Research Article

Improved Verifier-Based Three-Party Password-Authenticated Key Exchange Protocol from Ideal Lattices

Yongli Tang, Ying Li, Zongqu Zhao, Jing Zhang, Lina Ren, and Yuanhong Li

School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454000, China

Correspondence should be addressed to Zongqu Zhao; zhaozong_qu@hpu.edu.cn

Received 19 August 2021; Accepted 28 September 2021; Published 29 November 2021

Abstract

With the advent of large-scale social networks, two communication users need to generate session keys with the help of a remote server to communicate securely. In the existing three-party authenticated key exchange (3PAKE) protocols, users' passwords need to be stored on the server, which cannot resist the server disclosure attack. To solve this security problem, we propose a more efficient 3PAKE protocol based on the verification element by adopting a public-key cryptosystem and approximate smooth projection hash (ASPH) function on an ideal lattice. Using the structure of separating authentication from the server, the user can negotiate the session key only after two rounds of communication. The analysis results show that it can improve the efficiency of computation and communication and resist the server disclosure attack, quantum algorithm attack, and replay attack; moreover, it has session key privacy to the server. This protocol can meet the performance requirement of the current communication network.

1. Introduction

In 1976, Diffie and Hellman [1] first proposed a key exchange (KE) protocol to generate a session key between two users, to realize the secure transmission of information in the channel, but they did not consider the identity authentication of two users. Then, authenticated key exchange (AKE) protocol is proposed based on the KE protocol. AKE protocol ensures that it can still correctly generate the session key among two users in the case of an active attack by the adversary.

AKE protocol can be classified into identity-based AKE protocol, public-key infrastructure (PKI) based AKE protocol, and password-based AKE (PAKE) protocol. PAKE protocol dislodges the public-key infrastructure in the network and takes the low entropy password as the input so that the participants can negotiate a high entropy session key after mutual authentication on the insecure channel. The PAKE protocol has attracted extensive attention because the password is short, practical, and easy to remember.

With the emergence of large-scale user mutual communication, the implementation of the two-party authenticated key exchange (2PAKE) protocol increases the burden of user password management [2–5], and the 3PAKE protocol allows users to negotiate a session key with other users when they share a password with the server. In the 3PAKE protocol, if the user’s password is stored on the server in plaintext, it is called symmetric 3PAKE protocol. Once the server is attacked and the password file is leaked, the attacker can forge a legitimate user to access after obtaining the user’s password [6–10]. Rising is a company focusing on the research and development of anti-virus products and network security products. Its threat intelligence system is based on threat detection technology of big data analysis and can trace the trajectory of threat behavior by using threat intelligence, and it once captures an overseas virus transmission server. This virus transmission server scans the server in the network by using a weak password. Once the server is found, it will implant a virus to obtain the password file on the server. When the virus transmission server is captured, it has stored the IP address and account’s weak password of more than 2000 MSSQL servers and more than 600 phpMyAdmin servers.

Against this type of server file disclosure attack, Kwon et al. [11] constructed the first 3PAKE protocol based on verification element in 2007. The user sends the transformed value of the password to the server as a verification element. Now, even if the verification element on the server is leaked,
the adversary cannot speculate the user’s plaintext password. When the adversary carries out an offline dictionary attack to obtain the user’s password, the server can notify the user that the verification element has been leaked and allow the user to reexecute the registration phase and generate a new verification element [12]. This approach effectively solves the disadvantages of the symmetric 3PAKE protocol.

In 2016, Yang et al. [13] proposed the first 3PAKE protocol based on a verification element under the standard model. However, through analysis, it is found that the user calculates the message sent to the server according to the password in the process of key agreement. Now, the attacker can use the message authentication code (MAC) value in the information to execute the offline dictionary attack on the user’s password. In 2020, Zhang et al. [14] used the smooth projection hash function based on Yang’s protocol and improved it by using the DDH difficulty hypothesis, so that after four rounds of communication between participants, it can negotiate a session key of both sides.

With the advent of the quantum computer, the traditional number theory problems cannot effectively resist the solution of the quantum algorithm, and the difficult problems on the lattice make the complexity of the worst-case consistent with the average case. Therefore, as a lattice cryptosystem that can mitigate the quantum attack, it has attracted extensive attention. In 2012, Ding et al. [15] first constructed the KE protocol on the lattice. Ye et al. [16] first constructed the 3PAKE protocol on the lattice in 2013. With continuous research, in 2018, Yu et al. [17] constructed a new 3PAKE protocol by using the approximate smooth projection hash function on the lattice. They used the separable public-key encryption system, and a session key can be negotiated in only two rounds of communication, reducing the communication overhead. In the same year, Choi et al. [18] designed a new 3PAKE protocol; they introduced implicit server authentication based on Ding et al. so that it can still safely complete key negotiation under incomplete trusted servers. To improve the computational efficiency, Liu et al. [19] proposed a 3PAKE protocol based on RLWE. Their protocol only depends on the hardness of the RLWE problem; it has no additional primitive in the protocol designed and can resist undetectable online password attacks and offline password attacks. They struck a balance between efficiency and security. However, these symmetric 3PAKE protocols cannot resist server disclosure attack [20, 21]. In 2019, Zhang et al. [22] proposed a 3PAKE protocol based on the verification element under the standard model. To enhance the security of the protocol, it uses anonymous authentication for server and user authentication, which increases the computation cost. In 2021, Shu et al. [23] adopted the Peikert [24] error reconciliation mechanism and proposed a 3PAKE protocol based on the verification element on the ideal lattice; it reduces the space complexity, but it needs six rounds of communication to negotiate a session key and increases the communication overhead.

It can be seen that the existing 3PAKE protocols cannot solve the server disclosure attack and reduce the communication overhead, and the authentication cannot be separated from the server. Therefore, we construct a 3PAKE protocol based on the verification element using the approximate smooth projection hash function [25] on the ideal lattice. Specifically speaking, the main contributions are as follows:

(i) Reducing space complexity: We use the public-key encryption system on the ideal lattice to reduce the size of the key and ciphertext. Because the ciphertext will be transmitted on the channel, shortening the ciphertext size can effectively improve the communication overhead efficiency.

(ii) It can simplify the protocol flow and effectively reduce the communication overhead. The proposed 3PAKE protocol adopts an asymmetric structure to separate the authentication from the server. Using the approximate smooth projection hash function on the ideal lattice, each user can only transmit information with the server once to generate the session key. However, in the existing 2PAKE and 3PAKE, it needs at least four messages sent by the user and the server to generate a session key.

(iii) Lower computational overhead: Most of the calculations are carried out on the polynomial ring in our protocol, so we can use the Fast Fourier Transform (FFT) algorithm to decrease the number of operations. In addition, the operation in the protocol can be parallelized by using the particularity of polynomial $f(x) = x^n + 1$ and module $q \equiv 1 \mod 2^n$. Finally, our protocol can obtain the optimal implementation in the domain $\mathbb{Z}_{q^n}$ to significantly improve the operating rate.

(iv) Greater security: Not only does the proposed protocol depend on the server to distribute the session key, but also the user’s temporary private key determines part of the session key. Thus, the session key is private to the server, and it has forward security.

The organization of this paper is as follows: we introduce the ideal lattice, RLWE problem and discrete Gaussian function, and other knowledge and give the specific structure of the public-key encryption algorithm and the approximate smooth hash function on the ideal lattice in Section 2. Then, in Section 3, we introduce the security model of the proposed protocol. The complete structure and security proof of the protocol are given in Sections 4 and 5, respectively. Finally, in Section 6, we compare the security with the protocols in [2, 14, 17, 23], and the experimental results of the protocol under different initial parameters are given to show the computation and communication overhead of each stage of the protocol.

2. Preliminaries

2.1. Lattices

Definition 1. Let $B = \{b_1, b_2, \ldots, b_m\} \in \mathbb{R}^{m \times n}$ be a set of linearly independent vectors in $\mathbb{R}^n$, and $n$ and $m$ are positive integers. The lattice generated by $B$ is defined as
\( \mathcal{L}(B) = \{ \sum_{i=1}^{m} x_i b_i : x_i \in \mathbb{Z}^m \} \), and \( B \) is the basis of \( \mathcal{L}(B) \). \( n \) and \( m \) are the dimension and rank of the lattice, respectively.

**Definition 2** (circular lattice). Note that the once cyclic displacement of the vector \( a = (a_0, a_1, \ldots, a_{n-1})^T \) is written as \( \text{rot}(a) = (a_{n-1}, a_0, a_1, \ldots, a_{n-1})^T \). For lattice \( \mathcal{L} \), if \( \forall a \in \mathcal{L} \), \( \text{rot}(a) \in \mathcal{L} \), then the lattice \( \mathcal{L} \) is called a cyclic lattice, record as \( \text{Rot}(a) = \{ a, \text{rot}(a), \ldots, \text{rot}^{n-1}(a) \} \), and expressed as

\[
\text{Rot}(a) = \begin{bmatrix}
a_0 & a_{n-1} & \cdots & a_1 \\
a_1 & a_0 & \cdots & a_2 \\
\vdots & \vdots & \ddots & \vdots \\
a_{n-1} & a_{n-2} & \cdots & a_0 
\end{bmatrix}.
\]

**Definition 3** (ideal lattice). In 2006, Lyubashevsky et al. [26] extended the cyclic lattice and first proposed the ideal lattice. A lattice, having a special ring structure, is called an ideal lattice. \( q \) is a prime number and satisfies \( n = 2^k \) \((k > 1)\), \( q \equiv 1 \pmod{2n} \), \( f(x) = x^n + 1 \in \mathbb{Z}[x] \), and \( f(x) \) is irreducible on the set of all rational numbers. \( R = \mathbb{Z}[x]/f(x) \) is a ring of the integer polynomial \( f(x) \); then, the elements in \( R \) are usually represented by integer polynomials of less degree \( n \). If vector \( a \in R \), the cyclic lattice \( \mathcal{L}(\text{Rot}(a)) \) generated by \( a \) is an ideal ring of modules \( \mathcal{L}[x]/(x^n - 1) \). \( \mathcal{L}(a) = \mathbb{Z}[x]/f(x) \) is an integer polynomial ring of modules \( f(x) \) and \( q \), in which the elements can be represented by a polynomial of degree \( n - 1 \) and the coefficients are expressed as \( [0, 1, \ldots, q - 1] \). The ideal lattice lays down the space complexity by using a vector to represent an \( n \) dimensional lattice.

**Definition 4** (RLWE problem). Let \( R_q = \mathbb{R}/q \mathbb{R} \) be a quotient ring with a positive integer \( q \) as the module. We suppose \( n, m \geq 1, q > 2 \), an error distribution \( \chi_\beta \) which is a Gaussian distribution over \( R_q \), where \( \beta \) is the noise parameter, satisfying \( \beta \in (0, 1) \) and \( \sqrt{\log q} \lesssim \beta \lesssim (\sqrt{q})/4 \). For \( s \in R_q \), it will output \( (a, a \cdot s + e \text{ (mod } q)) \in R_q \times R_q \) through sampling \( a \leftarrow R_q \) and noise \( e \leftarrow \chi_\beta \).

**Definition 5** (decision RLWE problem). Given \( m \) independent uniformly distributed samples from \( R_q \times R_q \), there is no arbitrary probabilistic polynomial time (PPT) algorithm to distinguish whether the sample is chosen from RLWE distribution or uniform and random \( R_q \times R_q \).

**Definition 6** (search RLWE problem). Randomly generate polynomial \( a \in R_q, s \in R_q \), and \( e \leftarrow \chi_\beta \); they satisfy \( b_i = a_i + c_i \). Now, given multiple groups \( (a_i, b_i) \), it is needed to find \( s \).

**Theorem 1** (see [25]). We suppose \( n = 2^k \) \((k > 1)\), \( q \equiv 1 \pmod{2n} \), \( \beta \in (0, 1) \); they satisfy \( \beta q \geq w(\sqrt{\log m}) \). Then, the \( \tilde{O}(\sqrt{n}/\beta) \)-approximate SVP (shortest vector problem) can be reduced to \( R \rightarrow \text{DLEW}_{nm, q, \beta, \gamma} \) problem and \( R \rightarrow \text{SLEW}_{nm, q, \beta, \gamma} \) problem through quantum.

### 2.2. Discrete Gaussian Distribution

For any \( \sigma > 0 \), \( \rho_{\sigma, c}(x) = \exp(-\pi \|x - c\|^2/\sigma^2) \) is a Gaussian distribution function taking \( c \in R \) as the center and \( \sigma \) as the standard deviation. For any \( \sigma > 0 \), with taking \( c \in R^n \) as the center and \( \sigma \) as the parameter, the discrete Gaussian distribution over lattice \( \mathcal{L} \) is defined as \( D_{\phi_{\sigma,c}} = \{ \rho_{\sigma,c}(y) \} \), where \( y \in \mathcal{L} \phi_{\sigma,c} = \{ \sum \phi_{\sigma,c}(x) \} \). Note that \( c \) can be ignored not writing when \( c = 0 \).

**Theorem 2.** Gaussian distribution has the following characters: given the standard deviation \( \sigma \) and positive integer \( m \), the following formulas hold:

\[
\begin{align*}
\Pr[x \leftarrow D_{\phi_c}^\sigma : \|x\| > 2\sigma \sqrt{m}] &< 2^{-m}, \\
\Pr[x \leftarrow D_{\phi_c}^\sigma : \|x\| > \sigma(\sqrt{\log m})] &< 2^{-\log m}.
\end{align*}
\]

### 2.3. Public-Key Cryptosystem Based on Ideal Lattice

#### (i) \((pk, sk) \leftarrow \text{KeyGen}(1^n)(1/2)\): this is a key generation algorithm, which inputs a security parameter \( n \) and \( m \) and an odd prime and satisfies \( n \geq 2^k, q \equiv 1 \pmod{2n}, m \geq 6\log q \). A public-key cryptosystem, whose difficulty is based on the RLWE problem, is mainly composed of three algorithms.

1. \((pk, sk) \leftarrow \text{KeyGen}(1^n)(1/2)\)
2. \((c, m) \leftarrow \text{Enc}(pk, msg)\): this is an encryption algorithm. It takes as input a public key \( pk \) and a plaintext \( msg \) and outputs a ciphertext \( c = (c_1, c_2) \)
3. \(msg \leftarrow \text{Dec}(sk, (c_1, c_2))\): this is the decapsulation algorithm corresponding to the encapsulation algorithm; it takes as input the private key \( sk \) and ciphertext \( (c_1, c_2) \) and outputs the corresponding plaintext \( m \) or \( -1 \).

### 2.4. Approximate Smooth Projection Hash Function

The smooth projective hash function is an important component in lattice-based cryptography. It was first proposed by Cramer and Shoup [27]. Later, to construct a PAKE protocol against quantum attack, Katz et al. [28] improved it and extended it to the construction of the PAKE protocol for efficient communication. In this paper, we adopt the approximate smooth projection hash (ASPH) function [25] based on an ideal lattice and further modify it according to the requirements of our protocol.

We assume \( PK_e = (\text{KeyGen}, \text{Enc}, \text{Dec}) \), and it is a semantically secure public-key encryption system composed of functions on the lattice. \( C_{\sigma, \beta} \) represents the effective ciphertext space generated by public key \( pk \) encryption, and P
is the plaintext space. We define $X$ and language $L \in X$ as follows:

$$X = \{(c, m) \mid c \in C_{pk}; m \in P\},$$

$$T_m = \{(c, m) \in X \mid c = Enc_{pk}(m, r), r \in \{0,1\}^n\}, \bar{T} = \bigcup_{m \in T_m} T_m,$$

$$L_m = \{(c, m) \in X \mid m = \text{Dec}_{sk}(c), L = \bigcup_{m \in L_m} L_m\}. \tag{3}$$

For any word $c \in L$, the hash value of $c \in L$ can be expressed in two ways: using the hash key $hk$ and $c$ or using the projection key $hp$ and the evidence $w$ corresponding to $c \in L$. The function $\varepsilon - ASPH$ of a public-key encryption system corresponding to the public key $pk$ on the lattice is composed of four algorithms, which can be expressed as $\varepsilon - ASPH = (\text{HashKG}, \text{ProjKG}, \text{Hash}, \text{ProjH})$.

1. $\text{HashKG}(1^n)$: Given the security parameter $n$, the hash key generation algorithm outputs the hash key $hk$. Note $H = [H_{hk}]_{hk\in HK}$ represents the hash function cluster and $HK$ is the hash key space.

2. $\text{ProjKG}(hk, pk)$: The projection key generation algorithm takes a hash key $hk$ and a public key $pk$; it outputs the corresponding projection key $hp \in HP$, where $HP$ is the projection key space.

3. $\text{Hash}(hk, L, c)$: When it inputs the hash key $hk$, language $L$, and any word $c \in L$, the hash function outputs the hash value $h$. $f$ ($n$) correctness: for all $c \notin L$ and corresponding evidence $w$, let $\varepsilon(n) \in [0, 1]$. $\text{Ham}(a, b)$ represents the hamming distance between $a$ and $b$; and then $\Pr[\text{Ham}\left(\text{Hash}\left(hk, L, c\right), p \leftarrow \{0, 1\}\right), (hp, w) \geq \varepsilon(n) \cdot n] \leq \text{negl}(n)$ holds.

4. $\text{ProjH}(hp, w)$: This is a projection hash function. Let the projection key $hp$ and evidence $w$ of word $c \in T$ be the input, and it outputs the projection hash value $h'$.

$\varepsilon(n)$ correctness: for all $c \notin T$ and corresponding evidence $w$, let $\varepsilon(n) \in [0, 1]$. $\text{Ham}(a, b)$ represents the hamming distance between $a$ and $b$; and then $\Pr[\text{Ham}\left(\text{Hash}\left(hk, L, c\right), p \leftarrow \{0, 1\}\right), (hp, w) \geq \varepsilon(n) \cdot n] \leq \text{negl}(n)$ holds.

Smootheness: for all $c \notin T$, $hp = \text{ProjKG}(hk, L, c)$, $p \leftarrow \{0, 1\}$, the distributions of $(hp, \text{Hash}(hk, L, c))$ and $(hp, p)$ are indistinguishable in the statistical distance. When $n$ is the security parameter, $\varepsilon_{\text{ASPH}}(n)$ is defined as a negligible upper bound of the statistical distance of the two distributions.

2.5. Specific Internal Structure. Combined with the content introduced in Section 2.3, we construct a public-key encryption scheme on the ideal lattice and instantiate the approximate smooth projection hash function in the proposed protocol. The specific structure is as follows.

2.5.1. Public-Key Encryption Scheme Based on RLWE Problem

1. $(pk, sk) \leftarrow \text{KeyGen}(1^n)$: Input a security parameter $n$; then, it selects $B_0 \leftarrow \mathbb{R}_{q^n}$ and runs the trapdoor function to get $(B_1, T_1) \leftarrow \text{ideal - trapGen}$ and $(B_2, T_2) \leftarrow \text{ideal - trapGen}$, where $B_0, B_1, B_2 \in \mathbb{R}_{q^n}$. $T_1, T_2 \in \mathbb{R}_{\text{mon}}$. It finally outputs a public/private key pair $(pk, sk)$ of the system.

2. $\text{ProjH}(hp, \text{msg})$: Let $pk = (B_0, B_1, B_2)$ and a plaintext $m \in \mathbb{Z}_q^n$ be input; then, it selects $r \leftarrow \mathbb{R}_{\{0, 1\}}$, the coefficients of $c_1$ and $c_2$ obey distribution $\mathcal{X}_p$. It outputs $c = (c_1, c_2)$, where $c_1 = B_1 \cdot w_a + e_1(\text{mod}q), \ c_2 = B_0 \cdot w_a + B_2 \cdot m + e_2(\text{mod}q)$. $\text{msg} \leftarrow \text{Dec}(sk, (c_1, c_2))$: The private key $sk$ and ciphertext $(c_1, c_2)$ are input; this decapsulation algorithm outputs the corresponding plaintext $m$ or “⊥”.

2.5.2. Approximate Smooth Projective Hash Function on Ideal Lattice. (1) Hash key: The hash key space is $HK = (\mathbb{R}_q^n)^n$ used in this protocol to ensure the approximate correctness of $\varepsilon - ASPH$ function, and the coefficients of polynomial $e_j (j \leq n)$ must obey Gaussian distribution $\mathcal{X}_p$ for any $(e_1, e_2, \ldots, e_n) \in HK$.

2. Projection key: The projection key is generated by the hash key. For any $(e_1, e_2, \ldots, e_n) \in HK$, the corresponding projection key is $(u_1, u_2, \ldots, u_n) = a(e_1, e_2, \ldots, e_n) \in HP$, and $HP \in (\mathbb{Z}_q^{n+1})$ is the projection key space. The specific calculation process is as follows:

$$u_j = (\text{Map}_{M_{-1}}(e_j))^T \cdot \tilde{B}_0 (j \leq n). \tag{4}$$

$Map_{M_{-1}}(e_j)$ is the result of connecting the coefficients of polynomial $e_j \in \mathbb{R}_q^n$, and finally outputs one a one-dimensional column vector composed of coefficients $e_j$. After performing this type of operations on all $e_j \in (e_1, e_2, \ldots, e_n)$, the result of transpose operation will be point multiplied by $\tilde{B}_0$. $B_0 \leftarrow \mathbb{R}$ is a public parameter and $\tilde{B}_0$ is generated by the following calculations:

$$B_0 = \begin{bmatrix} b_{01} \\ b_{02} \\ \vdots \\ b_{0m} \end{bmatrix}, \tilde{B}_0 = \begin{bmatrix} \text{rot}(b_{01})^T \\ \text{rot}(b_{02})^T \\ \vdots \\ \text{rot}(b_{0m})^T \end{bmatrix} \in \mathbb{Z}_q^{m\times n}. \tag{5}$$

3. Hash function $\mathcal{H} = (H_{hk})_{hk \in HK}$: the hash key $hk = (e_1, e_2, \ldots, e_n) \in HK$ and $x = (c, m)$ are used as input, and then perform the following calculations:

$$z_j = (\text{Map}_{M_{-1}}(e_j))^T \cdot \text{Map}_{M_{-1}}(c_2 - B_2 \cdot m) \in \mathbb{Z}_q.'$$

Outputs:

$$b_j' = \begin{cases} 0, & \text{if } z_j < \frac{(q - 1)}{2} \\ 1, & \text{if } z_j > \frac{(q - 1)}{2} \end{cases} \tag{6}$$

4. Projection function $\{\text{ProjH}(hp)\}_{hp \in HK}$: let the projection key $hp = (u_1, u_2, \ldots, u_n)$ and the evidence $w \in T$ be input, and run the following calculations:
3.2. Long-Term Key. The long-term key in the protocol is the user’s password; we assume the length of the nonempty dictionary $D$ is $l$, and the password $pw_U$ of each user $U$ in $\mathcal{U}$ is randomly selected from $D$. The server $S$ has a password list $pw_S = <pw_S[U] | \forall U \in \mathcal{U}>$ of all users, where $pw_S[U] = \{salt_U, V_U\}$ is composed of the user’s salt value $salt_U$ and the verification element $V_U$ produced by the password $pw_U$ and salt value $salt_U$.

3.3. Security Model of 3PAKE Protocol. In the 3PAKE protocol, each participant can execute multiple sessions at the same time. Let $U_i$ represent the $i-th$ instance of user $U$ and let $S_j$ represent the $j-th$ instance of the server $S$. Here, an instance represents a session. We suppose there is a PPT algorithm adversary $A$ which knows all the malicious user’s password set $pw_A = <pw_U | \forall U \in \mathcal{U}>$ and also can control the communication channel among all users. Adversary can obtain the specific abilities by sending the following queries.

(i) Execute $(U_i^a, S_j, U_j^b)$: This query gives $A$ the ability to wiretap channels. After activating instances $U_i^a, S_j, U_j^b$, $A$ will get all the information transmitted between the user and the server.

(ii) Send $(U_i^a, msg)$: This query simulates $A’s$ active attack on the user instance. The adversary $A$ selects the message $msg$ and sends it to the user instance $U_i^a$. Finally, $A$ obtains the reply of $U_i^a$ to the message $msg$.

(iii) Send $(S_j, msg)$: This query helps $A’s$ actively attack the server instance. The adversary $A$ selects the message $msg$ and sends it to the server instance $S_j$. Finally, the adversary $A$ obtains the reply of the instance $S_j$ to the message $msg$.

(iv) Reveal $(U_i^a)$: This query simulates session key is lost or leaked. After the adversary $A$ executes this query, $A$ can obtain all session keys $sk_A^a$ of the instance $U_i^a$.

(v) Corrupt $(U)$: The execution of this query simulates the adversary’s corruption attack on user $U$. After $A$ executes this query, it will obtain the user’s password $pw_U$.

(vi) Corrupt $(S)$: Adversary $A$ sends this query to launch a corruption attack on the server $S$. After $A$ executes this query, it will obtain the password list $pw_S = <pw_S[U] | \forall U \in \mathcal{U}>$ of the server $S$.

(vii) Test $(U_i^a)$: When this query is executed, it is valid only when the user instance $U_i^a$ is fresh. We throw a random coin $b \in \{0, 1\}$. If $b = 1$, it will return the real session key $SK_{A,S}$ to the adversary $A$; otherwise, it will return a random bit string with the same length as the session key to the adversary $A$.

3.4. Accepted State. $ssid_i$ represents the session serial number of the user instance $U_i^a$ and $pid_i$ represents the intended communicator of $U_i$. If an instance executes successfully and generates the corresponding session key, we say it has been accepted.

3.5. Matching Session. If (1) the instances $U_i^a$ and $U_j^b$ all have been accepted; (2) $U_i^a$ and $U_j^b$ have the same session serial number $ssid$; and (3) $U_i^a$ and $U_j^b$ are mutually intended the verification element $V_U$ produced by the password $pw_U$ and salt value $salt_U$.

Theorem 3 (see [18]). If the parameters $m, n, q, \beta, \omega$ satisfy $\sqrt{q} \cdot w \cdot (\sqrt{\log m}) < r < \omega/(8 \cdot mn^2 \beta)$, then the above structure $X = \{H_{hk}\}_{hk \in HK}$ is an $\epsilon$-ASP function.
3.6. Fresh session. If the instance $U^i$ has been accepted and the adversary $\mathcal{A}$ has not asked a Reveal query to $U^i$, as well as the adversary $\mathcal{A}$ has not asked a Corrupt query to the user $U$ and server $S$ before instance $U^i$ is accepted, then $U^i$ is fresh.

3.7. Definition of Security. During executing a 3PAKE protocol, any PPT adversary $\mathcal{A}$ can ask Execute, Send, Reveal, and Corrupt queries in any order for many times. Note that Test query can only be asked once. At the end of the game, the adversary $\mathcal{A}$ outputs his guess $b'$ about $b$. If $b' = b$, it means that adversary $\mathcal{A}$ has broken the protocol. Let $D$ be the space of user password and let $P$ represent the protocol which $\mathcal{A}$ attacks. The advantage of $\mathcal{A}$ breaking the 3PAKE protocol is\( \text{Adv}_{\mathcal{A}}^{3PAKE}(\mathcal{A}) = 2pr[\text{Succ}_{\mathcal{F},D}^{3PAKE}(\mathcal{A})] - 1 \), where $\text{Succ}_{\mathcal{F},D}^{3PAKE}(\mathcal{A})$ represents the success probability of breaking the protocol.

4. Construction of Our Protocol

When users make a key agreement with the help of the server, they need to register with the server first, let the server store the verification element corresponding to the user password, and ensure that the server cannot recover the plaintext password through the verification element.

4.1. System Initialization Phase. Running key generation algorithm $(pk, sk) \leftarrow \text{KeyGen}(1^n)$; it takes as input a secure parameter $n$, then selects $b_0 \leftarrow \mathcal{R}$, and runs trapdoor generation algorithm to get $(B_1, T_1) \leftarrow \text{ideal-trapGen}$ and $(B_2, T_2) \leftarrow \text{ideal-trapGen}$. It finally outputs $pk = (B_0, B_1, B_2)$ as a public key and $sk = (T_1, T_2)$ as a secret key. Note that keep private key $sk = (T_1, T_2)$ secret, and $\text{Param} = \{B_0, B_1, B_2\}$ public.

4.2. User Registration. When users join the system for the first time, they need to register. The specific operations are as follows:

- (i) User $U_i$ chooses identification $ID_i$ and password $pw_i$, selects a salt value $\text{salt}_i$ randomly, and uses SHA-256 to generate seeds of two pseudo-random number generators (PRNG): one is seed1 = SHA-256(\text{salt}_i||SHA-256(\text{ID}_i||pw_i)) and the other is seed2 = SHA-256(seed1).

- (ii) Input the seeds of the pseudorandom number generator, then select $s_i, e_i$ from the discrete Gaussian distribution $\chi_{\beta}$, note that $s_i$ and $e_i$ are polynomials whose coefficients obey $\chi_{\beta}$, choose $a \leftarrow \mathcal{R}$, calculate the verification element $v_i = a \cdot s_i + e_i \in B_0$ corresponding to the user $U_i$, and let $a$ be public and the other secret.

- (iii) User $U_i$ sends $(a, ID_i, v_i)$ to the server $S$ through the secure channel. If $(a, ID_i, v_i)$ is not in the database list $\mathcal{G}$, $(a, ID_i, v_i)$ will be automatically added to $\mathcal{G}$.

Otherwise, it will send a new registration message to the user. After the user $U_i$ successfully executes the registration phase, $(\text{seed}_1, \text{seed}_2, s_i, e_i, v_i)$ needs to be deleted from the local memory and $(pw_i, \text{salt}_i)$ is stored locally.

4.3. Mutual Authentication and Key Agreement Phase. This stage is shown in Figure 1; the user $U_a$ negotiates a session key $SK_{ab}$ with the user $U_b$. When user $U_i$ has a session, $U_i$ will automatically generate a session $ID_{ssid_i}$ and ssid$_i$ exists in increasing form. At the same time, after each session is successfully executed, the server will record the user session serial number in the local database list to effectively prevent message replay.

(iii) The server $S$ first looks up the verification element corresponding to $U_i$ in the local list. If it cannot be found, exit. Otherwise, $S$ checks the $U_a$‘s session serial number $ssid_a$. If $ssid_a$ does not meet the requirements, exit. Finally, $m_a$ and $h_a$ are recovered according to the local information, and the $U_a$‘s identity authentication is realized by verifying the effectiveness of the ciphertext $(C_{a1}, C_{a2})$, the integrity verification of the message $x_a$ is realized by verifying the effectiveness of $\phi_{a1}$. If they are all valid, the server $S$ selects $\delta_{sa} \leftarrow \{0, 1\}$, $\delta_{sb} \leftarrow \{0, 1\}$, $e_4, e_5 \leftarrow \mathcal{R}$, $w_4, w_5 \leftarrow \mathcal{R}$ randomly, and $e_4$ and $e_5$ are polynomials whose coefficients obey discrete distribution $\chi_{\beta}$. The hash key $h_k = \text{Hash}^{\text{KG}}(1^n)$ of user $U_b$ is randomly selected, and $S$ calculates the projection key $h_pb = \text{Proj}^{\text{KG}}(pk, h_k)$, projection function value $h_pb = \text{Proj}^{\text{H}}(h_pb, w_4)$ and $m_b = ID_{ssid}||s||\text{ ssid}, \text{ ssid}, v_i || 1 \cdots$ and gets ciphertext $C_{b1} = B_1 \cdot w_4 + e_4 \pmod{m_b}$ and $C_{b2} = B_0 \cdot w_5 + B_2 \cdot m_b + e_5 \pmod{m_b}$ for massage $m_b$. Next, $S$ uses random values $\delta_{sa}$ and $\delta_{sb}$ to calculate $c_a = F_{\delta_{sa}}(1)B_1F_{\delta_{sb}}(3)$, $c_b = F_{\delta_{sa}b}(1)B_1F_{\delta_{sb}}(3)$, $\Delta_a = h_a \cdot \text{ECC}(\delta_{sa})$, $\Delta_b = h_b \cdot \text{ECC}(\delta_{sb})$. Finally, the projection function value $h_pb$ is used to regenerate the verifiable MAC value $\phi_{b1} = \text{MAC}(U_b, ID_b, ID_{ssid}||\text{ ssid}, x_a)$ for $x_a$ and $S$ sends $<c_a, \Delta_a, h_a, (ID_{a1}, ID_{a2}, S), c_b, \Delta_b, h_k, C_s = (C_{a1}, C_{a2}), x_a, \phi_{b1} >$ to user $U_b$.

(iii) After receiving the message sent by $S$, user $U_b$ can recover verification element $\phi_{b1} = a \cdot \phi_s + e_s$, message $m_b' = ID_{ssid}||s||h_bb||ssid, \phi_{b1} || 1 \cdots$, and hash function value $h_b' = \text{Hash}(pk, h_k, C_{a1}, C_{a2}, m_b')$ by using $pw_i$ and salt$_i$ stored locally. Using the correctness of the approximate smooth projection hash function, the user can verify whether $\phi_{b1}$ is correct by $h_b'$. After passing verification, $U_b$ selects $e_4, e_5 \leftarrow \mathcal{R}$ and $sk_b \leftarrow \mathcal{R}$ randomly and uniformly, the coefficients of $e_4$ and $e_5$ obey Gaussian distribution $\chi_{\beta}$, and $sk_b$ is regarded as the temporary private key. Then, $U_b$ calculates $x_b = a \cdot sk_b + 2e_a$, $k_b = x \cdot sk_b + 2e_a$, $\sigma_b = g(k_b)$, and $\rho_b = \text{Extr}(k_b, \sigma_b)$. According to the hash function value $h_b'$, the decoding algorithm ECC$^{-1}$ of the error
RLWE-3VAKE mutual authentication and key agreement phase.

**User** \( U_a(pw_a) \)

**Server** \( S(a, ID_a, \nu) \)

Input: \( ID_a, ID_b, S, pw_a \)

\[ e, e', e \leftarrow \mathbb{R}^n, w, s, k \leftarrow \mathbb{R}_q \]

\[ h_a = \text{HashKG}(1^n) \]

\[ \nu = a \cdot s + e \in \mathbb{R}_q \]

\[ m_a = (ID_a \parallel ID_b \parallel S \parallel h_a \parallel ssid_a \parallel \nu \parallel \cdots) \in \mathbb{Z}_q^* \]

\[ C_{a1} = B_1 \cdot w + e_1(mod q) \]

\[ C_{a2} = B_2 \cdot m_a + e_2(mod q) \]

\[ h_a = \text{Hash}(pk, h_a, C_{a1}, m_a) \]

\[ x_a = a \cdot s + 2e \in \mathbb{R}_q, \phi_a = MAC(h_a, ID_a, ID_b, S|x_a) \]

\[ <ID_a, ID_b, S, h_a, C_a = (C_{a1}, C_{a2}), ssid_a, x_a, \phi_a > \]

- Check \((C_{a1}, C_{a2}), \phi_a\), then select
  - \( \delta_{sa} \leftarrow \mathbb{R}^n, \delta_{sb} \leftarrow \mathbb{R}_q \)
  - \( h_{sb} = \text{HashKG}(1^n), h_{pa} = \text{ProjKG}(pk, h_a), h_a = \text{ProjH}(hp_a, w_a) \)
  - \( m_b = (ID_b \parallel ID_a \parallel S \parallel h_{sb} \parallel ssid_b \parallel \nu_b \parallel \cdots) \in \mathbb{Z}_q^* \)
  - \( C_{b1} = B_1 \cdot w + e_1(mod q), C_{b2} = B_2 \cdot w + m_b + e_2(mod q) \)
  - \( c_b = F_{\nu_b}(1) \oplus F_{\delta_{sb}}(3), c_b = F_{\nu_b}(1) \oplus F_{\delta_{sb}}(3) \)
  - \( \Delta_a = h_a \oplus ECC(\delta_{sa}), \Delta_b = h_b \oplus ECC(\delta_{sb}), \phi_a = MAC(h_a, ID_a, ID_b, S|x_a) \)

\[ <(c_a, \Delta_a, h_a), (ID_a, ID_b, S), c_b, \Delta_b, h_b, C_b = (C_{b1}, C_{b2}), x_b, \phi_b > \]

- Check \( \phi_b \), then select
  - \( e, e \leftarrow \mathbb{R}^n, s, k \leftarrow \mathbb{R}_q \)
  - \( x_b = a \cdot s + 2e \in \mathbb{R}_q, k_b = x_a \cdot s + 2e \)
  - \( \sigma_b = \text{sig}(k_b), \rho_b = \text{Extr}(k_b, \sigma_b) \)
  - \( \delta_{sb} = \text{ECC}^{-1}(h_b \oplus \Delta_b) \)
  - \( SK_{ab} = c_b \oplus F_{\nu_b}(1) \oplus F_{\delta_{sb}}(3) \oplus \sigma_b \oplus \rho_b \)
  - \( \phi_b = MAC(h_b, ID_a, ID_b, S|x_b) \)

\[ <ID_a, ID_b, S, \nu_a, \delta_a, x_a, \sigma_b, \phi_b, ssid_a > \]

\[ hp_a = \text{ProjKG}(pk, h_a) \]

\[ h_a = \text{ProjH}(hp_a, \nu_a) \]

- Check \( \phi_b \), then select
  - \( e \leftarrow \mathbb{R}^n \)
  - \( k_a = x_b \cdot s + 2e \)
  - \( \rho_a = \text{Extr}(k_b, \sigma_b) \)
  - \( \delta_{sa} = \text{ECC}^{-1}(h_a \oplus \Delta_b) \)
  - \( SK_{ab} = c_a \oplus F_{\nu_b}(1) \oplus F_{\delta_{sb}}(3) \oplus \sigma_b \oplus \rho_b \)

**Figure 1:** RLWE-3VAKE mutual authentication and key agreement phase.
correction code and $\Delta_b$. $U_b$ can calculate $\delta_{ab} = ECC^{-1}(h'_b \oplus \Delta_b)$ and generate the session key $SK_{ab} = c_a \cdot F_{\delta_b}(1) \oplus F_{\delta_b}(3) \oplus \sigma_b \oplus p_a$, with the user $U_a$, and the verifiable MAC value $\varphi_b = MAC(h'_a, ID_a || ID_b || x_b)$ is generated by using $h'_a$. Finally, $(c_a, \Delta_b, x_b, (ID_a, ID_b, S), \sigma_b, \varphi_b, ssid_b)$ are sent to user $U_a$.

(iv) $U_a$ first checks $U_b$’s session serial number $ssid_b$ after receiving the message. If it does not meet the requirements, exit. Otherwise, $U_a$ calculates the corresponding projection key $h_p_{ab} = ProjKG(pk, hk_a)$ according to the hash key $hk_a$ selected which is stored locally; then, the projection function value $h'_a = ProjH(h_p_{ab}, w_a)$ is calculated according to the projection key $h_p_{ab}$ and the evidence $w_a$ that can prove the ciphertext. Using the correctness of the approximate smooth projection hash function, the user can verify whether $\varphi_b$ is correct by $h'_a$. If it passes the verification, $U_a$ will select $c_b \in \mathbb{R}$ randomly. According to the projection function $h'_a$, the decoding algorithm $ECC^{-1}$ of the error correction code, and $\Delta_b$, $U_a$ calculates $\delta_{ab} = ECC^{-1}(h'_b \oplus \Delta_b)$, $k_a = y \cdot sk'_a + 2c_a$, and $\rho_a = Extr(k_a, \sigma_a)$. Finally, $U_a$ generates the session key $SK_{ab} = c_a \cdot F_{\delta_b}(1) \oplus F_{\delta_b}(3) \oplus \sigma_b \oplus p_a$ with the user $U_b$.

4.4. Correctness. When users $U_a$, $U_b$, and server $S$ run the protocol honestly and if a valid session key $SK_{ab} = SK_{ba}$ can be generated with overwhelming probability, it is said that a 3PAKE protocol based on verification element is correct.

Taking user $U_a$ as an example, $U_a$ encrypts the verification element corresponding to his password to obtain an effective ciphertext $(C_{a1}, C_{a2})$, the server $S$ uses the ciphertext and hash key sent by the user to calculate the hash function value, and the user uses the projection key to obtain a projection function value. According to the approximate correctness of the function $\varepsilon - ASPH$, the probability that the hamming distance of $h_b$ is $\varepsilon$ greater than $h'_b$ is negligible. The error correction code defined in this paper can correct error $2\varepsilon$. Finally, users $U_a$ and $U_b$ can obtain $\delta_{ab}$ and $\delta_{ba}$ which is same as the server, respectively. Meanwhile, user $U_a$ can use

$$\rho_a = Extr(k_a, \sigma_a) = \left( k_a + \sigma_a \frac{q - 1}{2} \mod q \right) \mod 2$$

$$= \left( k_a + 2\varepsilon + \sigma_a \frac{q - 1}{2} \mod q \right) \mod 2$$

$$= \left( k_a + \sigma_a \frac{q - 1}{2} \mod q \right) \mod 2 + (2\varepsilon) \mod 2$$

$$= \left( k_a + \sigma_a \frac{q - 1}{2} \mod q \right) \mod 2$$

$$= Extr(k_a, \sigma_a) = \rho_b,$$

to calculate $\rho_a$.

5. Security Analysis

A 3PAKE protocol based on the verification element can be widely used on the premise of ensuring its correctness and security. This section proves the security of the protocol under the security model given in Section 3. With that, we prove forward security of the protocol and the privacy of the session key to the server.

5.1. Security Proof

**Theorem 4.** If the public-key encryption system $PKE = (\text{keyGen}, \text{Enc}, \text{Dec})$ based on the ideal lattice is CCA secure $\varepsilon - \text{ASPH}$ function cluster corresponding to the public key system $PKE = (\text{keyGen}, \text{Enc}, \text{Dec})$. ECC: $(0, 1)^n \rightarrow (0, 1)^k$ is the coding algorithm of error correction code. It can correct the $2\varepsilon$ error part through the decoding algorithm $ECC^{-1}$: $(0, 1)^k \rightarrow (0, 1)^n$ corresponding to the coding algorithm ECC. $F = \{F_\delta : \delta \in \{0, 1\}^k\}$ is a pseudo-random function cluster; and then, this protocol is secure.

**Proof.** Assuming that any PPT attacker can make $q_{exe}, q_{send}, q_{rec}, q_{co}$ for Execute, Send, Receive and Corrupt inquiries at most, the total running time is $t$ at most. The advantage of the attacker is simulated by constructing a series of games named $G_0, G_1, \cdots, G_k$. In this series of games, $A$’s advantage of breaking the protocol gradually increases. Finally, as long as the advantage of the attacker’s success in the game $G_k$ is negligible, $\mathcal{A}$ cannot break the protocol. The advantage of $\mathcal{A}$ breaking the protocol is defined as $Adv_{\mathcal{A}}^{\mathcal{PKE}}(\mathcal{A}) = 2 pr[Success_{\mathcal{A}}^{\mathcal{PKE}}(\mathcal{A})] - 1$.

Game $G_0$. This game corresponds to the real attack in the security model. For all kinds of queries sent by the attacker $\mathcal{A}$, $\mathcal{A}$ may get honest answers. At this time, the advantage of the attacker $\mathcal{A}$ is $Adv_{\mathcal{A}}^{\mathcal{PKE}}(\mathcal{A}) = 0$.

**Game $G_1.$** In this game, we modify the simulation model of Execute inquiry and replace the answer generated by the user using $ProjH$ with the corresponding Hash calculation in each Execute inquiry, such as $h'_{U} = Hash(pk, hk_a, C_{a2}, m'_u)$, $u \in \{A, B\}$.

**Proof.** Due to the approximate correctness of $\varepsilon - \text{ASPH}$ and the coding algorithm $E = (0, 1)^n \rightarrow (0, 1)^k$ of the error correction code, $\mathcal{A}$ has the same advantages in the game $G_1$; it can be denoted by $Adv_1(\mathcal{A}) = Adv_{\mathcal{A}}^{\mathcal{PKE}}(\mathcal{A})$.

**Game $G_2.$** During the simulated Execute query, for any user $u \in \{A, B\}$, the only difference between game $G_2$ and game $G_1$ is that the ciphertext $C_a$ in the message sent by the user for the first time is replaced with the ciphertext of encrypting the virtual verification element $VT_e$ (i.e., the verification element that does not belong to the password space $D$). Finally, we can see that the advantage difference of $\mathcal{A}$ between game $G_2$ and game $G_1$ can be ignored. We recorded it as $|Adv_2(\mathcal{A}) - Adv_1(\mathcal{A})| < 2q_{exe} \cdot Adv_{\mathcal{A}}^{\mathcal{PKE}}(t + O(q_{send} + q_{exe}))$. □
Proof. Taking the public key $pk$ as a public parameter, $\mathcal{B}$ is the attacker of the public-key encryption system so that $\mathcal{B}$ can answer the queries of attacker $\mathcal{A}$. $\mathcal{B}$ sends $(v'_u, V_T^u)$ to their challenger as the challenge plaintext and replaces the ciphertext $C_u$ with the received challenge ciphertext $C'_u$ in the Execute query. Then, $\mathcal{A}$ makes their guess $b'$ about the random bit $b$ in the Test query. If $b' = b$, the guess is successful, and $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0. According to the CCA security of the public-key cryptosystem on the ideal lattice, the attacker $\mathcal{B}$ cannot distinguish two ciphertexts with negligible advantage. Therefore, the ciphertext in the $q_{exe}$ Execute query sent by the attacker can be replaced with the encrypted ciphertext of the virtual verification element $VT^u_0$. Considering that the attacker $\mathcal{A}$ only needs additional calculation time during Execute and Send query in the whole process of simulating the protocol and does not need additional calculation time during other queries because it only needs to return the corresponding state, it only needs $t + O(q_{send} + q_{exe})$ at most calculation time in the whole process.

Game $G_1$. In this game, $h_u = \text{Hash}(pk, hk_u, C_{u2}, m'_u)$ and $u \in \{A, B\}$ in the Execute query will be replaced by a randomly selected bit string of equal length. Let $\epsilon_{\text{SAPH}}(n)$ be a negligible upper bound of the statistical distance between the outputs of inputting nonverbal element and the uniform random distribution. Here, $|\text{Adv}_3(\mathcal{A}) - \text{Adv}_2(\mathcal{A})| < 2q_{exe} \cdot \epsilon_{\text{SAPH}}(n)$. □

Proof. It is known that the ciphertexts are replaced by the encryption results of the virtual verification element in game $G_2$, so the input of the approximate smooth projection hash function is a nonverbal element. According to the smoothness of the function, the statistical distance between the outputs of inputting nonverbal element and the uniform random distribution can be ignored.

Game $G_4$. In the Execute query, $(sk'_u, x_u)$ will be replaced by a randomly selected bit string of equal length, where $u \in \{A, B\}$. Here, $\text{Adv}_4(\mathcal{A}) = \text{Adv}_3(\mathcal{A})$. □

Proof. It is known that the ciphertext in game $G_2$ has been replaced by the encryption of the virtual verification element, and $h_u$ will be replaced by a random bit string with an equal length in the game $G_3$. Now, $(x_u, \varphi_u)$ is random. According to the decision $\text{RLWE}_{\text{non-v}}$ problem, the advantages of game $G_4$ and game $G_4$ are the same.

Game $G_5$. Game $G_5$ modifies the pseudorandom function in Execute query, and the others remain unchanged, the same as game $G_4$. $\delta_u$ and $\delta'_u$ are still randomly selected, but the values of $F_{\delta_u}(1)$, $F_{\delta_u}(3)$, $F_{\delta'_u}(1)$, and $F_{\delta'_u}(1)$ are replaced with independent and random numbers. $\text{Adv}_5(\mathcal{A}) = \text{Adv}_4(\mathcal{A})$ holds at this time because the pseudorandom function $F_{\delta_u}$ is hidden for the user.

Proof. The user and the server have the same $\Delta_u$, and according to the correctness of the decoding algorithm $E^{-1} : \{0, 1\}^k \rightarrow \{0, 1\}^n$ of the error correction code, the user will get the same $\delta_u$ as the server.

The above is to modify the Execute query. It can be known from the content of game $G_1$ to game $G_5$ that all messages are replaced with random values independent of the user’s password, and the attacker cannot obtain any information related to the user’s password in the Oracle query so that the communication parties can finally negotiate a completely random session key.

Next, we modify the Send query. $\text{Send}(S, msg_1)$ means sending message $msg_1$ to the server instance $S_j$, and $\text{Send}(U^n, msg_2)$ indicates sending message $msg_2$ to the user instance $U^n$. Only when the server instance $S_j$ receives a correct and valid message $msg_1$, it can return the corresponding valid message $msg_2$ to the attacker. Note that $\text{Send}_i(U^n, S_j)$ represents that the user instance $U^n$ and the server instance $S_j$ are activated and start to execute the protocol. In addition, the simulator $\mathcal{M}$ should record the private key $sk$ corresponding to the public key $pk$ in the key generation phase.

Game $G_6$. For $\text{Send}(S, <ID'^u_1, ID'^u_2, S, hk'_u, C_{u1}', C_{u2}', ssid'_u>)$ query received from the server, where $u \in \{A, B\}$, if there is no corresponding verification element $v'_u$ in the local password list $ID'^u_2$ of the server, the emulator $\mathcal{M}$ rejects the message, or if $ssid'_u$ is smaller than the value of the previous session serial number saved by the server, the message is rejected. Finally, $\mathcal{M}$ checks whether the ciphertext $(C_{u1}', C_{u2}')$ is valid. If not, $\mathcal{M}$ continues to reject the message. Otherwise, the emulator $\mathcal{M}$ decrypts the ciphertext $(C_{u1}', C_{u2}')$ with the corresponding private key $sk$ and gets the user’s verification element. If $v'_u = v'_u$, it is considered that the attacker has broken the protocol and ends the game simulation. The modification of the above game obviously increases the probability of an attacker’s success, so $\text{Adv}_6(\mathcal{A}) \leq \text{Adv}_5(\mathcal{A}) + 2q_{send} \cdot \epsilon_{\text{SAPH}}(n)$ holds. □

Proof. If $(C_{u1}', C_{u2}')$ is a valid ciphertext, the simulator $\mathcal{M}$ can successfully decrypt to obtain $v'_u$. If $v'_u = v'_u$, this situation significantly increases the success advantage of the attacker $\mathcal{A}$. When $v'_u \neq v'_u$, ciphertext $(C_{u1}', C_{u2}') \in X_{\mathcal{I}}$ is obvious. According to the smoothness of $\epsilon - \text{ASPH}$, $v'_u \neq v'_u$ does not increase the advantage of the attacker $\mathcal{A}$. It is found that $\text{Adv}_5(\mathcal{A}) \leq \text{Adv}_6(\mathcal{A}) + 2q_{send} \cdot \epsilon_{\text{SAPH}}(n)$ holds by applying the proof of the game $G_5$.

Game $G_7$. We modify the query received by the user in this game. Let the message $msg_1 = <ID'^u_1, ID'^u_2, S, hk'_u, C_{u1}', C_{u2}', ssid'_u>$ be a valid output of $\text{Send}(U^n, \langle\Delta_u, h_u\rangle)$ for the previous $\text{Send}(U^n, \langle\Delta_u, h_u\rangle)$ query. If $\langle\Delta_u, h_u\rangle$ is a replay of an honest simulated $\text{Send}(S, <ID'^u_1, ID'^u_2, S, hk'_u, C_{u1}', C_{u2}', ssid'_u>)$ query, $h'_u$ will no longer be generated for the user according to the protocol, but force the user to have the same $\delta_u$ as the
server. Because it is hidden from the attacker, the above modifications will not increase the attacker’s advantage, and \( \text{Adv}_V (A) = \text{Adv}_V (A') \) holds.

Game \( G_6 \). The model of the response to the activation message \( \text{Send}_b (U_w, S_i) \) is further modified. If the user is activated, the ciphertext is replaced with the encryption of the virtual verification element \( V_{T_b} \). Now, \( |\text{Adv}_V (A) - \text{Adv}_V (A')| < q_{\text{send}} \cdot \text{Adv}_{\text{PKE}} (t + O (q_{\text{send}} + q_{\text{exe}})) \). 

**Proof.** The same as the proof of game \( G_2 \) because the public-key encryption system is CPA secure, the advantage difference of attacker \( A' \) between this game and game \( G_7 \) can be ignored.

According to the constructed game \( G_8 \), the attacker \( A' \) can successfully break the protocol only if they meet the following conditions:

(i) The attacker forges a legal user to communicate, and the ciphertext is generated by encrypting the verification element \( V_{T_b} \) corresponding to the user password in the message sent to the server.

(ii) The attacker pretends to be the server and sends a valid message \( (\Delta_h, h_s) \) to the user.

(iii) In the Test query, the attacker’s guess \( b' \) of random bit \( b \) satisfies \( b' = b \).

It can be seen from the analysis that the first and second conditions can be satisfied only when \( A' \) can obtain the password of the legal user from the session. However, through the game constructed above, it can be found that the attacker cannot get any information related to the password during the session, even if the user verification element on the server has leaked, the attacker can only guess the user’s password by a dictionary attack. Let the user password space \( D \) in the proposed protocol obey the Zipf principle; \( C \) and \( \phi \) are the parameters of Zipf. Note that \( \mathcal{F} \) represents that the first two conditions are true; then \( \Pr[\mathcal{F}] \leq C \cdot q_{\text{send}}^{\phi} \). At the end of game \( G_8 \), the session key negotiated by the user has been replaced with a completely random value. Here, the probability of a successful guess by the attacker in the Test query is 1/2 at most. Therefore, \( \text{Adv}_V (A') \leq C \cdot q_{\text{send}}^{\phi} \).

To sum up, the advantages of the attacker in the game \( G_9 \) is \( \text{Adv}_V (A) \leq C \cdot q_{\text{send}}^{\phi} + 2 (q_{\text{send}} + q_{\text{exe}}) \cdot \text{Adv}_{\text{PKE}} (t + O (q_{\text{send}} + q_{\text{exe}}) + \epsilon_{\text{SAPF}} (n)) \). Meanwhile, \( \text{Adv}_V (A') = \text{Adv}_V (A') \); it can see that the attacker’s advantage is only a negligible difference from the dictionary attack advantage. Therefore, the proposed 3PAKE protocol based on the verification element is secure. □

5.2. Forward Security

**Theorem 5.** When the search RLWE difficult assumption is true, if the attacker still cannot use the known private key to interact with the participants to obtain the correct session key after obtaining the long-term private key of each participant in the protocol, the protocol has forward security.

**Proof.** In this protocol, the session key among users needs to be generated with the help of the server. The complete session key includes two parts: one is determined by the server and the other is determined by users. When the attacker has the long-term private key of the server, they can authenticate with the user and server by forging ciphertext and signature. Now, they can obtain part of the session key calculated by the server. The session key determined by the user is also related to the temporary key selected by the user for this communication. The temporary key \( s_{\text{K}} \) is not transmitted on the channel, and the attacker can only obtain \( x_0 = a \cdot s_{\text{K}} + 2e \) transmitted on the channel. On the premise that the attacker knows \( (a, x_0) \), the attacker needs to solve the search RLWE problem to obtain the temporary private key \( s_{\text{K}} \).

5.3. Privacy of Session Key

**Theorem 6.** When the search RLWE difficulty assumption holds, the session key negotiated by the user using our protocol is private to the server. In other words, it is inoperable when the honest server wants to recover the user’s session key because of curiosity.

**Proof.** Firstly, we construct an adversary \( A_{\text{RLWE}} \) against the search RLWE problem and let adversary \( A_{\text{privacy}} \) destroy session key privacy. It is assumed that adversary \( A_{\text{privacy}} \) can have \( q_{\text{exe}} + q_{\text{send}} \) times \( \text{Execute} \) and \( \text{Send} \) queries when launching the session key attack, and the total running time is \( t \) at most. In the process of an attack, a random coin \( b \in \{0, 1\} \) is thrown by \( A_{\text{RLWE}} \) to get a triplet \( (x_0, C_b, \phi_b) \). When \( b = 0 \), the triplet is the real RLWE generated by the running protocol. When \( b = 1 \), the triplet is random. Right now, adversary \( A_{\text{RLWE}} \) simulates protocol stipulates, honestly runs, and answers the \( \text{Execute} \) and \( \text{Send} \) query of adversary \( A_{\text{privacy}} \); but when responding to the query, \( A_{\text{RLWE}} \) replaces the relevant contents \( (x, C, \phi) \) with \((x_0, C_b, \phi_b)\) in the real protocol.

Finally, \( A_{\text{privacy}} \) outputs the guess \( b' \) according to the response. If \( b' = b \), \( A_{\text{RLWE}} \) outputs 1, the triplet \( (x_0, C_b, \phi_b) \) is a real RLWE, and the probability of \( A_{\text{RLWE}} \) guessing successfully can be denoted by \( \Pr[\text{RLWE wins}] = \Pr[b' = b | b = 1] \). If \( b' \neq b \), \( A_{\text{RLWE}} \) outputs 0, the triplet \( (x_0, C_b, \phi_b) \) is random, and the probability of \( A_{\text{RLWE}} \) guessing successfully can be expressed as below.

\[
\Pr[\text{RLWE wins}] = \Pr[b' \neq b | b = 0]
\]

Therefore, the probability \( A_{\text{RLWE}} \) solving the search RLWE problem can be calculated as

\[
\Pr[\text{RLWE wins}] = \frac{1}{2} (\Pr[b' = b | b = 1] + \Pr[b' \neq b | b = 0])
\]

\[
= \frac{1}{2} + \frac{1}{2} \cdot \text{Adv}_{\text{privacy}} (A_{\text{privacy}}) \cdot \left( 1 - \frac{1}{2} \right)
\]

\[
= \frac{1}{2} + \frac{1}{4} \cdot \text{Adv}_{\text{privacy}} (A_{\text{privacy}})
\]

□
6. Performance Analysis

This section analyzes the security and efficiency of the protocol. Table 1 lists the security comparison results of our protocol with references [2, 14, 17, 23]. A/B/S in Table 1 represents the number of messages to be sent by user A, user B, and server S, respectively. Table 2 shows the analysis results of the calculation overheads and communication overheads of our protocol.

### 6.1. Security Comparison

By analyzing the protocol of Guo et al. [29], Liu and Xue [2] found that the adversary in [29] can obtain the trust of the server by relaying the messages of other legitimate users. If an adversary uses this loophole to launch a DoS attack, it will consume the resources of the server and cause legitimate users to be unable to access the service. To solve this problem, Liu and Xue’s protocol introduces a timestamp and reduces a lot of unnecessary communication overheads; they reduce the six rounds of communication in the existing protocol to three and effectively solve the server spoofing attack and offline dictionary attack. However, their protocol is a 2PAKE protocol. With the emergence of large-scale end-to-end communication, users will communicate with each other frequently. 2PAKE protocol makes users need to store a large number of passwords for key agreement. Note that the 2PAKE protocol does not need a server to negotiate, so it is impossible to discuss whether the session key is private.

The protocol of Yu et al. [17] introduces the session sequence number to resist the replay attack. The number of communication rounds is reduced to two rounds by using the separable cryptosystem and the smooth hash projection function. Since the password is stored in plaintext on the server, it cannot resist the server disclosure attack. In addition, the honest and curious server can recover the session key between users, and their protocol cannot resist the server’s internal attack.

The protocols in [14, 23] are a 3PAKE protocol based on a verification element, effectively resisting the server disclosure attack. The protocol of Zhang et al. is based on the DDH assumption and cannot be against the quantum algorithm attack [14]. Although the session key can be negotiated only by four rounds of communication, it cannot resist the replay attack and cannot meet the security requirements in practical applications. Shu et al. [23] provided quantum level security, introduced the session sequence number to resist replay attack, and realized the privacy of the session key to the server, but they increased the communication overheads. It needs seven rounds of communication and sends nine massages to make a session key agreement.

Based on protecting the user password, our protocol can also resist replay attacks. The proposed protocol is constructed according to the RLWE problem that improves security and reduces the storage space of the key. We use FFT to accelerate the operation speed, effectively lower down the time complexity, and introduce a message authentication mechanism to solve the problem of session key disclosure caused by the dishonesty of the server. In general, this protocol effectively resists the server disclosure attack and has stronger security.

### 6.2. Efficiency Analysis

Our protocol adopts an asymmetric model. User A initiating the establishment of the session key needs to perform two calculations: the first calculation is used to generate the information related to the session request, and the session key negotiated with user B is generated in the second calculation. We analyze the efficiency of the proposed protocol on Windows 10 system, 11th Gen. Intel (R) core (TM) i5-1135g7 @ 2.40GHz processor and 16.0 GB running memory, and the computational complexity of each stage is the average value of running 10000 calculations. Table 2 lists the specific computation overhead of each communication stage and the total communication overhead of the protocol.

Set the parameters $n = 2^k$, $k \geq 2$, $q = n^3 - 1$, $m = 6 \log q$; now our protocol is safe, where $k$ and $m$ are integer. It can be seen from Table 2 that with the increase of $n$, the computation overhead of each stage also increases. Note that the computation overhead of user A for the first round of

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Type</th>
<th>Difficulty problem</th>
<th>Verification element</th>
<th>Replay attack</th>
<th>Privacy of session key</th>
<th>A/B/S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liu et al. [2]</td>
<td>2-party</td>
<td>Chebyshev chaotic map</td>
<td>No</td>
<td>Yes</td>
<td>Null</td>
<td>2/1/1</td>
</tr>
<tr>
<td>Zhang et al. [14]</td>
<td>3-party</td>
<td>DDH assumption</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>2/2/4</td>
</tr>
<tr>
<td>Yu et al. [17]</td>
<td>3-party</td>
<td>LWE</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>1/1/2</td>
</tr>
<tr>
<td>Shu et al. [23]</td>
<td>3-party</td>
<td>RLWE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>3/4/2</td>
</tr>
<tr>
<td>Ours</td>
<td>3-party</td>
<td>RLWE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>1/1/1</td>
</tr>
</tbody>
</table>

#### Table 1: Security comparison.

<table>
<thead>
<tr>
<th>Stage</th>
<th>n = 32</th>
<th>n = 64</th>
<th>n = 128</th>
<th>n = 256</th>
<th>n = 512</th>
</tr>
</thead>
<tbody>
<tr>
<td>First computation cost of user A</td>
<td>1.96 ms</td>
<td>3.97 ms</td>
<td>9.61 ms</td>
<td>22.82 ms</td>
<td>78.91 ms</td>
</tr>
<tr>
<td>Server’s computation cost</td>
<td>20.52 ms</td>
<td>63.10 ms</td>
<td>353.49 ms</td>
<td>3373.06 ms</td>
<td>45802.87 ms</td>
</tr>
<tr>
<td>User B’s computation cost</td>
<td>0.38 ms</td>
<td>0.63 ms</td>
<td>1.07 ms</td>
<td>3.39 ms</td>
<td>4.62 ms</td>
</tr>
<tr>
<td>Second computation cost of user A</td>
<td>0.31 ms</td>
<td>0.47 ms</td>
<td>0.77 ms</td>
<td>1.49 ms</td>
<td>3.20 ms</td>
</tr>
<tr>
<td>Total communication cost</td>
<td>7 KB</td>
<td>13 KB</td>
<td>24 KB</td>
<td>48 KB</td>
<td>79 KB</td>
</tr>
</tbody>
</table>

#### Table 2: Computation and communication cost of the proposed protocol.
communication and server, relatively speaking, is large. User A needs to generate ciphertext, message authentication code, and other information during the first round of communication; these operations are relatively complex. The server needs to verify the validity of the ciphertext and message authentication code of user A, and generate a valid ciphertext for user B. When verifying the validity of ciphertext, the server needs to recover the corresponding plaintext message using the Exhaustive Method. In the experiment, we assume that plaintext can be decrypted and recovered in the worst case; that is to say, the server needs to run \( q \) times to decrypt successfully, and the value of \( q \) will increase rapidly with the increase of \( n \). Therefore, the growth rate of computation overhead at this stage is high. And the computation overhead of the server will be optimized in the real operation process.

When \( n = 128 \) (\( n = 256 \) or \( n = 512 \)), the total communication overhead is 29 KB (56 KB or 95 KB). It can be seen that the computation and communication overhead of each stage is low when \( n = 128, n = 256, or n = 512 \); it can resist quantum algorithm attacks, and achieve the required security level, so the proposed protocol can be effectively applied to the large-scale communication networks.

7. Conclusion

Using approximate smooth projection hash function technology and message authentication mechanism on the ideal lattice, we construct a more efficient 3PAKE protocol based on the verification element. Compared with the existing verification element-based 3PAKE protocol, our protocol reduces space complexity and improves computation and communication efficiency. It only needs two rounds of communication to correctly negotiate a session key. Furthermore, it can be against server password disclosure attack, server internal attack, and replay attack; we give the semantic security proof of the new protocol. In short, the proposed protocol has high security and low overhead, which can meet the communication requirements of large-scale low bandwidth networks.

Data Availability

The experimental results are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Fund (no. 61802117), Shaanxi Key Laboratory of Information Communication Network and Security, Xi’an University of Posts and Telecommunications, Xi’an, Shaanxi, China (no. ICNS202006), Support Plan of Scientific and Technological Innovation Team in Universities of Henan Province (no. 20IRTSTHN013), Fundamental Research Funds for the Universities of Henan Province (no. NSFRF210312), and Youth Talent Support Program of Henan Association for Science and Technology (no. 2021HYTP008).

References


