Provably Secure Three-party Password-based Authenticated Key Exchange from RLWE (Full Version)

Chao Liu[1], Zhongxiang Zheng[2], Keting Jia[2(✉)], and Qidi You[3]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, P.R. China
liu_chao@mail.sdu.edu.cn
[2] Department of Computer Science and Technology, Tsinghua University, P.R. China
ktjia@mail.tsinghua.edu.cn
[3] Space Star Technology co., LTD, P.R. China

Abstract. Three-party key exchange, where two clients aim to agree a session key with the help of a trusted server, is prevalent in present-day systems. In this paper, we present a practical and secure three-party password-based authenticated key exchange protocol over ideal lattices. Aside from hash functions our protocol does not rely on external primitives in the construction and the security of our protocol is directly relied on the Ring Learning with Errors (RLWE) assumption. Our protocol attains provable security. A proof-of-concept implementation shows our protocol is indeed practical.

Keywords: Password authentication · Three-party Key exchange · Provable security · RLWE · Post-quantum.

1 Introduction

Key Exchange (KE), which is a fundamental cryptographic primitive, allows two or more parties to securely share a common secret key over insecure networks. KE is one of the most important cryptographic tools and is widely used in building secure communication protocols. Authenticated Key Exchange (AKE), which enables each party to authenticate the other party, can prevent the adversary from impersonating the honest party in the conversation. Password-based Authenticated Key Exchange (PAKE), which allows parties to share a low-entropy password that is easy for human memory, has become an important cryptographic primitive because it is easy to use and does not rely on special hardware to store high-entropy secrets.

The early solution to this problem was to achieve two-party password-based authenticated key exchange (2PAKE), in which both parties identified their communication partners with shared passwords. Many 2PAKE protocols have been proposed [2,6,22]. However, in a communication environment where only 2PAKE protocols are available, each party must remember many passwords, for each entity with which he may wish to establish a session key corresponds to a password. In detail, assuming that a communication network has n users, in which any two users exchange a key, there will be $n(n-1)/2$ passwords to be shared, and all these passwords must be stored securely. This is unrealistic when the network is relatively large. To solve this problem, three-party PAKE (3PAKE) was proposed. In 3PAKE, each client shares a password with the trusted server, and then two clients will establish a common session key with the help of the server. This solution is very realistic in practical setup, because it provides each client user with the ability to exchange secure keys with all other client users, and each user only needs to remember one password. The 3PAKE protocol can be applied to various electronic applications, such as in the JobSearch International, trusted third parties can help employers and employees to hire on Jobsearch.

In 1995, Steiner et al. proposed the first 3PAKE protocol [26]. Then many works about 3PAKE protocols have been proposed [16,27,1,11,7]. For a security 3PAKE protocol, there are two types of attacks it should resist: *undetectable on-line password guessing attacks* [10] and *off-line password guessing attacks* [16]. In 1995, Ding and Horster [10] and Sun et al. [27] pointed out that Steiner et al.'s protocol [26] was vulnerable to undetectable on-line password guessing attacks. That is, an adversary can stay un-detected and log into the server during an on-line transaction. In 2000, Lin et al. [16] further pointed out Steiner et al.'s protocols [26] also suffer from off-line password guessing attacks. In this attack, an attacker can guess passwords off-line until getting the correct one. There is another attack: *detectable on-line password guessing attacks*, which requires the participation of the authentication server. In this attack, the server will detect a failed guess and record it. Since after a few unsuccessful guesses, the server can stop any further attempts, this attack is less harmful. In practice, password-based authenticated key exchanges are required to have a property, *forward secrecy*, that when the password is compromised, it does not reveal the earlier established session keys and the updating password.

However, the existed 3PAKE are based on the classic hard problems, such as factoring, the RSA problem, or the computational/decisional DH problem. It is well known that those problems are vulnerable to quantum computers [25]. Since the vigorous development of quantum computers, searching other counterparts based on problems which are believed to be resistant to quantum attacks is more and more urgent. Hence the motivation of this paper is that can we propose a proven security 3PAKE that can resist quantum attacks? Note that lattice-based cryptographic have many advantages such as quantum attacks resistance, asymptotic efficiency, conceptual simplicity and worst-case hardness assumption, and it is a perfect choice to build lattice-based 3PAKE.

Our contributions. In this paper, we propose a 3PAKE protocol based on the Ring Learning with Errors (RLWE), which in turn is as hard as some lattice problems (SIVP) in the worst case on ideal lattices [20]. Our protocol is designed without extra primitives such as public-key encryption, signature or message authentication code, which usually lead to a high cost for certain applications. By having the 3PAKE as a self-contained system, we show that our protocol directly relys on the hardness of RLWE and Pairing with Errors problem (PWE), which can reduce to the RLWE problem, in the random oracle model. Our protocol RLWE-3PAK resists undetectable on-line passwords guessing attacks and off-line passwords guessing attacks, and enjoys forward secrecy and quantum attacks resistance. Furthermore, our protocol enjoys *mutual authentication*, which means that the users and the server can authenticate one another.

In terms of protocol design, benefitting from the growth of lattice-based key exchange protocols [8,23], we can utilize the key agreement technique to construct our protocol. We use Peikert's [23] reconciliation mechanism to achieve the key agreement in our protocol. At the same time, in order to make our protocol resist undetectable passwords guessing attacks and off-line passwords guessing attacks, we also use additional key reconciliation mechanism between server and clients to realize the mutual authentication. Our security model is modified from Bellare et al.'s model [2,3]. Since the interactions in three-party setting are more complex than that of two-party setting, proving the security of our 3PAKE protocol is a very tricky problem. We use a variant of the Pairing with errors problem [9] to simplify the proof and the proof strategy followed from [21]. Finally, we manage to establish a full proof of security for our protocol and show that our protocol enjoys forward security.

We select concrete choices of parameters and construct a proof-of-concept implementation. The performance results show that our protocol is efficient and practical.

Related works. In the existed literatures, 3PAKE protocols are based on public/private key cryptography [16,26,10], which usually incur additional computation and communication overheads. Asymmetric key cryptography based protocols [15,17,11] usually require "the ideal cipher model", which is a strong assumption, to prove the security of the protocols. There are some other types of protocols [13,18] which are with no formal security proof.

Lattice-based AKE or PAKE. Zhang et al. [32] proposed an authenticated RLWE-based key exchange which is similar to HMQV [14]. In 2009, Katz and Vaikuntanathan [12] proposed a CCA-secure lattice-based PAKE, which is proven secure in standard model security. In 2017, Ding et al. [9] proposed RLWE-based PAKE, whose proof is based on random oracle model (ROM), and its implementation is very efficient. Then in 2017, Zhang and Yu [31] proposed a two-round CCA-secure PAKE based on the LWE assumption.

Roadmap. In Sect. 2, we introduce our security model, notations and the Ring Learning with Errors background. Our protocol RLWE-3PAK is in Sect. 3. And in Sect. 4, we give the proof of the protocol's security. The parameter choices and proof-of-concept implementation of our protocol is presented in Sect. 5. Finally, we conclude and discuss some further works in Sect. 6.

2 Preliminaries

2.1 Security Models

The security model is modified from [2] and [3]. The 3PAKE protocol involves three parties, two clients A and B who wish to establish a shared secret session key and a trusted server S who try to help distribute a key to A and B. Let P be a 3PAKE protocol.

Security game. Given a security parameter κ, an algorithmic game initialized is played between $\mathcal{CH}$ - a challenger, and a probability polynomial time adversary $\mathcal{A}$. For simulating network traffic for the adversary, $\mathcal{CH}$ will essentially run P.

Users and passwords. There is a fixed set of *users*, which is partitioned into two non-empty sets of *clients* and *servers*. We also assume D is some fixed, non-empty dictionary with size of L. Then before the game starts, a password pw_U is drawn uniformly at random from D and assigned to each *clients* outside of the adversary's view. And for each server S, we set $pw_S := (f(pw_U))_U$, where U runs through all of *clients*. Usually, f is some efficiently computable one-way function (in our protocol we let f be a hash function).

User instances. We denote some instance i of a *user* U as Π_U^i. The adversary $\mathcal{A}$ controls all the communications that exchange between a fixed number of parties by interacting with a set of Π_U^i oracles. At any point of in time, an client *user* instance Π_U^i may *accept*. When Π_U^i accepts, it holds a *partner-id* (PID) pid_U^i, a *session-id* (SID) sid_U^i, and a *session key* (SK) sk_U^i. The PID is the identity of the user that the instance believes talking to, and SK is what the instance aims to compute after the protocol completed. The SID is an identifier and is used to uniquely name the ensuing session. Note that the SID and PID are open to the adversary, and the SK certainly is secret for $\mathcal{A}$.

Oracle queries. The adversary $\mathcal{A}$ has an endless supply of oracles and it models various queries to them with each query models a capability of $\mathcal{A}$. The oracle queries by the adversary $\mathcal{A}$ are described as follows:

- The **Send**(U, i, M) query allows the adversary to send some message M to oracle Π_U^i of her choice at will. The Π_U^i oracle, upon receiving such a query, will compute what the protocol P says, updates its state, and then returns to $\mathcal{A}$ the response message. If Π_U^i has accepted or terminated, this will be made known to the adversary $\mathcal{A}$. This query is for dealing with controlling the communications by the adversary.
- The **Execute**(A, i, B, j, S, t) query causes P to be executed to completion between two *clients* instances Π_A^i, Π_B^j and a *server* instance Π_S^t, and hands all the execution's transcripts to $\mathcal{A}$. This query is for dealing with off-line password guessing attacks.
- The **Reveal**(U, i) query allows $\mathcal{A}$ to expose session key SK that has been previously accepted. If Π_U^i has accepted and holds some SK, then Π_U^i, upon receiving such a query, will sends SK back to $\mathcal{A}$. This query is for dealing with known-key security, which means that when the session key is lost, it does not reveal the other session keys.

- The **Corrupt**(U) query allows $\mathcal{A}$ to corrupt the user U at will. If U is a *server*, returns $(f(pw_C))_C$ to $\mathcal{A}$, else returns pw_U to $\mathcal{A}$. This query is for dealing with forward secrecy.
- The **Test**(U, i) is a query that does not correspond to $\mathcal{A}$'s abilities. The oracle chooses a bit $b \in \{0, 1\}$ randomly. If Π_U^i has accepted with some SK and is being asked by such a query, then $\mathcal{A}$ is given the actual session key when $b = 1$; $\mathcal{A}$ is given a key chosen uniformly at random when $b = 0$. $\mathcal{A}$ is allowed to query this oracle once and only on a fresh Π_U^i (defined in the following). This query models the semantic security of the session key SK.

Ending the game. Eventually, the adversary ends the game, and then outputs a single bit b'.
And next we define what constitutes the breaking of the 3PAKE protocol. Firstly we introduce the notions of instance *partnering* and instance *freshness with forward secrecy.*

Definition 1. (Partnering) *Let Π_A^i and Π_B^j be two instances. We shall say that Π_A^i and Π_B^j are partnered if both instances accept, holding $(sk_A^i, sid_A^i, pid_A^i)$ and $(sk_B^j, sid_B^j, pid_B^j)$ respectively, and the followings hold:*

- *$sid_A^i = sid_B^j = sid$ is not null and $sk_A^i = sk_B^j$ and $pid_A^i = B$ and $pid_B^j = A$;*
- *No instance besides Π_A^i and Π_B^j accepts with a SID of sid.*

Definition 2. (Freshness) *Instance Π_A^i is fs-fresh or it holds a fresh session key at the end of the execution if none of the following events occur:*

- ***Reveal**(A, i) was queried;*
- *a **Reveal**(B, j) was queried where Π_B^j is parted with Π_A^i, if it has one;*
- *before the **Test** query, a **Corrupt**(U) was queried for some user U and a **Send**(A,i,M) query occurs for some string M.*

Password Security. We say the adversary breaks the password security of 3PAKE if he learns the password of a user by either on-line or off-line password guessing attacks.
AKE security. We now define the advantage of the adversary $\mathcal{A}$ against protocol P for the authenticated key exchange (ake). Let $\mathrm{Succ}_P^{ake}(\mathcal{A})$ be the event that the adversary makes a single **Test**(A, i) query directed to some terminated fresh instances Π_A^i, and outputs a bit b' eventually, and $b' = b$ where b is the bit selected in the **Test**(A, i) query. Then $\mathcal{A}$'s advantage is defined as:

$$\mathrm{Adv}_P^{ake}(\mathcal{A}) \overset{\text{def}}{=} 2\Pr\left[\mathrm{Succ}_P^{ake}(\mathcal{A})\right] - 1$$

It is easy to verify that

$$\Pr(\mathrm{Succ}_P^{ake}(\mathcal{A})) = \Pr(\mathrm{Succ}_{P'}^{ake}(\mathcal{A})) + \epsilon \Longleftrightarrow \mathrm{Adv}_P^{ake}(\mathcal{A}) = \mathrm{Adv}_{P'}^{ake}(\mathcal{A}) + 2\epsilon.$$

The protocol 3PAKE is AKE-secured if $\mathrm{Adv}_P^{ake}(\mathcal{A})$ is negligible for all probabilistic polynomial time adversaries.

2.2 Notations

Let n be an integer, which is a power of 2. We define the ring of integer polynomials $R := \mathbb{Z}[x]/(x^n+1)$. For any positive integer q, we set $R_q := \mathbb{Z}_q[x]/(x^n+1)$ as the ring of integer polynomials modulo x^n+1, where every coefficient is reduced modulo q. For a polynomial y in R, identify y with its coefficient vector in $\mathbb{Z}$. Let the norm of a polynomial to be the norm of its coefficient vector. Assume χ is a probability distribution over R, then $x \xleftarrow{\$} \chi$ means the coefficients of x are sampled from χ.

For any positive real $\beta \in \mathbb{R}$, we set $\rho_\beta(x) = exp(-\pi\frac{||x||^2}{\beta^2})$ as the Gaussian function, which is scaled by a parameter β. Let $\rho_\beta(\mathbb{Z}^n) = \sum_{\mathbf{x}\in\mathbb{Z}^n}\rho_\beta(\mathbf{x})$. Then for a vector $\mathbf{x} \in \mathbb{Z}^n$, let $D_{\mathbb{Z}^n,\beta}(\mathbf{x}) = \frac{\rho_\beta(\mathbf{x})}{\rho_\beta(\mathbb{Z}^n)}$ to indicate the n-dimensional discrete Gaussian distribution. Usually we denote this distribution as χ_β.

2.3 Ring Learning with Errors

The Learning with Errors (LWE) problem was first introduced by Oded Regev in [24]. He showed that under a quantum reduction, solving LWE problem in the average cases was as hard as solving the worst cases of the certain lattice problems. However since with a large key sizes of $O(n^2)$, LWE based cryptosystems are not efficient for practical applications. In 2010, Lyubashevsky, Peikert, and Regev [20] introduced the version of LWE in the ring setting: the Ring Learning with Errors problem, which could drastically improve the efficiency.

For uniform random elements $a, s \xleftarrow{\$} R_q$ and an error distribution χ, let $A_{s,\chi}$ denote the distribution of the RLWE pair $(a, as+e)$ with the error $e \xleftarrow{\$} \chi$. Then given polynomial number of such samples, the search version of RLWE is to find the secret s, and the decision version of the RLWE problem (DRLWE$_{q,\chi}$) is to distinguish $A_{s,\chi}$ from an uniform distribution pair on $R_q \times R_q$. RLWE enjoys a worst case hardness guarantee, which we state here.

Theorem 1. ([20], Theorem 3.6) *Let $R = \mathbb{Z}[x]/(x^n+1)$ where n is a power of 2, $\alpha = \alpha(n) < \sqrt{logn/n}$, and $q \equiv 1 \mod 2n$ which is a* ploy(n)*-bounded prime such that $\alpha q \geq \omega(\sqrt{logn})$. Then there exists a* ploy(n)*-time quantum reduction from $\tilde{O}(\sqrt{n}/\alpha)$-SIVP (Short Independent Vectors Problem) on ideal lattices in the ring R to solving DRLWE$_{q,\chi}$ with $l-1$ samples, where $\chi = D_{\mathbb{Z}^n,\beta}$ is the discrete Gaussian distribution with parameter $\beta = \alpha q \cdot (nl/log(nl))^{1/4}$.*

We have the following useful fact.

Lemma 1. ([19], Lemma 4.4) *For any $k > 0$, $\Pr_{x\leftarrow\chi_\beta}(|x| > k\beta) \leq 2e^{-\pi k^2}$.*

Note that taking $k = 6$ gives tail probability approximating 2^{-162}.
Reconciliation mechanism. We now recall the reconciliation mechanism defined in [23]. This technique is one of the foundations of our protocol.

For an integer p (e.g. $p = 2$) which divides q, define the modular rounding function $\lfloor\cdot\rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ as $\lfloor x\rceil_p := \lfloor \frac{p}{q}\cdot x\rceil$ and $\lfloor\cdot\rfloor_p : \mathbb{Z}_q \to \mathbb{Z}_p$ as $\lfloor x\rfloor_p :=$

$\lfloor \frac{p}{q} \cdot x \rceil$. Let the modulus $q \geq 2$ and be an even, define disjoint intervals $I_0 := \{0, 1, \ldots, \lfloor \frac{q}{4} \rceil - 1\}$, $I_1 := \{-\lfloor \frac{q}{4} \rceil, \ldots, -1\} \bmod q$. Note that when $v \in I_0 \cup I_1$, $\lfloor v \rceil_2 = 0$, and when $v \in (I_0 + \frac{q}{2}) \cup (I_1 + \frac{q}{2})$, $\lfloor v \rceil_2 = 1$. Define the cross-rounding function $\langle \cdot \rangle_2 : \mathbb{Z}_q \to \mathbb{Z}_2$ as $\langle v \rangle_2 := \lfloor \frac{4}{q} \cdot v \rfloor \bmod 2$. Note that $\langle v \rangle_2 = b \in \{0, 1\}$ such that $v \in I_b \cup (\frac{q}{2} + I_b)$;.

Define the set $E := [-\frac{q}{8}, \frac{q}{8}) \cap \mathbb{Z}$. Then suppose v, w are sufficiently close, and given w and $\langle v \rangle_2$, we can recover $\lfloor v \rceil_2$ using the reconciliation function rec: $\mathbb{Z}_q \times \mathbb{Z}_2 \to \mathbb{Z}_2$:

$$\mathrm{rec}(w, b) = \begin{cases} 0 & \text{if } w \in I_b + E (\bmod q), \\ 1 & \text{otherwise.} \end{cases}$$

When q is odd, to avoid the bias produced by the rounding function, Peikert introduced a randomized function dbl(): $\mathbb{Z}_q \to \mathbb{Z}_{2q}$. For $v \in \mathbb{Z}_q$, $\mathrm{dbl}(v) := 2v - \bar{e} \in \mathbb{Z}_{2q}$ for some random $\bar{e} \in \mathbb{Z}$ which is independent of v and uniformly random moduloes two. Usually we denote v with an overline to means that $\bar{v} \leftarrow \mathrm{dbl}(v)$.

For ease of presentation, we define function HelpRec(X): (1). $\overline{X} \leftarrow \mathrm{dbl}(X)$; (2). $W \leftarrow \langle \overline{X} \rangle_2$; $K \leftarrow \lfloor \overline{X} \rceil_2$; (3). return (K, W).

Note that for $w, v \in \mathbb{Z}_q$, we need apply the appropriated rounding function from $\mathbb{Z}_{2q}$ to $\mathbb{Z}_2$, which means that $\lfloor x \rceil_p = \lfloor \frac{p}{2q} \cdot x \rceil$, $\langle x \rangle_2 = \lfloor \frac{4}{2q} \cdot x \rfloor$, and similar with rec function. Obviously, if $(K, W) \leftarrow \mathrm{HelpRec}(X)$ and $Y = X + e$ with $||e||_\infty < \frac{q}{8}$, we have $\mathrm{rec}(2 \cdot Y, W) = K$. These definitions also can be extended to R_q by applying coefficient-wise to the coefficients in $\mathbb{Z}_q$ of a ring elements. In other words, for a ring element $v = (v_0, \ldots, v_{n-1}) \in R_q$, set $\lfloor v \rceil_2 = (\lfloor v_0 \rceil_2, \ldots, \lfloor v_{n-1} \rceil_2)$; $\langle v \rangle_2 = (\langle v_0 \rangle_2, \ldots, \langle v_{n-1} \rangle_2)$; $\mathrm{HelpRec}(v) = (\mathrm{HelpRec}(v_0), \ldots, \mathrm{HelpRec}(v_{n-1}))$. And for a binary-vector $b = (b_0, \ldots, b_{n-1}) \in \{0, 1\}^n$, set $\mathrm{rec}(v, b) = (\mathrm{rec}(v_0, b_0), \ldots, \mathrm{rec}(v_{n-1}, b_{n-1}))$.

Lemma 2. ([23]) *For $q \geq 2$ is even, if v is uniformly random chosen from $\mathbb{Z}_q$, then $\lfloor v \rceil_2$ is uniformly random when given $\langle v \rangle_2$; if $w = v + e \bmod q$ for some $v \in \mathbb{Z}_q$ and $e \in E$, then $\mathrm{rec}(w, \langle v \rangle_2) = \lfloor v \rceil_2$. For $q > 2$ is odd, if v is uniformly random chosen from $\mathbb{Z}_q$ and $\bar{v} \leftarrow \mathrm{dbl}(v) \in \mathbb{Z}_{2q}$, then $\lfloor \bar{v} \rceil_2$ is uniformly random given $\langle \bar{v} \rangle_2$.*

The PWE assumption. To prove the security of our protocol, we introduce the Pairing with Errors (PWE) assumption. This assumption is following the work in [9], and we replace the reconciliation mechanism of them by Peikert's version. For any $(a, s) \in R_q^2$, we set $\tau(a, s) := \lfloor \overline{as} \rceil_2$ and if there is $(c, W) \leftarrow \mathrm{HelpRec}(as)$, then $\tau(a, s) = c = \mathrm{rec}(\overline{as}, W)$. Assume that a PPT adversary $\mathcal{A}$ takes inputs of the form (a_1, a_2, b, W), where $(a_1, a_2, b) \in R_q^3$ and $W \in \{0, 1\}^n$, and outputs a list of values in $\{0, 1\}^n$. $\mathcal{A}$'s objective is to obtain the string $\tau(a_2, s)$ in its output, where s is randomly chosen from R_q, b is a "small additive perturbation" of $a_1 s$, W is $\langle \overline{a_2 s} \rangle_2$. Define

$$\mathrm{Adv}_{R_q}^{\mathrm{PWE}}(\mathcal{A}) \stackrel{\mathrm{def}}{=} Pr\Big[a_1 \xleftarrow{\$} R_q; a_2 \xleftarrow{\$} R_q; s, e \xleftarrow{\$} \chi_\beta; b \leftarrow a_1 s + e; \\ W \leftarrow \langle \overline{a_2 s} \rangle_2 : \tau(a_2, s) \in \mathcal{A}(a_1, a_2, b, W)\Big].$$

Let $\mathrm{Adv}_{R_q}^{\mathrm{PWE}}(t, N) = max_{\mathcal{A}} \left\{ \mathrm{Adv}_{R_q}^{\mathrm{PWE}}(\mathcal{A}) \right\}$, where the maximum is taken over all adversaries times complexity which at most t that output a list containing at most N elements of $\{0,1\}^n$. Then for t and N polynomial in κ, the PWE assumption states that $\mathrm{Adv}_{R_q}^{\mathrm{PWE}}(t, N)$ is negligible.

To states the hardness of PWE assumption, We define the decision version of PWE problem as follows. If DPWE is hard, so is PWE.

Definition 3. (DPWE) *Given* $(a_1, a_2, b, W, \sigma) \in R_q \times R_q \times R_q \times \{0,1\}^n \times \{0,1\}^n$ *where* $W = \langle \overline{K} \rangle_2$ *for some* $K \in R_q$*, where* $\overline{K} \leftarrow \mathrm{dbl}(K)$ *and* $\sigma = \mathrm{rec}(2 \cdot K, W)$*. The Decision Pairing with Errors problem* (DPWE) *is to decide whether* $K = a_2 s + e_1$*,* $b = a_1 s + e_2$ *for some* s, e_1, e_2 *is drawn from* χ_β*, or* (K, b) *is uniformly random in* $R_q \times R_q$*.*

In order to show the reduction of the DPWE problem to the RLWE problem, we would like to introduce a definition to what we called the RLWE-DH problem [9] which can reduce to RLWE problem.

Definition 4. (RLWE-DH) *Let* R_q *and* χ_β *be defined as above. Given an input ring element* (a_1, a_2, b, K)*, where* (a, X) *is uniformly random in* R_q^2*, The* DRLWE-DH *problem is to tell if* K *is* $a_2 s + e_1$ *and* $b = a_1 s + e_2$ *for some* $s, e_1, e_2 \xleftarrow{\$} \chi_\beta$ *or* (K, b) *is uniformly random in* $R_q \times R_q$*.*

Theorem 2. ([9], Theorem 1) *Let* R_q *and* χ_β *be defined as above, then the* RLWE-DH *problem is hard to solve if* RLWE *problem is hard.*

Theorem 3. *Let* R_q *and* χ_β *be defined as above. The* DPWE *problem is hard if the* RLWE-DH *problem is hard.*

Proof. Suppose there exists an algorithm D which can solve the DPWE problem on input (a_1, a_2, b, W, σ) where for some $K \in R_q$, $W = \langle \overline{K} \rangle_2$ and $\sigma = \mathrm{rec}(2 \cdot K, W)$ with non-negligible probability ϵ. By using D as a subroutine, we can build a distinguisher D' on input (a_1', a_2', b', K'), solve the RLWE-DH problem :

- Compute $W = \langle \overline{K'} \rangle$ and $\sigma = \mathrm{rec}(2 \cdot K', W)$.
- Run D using the input $(a_1', a_2', b', W, \sigma)$.
  - If D outputs 1 then K' is $a_2' s + e_1$ for some $e_1 \xleftarrow{\$} \chi_\beta$ and $b' = a_1 s + e_2$ for some $s, e_1 \xleftarrow{\$} \chi_\beta$.
  - Else (K', b') is uniformly random element from $R_q \times R_q$.

Note that if (a_1', b'), (a_2', K') is two RLWE pairs, with input $(a_1', a_2', b', W, \sigma)$ defined above, D outputs 1 with probability ϵ, hence RLWE-DH can be solved with probability ϵ using distinguisher D'. This means that RLWE-DH can be solved with non-negligible advantage, which contradicts RLWE-DH's hardness. □

3 A New Three-party Password Authenticated Key Exchange

In this section we introduce a new 3PAKE based on RLWE: RLWE-3PAK. The protocol RLWE-3PAK is given in Fig.1.

3.1 Description of RLWE-3PAK

Let $q = 2^{\omega(logn)} + 1$ be an odd prime such that $q \equiv 1 \bmod 2n$. Let $a \in R_q$ be a fixed element chosen uniformly at random and given to all users. Let χ_β be a discrete Gaussian distribution with parameter β. Let $H_1 : \{0,1\}^* \mapsto R_q$ be hash function, $H_l : \{0,1\}^* \to \{0,1\}^\kappa$ for $l \in \{2,3,4\}$ be hash functions which is used for verification of communications, and $H_5 : \{0,1\}^* \to \{0,1\}^\kappa$ be a Key Derivation Function (KDF), where κ is the bit-length of the final shared key. We model the hash functions H_l for $l \in \{1,2,3,4,5\}$ as random oracles. We will make use of $\langle\cdot\rangle_2$, $\lfloor\cdot\rceil_2$, HelpRec() and rec() defined in Sect.2.3.

The function f used to compute client passwords' verifiers for the server is instantiated as : $f(\cdot) = -H_1(\cdot)$. Our protocol which is illustrated in Fig.1 consists of the following steps:

Client B initiation. Client B sends the identity of A, the one who he wants to communicate with, and his own to S as an initial request. (Note that, this step also can be executed by A.)

Server S first response. Server S receivers B's message, then S chooses $s_f, e_f, s_g,\, e_g \xleftarrow{\$} \chi_\beta$ to compute $b_A = as_f + e_f$ and $b_B = as_g + e_g$, and computes public elements $m_A = b_A + \gamma'$ and $m_B = b_B + \eta'$ where $\gamma' := -H_1(pw_1)$, $\eta' := -H_1(pw_2)$. Then S sends $\langle m_A, m_B\rangle$ to B.

Client B first response. After receiving S's message, client B checks if $m_A, m_B \in R_q$. If not, aborts; otherwise retrieves $b'_B = m_B + \eta$ where $\eta = H_1(pw_2)$ and chooses $s_B, e_B, e'_B \xleftarrow{\$} \chi_\beta$ to compute $p_B = as_B + e_B$ and $v_1 = b'_B s_B + e'_B$. Then B uses v_1 to compute $(\sigma_B, w_B) \leftarrow$ HelpRec(v_1), and computes $k_{BS} \leftarrow H_2(\langle A, B, S, b'_B, \sigma_B\rangle)$. B sends $\langle m_A, m_B, p_B, k_{BS}, w_B\rangle$ to A.

Client A first response. After receiving B's message, A checks if $m_A, p_B \in R_q$. If not, aborts; otherwise similarly with B, retrieves $b'_A = m_A + \gamma$ where $\gamma = H_1(pw_1)$ and chooses $s_A, e_A, e'_A \xleftarrow{\$} \chi_\beta$ to compute $p_A = as_A + e_A$ and $v_2 = b'_A s_A + e'_A$. Then A uses v_2 to compute $(\sigma_A, w_A) \leftarrow$ HelpRec(v_2), and computes $k_{AS} \leftarrow H_2(\langle A, B, S, b'_A, \sigma_A\rangle)$. Finally A sends $\langle p_A, p_B, k_{AS}, k_{BS}, w_A, w_B\rangle$ to S.

Server S second response. After receiving A's message, S checks if $p_A, p_B \in R_q$. If not, aborts; otherwise computes $\sigma'_A \leftarrow$ rec$(2p_A s_f, w_A)$ and checks if $k_{AS} = H_2(\langle A, B, S, b_A, \sigma'_A\rangle)$. If not, aborts; otherwise computes $\sigma'_B \leftarrow$ rec$(2p_B s_g, w_B)$ and checks if $k_{BS} = H_2(\langle A, B, S, b_B, \sigma'_A\rangle)$. If not, aborts; otherwise continues. Then, S samples $s_S, e_1, e_2 \xleftarrow{\$} \chi_\beta$, and computes $c_B = p_A s_S + e_1$ and $c_A = p_B s_S + e_2$ which will be used to retrieve the final messages by A and B. To give the authentication of S to B and A, S computes $k_{SA} \leftarrow H_2(\langle A, B, S, p_B, \sigma'_A\rangle)$ and $k_{SB} \leftarrow H_2(\langle A, B, S, p_A, \sigma'_B\rangle)$. Finally S sends $\langle p_A, c_A, c_B, k_{SA}, k_{SB}\rangle$ to B.

Client B second response. After receiving S's message, B checks if $p_A, c_A, c_B \in R_q$. If not, aborts; otherwise checks if $k_{SB} = H_2(\langle A, B, S, p_A, \sigma_B\rangle)$. If not, aborts; otherwise samples $e''_B \xleftarrow{\$} \chi_\beta$ and computes $v_B = c_B s_B + e''_B$, $(\sigma, w) \leftarrow$ HelpRec(v_B), $k = H_3(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma\rangle)$, $k'' = H_4(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma\rangle)$ and $sk_B = H_5(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma\rangle)$. Finally B sends $\langle c_A, w, k, k_{SA}\rangle$ to A.

Client A		Client B		Server S
Input pw_1,B		Input pw_2, A		$\gamma' = -\gamma, \eta' = -\eta$
			$\xrightarrow{(A,B)}$	
				$b_A = as_f + e_f$ $b_B = as_g + e_g$ $m_A = b_A + \gamma'$ $m_B = b_B + \eta'$
		$\eta = H_1(pw_2)$ $b'_B = m_B + \eta$ $p_B = as_B + e_B$ $v_1 = b'_B s_B + e'_B$ $(\sigma_B, w_B) \leftarrow$HelpRec$(v_1)$	$\xleftarrow{m_A,m_B}$	
$\gamma = H_1(pw_1)$ $b'_A = m_A + \gamma$		$k_{BS} \leftarrow H_2(\langle A, B, S,$ $b'_B, \sigma_B\rangle)$		
$p_A = as_A + e_A$ $v_2 = b'_A s_A + e'_A$ $(\sigma_A, w_A) \leftarrow$ HelpRec(v_2) $k_{AS} \leftarrow H_2(\langle A, B, S,$ $b'_A, \sigma_A\rangle)$	$\xleftarrow{C_{B1}}$	$C_{B1} \leftarrow \langle m_A, m_B, p_B,$ $k_{BS}, w_B\rangle$		
		$\xrightarrow{\langle p_A,p_B,k_{AS},k_{BS},w_A,w_B\rangle}$		$\sigma'_A \leftarrow$ rec$(2p_A s_f, w_A)$ Abort if $k_{AS} \neq H_2(\langle A,$ $B, S, b_A, \sigma'_A\rangle)$ $\sigma'_B \leftarrow$ rec$(2p_B s_g, w_B)$ Abort if $k_{BS} \neq H_2(\langle A,$ $B, S, b_B, \sigma'_B\rangle)$
		Abort if $k_{SB} \neq H_2(\langle A,$ $B, S, p_A, \sigma_B\rangle)$	$\xleftarrow{C_S}$	$c_B = p_A s_S + e_1$ $c_A = p_B s_S + e_2$ $k_{SA} = H_2(\langle A, B, S, p_B, \sigma'_A\rangle)$ $k_{SB} = H_2(\langle A, B, S, p_A, \sigma'_B\rangle)$ $C_S = \langle p_A, c_A, c_B, k_{SA}, k_{SB}\rangle$
		$v_B = c_B s_B + e''_B$ $(\sigma, w) \leftarrow$ HelpRec(v_B) $k = H_3(\langle A, B, S, m_A,$ $m_B, p_A, p_B, \sigma\rangle)$		
Abort if $k_{SA} \neq H_2(\langle A,$ $B, S, p_B, \sigma_A\rangle)$ $\sigma \leftarrow$ rec$(2c_A s_A, w)$ Abort if $k \neq H_3(\langle A, B,$ $S, m_A, m_B, p_A, p_B, \sigma\rangle)$ else $k' = H_4(\langle A, B, S, m_A,$ $m_B, p_A, p_B, \sigma\rangle)$	$\xleftarrow{C_{B2}}$	$k'' = H_4(\langle A, B, S, m_A,$ $m_B, p_A, p_B, \sigma\rangle)$ $C_{B2} = \langle c_A, w, k, k_{SA}\rangle$		
	$\xrightarrow{k'}$	Abort if $k' \neq k''$		

Fig. 1. Three-party password authenticated protocol: RLWE-3PAK, where $s_S, e_S, s_f, e_f, s_g, e_g, s_B, e_B, e'_B, e''_B, e_A, e'_A, e_1, e_2$ is sampled from χ_β. Shared session key is $sk = H_5(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma\rangle)$.

Client A **second response.** After receiving B's message, A checks if $c_A \in R_q$. If not, aborts; otherwise checks if $k_{SA} = H_2(\langle A, B, S, p_B, \sigma_A \rangle)$. If not, aborts; otherwise computes $\sigma' \leftarrow \mathrm{rec}(2c_A s_A, w)$. Then checks if $k = H_3(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma' \rangle)$. If not, aborts; otherwise computes $k' = H_4(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma' \rangle)$ and $sk_A = H_5(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma' \rangle)$. Finally A sends k' to B.
Client B **finish.** After receiving k' from A, B checks if $k' = k''$. If not, aborts; otherwise terminates.

3.2 Design Rationale

In our protocol, the check for ring elements ensures that all ring operations are valid. The participants are split into clients and servers and servers are allowed to store a password file. By having the server store not pw_1, pw_2, but $\langle \gamma', \eta' \rangle$ allows us to improve the efficiency of the server.

Our 3PAKE may seem a bit complicated, but this is because of the need to provide authentication in the exchange sessions. When we remove all authentication functions, we will find that the main body of the protocol is very simple. In the absence of authentication, party A and party B send p_A and p_B to S, respectively. S computes c_A and c_B by using p_A, p_B and a random value s_S, and sends c_A (resp. c_B) to A (resp. B). Finally, A and B can calculate the same secret key by using the reconciliation mechanism with c_A, c_B and their own secret keys.

In the 3PAKE model, A and B can not authenticate each other, so they need the help of server S to provide the authentication. In our protocol, k_{AS} (k_{BS}) can be viewed as an authentication of A (resp. B) to S. Note that S and A share a password, so *only* A can calculate the corresponding b_A which is set by S, and *only* B can calculate b_B. Meanwhile, *only* A (resp. B) can calculate the same key value σ_A (resp. σ_B) with S through the reconciliation mechanism.

Note that the adversary can not guess the password in a limited number of times, so k_{AS} (or k_{BS}) can not be computed by adversary in a few tries, which makes our protocol resist undetectable on-line password guessing attacks [10]. Finally in order to resist off-line password guessing attacks [16], session values delivered by the server also need to provide authentication of S to A and B, that is why we add k_{SA} and k_{SB} in server's outputs. In the security proof, two types of password guessing attacks is discussed in detail. Note that the final **Client B finish** step may seems redundant, but it is indispensable for the property of forward security [2].

3.3 Correctness

Note that in protocol RLWE-3PAK, if $\mathrm{rec}(2p_A s_f, w_A) = \lfloor \overline{v_2} \rceil_2$, the verification for k_{AS} would be correct. By the definition of the reconciliation mechanism and Lemma 2, we have $||v_2 - p_A s_f||_\infty < \frac{q}{8}$ should be satisfied with overwhelming probability. We have

$$\begin{aligned} v_2 &= b_A s_A + e'_A = (as_f + e_f)s_A + e'_A \\ &= as_f s_A + e_f s_A + e'_A \end{aligned}$$

and

$$p_A s_f = as_A s_f + e_A s_f.$$

Hence we need $||v_2 - p_A s_f||_\infty = ||e_f s_A + e'_A - e_A s_f||_\infty < \frac{q}{8}$. Similarly for the verification of k_{BS}, we need $||v_1 - p_B s_g||_\infty = ||e_g s_B + e'_B - e_B s_g||_\infty < \frac{q}{8}$ with overwhelming probability. And to compute the correct key, it needs $\mathrm{rec}(2c_A s_A, w) = \lfloor \overline{v_B} \rceil_2$, which means that $||v_B - c_A s_A||_\infty < \frac{q}{8}$. We have

$$\begin{aligned} v_B &= c_B s_B + e''_B = (p_A s_S + e_1)s_B + e''_B \\ &= as_A s_S s_B + e_A s_S s_B + e_1 s_B + e''_B \end{aligned}$$

and

$$\begin{aligned} c_A s_A &= (p_B s_S + e_2)s_A \\ &= as_A s_B s_S + e_B s_A s_S + e_2 s_A. \end{aligned}$$

Therefore, it also needs $||v_B - c_A s_A||_\infty = ||e_A s_B s_S + e_1 s_B + e''_B - e_B s_A s_S - e_2 s_A||_\infty < \frac{q}{8}$ with overwhelming probability.

4 Security for RLWE-3PAK

Here we prove that the RLWE-3PAK protocol is secure, which means that an adversary $\mathcal{A}$ who attacks the system cannot determine the SK of fresh instances with greater advantage than that of an detectable on-line dictionary attack.

Theorem 4. *Let P:=*RLWE-3PAK*, described in Fig.1, using ring R_q, and with a password dictionary of size L. Fix an adversary $\mathcal{A}$ that runs in time t, and makes $n_{se}, n_{ex}, n_{re}, n_{co}$ queries of type* **Send, Execute, Reveal, Corrupt**, *respectively. Then for $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$:*

$$\mathrm{Adv}_P^{ake\text{-}fs}(\mathcal{A}) = C \cdot n_{se}^s + O\Big(n_{se}\mathrm{Adv}_{R_q}^{\mathrm{PWE}}(t', n_{ro}^2) + \mathrm{Adv}_{R_q}^{\mathrm{DRLWE}}(t', n_{ro}) + \frac{(n_{se} + n_{ex})(n_{ro} + n_{se} + n_{ex})}{q^n} + \frac{n_{se}}{2^\kappa}\Big)$$

where $s \in [0.15, 0.30]$ and $C \in [0.001, 0.1]$ are constant CDF-Zipf regression parameters depending on the password space L [29].

The proof of above theorem will proceed by introducing a series of protocols $P_0, P_1, \ldots, P_7$ related to P, with $P_0 = P$. In P_7, the only possible attack for the adversary $\mathcal{A}$ is natural detectable on-line password guessing attacks. Eventually, there are

$$\mathrm{Adv}_{P_0}^{ake} \leq \mathrm{Adv}_{P_1}^{ake} + \epsilon_1 \leq \cdots \leq \mathrm{Adv}_{P_7}^{ake} + \epsilon_7$$

where $\epsilon_1, \ldots, \epsilon_7$ are negligible values in k. Together with above relations, our result is given by computing the success probability of detectable on-line attack in P_7 in the end of the proof.For the convenience of readers, we give a informal description of protocols $P_0, P_1, \ldots, P_7$ in Fig.2, and given the proof sketches of negligible advantage gain from P_{i-1} to P_i in Fig.3.

We firstly explain our estimation parameters here. Let **correctpw** be the event that the adversary make a correct guess of password by detectable on-line passwords attacks. In most existing PAKE studies, passwords are assumed to follow a uniformly random distribution, and Pr(**correctpw**)$\leq \frac{n_{se}}{L}+negl(\kappa)$, where L is the size of the password dictionary, n_{se} is the max number of $\mathcal{A}$'s active on-line password guessing attempts before a **Corrupt** query and $negl()$ is a negligible function. Ding Wang and Ping Wang [29] introduced CDF-Zipf model and in this model Pr(**correctpw**)$\leq C \cdot n_{se}^{s}+negl(\kappa)$ for the Zipf parameters C and s which is depended on the password space L. CDF-Zipf model is more consistent with the real world attacks than traditional formulation. For example, when considering trawling guessing attacks, the actual advantage will be 6.84% when $n_{se} = 10^2$, and 12.45% when $n_{se} = 10^3$ [28], but the traditional formulation greatly underestimate Advantage to be 0.01% when $n_{se} = 10^2$, and 0.10% when $n_{se} = 10^3$. When further considering targeted guessing attacks (in which the adversary makes use of the target users personal information), advantage will be about 20% when $n_{se} = 10^2$, 25% when $n_{se} = 10^3$, and 50% when $n_{se} = 10^6$ [30]. So we prefer this model in our analysis.

Proof. Firstly, we distinguish Client of A Action (CAA) queries, Client of B Action (CBA) and Server Action (SA) queries. The adversary makes one of the following queries:

- **CBA0** query if it instructs some unused Π_B^i to send the first message to server S, and this corresponds to client B initiation in Sect.3.2;
- **SA1** query if it sends some message to a previously unused Π_S^t expecting some message which is intend to be sent to some B, and this corresponds to server S first response;
- **CBA1** query if it sends some message to some Π_B^j expecting some message which is intend to be sent to some A and this corresponds to client B first response;
- **CAA1** query if it sends some message to some unused Π_A^i expecting some message which is intend to be sent to S, and this corresponds to client A first response;
- **SA2** query if it sends some message to Π_S^t expecting some message which is intend to be sent to some B, and this corresponds to server S second response;

P_0 The original protocol P.
P_1 The hash function H_1's outputs are no longer a randomly chosen element γ in R_q, but a ring element $\gamma = as + e \in R_q$, where s, e is sampled from χ_β.
P_2 The honest parties randomly choose m_A, m_B, p_A or p_B values which are seen previously in the execution, the protocol halts and the adversary fails.
P_3 The protocol answers **Send** and **Execute** queries without using any random oracle queries. Subsequent random oracle queries made by A are backpatched, as much as possible, to be consistent with the responses to the **Send** and **Execute** queries. (This is a standard technique for proofs involving random oracles.)
P_4 If an $H_l(\cdot)$ query is made, for $l \in \{3, 4, 5\}$, it is not checked for consistency against **Execute** queries. That means instead of backpatching to maintain consistency with an **Execute** query, the protocol responds with a random output.
P_5 If before a **Corrput** query, a correct shared secret key guess is made against client A or B (This can be determined by an $H_l(\cdot)$ query, for $l \in \{3, 4, 5\}$, using the correct inputs to compute k, k' or session key), the protocol halts and the adversary automatically succeeds.
P_6 If the adversary makes a shared secret key guess against two partnered clients, the protocol halts and the adversary fails.
P_7 The protocol uses an internal password oracle, which holds all passwords and be used to exam the correctness of a given password. Such an oracle aims at the password security. (It also accepts **Corrupt**(U) queries, which returns $(f(pw_C))_C$ if U is an server and otherwise returns pw_U to A).

Fig. 2. Informal description of protocols $P_0, P_1, \ldots, P_7$

$P_0 \to P_1$ Unless the decision version of RLWE is solved with non-negligible advantage, theses two protocols are indistinguishable.
$P_1 \to P_2$ This is straightforward.
$P_2 \to P_3$ By inspection, the two protocols are indistinguishable unless the decision version of RLWE is solved with non-negligible advantage or the adversary makes an **Client** A **second response** (resp. **Client** B **finish.**) query with a k (resp. k') value that is not the output of an $H_3(\cdot)$ (resp. $H_4(\cdot)$) query that would be a correct shared secret key guess. However, the probability of these is negligible.
$P_3 \to P_4$ This can be shown using a standard reduction from PWE. On input $(a, X, Y = as_y + e_y, W)$, where s_y, e_y are unknown, we plug in Y added by random RLWE pair for client B' p_B values, and X added by random RLWE pair for server' c_B values. Then from a correct $H_l(\cdot)$ guess for $l \in \{3, 4, 5\}$, we can compute $\tau(X, s_y)$.
$P_4 \to P_5$ This is obvious.
$P_5 \to P_6$ This can be shown using a standard reduction from PWE, similar to the one for **Execute** queries. On input $(a, X, Y = as_y + e_y, W)$, where s_y, e_y are unknown, we plug in Y for client A' p_A values, and X added by random RLWE pair for server' c_A values. Then from a correct $H_l(\cdot)$ guess for $l \in \{3, 4, 5\}$, we can compute $\tau(X, s_y)$.
$P_6 \to P_7$ By inspection, there two protocols are indistinguishable. Finally, in P_7, the adversary success only if he breaks the password security or makes a correct shared secret key guess. We show these happens with negligible abilities by using a standard reduction from PWE.

Fig. 3. Proof sketches of negligible advantage gain from P_{i-1} to P_i

- **CBA2** query if it sends a message to some Π_B^j expecting some message which is intend to be sent to some A, and this corresponds client B second response;
- **CAA2** query if it sends some message to some Π_A^i expecting some message which is intend to be sent to some B, and this corresponds to client A second response;
- **CBA3** query if it sends some message to a Π_B^j expecting the last protocol message, and this corresponds to client B finish.

For the convenience of the reader, we define some events in the following. Those events correspond to the adversary $\mathcal{A}$ making a session key guess against the client instance or two partnered clients instances and verification key guess against the client and server, respectively.

- **testsk**(A, i, B, S, l): for some m_A, m_B, p_A, p_B, $\mathcal{A}$ makes an $H_l(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma\rangle)$ query, CAA1 query to a client instance Π_A^i with input $\langle m_A, p_B, k_{BS}, w_B\rangle$ and output $\langle p_A, p_B, k_{AS}, k_{BS}, w_A, w_B\rangle$ and a CAA2 query to Π_A^i with input $\langle c_A, w, k, k_{SA}\rangle$, where the latest query is either the $H_l(\cdot)$ query or the CAA1 query, $\sigma = \text{rec}(2c_A s_A, w)$. The associated value of this event is $H_l(\langle A, B, S,\ m_A, m_B, p_A, p_B, \sigma\rangle)$, $l \in \{3, 4, 5\}$ (respectively, the k, k', sk_A^i).
- **testsk!**(A, i, B, S): for some w and k a CAA2 query with input $\langle c_A, w, k, k_{SA}\rangle$ causes a **testsk**$(A, i, B, S, 3)$ event to occur, with associated value k.
- **testsk**(B, j, A, S, l): for some m_A, m_B, p_A, p_B, $\mathcal{A}$ makes an $H_l(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma\rangle)$ query, and previously made CBA0 query with output $\langle A, B\rangle$, CBA1 query to a client instance Π_B^j with input $\langle m_A, m_B\rangle$ and output $\langle m_A, p_B,\ k_{BS}, w_B\rangle$, and previously made CBA2 query to Π_B^j with input $\langle p_A, c_A, c_B, k_{SA},\ k_{SB}\rangle$ and output $\langle c_A, w, k, k_{SA}\rangle$, $\sigma = \lfloor 2c_B s_B \rceil_2$. The associated value of this event is $H_l(\langle A, B, S,\ m_A, m_B, p_A, p_B, \sigma\rangle)$, $l \in \{3, 4, 5\}$ (respectively, the k, k'', sk_B^j).
- **testsk!**(B, j, A, S): a CBA3 query to Π_B^j is made with k', where a **testsk**$(B, j, A,\ S, 4)$ event previously occurs with associated value k'.
- **testsk***(B, j, A, S): **testsk**(B, j, A, S, l) occurs for some $l \in \{3, 4, 5\}$.
- **testsk**(A, i, B, j, S) for some $l \in \{3, 4, 5\}$, both a **testsk**(A, i, B, S, l) event and a **testsk**(B, j, A, S, l) event occur, where Π_A^i is paired with Π_B^j and Π_B^i is paired with Π_A^j after its CBA2 query.
- **testexecsk**(A, i, B, j, S, t): for some m_A, m_B, p_A, p_B, $\mathcal{A}$ makes an $H_l(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma\rangle)$ query for $l \in \{3, 4, 5\}$, and previously $\mathcal{A}$ made an **Execute**(A, i, B, j, S, t) query that generates $m_A, m_B, p_A, p_B, w, c_A, c_B$, $\sigma = \text{rec}(2c_A s_A,\ w) = \text{rec}(2c_B s_B, w)$.
- **correctsk**: before any **Corrupt** query, either a **testsk!**(A, i, B, S) event occurs for some A, i, B and S, or a **testsk***(B, j, A, S) event occurs for some B, j, A and S.
- **correctskexec**: a **testexecsk**(A, i, B, j, S, t) event occurs for some A, i, B, j, S and t.
- **pairedskguess**: a **testsk**(A, i, B, j, S) event occurs, for some A, i, B, j, S.

- **correctkBS** (resp., **correctkAS**): for some m_B, p_B, (resp., m_A, p_A) $\mathcal{A}$ makes an $H_2(\langle A, B, S, b_B, \sigma_B\rangle)$ (resp., $H_2(\langle A, B, S, b_A, \sigma_A\rangle)$)query, a SA1 query to a server instance Π_S^t with input $\langle A, B\rangle$ and output $\langle m_A, m_B\rangle$, and a SA2 query to Π_S^t with input $\langle p_A, p_B, k_{AS}, k_{BS}, w_A, w_B\rangle$, and maybe a $H_1(pw_2)$ (resp., $H_1(pw_1)$) query returning η (resp., γ) with a password pw_2 (resp., pw_1), where the latest query is either the $H_2(\cdot)$ query or the SA2 query, $\sigma_B = \mathrm{rec}(2p_B s_g, w_B)$, $b_B = m_B - \eta$, $b_B = as_g + e_g$ (resp., $\sigma_A = \mathrm{rec}(2p_A s_f, w_A)$, $b_A = m_A - \gamma$, $b_A = as_f + e_f$). The associated value of this event is k_{BS} (resp., k_{AS}).
- **correctkSB**: for some m_B, p_B, $\mathcal{A}$ makes an $H_2(\langle A, B, S, p_A, \sigma_B\rangle)$ query, and a CBA1 query to a server instance Π_B^j with input $\langle m_A, m_B\rangle$ and output $\langle m_A, p_B, k_{BS}, w_B\rangle$, and a CBA2 query to Π_B^j with input $\langle p_A, c_A, c_B, k_{SA}, k_{SB}\rangle$, and maybe a $H_1(pw_2)$ query returning η with a password pw_2, where the latest query is either the $H_2(\cdot)$ query or the CBA2 query, $\sigma_B = \mathrm{rec}(2p_B s_g, w_B)$, $b_B = m_B - \eta$, $b_B = as_g + e_g$. The associated value of this event is k_{SB}.
- **correctkSA**: for some m_A, p_A, $\mathcal{A}$ makes an $H_2(\langle A, B, S, p_B, \sigma_A\rangle)$ query, and a CAA1 query to a instance Π_A^i with input $\langle m_A, p_B, k_{BS}, w_B\rangle$ and output $\langle p_A, p_B, k_{AS}, k_{BS}, w_A, w_B\rangle$, and a CAA2 query to Π_A^i with input $\langle c_A, w, k, k_{SA}\rangle$, and maybe a $H_1(pw_1)$ query returning γ with a password pw_1, where the latest query is either the $H_2(\cdot)$ query or the CAA2 query, $\sigma_A = \mathrm{rec}(2p_A s_f, w_A)$, $b_A = m_A - \gamma$, $b_A = as_f + e_f$. The associated value of this event is k_{SA}.

We assume that n_{ro} and $n_{se} + n_{ex}$ are both at least 1. And we make a standard assumption of the random oracle that a new query is answered with a fresh random value, and a query that is not new is answered identically to the past response. Furthermore, let $H_1 : \{0,1\}^* \to R_q$, be a hash function where the final hashed element $\gamma \in R_q$ is sampled uniformly from R_q. We assume that if the adversary $\mathcal{A}$ made an $H_l(\cdot)$ query for $l \in \{3, 4, 5\}$, then the corresponding $H_{l'}(\cdot)$ query is made automatically where $l' \in \{3, 4, 5\} \setminus \{l\}$. Even if all queries are considered to be made by $\mathcal{A}$, $\mathcal{A}$ is only able to see the outputs of the hash function.

Protocol P_0: Let P_0 be the original protocol P.

Protocol P_1: Let P_1 be identical to P_0 except that the hash function H_1's outputs are no longer a randomly chosen element γ in R_q, but a ring element $\gamma' = as + e \in R_q$, where s, e is sampled from χ_β. For details, hash function H_1 would map password $pw \in \{0,1\}^*$ to $(s, e) \in R_q \times R_q$ where s, e are sampled from χ_β, then computes a ring element $\gamma' = as + e \in R_q$, finally outputs γ'. Note that γ and γ' are indistinguishable under the assumption of RLWE. Hence P_1 is indistinguishable from P_0 until DRLWE is solved with non-negligible advantage. The reason why we define this protocol is to prove the security of our protocol correctly in P_4, P_6 and P_7.

Claim 1 *For any adversary $\mathcal{A}$,*

$$\mathrm{Adv}_{P_0}^{ake}(\mathcal{A}) \leq \mathrm{Adv}_{P_1}^{ake}(\mathcal{A}) + \mathrm{Adv}_{R_q}^{\mathrm{DRLWE}}(t', n_{ro}).$$

The claim above is straightforward from the definition of P_1.
Protocol P_2: Let P_2 be identical to P_1, except that if the honest parties randomly choose m_A, m_B, p_A or p_B values which are seen previously in the execution, the protocol aborts and thus the adversary fails.

For convenient, here we define four events:

- Let E_1 (resp., E_2) be the event that an m_A (resp., m_B) value generated in a SA1 or **Execute** query is equal to an m_A (resp., m_B) value already seen in a previous SA1 or **Execute** query, an m_A (resp., m_B) value which is used as input in a previous CAA1 (resp., CBA1) or SA1 query, or an m_A (resp., m_B) value in some previous $H_l(\cdot)$ query (made by $\mathcal{A}$), for $l \in \{3, 4, 5\}$.
- Let E_3 (resp., E_4) be the event that an p_A (resp., p_B) value generated in a CAA1 (resp., CBA1) query is equal to an p_A (resp., p_B) value already seen in a previous CAA1 (resp., CBA1) or **Execute** query, an p_A (resp., p_B) value which is used as input in a previous SA1 query, or an p_A (resp., p_B) value in some previous $H_l(\cdot)$ query (made by $\mathcal{A}$), for $l \in \{3, 4, 5\}$.

Let $E = E_1 \vee E_2 \vee E_3 \vee E_4$ then P_2 is identical to P_1 except that if E occurs, the protocol aborts (and the adversary fails). This protocol can make sure that every $H_l(\cdot)$ query returns a new one which is independent of anything that previously created.

Claim 2 *For any adversary $\mathcal{A}$,*

$$\text{Adv}_{P_1}^{ake}(\mathcal{A}) \leq \text{Adv}_{P_2}^{ake}(\mathcal{A}) + \frac{O((n_{se} + n_{ex})(n_{ro} + n_{se} + n_{ex}))}{q^n}.$$

Proof. The probability that m_A, m_B, p_A, p_B has previously been generated in a **Send, Execute**, or random oracle query is $\frac{n_{ro}+n_{ex}+n_{se}}{q^n}$. And if event E doesn't occur, there need $n_{se} + n_{ex}$ values to be unique. Hence the probability of any of m_A, m_B, p_A, p_B not being unique is $\frac{O((n_{se}+n_{ex})(n_{ro}+n_{se}+n_{ex}))}{q^n}$. The claim follows. □

Protocol P_3: Let P_3 be identical to P_2, except that in **Send** and **Execute** queries, outputs are answered without using any random oracle querie. Subsequent random oracle queries made by $\mathcal{A}$ are backpatched, as much as possible, to be consistent with the responses to the **Send** and **Execute** queries.

For details, the queries in P_3 are answered as follows:

- In an **Execute**(A, i, B, j, S, t) query, $m_A \leftarrow as_{ma} + e_{ma}, m_B \leftarrow as_{mb} + e_{mb}, p_A \leftarrow as_{pa} + e_{pa}, p_B \leftarrow as_{pb} + e_{pb}$, where $s_{ma}, e_{ma}, s_{mb}, e_{mb}, s_{pa}, e_{pa}, s_{pb}, e_{pb} \xleftarrow{\$} \chi_\beta$, $w, w_A, w_B \xleftarrow{\$} \{0,1\}^n$, $k, k', k_{AS}, k_{BS}, k_{SA}, k_{SB} \xleftarrow{\$} \{0,1\}^\kappa$ and $sk_A^i \leftarrow sk_B^j \xleftarrow{\$} \{0,1\}^\kappa$.
- In a SA1 query to instance Π_S^t, $m_A \leftarrow as_{ma} + e_{ma}$, $m_B \leftarrow as_{mb} + e_{mb}$, where $s_{ma}, e_{ma}, s_{mb}, e_{mb} \xleftarrow{\$} \chi_\beta$.
- In a CBA1 query to instance Π_B^j, $p_B \leftarrow as_{pb} + e_{pb}$, where $s_{pb}, e_{pb} \xleftarrow{\$} \chi_\beta$, $k_{BS} \xleftarrow{\$} \{0,1\}^\kappa$, $w_B \xleftarrow{\$} \{0,1\}^n$.

- In a CAA1 query to instance Π_A^i, $p_A \leftarrow as_{pa} + e_{pa}$, where $s_{pa}, e_{pa} \xleftarrow{\$} \chi_\beta$, $k_{AS} \xleftarrow{\$} \{0,1\}^\kappa$, $w_A \xleftarrow{\$} \{0,1\}^n$.
- In a SA1 query to instance Π_S^t, if this query causes a **correctkAS** and **correctkBS** event to occur, then set $c_A \leftarrow as_{ca} + e_{ca}, c_B \leftarrow as_{cb} + e_{cb}$, where $s_{ca}, e_{ca}, s_{cb}, e_{cb} \xleftarrow{\$} \chi_\beta$, $k_{SA}, k_{SB} \xleftarrow{\$} \{0,1\}^\kappa$, else, Π_S^t aborts.
- In a CBA2 query to instance Π_B^j, if this query causes a **correctkSB** event to occur, then set $w \xleftarrow{\$} \{0,1\}^n$, $sk_B^j \xleftarrow{\$} \{0,1\}^\kappa$, $k, k'' \xleftarrow{\$} \{0,1\}^\kappa$, otherwise Π_B^j aborts.
- In a CAA2 query to instance Π_A^i, do the following:
  - If this query causes a **testsk!**(A, i, B, S) event and **correctkSA** event to occur, then set k' to associated value of the event **testsk**$(A, i, B, S, 4)$, and set key sk_A^i to associated value of the event **testsk**$(A, i, B, S, 5)$.
  - Else if Π_A^i is paired with an instance Π_B^j, $sk_A^i \leftarrow sk_B^j$, $k' \xleftarrow{\$} \{0,1\}^\kappa$.
  - Otherwise, Π_A^i aborts.
- In a CBA3 query to instance Π_B^j, if this query causes a **testsk!**(B, j, A, S) event to occur, or if instance Π_B^j is paired with a client instance Π_A^i, terminates. Otherwise, Π_B^j aborts.
- In an $H_l(\langle A, B, S, \ldots\rangle)$ query, for $l \in \{2,3,4,5\}$, if this $H_l(\cdot)$ query causes a **testsk**(A, i, B, S, l) event, **testsk**(B, j, A, S, l) event, **testexecsk**(A, i, B, j, S, t) event, **correctkAS** event, **correctkBS** event, **correctkSA** event or **correctkSB** event to occur, then output the associated value of the event, else outputs a random value from $\{0,1\}^\kappa$.

Claim 3 *For any adversary $\mathcal{A}$,*

$$\mathrm{Adv}_{P_2}^{ake}(\mathcal{A}) = \mathrm{Adv}_{P_3}^{ake}(\mathcal{A}) + \frac{O(n_{se})}{2^\kappa} + O(\mathrm{Adv}_{R_q}^{\mathrm{RLWE}}(t', n_{ro})).$$

Proof. In P_2, in a CBA2 query of a client instance Π_B^j, if a **correctkBS** event occurs, produces a sk_B^j and k and k'' that are uniformly chosen from $\{0,1\}^\kappa$, otherwise aborts, and since the $H_l(\cdot)$ query that determines sk_B^j is a new one, this session key is independent of anything that previously created. Then in a CBA3 query, if a **testsk!**(B, j, A, S) event occurs, or Π_B^j is paired, the instance terminates, and if Π_B^j is unpaired and no **testsk!**(B, i, A, S) event occurs, then either the instance terminates or aborts, and it is easy to verify that the total probability of any instance terminating in this case is at most $\frac{n_{se}}{2^\kappa}$.

And in P_2, for any client instance Π_A^i, either: (1) a **correctkAS** and **testsk!**(A, i, B, S) event occurs, and then k' and sk_A^i are set to the values associated with the **testsk**$(A, i, B, S, 4)$ and **testsk**$(A, i, B, S, 5)$ events, respectively. or (2) no **testsk!**(A, i, B, S) event occurs, but exactly one instance Π_B^j is paired with Π_A^i, then set $sk_A^i = sk_B^j$, and k' is uniformly chosen from $\{0,1\}^\kappa$, independent of anything that previously occurred (since no **testsk**$(A, i, B, S, 3)$ event could have occurred in this case), or (3) no **testsk!**(A, i, B, S) event occurs and no instance is paired with Π_A^i, then either the instance terminates or aborts. Note that in the last case the total probability of any instance terminating is at most $\frac{n_{se}}{2^\kappa}$.

Then for any $H_l(\langle A, B, S, \cdot, \cdot, \cdot, \cdot\rangle)$ query, $l \in \{2, 3, 4, 5\}$, either: (1) it causes a **correctkAS** event, **correctkBS** event, **correctkSA** event, **correctkSB** event, **testsk**(B, j, A, S, l) event, or **testexecsk**(A, i, B, j, S, t) event to occur, in which case the output is set to be the associated value of the event, or (2) it does not cause a **testsk**(A, i, B, j, S, t) event, but does cause a **testsk**$(A, i, B, S, 3)$ event to occur, where the CAA2 query of the event had input $\langle\cdot, \cdot, k, \cdot\rangle$, in which case either Π_A^i terminated and the output is k, or Π_A^i aborted and the output is uniformly chosen from $\{0, 1\}^\kappa \backslash \{k\}$, or (3) $H_l(\cdot)$ output a uniformly chosen value from $\{0, 1\}^\kappa$, which is independent of anything that previously produced, since this is a new $H_l(\cdot)$ query, for $l \in \{2, 3, 4, 5\}$.

For $l = 3$, the second case above, where the output is fixed can only occur when an unpaired client instance terminated with no **testsk!**(A, i, B, S) event. For $l \in \{4, 5\}$, the second case occurs if only when an $H_3(\cdot)$ query causes a second case where its output is fixed.

If an unpaired client instance Π_A^i never terminates without a **testsk!**(A, i, B, S) event, an unpaired instance Π_B^j never terminates without a **testsk!**(B, j, A, S) event, and $\mathcal{A}$ can not solve decision version of RLWE, then P_3 is consistent with P_2. The claim follows. □

Protocol P_4: Let P_4 be identical to P_3 except that in an $H_l(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma\rangle)$ query, $l \in \{3, 4, 5\}$, there is no check for a **testexecsk**(A, i, B, j, S, t) event.

Claim 4 *For any adversary $\mathcal{A}$ running in time t, there is a $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$ such that*

$$\mathrm{Adv}_{P_3}^{ake}(\mathcal{A}) \leq \mathrm{Adv}_{P_4}^{ake}(\mathcal{A}) + 2\mathrm{Adv}_{R_q}^{\mathrm{PWE}}(t', n_{ro}).$$

Proof. Let E be the event that the **testexecsk**(A, i, B, j, S, t) occurs. Obviously, P_3 and P_4 are indistinguishable if E does not occur. When $\mathcal{A}$ is running against protocol P_3, we suppose that the probability that E occurs is ϵ, then $\Pr(\mathrm{Succ}_{P_3}^{ake}(\mathcal{A})) \leq \Pr(\mathrm{Succ}_{P_4}^{ake}(\mathcal{A})) + \epsilon$ and thus $\mathrm{Adv}_{P_3}^{ake}(\mathcal{A}) \leq \mathrm{Adv}_{P_4}^{ake}(\mathcal{A}) + 2\epsilon$.

Firstly, assume that the adversary can cause **testexecsk**(A, i, B, j, S, t) event occurs with non-negligible probability. Then we construct an algorithm D that attempts to solve PWE problem by running $\mathcal{A}$ on a simulation of the protocol P_3. Given (a, X, Y, W), the objective is to find $\tau(X, s_y)$ if $Y = as_y + e_y$ for some $s_y, e_y \xleftarrow{\$} \chi_\beta$. D simulates P_3 for $\mathcal{A}$ with following changes:

- In an **Execute**(A, i, B, j, S, t) query, set $p_B = Y + as_f + e_f$ where $s_f, e_f \xleftarrow{\$} \chi_\beta$ and $c_B = X + as_g + e_g$ where $s_g, e_g \xleftarrow{\$} \chi_\beta$;
- When $\mathcal{A}$ finishes, for every $H_l(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma\rangle)$ query, where p_B, c_B and w were generated in an **Execute**(A, i, B, j, S, t) query, $\sigma = \mathrm{rec}(2v_B, w)$ with $v_B = c_B s_B + e''_B$. Then we have: $b_B := as_B + e_B = m_B - \eta_1$

and

$$\begin{aligned} v_B &= c_B s_B + e''_B = (X + as_g + e_g)(s_y + s_f) + e''_B \\ &= Xs_y + (as_g + e_g)s_y + (X + as_g + e_g)s_f + e''_B \\ &\approx Xs_y + Y \cdot s_g + (X + as_g + e_g)s_f + e''_B \end{aligned}$$

So we have $Xs_y \approx v_B - Y \cdot s_g - (X + as_g + e_g)s_f - e''_B$. Let

$$\sigma' = \mathrm{rec}(2(v_B - Y \cdot s_g - (X + as_g + e_g)s_f - e''_B), W).$$

Add the value of σ' to the list of possible values for $\tau(X, s_y)$.

Note that the simulation sets $p_B = Y + (as_f + e_f)$ instead of $p_B = as_{pb} + e_{pb}$ which is distinguishable if there are anyone who can solve the decision version of RLWE problem. It is the same for the setting of c_B. Then if E occurs, D adds the correct $\tau(X, s_y)$ to the list with non-negligible advantage. Such a simulation D is indistinguishable from P_3 until E occurs or the decision version of DRLWE problem is solved with non-negligible advantage. If E occurs, D adds the correct $\tau(X, s_y)$ to the list with non-negligible advantage. After E occurs, the simulation would be distinguishable from P_3. But we do make the assumption that $\mathcal{A}$ still follows the appropriate time and query bounds even if $\mathcal{A}$ distinguishes the simulation from P_3.

If the running time of simulator is t', and they creates a list of size n_{ro} with advantage ϵ. Note that $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$, the claim follows from the fact that $\mathrm{Adv}_{R_q}^{\mathrm{PWE}}(D) \leq \mathrm{Adv}_{R_q}^{\mathrm{PWE}}(t', n_{ro})$. □

Protocol P_5: Let P_5 be identical to P_4 except that if **correctsk** occurs then the protocol halts and the adversary succeeds automatically. Note that this causes following changes:

- In a CAA2 query to Π_A^i, if a **testsk!**(A, i, B, S) event occurs and no **Corrupt** query has been made, the protocol halts and the adversary automatically succeeds.
- In an $H_l(\cdot)$ query, for $l \in \{3, 4, 5\}$, if a **testsk***(B, j, A, S) event occurs and no **Corrupt** query has been made, the protocol halts and the adversary automatically succeeds.

Claim 5 *For any adversary* $\mathcal{A}$

$$\mathrm{Adv}_{P_4}^{ake}(\mathcal{A}) \leq \mathrm{Adv}_{P_5}^{ake}(\mathcal{A}).$$

The above claim is obviously by the definition.

Note that in P_5, until **correctsk** event or a **Corrupt** query occurs, no unpaired client will *terminate*.

Protocol P_6: Let P_6 be identical to P_5 except that if a **pairedskguess** event occurs, the protocol halts and $\mathcal{A}$ fails. And we assume that the test for the event **pairedskguess** occurs before the test for **correctsk** while a query is made. Note that this involves the following change: if a **testsk**(A, i, B, S, l) event occurs (this should be checked in a CAA2 query, or an $H_l(\cdot)$ query) for $l \in \{3, 4, 5\}$, check if a **testsk**(A, i, B, j, S) event also occurs.

Claim 6 *For any adversary $\mathcal{A}$ running in time t, there is a $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$ such that*

$$\mathrm{Adv}_{P_5}^{ake}(\mathcal{A}) \leq \mathrm{Adv}_{P_6}^{ake}(\mathcal{A}) + 2n_{se}\mathrm{Adv}_{R_q}^{\mathrm{PWE}}(t', n_{ro}).$$

Proof. When $\mathcal{A}$ is running against protocol P_5, we suppose the probability that **pairedskguess** event occurs is ϵ. Then $\Pr(\mathrm{Succ}_{P_5}^{ake}(\mathcal{A}) \leq \Pr(\mathrm{Succ}_{P_6}^{ake}(\mathcal{A}))+\epsilon$, and we have $\mathrm{Adv}_{P_5}^{ake}(\mathcal{A}) \leq \mathrm{Adv}_{P_6}^{ake}(\mathcal{A})+2\epsilon$.

Here we construct an algorithm D which attempts to solve PWE by running the adversary on a simulation of the protocol P_5. Given (a, X, Y, W), D chooses a random $d \in \{1, \ldots, n_{se}\}$ and simulates P_5 for the adversary with following changes:

- In the dth CAA1 query, say to a client instance $\Pi_A^{i'}$, with some input $\langle m_A, p_B, k_{BS}, w_B\rangle$, set $p_A \leftarrow Y$.
- In a SA1 query to a server instance Π_S^t, with input $\langle p_A, p_B, k_{AS}, k_{BS}, w_A, w_B\rangle$ where there was a CBA1 query to Π_B^j with input $\langle m_A, m_B\rangle$ and output $\langle m_A, p_B, k_{BS}, w_B\rangle$, and a CAA1 query to $\Pi_A^{i'}$ (i.e., the instance with the dth CAA1 query) with input $\langle m_A, p_B, k_{BS}, w_B\rangle$ and output $\langle p_A, p_B, k_{AS}, k_{BS}, w_A, w_B\rangle$, set $c_A \leftarrow X + as_g + e_g$ where $s_g, e_g \stackrel{\$}{\leftarrow} \chi_\beta$.
- In a CAA2 query to $\Pi_A^{i'}$, if $\Pi_A^{i'}$ is unpaired, D outputs 0 and halts.
- In a CBA2 query to Π_B^j, if Π_B^j was paired with $\Pi_A^{i'}$ after its CBA2 query, but is not now paired with $\Pi_A^{i'}$, no test for **correctsk** is made, and Π_B^j aborts.
- When $\mathcal{A}$ finishes, for every $H_l(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma\rangle)$ query, $l \in \{3, 4, 5\}$ where p_A was generated by $\Pi_A^{i'}$, p_B, w were generated by Π_B^j and m_A, m_B, c_A were generated by a server instance Π_S^t, respectively, where Π_B^j was paired with $\Pi_A^{i'}$ after its CBA2 query, $\sigma = \mathrm{rec}(2v_A, w)$ with $v_A = c_A s_A$,

we can see that,

$$\begin{aligned} v_A = c_A s_A &= (X + as_g + e_g)s_y \\ &= Xs_y + (as_g + e_g)s_y \\ &\approx Xs_y + Y \cdot s_g \end{aligned}$$

So $Xs_y \approx v_A - Y \cdot s_g$. And,

$$\sigma' = \mathrm{rec}(2(v_A - Y \cdot s_g), W).$$

Finally, add the value of σ' to the list of possible values for $\tau(X, s_y)$.

This simulation is perfectly indistinguishable from P_5 until (1) a **testsk**(B, j, A, S) event occurs, where Π_B^j was paired with $\Pi_A^{i'}$ after the CBA2 query, or (2) $\Pi_A^{i'}$ is not paired with a client instance when the CBA2 query is made. Note that the probability of **pairedskguess** event occurring for $\Pi_A^{i'}$ is at least $\frac{\epsilon}{n_{se}}$, and this is at most the probability of an event of type (1) occurring. Since an event of type (2) implies that **pairedskguess** would never have occurred in P_5 for $\Pi_A^{i'}$. If an event of type (1) occurs, D adds the correct $\tau(X, s_y)$ to the list.

Note that in either case, such a simulation may be distinguishable from P_5, but the fact that a **pairedskguess** event occur with probability at least $\frac{\epsilon}{n_{se}}$ doesn't change. However even if $\mathcal{A}$ can distinguish the simulation from P_5, $\mathcal{A}$ still follows the appropriate time and query bounds by our assumption.

Note that with advantage $\frac{\epsilon}{n_{se}}$, D creates a list of size n_{ro}, and the running time of D is $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$. Then the claim follows from the fact that $\mathrm{Adv}_{R_q}^{\mathrm{PWE}}(D) \leq \mathrm{Adv}_{R_q}^{\mathrm{PWE}}(t', n_{ro})$. □

Protocol P_7: Let P_7 be identical to P_6 except that there is an internal *password oracle*, which holds all passwords and be used to exam the correctness of a given password. Such an oracle aims at the password security. *Password oracle* is not available to adversary and generates all passwords during initialization. Then this oracle accepts queries of the form $\mathbf{testpw}(U, pw)$ and returns TRUE if $pw = pw_U$, and FALSE otherwise. It also accepts $\mathbf{Corrupt}(U)$ queries, which returns $(f(pw_C))_C$ if U is an *server* and otherwise returns pw_U to $\mathcal{A}$. When a $\mathbf{Corrupt}(U)$ query is received in the protocol, it is answered using a $\mathbf{Corrupt}(U)$ query to the password oracle.

Claim 7 *For any adversary* $\mathcal{A}$,

$$\mathrm{Adv}_{P_6}^{ake}(\mathcal{A}) = \mathrm{Adv}_{P_7}^{ake}(\mathcal{A}).$$

Proof. Obviously, P_6 and P_7 are indistinguishable. □

Let **correctpw** be the event that the adversary make a correct guess of password. From the description of P_7, it is easy to find that the probability of an adversary $\mathcal{A}$ succeeding in P_7 is bounded by

$$\begin{aligned}\Pr(\mathrm{Succ}_{P_7}^{ake}(\mathcal{A})) \leq & \Pr(\mathbf{correctpw}) + \Big(\Pr(\mathbf{correctsk}|\neg\mathbf{correctpw}) \\ & + \Pr(\mathrm{Succ}_{P_7}^{ake}(\mathcal{A})|\neg\mathbf{correctsk} \cap \neg\mathbf{correctpw}) \\ & \cdot \Pr(\neg\mathbf{correctsk} \cap \neg\mathbf{correctpw})\Big) \cdot \Pr(\neg\mathbf{correctpw}).\end{aligned}$$

Firstly we compute $\Pr(\mathbf{correctpw})$. Consider two types of attacks: undetectable on-line password guessing attacks [10] and off-line password guessing attacks [16]. Note that in our protocol it provides the authentication of A and B to server S. In details, A would compute the correct b_A using his password pw_1, and provides a correct verification value k_{AS}, (and this is the same with B). Thus a malicious input of SA2 query from the adversary would cause a wrong b_A and hence a wrong verification (this can be detected). In other word the adversary can not cause **correctkBS** or **correctkAS** to occur within a few attempts.

And for off-line password guessing attack, let **correctpwoff** be the event that the adversary can correctly guess a password off-line and causes **correctkSA** event or **correctkSB** event to occur. Let $\mathrm{Succ}_{R_q}^{\mathrm{PWE}}(\mathcal{A})$ be the event that $\mathcal{A}$ solves the PWE problem. Let $\Pr[\mathrm{Succ}_{R_q}^{\mathrm{PWE}}(t, N)] = max_{\mathcal{A}}\{\Pr[\mathrm{Succ}_{R_q}^{\mathrm{PWE}}(\mathcal{A})]\}$, where the maximum is taken over all adversaries of time complexity which is at most t that output a list containing at most N elements of R_q.

Claim 8 *For any adversary $\mathcal{A}$ running in time t, there is a $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$ such that*

$$\Pr[\textbf{correctpwoff}] \leq 2\Pr[\mathrm{Succ}_{R_q}^{\mathrm{PWE}}(t', n_{ro})].$$

Proof. Let E be the event that a **correctkSA** event occurs with an off-line guessing password pw_2. Assume that the adversary can cause E event occurs with non-negligible probability. Then we construct an algorithm D that attempts to solve PWE problem by running $\mathcal{A}$ on a simulation of the protocol P_6. Given (a, X, Y, W), the objective is to find $\tau(X, s_y)$ if $Y = as_y + e_y$ for some $s_y, e_y \xleftarrow{\$} \chi_\beta$. D simulates P_6 for $\mathcal{A}$ with following changes:

- In an **Execute**(A, i, B, j, S, t) query, set $m_B = X + as_g + e_g$ where $s_g, e_g \xleftarrow{\$} \chi_\beta$ and $p_B = Y + as_f + e_f$ where $s_f, e_f \xleftarrow{\$} \chi_\beta$;
- When $\mathcal{A}$ finishes, for every $H_2(\langle A, B, S, p_A, \sigma_B \rangle)$ query, where m_B, p_A, p_B and w_B were generated in an **Execute**(A, i, B, j, S, t) query, and an $H_1(pw_2)$ query returned $\eta = as_\eta + e_\eta$, $\sigma_B = \mathrm{rec}(2v_1, w_B)$ with $v_1 = b_B s_B + e'_B$, $b_B = m_B - \eta$, and there is:

$$\begin{aligned} v_1 = b_B s_B + e'_B &= (X + as_g + e_g - \eta)(s_y + s_f) + e'_B \\ &= X s_y + (as_g + e_g - \eta)s_y + (X + as_g + e_g - \eta)s_f + e'_B \\ &\approx X s_y + Y \cdot (s_g - s_\eta) + (X + as_g + e_g - \eta)s_f + e'_B \end{aligned}$$

  So we have $X s_y \approx v_1 - Y \cdot (s_g - s_\eta) - (X + as_g + e_g - \eta)s_f - e'_B$. Let

$$\sigma' = \mathrm{rec}(2(v_1 - Y \cdot (s_g - s_\eta) - (X + as_g + e_g - \eta)s_f - e'_B), W).$$

  and add the value of σ' to the list of possible values for $\tau(X, s_y)$.

Note that the simulation sets $p_B = Y + (as_f + e_f)$ instead of $p_B = as_{pb} + e_{pb}$ which is distinguishable if there is anyone who can solve the decision version of RLWE problem. It is the same for the setting of m_B. Then if E occurs, D adds the correct $\tau(X, s_y)$ to the list with non-negligible advantage. Such a simulation D is indistinguishable from P_6 until E occurs or the decision version of DRLWE problem is solved with non-negligible advantage. If E occurs, D adds the correct $\tau(X, s_y)$ to the list with non-negligible advantage. After E occurs, the simulation would be distinguishable from P_6. But we do make the assumption that $\mathcal{A}$ still follows the appropriate time and query bounds even if $\mathcal{A}$ distinguishes the simulation from P_6. And for event **correctkSB**, the proof is the same.

If the running time of simulator is t', and they creates a list of size n_{ro} with advantage ϵ. Note that $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$, the claim follows from the fact that $\Pr[\mathrm{Succ}_{R_q}^{\mathrm{PWE}}(D)] \leq \Pr[\mathrm{Succ}_{R_q}^{\mathrm{PWE}}(t', n_{ro})]$. □

Therefore, we can only consider the detectable on-line attacks. In most existing PAKE studies, passwords are assumed to follow a uniformly random distribution, and in this model $\Pr(\textbf{correctpw}) \leq \frac{n_{se}}{L} + negl(\kappa)$, where L is the size of the password dictionary, n_{se} is the max number of $\mathcal{A}$'s active on-line password guessing attempts before a **Corrupt** query and $negl()$ is a negligible function. However, Ding Wang and Ping Wang [29] introduced CDF-Zipf model, which is more consistent with the real world attacks and we prefer this model in our analysis. In this model, Pr(**correctpw**)$\leq C \cdot n_{se}^{s} + negl(\kappa)$ for the Zipf parameters C and s which is depended on the password space L. That is Pr(**correctpw**)$\leq C \cdot n_{se}^{s} + 2\Pr[Succ_{R_q}^{\mathrm{PWE}}(t', n_{ro})]$.

Next we also need that probability that **correctsk** event occurs is negligible.

Claim 9 *For any adversary $\mathcal{A}$ running in time t, there is a $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$ such that*

$$\Pr[\mathbf{correctsk}] \leq 2\Pr[\mathrm{Succ}_{R_q}^{\mathrm{PWE}}(t', n_{ro})].$$

Proof. Assume the adversary can cause event **correctsk** occurs with non-negligible probability, we construct an algorithm D which attempts to solve PWE by running the adversary on a simulation of the protocol P_6. Given (a, X, Y, W), D chooses a random $d \in \{1, \ldots, n_{se}\}$ and simulates P_6 for the adversary with following changes:

- In the dth CAA1 query, say to a client instance $\Pi_A^{i'}$, with input B, S, set $p_A \leftarrow Y$.
- In a SA2 query to a server instance Π_S^t, with input $\langle p_A, p_B, k_{AS}, k_{BS}, w_A, w_B\rangle$ where there was a CAA1 query to $\Pi_A^{i'}$ (i.e., the instance with the dth CAA0 query) with input m_A, p_B, k_{BS}, w_B and output $\langle p_A, p_B, k_{AS}, k_{BS}, w_A, w_B\rangle$, and a query to Π_B^j with input m_A, m_B and output m_A, p_B, k_{BS}, w_B, set $c_A \leftarrow X + as_g + e_g$ where $s_g, e_g \xleftarrow{\$} \chi_\beta$.
- Tests for **correctsk** and **pairedskguess** are not made. In particular, unpaired client instances that receive a CAA2 query abort and unpaired client instances that receive a CBA2 query abort. Also, $H_l(\cdot)$ queries always return a value which is uniformly chosen from $\{0,1\}^\kappa$.
- When $\mathcal{A}$ finishes, for every $H_l(\langle A, B, S, m_A, m_B, p_A, p_B, \sigma\rangle)$ query, $l \in \{3, 4, 5\}$ where p_A was generated by $\Pi_A^{i'}$, p_B, w were generated by Π_B^j and c_A, c_B were generated by a server instance Π_S^t, respectively, $\sigma = \mathrm{rec}(2v_A, w)$, $v_A = c_A s_A$,

we can see that,

$$\begin{aligned} v_A = c_A s_A &= (X + as_g + e_g)s_y \\ &= Xs_y + as_g s_y + e_g s_y \\ &\approx Xs_y + Ys_g \end{aligned}$$

So $Xs_y \approx v_A - Ys_g$. And,

$$\sigma' = \mathrm{rec}(2(v_A - Ys_g), W)$$

Finally, add the value of σ' to the list of possible values for $\tau(X, s_y)$.

Note that such a simulation D is indistinguishable from P_6 until a **correctsk** event or a **pairedpwguess** event event occurs, or $\mathcal{A}$ makes a **Corrupt** query.

But this does not change the fact that **pairedpwguess** event occurs with negligible probability. When a **correctsk** event occurs, D adds the correct $\tau(X, s_y)$ to the list with non-negligible probability. Hence for $t' = O(t+(n_{ro}+n_{se}+n_{ex})t_{exp})$, Pr(**correctsk**)$\leq$ $2\Pr[\mathrm{Succ}_{R_q}^{\mathrm{PWE}}(t', n_{ro})]$, and obviously Pr(**correctsk**|$\neg$ **correctpw**)$\leq$ $2\Pr[\mathrm{Succ}_{R_q}^{\mathrm{PWE}}(t', n_{ro})]$ □

Now we analysis the value of $\Pr(\mathrm{Succ}_{P_7}^{ake}(\mathcal{A})|\neg$**correctsk** $\cap$ $\neg$**correctpw**). Here we note that suppose **correctsk** event doesn't occur, and the password security is not broken, then the adversary succeeds if and only if when $\mathcal{A}$ makes a **Test** query to a fresh instance Π_U^i, and he guesses the bit used in the **Test** query. Next we will prove that the view of $\mathcal{A}$ is independent of sk_U^i, and then the probability of the adversary success is exactly $\frac{1}{2}$.

Firstly we examine **Reveal** queries here. For **Reveal** queries, we know that if Π_U^i is fresh, there could be no one for **Reveal**(U, i) query. And suppose Π_U^i is partnered with some $\Pi_{U'}^j$, there is no **Reveal**(U', j) query. Furthermore, since sid includes the exchanged messages (m_A, m_B, p_A, p_B), if more than a single client A instance and a single client B instance accept with a same sid, the adversary fails (by protocol P_2). Hence the output of **Reveal** queries is not dependent on sk_U^i.

Second for $H_5(\cdot)$ queries, note that from P_5, until a **correctpw** event or a **Corrupt** query, there will be no unpaired client or server instance terminates, and this means that an instance may only be fresh and receives a **Test** query if it is partnered. However by P_6 if Π_U^i is partnered by an $H_5(\cdot)$ query will never reveal sk_U^i. Therefore, the view of $\mathcal{A}$ is independent of sk_U^i and the probability of success is exactly $\frac{1}{2}$. So,

$$
\begin{aligned}
\Pr(\mathrm{Succ}_{P_7}^{ake}(\mathcal{A})) \leq & \Pr(\mathbf{correctpw}) + \Big(\Pr(\mathbf{correctsk}|\neg\mathbf{correctpw}) \\
& + \Pr(\mathrm{Succ}_{P_7}^{ake}(\mathcal{A})|\neg\mathbf{correctsk} \cap \neg\mathbf{correctpw}) \\
& \cdot \Pr(\neg\mathbf{correctsk} \cap \neg\mathbf{correctpw})\Big) \cdot \Pr(\neg\mathbf{correctpw}) \\
= & \Pr(\mathbf{correctpw}) + \Big(\Pr(\mathbf{correctsk}|\neg\mathbf{correctpw}) \\
& + \frac{1}{2} \cdot (1 - \Pr(\mathbf{correctsk}|\neg\mathbf{correctpw}))\Big) \cdot \Pr(\neg\mathbf{correctpw}) \\
\leq & \frac{1}{2} + \frac{1}{2}\Pr(\mathbf{correctpw}) + \frac{1}{2}\Pr(\mathbf{correctsk}) \\
= & \frac{1}{2} + \frac{1}{2}C \cdot n_{se}^s + 2 \cdot \Pr[\mathrm{Succ}_{R_q}^{\mathrm{PWE}}(t', n_{ro})]
\end{aligned}
$$

The theorem follows from Claim 1-9 . □

5 Concrete Parameters and Implementation of RLWE-3PAK

In this section, we present our choices of parameters and outline the performance of our RLWE-3PAK.

Here we use the fact of the product of two Gaussian distributed random values that are stated in [32]. Let $x, y \in R$ be two polynomials with degree of n, and the coefficients of x and y are distributed according to discrete Gaussian distribution with parameter β_x, β_y, respectively. Then the individual coefficients of the polynomial xy are approximately normally distributed around zero with parameter $\beta_x \beta_y \sqrt{n}$. Hence for $||v_B - c_A s_A||_\infty = ||e_A s_B s_S + e_1 s_B + e''_B - e_B s_A s_S - e_2 s_A||_\infty < \frac{q}{8}$, by applying Lemma 1 we have that $||v_B - c_A s_A||_\infty > 6\sqrt{2n^2\beta^6 + 2n\beta^4 + \beta^2}$ with probability approximating 2^{-162}. Hence we set $6\sqrt{2n^2\beta^6 + 2n\beta^4 + \beta^2} < \frac{q}{8}$, then the two clients will end with the same key with overwhelming probability. And such choices of parameter also make $||v_2 - p_A s_f||_\infty < \frac{q}{8}$ and $||v_1 - p_B s_g||_\infty < \frac{q}{8}$ with overwhelming probability be satisfied.

We take $n = 1024$, $\beta = 8$ and $q = 2^{32} - 1$. Our implementations are written in C without any parallel computations or multi-thread programming techniques. The program is run on a 3.5GHz Intel(R) Core(IM) i7-4770K CPU and 4GB RAM computer running on Ubuntu 16.04.1 64 bit system. The timings for server and clients actions of the authentication protocol are presented in Table 1.

Table 1. Timings of proof-of-concept implementations in ms

B initiation	S first response	B first response	A first response
<0.001 ms	0.165 ms	1.960 ms	1.779 ms
S second response	B second response	A second response	B finish
2.030 ms	2.195 ms	2.088 ms	<0.001 ms

Sampling and multiplication operations are the mainly time cost. The sampling technique used in our protocol is the same with [5], which use the Discrete Gaussian to approximate the continuous Gaussian. And to improve performance, we have used multiplication with FFT. Note that by the proof of concept implementation, our protocol can be very efficient.

6 Conclusion

In this paper, we propose a 3PAKE protocol based on RLWE: RLWE-3PAK. We provide a full proof of security of our protocol in the random oracle model. Finally, we construct a proof-of-concept implementation to examine the efficiency of our protocol. The performance results indicate that our protocol is very efficient and practical. Since some literature [4] show that it is delicate to prove quantum resistance with random oracle. It is meaningful to design an efficient 3PAKE protocol without random oracle heuristics in the future.

Acknowledgments. This article is supported by The National Key Research and Development Program of China (Grant No. 2017YFA0303903), National Cryptography Development Fund (No. MMJJ20170121), and Zhejiang Province

Key R&D Project(No. 2017C01062). Authors thank Aijun Ge for discussions and the anonymous ISPEC'19 reviewers for helpful comments.

References

1. Abdalla, M., Fouque, P., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Public Key Cryptography - PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005, Proceedings. pp. 65–84 (2005)
2. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. pp. 139–155 (2000)
3. Bellare, M., Rogaway, P.: Provably secure session key distribution: the three party case. In: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA. pp. 57–66 (1995)
4. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. pp. 41–69 (2011)
5. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015. pp. 553–570 (2015)
6. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using diffie-hellman. In: Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. pp. 156–171 (2000)
7. Chang, T.Y., Hwang, M., Yang, W.: A communication-efficient three-party password authenticated key exchange protocol. Inf. Sci. **181**(1), 217–226 (2011)
8. Ding, J.: A simple provably secure key exchange scheme based on the learning with errors problem. IACR Cryptology ePrint Archive **2012**, 688 (2012), `http://eprint.iacr.org/2012/688`
9. Ding, J., Alsayigh, S., Lancrenon, J., RV, S., Snook, M.: Provably secure password authenticated key exchange based on RLWE for the post-quantum world. In: Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings. pp. 183–204 (2017)
10. Ding, Y., Horster, P.: Undetectable on-line password guessing attacks. Operating Systems Review **29**(4), 77–86 (1995)
11. Dongna, E., Cheng, Q., Ma, C.: Password authenticated key exchange based on RSA in the three-party settings. In: Provable Security, Third International Conference, ProvSec 2009, Guangzhou, China, November 11-13, 2009. Proceedings. pp. 168–182 (2009)
12. Katz, J., Vaikuntanathan, V.: Smooth projective hashing and password-based authenticated key exchange from lattices. In: Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. pp. 636–652 (2009)

13. Kim, H.S., Choi, J.: Enhanced password-based simple three-party key exchange protocol. Computers & Electrical Engineering **35**(1), 107–114 (2009)
14. Krawczyk, H.: HMQV: A high-performance secure diffie-hellman protocol. In: Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. pp. 546–566 (2005)
15. Lee, T., Hwang, T., Lin, C.: Enhanced three-party encrypted key exchange without server public keys. Computers & Security **23**(7), 571–577 (2004)
16. Lin, C., Sun, H., Hwang, T.: Three-party encrypted key exchange: Attacks and A solution. Operating Systems Review **34**(4), 12–20 (2000)
17. Lin, C., Sun, H., Steiner, M., Hwang, T.: Three-party encrypted key exchange without server public-keys. IEEE Communications Letters **5**(12), 497–499 (2001)
18. Lu, R., Cao, Z.: Simple three-party key exchange protocol. Computers & Security **26**(1), 94–97 (2007)
19. Lyubashevsky, V.: Lattice signatures without trapdoors. IACR Cryptology ePrint Archive **2011**, 537 (2011), `http://eprint.iacr.org/2011/537`
20. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings. pp. 1–23 (2010)
21. MacKenzie, P.: The pak suite: Protocols for password-authenticated key exchange. In: DIMACS Technical Report 2002-46 (2002). p. 7 (2002)
22. MacKenzie, P.D., Patel, S., Swaminathan, R.: Password-authenticated key exchange based on RSA. In: Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings. pp. 599–613 (2000)
23. Peikert, C.: Lattice cryptography for the internet. In: Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings. pp. 197–219 (2014)
24. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 34:1–34:40 (2009)
25. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997)
26. Steiner, M., Tsudik, G., Waidner, M.: Refinement and extension of encrypted key exchange. Operating Systems Review **29**(3), 22–30 (1995)
27. Sun, H., Chen, B., Hwang, T.: Secure key agreement protocols for three-party against guessing attacks. Journal of Systems and Software **75**(1-2), 63–68 (2005)
28. Wang, D., Jian, G., Wang, P.: Zipf's law in passwords. IACR Cryptology ePrint Archive **2014**, 631 (2014), `http://eprint.iacr.org/2014/631`
29. Wang, D., Wang, P.: On the implications of zipf's law in passwords. In: Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part I. pp. 111–131 (2016)
30. Wang, D., Zhang, Z., Wang, P., Yan, J., Huang, X.: Targeted online password guessing: An underestimated threat. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 1242–1254 (2016)
31. Zhang, J., Yu, Y.: Two-round PAKE from approximate SPH and instantiations from lattices. In: Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information

Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III. pp. 37–67 (2017)

32. Zhang, J., Zhang, Z., Ding, J., Snook, M., Dagdelen, Ö.: Authenticated key exchange from ideal lattices. In: Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II. pp. 719–751 (2015)