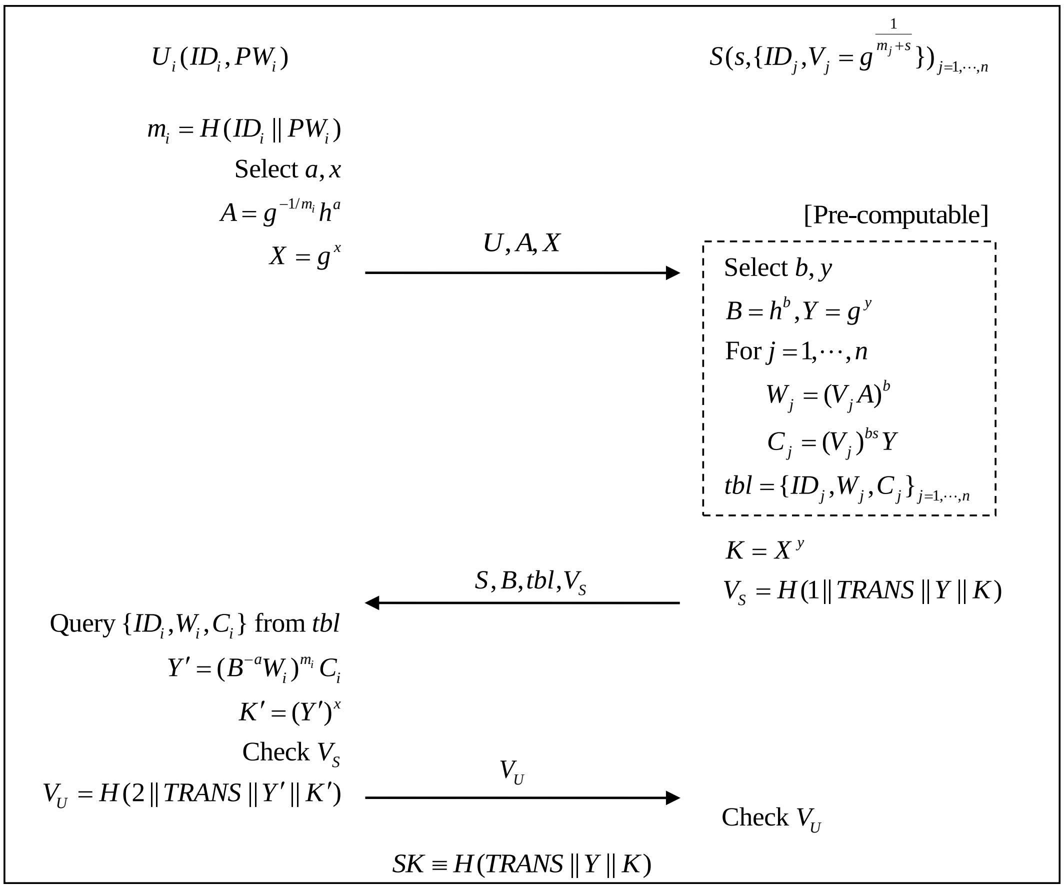

Figure 2. The VAPAKE protocol presented in [28].

We find that Chen et al.'s protocol [28] is also vulnerable to an impersonation attack. An outside adversary who does not know any secret of the server and the client can impersonate a legitimate server to the client. The specific attack is as follows. When an adversary gets the first message $< U, A, X >$ sent by some client U_i, he generates the second message by setting $B = h^{-1}$, $Y = g^y$, $W_j = A^{-1}$, $C_j = g^{y-1}$, $1 \# j \quad n$ and $K = X^y = g^{xy}$, $V_S = H(1 \| TRANS \| Y \| K)$. The adversary then sends the message $< S, B, tbl, V_S >$ to the client by pretending to be the legitimate server. When this message is received by the client, she will compute $Y' = (B^{-a} W_i)^{m_i} C_i = (h^a \cdot g^{1/m_i} h^{-a})^{m_i} \cdot g^{y-1} = g^y$ and K ii= $Y^x = g^{xy} = K$. As a consequence, all the subsequent verification will be successful and the adversary successfully impersonates the server.

4.2. Construction of our protocol

In this section, we give out a new verifier-based APAKE protocol constructed from standard cryptographic primitives such as public key encryption scheme, smooth projective hash functions and password hashing scheme. The underlying techniques are from [17,25,34,38], which explored SPHFs for complex languages dealing with anonymous and verifier-based authentication.

The resulting protocol, depicted in Figure 3, is a 3-flow verifier-based APAKE protocol with explicit mutual authentication. We emphasize that, when the messages in boxes, i.e., t_C, t_S, are set to be empty strings and the last flow of message $< t_C >$ is removed like [38], one would get a 2-flow protocol but with only explicit unilateral authentication.

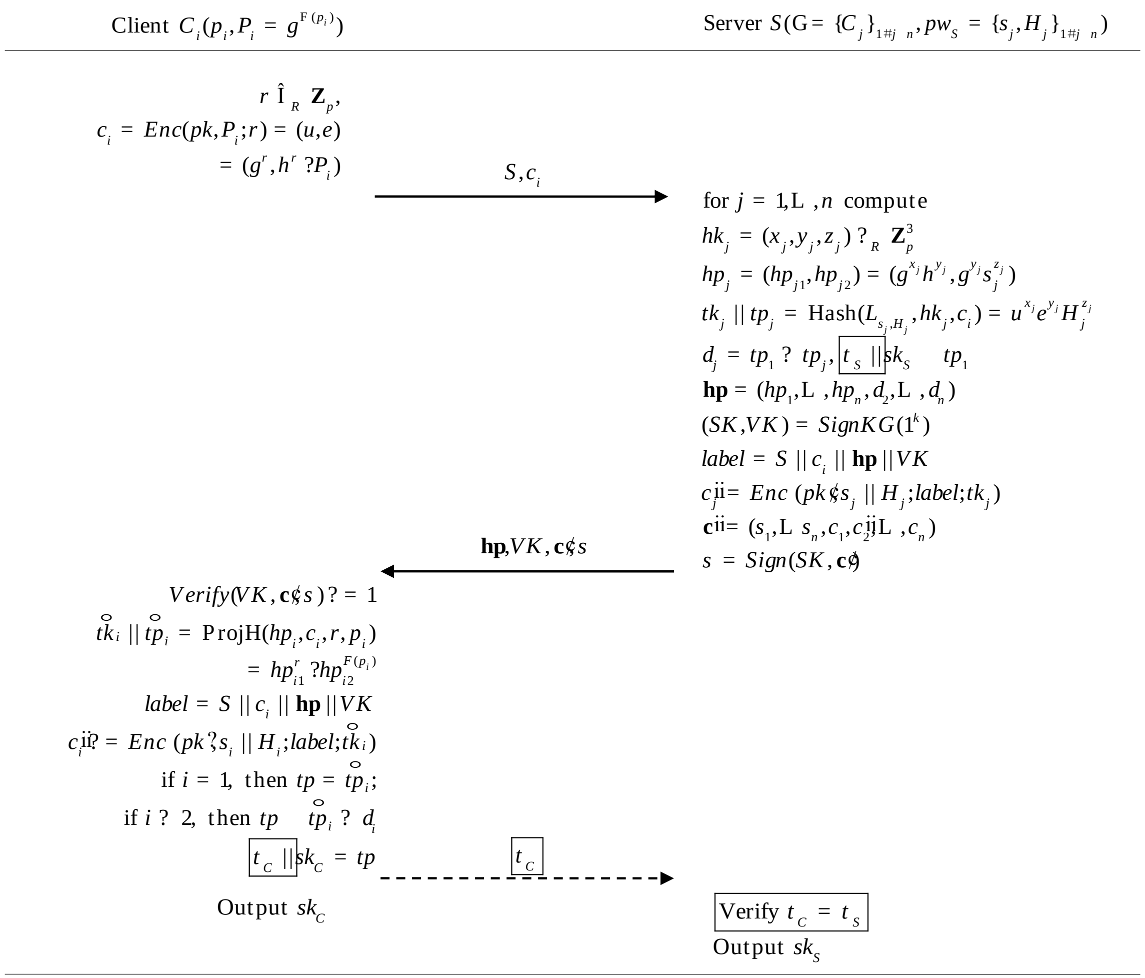
Client $C_i(p_i, P_i = g^{F(p_i)})$
Server $S(\Gamma = \{C_j\}_{1 \le j \le n}, pw_S = \{s_j, H_j\}_{1 \le j \le n})$
$r \in_R \mathbf{Z}_p$,
$c_i = Enc(pk, P_i; r) = (u, e) = (g^r, h^r \cdot P_i)$
S, c_i
for $j = 1, \ldots, n$ compute
$hk_j = (x_j, y_j, z_j) \in_R \mathbf{Z}_p^3$
$hp_j = (hp_{j1}, hp_{j2}) = (g^{x_j} h^{y_j}, g^{y_j} s_j^{z_j})$
$tk_j \| tp_j = \mathrm{Hash}(L_{s_j, H_j}, hk_j, c_i) = u^{x_j} e^{y_j} H_j^{z_j}$
$d_j = tp_1 \cdot tp_j$, $\boxed{t_S \|} sk_S \leftarrow tp_1$
$\mathbf{hp} = (hp_1, \ldots, hp_n, d_2, \ldots, d_n)$
$(SK, VK) = SignKG(1^k)$
$label = S \| c_i \| \mathbf{hp} \| VK$
$c''_j = Enc''(pk'', s_j \| H_j; label; tk_j)$
$\mathbf{c}'' = (s_1, \ldots, s_n, c''_1, c''_2, \ldots, c''_n)$
$\sigma = Sign(SK, \mathbf{c}'')$
$\mathbf{hp}, VK, \mathbf{c}'', \sigma$
$Verify(VK, \mathbf{c}'', \sigma) \stackrel{?}{=} 1$
$\tilde{tk}_i \| \tilde{tp}_i = \mathrm{ProjH}(hp_i, c_i, r, p_i) = hp_{i1}^r \cdot hp_{i2}^{F(p_i)}$
$label = S \| c_i \| \mathbf{hp} \| VK$
$c''_i \stackrel{?}{=} Enc''(pk'', s_i \| H_i; label; \tilde{tk}_i)$
if $i = 1$, then $tp = \tilde{tp}_i$;
if $i \ge 2$, then $tp \leftarrow \tilde{tp}_i \cdot d_i$
$\boxed{t_C \|} sk_C = tp$
$\boxed{t_C}$
Output sk_C
$\boxed{\text{Verify } t_C = t_S}$
Output sk_S

Figure 3. Our verifier-based APAKE protocol.

Assume that $\mathbf{G}$ be a cyclic multiplicative group with prime order p, and g, h be its two random generators. Our construction uses an ElGamal encryption scheme $\mathcal{E} = (KGen, Enc, Dec)$, which takes g, h as its public key pk. The encryption on message $M \in \mathbf{G}$ with randomness r is computed as $c = Enc(pk, M; r) = (u, e) = (g^r, h^r \cdot M)$. The construction also needs a PCA-secure public key encryption scheme $\mathcal{E}'' = (KGen'', Enc'', Dec'')$ and a one-time signature $\mathcal{S} = (SignKG, Sign, Verify)$, which could be instantiated arbitrarily. We emphasize that, the public keys of public key encryption schemes are contained in the common reference string as [5,33], thus no PKI is needed here.

Our construction also uses an algebraic password hashing scheme proposed by Benhamouda et al. [25], where $param = g\,||\,\mathbf{G}||\,F(?)$, $P = PPreHash(param, p) = g^{F(p)}$, $s = PSalt(param)\,?_R\,\mathbf{G}$, and $H = PHash(param, s, p) = s^{F(p)}$, where $F(?)$ is a hash function from the password dictionary to $\mathbf{Z}_p^*$. Given a salt s and a password hashing value H, we define the following complex language

$$L_{s.H} = \{(u,e)\,|\,\$r, \$p, u = g^r, e = h^r g^{F(p)}, H = s^{F(p)}\}, \tag{1}$$

which could be seem as the set of ciphertexts of a pre-hash value that is consistent with the password hashing value H. By exploiting the technique of [25], we also construct the following SPHF system H . The hash key is generated as $hk = HashKG(L_{s,H}) = (x,y,z)\,?_R\,\mathbf{Z}_p^3$; the projection key is computed as $hp = ProjKG(L_{s,H}, hk, c) = (hp_1, hp_2) = (g^x h^y, g^y s^z)$. The hash value could be computed through hk as $h = Hash(L_{s,H}, hk, c) = u^x e^y H^z$. It could also be generated by using the projection key hp and a pair of witnesses r, p as $h = ProjH(L_{s,H}, hp, c, r, p) = hp_1^r hp_2^{F(p)}$.

Let $G = \{C_j\}_{1\#j\ n}$ denote a set of legitimate clients and n denote the set size. Assume that each client $C_i \in \Gamma$ holds a password p_i, thus it could pre-compute $P_i = g^{F(p_i)}$; assume that the server S holds the verifiers $pw_S = \{s_j, H_j\}_{1\#j\ n}$ for all the clients. Suppose a client $C_i \in \Gamma$ wants to establish a session key with the server S in an anonymous way, the verifier-based APAKE protocol proceeds as follows.

(1) The client $C_i \in \Gamma$ selects a random number r Î $\mathbf{Z}_p$ and computes an ElGamal ciphertext of its pre-hash value $P_i = g^{F(p_i)}$ as $c_i = Enc(pk, P_i; r) = (u,e) = (g^r, h^r\,?P_i)$. Then she sends the message $< S, c_i >$ to the server.

(2) Upon receiving the message $< S, c_i >$, the server first does the following computations for each index $1\,\#j\ \ n$. He chooses random $hk_j = (x_j, y_j, z_j)\,?_R\,\mathbf{Z}_p^3$, computes the projection key as $hp_j = (hp_{j1}, hp_{j2}) = (g^{x_j} h^{y_j}, g^{y_j} s^{z_j})$ and the hash values as $tk_j\,||\,tp_j = \mathrm{Hash}(L_{s,H}, hk_j, c_i) = u^{x_j} e^{y_j} H_j^{z_j}$. Note that, similar to [38], the hash value is divided into two substrings for subsequent use. Next, the server S computes $d_j = tp_1\ ?\ tp_j, t_S\,||\,sk_S\ \ tp_1$, sets $\mathbf{hp} = (hp_1, \mathrm{L}, hp_n, d_2, \mathrm{L}, d_n)$, and generates a signing/ verification key pair as $(SK, VK) = SignKG(1^k)$. Then, he sets $label = S\,||\,c_i\,||\,\mathbf{hp}\,||\,VK$ and computes a ciphertext as $c_j$ii$= Enc\,(pk?s_j\,||\,H_j; label; tk_j)$ where the random string is the first part of the SPHF's hash value. After that, the server sets $\mathbf{c}$ii$= (s_1, \mathrm{L}\ s_n, c_1, cii_2\mathrm{L}, c_n)$, signs it with his signing key SK as $s = Sign(SK, \mathbf{c}¢)$ and sends $< \mathbf{hp}, VK, \mathbf{c}¢s >$ to the server.

(3) When the client C_i receives the message, she first verifies the validity of the signature s. If the verification is successful, the client computes the SPHF's hash value by using her witnesses r, p_i as $\overset{\circ}{tk_i}\,||\,\overset{\circ}{tp_i} = \mathrm{ProjH}(hp_i, c_i, r, p_i) = hp_{i1}^r\,?hp_{i2}^{F(p_i)}$. Next, the client sets $label = S\,||\,c_i\,||\,\mathbf{hp}\,||\,VK$, computes $H_i = s_i^{F(p_i)}$ and checks whether it holds $c_i$ii$= Enc\,(pk?s_i\,||\,H_i; label; \overset{\circ}{tk_i})$. Note that only the *i*-th ciphertext is recomputed but the whole vector $\mathbf{c}¢$ is guaranteed unmodified, because the verify key VK is contained in the label. Then, the client recovers the value tp_1 as follows: if $i = 1$, this values is already known; if $i\ ^3\ 2$, she would compute it as $tp_i \oplus \delta_i$. Denote the value obtained by the client as tp. Finally, the client divides tp into two parts as $t_C\,||\,sk_C = tp$, sets its session key as sk_C and sends t_C to the server as an authentication value.

(4) While the message $< t_C >$ is received, the server S verifies this value by checking whether

$t_C = t_S$ holds. If the message passes the check, the server will complete the session and set his session key as sk_S.

We stress that if a verification conducted by the client (the server) fails, she (or he) will abort the session immediately.

4.3. Designing rationales

The construction of our verifier-based APAKE protocol takes the Groce-Katz framework [38] of traditional PAKE protocols as a starting point. This framework achieves mutual authentication in 3 flows of communication, or alternatively unilateral authentication in 2 flows. Although the Groce-Katz framework does not achieve the optimal rounds for PAKE protocols, it enjoys the advantage of a high computational efficiency as the encryption scheme adopted by the client side only needed to be CPA secure instead of CCA secure. More precisely, the Groce-Katz framework is more efficient in terms of computation than those one-round PAKE protocols presented in [6,34]. As a consequence, the resulting verifier-based APAKE protocol is also computationally efficient.

In order to achieve anonymity by utilizing SPHFs, we let the server generate a sequence of projection keys for each possible client, i.e., for each language $L_{s_j,H_j}, 1 \# j \quad n$. The smooth property of the SPHF system guarantees that only when the ciphertext c_i matches the languages L_{s_i,H_i}, the hash value could be reconstructed by the client by using the projection key; otherwise, it is totally random to her. However, since only one of these projection keys is actually used by the client, special attention should be paid to the protection of these projection keys. We address this problem by containing them in the labels of a CCA-secure public encryption scheme and binding these ciphertext together with a one-time signature.

For the purpose of checking the verifiers in an implicit way, we choose to use the algebraic password hashing scheme proposed by Benhamouda et al. [25]. Recall that the pre-hash value P and the hash value H_j are with the same exponent with respect to bases g and s_j respectively. Henceforth the server could prove this fact by selecting a projective hash key of the form $g^{y_j} s_j^{z_j}$. Furthermore, this password hashing scheme meets all the security notions presented in Section 3.2, including tight one-wayness. On the contrary, although the password hashing scheme [37] utilized in Yang et al.'s protocol [27] is also an algebraic one, it is not tightly one-way as claimed [25].

5. Security analysis of the protocol

In this section, we first analyze the security of the verifier-based APAKE protocol heuristically. Then, we give out a rigorous security proof within the security model presented in Section 2.

5.1. Heuristic security analysis

In this subsection, we show that the new protocol guarantees typical security goals for verifier-based APAKE protocols and resists various known attacks.

(1) *Resistance to off-line dictionary attacks.* In the protocol, the passwords of clients only appear in the CPA-secure ciphertext c_i generated by the client or those CCA-secure ciphertexts $c¢_j 1 \# j \quad n$ generated by the server. According to the semantic security of these public key

encryption schemes, an adversary cannot mount an off-line dictionary attack on them. On the other hand, the smoothness property of the SPHF system guarantees that, except for negligible probability, no information of the passwords could be obtained by the adversary from the outputs of the SPHFs. Hence, the adversary cannot conduct off-line dictionary attacks.

(2) *Mutual authentication.* Recall that the ciphertext c'_i generated by the server using a substring tk_i derived from the SPHF hash value as its randomness. If the server holds the right verifier (s_i, H_i) corresponding to the client's password p_i, the correctness property of the SPHF system would guarantee that this value can be recomputed by the client. However, if the client's password and the server's i-th verifier are not consistent, they will get different tk_i. Therefore, the ciphertext c'_i ensures the authentication from the server to the client. In addition, the message t_C is used as an authenticator from the client to the server. Hence, we conclude that this protocol provides explicit mutual authentication.

(3) *Client anonymity.* The server generates a projective key hp_j and computes a SPHF hash value for each of the potential client. No matter which client is involved in the communication by using the correct password, she will compute the same SPHF hash value as the server. Meanwhile, the server cannot detect the actual identity of the client since the same substring tp_1 is used by any client $C_i \in \Gamma$ for client's authentication and the final session key's generation.

(4) *Security against stolen-verifier attack.* If the server is compromised and the verifiers are leaked, the adversary would obtain all tuples $\{s_j, H_j\}_{1 \le j \le n}$. It is obvious that the best way for the adversary to derive the password from $H_j = s_j^{\mathcal{F}(p_j)}$ is to mount a brute-force guessing attack. Additionally, since different bases are used for different client, the adversary can only guess the password one by one. In fact, the above security is actually guaranteed by the tight one-wayness of the underlying password hashing scheme.

5.2. Rigorous security proof

In this subsection, we prove that the 2-flow version of our verifier-based APAKE protocol satisfies the AKE security and anonymity defined in the security model presented in Section 2. It is obvious that the 3-flow version will additionally provide explicit client authentication by sending the last authenticating message to the server.

Theorem 1. Assume that $(KGen, Enc, Dec)$ is the ElGamal encryption scheme which is CPA secure, $(KGen', Enc', Dec')$ is a labeled public key encryption scheme with PCA security [31], $(SignKG, Sign, Verify)$ is an existential unforgeable one-time signature, and the SPHF system and password hashing scheme are secure. Then, the verifier-based APAKE protocol presented in Section 4.2 guarantees AKE security. More specifically, the advantage of any adversary $\mathcal{A}$ against the AKE security in the security model is

$$Adv_P^{AKE}(\mathcal{A}) \le C' q_{send}^{s'} + (q_{exe} + q_{send}) \cdot (Adv_{\mathcal{E}}^{cpa}(t) + n \cdot Adv_{\mathcal{E}'}^{pca}(t) + n \cdot \varepsilon_{SPHF}(k)) + Adv_{\mathcal{S}}^{EUF}(t), \quad (2)$$

where t is the maximum running time of the adversary $\mathcal{A}$, q_{send}, q_{exe} are the maximum number of *Execute* and *Send* queries asked by $\mathcal{A}$, $Adv_{\mathcal{E}}^{cpa}(t), Adv_{\mathcal{E}'}^{pca}(t), Adv_{\mathcal{S}}^{EUF}(t)$ denote the adversaries' advantages against the encryption scheme $\mathcal{E}, \mathcal{E}'$ and the one-time signature $\mathcal{S}$, $\varepsilon_{SPHF}(k)$ denotes a negligible upper bound of the statistical distance involved in the smoothness definition of the SPHF system, where C' and s' are the Zipf parameters [1,29].

Proof. The proof consists of a sequence of incrementally changed games $G_0, G_1, \ldots, G_9$, starting from the game G_0 representing the real attack in the security model, and ending with a game G_9 in which the adversary cannot get any information of the passwords from the session keys and the exchanging messages. For this purpose, we will modify the simulation of *Execute* and *Send* queries step by step. Denote by $Adv_k(A)$ the adversary's advantage in the k-th game. Denote by $Send_0(C_i^r, \text{start})$ the first send query to start the r-th session C_i^r of some client $C_i \in \mathcal{C}$, and denote by $Send_1(S^d, < S, c_i >)$ and $Send_2(C_i^r, < \mathbf{hp}, VK, c_i', \sigma >)$ the queries relative to the first and second message respectively. Furthermore, assume that the simulator maintains a list of the form $PWList = \{(P_j, s_j, H_j)\}_{1 \le j \le n}$ recording the compatible password and verifier pair registered by all honest clients.

Game G_0: This game is the real attack as defined in Section 2, where the simulator simulates all the honest sessions and all the queries asked by the adversary are answered honestly. Thus, we have

$$Adv_P^{AKE}(\mathcal{A}) = Adv_0(\mathcal{A}). \tag{3}$$

Game G_1: In this game, we begin to modify the way *Execute* queries between honest client and server sessions are answered. When a query of the form $Execute(C^r, S^d)$ is received, the simulator first discriminates whether the client C and the server S hold compatible passwords, through checking whether the corresponding tuple (P, s, H) appears in the list $PWList$.

(1) If they hold incompatible passwords, we simply let the client session abort. Note that the client session indeed aborts in this situation because of the smooth property of the SPHF system.

(2) If they hold compatible passwords, the simulator computes the SPHF value from the client side as $\overset{\circ}{tk}_i \,||\, \overset{\circ}{tp}_i = tk_j \,||\, tp_j = \mathrm{Hash}(L_{s,H}, hk_j, c_i)$, and removes the verifying process of the signature σ and the ciphertext c_i'. The correctness property of the SPHF system guarantees that these modifications do not alter the adversary's view at all. Therefore, $Adv_1(\mathcal{A}) = Adv_0(\mathcal{A})$.

Game G_2: In this game, we continue to modify the way *Execute* queries are answered, through replacing the ciphertext c_i by an encryption corresponding to a dummy password p_0 as $c_i = Enc(pk, P_0; r)$ where $P_0 = g^{\Phi(p_0)}$ and p_0 is not contained in the password dictionary. Because the encryption scheme $\mathcal{E}$ is CPA-secure, the ciphertexts of P_i and P_0 are indistinguishable. Consequently, one can easily bound the gap of the adversary's advantage in these two games as

$$|Adv_2(\mathcal{A}) - Adv_1(\mathcal{A})| \le q_{exe} \quad Adv_{\mathcal{E}}^{cpa}(t) \tag{4}$$

via a classical hybrid argument.

Game G_3: This game replaces the SPHF values $tk_j \,||\, tp_j, 1 \le j \le n$ by truly random strings. Since the ciphertext c_i no longer belongs to the language $L_{s.H}$ according to the above game, we could bound the change of the adversary's advantage via resorting to the smoothness property of the SPHF system and the hybrid technique. Note that there are at most $q_{exe} \cdot n$ *Execute* queries are modified in this game. Thus, it can be concluded that

$$|Adv_3(\mathcal{A}) - Adv_2(\mathcal{A})| \le q_{exe} \quad n \cdot \varepsilon_{SPHF}(k). \tag{5}$$

Game G_4: In this game, we replace the ciphertext c'_j by an encryption corresponding to a dummy password p_0 as $c''_j = Enc(pk', s_j \| H_{j,0}; label; tk_j)$, where $H_{j,0} = s_j^{\mathcal{F}(p_0)}$, for all $1 \le j \le n$. Owing to the PCA security of the encryption scheme $\mathcal{E}'$, which implies CPA security as well [31], the advantage gap caused by replacing one of the above ciphertext is at most $Adv_{\mathcal{E}'}^{pca}(t)$. As a result of this fact, with the aid of a classical hybrid argument, the overall difference of the adversary's advantage in these two games is at most

$$|Adv_4(\mathcal{A}) - Adv_3(\mathcal{A})| \le q_{exe} \cdot n \cdot Adv_{\mathcal{E}'}^{pca}(t). \tag{6}$$

Game G_5: Till now, we have finished the modification of $Execute$ queries. Before modifying the $Send$ queries, we first exclude the situations that the adversary forges a valid signature with respect to the one-time signature scheme. Denote by $Adv_{\mathcal{S}}^{EUF}(t)$ the advantage of the adversary against the one-time signature. Then, we can conclude that the gap of advantages caused by the above modification is at most $Adv_{\mathcal{S}}^{EUF}(t)$.

Game G_6: From now on, we will begin to modify the $Send$ queries. In this game, we first modify the way how $Send_2$ queries are answered. If a query of the form $Send_2(C_i^r, < \mathbf{hp}, VK, \mathbf{c}'s >)$ is sent to a client session C_i^r in the name of a server session S^d, the simulator first gets the verifier (s_i, H_i) corresponding to the client C_i via retrieving the list $PWList$. We say that the message $< \mathbf{hp}, VK, \mathbf{c}'s >$ is match-session-generated, if it was outputted by S^d as an answer to the $Send_1(S^d, < S, c_i >)$ query and $< S, c_i >$ is exactly the message outputted by C_i^r as an answer to a $Send_0$ query.

(1) If the message is not match-session-generated, the simulator checks through its PCA oracle whether $(c'_i s_i, H_i)$ is valid. If it is valid, we simply let the adversary win the game and denote this event by Win1; if it is not, we let the client directly abort this session.

(2) If the message is match-session-generated, the simulator checks whether (s_i, H_i) obtained above is compatible with the verifier held by the server session. If they are compatible, the simulator answers the query as in game G_1, computing the SPHF value from the client side as $\mathring{tk}_i \| \mathring{tp}_i = tk_j \| tp_j = \text{Hash}(L_{s,H}, hk_j, c_i)$, and removes the verification process against the ciphertext c'_i; If they are compatible, we let the client directly abort this session.

Note that except the event Win1 happens, modification in this game does not affect the adversary's advantage. Hence the advantage of the adversary satisfies $Adv_5(\mathcal{A}) \le Adv_6(\mathcal{A})$.

Game G_7: This game modifies the way $Send_0$ queries are answered, through replacing the ciphertext c_i by an encryption corresponding to a dummy password p_0. Similar to the analysis in game G_2, we could bound the advantage gap as

$$|Adv_7(\mathcal{A}) - Adv_6(\mathcal{A})| \le q_{send} \cdot Adv_{\mathcal{E}}^{cpa}(t). \tag{7}$$

Game G_8: In this game, we change the simulation of $Send_1$ queries as follows. If the message $< S, c_i >$ is generated by some honest client session or is generated by the adversary in the name of some honest client session but with incompatible password, the simulator replaces the SPHF values $tk_j \| tp_j, 1 \le j \le n$ by truly random strings. Similar to the analysis in game G_3, the advantage gap between this game and the last one is at most

$$| Adv_8(\mathrm{A}) - Adv_7(\mathrm{A}) | \mathrm{W} q_{send} \quad n \, ? e_{SPHF}(k) . \qquad (8)$$

Game G_9: This game continues the modification of $Send_1$ queries. Upon receiving a message $< S, c_i >$ that is generated by the adversary in the name of some client session C_i^r, where C_i and S hold incompatible passwords, we simply let the adversary win the game and denote this event by $\mathrm{Win2}$. As a consequence, $Adv_8(\mathrm{A}) \pounds Adv_9(\mathrm{A})$.

Game G_{10}: In this game, similar to game G_4, we replace the ciphertext $c_j¢$ by an encryption corresponding to a dummy password p_0 as $c_j ¢¢ = Enc\,(pk ? s_j \,||\, H_{j,0}; label; tk_j)$, where $H_{j,0} = s_j^{\mathrm{F}(p_0)}$, for all $1 \# j \quad n$. Recall that the adversary needs the access to the PCA oracles since in game G_6. By utilizing a classical hybrid argument, we could estimate the advantage gap as

$$| Adv_{10}(\mathrm{A}) - Adv_9(\mathrm{A}) | \mathrm{W} q_{send} \quad n \, ? Adv_{\mathrm{E}¢}^{pca}(t) \qquad (9)$$

In the last game G_{10}, all the messages and session keys are changed such that no information of the passwords is included. Thus, the adversary can only win the game on the case that the events $\mathrm{win1}$ or $\mathrm{win2}$ happens. Recall that all the passwords are chosen at random but not actually used, the probability that events $\mathrm{win1}$ or $\mathrm{win2}$ happens is bounded by $C¢ \times q_{send}^{s¢}$, , where $C¢$ and $s¢$ are the Zipf parameters [1,29]. To sum up, the global advantage of any adversary is bounded by the inequality (2) in the theorem.

Theorem 2. Based on the same assumptions as theorem 1, the verifier-based APAKE protocol guarantees client anonymity against the honest-but-curious server.

Proof. Note that the adversary against the client anonymity is considered to be an honest-but-curious one. We also recall that in the proof of theorem 1, all the messages outputted by $Execute, Send$ queries have been incrementally changed into messages and ciphertexts corresponding to the same dummy password p_0. Henceforth, no matter which client is actually involved in the protocol, by following the similar steps as in game G_1 to G_4, we can prove that the adversary's view is subject to the same distribution. In other words, for any two clients $C_i, C_j \in \Gamma$, the two distributions $\mathrm{P}_i, \mathrm{P}_j$ are identical to each other, thus are computational distinguishable.

6. Performance comparisons

In this section, we compare the proposed verifier-based APAKE protocol with other representative verifier-based APAKE, as well as PAKE, protocols, in terms of both security and efficiency.

Table 1. Performance comparisons.

	Protocol type	Security model	Resistance to impersonation	Number of rounds	Client computation	Server computation
Benhamouda et al. [25]	V-PAKE	standard	Yes	2	3 E + 4 DE	2 E + 4 DE
Hu et al. [17]	APAKE	standard	Yes	3	4E + 3 DE	3n E + 3n DE
Chen et al. [28]	V-APAKE	ROM	No	3	6 E	(2n+3) E
Yang et al. [27]	V-APAKE	standard	No	2	3 E + 4 DE	2n E + 4n DE
Ours protocol	V-APAKE	standard	Yes	2	4 E + 2 DE	2n E + 4n DE

The comparison results are presented in Table 1. The protocol type indicates the concrete type of the protocol, where V-PAKE, APAKE and V-APAKE are abbreviation of verifier-based PAKE, anonymous PAKE, and verifier-based anonymous PAKE respectively. The client and server computation stand for the main computational costs required by the client side and server side, where E denotes the time needed for computing one exponentiation and DE denotes the time needed for the computation of a double-exponentiation or multi-exponentiation.

As show in Table 1, our protocol is the only secure verifier-based APAKE protocol, while the verifier-based APAKE protocols in [27,28] are both vulnerable to impersonation attacks. The protocol requires only 2 rounds of communication, which is quite efficient compared to other protocols. When it comes to computational costs, the new protocol enjoys considerable efficiency in terms of number of exponentiations and double-exponentiations. The computational cost needed on the server side is almost equal to those in [17,27], and the cost on the client side is less than those in [17,25,27] thanks to the application of the PCA-secure encryption instead of CCA-secure encryption. Although Chen et al.'s protocol [28] is more efficient than ours, but it depends on the ROM in depth. On the contrary, our protocol is provably secure in the standard model.

7. Conclusions

In this paper, we first show that two recently proposed verifier-based APAKE protocols are vulnerable to impersonation attacks, thus does not reach their security goals. Then, we put forward a new verifier-based APAKE protocol based on implicitly checking the validity of the verifier via smooth projective hash functions for complex languages. The new protocol is with provable security in the standard model, and enjoys considerable computation and communication efficiency.

Acknowledgments

This work was supported in part by the National Nature Science Foundation of China under Grants Nos. 61702549, 61862011, 61872449, and in part by Guangxi Natural Science Foundation under Grant No. 2018GXNSFAA138116 and the Guangxi Key Laboratory of Cryptography and Information Security under Grant No. GCIS201704, and in part by Foundation of Science and Technology on Information Assurance Laboratory under Grant No. KJ-17-001.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. D. Wang, H. Cheng, P. Wang, et al., Zipf's law in passwords. *IEEE T. Inf. Foren. Sec.*, **12** (2017), 2776–2791.
2. S. M. Bellovin and M. Merritt, Encrypted key exchange: Password-based protocols secure against dictionary attacks, in Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy, *IEEE*, (1992), 72–84.

3. E. Bresson, O. Chevassut and D. Pointcheval, Security proofs for an efficient password-based key exchange, in Proceedings of ACM CCS 2003, *ACM Press*, (2003), 241–250.
4. M. Abdalla and D. Pointcheval, Simple password-based encrypted key exchange protocols, in Proceedings of CT-RSA 2005, *Springer*, (2005), 191–208.
5. J. Katz, R. Ostrovsky and M. Yung, Efficient password-authenticated key exchange using human-memorable passwords, in Proceedings of EUROCRYPT 2001, *Springer*, (2001), 475–494.
6. F. Benhamouda, O. Blazy, C. Chevalier, et al., New techniques for SPHFs and efficient one-round PAKE protocols, in Proceedings of CRYPTO 2013, *Springer*, (2013), 449–475.
7. X. Yi, F. Y. Rao, Z. Tari, et al., ID2S password-authenticated key exchange protocols, *IEEE T. Comput.*, **65** (2016), 3687–3701.
8. Y. Zhang, Y. Xiang, W. Wu, et al., A variant of password authenticated key exchange protocol, *Future Gener. Comput. Sy.*, **78** (2018), 699–711.
9. Z. Li and D. Wang, Two-round PAKE protocol over lattices without NIZK, in Proceedings of ICISC 2018, *Springer*, (2018), 138–159.
10. M. Abdalla, P. A. Fouque and D. Pointcheval, Password-based authenticated key exchange in the three-party setting, in Proceedings of PKC 2005, *Springer*, (2005), 65–84.
11. F. Wei, N. Kumar, D. He, et al., A general compiler for password-authenticated group key exchange protocol in the standard model, *Discrete Appl. Math.*, **241** (2018), 78–86.
12. M. Bellare, D. Pointcheval and P. Rogaway, Authenticated key exchange secure against dictionary attacks, in Proceedings of EUROCRYPT 2000, *Springer*, (2000), 139–155.
13. R. Canetti, S. Halevi, J. Katz, et al., Universally composable password-based key exchange, in Proceedings of EUROCRYPT 2005, *Springer*, (2005), 404–421.
14. D. Q. Viet, A. Yamamura and H. Tanaka, Anonymous password-based authenticated key exchange, in Proceedings of INDOCRYPT 2005, *Springer*, (2005), 244–257.
15. J. Yang and Z. Zhang, A new anonymous password-based authenticated key exchange protocol, in Proceedings of INDOCRYPT 2008, *Springer*, (2008), 200–212.
16. S. H. Shin, K. Kobar and H. Imai, Very-efficient anonymous password-authenticated key exchange and its extensions, in Proceedings of AAECC 2009, *Springer*, (2009), 149–158.
17. X. Hu, J. Zhan, Z. Zhang, et al., Anonymous Password Authenticated Key Exchange Protocol in the Standard Model, *Wirel. Pers. Commun.*, **96** (2017), 1451–1474.
18. Y. Yang, J. Zhou, J. Weng, et al., A new approach for anonymous password authentication, in Proceedings of ACSAC 09, *IEEE*, (2009), 199–208.
19. S. H. Shin and K. Kobara, Simple anonymous password-based authenticated key exchange (SAPAKE), reconsidered, *IEICE T. Fundam. Electron. Commun. Comput. Sci.*, **100** (2017), 639–652.
20. Z. Zhang, K. Yang, X. Hu, et al., Practical anonymous password authentication and TLS with anonymous client authentication, in Proceedings of ACM CCS 2016, *ACM Press*, (2016), 1179–1191.
21. Information technology—Security techniques—Anonymous entity authentication—Part 4: Mechanisms based on weak secrets, ISO/IEC standard 20009-4, 2017. Available from: https:// www. iso. org/ standard/64288.html.
22. K. Thomas, F. Li, A. Zand, et al., Data breaches, phishing, or malware? Understanding the risks of stolen credentials, in Proceedings of ACM CCS 2017, *ACM Press*, (2017), 1421–1434.

23. J. Li, L. Stecker, E. Zeigler, et al., Scramble the password before you type it, in Proceedings of World Conference on Information Systems and Technologies, *Springer*, (2018), 1097–1107.
24. Facebook Security Breach Exposes Accounts of 50 Million Users, 2018. Available from: https://www.nytimes.com/ 2018/09/28/ technology/facebook-hack-data-breach.html.
25. F. Benhamouda and D. Pointcheval, Verifier-based password-authenticated key exchange: new models and constructions. *IACR Crypt. ePrint Archive*, 2013: 833.
26. D. Pointcheval and G. Wang, VTBPEKE: verifier-based two-basis password exponential key exchange, in Proceedings of Asia CCS 2017, *ACM Press*, (2017), 301–312.
27. X. Yang, H. Jiang, Q. Xu, et al., A provably-secure and efficient verifier-based anonymous password-authenticated key exchange protocol, in Proceedings of Trustcom/BigDataSE/ISPA, 2016, *IEEE*, (2016), 670–677.
28. C. M. Chen, G. J. Wang, W. C. Fang, et al., A new verifier-based anonymous password-authenticated key exchange protocol, *J. Info. Hiding Multimedia Signal Process.*, **9** (2018), 1595–1602.
29. D. Wang and P. Wang, Two birds with one stone: Two-factor authentication with security beyond conventional bound, *IEEE T. Depend. Secure Comput.*, **15** (2018), 708–722.
30. F. Wei, P. Vijayakumar, Q. Jiang, et al., A mobile intelligent terminal based anonymous authenticated key exchange protocol for roaming service in global mobility networks, *IEEE T. Sustain. Comput.*, 2018.
31. M. Abdalla, F. Benhamouda and D. Pointcheval, Public-key encryption indistinguishable under plaintext-checkable attacks, *in* Proceedings of PKC 2015, *Springer*, (2015), 332–352.
32. R. Cramer and V. Shoup, Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption, in Proceedings of EUROCRYPT 2002, *Springer*, (2002), 45–64.
33. R. Gennaro and Y. Lindell, A framework for password-based authenticated key exchange, in Proceedings of EUROCRYPT 2003, *Springer*, (2003), 524–543.
34. J. Katz and V. Vaikuntanatha, Round-optimal password-based authenticated key exchange, in Proceedings of TCC 2011, *Springer*, (2011), 293–310.
35. M. Abdalla, F. Benhamouda and D. Pointcheval, Disjunctions for hash proof systems: New constructions and applications, in Proceedings of EUROCRYPT 2015, *Springer,* (2015), 69–100.
36. S. Even, O. Goldreich and S. Mical, On-line/off-line digital signatures, in Proceedings of CRYPTO 89, *Springer*, (1989), 263–275.
37. F. Kiefer and M. Manulis, Zero-knowledge password policy checks and verifier-based PAKE, in Proceedings of ESORICS 2014, *Springer,* (2014), 295–312.
38. A. Groce and J. Katz, A new framework for efficient password-based authenticated key exchange, in Proceedings of ACM CCS 2010, *ACM Press*, (2010), 516–525.