Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8893628, 13 pages
https://doi.org/10.1155/2020/8893628

Research Article

Two-Round Password-Based Authenticated Key Exchange from Lattices

Anqi Yin,[1] **Yuanbo Guo**,[1] **Yuanming Song**,[2] **Tongzhou Qu**,[1] **and Chen Fang**[1]

[1]*Zhengzhou Institute of Information Science and Technology, Henan 450001, China*
[2]*School of EECS, Peking University, Beijing 100871, China*

Correspondence should be addressed to Yuanbo Guo; guo_yuanbo@126.com

Received 7 August 2020; Revised 7 September 2020; Accepted 30 September 2020; Published 14 December 2020

Academic Editor: Qi Jiang

Password-based authenticated key exchange (PAKE) allows participants sharing low-entropy passwords to agree on cryptographically strong session keys over insecure networks. In this paper, we present two PAKE protocols from lattices in the two-party and three-party settings, respectively, which can resist quantum attacks and achieve mutual authentication. The protocols in this paper achieve two rounds of communication by carefully utilizing the splittable properties of the underlying primitive, a CCA (Chosen-Ciphertext Attack)-secure public key encryption (PKE) scheme with associated nonadaptive approximate smooth projection hash (NA-ASPH) system. Compared with other related protocols, the proposed two-round PAKE protocols have relatively less communication and computation overhead. In particular, the two-round 3PAKE is more practical in large-scale communication systems.

1. Introduction

Password-based authentication key exchange (PAKE) is theoretically fascinating, since it allows participants sharing short, low-entropy passwords to agree on cryptographically strong session keys over insecure networks [1, 2]. PAKE protocols are very practical as passwords are probably the most common and widely used authentication method [3–6], and password-based authentication can avoid the dependence on public key infrastructure and secure hardware; thereby, it improves the convenience of the system.

However, the use of shared short, low-entropy passwords will expose PAKE to greater security threats. This is because it must be ensured that the protocol is immune to off-line dictionary attacks, in which the adversary can exhaust all possible passwords to determine the correct one [7, 8]. Another observation is that an adversary can always succeed by guessing the password as the password dictionary is relatively small (usually polynomial in the security parameter), referred to as an on-line attack. The aim of PAKE is thus to limit the adversary to such an attack only [9].

The first successful password-based authenticated key exchange agreement methods were Encrypted Key Exchange methods described by Steven M. Bellovin and Michael Merritt in 1992 [10]. Initial PAKE protocols are generally based on "hybrid" models [11, 12], in which the clients need to store the public key of the server besides sharing a password with the server. The requirement of securely storing long, high-entropy public key is not friendly in multiserver environment. This motivates the study towards password-only protocols [13–16] where clients need to remember only a short password. Most PAKE protocols above provide only informal security arguments. Thus, Bellare et al. [17] and Boyko et al. [18] gave formal models of security of the password-only setting and proved security in the ideal cipher model and random oracle model, respectively. And then, Goldreich and Lindell [19] presented a provably secure password-only key exchange under standard cryptographic assumptions. Their work shows the possibility for password-based authentication under very weak assumptions, but the protocol itself is far from practical.

Later, many provably secure PAKE protocols based on various hardness assumptions were proposed. The research is mainly divided into two directions. The former is PAKE in the random oracle/ideal cipher model, which aims to achieve the highest possible levels of performance [20–22]. The latter is dedicated to seeking more efficient PAKE in the standard model [5, 23–25].

The first efficient PAKE protocol under standard model was proposed by Katz et al. [7]. They utilized CCA2 (Adaptive Chosen-Ciphertext Attack)-secure encryption system and corresponding smooth projection hash (SPH) function for key exchange to construct their scheme. Then, Gennaro and Lindell [9] abstracted their work and presented a corresponding PAKE framework, referred to as KOY/GL framework without mutual authentication. Based on KOY/GL framework, Jiang and Gong [26] showed a three-round PAKE supporting mutual authentication. Groce and Katz [5] then generalized the protocol by Jiang and Gong and gave a new PAKE framework in the common reference string model (CRS), referred to as JG/GK framework. Subsequently, based on the above two framework, a series of PAKE protocols [27–29] with different security are proposed for different application scenarios.

Most above schemes are two-party PAKE (2PAKE), requiring every two participants to share a password, which is not adaptable to large communication systems. In contrast, three-party PAKE (3PAKE) enables each client to share a password with the server for authentication, thereby avoiding the limitation of 2PAKE. Abdalla et al. [30] gave a general structure of the 3PAKE protocol for the first time. Subsequently, cryptographers designed a series of 3PAKE protocols with different efficiency and security [31, 32].

The security of most protocols above relies on traditional difficult problems (such as large integer factorization and discrete logarithm problems), so they cannot resist quantum attacks. However, the public-key cryptosystem based on the lattice assumption can resist quantum attacks. In addition, the operation on the lattice is matrix-vector multiplication which can be practically implemented by parallel computing. Therefore, the public-key cryptosystem from lattices will be more secure and efficient.

In lattice-based cryptosystem, the research on PAKE is relatively insufficient. In 2009, Katz et al. [33] presented the first lattice-based 2PAKE protocol based on KOY/GL framework. They proposed their protocol by constructing the first lattice-based CCA-secure encryption system and its corresponding approximate smooth projected hash (ASPH) function. Ding et al. [34] then applied the encryption system and ASPH function from Katz et al. [33] to JG/GK framework, and a more efficient protocol is given in the standard model.

Ye et al. [35] proposed the first 3PAKE protocol based on the JG/GK framework from lattices, and they proved its security under the standard model. This is a three-round protocol that implements explicit mutual authentication between the client and the server. In 2017, Xu et al. [36] proposed a provably secure 3PAKE protocol based on the R-LWE (ring learning with error) problem according to the idea of DH, but this protocol suffers from low efficiency.

Zhang et al. [1] applied a splittable public key encryption system to the KOY/GL framework and proposed a lattice-based PAKE, requiring only two-round communication, so it is more efficient. However, Zhang's 2PAKE cannot be directly applied to the 3PAKE protocol, because another function is needed to compute the client-side information.

In this paper, we present efficient new constructions of 2PAKE and 3PAKE based on the learning with error (LWE) problem based on ideas of [1, 34, 35]. We then prove the security of the proposed protocols in the random oracle model. Significant security (resistance to quantum attacks) and efficiency improvements would also be obtained when basing the protocol on lattice assumption. Compared with the general structure [30], the new protocols reduce the number of communications, thereby improving efficiency. Our protocols also achieve mutual authentication between participants, so they can resist unpredictable on-line attacks. And the proposed two-round 3PAKE is adaptable to large-scale communication systems.

2. Preliminaries

2.1. Notations. We denote the logarithm with base 2 (resp., the natural logarithm) by lb (resp., log). Vectors are expressed in columns and bold lower-case letters (for example, $\mathbf{x}$). A matrix is considered as a collection of column vectors and is represented by bold capital letter (such as $\mathbf{X}$). We denote the concatenation of $\mathbf{X}$ and $\mathbf{Y}$ as $(\mathbf{X}||\mathbf{Y})$. Let $x \longleftarrow_r K$ denote the random sampling of variable x from the distribution K. For any string $x, y \in \{0,1\}^\ell$, $\mathrm{Ham}(x, y)$ represents the Hamming distance between x and y. Table 1 summarizes the description of other symbols used in this paper.

2.2. Security Model

2.2.1. Participants. Participants include honest clients A, B, D, $\cdots \in \mathcal{U}$, malicious clients $\mathcal{A}, \mathcal{M}, \cdots \in \mathcal{E}$ and trusted server $C \in \mathcal{S}$. For simplicity, we assume that the server set $\mathcal{S}$ contains only one element C. For each distinct $A, B \in$ Client, assume that A and B share a long-term key called password pw_{AB}. We simply assume that pw_{AB} is independently and uniformly chosen at random from the password dictionary $\mathcal{D}$. But our proof of security extends to more general cases. In the following, we refer to honest clients directly as clients, and the malicious clients as adversaries.

Each participant can execute multiple protocols with different partners at the same time. We call the execution of a protocol an instance. Denote the instance i of client A and the instance j of server C by Π_A^i, and Π_C^j, respectively.

Each instance Π_A^i maintains a local state vector $(\mathrm{sid}_A^i, \mathrm{pid}_A^i, \mathrm{sk}_A^i, \mathrm{acc}_A^i, \mathrm{term}_A^i)$, where sid_A^i represents the session ID, recording all messages sent and received by Π_A^i in order; pid_A^i ($\mathrm{pid}_A^i \neq A$) represents the partner ID, the participant with which Π_A^i believes it is interacting; sk_A^i denotes the session key of Π_A^i; acc_A^i and term_A^i are Boolean variables, indicating whether the Π_A^i is accepted or terminated.

Table 1: Symbol description.

Symbol	Descriptions
κ	Security parameter
pw	Password
$\mathcal{D}$	Password dictionary
$\lvert S \rvert$	The size of set S
ε	A real number in $(0, 1/2)$.
$\ell(\ell = \ell(\kappa))$	An integer related to κ
sk	Session key
$\mathrm{sk}_{\mathrm{pk}}$	The corresponding private key of pk

Before giving a formal definition of adversarial abilities, we first define partnering, correctness, and freshness as follows.

Partnering. If (1) $\mathrm{sid}_A^i = \mathrm{sid}_B^j \neq \perp$ and (2) $\Pi_A^i = B$ and $\Pi_B^j = A$, instances Π_A^i and Π_B^j are partnered.

Correctness. We say that the protocol between partnered instances Π_A^i and Π_B^j is correct if they are both accepted and establish the same session key, that is, $\mathrm{acc}_A^i = \mathrm{acc}_B^j = 1$ and $\mathrm{sk}_A^i = \mathrm{sk}_B^j$.

Freshness. If none of the following cases happens, the instance Π_A^i is fresh: (1) adversary $\mathcal{A}$ have sent a Reveal query to Π_A^i; (2) Π_A^i and Π_B^j have become partners and adversary $\mathcal{A}$ has sent a Reveal query to Π_B^j. The definition of Reveal query will be given below.

2.2.2. Adversarial Abilities. It is assumed that the protocol is executed over a generally insecure network. Adversary $\mathcal{A}$ can eavesdrop, intercept, inject, and tamper with messages among different participants. $\mathcal{A}$ can also obtain the session key of the accepted instances. The following oracle queries model the adversarial abilities, that is, the adversary's interaction with various instances.

(i) *Execute* (A, i, B, j) *Query*. The oracle models off-line attacks for passive adversaries. This oracle executes the protocol between the client instances Π_A^i and Π_B^j, and it updates the state vectors according to the specific protocol. And return the transcript of this execution to $\mathcal{A}$.

(ii) *Send* (A, i, M). This oracle sends the message M to the client instance Π_A^i to update the corresponding state vector appropriately. Finally, it returns the output message of Π_A^i to $\mathcal{A}$. This oracle models on-line attacks from active adversaries.

(iii) *Reveal* (A, i). This oracle returns the session key of the accepted instance Π_A^i to $\mathcal{A}$, thereby modelling the leakage of the session key. This oracle corresponds to an on-line attack from active adversaries.

(iv) *Test* (A, i). The oracle selects a random bit $b \longleftarrow_r \{0, 1\}$. And if $b = 1$, the real session key of Π_A^i is returned to $\mathcal{A}$. Otherwise, it returns a uniform string of appropriate length. Note that $\mathcal{A}$ can only query this oracle once, and $\mathcal{A}$ is only allowed to query a fresh instance. This oracle is used to define security and does not model any adversarial capability in the real world.

2.2.3. The Advantage of the Adversary. The security of the protocol is defined by a security experiment: the adversary is allowed to send a series of queries above, but the Test query can only be sent once; and the experiment ends with $\mathcal{A}$ outputting bit b', a guess of b. Informally, $\mathcal{A}$ succeeds if (1) $b' = b$, that is, $\mathcal{A}$'s guess is correct, representing that the session key is insecure, (2) $\mathcal{A}$ makes the instance accepted but there is no corresponding partner, indicating that the protocol cannot achieve mutual authentication. Formally, we use Success to indicate the success of $\mathcal{A}$. The advantage of the adversary in attacking the protocol Π is defined as $\mathrm{Adv}_{\Pi,A} \stackrel{\mathrm{def}}{=} 2\Pr[\mathrm{Success}] - 1$.

2.2.4. Secure Protocol. Since the size of the password dictionary $\mathcal{D}$ is usually small, a PPT (Probabilistic Polynomial Time) adversary can always succeed by exhausting $\mathcal{D}$ in an on-line attack. Therefore, informally, if on-line attack is the best attack method for all PPT adversaries, the PAKE protocol is secure. Formally, we give the following definition of the secure protocol.

Definition 1. (secure protocol). A protocol is a secure PAKE with mutual authentication if for all password in dictionary $\mathcal{D}$ and for any PPT adversary making at most $Q(\kappa)$ on-line attacks, it holds that $\mathrm{Adv}_{\Pi,\mathcal{A}}(\kappa) \leq Q(\kappa)/|\mathcal{D}| + \mathrm{negl}(\kappa)$ for some negligible function $\mathrm{negl}(\bullet)$.

2.3. Splittable Labeled PKE System from Lattices

2.3.1. Splittable Labeled PKE. Let the splittable labeled CCA-secure PKE be SPKE = (KeyGen, Enc, Dec). KeyGen is a key generation algorithm outputting the public and secret key pair (pk, sk). Enc is an encryption algorithm that returns $c = (u, v) = \mathrm{Enc}(\mathrm{pk}, \mathrm{label}, \mathrm{pw})$, where $u = f(\mathrm{pk}, \mathrm{pw})$, $v = g(\mathrm{pk}, \mathrm{label}, \mathrm{pw})$, f and g are two different subfunctions that constitute SPKE. Dec is a decryption algorithm defined as $\mathrm{pw} \longleftarrow \mathrm{Dec}(\mathrm{label}, \mathrm{sk}, c)$. For any v' and $\mathrm{label}' \in \{0, 1\}^*$, under the random selection of sk and r, the probability that $\mathrm{Dec}(\mathrm{sk}, \mathrm{label}, (u, v)) \notin \{\perp, \mathrm{pw}\}$ is negligible in κ.

The "splittable" attribute is also reflected in the security of the public-key cryptosystem. When proving the CCA security of the splittable cryptosystem, the challenge phase of the CCA game should be modified as follows: (1) the adversary $\mathcal{M}$ first sends two plaintexts $\mathrm{pw}_0, \mathrm{pw}_1 \in \mathcal{D}$ of equal length. (2) The challenger $\mathcal{CL}$ randomly chooses $b^* \longleftarrow_r \{0, 1\}$ and $r^* \longleftarrow_r \{0, 1\}^*$. Then, $\mathcal{CL}$ computes $\mathrm{u}^* = \mathrm{f}(\mathrm{pk}, \mathrm{pw}_{b^*}, r^*)$ and returns u^* to $\mathcal{M}$. (3) Upon receiving u^*, the adversary $\mathcal{M}$ submits $\mathrm{label} \longleftarrow \{0, 1\}^*$. (4) $\mathcal{CL}$ computes $\mathrm{v}^* = \mathrm{g}(\mathrm{pk}, \mathrm{label}, \mathrm{pw}_{b^*}, r^*)$ and returns v^* to $\mathcal{M}$.

Definition 2. (CCA security of SPKE). SPKE is a secure CCA-secure public-key encryption scheme if for any PPT

adversary, it holds that $\mathrm{Adv}_{\mathrm{SPKE},\mathcal{M}}^{IND-CCA}(\kappa) \leq negl(\kappa)$ for some negligible function negl(•).

In this paper, we denote the splittable labeled CCA-secure PKE based on LWE problem [1] by $\Sigma = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$, and we will use it to construct two-round PAKEs. The definitions of the cryptographic primitives (TrapGen, CRSGen, Prove, Verify, and Solve) on which Σ is based can be found in [1]. TrapGen is a trapdoor generation algorithm for generating public keys and corresponding trapdoors; CRSGen is a common reference string generator, usually implemented by hardware; the Proof/Verify algorithm is similar to the signature/verification algorithm to ensure the integrity of (A_0, A_1, u, v, β); Solve is a trapdoor solving algorithm corresponding to TrapGen.

2.3.2. A Splittable Labeled PKE from Lattices [1]. Suppose $n_1, n_2 \in \mathbb{Z}$ and prime q is polynomial with respect to the security parameter κ. Let $n = n_1 + n_2 + 1$, $m = \mathrm{O}(\mathrm{nlogq}) \in \mathbb{Z}$. $\alpha, \beta \in \mathbb{R}$ are the parameters of the systems. The splittable labeled PKE from lattices $\Sigma = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ is defined as follows:

KeyGen(1^κ): given security parameter κ, we have $(A_0, R_0) \longleftarrow \mathrm{TrapGen}(1^n, 1^m, q)$, $(A_1, R_1) \longleftarrow \mathrm{TrapGen}(1^n, 1^m, q)$, and $\mathrm{CRS} \longleftarrow \mathrm{CRSGen}(1^\kappa)$. And return $(\mathrm{pk}, \mathrm{sk}) = ((A_0, A_1, \mathrm{CRS}), R_0)$.

Enc(pk, label, pw): given $\mathrm{pk} = (A_0, A_1, \mathrm{CRS})$, $\mathrm{label} \longleftarrow \{0,1\}^*$, and plaintext $\mathrm{pw} \in \mathcal{D}$, choose $\mathrm{s}_0, \mathrm{s}_1 \longleftarrow_r \mathbb{Z}_q^{n_1}$, $e_0, e_1 \longleftarrow_r D_{\mathbb{Z}^m,\alpha q}$. Return the ciphertext $\mathrm{C} = (u, v, \pi)$, where

$$u = A_0^T \begin{pmatrix} s_0 \\ 1 \\ pw \end{pmatrix} + e_0, v = A_1^T \begin{pmatrix} s_1 \\ 1 \\ pw \end{pmatrix} + e_1 \quad (1)$$

and $\pi \longleftarrow \mathrm{Prove}(\mathrm{CRS}, (A_0, A_1, u, v, \beta), (s_0, s_1, \mathrm{pw}), \mathrm{label})$.

Dec(sk, label, C): given $\mathrm{sk} = R_0$, $\mathrm{label} \longleftarrow \{0,1\}^*$, and the ciphertext $\mathrm{C} = (u, v, \pi)$, if $\mathrm{Verify}(\mathrm{CRS}, (A_0, A_1, u, v, \beta), \pi, \mathrm{label}) = 0$, return $\perp$. Otherwise, compute

$$t = \begin{pmatrix} s_0 \\ 1 \\ pw \end{pmatrix} \longleftarrow Solve(A_0, R_0, C) \quad (2)$$

Finally, return $\mathrm{pw} \in \mathbb{Z}_q^{n_2}$.

2.4. Nonadaptive Approximate Smooth Projective Hash (NA-ASPH) System. Based on the smooth projected Hash function [37], Katz et al. [33] proposed an Approximate Smooth Projective Hash (ASPH) function that can be used to construct a lattice-based PAKE protocol. In our application, we use a modified definition of ASPH [1] from Katz's, referred to as nonadaptive approximate smooth projection hash (NA-ASPH) function.

Suppose that $\Sigma(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ is a semantically secure PKE system from lattices. Assume that a valid ciphertext $c = (u, v)$ can be easily parsed as the output of the function pair (f, g). We use KeyGen to generate a key pair (pk, sk) and we use C_{pk} to represent the valid ciphertext space corresponding to the public key pk. Define

$$\begin{aligned} X &= \{(\mathrm{label}, c, \mathrm{pw}) \mid (\mathrm{label}, c) \in \mathrm{C}_{\mathrm{pk}}, \mathrm{pw} \in \mathcal{D}\} \\ L &= \{(\mathrm{label}, c, \mathrm{pw}) \mid \mathrm{label} \in \{0,1\}^*, c = \mathrm{Enc}(\mathrm{pk}, \mathrm{label}, \mathrm{pw}, r)\} \\ \bar{L} &= \{(\mathrm{label}, c, \mathrm{pw}) \mid \mathrm{label} \in \{0,1\}^*, \mathrm{pw} = \mathrm{Dec}(\mathrm{sk}, \mathrm{label}, c)\}. \end{aligned} \quad (3)$$

Based on $\Sigma(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, we introduce the ϵ-NA-ASPH function defined by the sampling algorithm, which outputs $(K, \ell, \mathbb{H} = \{H_k : X \longrightarrow \{0,1\}^\ell\}, S, \alpha : K \longrightarrow S)$ given the public key pk of Σ (where K is the hash key space and $k \in K$, α is the key projection function from K to S, and S is the projection key space; the domain and value range of the ϵ-NA-ASPH are X and $\{0,1\}^\ell$, respectively), such that

(1) There exist efficient algorithms for sampling a hash key $k \longleftarrow_r K$, computing $H_k(x) = H_k(u, \mathrm{pw})$ for all $x = (\mathrm{label}, (u, v), \mathrm{pw}) \in X$ and computing $s = \alpha(k)$ for all $k \in K$

(2) For all $k \longleftarrow_r K$, $x = (\mathrm{label}, (u, v), \mathrm{pw}) \in L$ and randomness r, there are efficient algorithms for computing $\mathrm{Hash}(s, x, r) = \mathrm{Hash}(s, (u, \mathrm{pw}), r)$ given $u = f(\mathrm{pk}, \mathrm{pw}, r)$ and $v = g(\mathrm{pk}, \mathrm{label}, \mathrm{pw}, r)$.

The ϵ-NA-ASPH has the following properties:

(i) *Correctness.* For all $x = (\mathrm{label}, (u, v), \mathrm{pw}) \in L$ and $s = \alpha(k)$, it holds that $\Pr[\mathrm{Ham}((H_k(u, \mathrm{pw})), \mathrm{Hash}(s, (u, \mathrm{pw}), r)) \geq \epsilon] = \mathrm{negl}(\kappa)$ for some negligible function negl(•).

(ii) *Smoothness.* For any (even unbounded) function h: $S \longrightarrow X \setminus \bar{L}$, $k \longleftarrow_r K$, $s = \alpha(k)$, $x = h(s)$ and $\gamma \longleftarrow_r \{0,1\}^\ell$, the distributions $(\mathrm{s}, H_k(x))$ and (s, γ) are statistically indistinguishable in the security parameter κ

NA-ASPH has three modifications compared with ASPH in [33]: (1) the projection function α depends only on the hash key k; (2) $H_k(x) = H_k(u, \mathrm{pw})$ is determined by the hash key k, the first part of the ciphertext $c = (u, v)$ and the plaintext pw; (3) for all $x = h(s) \notin \bar{L}$, the smoothness holds. The first modification here enables the protocol proposed to achieve two rounds of communication, and the latter two are prepared to prove the security of the proposed protocol.

3. A Two-Round 2PAKE Protocol

We now describe the proposed two-round 2PAKE, which is based on the protocol by Groce and Katz [9] and the splittable PKE scheme by Zhang and Yu [1].

3.1. Primitives. The primitives we use are the following: (1) a splittable labeled PKE scheme $\Sigma(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ with an associated ϵ-NA-ASPH $(\mathrm{K}, \ell, \mathbb{H} = \{H_k : X \longrightarrow \{0,1\}^\ell\}, S, \alpha : K \longrightarrow S)$, where the scheme Σ can be divided into function

Table 2: An honest execution of the two-round 2PAKE protocol.

Two-round 2PAKE	
Client A	Server C
$r_1 \longleftarrow_r \{0,1\}^*$	
$k_1 \longleftarrow_r K$	
$s_1 = \alpha(k_1)$	
$\text{label}_1 = A\|\|C\|\|s_1$	
$u_1 = f(\text{pk}, \text{pw}_A, r_1)$	
$v_1 = g(\text{pk}, \text{label}_1, \text{pw}_A, r_1)$	
$A\|\|s_1\|\|c_1 = (u_1, v_1) \longrightarrow$	
	$k_2 \longleftarrow_r K, k_2^* \longleftarrow_r K$
	$s_2 = \alpha(k_2), s_2^* = \alpha(k_2^*)$
	$r_j\|\|\tau_j\|\|\text{sk}_j \longleftarrow H_{k_2^*}(u_1, \text{pw}_A)$
	$u_2 = f(\text{pk}, \text{pw}_A, r_j)$
	$\text{tk} = \text{Hash}(s_1, (u_2, \text{pw}_A), r_j) \oplus H_{k_2}(u_1, \text{pw}_A)$
	$\Delta = \text{tk} \oplus \text{ECC}(H_{k_2^*}(u_1, \text{pw}_A))$
	$\text{label}_2 = C\|\|A\|\|s_1\|\|s_2\|\|s_2^*\|\|\Delta\|\|c_1$
	$v_2 = g(\text{pk}, \text{label}_2, \text{pw}_A, r_j)$
$s_2\|\|s_2^*\|\|\Delta\|\|c_2 = (u_2, v_2) \longleftarrow$	
$\text{tk}' = H_{k_1}(u_2, \text{pw}_A) \oplus \text{Hash}(s_2, (u_1, \text{pw}_A), r_1)$	
$H' = \text{ECC}^{-1}(\text{tk}' \oplus \Delta)$	
If $\text{Ham}(H', \text{Hash}(s_2^*, (u_1, \text{pw}_A), r_1)) \le 2\varepsilon/\ell$	
$r_i\|\|\tau_i\|\|\text{sk}_i \longleftarrow H'$	
$\text{sk}_{AC} = \text{sk}_i$	$\text{sk}_{CA} = \text{sk}_j$

pair (f, g); (2) error-correcting code $\text{ECC} : \{0,1\}^\kappa \longrightarrow \{0,1\}^\ell$ and the corresponding decoding algorithm $\text{ECC}^{-1} : \{0,1\}^\ell \longrightarrow \{0,1\}^n$. ECC can correct 2ϵ-fraction of errors. We assume that if ρ is sampled uniformly from $\{0,1\}^\ell$, $\mu = \text{ECC}^{-1}(\rho)$ is uniformly distributed in $\{0,1\}^n$ provided that $\mu \neq \perp$.

3.2. Initialization. The proposed protocol requires the public key pk of the scheme Σ, also known as the common reference string (CRS). We want to emphasize that during the execution of the entire protocol, no participant needs to know the private key corresponding to the public key.

3.3. Protocol Execution. A high-level depiction of the two-round 2PAKE protocol is given in Table 2. We assume that the execution of the protocol is between client A and server C. Client A and server C share a password $\text{pw}_A \in \mathcal{D}$. When client A wants to initialize an authentication with the server C, A chooses a random tape $r_1 \longleftarrow_r \{0,1\}^*$ for encryption and a hash key $k_1 \longleftarrow_r K$ for the NA-ASPH. Then, client A computes the projection key $s_1 = \alpha(k_1)$, sets $\text{label}_1 = A\|C\|s_1$. And A computes $c_1 = (u_1, v_1) = \Sigma(\text{pk}, \text{label}_1, \text{pw}_A, r_1)$, where $u_1 = f(\text{pk}, \text{pw}_A, r_1)$ and $v_1 = \text{g}(\text{pk}, \text{label}_1, \text{pw}_A, r_1)$. Finally, client A sends the message $(A\|s_1\|c_1 = (u_1, v_1))$ to server C.

After receiving $(A\|s_1\|c_1)$ from client A, server C checks whether c_1 is a valid ciphertext with respect to pk and $\text{label}_1 = A\|C\|s_1$. If not, C rejects and the protocol aborts. Otherwise, C chooses hash keys $k_2 \longleftarrow_r K$ and $k_2^* \longleftarrow_r K$, and it computes projection keys $s_2 = \alpha(k_2)$, $s_2^* = \alpha(k_2^*)$, $r_j\|r_j\|\text{sk}_j \longleftarrow H_{k_2^*}(u_1, \text{pw}_A)$, $u_2 = f(\text{pk}, \text{pw}_A, r_j)$, $\text{tk} = \text{Hash}(s_1, (u_2, \text{pw}_A), r_j) \oplus H_{k_2}(u_1, \text{pw}_A)$, and $\Delta = \text{tk} \oplus \text{ECC}(H_{k_2^*}(u_1, \text{pw}_A))$. Server C then sets $\text{label}_2 = C\|A\|s_1\|s_2\|s_2^*\|\Delta$

$\|c_1$and computes $v_2 = g(\text{pk}, \text{label}_2, \text{pw}_A, r_j)$. Finally, server C sends to client A the message $(\|s_2\|s_2^*\|\Delta\|c_2 = (u_2, v_2))$ and outputs $\text{sk}_C = \text{sk}_j$.

Upon receiving $(s_2\|s_2^*\|\Delta\|c_2)$ from server C, client A checks whether c_2 is a valid ciphertext with respect to pk and $\text{label}_2 = C\|A\|s_1\|s_2\|s_2^*\|\Delta\|c_1$. If not, client A rejects and the protocol aborts. Otherwise, client A computes $\text{tk}' = H_{k_1}(u_2, \text{pw}_A) \oplus \text{Hash}(s_2, (u_1, \text{pw}_A), r_1)$, $H' = \text{ECC}^{-1}(\text{tk}' \oplus \Delta)$. Then it checks whether the Hamming distance between H' and $\text{Hash}(s_2^*, (u_1, \text{pw}_A), r_1)$ is less than $2\varepsilon/\ell$. If not, client A rejects and the protocol aborts. Otherwise, client A sets $r_i\|\tau_i\|\text{sk}_i \longleftarrow H'$ and outputs $\text{sk}_A = \text{sk}_i$.

3.4. Correctness. After honestly executing the protocol, participants can obtain different session keys with negligible probability. First, according to the smoothness of NA-ASPH, we can conclude that both $H_{k_1}(u_2, \text{pw}_A) \oplus \text{Hash}(s_2, (u_1,$

$\mathrm{pw}_A), r_1)$ and $\mathrm{Hash}(s_1, (u_2, \mathrm{pw}_A), r_j) \oplus H_{k_2}(u_1, \mathrm{pw}_A)$ have at most ϵ-fraction of nonzeros. Therefore, $\mathrm{tk} \oplus \mathrm{tk}'$ has at most 2ϵ-fraction of nonzeros. Then, we can obtain that $H' = \mathrm{ECC}^{-1}(\mathrm{tk}' \oplus \Delta) = \mathrm{ECC}^{-1}(\mathrm{tk}'\mathrm{oplus;tk} \oplus \mathrm{ECC}(H_{k_2^*}(u_1, \mathrm{pw}_A))) = H_{k_2^*}(u_1, \mathrm{pw}_A)$ as we assume that ECC can correct 2ϵ-fraction of errors. Second, we verify the validity of $H_{k_2^*}(u_1, \mathrm{pw}_A)$ by checking whether the Hamming distance between H' and $\mathrm{Hash}(s_2^*, (u_1, \mathrm{pw}_A), r_1)$ is less than $2\varepsilon/\ell$. If it is the case, it holds that $\mathrm{sk}_i = \mathrm{sk}_j$. This completes the correctness argument.

3.5. Security. We now show that the above two-round 2PAKE is secure through the proof of the following theorem.

Theorem 1. *If $\Sigma(Gen, Enc, Dec)$ is a splittable CCA-secure PKE scheme associated with an ϵ-NA-ASPH $(K, \ell, \mathbb{H} = \{H_k : X \longrightarrow \{0,1\}^\ell\}, S, \alpha : K \longrightarrow S)$ and $ECC : \{0,1\}^\kappa \longrightarrow \{0,1\}^\ell$ is an error-correcting code which can correct 2ϵ-fraction of errors, then the protocol in Table 2 is a secure PAKE protocol.*

Proof. Suppose $\mathscr{A}$ is a PPT attacker targeting this protocol. We estimate the advantage of adversary $\mathscr{A}$ through a series of experiments $\mathscr{T}_0, \mathscr{T}_1, \mathscr{T}_2, \cdots$, where $\mathscr{T}_0$ represents the experiment in the real protocol. By analyzing the difference of adversary's advantage between two adjacent experiments and defining the adversary's advantage in the final experiment, we can finally get the adversary's advantage in experiment $\mathscr{T}_0$, that is, the adversary's advantage when attacking the real protocol. In experiment $\mathscr{T}_i$, the event $\mathrm{Success}_i$ indicates that adversary $\mathscr{A}$ succeeds, and the adversary's advantage is defined as $\mathrm{Adv}_{\mathscr{A},i}(\kappa) = 2 \Pr[\mathrm{Success}_i] - 1$.

Experiment $\mathscr{T}_0$. This experiment corresponds to the security experiment of the real protocol. Attackers can send all queries according to the regulations of the secure model, and the instance being queried will respond according to the actual protocol specifications.

Experiment $\mathscr{T}_1$. We change the simulation method of Execute query. The only difference from the experiment $\mathscr{T}_0$ is that the calculation method of tk' is changed to $\mathrm{tk}' = H_{k_1}(u_2, \mathrm{pw}_A) \oplus H_{k_2}(u_1, \mathrm{pw}_A)$.

Lemma 1. *If $(K, \ell, \mathbb{H} = \{H_k : X \longrightarrow \{0,1\}^\ell\}, S, \alpha : K \longrightarrow S)$ is an ϵ-NA-ASPH, and $ECC : \{0,1\}^\kappa \longrightarrow \{0,1\}^\ell$ is an error-correcting code which can correct 2ϵ-fraction of errors, then $|Adv_{\mathscr{A},1}(\kappa) - Adv_{\mathscr{A},0}(\kappa)| \leq negl(\kappa)$.*

Proof. Since the simulator knows k_1 and k_2, it is easy to know that Lemma 1 holds according to the approximate correctness of NA-ASPH and the correctness of ECC.

Experiment $\mathscr{T}_2$. Compared with experiment $\mathscr{T}_1$, we modify the response to the Execute query as shown below. The ciphertext c_1 is replaced by the encryption of the illegal password $\mathrm{pw}_A' \notin \mathscr{D}$, and tk' calculated by the client A is forced to be equal to the tk calculated by the server C, and other calculations remain unchanged.

Lemma 2. *If $\Sigma(Gen, Enc, Dec)$ is a CCA-secure PKE scheme, then $|Adv_{\mathscr{A},2}(\kappa) - Adv_{\mathscr{A},1}(\kappa)| \leq negl(\kappa)$.*

Proof. We use standard hybrid argument to analyze the impact of replacing pw_A with pw_A' on the adversary's advantage. We set the number of queries to q_{exe} and define a series of intermediate experiments. The first η queries in the experiment are same as those in $\mathscr{T}_2$, the remaining $(q_{\mathrm{exe}} - \eta)$ queries are same as those in $\mathscr{T}_1$. The Send query conforms to the security model. It can be seen that experiments $\mathscr{T}_1^{(0)}$ and $\mathrm{F}_1^{(q_{\mathrm{exe}})}$ are completely consistent with the experiments $\mathscr{T}_1$ and $\mathscr{T}_2$, respectively. If the lemma is not true, that is, the difference of $\mathscr{A}$'s advantage between experiments $\mathscr{T}_1$ and $\mathscr{T}_2$ is not negligible, there must be some η such that the difference of $\mathscr{A}$'s advantage between $\mathscr{T}_1^{(\eta-1)}$ and $\mathscr{T}_1^{(\eta)}$ cannot be ignored. Then, we can construct an attacker $\mathscr{M}$ for the security of the encryption system Σ so that it can successfully attack Σ with nonnegligible probability.

We now construct an adversary $\mathscr{M}$ who attacks the CCA-secure PKE scheme Σ in the following way: given the public key pk, the adversary $\mathscr{M}$ simulates the entire experiment for $\mathscr{A}$ according to the experiment $\mathscr{T}_1^{(\eta)}$, including selecting random passwords for the participants and selecting the random bit b for $\mathscr{A}$ in the Test query. When answering the Execute query, $\mathscr{M}$ sends $(\mathrm{pw}_A, \mathrm{pw}_A')$ as its challenge plaintext pair to $\mathscr{M}$'s own challenger. After receiving the challenge ciphertext c_1', $\mathscr{M}$ replaces c_1 with c_1' in the Execute query. Finally, $\mathscr{M}$ checks whether $\mathscr{A}$ guesses the random bit in the Test query. If $\mathscr{A}$ succeeds, $\mathscr{M}$ outputs 1; otherwise, $\mathscr{M}$ outputs 0.

Let $\mathrm{Event}_\Sigma^{\mathrm{pw}_A}(\mathscr{M})$ denote that $\mathscr{M}$ obtains the challenge ciphertext of the real password pw_A and outputs 1 at the end of the experiment. Let $\mathrm{Event}_\Sigma^{\mathrm{pw}_A'}(\mathscr{M})$ indicate that $\mathscr{M}$ obtains the challenge ciphertext of the invalid password pw_A' and outputs 1 at the end of the experiment. For the η^{th} query, that is, $\mathscr{M}$ gets the challenge ciphertext of the real password pw_A, the environment provided by $\mathscr{M}$ for the protocol attacker $\mathscr{A}$ is the same as experiment $\mathscr{T}_1^{(\eta-1)}$. Therefore, in experiment $\mathscr{T}_1^{(\eta-1)}$, the probability that $\mathscr{M}$ outputs 1 is exactly the same as $\mathscr{A}$'s success probability ($\Pr[\mathrm{Success}_{\mathscr{T}_1^{(\eta-1)}}]$), i.e., $\Pr[\mathrm{Event}_\Sigma^{\mathrm{pw}_A}(\mathscr{M}) = 1] = \Pr[\mathrm{Success}_{\mathscr{T}_1^{(\eta-1)}}]$. Similarly, when $\mathscr{M}$ gets the challenge ciphertext of the invalid password pw_A', the probability that $\mathscr{M}$ outputs 1 is the probability that $\mathscr{A}$ succeeds ($\Pr[\mathrm{Success}_{\mathscr{T}_1^{(\eta)}}]$) in attacking the protocol in experiment $\mathscr{T}_1^{(\eta)}$, namely $\Pr[\mathrm{Event}_\Sigma^{\mathrm{pw}_A'}(\mathscr{M}) = 1] = \Pr[\mathrm{Success}_{\mathscr{T}_1^{(\eta)}}]$. Let $\mathrm{Adv}_{\Sigma,\mathscr{M}}^{IND-CCA}(\kappa)$ be $\mathscr{M}$'s advantage in attacking the encryption system Σ, then

$$\begin{aligned}&\left|\mathrm{Adv}_{\mathscr{T}_1^{(\eta)}}(\kappa)-\mathrm{Adv}_{\mathscr{T}_1^{(\eta-1)}}(\kappa)\right|\\&=2\left|\Pr\left[\mathrm{Success}_{\mathscr{T}_1^{(\eta)}}\right]-\Pr\left[\mathrm{Success}_{\mathscr{T}_1^{(\eta-1)}}\right]\right|\\&=2\left|\Pr\left[\mathrm{Event}_{\Sigma}^{\mathrm{pw}_A}(\mathscr{M})=1\right]-\Pr\left[\mathrm{Event}_{\Sigma}^{\mathrm{pw}_A}(\mathscr{M})=1\right]\right|\\&=2\mathrm{Adv}_{\Sigma,\mathscr{M}}^{\mathrm{IND-CCA}}(\kappa).\end{aligned}\tag{4}$$

According to the CCA security of encryption system Σ, the lemma holds. We emphasize that only the CPA security of Σ is actually used here.

Experiment $\mathscr{T}_3$. We change the response to the Execute query: (1) change the calculation method of tk to $\mathrm{tk}=H_{k_1}(u_2,\mathrm{pw}_A)\oplus H_{k_2}(u_1,\mathrm{pw}_A)$. (2) Replace the ciphertext c_2 with the encryption of an illegal password $\mathrm{pw}_A'\notin\mathscr{D}$.

Lemma 3. *If $\Sigma(Gen, Enc, Dec)$ is a splittable CCA-secure PKE scheme associated with and ϵ-NA-ASPH, and ECC : $\{0,1\}^{\kappa}\longrightarrow\{0,1\}^{\ell}$ is an error-correcting code which can correct 2ϵ-fraction of errors, then $|Adv_{\mathscr{A},3}(\kappa)-Adv_{\mathscr{A},2}(\kappa)|\le negl(\kappa)$.*

Proof. This lemma is shown through a series of experiments similar to $\mathscr{T}_1$, $\mathscr{T}_2$, and $\mathscr{T}_3$. In addition, this experiment utilizes the modified CCA security experiment shown in Section 2.3 instead of the standard CCA security experiment.

Experiment $\mathscr{T}_4$. We continue to modify the response to the Execute query as follows. We set $r_j\|\tau_j\|\mathrm{sk}_j$ to a random string of the appropriate length and the $r_i\|\tau_i\|\mathrm{sk}_i$ calculated by client A to be equal to the $r_j\|\tau_j\|\mathrm{sk}_j$ calculated by server C.

Lemma 4. *If $(K,\ell,\mathbb{H}=\{H_k:X\longrightarrow\{0,1\}^{\ell}\},S,\alpha:K\longrightarrow S)$ is an ε-NA-ASPH, then $|Adv_{\mathscr{A},4}(\kappa)-Adv_{\mathscr{A},3}(\kappa)|\le negl(\kappa)$.*

Proof. This comes from the smoothness of NA-ASPH, because when responding to an Execute query in $\mathscr{T}_3$, the hash function H_k is always applied to $\mathrm{pw}_A'\notin\mathscr{D}_n$, so even if s is given, the output is statistically close to uniform. In addition, in $\mathscr{T}_3$ and $\mathscr{T}_4$ the string $r_i\|\tau_i\|\mathrm{sk}_i$ used by the client is equal to the string $r_j\|\tau_j\|\mathrm{sk}_j$ computed by the server.

Note that the Execute query in $\mathscr{T}_4$ will generate a random session key and random transcripts, which have nothing to do with the actual password of any participant. In the following experiment, we begin to modify the responses to the Send queries. Let $\mathrm{Send}_0(A,i,C)$ represent the "start" message, which enables the client instance Π_A^i to initiate authentication with the server S. Note that when calculating the number of communication rounds, we ignore the "start" message like other related research. Let $\mathrm{Send}_1(C,j,\mathrm{msg1}=(s_1\|c_1=(u_1,v_1)))$ represent the first message of the protocol sent to the server instance Π_c^j. Let $\mathrm{Send}_2(A,i,\mathrm{msg2}=(s_2\|s_2^*\|\Delta\|c_2=(u_1,v_1)))$ denote the second message of the protocol sent to the client instance Π_A^i. We also record the secret key $\mathrm{sk}_{\mathrm{pk}}$, corresponding to the public key in the generated CRS.

Now, we make some explanations for msg1 and msg2. The output of send_0 oracle or the input of send_1 oracle are msg1. Similarly, msg2 may be the output of send_1 oracle or the output of send_2 oracle. If msg1/msg2 is output by a previous $\mathrm{send}_0/\mathrm{send}_1$ oracle, then we call msg1/msg2 oracle-generated.

Experiment $\mathscr{T}_5$. In experiment $\mathscr{T}_5$, we change the response to send_1 queries. If msg1 is oracle-generated, the experiment is the same as $\mathscr{T}_4$. Otherwise, we set $\mathrm{label}_1=A\|C\|s_1$. We check the validity of c_1 according to label_1 and pk.

(i) If c_1 is invalid, the experiment just aborts as the real protocol

(ii) Else, we can get pw_A^{ad} by decrypting c_1, because we have $\mathrm{sk}_{\mathrm{pk}}$. If $\mathrm{pw}_A^{ad}=\mathrm{pw}_A$, we just declare that adversary $\mathscr{A}$ succeeds, and the experiment is terminated. If $\mathrm{pw}_A^{ad}\neq\mathrm{pw}_A$, we set tk and $H_{k_2^*}(u_1,\mathrm{pw}_A)$ computed by the server as random tapes of the appropriate length

Lemma 5. *If $(K,\ell,\mathbb{H}=\{H_k:X\longrightarrow\{0,1\}^{\ell}\},S,\alpha:K\longrightarrow S)$ is an ϵ-NA-ASPH, then $Adv_{\mathscr{A},4}(\kappa)\le Adv_{\mathscr{A},5}(\kappa)+negl(\kappa)$.*

Proof. In the actual protocol, server C simply refuses, and the protocol terminates when c_1 is invalid. Therefore, if msg1 is oracle-generated, or msg1 is not oracle-generated and c_1 is invalid, then experiment $\mathscr{T}_5$ is consistent with experiment $\mathscr{T}_4$. Now, we only need to consider the case where msg1 is not oracle-generated and c_1 is valid.

(i) If $\mathrm{pw}_A^{ad}=\mathrm{pw}_A$, adversary $\mathscr{A}$ succeeds. Note that this only improves the adversary's advantage

(ii) If $\mathrm{pw}_A^{ad}\neq\mathrm{pw}_A$, tk and $H_{k_2^*}(u_1,\mathrm{pw}_A)$ computed by the server are both set to random tapes. From the view of adversary $\mathscr{A}$, there is no difference between these changes. First, as $(c_1,\mathrm{pw}_A^{ad})\notin\bar{L}$, in the view of $\mathscr{A}$, both $\mathrm{tk}=\mathrm{Hash}(s_1,(u_2,\mathrm{pw}_A),r_j)\oplus H_{k_2}$

$(u_1,\mathrm{pw}_A)$and $\Delta=\mathrm{tk}\oplus\mathrm{ECC}(H_{k_2^*}(u_1,\mathrm{pw}_A^{ad}))$ are statistically indistinguishable from random uniform distribution. This can be derived directly from the smoothness of NA-ASPH. Similarly, from the view of $\mathscr{A}$, $r_j\|\tau_j\|\mathrm{sk}_j=H_{k_2^*}(u_1,\mathrm{pw}_A^{ad})$ is also statistically indistinguishable from random uniform distribution. Therefore, $\mathrm{pw}_A^{ad}\neq\mathrm{pw}_A$ only introduces a negligible difference in experiment $\mathscr{T}_5$.

Finally, we obtain that $\mathrm{Adv}_{\mathscr{A},4}(\kappa)\le\mathrm{Adv}_{\mathscr{A},5}(\kappa)+\mathrm{negl}(\kappa)$.

Experiment $\mathscr{T}_6$. In experiment $\mathscr{T}_6$, let msg1 be the output from a previous Send_0 query (A,i,C) (note that such a query must exist). Send_2 query is handled as follows: If msg2 is oracle-generated by a previous Send_1 query, the experiment is similar to $\mathscr{T}_5$ except for (1) computing tk' as $\mathrm{tk}'=H_{k_1}(u_2,\mathrm{pw}_A)\oplus H_{k_2}(u_1,\mathrm{pw}_A)$ and (2) setting $r_i\|\tau_i\|sk_i$

$= r_j \| \tau_j \| \mathrm{sk}_j$. Otherwise, we set $\mathrm{label}_2 = C \| A \| s_1 \| s_2 \| s_2^* \| \Delta \| c_1$. We check the validity of c_2 according to label_2 and pk.

(i) If c_2 is invalid, the experiment just aborts as the real protocol

(ii) Else, we can obtain pw_A^{ad} similar to $\mathscr{T}_5$. If $\mathrm{pw}_A^{ad} = \mathrm{pw}_A$, we declare that $\mathscr{A}$ succeeds and the experiment terminates. Otherwise, if Π_A^i is accepted, let $r_i \| \tau_i \| sk_i$ to be a random tape of appropriate length

Lemma 6. *If* $(K, \ell, \mathbb{H} = \{H_k : X \longrightarrow \{0,1\}^\ell\}, S, \alpha : K \longrightarrow S)$ *is an* ϵ*-NA-ASPH, and ECC:*$\{0,1\}^\kappa \longrightarrow \{0,1\}^\ell$ *is an error-correcting code which can correct* 2ϵ*-fraction of errors, then* $Adv_{\mathscr{A},5}(\kappa) \le Adv_{\mathscr{A},6}(\kappa) + negl(\kappa)$.

Proof. We prove different situations separately. First, if both msg1 and msg2 are oracle-generated, the simulator will know the hash keys k_1 and k_2. According to the smoothness of the NA-ASPH, the changes in computing tk' and$r_i \| \tau_i \| sk_i$ are just conceptual (in this case, it holds that $r_i \| \tau_i \| sk_i = r_j \| \tau_j \| \mathrm{sk}_j$ in both $\mathscr{T}_5$ and $\mathscr{T}_6$). Second, if msg2 is not oracle-generated, the simulator sets $\mathrm{label}_2 = C \| A \| s_1 \| s_2 \| s_2^* \| \Delta \| c_1$ and then uses label_2 and pk to check whether c_2 is valid. If not, $\mathscr{T}_5$ and $\mathscr{T}_6$ are the same as the real protocol. Otherwise, the simulator uses $\mathrm{sk}_{\mathrm{pk}}$ to decrypt c_2 and obtains pw_A^{ad}.

(i) If $\mathrm{pw}_A^{ad} = \mathrm{pw}_A$, adversary $\mathscr{A}$ succeeds. Note that this just improves the adversary advantage

(ii) If $\mathrm{pw}_A^{ad} \neq \mathrm{pw}_A$, according to section 2.4, $(\mathrm{label}_2, c_2, \mathrm{pw}_A^{ad})$ does not belong to $\bar{L}$. Then, by the smoothness of NA-ASPH, $H_{k_1}(u_2, \mathrm{pw}_A)$ and thus $\mathrm{tk}' = H_{k_1}(u_2, \mathrm{pw}_A) \oplus H_{k_2}(u_1, \mathrm{pw}_A)$ are both statistically close to uniform over $\{0,1\}^\ell$. Furthermore, we have $r_i \| \tau_i \| sk_i$ $(r_i \| \tau_i \| sk_i \longleftarrow H' = \mathrm{ECC}^{-1}(\mathrm{tk}' \oplus \Delta))$ is statistically close to uniform over $\{0,1\}^\kappa$. Therefore, the modifications of $\mathrm{pw}_A^{ad} \neq \mathrm{pw}_A$ bring a negligible statistical difference. Note that the output of $\mathrm{ECC}^{-1}(\mathrm{tk}' \oplus \Delta)$ may be $\perp$. In this case, client A rejects

Experiment $\mathscr{T}_7$. Compared with experiment $\mathscr{T}_6$, we modify the response to a Send_0 query. The only difference is that we use $\mathrm{pw}_A' \notin \mathscr{D}$ to compute c_1.

Lemma 7. *If* $\Sigma(Gen, Enc, Dec)$ *is a CCA-secure PKE scheme, then* $|Adv_{\mathscr{A},7}(\kappa) - Adv_{\mathscr{A},6}(\kappa)| \le negl(\kappa)$.

Proof. We analyze the impact of replacing pw_A with $\mathrm{pw}_A' \notin \mathscr{D}$ on the adversary's advantage similar to $\mathscr{T}_2$. But for the sake of simplicity, we consider that $\mathscr{A}$ only executes a single Send_0 query. The correctness still holds according to standard hybrid argument. If the lemma is not true, that is, the difference of $\mathscr{A}$'s advantage between experiments $\mathscr{T}_6$ and $\mathscr{T}_7$ is not negligible, then an attacker $\mathscr{M}$ can be constructed for the security experiment of the encryption system Σ, which can successfully attack Σ with nonnegligible probability.

We now construct an adversary $\mathscr{M}$ who attacks the CCA-secure PKE scheme Σ in the following way: given the public key pk, the adversary $\mathscr{M}$ simulates the entire experiment for $\mathscr{A}$ according to experiment $\mathscr{T}_7$, including selecting random passwords for the participants and selecting the random bit b for $\mathscr{A}$ in the Test query. When answering the Send_0 query, $\mathscr{M}$ will send $(\mathrm{pw}_A, \mathrm{pw}_A')$ as its challenge plaintext pair to $\mathscr{M}$'s own challenger. After receiving the challenge ciphertext c_1', $\mathscr{M}$ replaces c_1 with c_1' in the Send_0 query. Finally, $\mathscr{M}$ checks whether $\mathscr{A}$ guesses the random bit in the Test query. If $\mathscr{A}$ succeeds, $\mathscr{M}$ outputs 1; otherwise, $\mathscr{M}$ outputs 0.

Let $\mathrm{Event}_\Sigma^{\mathrm{pw}_A}(\mathscr{M})$ denote that $\mathscr{M}$ gets the challenge ciphertext of the real password pw_A and outputs 1 at the end of the experiment. Let $\mathrm{Event}_\Sigma^{\mathrm{pw}_A'}(\mathscr{M})$ represent that $\mathscr{M}$ gets the challenge ciphertext of the invalid password pw_A' and outputs 1 at the end of the experiment. If $\mathscr{M}$ gets the challenge ciphertext of the real password pw_A, the environment provided by $\mathscr{M}$ for the protocol adversary $\mathscr{A}$ is the same as experiment $\mathscr{T}_6$. Therefore, the probability that $\mathscr{M}$ outputs 1 is exactly the same as $\mathscr{A}$'s success probability $(\Pr[\mathrm{Success}_{\mathscr{T}_6}])$ in experiment $\mathscr{T}_6$, i.e., $\Pr[\mathrm{Event}_\Sigma^{\mathrm{pw}_A}(\mathscr{M}) = 1] = \Pr[\mathrm{Success}_{\mathscr{T}_6}]$. Similarly, $\Pr[\mathrm{Event}_\Sigma^{\mathrm{pw}_A'}(\mathscr{M}) = 1] = \Pr[\mathrm{Success}_{\mathscr{T}_7}]$. Let $\mathrm{Adv}_{\Sigma,\mathscr{M}}^{\mathrm{IND-CCA}}(\kappa)$ be $\mathscr{M}$'s advantage in attacking the encryption system Σ, then

$$\begin{aligned}\left|\mathrm{Adv}_{\mathscr{T}_1^{(\eta)}}(\kappa) - \mathrm{Adv}_{\mathscr{T}_1^{(\eta-1)}}(\kappa)\right| &= 2\left|\Pr\left[\mathrm{Success}_{\mathscr{T}_1^{(\eta)}}\right] - \Pr\left[\mathrm{Success}_{\mathscr{T}_1^{(\eta-1)}}\right]\right| \\ &= 2\Big|\Pr\left[\mathrm{Event}_\Sigma^{\mathrm{pw}_A'}(\mathscr{M}) = 1\right] \\ &\quad - \Pr\left[\mathrm{Event}_\Sigma^{\mathrm{pw}_A}(\mathscr{M}) = 1\right]\Big| \\ &= 2\mathrm{Adv}_{\Sigma,\mathscr{M}}^{\mathrm{IND-CCA}}(\kappa).\end{aligned} \tag{4}$$

According to the CCA security of the encryption system Σ, the lemma holds.

Experiment $\mathscr{T}_8$. Experiment $\mathscr{T}_8$ is similar to $\mathscr{T}_7$ except that if msg1 is oracle-generated: (1) tk computed by the server is set to be $H_{k_1}(u_2, \mathrm{pw}_A) \oplus H_{k_2}(u_1, \mathrm{pw}_A)$; (2) a random string $r_j \| \tau_j \| \mathrm{sk}_j$ of appropriate length is set for Π_C^j.

Lemma 8. *If* $\Sigma(Gen, Enc, Dec)$ *is a splittable CCA-secure PKE scheme associated with an* ϵ*-NA-ASPH* $(K, \ell, \mathbb{H} = \{H_k : X \longrightarrow \{0,1\}^\ell\}, S, \alpha : K \longrightarrow S)$*, and ECC* $: \{0,1\}^\kappa \longrightarrow \{0,1\}^\ell$ *is an error-correcting code which can correct* 2ϵ*-fraction of errors, then* $|Adv_{\mathscr{A},8}(\kappa) - Adv_{\mathscr{A},7}(\kappa)| \le negl(\kappa)$.

Proof. First, if msg1 is oracle-generated, the simulator has hash keys k_1 and k_2; thus, it can compute $\mathrm{tk} = H_{k_1}(u_2, \mathrm{pw}_A) \oplus H_{k_2}(u_1, \mathrm{pw}_A)$.Secondly, since the ciphertext c_1 is the encryption of $\mathrm{pw}_A' \notin \mathscr{D}$, $H_{k_2}(u_1, \mathrm{pw}_A)$ and tk are statistically close to uniform. Similarly, $r_j \| \tau_j \| sk_j$ $(r_j \| \tau_j \| sk_j \longleftarrow H_{k_2^*}(u_1, \mathrm{pw}_A))$ is also statistically close to uniform. Thus, we have that

the modifications here introduce only a statistically negligible difference. Therefore, $|\mathrm{Adv}_{\mathscr{A},8}(\kappa) - \mathrm{Adv}_{\mathscr{A},7}(\kappa)| \leq \mathrm{negl}(\kappa)$.

Experiment $\mathscr{T}_9$. For the final experiment, we again modify the response to the Send_1 queries. If msg1 is oracle-generated, the ciphertext c_2 is now computed as the encryption of $\mathrm{pw}'_A \notin \mathscr{D}$.

Lemma 9. *If $\Sigma(Gen, Enc, Dec)$ is a splittable CCA-secure PKE scheme, then $|Adv_{\mathscr{A},9}(\kappa) - Adv_{\mathscr{A},8}(\kappa)| \leq negl(\kappa)$.*

Proof. We analyze the impact of replacing pw_A with $\mathrm{pw}'_A \notin \mathscr{D}$ on the adversary's advantage. We consider that $\mathscr{A}$ only executes a single Send_1 query similar to $\mathscr{T}_7$. Now, we show that if any PPT adversary $\mathscr{A}$ can distinguish these two experiments, then we can construct an attacker $\mathscr{M}$ breaking the CCA security experiment (in Section 2.3) of the CCA encryption system Σ with a nonnegligible probability.

We now construct an adversary $\mathscr{M}$ interacting with $\mathscr{A}$ in $\mathscr{T}_8$ to attack the CCA-secure PKE scheme Σ in the following way: given the public key pk, the adversary $\mathscr{M}$ simulates the entire experiment, including selecting random passwords for the participants and selecting the random bit b for $\mathscr{A}$ in the Test query. When answering the Send_1 query, $\mathscr{M}$ will send $(\mathrm{pw}_A, \mathrm{pw}'_A)$ as its challenge plaintext pair to $\mathscr{M}$'s own challenger. After receiving the challenge ciphertext c'_1, $\mathscr{M}$ replaces c_1 with c'_1 in the Send_1 query. When $\mathscr{M}$ needs to decrypt some valid ciphertext c'_2, it will send $(\mathrm{label}'_2, c'_2)$ to its own challenger to obtain the corresponding pw'_A. Finally, $\mathscr{M}$ checks whether $\mathscr{A}$ guesses the random bit correctly in the Test query. If $\mathscr{A}$ succeeds, $\mathscr{M}$ outputs 1; otherwise, $\mathscr{M}$ outputs 0.

Let $\mathrm{Event}_\Sigma^{\mathrm{pw}_A}(\mathscr{M})/\mathrm{Event}_\Sigma^{\mathrm{pw}'_A}(\mathscr{M})$ denote that $\mathscr{M}$ gets the challenge ciphertext of the password $\mathrm{pw}_A/\mathrm{pw}'_A$ and outputs 1. If $\mathscr{M}$ gets the challenge ciphertext of $\mathrm{pw}_A/\mathrm{pw}'_A$, the environment provided by $\mathscr{M}$ for the protocol adversary $\mathscr{A}$ is the same as experiment $\mathscr{T}_8/\mathscr{T}_9$. Therefore, we have $\Pr[\mathrm{Event}_\Sigma^{\mathrm{pw}_A}(\mathscr{M}) = 1] = \Pr[\mathrm{Success}_{\mathscr{T}_8}]$ and $\Pr[\mathrm{Event}_\Sigma^{\mathrm{pw}'_A}(\mathscr{M}) = 1] = \Pr[\mathrm{Success}_{\mathscr{T}_9}]$. Let $\mathrm{Adv}_{\Sigma,\mathscr{M}}^{\mathrm{IND-CCA}}(\kappa)$ be $\mathscr{M}$'s advantage in attacking the encryption system Σ, then

$$\begin{aligned}
&\left|\mathrm{Adv}_{\mathscr{T}_9}(\kappa) - \mathrm{Adv}_{\mathscr{T}_8}(\kappa)\right| = 2\left|\Pr\left[\mathrm{Success}_{\mathscr{T}_9}\right] - \Pr\left[\mathrm{Success}_{\mathscr{T}_8}\right]\right| \\
&= 2\left|\Pr\left[\mathrm{Event}_\Sigma^{\mathrm{pw}'_A}(\mathscr{M}) = 1\right] - \Pr\left[\mathrm{Event}_\Sigma^{\mathrm{pw}_A}(\mathscr{M}) = 1\right]\right| \\
&= 2\mathrm{Adv}_{\Sigma,\mathscr{M}}^{\mathrm{IND-CCA}}(\kappa).
\end{aligned} \tag{6}$$

According to the CCA security of encryption system Σ, the lemma holds.

So far, we have completed the modification of Send query. We now analyze the adversary's advantage in the final experiment $\mathscr{T}_9$. If adversary $\mathscr{A}$ cannot guess the correct password, $\mathscr{A}$ can only rely on guessing the random bit b in the Test query to succeed. Note that all session keys are replaced with random tapes, so the probability of $\mathscr{A}$ guessing b is only 1/2. At the same time, as described in experiments $\mathscr{T}_5$ and $\mathscr{T}_6$, the adversary $\mathscr{A}$ can succeed by guessing the password, and the probability of each correct guess is at most $1/|\mathscr{D}|$, so the ultimate advantage of adversary $\mathscr{A}$ is at most $q_{\mathrm{send}}/|\mathscr{D}|$. Combining the conclusions of Lemma 1 to Lemma 9, we can see that $\mathrm{Adv}_{\mathscr{T}_0}(n) \leq q_{\mathrm{send}}/|\mathscr{D}| + \mathrm{negl}(\kappa)$, that is, the conclusion of Theorem 1 is established.

4. A Two-Round 3PAKE Protocol

In this section, we propose a two-round 3PAKE protocol based on the two-round 2PAKE protocol in Section 3. Client A and server C share the password pw_A, and client B and server C share the password pw_B. The primitives and the initialization process here are the same as the two-round 2PAKE protocol above. The clients and server implement the honest 3PAKE protocol on lattice, as shown in Table 3.

4.1. Protocol Execution. Clients A and B, respectively, choose random tapes $r_{1A} \longleftarrow_r \{0,1\}^*$ and $r_{1B} \longleftarrow_r \{0,1\}^*$ for encryption hash keys $k_{1A}, k_{1B} \longleftarrow_r K$. Then, A/B computes the projection key $s_{1A} = \alpha(k_{1A})/s_{1B} = \alpha(k_{1B})$ and sets $\mathrm{label}_{1A} = A\|B\|S\|s_{1A}/\mathrm{label}_{1B}B\|A\|S\|s_{1B}$. A/B continues to compute $c_{1A} = (u_{1A}, v_{1A}) = \Sigma(\mathrm{pk}, \mathrm{label}_{1A}, \mathrm{pw}_A, r_{1A})/c_{1B} = (u_{1B}, v_{1B}) = \Sigma(pk, \mathrm{label}_{1B}, \mathrm{pw}_B, r_{1B})$. Finally, client A/B sends to server C a message $(A\|B\|S\|s_{1A}\|c_{1A} = (u_{1A}, v_{1A}))/(B\|A\|S\|s_{1B}\|c_{B1} = (u_{B1}, v_{1B}))$.

After receiving $(A\|B\|S\|s_{1A}\|c_{1A})$ and $(B\|A\|S\|s_{1B}\|c_{B1})$ from clients A and B, the server C checks whether c_{1A} and c_{1B} are valid ciphertexts with respect to pk, label_{1A} and label_{1B}. If not, C refuses and the protocol is terminated. Otherwise, C chooses hash keys $k_{2A}, k_{2B} \longleftarrow_r K$ and $k^*_{2A}, k^*_{2B} \longleftarrow_r K$. It computes $s_{2A} = \alpha(k_{2A})$, $s_{2B} = \alpha(k_{2B})$, $s^*_{2A} = \alpha(k^*_{2A})$, $s^*_{2B} = \alpha(k^*_{2B})$, $r_{jA}\|\tau_{jA}\|\mathrm{sk}_{jA} \longleftarrow H_{k^*_{2A}}(u_{1A}, \mathrm{pw}_A)$, $m_A = \tau_{jA} \oplus \mathrm{sk}_{jB}$, $m_B = \tau_{jB} \oplus \mathrm{sk}_{jA}$, $r_{jB}\|\tau_{jB}\|\mathrm{sk}_{jB} \longleftarrow H_{k^*_{2B}}(u_{1B}, \mathrm{pw}_B)$, $u_{2A} = f(pk, \mathrm{pw}_A, r_{jA})$, $u_{2B} = f(pk, \mathrm{pw}_B, r_{jB})$, $\mathrm{tk}_A = \mathrm{Hash}(s_{1A}, (u_{2A}, \mathrm{pw}_A)\, r_{jA}) \oplus H_{k_{2A}}(u_{1A}, \mathrm{pw}_A)$, $\mathrm{tk}_B = \mathrm{Hash}(s_{1B}, (u_{2B}, \mathrm{pw}_B), r_{jB}) \oplus H_{k_{2B}}(u_{1B}, \mathrm{pw}_B)$, $\Delta_A = \mathrm{tk}_A \oplus \mathrm{ECC}(H_{k^*_{2A}}(u_{1A}, \mathrm{pw}_A))$, and $\Delta_B = \mathrm{tk}_B \oplus \mathrm{ECC}(H_{k^*_{2B}}(u_{1B}, \mathrm{pw}_B))$. Then, C sets $\mathrm{label}_{2A} = A\|B\|C\|s_{1A}\|s_{2A}\|s^*_{2A}\|\Delta_A\|c_{1A}$ and $\mathrm{label}_{2B} = B\|A\|C\|s_{1B}\|s_{2B}\|s^*_{2B}\|\Delta_B\|c_{1B}$ and computes $v_{2A} = \mathrm{g}(\mathrm{pk}, \mathrm{label}_{2A}, \mathrm{pw}_A, r_j\mathrm{A})$, $v_{2B} = g(\mathrm{pk}, \mathrm{label}_{2B}, \mathrm{pw}_B, r_{jB})$. Finally, C sends to A/B the message $(s_{2A}\|s^*_{2A}\|\Delta_A\|c_{2A} = (u_{2A}, v_{2A})\|m_A)/(s_{2B}\|s^*_{2B}\|\Delta_B\|c_{2B} = (u_{2B}, v_{2B})\|m_B)$ and outputs $\mathrm{sk}_{AB} = \mathrm{sk}_{jA} \oplus \mathrm{sk}_{jB}$.

Received from server C $(s_{2A}\|s^*_{2A}\|\Delta_A\|c_{2A} = (u_{2A}, v_{2A})\|m_A)/(s_{2B}\|s^*_{2B}\|\Delta_B\|c_{2B} = (u_{2B}, v_{2B})\|m_B)$, client A/B checks whether c_{2A}/c_{2B} is a valid ciphertext with respect to pk and $\mathrm{label}_{2A}/\mathrm{label}_{2B}$. If not, A/B rejects and the protocol aborts. Otherwise, A/B computes $\mathrm{tk}'_A = H_{k_{1A}}(u_{2A}, \mathrm{pw}_A) \oplus \mathrm{Hash}(s_{2A}, (u_{1A}, \mathrm{pw}_A), r_{1A})/\mathrm{tk}'_B = H_{k_{1B}}(u_{2B}, \mathrm{pw}_B) \oplus \mathrm{Hash}(s_{2B}, (u_{1B}, \mathrm{pw}_B), r_{1B})$, $H'_A = \mathrm{ECC}^{-1}(\mathrm{tk}'_A \oplus \Delta_A)/H'_B = \mathrm{ECC}^{-1}(\mathrm{tk}'_B \oplus \Delta_B)$. Client A/B then checks whether the Hamming distance between H'_A/H'_B and $\mathrm{Hash}(s^*_{2A}, (u_{1A}, \mathrm{pw}_A), r_{1A})/\mathrm{Hash}(s^*_{2B}, (u_{1B}, \mathrm{pw}_B), r_{1B})$ is less than $2\varepsilon/\ell$. If not, Client A/B rejects and the protocol aborts. Otherwise, A/B sets $r_{iA}\|\tau_{iA}\|\mathrm{sk}_{iA}$

Table 3: An honest execution of two-round 3PAKE protocol.

Two-round 3PAKE protocol		
Client A	Server	Client B
$r_{1A} \longleftarrow_r \{0,1\}^*$		$r_{1B} \longleftarrow_r \{0,1\}^*$
$k_{1A} \longleftarrow_r K$		$k_{1B} \longleftarrow_r K$
$s_{1A} = \alpha(k_{1A})$		$s_{1B} = \alpha(k_{1B})$
$\text{label}_{1A} = A\|\|B\|\|S\|\|s_{1A}$		$\text{label}_{1B} = B\|\|A\|\|S\|\|s_{1B}$
$u_{1A} = f(\text{pk}, \text{pw}_A, r_{1A})$		$u_{1B} = f(\text{pk}, \text{pw}_B, r_{1B})$
$v_{1A} = \text{g}(\text{pk}, \text{label}_{1A}, \text{pw}_A, r_{1A})$		$v_{1B} = \text{g}(\text{pk}, \text{label}_{1B}, \text{pw}_B, r_{1B})$
$A\|\|B\|\|S\|\|s_{1A}\|\|c_{1A} = (u_{1A}, v_{1A})$ $\longrightarrow$		$\longleftarrow$ $B\|\|A\|\|S\|\|s_{1B}\|\|c_{1B} = (u_{1B}, v_{1B})$
	$k_{2A} \longleftarrow_r K, k_{2A}^* \longleftarrow_r K$	$k_{2B} \longleftarrow_r K, k_{2B}^* \longleftarrow_r K$
	$s_{2A} = \alpha(k_{2A}), s_{2A}^* = \alpha(k_{2A}^*)$	$s_{2B} = \alpha(k_{2B}), s_{2B}^* = \alpha(k_{2B}^*)$
	$r_{jA}\|\|r_{jA}\|\|\text{sk}_{jA} \longleftarrow H_{k_{2A}^*}(u_{1A}, \text{pw}_A)$	$r_{jB}\|\|\tau_{jB}\|\|\text{sk}_{jB} \longleftarrow H_{k_{2B}^*}(u_{1B}, \text{pw}_B)$
	$m_A = \tau_{jA} \oplus \text{sk}_{jB}$	$m_B = \tau_{jB} \oplus \text{sk}_{jA}$
	$u_{2A} = f(pk, \text{pw}_A, r_{jA})$	$u_{2B} = f(pk, \text{pw}_B, r_{jB})$
	$\text{tk}_A = \text{Hash}(s_{1A}, (u_{2A}, \text{pw}_A), r_{jA}) \oplus H_{k_{2A}}(u_{1A}, \text{pw}_A)$	$\text{tk}_B = \text{Hash}(s_{1B}, (u_{2B}, \text{pw}_B), r_{jB}) \oplus H_{k_{2B}}(u_{1B}, \text{pw}_B)$
	$\Delta_A = \text{tk}_A \oplus \text{ECC}(H_{k_{2A}^*}(u_{1A}, \text{pw}_A))$	$\Delta_B = \text{tk}_B \oplus \text{ECC}(H_{k_{2B}^*}(u_{1B}, \text{pw}_B))$
	$\text{label}_{2A} = A\|\|B\|\|C\|\|s_{1A}\|\|s_{2A}\|\|s_{2A}^*\|\|\Delta_A\|\|c_{1A}$	$\text{label}_{2B} = B\|\|A\|\|C\|\|s_{1B}\|\|s_{2B}\|\|s_{2B}^*\|\|\Delta_B\|\|c_{1B}$
	$v_{2A} = \text{g}(\text{pk}, \text{label}_{2A}, \text{pw}_A, r_j\text{A})$	$v_{2B} = \text{g}(\text{pk}, \text{label}_{2B}, \text{pw}_B, r_{jB})$
	$\longleftarrow$ $s_{2A}\|\|s_{2A}^*\|\|\Delta_A\|\|c_{2A} = (u_{2A}, v_{2A})\|\|m_A$	$s_{2B}\|\|s_{2B}^*\|\|\Delta_B\|\|c_{2B} = (u_{2B}, v_{2B})\|\|m_B$ $\longrightarrow$
$\text{tk}_A' = H_{k_{1A}}(u_{2A}, \text{pw}_A) \oplus \text{Hash}(s_{2A}, (u_{1A}, \text{pw}_A), r_{1A})$		$\text{tk}_B' = H_{k_{1B}}(u_{2B}, \text{pw}_B) \oplus \text{Hash}(s_{2B}, (u_{1B}, \text{pw}_B), r_{1B})$
$H_A' = \text{ECC}^{-1}(\text{tk}_A' \oplus \Delta_A)$		$H_B' = \text{ECC}^{-1}(\text{tk}_B' \oplus \Delta_B)$
If $\text{Ham}(H_A', \text{Hash}(s_{2A}^*, (u_{1A}, \text{pw}_A), r_{1A})) \le 2\varepsilon/\ell$		If $\text{Ham}(H_B', \text{Hash}(s_{2B}^*, (u_{1B}, \text{pw}_B), r_{1B})) \le 2\varepsilon/\ell$
$r_{iA}\|\|r_{iA}\|\|\text{sk}_{iA} \longleftarrow H_A'$		$r_{iB}\|\|\tau_{iB}\|\|\text{sk}_{iB} \longleftarrow H_B'$
$\text{sk}_{AB} = m_A \oplus \tau_{iA} \oplus \text{sk}_{iA}$		$\text{sk}_{BA} = m_B \oplus \tau_{iB} \oplus \text{sk}_{iB}$

$\longleftarrow H_A'/r_{iB}\|\|\tau_{iB}\|\|\text{sk}_{iB} \longleftarrow H_B'$ and outputs $\text{sk}_{AB} = m_A \oplus \tau_{iA} \oplus \text{sk}_{iA}/\text{sk}_{BA} = m_B \oplus \tau_{iB} \oplus \text{sk}_{iB}$.

4.2. Correctness. After the protocol is executed honestly, the probability of a mismatch between the session keys obtained by the two clients A and B is negligible. From the approximate correctness of the NA-ASPH, the probability that the Hamming distance between $\text{tk}_A(\text{tk}_B)$ calculated by client A (B) and $\text{tk}_A'(\text{tk}_B')$ calculated by the clients is greater than $(\varepsilon(n) \bullet n)$ can be neglected. Then, from the definition of error correction code ECC, client A (B) and server C can obtain the same $r_{iA}\|\tau_{iA}\|\text{sk}_{iA}(r_{jB}\|\tau_{jB}\|\text{sk}_{jB})$, so A and B can get the same session key,

$$\begin{aligned}\text{sk}_{AB} &= m_A \oplus \tau_{iA} \oplus \text{sk}_{iA} = \tau_{jA} \oplus \text{sk}_{jB} \oplus \tau_{iA} \oplus \text{sk}_{iA} \\ &= \text{sk}_{jB} \oplus \text{sk}_{iA} = \tau_{jB} \oplus \text{sk}_{jA} \oplus \tau_{iB} \oplus \text{sk}_{iB} \\ &= m_B \oplus \tau_{iB} \oplus \text{sk}_{iB} = \text{sk}_{AB}.\end{aligned} \tag{7}$$

4.3. Security. Since the protocol here is symmetrical with respect to the clients, the proof usually can only take one client as an example. The security proof of the protocol follows Section 3 closely. We outline the main ideas. First, based on the CCA security of the underlying primitive Σ, the adversary cannot obtain any useful information about the real password through Execute query. In the Execute query, if the simulator replaces the valid password with an illegal one, guaranteed by the smoothness of NA-ASPH, the adversary cannot distinguish the corresponding two experiments computationally. Second, if the adversary simply replays the messages between participants, the proof is the same as the Execute query. Third, if the adversary modifies the output message of some instances (that is, it modifies (label, c)), the simulator can obtain the corresponding plaintext $\text{pw}_{A/B}^{\text{ad}}$ through the decryption oracle provided by CCA security. If $\text{pw}_{A/B}^{\text{ad}} = \text{pw}_{A/B}$ holds, the corresponding attack is successful, which will increase the adversary's advantage. Using the CCA security of Σ, $\text{pw}_{A/B}$ is uniformly sampled from the password dictionary $\mathcal{D}$, so $\Pr[\text{pw}_{A/B}^{\text{ad}} = \text{pw}_{A/B}] \le 1/|\mathcal{D}|$. Assuming that the adversary can perform at most $Q(\kappa)$ online attacks, then the adversary's advantage is at most $Q(\kappa)/|\mathcal{D}|$. And if $\text{pw}_{A/B}^{\text{ad}} \ne \text{pw}_{A/B}$ holds, then from the adversary's view, the session key obtained is indistinguishable from the uniform distribution (according to the smoothness of NA-ASPH).

5. Protocol Performance Analysis

In this section, we will compare the performance of the two proposed protocols with other related protocols in terms of safety and efficiency. The comparison results are shown in Table 4, where Type represents the protocol type, M-Auth

Table 4: Performance comparisons of PAKE protocols.

Protocol	Type	M-Auth	Anti-Qu	Round	C-method	C-cost
Z-2PAKE [1]	2-party	√	√	2	V-mul	$(2m+2n_1)lbq+n$
K-2PAKE [33]	2-party	✗	√	3	V-mul	$(mn+2n)lbq+3n$
D-2PAKE [34]	2-party	√	√	3	V-mul	$(mn/2+m/2+3n)lbq+2n$
2PAKE	2-party	√	√	2	V-mul	$(2m+3n_1)lbq+n$
A-3PAKE [30]	3-party	✗	✗	4	Exp	—
Y-3PAKE [35]	3-party	√	√	3	V-mul	$2[(mn+n)lbq+3n]$
X-3PAKE [36]	3-party	√	√	3	V-mul	$7nlbq+9n+5$
3PAKE	3-party	√	√	2	V-mul	$2[(2m+3n_1)lbq+2n]$

indicates whether the protocol can provide mutual authentication, Round denotes the number of communication rounds required by the protocol, Anti-Qu represents whether the protocol can resist quantum attacks, C-method indicates the operation method of the protocol, C-cost represents the communication cost of the protocol, V-mul denotes vector multiplication, Exp indicates exponentiation, and $n = n_1 + n_2$.

In terms of security, we mainly compare with other protocols in (1) whether it can resist quantum attacks; (2) whether it can achieve mutual authentication. In terms of efficiency, we mainly compare from the following three aspects: (1) the selection of cryptographic primitives, (2) the calculation method, and (3) the communication overhead. Note that the calculation method adopted is used to roughly measure the computational cost of the corresponding protocol. Moreover, the computational cost of modular exponential operations is much greater than linear operations on matrices and vectors.

Compared with the K-PAKE [33] and D-PAKE [34], the advantage of the 2PAKE is that mutual authentication and key exchange can be achieved within two rounds of transmission. And the size of the ciphertext of K-PAKE and D-PAKE are O (n) larger than 2PAKE. The size of the projection key of K-PAKE and D-PAKE is determined by the ciphertext and the hash key, larger than that of 2PAKE.

Compared with the typical three-party PAKE, A-3PAKE [30], the 3PAKE in this paper is lattice-based and can resist quantum attacks. The proposed 3PAKE can achieve mutual authentication within two rounds of transmission. In addition, A-3PAKE uses exponential operation, while the 3PAKE protocol uses vector multiplication, which has higher computational efficiency.

Compared with Y-3PAKE [35] and X-3PAKE [36] protocols, 3PAKE only requires 2 rounds of transmission. The communication cost of Y-3PAKE protocol mainly depends on the size of the ciphertext, the projection key, and the message authentication code. The size of the ciphertext is O (n) larger than 3PAKE. The size of the projection key of Y-3PAKE is determined by the ciphertext and the hash key, larger than that of 3PAKE. In addition, the Y-PAKE protocol needs to calculate and send a message authentication code for mutual authentication, while 3PAKE performs mutual authentication by verifying the validity of the ciphertext. The amount of messages that needs to be transmitted in X-PAKE is large, resulting in increased communication overhead. Therefore, the communication overhead of the Y-3PAKE and X-PAKE protocols is greater than 3PAKE in this paper.

Z-PAKE [1] is also a two-round protocol, but it is designed for two parties. Compared with the Z-PAKE protocol, 2PAKE adds a projection key to the communication overhead. However, if the three-party PAKE based on Z-PAKE is implemented in a traditional way, at least 4 rounds of communication, that is, 8 message transmissions, are required. But the 3PAKE in this paper only needs 2 rounds of communication, namely 4 message transmissions.

The protocols in this paper have advantages of efficiency and security over traditional protocols based on finite fields, since lattice operations (vector multiplication) are more efficient than exponentiation and lattice problems remain hard for quantum attacks and subexponential-time adversaries. Both protocols in this article can achieve mutual authentication; thus, they can resist imperceptible on-line dictionary attacks. Besides, they are both two-round protocols with less number of transmissions. And the underlying primitive is an improved lattice-based CCA-secure PKE, which can reduce encryption parameters and further reduce computational overhead. In particular, compared with other three-party protocols, the two-round 3PAKE protocol proposed has smaller communication and computation overhead, so it is adaptable to large-scale communication systems.

6. Conclusions

This paper proposes two password-based authenticated key exchange protocols based on the LWE problem from lattices, which can resist quantum attacks and have high efficiency. In the random oracle model, this paper gives a strict security proof of the proposed protocols. In addition, the proposed PAKE protocols can achieve mutual authentication in two rounds of transmission. And the 3PAKE protocol is practical for large-scale communication systems. Compared with the existing related protocols, the protocols in this paper have higher security and lower communication and computing overhead. In our protocols, the client's password is stored on a single server, so the proposed protocols are not resistant to hacker attacks. In the future, we will study the multiserver PAKE protocol that can resist hacker attacks.

Data Availability

The extra data used to support the findings of this study are available from the corresponding author. Email: guo_yuanbo@126.com.

Conflicts of Interest

The authors declare no competing financial interest.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China (Grant No. 61501515) and Foundation of Science and Technology on Information Assurance Laboratory (No. KJ-15-108).

References

[1] J. Zhang and Y. Yu, "Two-Round PAKE from Approximate SPH and Instantiations from Lattices," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 37–67, Hong Kong, China, 2017.

[2] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2776–2791, 2017.

[3] J. Zhao and D. Gu, "Provably secure three-party password-based authenticated key exchange protocol," *Information Sciences*, vol. 184, no. 1, pp. 310–323, 2012.

[4] M. S. Farash, S. H. Islam, and M. S. Obaidat, "A provably secure and efficient two-party password-based explicit authenticated key exchange protocol resistance to password guessing attacks," *Concurrency & Computation Practice & Experience*, vol. 27, no. 17, pp. 4897–4913, 2017.

[5] A. Groce and J. Katz, "A new framework for efficient password-based authenticated key exchange," in *Proceedings of the ACM Conference on Computer and Communications*, pp. 516–525, Chicago, Illinois, 2010.

[6] D. Wang, W. Li, and P. Wang, "Measuring two-factor authentication schemes for real-time data access in industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4081–4092, 2018.

[7] J. Katz, R. Ostrovsky, and M. Yung, "Efficient and secure authenticated key exchange using weak passwords," *Journal of the Association for Computing Machinery*, vol. 57, no. 1, pp. 79–117, 2010.

[8] Z. Li and D. Wang, "Achieving one-round password-based authenticated key exchange over lattices," *IEEE Transactions on Services Computing*, vol. 2019, no. 8, pp. 1–14, 2019.

[9] R. Gennaro and Y. Lindell, "A framework for password-based authenticated key exchange1," *ACM Transactions on Information & System Security*, vol. 9, no. 2, pp. 181–234, 2006.

[10] S. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks," in *2012 IEEE Symposium on Security and Privacy*, p. 72, Oakland, California, 1992.

[11] T. Lomas, L. Gong, J. Saltzer, and R. Needhamn, "Reducing risks from poorly chosen keys," in *Proceedings of the twelfth ACM symposium on Operating systems principle (SOSP '89)*, pp. 14–18, New York, NY, 1989.

[12] L. Gong, M. A. Lomas, R. M. Needham, and J. H. Saltzer, "Protecting poorly chosen secrets from guessing attacks," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 648–656, 1993.

[13] S. M. Bellovin and M. Merritt, "Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 244–250, New York, NY, 1989.

[14] L. Gong, "Optimal authentication protocols resistant to password guessing attacks," in *Proceedings of the 8th IEEE Computer Security Foundations Workshop*, pp. 24–29, Los Alamitos, California, 1995.

[15] M. Steiner, G. Tsudik, and M. Waidner, "Refinement and extension of encrypted key exchange," *Acm Sigops Operating Systems Review*, vol. 29, no. 3, pp. 22–30, 1995.

[16] W. Thomas, "The Secure Remote Password Protocol," in *Proceedings of the Network and Distributed System Security Symposium (NDSS'98)*, pp. 97–111, San Diego, California, 2000.

[17] M. Bwllare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proceedings of the Advances in Cryptology (Eurocrypt'00)*, pp. 139–155, Berlin, Germany, 2000.

[18] V. Boyko, P. D. Mackenzie, and S. Patel, "Provably secure password-authenticated key exchange using Diffie-Hellman," in *Proceedings of Advances in Cryptology (Eurocrypt'00)*, pp. 156–171, Berlin, Germany, 2000.

[19] O. A. Goldreich and Y. Lindell, "Session-Key generation using human passwords only," *Journal of Cryptology*, vol. 19, no. 3, pp. 241–340, 2006.

[20] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques*, pp. 139–155, Bruges, Belgium, 2000.

[21] V. Boyko, P. Mac Kenzie, and S. Patel, "Provably secure password-authenticated key exchange using Diffie-Hellman," in *EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques*, pp. 156–171, Bruges, Belgium, 2000.

[22] E. Bresson, O. Chevassut, and D. Pointcheval, "Security proofs for an efficient password-based key exchange," in *Proceedings of the 10th ACM conference on Computer and communications security*, pp. 241–250, Washington DC, 2003.

[23] M. Abdalla, F. Benhamouda, and D. Pointcheval, "Disjunctions for hash proof systems: new constructions and applications," in *EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 69–100, Sofia, Bulgaria, 2015.

[24] F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud, "New techniques for SPHFs and efficient one-round PAKE protocols," in *Advances in Cryptology–CRYPTO 2013*, pp. 449–475, Santa Barbara, CA, 2013.

[25] J. Katz and V. Vaikuntanathan, "Round-optimal password-based authenticated key exchange," in *8th Theory of Cryptography Conference*, pp. 293–310, Providence, RI, 2011.

[26] S. Jiang and G. Gong, "Password Based Key Exchange with Mutual Authentication," in *Selected Areas in Cryptography (SAC 2004)*, pp. 267–279, Waterloo, Canada, 2004.

[27] Y. Mao, *Research on Password-Based Authenticated Key Exchange Protocols and Associated Encryption Algorithms from Lattices*, PLA Information Engineering University, 2012.

[28] J. Katz, P. MacKenzie, G. Taban, and V. Gligor, "Two-server password-only authenticated key exchange," *Journal of Computer & System ences*, vol. 78, no. 2, pp. 651–669, 2005.

[29] R. Mario and G. Raimondo, "Provably secure threshold password-authenticated key exchange," *Journal of Computer and System Sciences*, vol. 72, no. 6, pp. 507–523, 2006.

[30] M. Abdalla, P. A. Fouque, and D. Pointcheval, "Password-Based Authenticated Key Exchange in the Three-Party Setting," in *8th International Workshop on Theory and Practice in Public Key Cryptography*, pp. 65–84, Les Diablerets, Switzerland, 2005.

[31] W. Minghui and W. Jiandong, "Three-party authentication key exchange protocol based on password," *Computer Engineering*, vol. 38, no. 2, pp. 146–150, 2012.

[32] W. Guocai, K. Fusong, and W. Fang, "ECDSA-based password authenticated key exchange protocol for threeparty," *Computer Engineering*, vol. 38, no. 6, pp. 153–155, 2012.

[33] K. Jonathan and V. Vaikuntanathan, "Smooth Projective Hashing and Password-Based Authenticated Key Exchange from Lattices," in *Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, pp. 636–652, Tokyo, Japan, 2009.

[34] Y. Ding and L. Fan, "Efficient password-based authenticated key exchange from lattices," *International Journal of Advancements in Computing Technology*, vol. 1, no. 22, pp. 934–938, 2011.

[35] M. Ye, X. Hu, and W. Liu, "Password authenticated key exchange protocol in the three party setting based on lattices," *Journal of Electronics & Information Technology*, vol. 35, no. 6, pp. 1376–1381, 2013.

[36] D. Xu, "Provably Secure Three-party Password Authenticated Key Exchange Protocol Based on Ring Learning with Error," *IACR Cryptol. ePrint Arch*, vol. 2017, p. 360, 2017.

[37] R. Cramer and V. Shoup, "Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption," in *EUROCRYPT 2002: International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 45–64, Amsterdam, The Netherlands, 2002.