Hindawi
Security and Communication Networks
Volume 2021, Article ID 6952869, 13 pages
https://doi.org/10.1155/2021/6952869

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

Academic Editor: Zhe-Li Liu

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; it 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. The 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 see 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 $q \equiv 1 \bmod 2n$. Finally, our protocol can obtain the optimal implementation in the domain $\mathbb{Z}_q$, 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 $B = \{b_1, b_2, \cdots, b_m\} \in \mathbf{R}^{n \times m}$ be a set of linearly independent vectors in $\mathbf{R}^n$, and n and m are positive integers. The lattice generated by B is defined as

$\mathscr{L}(\mathrm{B}) = \{\sum_{i=1}^{m} x_i b_i : x_i \in \mathbb{Z}^m\}$, and B is the basis of $\mathscr{L}(\mathrm{B})$. 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 $a = (a_0, a_1, \cdots, a_{n-1})^T$ is written as $\mathrm{rot}(a) = (a_{n-1}, a_0, a_1, \cdots, , a_{n-1})^T$. For lattice $\mathscr{L}$, if $\forall a \in \mathscr{L}$, $\mathrm{rot}(a) \in \mathscr{L}$, then the lattice $\mathscr{L}$ is called a cyclic lattice, record as $Rot(a) = \{a, rot(a), \cdots, rot^{n-1}(a)\}$, and expressed as

$$Rot(a) = \begin{bmatrix} a_0 & a_{n-1} & \cdots & a_1 \\ a_1 & a_0 & \cdots & a_2 \\ \cdots & \cdots & \cdots & \cdots \\ a_{n-1} & a_{n-2} & \cdots & a_0 \end{bmatrix}. \tag{1}$$

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 $n = 2^k (k > 1)$, $q \equiv 1 (\mathrm{mod}\, 2n)$, $f(x) = x^n + 1 \in \mathbb{Z}[x]$, and $f(x)$ is irreducible on the set of all rational numbers. $\mathbf{R} = \mathbb{Z}[x]/f(x)$ is a ring of the integer polynomial $f(x)$; then, the elements in $\mathbf{R}$ are usually represented by integer polynomials of less degree n. If vector $a \in \mathbf{R}$, the cyclic lattice $\mathscr{L}(Rot(a))$ generated by a is an ideal of ring $\mathbb{Z}[x]/(x^n - 1)$. $\mathbf{R}_q = \mathbb{Z}_q[x]/f(x)$ 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 $\{0, 1, \cdots, q - 1\}$. The ideal lattice lows down the space complexity by using a vector to represent an n dimensional lattice.

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

Definition 5 (decision $RLWE_{n,m,q,\chi_\beta}$ problem). Given m independent uniformly distributed samples from $\mathbf{R}_q \times \mathbf{R}_q$, there is no arbitrary probabilistic polynomial time (PPT) algorithm to distinguish whether the sample is chosen from RLWE distribution or uniform and random $\mathbf{R}_q \times \mathbf{R}_q$.

Definition 6 (search $RLWE_{n,m,q,\chi_\beta}$ problem). Randomly generate polynomial $a \in \mathbf{R}_q, s \in \mathbf{R}_q$, and $e \longleftarrow \chi_\beta$; they satisfy $b_i = a_i s + e_i$. Now, given multiple groups (a_i, b_i), it is needed to find s.

Theorem 1 (see [25]). *We suppose $n = 2^k (k > 1)$, $q \equiv 1 (\mathrm{mod}\, 2n)$, $\beta \in (0, 1)$; they satisfy $\beta q \geq w(\sqrt{\log n})$. Then, the $\widetilde{O}(\sqrt{n}/\beta)$-approximate SVP (shortest vector problem) can be reduced to $R - DLEW_{n,m,q,\chi_\beta}$ problem and $R - SLEW_{n,m,q,\chi_\beta}$ 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 $c \in \mathbf{R}$ as the center and σ as the standard deviation. For any $\sigma > 0$, with taking $c \in \mathbf{R}^m$ as the center and σ as the parameter, the discrete Gaussian distribution over lattice $\mathscr{L}$ is defined as $D_{\mathscr{L},\sigma,c}(y) = (\rho_{\sigma,c}(y)/\rho_{\sigma,c}(\mathscr{L}))$, where $y \in \mathscr{L}, \rho_{\sigma,c}(\mathscr{L}) = \sum_{x \in \mathscr{L}} \rho_{\sigma,c}(x)$. Note that c can be ignored not writing when c is 0.

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

$$\begin{aligned} &\Pr\left[x \longleftarrow D_\sigma^m : \|x\| > 2\sigma\sqrt{m}\right] < 2^{-m}, \\ &\Pr\left[x \longleftarrow D_\sigma^1 : \|x\| > \omega\left(\sigma\sqrt{\log m}\right)\right] = 2^{-\omega \log m}. \end{aligned} \tag{2}$$

2.3. Public-Key Cryptosystem Based on Ideal Lattice. Let n and m be positive integers; the security parameter is n; and q is an odd prime and satisfies $q \geq n^{2.5}$, $q \equiv 1 (\mathrm{mod}\, 2n)$, $m \geq 6 \log q$. A public-key cryptosystem, whose difficulty is based on the RLWE problem, is mainly composed of three algorithms.

(i) $(pk, sk) \longleftarrow \mathrm{KeyGen}(1^n)(1/2)$: this is a key generation algorithm, which inputs a security parameter n and outputs the public-private key pair (pk, sk) of the system

(ii) $(c_1, c_2) \longleftarrow \mathrm{Enc}(pk, msg)$: this is called an encapsulation algorithm. It takes as input a public key pk and a plaintext msg and outputs $c = (c_1, c_2)$

(iii) $msg \longleftarrow \mathrm{Dec}(sk, (c_1, c_2))$: 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 "$\perp$"

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 $PK\varepsilon = (KeyGen, Enc, Dec)$, and it is a semantically secure public-key encryption system composed of functions on the lattice. C_{pk} 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:

$$X = \{(c, m) \mid c \in C_{pk}; m \in \mathrm{P}\},$$
$$\overline{L}_m = \{(c, m) \in X \mid c = \mathrm{Enc}_{pk}(m, r), r \in \{0,1\}^*\}, \overline{L} = \cup_{m \in \mathrm{P}} \overline{L}_m,$$
$$L_m = \{(c, m) \in X \mid m = \mathrm{Dec}_{sk}(c)\}, L = \cup_{m \in \mathrm{P}} L_m. \tag{3}$$

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 $\varepsilon - ASPH = (HashKG, ProjKG, Hash, ProjH)$.

(i) $HashKG(1^n)$: 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.

(ii) $ProjKG(hk, pk)$: 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.

(iii) $Hash(hk, L, c)$: When it inputs the hash key hk, language $\overline{L}$, and any word $c \in \overline{L}$, the hash function outputs the hash value h.

(iv) $ProjH(hp, w)$: This is a projection hash function. Let the projection key hp and evidence w of word $c \in \overline{L}$ be the input, and it outputs the projection hash value h'.

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

Smoothness: for $\forall c \notin \overline{L}$, $hp = ProjKG(hk, L, c)$, $p \longleftarrow_r \{0,1\}$, the distributions of $(hp, Hash(hk, L, c))$ and (hp, p) are indistinguishable in the statistical distance. When n is the security parameter, $\varepsilon_{SASPH}(n)$ 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

(i) $(pk, sk) \longleftarrow \mathrm{KeyGen}(1^n)$: Input a security parameter n ; then, it selects $B_0 \longleftarrow_r \mathbf{R}$and runs the trapdoor function to get $(B_1, T_1) \longleftarrow ideal - trapGen$ and $(B_2, T_2) \longleftarrow ideal - trapGen$, where $B_0, B_1, B_2 \in \mathbf{R}_q^m$, $T_1, T_2 \in \mathbf{R}^{m \times m}$. It finally outputs a public/private key pair (pk, sk) of the system;

(ii) $(c_1, c_2) \longleftarrow \mathrm{Enc}(pk, msg)$: Let $pk = (B_0, B_1, B_2)$ and a plaintext $m \in \mathbb{Z}_q^n$ be input; then, it selects $r \longleftarrow_r \mathbf{R}$, $e_1, e_2 \longleftarrow_r \mathbf{R}$, the coefficients of e_1 and e_2 obey distribution χ_β. It outputs $c = (c_1, c_2)$, where $c_1 = B_1 \cdot w_a + e_1 (\mathrm{mod} q)$, $c_2 = B_0 \cdot w_a + B_2 \cdot m + e_2 (\mathrm{mod} q)$.

(iii) $msg \longleftarrow \mathrm{Dec}(sk, (c_1, c_2))$: The private key sk and ciphertext (c_1, c_2) are input; this decapsulation algorithm outputs the corresponding plaintext m or "$\perp$".

2.5.2. Approximate Smooth Projective Hash Function on Ideal Lattice. (1) Hash key: The hash key space is $HK = (\mathbf{R}_q^m)^n$ 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 χ_β for any $(e_1, e_2, \cdots, e_n) \in HK$.

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

$$u_j = \left(Map_{M-v}\left(e_j\right)\right)^T \cdot \widehat{B}_0 (j \leq n). \tag{4}$$

$Map_{M-v}(e_j)$ is the result of connecting the coefficients of polynomial $e_j \in \mathbf{R}_q^m$, and it finally outputs a one-dimensional column vector composed of coefficients e_j. After performing this type of operations on all $e_j \in (e_1, e_2, \cdots, e_n)$, the result of transpose operation will be point multiplied by $\widehat{B}_0$. $B_0 \longleftarrow_r \mathbf{R}$ is a public parameter and $\widehat{B}_0$ is generated by the following calculations:

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

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

$z_j = (Map_{M-v}(e_j))^T \cdot Map_{M-v}(c_2 - B_2 \cdot m) \in \mathbb{Z}_q$, where $m \in \mathbb{Z}_q^n$.

Outputs:

$$b'_j = \begin{cases} 0, if\ z'_j < \dfrac{(q-1)}{2} \\ \\ 1, if\ z'_j > \dfrac{(q-1)}{2} \end{cases} \tag{6}$$

(4) Projection function $\{ProjH_{hp}\}_{hp \in HP}$: let the projection key $hp = (u_1, u_2, \cdots, u_n)$ and the evidence w of $x \in \overline{L}$ be input, and run the following calculations:

$$z_j' = u_j w \in \mathbb{Z}_q. \tag{7}$$

Outputs:

$$b_j' = \begin{cases} 0, if\ z_j' < \dfrac{(q-1)}{2} \\ \\ 1, if\ z_j' > \dfrac{(q-1)}{2}. \end{cases} \tag{8}$$

Theorem 3 (see [18]). *If the parameters m, n, q, β, ω satisfy $\sqrt{q} \cdot w(\sqrt{\log mn}) < r < \varepsilon/(8 \cdot mn^2\beta)$, then the above structure $\mathcal{H} = \{H_{hk}\}_{hk \in HK}$ is a ε − ASPH function.*

2.5.3. MAC Based on Key Hash Function. Message authentication mechanism can verify the identity of information source and integrity of data. Our protocol uses the MAC technology to authenticate information. The MAC based on the key hash function takes as input a key and a message and outputs an information summary by using the hash algorithm. By verifying the correctness of the information summary, the receiver can realize the identity authentication of the information source and the integrity authentication of the message.

Using the correctness of approximate smooth projection hash function, the hash function value and projection function value between the user and the server can be input as a key; then, an information summary is generated by SHA-256. The specific structure is as follows:

$$\mathrm{P} = mac(key, x) = a \cdot key + e + h(x)(q-1)/2 \,(\mathrm{mod} q). \tag{9}$$

After receiving the MAC value, the receiver uses the key key^* to generate a new information summary:

$$\mathrm{P}^* = mac(key^*, x) = a \cdot key^* + e^* + h(x)(q-1)/2 \,(\mathrm{mod} q). \tag{10}$$

Verify $|\mathrm{P}^* - \mathrm{P}| \le \chi_\beta$; if it holds, it proves that the information is indeed sent by the claimant, is complete, and has not been modified.

3. Security Model

3.1. Parties of the Protocol. 3PAKE protocol participants include users and servers. $\mathcal{U}$ represents a user collection, $C \in \mathcal{U}$ is an honest user, and $\mathcal{V} \in \mathcal{U}$ is a malicious user. $\mathcal{S}$ represents a server collection; we usually assume that the server collection has only one element, that is, $\mathcal{S} = \{S\}$.

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 $pw_S = < pw_S[U] >_{U \in \mathcal{U}}$ 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 $\mathcal{A}$ which knows all the malicious user's password set $pw_{\mathcal{A}} = < pw_\varepsilon >_{\varepsilon \in V}$ and also can control the communication channel among all users. Adversary can obtain the specific abilities by sending the following queries.

(i) $Execute(U_1^a, S_j, U_2^b)$: This query gives $\mathcal{A}$ the ability to wiretap channels. After activating instances U_1^a, S_j, and U_2^b, $\mathcal{A}$ will get all the information transmitted between the user and the server.

(ii) $Send(U^u, msg)$: This query simulates $\mathcal{A}$'s active attack on the user instance. The adversary $\mathcal{A}$ selects the message msg and sends it to the user instance U^u. Finally, $\mathcal{A}$ obtains the reply of U^u to the message msg.

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

(iv) $Reveal(U^a)$: This query simulates session key is lost or leaked. After the adversary $\mathcal{A}$ executes this query, $\mathcal{A}$ can obtain all session keys sk_U^a of the instance U^a.

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

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

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

3.4. Accepted State. $ssid_i$ represents the session serial number of the user instance U^i and pid_U^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_1^a and U_2^b all have been accepted; (2) U_1^a and U_2^b have the same session serial number $ssid$; and (3) U_1^a and U_2^b are mutually intended

communicators, the instances U_1^a and U_2^b are a matching session.

3.6. Fresh session. If the instance U^i has been accepted and the adversary $\mathcal{A}$ has not asked a Re*veal* 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*, Re*veal*, 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 $b' = b$ holds, 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 $Adv_{P,D}^{ake}(\mathcal{A}) = 2pr[Succ_{P,D}^{ake}(\mathcal{A})] - 1$, where $Succ_{P,D}^{ake}(\mathcal{A})$ 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 $(pk, sk) \longleftarrow \mathrm{KeyGen}(1^n)$: it takes as input a secure parameter n, then selects $B_0 \longleftarrow_r \mathbf{R}$, and runs trapdoor generation algorithm to get $(B_1, T_1) \longleftarrow ideal - trapGen$ and $(B_2, T_2) \longleftarrow ideal - trapGen$. 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 $Param = \{B_0, B_1, B_2\}$ 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 $salt_i$ randomly, and uses SHA − 256 to generate seeds of two pseudo-random number generators (PRNG): one is seed1 = SHA − 256(salt$_i$||SHA − 256($ID_i||pw_i$)) 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 χ_β, note that s_i and e_i are polynomials whose coefficients obey χ_β, choose $a \longleftarrow_r \mathbf{R}$, calculate the verification element $v_i = a \cdot s_i + e_i \in \mathbf{R}_q$ 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 $\mathcal{Q}$, (a, ID_i, v_i) will be automatically added to $\mathcal{Q}$. Otherwise, it will send a new registration message to the user. After the user U_i successfully executes the registration phase, $(seed_1, seed_2, s_i, e_i, v_i)$ needs to be deleted from the local memory and $(pw_i, salt_i)$ 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_a 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 φ_a. If they are all valid, the server S selects $\delta_{sa} \longleftarrow_r \{0,1\}$, $\delta_{sb} \longleftarrow_r \{0,1\}$, $e_4, e_5 \longleftarrow_r \mathbf{R}$, $w_b \longleftarrow_r \mathbf{R}$ randomly, and e_4 and e_5 are polynomials whose coefficients obey discrete distribution χ_β. The hash key $hk_b = HashKG(1^n)$ of user U_b is randomly selected, and S calculates the projection key $hp_b = \mathrm{Pro}jKG(pk, hk_b)$, projection function value $h_b = \mathrm{Pro}jH(hp_b, w_b)$ and $m_b = ID_b||ID_a||S||hk_b||ssid_b||v_b||1\cdots$ and gets ciphertext $C_{b1} = B_1 \cdot w_b + e_4 (\mathrm{mod} q)$ and $C_{b2} = B_0 \cdot w_b + B_2 \cdot m_b + e_5 (\mathrm{mod} q)$ for massage m_b. Next, S uses random values δ_{sa} and δ_{sb} to calculate $c_a = F_{\delta_{sb}}(1) \oplus F_{\delta_{sa}}(3)$, $c_b = F_{\delta_{sa}}(1) \oplus F_{\delta b}(3)$, $\Delta_a = h_a \oplus \mathrm{ECC}(\delta_{sa})$, $\Delta_b = h_b \oplus \mathrm{ECC}(\delta_{sb})$. Finally, the projection function value h_b is used to regenerate the verifiable MAC value $\varphi_a' = MAC(h_b, ID_a \| ID_b \| S \| x_a)$ for x_a and S sends $<(c_a, \Delta_a, h_a), (ID_a, ID_b, S), c_b, \Delta_b, hk_s, C_s = (C_{s1}, C_{s2}), x_a, \varphi_a'>$ to user U_b.

(iii) After receiving the message sent by S, user U_b can recover verification element $v_b' = a \cdot s_b + e_b$, message $m_b' = ID_b \| ID_a \| S \| hk_b \| ssid_b \| v_b' \| 1 \cdots$, and hash function value $h_b' = Hash(pk, hk_b, C_{b1}, m_b')$ by using pw_b and $salt_b$ stored locally. Using the correctness of the approximate smooth projection hash function, the user can verify whether φ_a' is correct by h_b'. After passing verification, U_b selects $e_6, e_7 \longleftarrow_r \mathbf{R}$ and $sk_b' \longleftarrow_r \mathbf{R}$ randomly and uniformly, the coefficients of e_6 and e_7 obey Gaussian distribution χ_β, and sk_b' is regarded as the temporary private key. Then, U_b calculates $x_b = a \cdot sk_b' + 2e_6$, $k_b = x \cdot sk_b' + 2e_7$, $\sigma_b = g(k_b)$, and $\rho_b = Extr(k_b, \sigma_b)$. According to the hash function value h_b', the decoding algorithm ECC^{-1} of the error

User $U_a(pw_a)$ Server $S(a, ID_i, v_i)$

User $U_b(pw_b)$

Input ID_a, ID_b, S, pw_a

$e_1, e_2, e_3 \leftarrow_r \boldsymbol{R}^m, w_a, sk_a^{'} \leftarrow_r \boldsymbol{R}_q$

$hk_a = HashKG(1^n)$

$v_a^{'} = a \cdot s_a + e_a \in \boldsymbol{R}_q$

$m_a = (ID_a \,||\, ID_b \,||\, S \,||\, hk_a \,||\, ssid_a \,||\, v_a^{'} || 1 \cdots) \in \mathbf{Z}_q^n$

$C_{a1} = B_1 \cdot w_a + e_1 (\mathrm{mod}\, q)$

$C_{a2} = B_0 \cdot w_a + B_2 \cdot m_a + e_2 (\mathrm{mod}\, q)$

$h_a = Hash(pk, hk_a, C_{a1}, m_a)$

$x_a = a \cdot sk_a^{'} + 2e_3 \in \boldsymbol{R}_q, \varphi_a = MAC(h_a, ID_a || ID_b || S || x_a)$

$< ID_a, ID_b, S, hk_a, C_a = (C_{a1}, C_{a2}), ssid_a, x_a, \varphi_a >$ →

check $(C_{a1}, C_{a2}), \varphi_a$, then select

$\delta_{sa} \leftarrow_r \{0,1\}^\mu, \delta_{sb} \leftarrow_r \{0,1\}^\mu, e_4, e_5 \leftarrow_r \boldsymbol{R}^m, \; w_b \leftarrow_r \boldsymbol{R}_q$

$hk_b = HashKG(1^n), \; hp_b = \mathrm{Pr}ojKG(pk, hk_b), h_b = \mathrm{Pr}ojH(hp_b, w_b)$

$m_b = (ID_b \,||\, ID_a \,||\, S \,||\, hk_b \,||\, ssid_b \,||\, v_b || 1 \cdots) \in \mathbf{Z}_q^n$

$C_{b1} = B_1 \cdot w_b + e_4 (\mathrm{mod}\, q), C_{b2} = B_0 \cdot w_b + B_2 \cdot m_b + e_5 (\mathrm{mod}\, q)$

$c_a = F_{\delta_{sb}}(1) \oplus F_{\delta_{sa}}(3), c_b = F_{\delta_{sa}}(1) \oplus F_{\delta b}(3)$

$\Delta_a = h_a \oplus \mathrm{ECC}(\delta_{sa}) \quad , \quad \Delta_b = h_b \oplus \mathrm{ECC}(\delta_{sb}) \quad , \quad \varphi_a^{'} = MAC(h_b, ID_a || ID_b || S || x_a)$

$< (c_a, \Delta_a, h_a), (ID_a, ID_b, S), c_b, \Delta_b, hk_b, C_b = (C_{b1}, C_{b2}), x_a, \varphi_a^{'} >$ →

$v_b^{'} = a \cdot s_b + e_b$

$m_b^{'} = (ID_b \,||\, ID_a \,||\, S \,||\, hk_b \,||\, ssid_b \,||\, v_b^{'} || 1 \cdots) \in \mathbf{Z}_q^n$

$h_b^{'} = Hash(pk, hk_b, C_{b1}, m_b^{'})$, check $\varphi_a^{'}$

then select $e_6, e_7 \leftarrow_r \boldsymbol{R}^m, \; sk_b^{'} \leftarrow_r \boldsymbol{R}_q$

$x_b = a \cdot sk_b^{'} + 2e_6 \in \boldsymbol{R}_q, k_b = x_a \cdot sk_b^{'} + 2e_7$

$\sigma_b = \mathrm{si}\, g(k_b), \rho_b = Extr(k_b, \sigma_b)$

$\delta_{sb} = \mathrm{ECC}^{-1}(h_b^{'} \oplus \Delta_b)$

$SK_{ba} = c_b \oplus F_{\delta_{sb}}(1) \oplus F_{\delta_{sb}}(3) \oplus \sigma_b \oplus \rho_b$

$\varphi_b = MAC(h_a, ID_a || ID_b || S || x_b)$

$< (ID_a, ID_b, S), c_a, \Delta_a, x_b, \sigma_b, \varphi_b, ssid_b >$ ←

$hp_a = \mathrm{Pr}ojKG(pk, hk_a)$

$h_a^{'} = \mathrm{Pr}ojH(hp_a, w_a)$

check φ_b, then select $e_8 \leftarrow_r \boldsymbol{R}^m$

$k_a = x_b \cdot sk_a^{'} + 2e_8$

$\rho_a = Extr(k_a, \sigma_b)$

$\delta_{sa} = \mathrm{ECC}^{-1}(h_a^{'} \oplus \Delta_a)$

$SK_{ab} = c_a \oplus F_{\delta_{sa}}(1) \oplus F_{\delta_{sa}}(3) \oplus \sigma_b \oplus \rho_a$

Figure 1: RLWE-3VAKE mutual authentication and key agreement phase.

correction code and Δ_b, $U_b$can calculate $\delta_{sb} = \mathrm{ECC}^{-1}(h_b' \oplus \Delta_b)$ and generate the session key $SK_{ba} = c_b \oplus F_{\delta_{sb}}(1) \oplus F_{\delta_{sb}}(3) \oplus \sigma_b \oplus \rho_b$, with the user U_a, and the verifiable MAC value $\varphi_b = MAC(h_a, ID_a \| ID_b \| S \| x_b)$ is generated by using h_a. Finally, $\langle (c_a, \Delta_a, x_b), (ID_a, ID_b, S), \sigma_b, \varphi_b, ssid_b \rangle$ 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 $hp_a = \mathrm{Pro}jKG(pk, hk_a)$ according to the hash key hk_a selected which is stored locally; then, the projection function value $h_a' = \mathrm{Pro}jH(hp_a, w_a)$ is calculated according to the projection key hp_a 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 φ_b is correct by h_a'. If it passes the verification, U_a will select $e_8 \longleftarrow_r \mathbf{R}$ randomly. And according to the projection function h_a', the decoding algorithm ECC^{-1} of the error correction code, and Δ_a, U_a calculates $\delta_{sa} = \mathrm{ECC}^{-1}(h_a' \oplus \Delta_a)$, $k_a = y \cdot sk_a' + 2e_8$, and $\rho_a = Extr(k_a, \sigma_b)$. Finally, U_a generates the session key $SK_{ab} = c_a \oplus F_{\delta_{sa}}(1) \oplus F_{\delta_{sa}}(3) \oplus \sigma_b \oplus \rho_a$ 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_a is ε greater than h_a' is negligible. The error correction code defined in this paper can correct error 2ε. Finally, users U_a and U_b can obtain δ_{sa} and δ_{sb} which is same as the server, respectively. Meanwhile, user U_a can use

$$\begin{aligned}
\rho_a = Extr(k_a, \sigma_b) &= \left(\mathrm{k}_a + \sigma_b \frac{q-1}{2} \bmod q\right) \bmod 2 \\
&= \left(\mathrm{k}_b + 2\varepsilon + \sigma_b \frac{q-1}{2} \bmod q\right) \bmod 2 \\
&= \left(\mathrm{k}_b + \sigma_b \frac{q-1}{2} \bmod q\right) \bmod 2 + (2\varepsilon) \bmod 2 \\
&= \left(\mathrm{k}_b + \sigma_b \frac{q-1}{2} \bmod q\right) \bmod 2 \\
&= Extr(k_b, \sigma_b) = \rho_b,
\end{aligned} \tag{11}$$

to calculate ρ_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 the 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 $PKE = (keyGen, Enc, Dec)$ based on the ideal lattice is CCA secure $\varepsilon - ASPH$ function cluster corresponding to the public key system $PKE = (keyGen, Enc, Dec)$.ECC: $(0,1)^n \longrightarrow \{0,1\}^k$ is the coding algorithm of error correction code. It can correct the 2ε error partthrough the decoding algorithm ECC^{-1}: $(0,1)^k \longrightarrow \{0,1\}^n$ corresponding to the coding algorithm ECC. $F = \left\{F_\delta : \delta \in \{0,1\}^k\right\}_{k \in \mathbb{N}}$ is a pseudo-random function cluster; and then, this protocol is secure.*

Proof. Assuming that any PPT attacker can make $q_{exe}, q_{send}, q_{re}, q_{co}$ for *Execute*, *Send*, R*eveal* 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 $G_0, G_1, \cdots, G_8$. In this series of games, $\mathscr{A}$'s advantage of breaking the protocol gradually increases. Finally, as long as the advantage of the attacker's success in the game G_8 is negligible, $\mathscr{A}$ cannot break the protocol. The advantage of $\mathscr{A}$ breaking the protocol is defined as $Adv_{P,D}^{ake}(\mathscr{A}) = 2pr[Succ_{P,D}^{ake}(\mathscr{A})] - 1$.

Game G_0. This game corresponds to the real attack in the security model. For all kinds of queries sent by the attacker $\mathscr{A}$, $\mathscr{A}$ may get honest answers. At this time, the advantage of the attacker $\mathscr{A}$ is $Adv_{P,D}^{ake}(\mathscr{A}) = Adv_0(\mathscr{A})$.

Game G_1. In this game, we modify the simulation model of *Execute* inquiry and replace the answer generated by the user using $\mathrm{Pro}jH$ with the corresponding *Hash* calculation in each *Execute* inquiry, such as $h_{U'}^{'} = Hash(pk, hk_u, C_{u2}, m_u')$, $u \in \{A, B\}$. □

Proof. Due to the approximate correctness of $\varepsilon - ASPH$and the coding algorithm E: $(0,1)^n \longrightarrow \{0,1\}^k$ of the error correction code, $\mathscr{A}$ has the same advantages in the game G_1; it can be denoted by $Adv_1(\mathscr{A}) = Adv_0(\mathscr{A})$.

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_u in the message sent by the user for the first time is replaced with the ciphertext of encrypting the virtual verification element VT_0 (i.e., the verification element that does not belong to the password space D). Finally, we can see that the advantage difference of $\mathscr{A}$ between game G_2 and game G_1 can be ignored. We recorded it as $|Adv_2(\mathscr{A}) - Adv_1(\mathscr{A})| < 2\ q_{exe} \cdot Adv_{PKE}^{ake}(\mathrm{t} + O(q_{send} + q_{exe}))$. □

Proof. Taking the public key pk as a public parameter, $\mathscr{B}$ is the attacker of the public-key encryption system so that $\mathscr{B}$ can answer the queries of attacker$\mathscr{A}$. $\mathscr{B}$ sends (v_u', VT_0) 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, $\mathscr{A}$ makes their guess b' about the random bit b in the *Test* query. If $b' = b$, the guess is successful, and $\mathscr{B}$ outputs 1. Otherwise, $\mathscr{B}$ outputs 0. According to the CCA security of the public-key cryptosystem on the ideal lattice, the attacker $\mathscr{B}$ 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_0. Considering that the attacker $\mathscr{A}$ 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_3. In this game, $h_u = Hash(pk, hk_u, C_{u2}, m_u')$ and $u \in \{A, B\}$ in the *Execute* query will be replaced by a randomly selected bit string of equal length. Let $\varepsilon_{ASPH}(n)$ be a negligible upper bound of the statistical distance between the outputs of inputting nonverbal element and the uniform random distribution. Here, $|Adv_3(\mathscr{A}) - Adv_2(\mathscr{A})| < 2q_{exe} \cdot \varepsilon_{SAPH}(n)$. □

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 the 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, $Adv_4(\mathscr{A}) = Adv_3(\mathscr{A})$. □

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, φ_u) is random. According to the decision $RLWE_{n,m,q,\chi_\beta}$ problem, the advantages of game G_4 and game G_3 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. δ_{sa} and δ_{sb} are still randomly selected, but the values of $F_{\delta_{sa}}(1)$, $F_{\delta a}(3)$, $F_{\delta b}(1)$, and $F_{\delta_{sb}}(1)$ are replaced with independent and random numbers. $Adv_5(\mathscr{A}) = Adv_4(\mathscr{A})$ holds at this time because the pseudorandom function $F_{\delta_{su}}$ is hidden for the user. □

Proof. The user and the server have the same Δ_a, and according to the correctness of the decoding algorithm $E^{-1}: (0,1)^k \longrightarrow \{0,1\}^n$ of the error correction code, the user will get the same δ_{sa} 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, $Send(S_j, msg_1)$ means sending message msg_1 to the server instance S_j, and $Send(U^u, msg_2)$ indicates sending message msg_2 to the user instance U^u. 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 $Send_0(U^u, S_j)$ represents that the user instance U^u and the server instance S_j are activated and start to execute the protocol. In addition, the simulator $\mathscr{M}$ should record the private key sk corresponding to the public key pk in the key generation phase.

Game G_6**.** For $Send(S, < ID_u', ID_u, S, hk_u', C_{u1}', C_{u2}', ssid_u' >)$ 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' of the server, the emulator $\mathscr{M}$ 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, $\mathscr{M}$ checks whether the ciphertext (C_{u1}', C_{u2}') is valid. If not, $\mathscr{M}$ continues to reject the message. Otherwise, the emulator $\mathscr{M}$ 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 $Adv_5(\mathscr{A}) \le Adv_6(\mathscr{A}) + 2q_{send} \cdot \varepsilon_{SAPH}(n)$ holds. □

Proof. If (C_{u1}', C_{u2}') is a valid ciphertext, the simulator $\mathscr{M}$ can successfully decrypt to obtain v_u^*. If $v_u^* = v_u'$, this situation significantly increases the success advantage of the attacker $\mathscr{A}$. When $v_u^* \neq v_u'$, ciphertext $(C_{u1}', C_{u2}') \in X \backslash \overline{L}$ is obvious. According to the smoothness of $\varepsilon - ASPH$, $v_u^* \neq v_u'$ does not increase the advantage of the attacker $\mathscr{A}$. It is found that $Adv_5(\mathscr{A}) \le Adv_6(\mathscr{A}) + 2q_{send} \cdot \varepsilon_{SAPH}(n)$ holds by applying the proof of the game G_3.

Game G_7. We modify the query received by the user in this game. Let the message $msg_1 = < ID_u', ID_u, S, hk_u', C_{u1}', C_{u2}', ssid_u' >$ be a valid output of $Send(U^u, \langle \Delta_u, h_u >)$ for the previous $Send(U^u, < \Delta_u, h_u >)$ query. If $\langle \Delta_u, h_u \rangle$ is a replay of an honest simulated $Send(S, < ID_u', ID_u, S, hk_u', C_{u1}', C_{u2}', ssid_u' >)$ query, h_u' will no longer be generated for the user according to the protocol, but force the user to have the same δ_{su} as the

server. Because it is hidden from the attacker, the above modifications will not increase the attacker's advantage, and $Adv_7(\mathscr{A}) = Adv_6(\mathscr{A})$ holds.

Game G_8. The model of the response to the activation message $Send_0(U^u, S_j)$ is further modified. If the user is activated, the ciphertext is replaced with the encryption of the virtual verification element VT_0. Now, $|Adv_7(\mathscr{A}) - Adv_8(\mathscr{A})| < 2q_{send} \cdot Adv^{ake}_{PKE'}(t + O(q_{send} + q_{exe}))$. □

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

According to the constructed game G_8, the attacker $\mathscr{A}$ 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_U 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 $\langle \Delta_a, h_a \rangle$ 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 $\mathscr{A}$ can obtain the password of the legal user from the session. However, through the game constructed above, it can be found that the attacker can no longer obtain 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 ϖ are the parameters of Zipf. Note that $\mathscr{F}$ represents that the first two conditions are true; then $\Pr[\mathscr{F}] \le C \cdot q^{\varpi}_{send}$. 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, $Adv_8(\mathscr{A}) \le C \cdot q^{\varpi}_{send}$.

To sum up, the advantages of the attacker in the game G_0 is $Adv_0(\mathscr{A}) \le C \cdot q^{\varpi}_{send} + 2(q_{send} + q_{exe}) \cdot [Adv^{ake}_{PKE'}(t + O(q_{send} + q_{exe})) + \varepsilon_{SAPH}(n)]$. Meanwhile, $Adv_1(\mathscr{A}) = Adv_0(\mathscr{A})$; 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 sk'_u is not transmitted on the channel, and the attacker can only obtain $x_u = a \cdot sk'_u + 2e$ transmitted on the channel. On the premise that the attacker knows (a, x_u), the attacker needs to solve the search RLWE problem to obtain the temporary private key sk'_u. □

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 $\mathscr{A}_{RLWE}$ against the search RLWE problem and let adversary $\mathscr{A}_{privacy}$ destroy session key privacy. It is assumed that adversary $\mathscr{A}_{privacy}$ can have q_{exe}, q_{send} times *Execute* and *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 $\mathscr{A}_{RLWE}$ to obtain a triplet (x_b, C_b, φ_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 $\mathscr{A}_{RLWE}$ simulates protocol stipulates, honestly runs, and answers the *Execute* and *Send* query of adversary $\mathscr{A}_{privacy}$, but when responding to the query, $\mathscr{A}_{RLWE}$ replaces the relevant contents (x, C, φ) with (x_b, C_b, φ_b) in the real protocol.

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

$\Pr\{\mathscr{A}_{RLWE}\text{wins}\} = \Pr[b' \neq b | b = 0]$. Therefore, the probability $\mathscr{A}_{RLWE}$ solving the search RLWE problem can be calculated as

$$\begin{aligned}\Pr\{\mathscr{A}_{RLWE}\text{wins}\} &= \frac{1}{2}\left(\Pr[b' = b \mid b = 1] + \Pr[b' \neq b \mid b = 0]\right)\\ &= \frac{1}{2}\left(\frac{1}{2} + \frac{1}{2}\mathscr{A}^{privacy}_{P,D}\left(\mathscr{A}_{privacy}\right) + \left(1 - \frac{1}{2}\right)\right)\\ &= \frac{1}{2} + \frac{1}{4}\mathscr{A}^{privacy}_{P,D}\left(\mathscr{A}_{privacy}\right).\end{aligned} \tag{12}$$

□

Table 1: Security comparison.

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

Table 2: Computation and communication cost of the proposed protocol.

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

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 and Zhang [29], Liu and Xue [2] found that the adversary in [29] can obtain the trust of the server by replaying 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, 11^{th} Gen. Intel (*R*) core (TM) i5-1135g7 @ 2.40 GHz 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

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, regenerate the 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

[1] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[2] Y. Liu and K. Xue, "An improved secure and efficient password and chaos-based two-party key agreement protocol," *Nonlinear Dynamics*, vol. 84, no. 2, pp. 549–557, 2016.

[3] C. M. Chen, W. Fang, K. H. Wang, and T.-Y. Wu, "Comments on "An improved secure and efficient password and chaos-based two-party key agreement protocol"," *Nonlinear Dynamics*, vol. 87, no. 3, pp. 1–3, 2016.

[4] Y. Li, Q. Cheng, and W. Shi, "Security analysis of a lightweight identity-based two-party Authenticated key agreement protocol for IIoT environments," *Security and Communication Networks*, vol. 2021, pp. 1–6, 2021.

[5] L. Deng, J. Shao, and Z. Hu, "Identity based two-party authenticated key agreement scheme for vehicular ad hoc networks," *Peer-to-Peer Networking and Applications*, vol. 14, no. 4, pp. 2236–2247, 2021.

[6] J. Yang, B. Su, C. Guo, W. Han, and Y. Xiao, "Provably secure CL-KEM-based password-authenticated key exchange protocol," *International Journal of Sensor Networks*, vol. 23, no. 2, pp. 113–122, 2017.

[7] M. S. Farash and M. A. Attari, "An efficient and provably secure three-party password-based authenticated key exchange protocol based on Chebyshev chaotic maps," *Nonlinear Dynamics*, vol. 77, no. 1, pp. 399–411, 2014.

[8] A. Yin, Y. Guo, Y. Song, T. Qu, and C. Fang, "Two-round password-based authenticated key exchange from lattices," *Wireless Communications and Mobile Computing*, vol. 2020, no. 17, pp. 1–13, 2020.

[9] C.-M. Chen, K.-H. Wang, K.-H. Yeh, B. Xiang, and T.-Y. Wu, "Attacks and solutions on a three-party password-based authenticated key exchange protocol for wireless communications," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 8, pp. 3133–3142, 2019.

[10] H. Zhu and S. Geng, "A three-party dynamic identity-based authenticated key exchange protocol with forward anonymity," *Wireless Personal Communications*, vol. 109, no. 3, pp. 1911–1924, 2019.

[11] J. O. Kwon, I. R. Jeong, K. Sakurai, and D. H. Lee, "Efficient verifier-based password-authenticated key exchange in the three-party setting," *Computer Standards & Interfaces*, vol. 29, no. 5, pp. 513–520, 2007.

[12] C.-T. Li, C.-L. Chen, C.-C. Lee, C.-Y. Weng, and C.-M. Chen, "A novel three-party password-based authenticated key exchange protocol with user anonymity based on chaotic maps," *Soft Computing*, vol. 22, no. 8, pp. 2495–2506, 2018.

[13] X. Yang, M. Hou, and X. Wei, "Verifier-based three-party password authenticated key exchange protocol," *Journal of Computer Research and Development*, vol. 53, no. 10, pp. 2230–2238, 2016.

[14] Q. H. Zhang, X. X. Hu, W. F. Liu, and J. H. Wei, "Improved verifier-based three-party password-authenticated key exchange protocol," *Journal of Software*, vol. 31, no. 10, pp. 3238–3250, 2020.

[15] J. Ding, X. Xie, and X. Lin, "A simple provably secure key exchange scheme based on the learning with errors problem," *IACR Cryptology EPrint Archive*, vol. 2012, p. 688, 2012.

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

[17] J. X. Yu, H. H. Lian, Y. L. Tang et al., "Password-based three-party authenticated key exchange protocol form lattices," *Journal on Communications*, vol. 39, no. 11, pp. 87–97, 2018.

[18] R. Choi, H. An, K. Kim et al., "AtLast: another three-party lattice-based PAKE scheme," in *Proceedings of the 2018 Symposium on Cryptography and Information Security (SCIS 2018)*, Niigata, Japan, January 2018.

[19] C. Liu, Z. Zheng, K. Jia, and Q. You, "Provably secure three-party password-based authenticated key exchange from RLWE," in *Proceedings of the International Conference on Information Security Practice and Experience*, pp. 56–72, Kuala Lumpur, Malaysia, November 2019.

[20] M. Heydari, S. M. S. Sadough, M. S. Farash, S. A. Chaudhry, and K. Mahmood, "An efficient password-based authenticated key exchange protocol with provable security for mobile client-client networks," *Wireless Personal Communications*, vol. 88, no. 2, pp. 337–356, 2016.

[21] T. Y. Youn, E. S. Kang, and C. Lee, "Efficient three-party key exchange protocols with round efficiency," *Telecommunication Systems*, vol. 52, no. 2, pp. 1367–1376, 2013.

[22] Q. Zhang, P. Chaudhary, S. Kumari, Z. Kong, and W. Liu, "Verifier-based anonymous password-authenticated key exchange protocol in the standard model," *Mathematical Biosciences and Engineering*, vol. 16, no. 5, pp. 3623–3640, 2019.

[23] Q. Shu, S. B. Wang, B. Hu, and L. D. Han, "Verifier-based three-party password-authenticated key exchange protocol from ideal lattices," *Journal of Cryptologic Research*, vol. 8, no. 2, pp. 294–306, 2021.

[24] C. Peikert, "Lattice cryptography for the internet," in *Proceedings of the International Workshop on Post-quantum Cryptography*, pp. 197–219, Waterloo, ON, Canada, October 2014.

[25] M. Ye, X. X. Hu, and W. F. Liu, "Approximate smooth projective hash functions from ideal lattices," *Journal of Information Engineering University*, vol. 14, no. 1, pp. 13–21, 2013.

[26] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," *Journal of the Association for Computing Machinery*, vol. 60, no. 6, p. 43, 2013.

[27] R. Camer and V. Shoup, "Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption," in *Proceedings of the International Conference on the Theory & Applications of Cryptographic Techniques: Advances in Cryptology*, pp. 45–64, Zagreb, Croatia, May 2002.

[28] J. Katz and V. Vaikuntanathan, "Round-optimal password-based authenticated key exchange," in *Proceedings of the-Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011*, pp. 293–310, Springer-Verlag, Providence, RI, USA, March 2011.

[29] X. Guo and J. Zhang, "Secure group key agreement protocol based on chaotic Hash," *Information Sciences*, vol. 180, no. 20, pp. 4069–4074, 2010.