Provably Secure Password Authenticated Key Exchange Based on RLWE for the Post-Quantum World

Jintai Ding[1], Saed Alsayigh[1], Jean Lancrenon[2], Saraswathy RV[1], and Michael Snook[1]

[1] University of Cincinnati
[2] University of Luxembourg

Abstract. Authenticated Key Exchange (AKE) is a cryptographic scheme with the aim to establish a high-entropy and secret session key over a insecure communications network. *Password*-Authenticated Key Exchange (PAKE) assumes that the parties in play share a simple password, which is cheap and human-memorable and is used to achieve the authentication. PAKEs are practically relevant as these features are extremely appealing in an age where most people access sensitive personal data remotely from more-and-more pervasive hand-held devices. Theoretically, PAKEs allow the secure computation and authentication of a high-entropy piece of data using a low-entropy string as a starting point. In this paper, we apply the recently proposed technique introduced in [29] to construct two lattice-based PAKE protocols enjoying a very simple and elegant design that is an parallel extension of the class of Random Oracle Model (ROM)-based protocols PAK and PPK [11, 38], but in the lattice-based setting. The new protocol resembling PAK is three-pass, and provides *mutual explicit authentication*, while the protocol following the structure of PPK is two-pass, and provides *implicit authentication*. Our protocols rely on the Ring-Learning-with-Errors (RLWE) assumption, and exploit the additive structure of the underlying ring. They have a comparable level of efficiency to PAK and PPK, which makes them highly attractive. We present a preliminary implementation of our protocols to demonstrate that they are both efficient and practical. We believe they are suitable quantum safe replacements for PAK and PPK.

Key words: Diffie-Hellman, Key Exchange, Authenticated, PAKE, RLWE

1 Introduction

Password-authenticated key exchange and dictionary attacks. Authenticated Key Exchange (AKE) is a cryptographic service with the aim of allowing several entities to jointly establish a high-entropy and secret session key over a completely insecure communications network. That the protocol includes authentication of the purported peers is essential to prevent man-in-the-middle attacks. In order to achieve this, it is required that some form of long-term authentication material already be in place prior to the exchange occurring. For instance, the entities could each have their own public-key/secret-key pair (e.g. for STS [16], or HMQV [33]), certified by a trusted authority, or they can all share a single symmetric key specifically dedicated to running an AKE with which to establish other session keys (e.g. the protocols in [6]).

In *Password*-Authenticated Key Exchange (PAKE), it is assumed that the parties in play share a simple password. This differs from the shared-symmetric-key case in that the password is not necessarily a cryptographically strong piece of data. Indeed, most passwords have very low entropy so that they can retain their main advantage over strong keying material: they are cheap and human-memorable. Moreover, these features are extremely appealing in an age where most people access sensitive personal data remotely from more-and-more pervasive hand-held devices. Thus, PAKEs are practically relevant. From a theoretical standpoint, they are quite unique in that they allow the secure computation and authentication of a high-entropy piece of data using a low-entropy string as a starting point.

From a security modeling perspective, the use of passwords as authentication material presents specific challenges. A password's low entropy makes it easy to discover by brute force, assuming an attacker can get its hands on a piece of password-dependent data that a guess can be checked against. Such attacks are known as *dictionary attacks*. There are two types: In an *offline* attack, the adversary observes protocol runs - possibly also interacting with the entities involved - and then goes offline to do password testing privately. To avoid this, the protocol messages and session keys must look computationally independent from the password. In an *online attack*, the attacker needs to be *actively involved* in a protocol interaction in order to determine the verdict on its guess(es). The most natural online attack available is to simply run the protocol with an arbitrary password guess as input, and observe whether the protocol run succeeds or fails. It is clear that this attack is unavoidable; thus a PAKE must be designed such that the adversary can test at most a *constant* (ideally, *one*) number of passwords per online interaction.

PAKEs and the post-quantum world. Based on the above reasons, PAKEs have been very heavily studied in the past three decades. Adequate formal security models have appeared [5, 11], and a plethora of protocols have been designed and analyzed (e.g., [12, 38, 28, 32, 1]). The current pool of practical protocols[3] can essentially be classified into two categories: the first we shall call the class of *Random Oracle Model* (ROM)-based PAKEs (such as [5, 12, 38, 13, 27, 8, 3]), and the second, the class of *Common Reference String* (CRS)-based PAKEs (such as [30, 14, 20, 24, 32]). Roughly, the protocols in the first category have very simple and elegant designs, but rely crucially on the ROM [7] for their proofs of security, while the protocols in the second category use sophisticated cryptographic tools[4] to achieve standard-model security (assuming a CRS is in place[5]). The bottom line is that the simplicity and efficiency of ROM-based protocols (and the fact that if carefully instantiated, they are not known to have been broken) makes them much more attractive for concrete deployment than CRS-based ones.

Searching for tools that can resist against adversaries attacking using a quantum computer is currently one of the fundamental issues in cryptographic research. Indeed, the security of all public-key algorithms based on classical hard problems will no longer be assured as soon as a quantum computer of satisfactory size exists. In the US, the National Security Agency (NSA) [41] published a webpage announcing preliminary plans for transitioning from its suite B cryptographic tools to quantum resistant algorithms, which are specified by the National Institute of Standards and Technology (NIST) and are used by NSA's Information Assurance Directorate in solutions approved for protecting classified and unclassified National Security Systems (NSS). It is clear that the effort to develop quantum-resistant technologies is intensifying and, in the immediate future, NIST, which has the authority to establish the security standards of the US government, shall have a call to select post-quantum cryptosystem standards. Regardless of which of the aforementioned categories they belong to, most known PAKEs rest their security on either group-type or factoring-type complexity assumptions, making them unsuitable in a possibly upcoming post-quantum world. Therefore, searching for PAKEs that can be based on provably secure *lattice* assumptions is natural. In the current literature, as far as we know one single PAKE stands out precisely for this reason: the Katz–Vaikuntanathan protocol [31] relies instead on lattice-type assumptions for its security. Unfortunately, it is CRS-based, and therefore not very efficient.

[3] Some impractical yet complexity-theoretically efficient protocols have been studied, for theoretical reasons. See e.g. [22, 40, 23].

[4] In particular, they use *universal hash proof systems* [15] over complex languages.

[5] A CRS is essentially a publicly available string to which a secret trapdoor is theoretically associated, but never used by protocol participants. During a proof of security, the simulator gets access to this trapdoor.

1.1 Our contributions

In this paper, we propose two lattice-based PAKE protocols enjoying a very simple and elegant design that is extremely similar to that of the class of ROM-based protocols. More specifically, our protocols can be viewed as direct analogues of the PAK and PPK [11, 38] protocols in the lattice-based setting. The protocol resembling PAK is three-pass, and provides *mutual explicit authentication*, while the protocol following the structure of PPK is two-pass, and provides *implicit authentication*. Most importantly, our protocols have a comparable level of efficiency to PAK and PPK, which makes them highly attractive.

The starting point for our construction is the recently proposed technique introduced in [29], and used in [48] to design a lattice-based variant of the HMQV protocol. As in the latter paper, our protocols rely on the Ring-Learning-with-Errors (RLWE) assumption, and exploit the additive structure of the underlying RLWE ring. We therefore obtain two protocols which are suitable quantum-safe replacements for PAK and PPK.

It is indeed true that we can build the PAKE protocol using RLWE in a rather straightforward manner. Though the general structure of the proofs for our protocols is very similar to that of the original PAK protocol's security proof, where the security of PAK relies on an adversary being unable to solve the Diffie–Hellman Problem, the techniques used in our paper are intricate and completely different.

We manage to establish a full proof of security for our RLWE-PAK protocol, in the classic security model proposed by Bellare et al. [5]. To simplify the proof, first we define the Pairing with Errors problem, which can be reduced to the RLWE problem. This new problem is used multiple times in the proof, which allows us to build intermediate steps to fill the gap of the proof, which did not exist in the proof for the classical PAKE.

The complete replacement of the Diffie–Hellman core of PAK with the new lattice-based core means that the distinguishers used in the PAK proof have to be completely replaced, and are of no use in constructing the new distinguishers. The new distinguishers have to compensate for the presence of the password in the protocol without being able to directly remove its influence, as they have no access to the value of the password itself. In the proof, there are three places where we have to build distinguishers to solve the PWE problem. Since such distinguishers are completely new and subtle, we need to use novel methods to construct them. Only by applying these new distinguishers are we able to link the security directly to the PWE problem.

From the construction in [29], we can use the same idea to build in a completely parallel way a PAKE using the LWE problem instead of the RLWE problem. Here we need to use matrix multiplications, and need to make sure that the order of multiplications is correct.

We created a proof-of-concept implementation of our new PAKE to show that our new PAKE is efficient and practical, as well as an implementation of the implicit version to show it is efficient and practical as well.

1.2 Related work

AKE protocol research is far too vast to describe in full, hence we only survey those portions of it most relevant to this work. These are PAKE, and AKE based on lattice-type assumptions. We also only consider protocols in the two-party setting.

PAKE protocols and security models. PAKE was essentially invented by Bellovin and Merritt in [8]. The authors raised the problem of dictionary attacks in this particular setting, proposed some protocols - most notably the Encrypted Key Exchange (EKE) protocol - and offered an informal security analysis of their designs. Jablon [27] later proposed another protocol - Simple Password Exponential Key Exchange (SPEKE) - avoiding some of the pitfalls of EKE, but again with only an informal analysis.

The search for good security models began with the work of Lucks [35] and Halevi and Krawczyk [25]. Laying down adequate foundations for the provable security of PAKE was a particularly subtle task, since one cannot prevent the adversary from guessing and trying out candidate passwords in on-line impersonation attempts, and the small size of the dictionary means that the adversary's natural advantage in succeeding at this is *non-negligible*. Good models capturing this phenomenon were finally exhibited by Bellare et al. [5] and Boyko et al. [11], building respectively on the AKE models proposed by Bellare et al. in [6] and Shoup in [44]. The model in [5] was further refined by Abdalla et al. in [4]. The notion of universally composable PAKE has also since been introduced by Canetti et al. [14].

A great deal of protocols have been proposed and analyzed, especially since the apparition of adequate security models. Some extremely efficient examples include the protocols in [5, 12, 13, 11, 38, 37, 34, 27, 26, 2, 3]. On one hand, these are mostly two-or-three-pass protocols, with very few group operations. For instance, the explicitly authenticated PAK protocol in [38] is three-pass, and sends 2 group elements (and 2 confirmation bitstrings of length linear in the security parameter) in total over the network. It also requires a total of only 4 exponentiations, and 2 group multiplications. On the other hand, these protocols' security is very heavily reliant on idealized assumptions[6]. In 2001, a milestone was reached with the work of Katz et al. [30], which showed that it is possible to provably realize PAKE in a *practical way without idealized assumptions*, but at the expense of having a CRS in place. Many works generalizing and optimizing this result followed, such as [28, 24, 32, 20, 19], all using a CRS. It was further shown in [14] that without idealized assumptions, universally composable PAKE is possible only if some other trusted setup - e.g. a CRS - is in place. However, all of these CRS-using protocols are generally much less practical than the ROM-using ones mentioned before. While it is possible to achieve a low number of passes using a CRS - e.g. [32] is a two-pass protocol - the number of group computations and elements sent is typically high. To our knowledge, the latest techniques [2] discovered to reduce this still do not beat ROM-based PAKEs in efficiency. For instance, Abdalla et al. [2] report on being able to bring the total group element and exponentiation counts of the Groce-Katz protocol [24] down to 6 and 18 respectively, and those of [32] down to 10 and 28 respectively.

Finally, some work has been devoted to determining if PAKE can be efficiently realized in a reasonable security model with *neither idealized assumptions nor any form of trusted setup*. Goldreich et al. [22] were the first to answer in the affirmative, but assuming non-concurrent protocol executions. Their work was followed up by Nguyen et al. [40], who found a more efficient construction, but in a weaker model. Later, Jain et al. [23] were able to further lift the restriction on concurrent executions. These works are viewed as being mainly of theoretical interest, as the protocols, although *theoretically* efficient, are far less practical then even the CRS-based protocols.

AKE from lattices. Several works have begun addressing the problem of finding AKE protocols based on lattice-type assumptions. The protocols in [43, 10, 17, 18] are essentially lattice-based instantiations of generic constructions that use key-encapsulation mechanisms to construct AKEs. Recently, this pattern was broken by the work of Zhang et al. [48], where a RLWE analogue of the unauthenticated Diffie-Hellman protocol proposed by Ding et al. [29] was leveraged to build a kind of RLWE version of the HMQV protocol.

In all of these works, the authentication mechanism used is reliant on the deployment of a *public-key infrastructure*. In the case of password authentication, the only known protocol to this day appears to be that of Katz et al. [31]. It too can be viewed as a lattice-based instantiation of a generic construction. This is because most known CRS-based frameworks for PAKE make use of an encryption scheme that is both *secure against adaptive chosen-ciphertext attacks* and

[6] The ROM is one of them; another is the *ideal cipher* model, see [5].

equipped with a universal hash proof system [15], and the heart of [31] is essentially a lattice-based instantiation of such a scheme.

2 Preliminaries

2.1 Security model

Here, we review the security model from [5]. It basically models the communications between a fixed number of users - which are clients and servers - over a network that is fully controlled by a probabilistic, polynomial-time adversary $\mathcal{A}$. Users are expected to both establish and use session keys over this network. Therefore, $\mathcal{A}$ is given access to a certain number of queries which reflect this usage. It may initialize protocol communications between user instances of its choice, deliver any message it wants to these instances, and observe their reaction according to protocol specification. It may also reveal session keys established by instances, thereby modeling loss of keys through higher-level protocol use. Finally, we even allow the adversary to obtain user passwords, in order to capture forward secrecy. We describe this formally now.

Let P be a PAKE protocol.

Security game. An algorithmic game initialized with a security parameter k is played between a challenger $\mathcal{CH}$ and a probabilistic polynomial time adversary $\mathcal{A}$. $\mathcal{CH}$ will essentially run P on behalf of honest users, thereby simulating network traffic for $\mathcal{A}$.

Users and passwords. We assume a fixed set $\mathfrak{U}$ of *users*, partitioned into two non-empty sets $\mathfrak{C}$ of *clients* and $\mathfrak{S}$ of *servers*. We also assume some fixed, non-empty dictionary D of size L. Before the game starts, for each $\mathcal{C} \in \mathfrak{C}$ a password $pw_{\mathcal{C}}$ is drawn uniformly at random from D and assigned to $\mathcal{C}$ outside of $\mathcal{A}$'s view. For each server $\mathcal{S} \in \mathfrak{S}$, we set $pw_{\mathcal{S}} := \big(f(pw_{\mathcal{C}})\big)_{\mathcal{C}}$, where $\mathcal{C}$ runs through all of $\mathfrak{C}$, and f is some efficiently computable one-way function specified by P. (In our case, f will be essentially a hash of the password.) $\mathcal{CH}$ also generates P's public parameters on input 1^k, and gives these to $\mathcal{A}$. We assume that $\mathcal{A}$ is polynomial-time in k as well. The game can then begin.

User instances. During the game, to any user $\mathcal{U} \in \mathfrak{U}$ is associated an unlimited number of user instances $\Pi^i_{\mathcal{U}}$, where i is a positive integer. The adversary may activate any of these instances using the queries listed below, causing them to initiate and run the protocol.

At any point in time, an instance $\Pi^i_{\mathcal{U}}$ may *accept*. When this happens, it holds a *Partner IDentity* (PID) $pid^i_{\mathcal{U}}$, a *Session IDentity* (SID) $sid^i_{\mathcal{U}}$, and a *Session Key* (SK) $sk^i_{\mathcal{U}}$. The PID is the identity of the user that instance believes it is talking to. The SK is what $\Pi^i_{\mathcal{U}}$ is aiming to ultimately compute. The SID is a string which uniquely identifies the protocol run and ensuing session in which the SK is to be used in. Without loss of generality, the SID is usually defined as the ordered concatenation of messages sent and received by an instance, except possibly the last message.

Queries. The queries $\mathcal{A}$ may make to any given instance $\Pi^i_{\mathcal{U}}$ during the game are as follows:

- $\mathsf{Send}(\mathcal{U}, i, M)$: Causes message M to be sent to instance $\Pi^i_{\mathcal{U}}$. The instance computes what the protocol P says, updates its state, and gives the output to $\mathcal{A}$. We also assume that $\mathcal{A}$ sees if the query causes $\Pi^i_{\mathcal{U}}$ to accept or terminate.
- $\mathsf{Execute}(\mathcal{C}, i, \mathcal{S}, j)$: Causes P to be executed to completion between $\Pi^i_{\mathcal{C}}$ (where $\mathcal{C} \in \mathfrak{C}$) and $\Pi^j_{\mathcal{S}}$ (where $\mathcal{S} \in \mathfrak{S}$) and hands $\mathcal{A}$ the execution's transcript.
- $\mathsf{Reveal}(\mathcal{U}, i)$: Returns the SK $sk^i_{\mathcal{U}}$ held by $\Pi^i_{\mathcal{U}}$ to $\mathcal{A}$.
- $\mathsf{Test}(\mathcal{U}, i)$: For this query to be valid, instance $\Pi^i_{\mathcal{U}}$ must be *fresh*, as defined below. If this is the case, the query causes a bit b to be flipped. If $b = 1$, the actual SK sk^U_i is returned to $\mathcal{A}$; otherwise a string is drawn uniformly from the SK space and returned to $\mathcal{A}$. Note that this query can be asked only once during the game.
- $\mathsf{Corrupt}(\mathcal{U})$: Returns $\big(f(pw_{\mathcal{C}})\big)_{\mathcal{C}}$ to $\mathcal{A}$ if $\mathcal{U} \in \mathfrak{S}$ else returns $pw_{\mathcal{U}}$ to $\mathcal{A}$.

Ending the game. Eventually, $\mathcal{A}$ ends the game, and outputs a single bit b'. We return to the use of this bit in the definition of security below.

Partnering and freshness. In order to have a meaningful definition of security, we need to introduce the notions of instance *partnering* and instance *freshness*. Essentially, an instance $\Pi_{\mathcal{U}}^{i}$ is *fresh* if the adversary does not already know that instance's SK through trivial means provided by the security model's queries, for instance by using a Reveal query on the instance in question. Furthermore, since instances are supposed to be sharing keys under normal circumstances, it also makes sense to consider freshness destroyed if an instance's proper communicant has been revealed as well. Thus, we need to formally define what this proper communicant is:

Definition 1. *Let $\Pi_{\mathcal{U}}^{i}$ and $\Pi_{\mathcal{V}}^{j}$ be two instances. We shall say that $\Pi_{\mathcal{U}}^{i}$ and $\Pi_{\mathcal{V}}^{j}$ are* partnered *if* ***i)*** *one is in $\mathfrak{C}$ and one is in $\mathfrak{S}$,* ***ii)*** *both have accepted,* ***iii)*** *$pid_{\mathcal{U}}^{i} = \mathcal{V}$ and $pid_{\mathcal{V}}^{j} = \mathcal{U}$,* ***iv)*** *$sid_{\mathcal{U}}^{i} = sid_{\mathcal{V}}^{j} =: sid$ and this value is not null, and* ***v)*** *no other instance accepts with a SID of sid.*

Capturing the notion of *forward secrecy* requires freshness to be carefully defined around the *corrupt* query. Intuitively, if a corruption occurs after an instance has had a correct exchange with a proper partner, then those instances' shared session key should still remain secure. However, we cannot guarantee anything for an instance that has interacted with the adversary after a corruption. More formally:

Definition 2. *An instance $\Pi_{\mathcal{U}}^{i}$ is* fresh *if none of the following events occur:* ***i)*** $\mathsf{Reveal}(\mathcal{U}, i)$ *was queried,* ***ii)*** *a* $\mathsf{Reveal}(\mathcal{V}, j)$ *was queried, where $\Pi_{\mathcal{V}}^{j}$ is $\Pi_{\mathcal{U}}^{i}$'s partner, if it has one, or* ***iii)*** $\mathsf{Corrupt}(\mathcal{V})$ *was queried for some $\mathcal{V}$ before the* Test *query and a* $\mathsf{Send}(\mathcal{U}, i, M)$ *query occurs for some M.*

Definition of security. We now turn to actually measuring the adversary's success rate in breaking P. $\mathcal{A}$'s objective is to tell apart a random string from a true SK belonging to a fresh instance. This is the whole purpose of the Test query. Let $Succ_{\mathsf{P}}^{ake}(\mathcal{A})$ be the event:

"$\mathcal{A}$ makes a $\mathsf{Test}(\mathcal{U}, i)$ query where $\Pi_{\mathcal{U}}^{i}$ has terminated and is fresh and $b' = b$, where b is the bit selected when $\mathsf{Test}(\mathcal{U}, i)$ was made, and b' is the bit $\mathcal{A}$ output at the end of the game."

$\mathcal{A}$'s *advantage* is then defined as:

$$Adv_{\mathsf{P}}^{ake}(\mathcal{A}) = 2Pr[Succ_{\mathsf{P}}^{ake}(\mathcal{A})] - 1$$

It is easy to see that if we have two protocols P and P' then for any adversary $\mathcal{A}$ we have $Pr[Succ_{\mathsf{P}}^{ake}(\mathcal{A})] = Pr[Succ_{\mathsf{P}'}^{ake}(\mathcal{A})] + \epsilon$ if and only if $Adv_{\mathsf{P}}^{ake}(\mathcal{A}) = Adv_{\mathsf{P}'}^{ake}(\mathcal{A}) + 2\epsilon$.

2.2 Ring Learning with Errors

Ring Learning with Errors. Here, we introduce some notation and recall informally the *Ring Learning with Errors* assumption, introduced in [36]. For our purpose, it will be more convenient to use an assumption we call the Pairing with Errors PWE , which we state formally at the end of the section, and which can easily be shown holds under RLWE.

We denote the security parameter k. Recall that a function f is negligible in k if for every $c > 0$, there exists a N such that $f(k) < \frac{1}{k^c}$ for all $k > N$. The ring of polynomials over $\mathbb{Z}$ (respectively, $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$) we denote by $\mathbb{Z}[x]$ (resp., $\mathbb{Z}_q[x]$). Let $n \in \mathbb{Z}$ be a power of 2. We consider the ring $R = \mathbb{Z}[x]/(x^n + 1)$. For any positive $q \in \mathbb{Z}$, we set $R_q = \mathbb{Z}_q[x]/(x^n + 1)$. For any polynomial y in R or R_q, we identify y with its coefficient vector in $\mathbb{Z}^n$ or $\mathbb{Z}_q^n$, respectively. Recall that for a fixed $\beta > 0$, the *discrete Gaussian distribution over R_q* (parametrized by β)

is naturally induced by that over $\mathbb{Z}^n$ (centered at 0, with standard deviation β). We denote this distribution over R_q by χ_β. More details can be found in [48].

For a fixed $s \in R_q$, let A_{s,χ_β} be the distribution over pairs $(a, as + 2x) \in R_q \times R_q$, where $a \leftarrow R_q$ is chosen uniformly at random and $x \leftarrow \chi_\beta$ is independent of a. The *Ring Learning with Errors* assumption is the assumption that for a fixed s sampled from χ_β, the distribution A_{s,χ_β} is computationally indistinguishable from the uniform distribution on R_q^2, given polynomially many samples.

In doing so we also define the norm of a polynomial to be the norm of its coefficient vector. Then we have the following useful facts:

Lemma 1. *Let R be defined as above. Then, for any $s, t \in R$, we have $\|s \cdot t\| \le \sqrt{n} \cdot \|s\| \cdot \|t\|$ and $\|s \cdot t\|_\infty \le n \cdot \|s\|_\infty \cdot \|t\|_\infty$.*

Lemma 2 ([39,21]). *For any real number $\alpha = \omega(\sqrt{\log n})$, we have $\Pr_{\mathbf{x} \leftarrow \chi_\alpha}[\|\mathbf{x}\| > \alpha\sqrt{n}] \le 2^{-n+1}$.*

We now recall the Cha and Mod_2 functions defined in [48]. We denote $\mathbb{Z}_q = \{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\}$ and consider the set $E := \{-\lfloor \frac{q}{4} \rfloor, \ldots, \lfloor \frac{q}{4} \rceil\}$, i.e. the "middle" of $\mathbb{Z}_q$. Recall that Cha is the characteristic function of *the complement of* E, which returns 0 if the input in E and 1 if it is not in E, and that $\mathsf{Mod}_2 \colon \mathbb{Z}_q \times \{0,1\} \to \{0,1\}$ is defined as:

$$\mathsf{Mod}_2(v, b) = (v + b \cdot \frac{q-1}{2}) \bmod q \bmod 2.$$

These two functions have a fundamental features which can be seen from the following two lemmas.

Lemma 3 ([48]). *Let n be the security parameter, and let $q = 2^{\omega(\log n)} + 1$ be an odd prime. Let $v \in \mathbb{Z}_q$ be chosen uniformly at random. For any $b \in \{0,1\}$ and any $v' \in \mathbb{Z}_q$, the output distribution of $\mathsf{Mod}_2(v + v', b)$ given $\mathsf{Cha}(v)$ is statistically close to uniform on $\{0,1\}$.*

Lemma 4 ([48]). *Let q be an odd prime, $v \in \mathbb{Z}_q$ and $e \in \mathbb{Z}_q$ such that $|e| < q/8$. Then, for $w = v + 2e$, we have $\mathsf{Mod}_2(v, \mathsf{Cha}(v)) = \mathsf{Mod}_2(w, \mathsf{Cha}(v))$.*

They also can be extended to R_q by applying them coefficient-wise to the coefficients in $\mathbb{Z}_q$ that define the ring elements. In other words, for any ring element $v = (v_0, \ldots, v_{n-1}) \in R_q$ and binary-vector $\mathbf{b} = (b_0, \ldots, b_{n-1}) \in \{0,1\}^n$, we set $\mathsf{Cha}(v) = (\mathsf{Cha}(v_0), \ldots, \mathsf{Cha}(v_{n-1}))$ and $\mathsf{Mod}_2(v, \mathbf{b}) = (\mathsf{Mod}_2(v_0, b_0), \ldots, \mathsf{Mod}_2(v_{n-1}, b_{n-1}))$.

The PWE assumption. We now state the Pairing with Errors (PWE) assumption, under which we prove that our protocols are secure. We return to the general notations of paragraph 2.2, but using the Gaussian distribution χ_β for a fixed $\beta \in \mathbb{R}_+^*$. For any $(X, s) \in R_q^2$, we set $\tau(X, s) = \mathsf{Mod}_2(Xs, \mathsf{Cha}(Xs))$. Let $\mathcal{A}$ be probabilistic, polynomial-time algorithm taking inputs of the form (a, X, Y, W), where $(a, X, Y) \in R_q^3$ and $W \in \{0,1\}^n$, and outputting a list of values in $\{0,1\}^n$. $\mathcal{A}$'s objective will be for the string $\tau(X, s)$ to be in its output, where s is randomly chosen from R_q, Y is a "small additive perturbation" of as, and W is $\mathsf{Cha}(Xs)$ itself. Formally, let

$$Adv_{R_q}^{PWE}(\mathcal{A}) \overset{def}{=} Pr\Big[a \leftarrow R_q; s \leftarrow \chi_\beta; X \leftarrow R_q; e \leftarrow \chi_\beta; \\ Y \leftarrow as + 2e; W \leftarrow \mathsf{Cha}(Xs) : \tau(X, s) \in \mathcal{A}(a, X, Y, W)\Big]$$

Let $Adv_{R_q}^{PWE}(t, N) = \max_{\mathcal{A}} \left\{ Adv_{R_q}^{PWE}(\mathcal{A}) \right\}$, where the maximum is taken over all adversaries of time complexity at most t that output a list containing at most N elements of $\{0,1\}^n$.

The PWE assumption states that for t and N polynomial in k, $Adv_{R_q}^{PWE}(t, N)$ is negligible in k.

We also have decision version of PWE problem that can be defined as follows. Clearly, if DPWE is hard, so is PWE.

Definition 1. *(DPWE) Given $(a, X, Y, W, \sigma) \in R_q \times R_q \times R_q \times \{0,1\}^n \times \{0,1\}^n$ where $w = \mathsf{Cha}K$ for some $K \in R_q$ and $\sigma = \mathsf{Mod}_2(K, w)$. The decision Pairing with Errors problem (DPWE) is to decide whether $K = Xs + 2g$ and $Y = as + 2e$ for some s, g, and e drawn from χ_β, or (K, Y) is uniformly random in R_q^2.*

Before we show the reduction of the DPWE problem to the RLWE problem, we would like to give a definition to what we called the RLWE-DH problem which can be reduced to RLWE problem.

Definition 2. *(RLWE-DH) Let R_q and χ_β be defined as above. Given as input ring elements $a, X, Y,$ and K, where (a, X) is uniformly random in R_q^2, the RLWE-DH problem is to tell if K is $X \cdot s_y + 2g_y$ for some $g_y \leftarrow \chi_\beta$ and $Y = a \cdot s_y + 2e_y$ for some $s_y, e_y \leftarrow \chi_\beta$, or (K, Y) is uniformly random in R_q^2.*

Theorem 1. *Let R_q and χ_β be defined as above. The RLWE-DH problem is hard to solve if RLWE problem is hard.*

Proof. Suppose that RLWE is hard and that there exists an algorithm D which can solve RLWE-DH on input (a, X, Y, K) with non-negligible advantage. This means that D can determine if $K = X \cdot s_y + 2g_y$ for some $g_y \leftarrow \chi_\beta$ and $Y = a \cdot s_y + 2e_y$ for some $s_y, e_y \leftarrow \chi_\beta$, or (K, Y) is uniformly random in R_q^2.

Now given two samples of a RLWE challenge (a_1, b_1) and (a_2, b_2) where both share the same $s \leftarrow \chi_\beta$, using D we can build a distinguisher D that can solve the RLWE problem as follows:

1. Set $(a, X, Y, K) = (a_1, a_2, b_1, b_2)$.
2. Run D on input (a, X, Y, K). By construction, we have:
   - $b_2 = a_2 \cdot s + e_2$ for some $e_2 \leftarrow \chi_\beta$ and $b_1 = a_1 \cdot s + e_1$ for some $e_1 \leftarrow \chi_\beta$.
   - Or b_2 and b_1 are a uniformly random from R_q.

Since D solves the RLWE-DH with non-negligible advantage, then D' solves RLWE with non-negligible advantage advantage as well, which contradicts the hardness of RLWE. Hence if RLWE hard to solve then RLWE-DH is also hard to solve.

□

Now we show the reduction of the DPWE problem to the RLWE-DH problem by the following theorem.

Theorem 2. *Let R_q and χ_β be defined as above. The DPWE problem is hard if the RLWE-DH problem is hard.*

Proof. Suppose that RLWE-DH is hard to solve and assume that there exists an algorithm D which can solve the DPWE problem on input (a, X, Y, w, σ) where for some $K \in R_q$, $w = \mathsf{Cha}(K)$ and $\sigma = \mathsf{Mod}_2(K, w)$ with non-negligible advantage. This means that D can determine if

1. $K = Xs + 2g$ and $Y = as + 2e$ for some s, g, and e drawn from χ_β, or
2. (K, Y) is uniformly random in R_q^2.

Now we can build a distinguisher D' that, on input (a, X, Y, K), solves the RLWE-DH problem by using D as a subroutine in the following manner:

1. Compute $w = \mathsf{Cha}(K)$ and $\sigma = \mathsf{Mod}_2(K, w)$.
2. Run D to solve DPWE problem using the input (a, X, Y, w, σ).
   - If D outputs 1 then $K = Xs + 2g$ and $Y = as + 2e$ for some $s, g, e \leftarrow \chi_\beta$,
   - else (K, Y) is a uniformly random element from R_q^2.

Since D solves DPWE with non-negligible advantage, it follows that D' solves RLWE-DH with non-negligible advantage as well, contradicting RLWE-DH's hardness.

□

As a result from Theorem 1 and Theorem 2, we can say that if RLWE is a hard problem then DPWE is also hard, and thus so is PWE.

3 Protocol description

We turn to describing the protocols RLWE-PAK and RLWE-PPK themselves, and examining their security.

3.1 Password-Authenticated RLWE Key Exchange

Let n be a power of 2, and $f(x) = x^n + 1$. Let $q = 2^{\omega(\log n)} + 1$ be an odd prime such that $q \bmod 2n = 1$. Let $H_1 \colon \{0,1\}^* \to R_q$ be a hash function, $H_l \colon \{0,1\}^* \to \{0,1\}^\kappa$ for $l \in \{2,3\}$ be hash functions for verification of communications, and $H_4 \colon \{0,1\}^* \to \{0,1\}^\kappa$ be a Key Derivation Function (KDF), where κ is the bit-length of the final shared key. We model the hash functions and KDF as random oracles. Let a be a fixed element chosen uniformly at random from R_q and given to all users. Let χ_β be a discrete Gaussian distribution with parameter $\beta \in \mathbb{R}_+^*$. We will make use of the Cha and Mod_2 functions defined in [48] and recalled above.

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

Initiation Client $\mathcal{C}$ randomly samples $s_\mathcal{C}, e_\mathcal{C} \leftarrow \chi_\beta$, computes $\alpha = as_\mathcal{C} + 2e_\mathcal{C}$, $\gamma = H_1(pw_\mathcal{C})$ and $m = \alpha + \gamma$ and sends $< \mathcal{C}, m >$ to party $\mathcal{S}$.

Response Server $\mathcal{S}$ receives $< \mathcal{C}, m >$ from party $\mathcal{C}$ and checks that $m \in R_q$; if not, it aborts. Otherwise it computes $\alpha = m + \gamma'$ where $\gamma' = -H_1(pw_\mathcal{C})$. Server $\mathcal{S}$ then randomly samples $s_\mathcal{S}, e_\mathcal{S} \leftarrow \chi_\beta$ and computes $\mu = as_\mathcal{S} + 2e_\mathcal{S}$ and $k_\mathcal{S} = \alpha \cdot s_\mathcal{S}$.
Next, Server $\mathcal{S}$ computes $w = \mathsf{Cha}(k_\mathcal{S}) \in \{0,1\}^n$ and $\sigma = \mathsf{Mod}_2(k_\mathcal{S}, w)$. Server $\mathcal{S}$ sends μ, w, and $k = H_2(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$ to party $\mathcal{C}$ and computes the value $k'' = H_3(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$.

Initiator finish Client $\mathcal{C}$ checks that $\mu \in R_q$, and computes $k_\mathcal{C} = s_\mathcal{C} \cdot \mu$ and $\sigma = \mathsf{Mod}_2(k_\mathcal{C}, w)$. Client $\mathcal{C}$ verifies that $H_2(\mathcal{C}, \mathcal{S}, m, \mu, \sigma_\mathcal{C}, \gamma')$ matches the value of k received from Server $\mathcal{S}$ where $\gamma' = -\gamma$. If it does not, Client $\mathcal{C}$ ends the communication.
If it does, Client $\mathcal{C}$ computes $k' = H_3(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$ and derives the session key $sk_\mathcal{C} = H_4(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$. It then sends k' back to Server $\mathcal{S}$.

Responder finish Finally, Server $\mathcal{S}$ verifies that $k' = H_3(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$ the same way Client $\mathcal{C}$ verified k. If this is correct, Server $\mathcal{S}$ then derives the session key by computing $sk_\mathcal{S} = H_4(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$. Otherwise, $\mathcal{S}$ refuses to compute a session key.

Client $\mathcal{C}$		Server $\mathcal{S}$
Input $\mathcal{S}, pw_{\mathcal{C}}$ Sample $s_{\mathcal{C}}, e_{\mathcal{C}} \leftarrow \chi_\beta$ $\alpha = as_{\mathcal{C}} + 2e_{\mathcal{C}}$ $\gamma = H_1(pw_{\mathcal{C}})$ $m = \alpha + \gamma$	$\xrightarrow{<\mathcal{C}, m>}$	$\gamma' = -H_1(pw_{\mathcal{C}})$ Abort if $m \notin R_q$ Sample $s_{\mathcal{S}}, e_{\mathcal{S}} \leftarrow \chi_\beta$ $\mu = as_{\mathcal{S}} + 2e_{\mathcal{S}} \in R_q$ $\alpha = m + \gamma'$ $k_{\mathcal{S}} = \alpha s_{\mathcal{S}}$ $w = \mathsf{Cha}(k_{\mathcal{S}}) \in \{0,1\}^n$ $\sigma = \mathsf{Mod}_2(k_{\mathcal{S}}, w)$ $k = H_2(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$ $k'' = H_3(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$
Abort if $\mu \notin R_q$ $k_{\mathcal{C}} = s_{\mathcal{C}}\mu$ $\sigma = \mathsf{Mod}_2(k_{\mathcal{C}}, w)$ $\gamma' = -\gamma$	$\xleftarrow{\mu, w, k}$	
Abort if $k \neq H_2(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$ else $k' = H_3(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$ $sk_{\mathcal{C}} = H_4(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$	$\xrightarrow{k'}$	Abort if $k' \neq k''$ $sk_{\mathcal{S}} = H_4(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$

Fig. 1. Explicitly Authenticated Protocol

3.2 Correctness

Theorem 3 (Correctness). *Let q be an odd prime such that $q > 16\beta^2 n^{3/2}$. Let two parties, $\mathcal{C}$ and $\mathcal{S}$, establish a shared key by honestly following the protocol described above. Then, the two will end with the same key with overwhelming probability.*

Proof. To show the correctness of our RLWE-PAK protocol, it is sufficient to show that the key material derived as $\mathsf{Mod}_2(k_{\mathcal{C}}, \mathsf{Cha}(k_{\mathcal{S}})) = \mathsf{Mod}_2(k_{\mathcal{S}}, \mathsf{Cha}(k_{\mathcal{S}}))$. By Lemma 4, if $k_{\mathcal{C}}$ and $k_{\mathcal{S}}$ are sufficiently close then we done. specifically, if $|k_{\mathcal{C}} - k_{\mathcal{S}}| < q/4$ then both sides have the same value, σ. If we compare the two:

$$\begin{aligned} k_{\mathcal{C}} &= s_{\mathcal{C}}\mu = s_{\mathcal{C}}.(as_{\mathcal{S}} + 2e_{\mathcal{S}}) \\ &= as_{\mathcal{C}}.s_{\mathcal{S}} + 2e_{\mathcal{S}}s_{\mathcal{C}} \\ k_{\mathcal{S}} &= \alpha s_{\mathcal{S}} = (as_{\mathcal{C}} + 2e_{\mathcal{C}}).s_{\mathcal{S}} \\ &= as_{\mathcal{C}}.s_{\mathcal{S}} + 2e_{\mathcal{C}}.s_{\mathcal{S}} \end{aligned}$$

we find that $k_{\mathcal{C}} - k_{\mathcal{S}} = 2[e_{\mathcal{S}}s_{\mathcal{C}} - e_{\mathcal{C}}.s_{\mathcal{S}}]$ By Lemma 2, each individual $e_{\mathcal{S}}, s_{\mathcal{C}}, e_{\mathcal{C}}, s_{\mathcal{S}}$ term has norm less than $\beta\sqrt{n}$ with overwhelming probability. Applying Lemma 1 and the triangle inequality, we have that $\|k_{\mathcal{C}} - k_{\mathcal{S}}\| \leq 4\beta^2 n^{3/2} < q/4$ with overwhelming probability. Hence $\mathsf{Mod}_2(k_{\mathcal{C}}, \mathsf{Cha}(k_{\mathcal{S}})) = \mathsf{Mod}_2(k_{\mathcal{S}}, \mathsf{Cha}(k_{\mathcal{S}}))$. □

4 Proof of security for RLWE-PAK

Our proof of security follows the one in the PAK suite paper by MacKenzie [38]. We essentially adapt it to our PWE instantiation. The objective is to show that an adversary $\mathcal{A}$ attacking the system is unable to determine the SK of a fresh instance with greater advantage than that of an online dictionary attack.

In what follows, we distinguish Client Action (CA) queries and Server Action (SA) queries. The adversary makes a:

- ***CA0*** query if it instructs some unused $\Pi^i_{\mathcal{C}}$ to send the first message to some $\mathcal{S}$;
- ***SA1*** query if it sends some message to a previously unused $\Pi^j_{\mathcal{S}}$;

– ***CA1*** query if it sends a message to some $\Pi_{\mathcal{C}}^{i}$ expecting the second protocol message;
– ***SA2*** query if it sends some message to a $\Pi_{\mathcal{S}}^{j}$ expecting the last protocol message.

For the convenience of the reader, certain events corresponding to $\mathcal{A}$ making password guesses - against a client instance, against a server instance, and against a client instance and server instance that are partnered - are defined:

– $testpw(\mathcal{C}, i, \mathcal{S}, pw, l)$: for some m, μ, γ', w and k, $\mathcal{A}$ makes an $H_l(<\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma'>)$ query, a CA0 query to $\Pi_{\mathcal{C}}^{i}$ with input $\mathcal{S}$ and output $<\mathcal{C}, m>$, a CA1 query to $\Pi_{\mathcal{C}}^{i}$ with input $<\mu, k, w>$ and an $H_1(pw)$ query returning $-\gamma' = as_h + 2e_h \in R_q$, where the latest query is either the $H_l(.)$ query or the CA1 query. $\sigma = \mathsf{Mod}_2(k_{\mathcal{S}}, w) = \mathsf{Mod}_2(k_{\mathcal{C}}, w)$, $k_{\mathcal{S}} = \alpha s_{\mathcal{S}}, k_{\mathcal{C}} = \mu s_{\mathcal{C}}$ and $m = \alpha - \gamma'$. The associated value of this event is output of $H_l(.), l \in \{2, 3, 4\}$.
– $testpw!(\mathcal{C}, i, \mathcal{S}, pw)$: for some w and k a CA1 query with input $<\mu, k, w>$ causes a $testpw(\mathcal{C}, i, \mathcal{S}, pw, 2)$ event to occur, with associated value k.
– $testpw(\mathcal{S}, j, \mathcal{C}, pw, l)$: for some m, μ, γ', w and k $\mathcal{A}$ makes an $H_l(<\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma'>)$ query and previously made SA1 query to $\Pi_{\mathcal{S}}^{j}$ with input $<\mathcal{C}, m>$ and output $<\mu, k, w>$, and an $H_1(pw)$ query returning $-\gamma'$, where $\sigma = \mathsf{Mod}_2(k_{\mathcal{S}}, w) = \mathsf{Mod}_2(k_{\mathcal{C}}, w)$, $k_{\mathcal{S}} = \alpha s_{\mathcal{S}}, k_{\mathcal{C}} = \mu s_{\mathcal{C}}$ and $m = \alpha - \gamma'$. The associated value of this event is output of $H_l(.), l \in \{2, 3, 4\}$ generated by $\Pi_{\mathcal{S}}^{j}$.
– $testpw!(\mathcal{S}, j, \mathcal{C}, pw)$: a SA2 query to $\Pi_{\mathcal{S}}^{j}$ is made with k', where a $testpw(\mathcal{S}, j, \mathcal{C}, pw, 3)$ event previously occured with associated value k'.
– $testpw^*(\mathcal{S}, j, \mathcal{C}, pw)$: $testpw(\mathcal{S}, j, \mathcal{C}, pw, l)$ occurs for some $l \in \{2, 3, 4\}$.
– $testpw(\mathcal{C}, i, \mathcal{S}, j, pw)$: for some $l \in \{2, 3, 4\}$, both a $testpw(\mathcal{C}, i, \mathcal{S}, pw, l)$ event and a $testpw(\mathcal{S}, j, \mathcal{C}, pw, l)$ event occur, where $\Pi_{\mathcal{C}}^{i}$ is paired with $\Pi_{\mathcal{S}}^{j}$ and $\Pi_{\mathcal{S}}^{j}$ is paired with $\Pi_{\mathcal{C}}^{i}$ after its SA1 query.
– $testexecpw(\mathcal{C}, i, \mathcal{S}, j, pw)$: for some m, μ, γ', w, $\mathcal{A}$ makes an $H_l(<\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma'>)$ query for $l \in \{2, 3, 4\}$, and previously made an $\mathsf{Execute}(\mathcal{C}, i, \mathcal{S}, j)$ query that generates m and μ and an $H_1(pw)$ query returning $-\gamma' = as_h + 2e_h \in R_q$, where $\sigma = \mathsf{Mod}_2(k_{\mathcal{S}}, w) = \mathsf{Mod}_2(k_{\mathcal{C}}, w)$, $k_{\mathcal{S}} = \alpha s_{\mathcal{S}}, k_{\mathcal{C}} = \mu s_{\mathcal{C}}$ and $m = \alpha - \gamma'$.
– $correctpw$: before any $\mathsf{Corrupt}$ query, either a $testpw!(\mathcal{C}, i, \mathcal{S}, pw)$ event occurs for some $\mathcal{C}, i$ and $\mathcal{S}$, or a $testpw^*(\mathcal{S}, j, \mathcal{C}, pw_{\mathcal{C}})$ event occurs for some $\mathcal{S}, j$, and $\mathcal{C}$.
– $correctpwexec$: a $testexecpw(\mathcal{C}, i, \mathcal{S}, j, pw_{\mathcal{C}})$ event occurs for some $\mathcal{C}, i, \mathcal{S}$, and j.
– $doublepwserver$: before any $\mathsf{Corrupt}$ query happens, both a $testpw^*(\mathcal{S}, j, \mathcal{C}, pw)$ event and $testpw^*(\mathcal{S}, j, \mathcal{C}, pw')$ occur for some $\mathcal{S}, j, \mathcal{C}, pw$ and pw', with $pw \neq pw'$.
– $pairedpwguess$: a $testpw(\mathcal{C}, i, \mathcal{S}, j, pw_{\mathcal{C}})$ event occurs, for some $\mathcal{C}, i, \mathcal{S}$, and j.

Theorem 4. *Let* P:=$\mathsf{RLWE\text{-}PAK}$*, described in Figure 1, using group* 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, $\mathsf{Execute}$, Reveal, $\mathsf{Corrupt}$, *respectively, and* n_{ro} *queries to the random oracles. Then for* $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$*:*

$$Adv_{\mathsf{P}}^{ake}(\mathcal{A}) = \frac{n_{se}}{L} + O\Big(n_{se} Adv_{R_q}^{PWE}(t', {n_{ro}}^2) + Adv_{R_q}^{DRLWE}(t', n_{ro}) + \frac{(n_{se} + n_{ex})(n_{ro} + n_{se} + n_{ex})}{q^n} + \frac{n_{se}}{2^{\kappa}}\Big)$$

Proof. We study a sequence of protocols - $\mathsf{P}_0, \mathsf{P}_1, \cdots, \mathsf{P}_7$ - with the following properties. First $\mathsf{P}_0 = \mathsf{P}$ and P_7 is by design only possible to attack using natural online guessing. Secondly, we have

$$Adv_{\mathsf{P}_0}^{ake}(\mathcal{A}) \le Adv_{\mathsf{P}_1}^{ake}(\mathcal{A}) + \epsilon_1 \le \cdots \le Adv_{\mathsf{P}_7}^{ake}(\mathcal{A}) + \epsilon_7$$

where $\epsilon_1, \cdots, \epsilon_7$ are all negligible values in k. Adding up the negligible values and counting the success probability of the online attack in P_7 then gives the desired result.

We can assume that n_{ro} and $n_{se} + n_{ex}$ are both ≥ 1. Random oracle queries are answered in the usual way: new queries are answered with uniformly random values, and previously made queries are answered identically to the past response. We further assume that the $H_1(pw)$ query is answered by the simulator by computing the response as $as_h + 2e_h \in R_q$, where (s_h, e_h) is sampled uniformly at random from R_q^2. Finally, if $\mathcal{A}$ makes an $H_l(v)$ query for $l \in \{2,3,4\}$ and some v then the corresponding $H_{l'}(v)$ and $H_{l''}(v)$ queries are computed and stored, where $l', l'' \in \{2,3,4\} \setminus \{l\}$. $\mathcal{A}$ only sees the output of $H_l(v)$, but the other two queries are still considered to have been made by $\mathcal{A}$.

We now detail our squence of protocols, and bound $\mathcal{A}$'s advantage difference from each protocol to the next.

Protocol P_0**:** is just the original protocol P.

Protocol P_1**:** P_1 is nearly identical to P_0, but is forcefully halted as soon as honest parties randomly choose m or μ values seen previously in the execution.

Specifically, let E_1 be the event that an m value generated in a CA0 or Execute query yields an m value already seen in some previous CA0 or Execute query, an m value already used as input in some previous SA1 query, or an m value from some previous $H_l(.)$ query made by $\mathcal{A}$. Let E_2 be the event that a μ value generated in SA1 or Execute query yields a μ from a previous SA1 or Execute query, a μ value sent as input in some previous CA1 query, or a μ value from a previous $H_l(.)$ query. Setting $E = E_1 \vee E_2$ then P_1 is defined as being identical to P_0 except that the protocol halts and the adversary fails when E occurs.

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

$$Adv_{\mathsf{P}_0}^{ake}(\mathcal{A}) \leq Adv_{\mathsf{P}_1}^{ake}(\mathcal{A}) + \frac{O((n_{se} + n_{ex})(n_{ro} + n_{se} + n_{ex}))}{q^n}$$

Proof. Consider the latest m or μ value generated. The probability that this value has previously been generated in a Send, Execute, or random oracle query is $\frac{n_{ro}+n_{se}+n_{ex}}{q^n}$. There are $n_{se} + n_{ex}$ values that are required to be unique if event E does not occur. Thus, the probability of any of the m or μ values not being unique is indeed $\frac{O((n_{se}+n_{ex})(n_{ro}+n_{se}+n_{ex}))}{q^n}$. Note that the cardinal of R_q is exactly q^n, and that m is random because γ is. □

Protocol P_2**:** This protocol is identical to P_1 except that Send and Execute queries are answered without using random oracles. Any random oracle queries $\mathcal{A}$ subsequently makes are answered in such a way as to be consistent with the results of these Send and Execute queries.

In more detail, the queries in P_2 are now answered as follows:

- In an Execute$(\mathcal{C}, i, \mathcal{S}, j)$ query, $m = as_m + 2e_m$ where $s_m, e_m \leftarrow \in R_q$, $\mu = as_{\mathcal{S}} + 2e_{\mathcal{S}}$ where $s_{\mathcal{S}}, e_{\mathcal{S}} \leftarrow \in \chi_\beta$, $w \leftarrow \in \{0,1\}^n$, $k, k' \leftarrow \in \{0,1\}^\kappa$, and $sk_{\mathcal{C}}^i \leftarrow sk_{\mathcal{S}}^j \leftarrow \{0,1\}^\kappa$.
- In a CA0 query to instance $\Pi_{\mathcal{C}}^i$, $m = as_m + 2e_m$ where $s_m, e_m \leftarrow \in R_q$.
- In a SA1 query to instance $\Pi_{\mathcal{S}}^j$, $\mu = as_{\mathcal{S}} + 2e_{\mathcal{S}}$ where $s_{\mathcal{S}}, e_{\mathcal{S}} \leftarrow \in \chi_\beta$, $w \leftarrow \{0,1\}^n$, and $sk_{\mathcal{S}}^j, k, k'' \leftarrow \{0,1\}^\kappa$.
- In a CA1 query to instance $\Pi_{\mathcal{C}}^i$, do the following.
  - If this query causes a $testpw!(\mathcal{C}, i, \mathcal{S}, pw_{\mathcal{C}})$ event to occur, then set k' to the associated value of the $testpw(\mathcal{C}, i, \mathcal{S}, pw_{\mathcal{C}}, 3)$ event, and set $sk_{\mathcal{C}}^i$ to the associated value of the $testpw(\mathcal{C}, i, \mathcal{S}, pw_{\mathcal{C}}, 4)$ event.
  - Else if $\Pi_{\mathcal{C}}^i$ is paired with a server instance $\Pi_{\mathcal{S}}^j$, set $sk_{\mathcal{C}}^i \leftarrow sk_{\mathcal{S}}^j$, then $k' \leftarrow \{0,1\}^\kappa$.
  - Otherwise, $\Pi_{\mathcal{C}}^i$ aborts.
- In a SA2 query to instance $\Pi_{\mathcal{S}}^j$, if this query causes a $testpw!(\mathcal{S}, j, \mathcal{C}, pw_{\mathcal{C}})$ event to occur, or if $\Pi_{\mathcal{S}}^j$ is paired with a client instance $\Pi_{\mathcal{C}}^i$, terminate. Otherwise, $\Pi_{\mathcal{S}}^j$ aborts.

– In an $H_l(<\mathcal{C},\mathcal{S},m,\mu,\sigma,\gamma'>)$ query, for $l\in\{2,3,4\}$, if this $H_l(.)$ query causes a $testpw(\mathcal{S},j,\mathcal{C},pw_{\mathcal{C}},l)$ event, or $testexecpw(\mathcal{C},i,\mathcal{S},j,pw_{\mathcal{C}})$ event to occur, then output the associated value of the event. Otherwise, output a random value from $\{0,1\}^{\kappa}$.

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

$$Adv^{ake}_{\mathsf{P}_1}(\mathcal{A}) = Adv^{ake}_{\mathsf{P}_2}(\mathcal{A}) + \frac{O(n_{ro})}{q^n} + \frac{O(n_{se})}{2^{\kappa}}$$

Proof. In P_1 we can see that $H_2(.), H_3(.)$ and$H_4(.)$ queries are new therefore the values $sk^j_{\mathcal{S}}, k$ and k'' which are created by a server instance $\Pi^j_{\mathcal{S}}$ in the SA1 query are uniformly chosen from $\{0,1\}^{\kappa}$, independent of anything that previously occured.Then in SA2 query, if a $testpw!(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}})$ event occurs, or $\Pi^j_{\mathcal{S}}$ is paired, the instance terminates, and if $\Pi^j_{\mathcal{S}}$ is unpaired and no $testpw!(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}})$ event occurs, then either the instance terminates or aborts. It is easy to show that the total probability of any instance terminating in this case is at most $\frac{n_{se}}{2^{\kappa}}$.

Also in P_1, for any client instance $\Pi^i_{\mathcal{C}}$, either

1. a $testpw!(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}})$ event occurs, and then k' and $sk^i_{\mathcal{C}}$ are set to the values associated with the $testpw(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}},3)$ and $testpw(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}},4)$ events respectively which are guaranteed to occur by our orginal assumption,or
2. no $testpw!(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}})$ event occurs, but exactly one instance $\Pi^j_{\mathcal{S}}$ is paired with $\Pi^i_{\mathcal{C}}$, in which case $sk^i_{\mathcal{C}} = sk^j_{\mathcal{S}}$, and k' is uniformly chosen from $\{0,1\}^{\kappa}$, independent of anything that previously occurred (since no $testpw(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}},3)$ event could have occurred in this case), or
3. no $testpw!(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}})$ event occurs and no instance is paired with $\Pi^i_{\mathcal{C}}$ then either the instance terminates or aborts. In this case the total probability of any instance terminating is at most $\frac{n_{se}}{2^{\kappa}}$.

For any $H_l(.)$ query, $l\in\{2,3,4\}$, either (1) it causes a $testpw(\mathcal{S},j,\mathcal{C},pw_{\mathcal{C}},l)$, or a $testexecpw(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}})$ event to occur, in which case the output is the associated value of the event (i.e., the k value associated with the particular event that occurs), (2) it does not cause a $testpw(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}})$ event, but does cause a $testpw(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}},l)$ event to occur, where the CA1 query of the event had input $<\mu,k>$ for some μ, in which case either $\Pi^i_{\mathcal{C}}$ terminated and the output is k,k',or $sk^{\mathcal{C}}_i$ or $\Pi^i_{\mathcal{C}}$ aborted and the output is uniformly chosen from $\{0,1\}^{\kappa}\setminus\{k,k',sk^{\mathcal{C}}_j\}$, (3) γ' but the adversary has not made an $H_1(pw_{\mathcal{C}})$query, or (4) the output of $H_l(.)$ is uniformly chosen from $\{0,1\}^{\kappa}$ independent of anything that previously occurred, since this is a new $H_l(.)$ query. The total probability of an $H_l(.)$ query causing the third case above is bounded by $\frac{n_{ro}}{q^n}$. The second case above where the output is fixed can only occure when an unpaired client instance terminated with no $testpw!(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}})$ event and for $l\in\{3,4\}$ if can only occur when an $H_2(.)$ query causes a second case where its output is fixed.

If an unpaired client instance $\Pi^i_{\mathcal{C}}$ never terminates without a $testpw!(\mathcal{C},i,\mathcal{S},pw_{\mathcal{C}})$ event, an unpaired server instance $\Pi^j_{\mathcal{S}}$ never terminates without a $testpw!(\mathcal{S},j,\mathcal{C},pw_{\mathcal{C}})$ event, and the third case of the H_2, H_3, and H_4 queries never occurs, then P_2 is consistent with P_1. The claim follows.

□

Protocol P_3**:** is identical to P_2 except that in an $H_l(<\mathcal{C},\mathcal{S},m,\mu,\sigma,\gamma'>)$ query, for $l\in\{2,3,4\}$, it is not checked for consistency against Execute query. So the protocol responds with a random output instead backpatching to preserve consistency with an Execute query. Simply there is no $testexecpw(\mathcal{C},i,\mathcal{S},j,pw_{\mathcal{C}})$ event checking.

Claim 3. *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,*

$$Adv^{ake}_{\mathsf{P}_2}(\mathcal{A}) \le Adv^{ake}_{\mathsf{P}_3}(\mathcal{A}) + Adv^{DRLWE}_{R_q}(t',n_{ro}) + 2Adv^{PWE}_{R_q}(t',n_{ro})$$

Proof. Let E be the event that a *correctpwexec* event occurs then clearly P_2 and P_3 are indistinguishable if E does not occur. So if we have a probability ϵ for E to occur when $\mathcal{A}$ is running against protocol P_2 then we have $Pr[Succ^{ake}_{\mathsf{P}_2}(\mathcal{A})] \leq Pr[Succ^{ake}_{\mathsf{P}_3}(\mathcal{A})] + \epsilon$ and thus $Adv^{ake}_{\mathsf{P}_2}(\mathcal{A}) \leq Adv^{ake}_{\mathsf{P}_3}(\mathcal{A}) + 2\epsilon$.

Now we construct an algorithm D that attempts to solve PWE by running $\mathcal{A}$ on a simulation of the protocol. Given (a, X, Y, W), D simulates P_2 for$\mathcal{A}$ with these changes:

1. In an $\mathsf{Execute}(\mathcal{C}, i, \mathcal{S}, j)$ query, set $m = X + (as_f + 2e_f)$ where $s_f, e_f \leftarrow R_q$, $\mu = Y + (as_{ff} + 2e_{ff})$ where $s_{ff}, e_{ff} \overset{R}{\in} \chi_\beta$, and selects $w \leftarrow \{0,1\}^n$
2. When $\mathcal{A}$ finishes, for every $H_l(<\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma'>)$ query for $l \in \{2,3,4\}$, where m and μ were generated in an Execute query, and an H_1 query returned $-\gamma' = as_h + 2e_h \in R_q$, then the simulator can compute,

$$\begin{aligned} k_{\mathcal{S}} &= \alpha \cdot (s_y + s_{ff}) = (X + a(s_f - s_h) + 2(e_f - e_h)) \cdot (s_y + s_{ff}) \\ &= X \cdot s_y + (a(s_f - s_h) + 2(e_f - e_h)) \cdot s_y + (X + a(s_f - s_h) + 2(e_f - e_h)) \cdot s_{ff} \\ &\approx X \cdot s_y + Y \cdot (s_f - s_h) + (X + a(s_f - s_h) + 2(e_f - e_h)) \cdot s_{ff} \\ &= X \cdot s_y + Y \cdot (s_f - s_h) + (X + \gamma' + (as_f + 2e_f)) \cdot s_{ff} \end{aligned}$$

So,

$$X \cdot s_y = k_{\mathcal{S}} - Y \cdot (s_f - s_h) - (X + \gamma' + (as_f + 2e_f)) \cdot s_{ff}.$$

And,

$$\sigma' = \mathsf{Mod}_2(k_{\mathcal{S}} - Y \cdot (s_f - s_h) - (X + \gamma' + (as_f + 2e_f)) \cdot s_{ff}, W)$$

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

Note that during an execute query, the simulation sets $m = X + (as_f + 2e_f)$ instead of $m = as_m + 2e_m$ and since X is chosen uniformly at random from R_q and m is randomized by γ then they are definitely indistinguishable. However the simulation also sets $\mu = Y + (as_{ff} + 2e_{ff})$ instead of it being $\mu = as_{\mathcal{S}} + 2e_{\mathcal{S}}$ which is distinguishable if someone can solve the decision RLWE problem. As a result this simulation is indistinguishable from P_2 until E occurs or DRLWE is solved with non negligible advantage, and in former case, D adds the correct $\tau(X, s)$ to the list. After E occurs, the simulation will be distinguishable from P_2. However 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_2.

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

Protocol P_4**:** is identical to P_3 except that if *correctpw* occurs then the protocol halts and the adversary automatically succeeds. This causes theses changes:

1. In a CA1 query to $\Pi^i_{\mathcal{C}}$, if a $testpw!(\mathcal{C}, i, \mathcal{S}, pw_{\mathcal{C}})$ event occurs and no $\mathsf{Corrupt}$ query has been made, halt and say the adversary automatically succeeds.
2. In an $H_l(<\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma'>)$ query for $l \in \{2,3,4\}$, if a $testpw^*(\mathcal{S}, j, \mathcal{C}, pw_{\mathcal{C}})$ event occurs and no $\mathsf{Corrupt}$ query has been made, halt and say the adversary automatically succeeds.

Claim 4. *For any adversary* $\mathcal{A}$,

$$Adv^{ake}_{\mathsf{P}_3}(\mathcal{A}) \leq Adv^{ake}_{\mathsf{P}_4}(\mathcal{A})$$

Proof. This change can only increase the adversary's chances at winning the game, hence the inequality. □

Protocol P_5**:** is identical to P_4 except that if the adversary makes a password guess against partnered client and server instances, the protocol halts and the adversary fails. Simply if a $pairedpwguess$ event occurs, the protocal halts and the adversary fails.We suppose that when a query is made, the test for $correctpw$ occurs after the test for $pairedpwguess$. Note that this causes the following change: if a $testpw(\mathcal{C}, i, \mathcal{S}, pw, l)$ event occurs, this should be checked in a CA1 query, or an $H_l(.)$ query for $l \in \{2, 3, 4\}$ check if a $testpw(\mathcal{C}, i, \mathcal{S}, pw)$ event also occurs.

Claim 5. *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,*

$$Adv_{\mathsf{P}_4}^{ake}(\mathcal{A}) \le Adv_{\mathsf{P}_5}^{ake}(\mathcal{A}) + 2n_{se}Adv_{R_q}^{PWE}(t', n_{ro})$$

Proof. Let E be the event that a $pairedpwguess$ event occurs then clearly P_4 and P_5 are indistinguishable if E does not occur. So if we have a probability ϵ for E to be occured, when $\mathcal{A}$ is running against protocol P_4 then we have $Pr[Succ_{\mathsf{P}_4}^{ake}(\mathcal{A})] \le Pr[Succ_{\mathsf{P}_5}^{ake}(\mathcal{A})] + \epsilon$ and thus $Adv_{\mathsf{P}_4}^{ake}(\mathcal{A}) \le Adv_{\mathsf{P}_5}^{ake}(\mathcal{A}) + 2\epsilon$.

Now we construct an algorithm D that attempts to solve PWE by running $\mathcal{A}$ on a simulation of the protocol. Given (a, X, Y, W), D chooses a random $d \in \{1, \ldots, n_{se}\}$ and simulates P_4 for$\mathcal{A}$ with these changes:

1. In the d^{th} CA0, say to a client instance $\Pi_{\mathcal{C}}^{i'}$, with input $\mathcal{S}$, set $m = X$
2. In a SA1 query to server instance $\Pi_{\mathcal{S}}^{j}$ with input $< \mathcal{C}, m >$ where there was a CA0 query to $\Pi_{\mathcal{C}}^{i'}$ (i.e. the instance with the d^{th} CA0 query) with input $\mathcal{S}$ and output $< \mathcal{C}, m >$, let $\mu = Y + (as_{ff} + 2e_{ff})$ where $s_{ff}, e_{ff} \leftarrow \chi_\beta$.
3. In a CA1 query to $\Pi_{\mathcal{C}}^{i'}$. If $\Pi_{\mathcal{C}}^{i'}$ is unpaired, D outputs 0 and halts.
4. In a SA2 query to $\Pi_{\mathcal{S}}^{j}$, if $\Pi_{\mathcal{S}}^{j}$ was paired with $\Pi_{\mathcal{C}}^{i'}$ after its SA1 query, but is not now paired with $\Pi_{\mathcal{C}}^{i'}$, no test for $correctpw$ event is made and $\Pi_{\mathcal{S}}^{j}$ aborts.
5. Run $\mathcal{A}$ on the simulation of P_4 when if finishes, for every $H_l(< \mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma' >)$ query for $l \in \{2, 3, 4\}$, where m and μ were generated in an $\Pi_{\mathcal{C}}^{i'}$ query, and an $H_1(pw)$ query returned $-\gamma' = as_h + 2e_h \in R_q$, we can see that the simulation computes,

$$\begin{aligned} k_{\mathcal{S}} &= \alpha \cdot (s_y + s_{ff}) = (X - (as_h + 2e_h)) \cdot (s_y + s_{ff}) \\ &= X \cdot s_y - (as_h + 2e_h) \cdot s_y + (X - (as_h + 2e_h)) \cdot s_{ff} \\ &\approx X \cdot s_y - Y \cdot s_h + (X - (as_h + 2e_h)) \cdot s_{ff} \\ &= X \cdot s_y - Y \cdot s_h + (X + \gamma') \cdot s_{ff} \end{aligned}$$

So,

$$X \cdot s_y = k_{\mathcal{S}} + Y \cdot s_h - (X + \gamma') \cdot s_{ff}.$$

And,

$$\sigma' = \mathsf{Mod}_2(k_{\mathcal{S}} + Y \cdot s_h - (X + \gamma') \cdot s_{ff}, W)$$

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

This simulation is perfectly indistinguishable from P_4 until (1) a $testpw(\mathcal{S}, j, \mathcal{C}, pw)$ event occurs, where $\Pi_{\mathcal{S}}^{j}$ was paired with $\Pi_{\mathcal{C}}^{i'}$ after the SA1 query, or (2) $\Pi_{\mathcal{C}}^{i'}$ is not paired with server instance when the CA1 query is made. Note that the probability of a $pairedpwguess$ event occuring for $\Pi_{\mathcal{C}}^{i'}$, is at least $\frac{\epsilon}{n_{se}}$ and this is at most the probability of an event of type (1) occuring. Since an event of type (2) implies that $pairedpwguess$ event would never have occurred in P_4 for $\Pi_{\mathcal{C}}^{i'}$ (from P_1). If an event of type (1) occurs, D adds the correct $\tau(X, s)$ to the list.

Note that in either case, the simulation may be distinguishable from P_4, but this doesn't change the fact that a $pairedpwguess$ event will occur for $\Pi_{\mathcal{C}}^{i'}$ with probability at least $\frac{\epsilon}{n_{se}}$ in the simulation. However, we do make the assumption that $\mathcal{A}$ still follows the appropriate time and query bounds or at least the simulator can stop $\mathcal{A}$ from exceeding these bounds, even if $\mathcal{A}$ distinguishes the simulation from P_4.Hence if t' is the running time of D and since D creates a list of size n_{ro} with advantage $\frac{\epsilon}{n_{se}}$ then $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$. The claim follows from the fact that $Adv_{R_q}^{PWE}(D) \leq Adv_{R_q}^{PWE}(t', n_{ro})$.

□

Protocol P_6**:** is identical to P_5 except that if the adversary makes two password guesses against the same server instance, i.e. if a $doublepwserver$ event occurs, the protocol halts and the adversary fails. We suppose that when a query is made, the test for $pairedpwguess$ or $correctpw$ occurs after the test for $doublepwserver$.

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,*

$$Adv_{\mathsf{P}_5}^{ake}(\mathcal{A}) \leq Adv_{\mathsf{P}_6}^{ake}(\mathcal{A}) + 4Adv_{R_q}^{PWE}(t', {n_{ro}}^2)$$

Proof. Let us assume that we have a probability ϵ for $doublepwserver$ event to be occured, when $\mathcal{A}$ is running against a simulation of protocol P_5. We build a distinguisher D to solve PWE problem.

Now Given (a, X, Y, W), D simulates P_5 for$\mathcal{A}$ with these changes:

1. In an $H_1(pw)$ query output $X \cdot s_h + (as_f + 2e_f)$.
2. In a SA1 query to a server instance $\Pi_{\mathcal{S}}^{j}$ with entry $< \mathcal{C}, m >$ where $m \in R_q$, set $\mu = Y + (as_{ff} + 2e_{ff})$ where $s_{ff}, e_{ff} \leftarrow \chi_\beta$.
3. Tests for $correctpw$(from P_4) and $pairedpwguess$ (from P_5) are not made. In particular, unpaired client instances that receive a CA1 query abort and unpaired server instances that receive a SA2 query abort. Also, $H_l(.)$ queries always give values that are uniformly chosen from $\{0,1\}^\kappa$.
4. When $\mathcal{A}$ finishes, for every pair of queries of the form $H_l(< \mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma' >)$ and $H_{l'}(< \mathcal{C}, \mathcal{S}, m, \mu, \hat{\sigma}, \hat{\gamma'} >)$ for $l, l' \in \{2, 3, 4\}$, where $\sigma \in R_q$ and $\hat{\sigma} \in R_q$, there was a SA1 query to a server instance $\Pi_{\mathcal{S}}^{j}$ with input $< \mathcal{C}, m >$ and output $< \mu, k, w' >$ (and thus $m \in R_q$), an $H_1(pw)$ query returned $-\gamma' = Xs_h + (as_f + 2e_f) \in R_q$, an $H_1(\hat{pw})$ query returned $-\hat{\gamma'} = Xs_{\hat{h}} + (as_{\hat{f}} + 2e_{\hat{f}}) \in R_q$ and $s_h \neq s_{\hat{h}}$, we can see that the simulation computes,

$$\begin{aligned}
k_{\mathcal{S}} &= \alpha \cdot (s_y + s_{ff}) = (m + \gamma') \cdot (s_y + s_{ff}) \\
\hat{k_{\mathcal{S}}} &= \hat{\alpha} \cdot (s_y + s_{ff}) = (m + \hat{\gamma'}) \cdot (s_y + s_{ff}) \\
\hat{k_{\mathcal{S}}} - k_{\mathcal{S}} &= (\hat{\gamma'} - \gamma') \cdot (s_y + s_{ff}) \\
&= (\hat{\gamma'} - \gamma') \cdot s_y + (\hat{\gamma'} - \gamma') \cdot s_{ff} \\
&= (Xs_h + (as_f + 2e_f) - (Xs_{\hat{h}} + (as_{\hat{f}} + 2e_{\hat{f}}))) \cdot s_y + (\hat{\gamma'} - \gamma') \cdot s_{ff} \\
&= Xs_y(s_h - s_{\hat{h}}) + (a(s_f - s_{\hat{f}}) + 2(e_f - e_{\hat{f}}))s_y + (\hat{\gamma'} - \gamma') \cdot s_{ff} \\
&\approx Xs_y(s_h - s_{\hat{h}}) + Y \cdot (s_f - s_{\hat{f}}) + (\hat{\gamma'} - \gamma') \cdot s_{ff}
\end{aligned}$$

So,

$$X \cdot s_y = (\hat{k_{\mathcal{S}}} - k_{\mathcal{S}} - Y \cdot (s_f - s_{\hat{f}}) - (\hat{\gamma'} - \gamma') \cdot s_{ff}) \cdot (s_h - s_{\hat{h}})^{-1}.$$

Add,

$$\mathsf{Mod}_2[(\hat{k_{\mathcal{S}}} - k_{\mathcal{S}} - Y \cdot (s_f - s_{\hat{f}}) - (\gamma' - \hat{\gamma'}) \cdot s_{ff}) \cdot (s_h - s_{\hat{h}})^{-1}, W]$$

to the list of possible values for $\tau(X, s)$.

This simulation is perfectly indistinguishable from P_5 up until a *doublepwserver* event, a *pairedpwguess* event or a *correctpw* event occurs, or $\mathcal{A}$ makes a Corrupt query. If an event of type *doublepwserver* occurs, then with probability $\frac{1}{2}$ it occurs for two passwords pw and $\hat{pw}$ with $s_h \neq s_{\hat{h}}$ and in this case D adds the correct $\tau(X, s)$ to the list. If a *correctpw* event or a *pairedpwguess* event occurs, then P_5 will halt and the *doublepwserver* event would never have occured. Also, the *doublepwserver* event will not occur if a Corrupt query made by definition.

Note that in either case, the simulation may be distinguishable from P_5, but this doesn't change the fact that a *doublepwserver* event will occur with probability at least ϵ in the simulation. However, we do make the assumption that $\mathcal{A}$ still follows the appropriate time and query bounds or at least the simulator can stop $\mathcal{A}$ from exceeding these bounds, even if $\mathcal{A}$ distinguishes the simulation from P_5. Hence if t' is the running time of D and since D creates a list of size ${n_{ro}}^2$ with advantage $\frac{\epsilon}{2}$ then $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$. The claim follows from the fact that $Adv_{R_q}^{PWE}(D) \leq Adv_{R_q}^{PWE}(t', {n_{ro}}^2)$.

□

Protocol P_7: is identical to P_6 except that this protocol has an internal password oracle that holds all passwords and accepts queries that examine the correctness of a given password. Note that this internal oracle *passwordoracle* is not available to the adversary. So this oracle generates all passwords during initialization. It accepts queries of the form $testpw(\mathcal{C}, pw)$ and returns TRUE if $pw = pw_{\mathcal{C}}$, and FALSE otherwise. It also accepts $\mathsf{Corrupt}(U)$ queries whether $U \in \mathfrak{S}$ or $U \in \mathfrak{C}$. When a $\mathsf{Corrupt}(U)$ query made in the protocol, it is answered using a $\mathsf{Corrupt}(U)$ query to the password oracle. The protocol is also test if *correctpw* occurs, whenever the first $testpw(\mathcal{C}, i, \mathcal{S}, pw)$ event occurs for an instance $\Pi_{\mathcal{C}}^i$ and password pw, or the first $testpw(\mathcal{S}, j, \mathcal{C}, pw)$ event occurs for an instance $\Pi_{\mathcal{S}}^j$ and password pw, a $testpw(\mathcal{C}, pw)$ query is made to the password oracle to see if $pw = pw_{\mathcal{C}}$.

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

$$Adv_{\mathsf{P}_6}^{ake}(\mathcal{A}) = Adv_{\mathsf{P}_7}^{ake}(\mathcal{A})$$

Proof. By observation, P_6 and P_7 are perfectly indistinguishable. □

Now we analyze the advantage of an adversary $\mathcal{A}$ against the protocol P_7. From the definition of P_7, one can easily bounds the probability of adversary $\mathcal{A}$ succeeding in P_7 as the following:

$$Pr(Succ_{\mathsf{P}_7}^{ake}(\mathcal{A})) \leq Pr(correctpw) + Pr(Succ_{\mathsf{P}_7}^{ake}(\mathcal{A}) \mid \neg correctpw)Pr(\neg correctpw)$$

.

Note that $Pr(correctpw) \leq \frac{n_{se}}{L}$ if the passwords are uniformly chosen from a dictionary of size L, because a Corrupt query occurs after at most n_{se} queries were occurred to the password oracle.

Next we compute $Pr(Succ_{\mathsf{P}_7}^{ake}(\mathcal{A}) \mid \neg correctpw)$. Since *correctpw* event does not occur then the only way for $\mathcal{A}$ to succeed is making a Test query to a fresh instance $\Pi_{\mathcal{U}}^i$ and guessing the bit used in the Test query. Note that if we can prove that the view of the adversary is not dependent on sk_U^i then the probability of success is exactly $\frac{1}{2}$ and to do that we have to examine Reveal and $H_4(.)$ queries.

For the first type, we know by definition of $\mathsf{Reveal}(U, i)$ query that there could be no one for the fresh instance $\Pi^i_{\mathcal{U}}$. Also there is no $\mathsf{Reveal}(U', j)$ query for the instance $\prod^{U'}_j$ which is partnered with $\Pi^i_{\mathcal{U}}$. Moreover the adversary fails if more than a single client instace and a single server instance accept with the same sid by protocol P_1. Thus the output of Reveal queries is independent of sk^i_U.

For the second type, from P_4 the unpaired client or server instance will not terminate before a $correctpw$ event or a Corrupt query which means an instance may only be fresh and receive a Test query if it is partnered. However if $\Pi^i_{\mathcal{U}}$ is partnered, $H_4(.)$ query will never reveal sk^i_U by P_5.

So, the view of the adversary not dependent on sk^i_U then the probability of success is exactly $\frac{1}{2}$. Therefore,

$$\begin{aligned}
Pr(Succ^{ake}_{\mathsf{P}_7}(\mathcal{A})) &\leq Pr(correctpw) + Pr(Succ^{ake}_{\mathsf{P}_7}(\mathcal{A}) \mid \neg correctpw)Pr(\neg correctpw) \\
&\leq Pr(correctpw) + Pr(Succ^{ake}_{\mathsf{P}_7}(\mathcal{A}) \mid \neg correctpw)(1 - Pr(correctpw)) \\
&\leq \frac{n_{se}}{L} + \frac{1}{2}(1 - \frac{n_{se}}{L})) \\
&\leq \frac{1}{2} + \frac{n_{se}}{2L}.
\end{aligned}$$

And $Adv^{ake}_{\mathsf{P}_7}(\mathcal{A}) \leq \frac{n_{se}}{L}$. The theorem follows from this and the Claims 1-7 above.

□

5 Implicit authentication

In this section, we describe a variant of the protocol that gives implicit authentication, similar to the PPK variant on the PAK protocol. We call it the RLWE-PPK protocol. This is illustrated in Figure 2. If either party provides an incorrect password, then the parties' keys will not actually match, and neither party will learn anything about the key held by the other. This effectively prevents communication without explicitly checking for matching passwords.

5.1 RLWE-PPK

The setup is slightly different from that of RLWE-PAK. Here, we need two hash functions H_1 and H_2 from $\{0,1\}^*$ into R_q, and one KDF H_3 from $\{0,1\}^*$ into $\{0,1\}^\kappa$, where κ is again the length of the derived SK. Of course, these are modeled as random oracles. Also, the function f used to compute password verifiers for the server is instantiated as follows: $f(\cdot) = \big(-H_1(\cdot), H_2(\cdot)\big)$.

Initiation Client $\mathcal{C}$ randomly samples $s_\mathcal{C}, e_\mathcal{C} \leftarrow \chi_\beta$, computes $\alpha = as_\mathcal{C} + 2e_\mathcal{C}$, $\gamma_1 = H_1(pw_\mathcal{C})$, $\gamma_2 = H_2(pw_\mathcal{C})$ and $m = \alpha + \gamma_1$ and sends$< \mathcal{C}, m >$ to party $\mathcal{S}$.

Response Server $\mathcal{S}$ receives $< \mathcal{C}, m >$ from party $\mathcal{C}$ and checks if $m \in R_q$. If not, abort; otherwise Server $\mathcal{S}$ randomly samples $s_\mathcal{S}, e_\mathcal{S} \leftarrow \chi_\beta$ and computes $\nu = as_\mathcal{S} + 2e_\mathcal{S}$ and recovers $\alpha = m + \gamma'_1$ where $< \gamma'_1, \gamma_2 >$. Then compute $\mu = \nu + \gamma_2$ and $k_\mathcal{S} = \alpha \cdot s_\mathcal{S}$. Next, Server $\mathcal{S}$ computes $w = \mathsf{Cha}(k_\mathcal{S}) \in \{0,1\}^n$ and $\sigma = \mathsf{Mod}_2(k_\mathcal{S}, w)$. Server $\mathcal{S}$ sends μ and w to party $\mathcal{C}$ and computes $sk_\mathcal{S} = H_3(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma'_1)$.

Initiator finish Client $\mathcal{C}$ receives $< \mu, w >$ from party $\mathcal{S}$ and checks if $\mu \in R_q$. If not, it aborts, and otherwise $\mathcal{C}$ recovers $\nu = \mu - \gamma_2$, computes $k_\mathcal{C} = s_\mathcal{C} \cdot \nu$ and $\sigma = \mathsf{Mod}_2(k_\mathcal{C}, w)$. Finally, Client $\mathcal{C}$ derives the session key $sk_\mathcal{C} = H_3(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma'_1)$.

Client $\mathcal{C}$		Server $\mathcal{S}$
Input $\mathcal{S}, pw$		$< \gamma_1', \gamma_2 > = < -H_1(pw_{\mathcal{C}}), H_2(pw_{\mathcal{C}}) >$
Sample $s_{\mathcal{C}}, e_{\mathcal{C}} \leftarrow \chi_\beta$ $\alpha = as_{\mathcal{C}} + 2e_{\mathcal{C}}$ $\gamma_1 = H_1(pw_{\mathcal{C}})$ $\gamma_2 = H_2(pw_{\mathcal{C}})$ $m = \alpha + \gamma_1$	$\xrightarrow{< \mathcal{C}, m >}$	Abort if $m \notin R_q$ Sample $s_{\mathcal{S}}, e_{\mathcal{S}} \leftarrow \chi_\beta$ $\nu = as_{\mathcal{S}} + 2e_{\mathcal{S}} \in R_q$ $\alpha = m + \gamma_1'$
Abort if $\mu \notin R_q$ $\nu = \mu - \gamma_2$ $k_{\mathcal{C}} = s_{\mathcal{C}}\nu$ $\sigma = \mathsf{Mod}_2(k_{\mathcal{C}}, w)$ $\gamma_1' = -\gamma_1$ $sk_{\mathcal{C}} = H_3(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma_1')$	$\xleftarrow{< \mu, w >}$	$\mu = \nu + \gamma_2$ $k_{\mathcal{S}} = \alpha s_{\mathcal{S}}$ $w = \mathsf{Cha}(k_{\mathcal{S}}) \in \{0,1\}^n$ $\sigma = \mathsf{Mod}_2(k_{\mathcal{S}}, w)$ $sk_{\mathcal{S}} = H_3(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma_1')$

Fig. 2. Implicitly Authenticated Protocol

5.2 Proof of security for RLWE-PPK

The proof of security for our Implicitly Authenticated Protocol follows the model of security in the PAK suite paper by Mackenzie, Laboratories and Technologies [38] and it is similar to our proof for the Explicitly Authenticated Protoco above. Therefore we don't want to go through the proof details. However we give a sketch of the proof. We first define some similar events to what we have in section 4, corresponding to the adversary making a password guess against a client instance, against a server instance, and against a client instance and server instance that are partnered:

- $testpw(\mathcal{C}, i, \mathcal{S}, pw)$: for some $m, \mu, \gamma_1', \gamma_2, w$ and k, $\mathcal{A}$ makes an $H_3(< \mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma_1' >$) query. The associated value of this event is $sk^i_{\mathcal{C}}$.
- $testpw(\mathcal{S}, j, \mathcal{C}, pw)$: for some $m, \mu, \gamma_1', \gamma_2, w$ and k, $\mathcal{A}$ makes an $H_3(< \mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma_1' >$) query. The associated value of this event is $sk^j_{\mathcal{S}}$.
- $testexecpw(\mathcal{C}, i, \mathcal{S}, j, pw)$: for $m, \mu, \gamma_1', \gamma_2$ and w, $\mathcal{A}$ makes an $H_3(< \mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma_1' >)$ query, and previously made an $\mathsf{Execute}(\mathcal{C}, i, \mathcal{S}, j)$ query that generates m and μ, an $H_1(pw)$ query returning $-\gamma_1' = as_h + 2e_h \in R_q$, and an $H_2(pw)$ query returning $\gamma_2 = as_{h'} + 2e_{h'} \in R_q$ where $\sigma = \mathsf{Mod}_2(k_{\mathcal{S}}, w) = \mathsf{Mod}_2(k_{\mathcal{C}}, w)$, $k_{\mathcal{S}} = \alpha s_{\mathcal{S}}, k_{\mathcal{C}} = \nu s_{\mathcal{C}}$ where $m = \alpha - \gamma'$ and $\mu = \nu + \gamma_2$.
- $correctpw$: either a $testpw(\mathcal{C}, i, \mathcal{S}, pw_{\mathcal{C}})$ event occurs for some $\mathcal{C}, i$ and $\mathcal{S}$, or some $testpw(\mathcal{S}, j, \mathcal{C}, pw_{\mathcal{C}})$ event occurs for some $\mathcal{S}, j$, and $\mathcal{C}$.
- $correctpwexec$: a $testexecpw(\mathcal{C}, i, \mathcal{S}, j, pw_{\mathcal{C}})$ event occurs for some $\mathcal{C}, i, \mathcal{S}$, and j.
- $doublepwserver$: both a $testpw(\mathcal{S}, j, \mathcal{C}, pw)$ event and $testpw(\mathcal{S}, j, \mathcal{C}, pw')$ occur for some $\mathcal{S}, j, \mathcal{C}, pw$ and pw', with $pw \neq pw'$.
- $doublepwclient$: both a $testpw(\mathcal{C}, i, \mathcal{S}, pw)$ event and $testpw(\mathcal{C}, i, \mathcal{S}, pw')$ occur for some $\mathcal{C}, i, \mathcal{S}, pw$ and pw', with $pw \neq pw'$.

Now we need to show that an adversary attacking the system unable to determine the session key of a fresh instance with greater advantage than that of an online dictionary attack.

Theorem 5. *Let* P=RLWE-PPK *as described in Figure 2, using group 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, and n_{ro} queries to the random oracles. Then for $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$:*

$$Adv^{ake}_{P}(\mathcal{A}) = \frac{n_{se}}{L} + O\left(Adv^{PWE}_{R_q}(t', {n_{ro}}^2) + \frac{(n_{se} + n_{ex})(n_{ro} + n_{se} + n_{ex})}{q^n}\right)$$

Proof. We introduce a series of protocols $\mathsf{P}_0, \mathsf{P}_1, \cdots, \mathsf{P}_7$ where $\mathsf{P}_0 = \mathsf{P}$ and P_7 is a protocol that can be only attacked by simple online guessing attack which admit a straight forward analysis. Furthermore we should have

$$Adv^{ake}_{\mathsf{P}_0}(\mathcal{A}) \leq Adv^{ake}_{\mathsf{P}_1}(\mathcal{A}) + \epsilon_1 \leq \cdots \leq Adv^{ake}_{\mathsf{P}_7}(\mathcal{A}) + \epsilon_7$$

where $\epsilon_1, \cdots, \epsilon_7$ are negligible values.

Here are some assumptions that we need in our proof. First, we assume without loss of generality that $n_{ro} \geq 1$ and $n_{se} + n_{ex} \geq 1$. We also assume that the random oracle respond with a fresh output for any new query and a consistent one with the previous queries for not new query. The third assumption is that $H_1(pw)$ query outputs $(as_h + 2e_h) \in R_q$ and $H_2(pw)$ query outputs $(as_{h'} + 2e_{h'}) \in R_q$ where (s_h, e_h) and $(s_{h'}, e_{h'})$ are known. The last assumption is that if $\mathcal{A}$ made an $H_l(.)$ query for $l \in \{1, 2\}$ then the corresponding $H_{l'}(.)$ query is made automatically, where $l' \in \{1, 2\} \setminus \{l\}$. Even though all queries are considered to be made by $\mathcal{A}$, $\mathcal{A}$ only sees the output of $H_l(.)$.

Protocol P_0: is just the original protocol P.

Protocol P_1: is similar to P_0 protocol however this protocol halts if honest parties randomly choose m or μ values seen previously in the execution of the protocol.

Claim 8. *For any adversary $\mathcal{A}$ we can show that*

$$Adv^{ake}_{\mathsf{P}_0}(\mathcal{A}) \leq Adv^{ake}_{\mathsf{P}_1}(\mathcal{A}) + \frac{O((n_{se} + n_{ex})(n_{ro} + n_{se} + n_{ex}))}{q^n}$$

Proof. The proof is similar to what we did to prove Claim 1 in section 4. □

Protocol P_2: is identical to P_1 except this protocol answers Send and Execute queries without making any random oracle queries, and following random oracle queries by $\mathcal{A}$ are backpatched, while it is possible, to be proper with the results of Send and Execute queries.

Claim 9. *For any adversary $\mathcal{A}$,*

$$Adv^{ake}_{\mathsf{P}_1}(\mathcal{A}) = Adv^{ake}_{\mathsf{P}_2}(\mathcal{A}) + \frac{O(n_{ro})}{q^n}$$

Proof. This claim can be proved by inspection, P_1 and P_2 can be distinguished only if the adversary makes an $H_3(< \mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma'_1 >)$ query where $\gamma'_1 = H_1(pw_{\mathcal{C}})$. However the adversary doesn't makes $H_1(.)$ query and the probability to make it is negligible. □

Protocol P_3: is identical to P_2 except that in an $H_3(< \mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma'_1 >)$ query, it is not checked for consistency against Execute query. So the protocol responds with a random output instead backpatching to preserve consistency with an Execute query.

Claim 10. *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,*

$$Adv^{ake}_{\mathsf{P}_2}(\mathcal{A}) \leq Adv^{ake}_{\mathsf{P}_3}(\mathcal{A}) + 2Adv^{PWE}_{R_q}(t', n_{ro})$$

Proof. This can be shown by using reduction from PWE similar to what we did in the proof of Claim 3 in section 4. □

Protocol P_4: is identical to P_3 except that if *correctpw* occurs then the protocol halts and the adversary automatically succeeds.

Claim 11. *For any adversary $\mathcal{A}$,*

$$Adv^{ake}_{\mathsf{P}_3}(\mathcal{A}) \leq Adv^{ake}_{\mathsf{P}_4}(\mathcal{A}$$

Proof. This is obvious because the change here can only increase the adversary's chances at winning the game. □

Protocol P_5**:** is identical to P_4 except that if the adversary makes two password guesses against the same server instance, the protocol halts and the adversary fails.

Claim 12. *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,*

$$Adv^{ake}_{\mathsf{P}_4}(\mathcal{A}) \leq Adv^{ake}_{\mathsf{P}_5}(\mathcal{A}) + 4Adv^{PWE}_{R_q}(t', (n_{ro})^2)$$

Proof. This can be proved by using a reduction from PWE similar to what we did in Claim 6. □

Protocol P_6**:** is identical to P_5 except that if the adversary makes two password guesses against the same client instance, the protocol halts and the adversary fails.

Claim 13. *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,*

$$Adv^{ake}_{\mathsf{P}_5}(\mathcal{A}) \leq Adv^{ake}_{\mathsf{P}_6}(\mathcal{A}) + 4Adv^{PWE}_{R_q}(t', {n_{ro}}^2)$$

Proof. This can be proved by using a reduction from PWE similar to what we did in Claim 6. However here we randomly plug in Y added to a random element of R_q into the output of $H_2(.)$ queries, and X added to random element of R_q. □

Protocol P_7**:** is identical to P_6 except that this protocol has an internal password oracle that holds all passwords and accepts queries that exam the correctness of a given password. Note that this internal oracle $passwordoracle$ is not available to the adversary. So this oracle generates all passwords during initialization. That changes the test for correct password guesses from P_4 as the follwoing a query is submitted to the oracle to determine if it is correct when the the adversary makes a password guess.

Claim 14. *For any adversary $\mathcal{A}$,*

$$Adv^{ake}_{\mathsf{P}_6}(\mathcal{A}) = Adv^{ake}_{\mathsf{P}_7}(\mathcal{A})$$

Proof. By observation that P_6 and P_7 are perfectly indistinguishable. □

Now we analyze the advantage of an adversary $\mathcal{A}$ against the protocol P_7. From the definition of P_7, one can easily bounds the probability of adversary $\mathcal{A}$ succeeding in P_7 as the following:

$$Pr(Succ^{ake}_{\mathsf{P}_7}(\mathcal{A})) \leq Pr(correctpw) + Pr(Succ^{ake}_{\mathsf{P}_7}(\mathcal{A}) \mid \neg correctpw)Pr(\neg correctpw)$$

.

Note that $Pr(correctpw) \leq \frac{n_{se}}{L}$ if the passwords are uniformly chosen from a dictionary of size L, because a $\mathsf{Corrupt}$ query occurs after at most n_{se} queries were occurred to the password oracle.

Next we compute $Pr(Succ^{ake}_{\mathsf{P}_7}(\mathcal{A}) \mid \neg correctpw)$. Since $correctpw$ event does not occur then the only way for $\mathcal{A}$ to succeed is making a Test query to a fresh instance $\Pi^i_{\mathcal{U}}$ and guessing the bit used in the Test query. Note that if we can prove that the view of the adversary not dependent on sk^i_U then the probability of success is exactly $\frac{1}{2}$ and to do that we have to examine Reveal and $H_3(.)$ queries.

For the first type, we know by definition of $\mathsf{Reveal}(U, i)$ query that there could be no one for the fresh instance $\Pi^i_{\mathcal{U}}$. Also there is no $\mathsf{Reveal}(U', j)$ query for the instance $\prod^{U'}_j$ which is partnered with $\Pi^i_{\mathcal{U}}$. Moreover the adversary fails if more than a single client instace and a single server instance accept with the same sid by protocol P_1. Thus the output of Reveal queries is independent of sk_U^i.

For the second type, from P_4, an $H_3(.)$ query returns random values independent of what previously occurred. Therefore $H_3(.)$ queries that occur after sk_U^i is independent of sk_U^i. So, the view of the adversary not dependent on sk_U^i then the probability of success is exactly $\frac{1}{2}$. Therefore,

$$\begin{aligned}Pr(Succ^{ake}_{\mathsf{P}_7}(\mathcal{A})) &\leq Pr(correctpw) + Pr(Succ^{ake}_{\mathsf{P}_7}(\mathcal{A}) \mid \neg correctpw)Pr(\neg correctpw)\\ &\leq Pr(correctpw) + Pr(Succ^{ake}_{\mathsf{P}_7}(\mathcal{A}) \mid \neg correctpw)(1 - Pr(correctpw))\\ &\leq \frac{n_{se}}{L} + \frac{1}{2}(1 - \frac{n_{se}}{L}))\\ &\leq \frac{1}{2} + \frac{n_{se}}{2L}.\end{aligned}$$

And $Adv^{ake}_{\mathsf{P}_7}(\mathcal{A}) \leq \frac{n_{se}}{L}$. The theorem follows from this and the Claims 8-14 above.

□

6 Implementation

We provide a proof of concept implementation for the PAKE protocol presented in this paper to verify the practicality of the protocol. We take n = 1024, $\beta = 8/\sqrt{2\pi} \approx 3.2$ and q = $2^{32} - 1$. and note that this choice of parameters is suitable for use in RLWE based post quantum protocols as explained in [10].

We use the NTL library from Shoup with C++ for the implementation, without any parallel programming or multi threading techniques. Also, to improve the performance of the implementation, we compile NTL with `NTL_GMP_LIP` = on. The code is executed on a 3.40 GHz Intel Xeon(R) CPU E5-2687W v2 and 64 GB RAM computer running on Ubuntu 14.04 LTS 64 bit system. The timings for the Initiation, Response, Initiator Finish and Responder Finish parts of the implicit authentication and explicit authentication versions of the protocol are presented in Table1.

Table 1. Timings of PAKE protocol as described in Fig. 1 and Fig. 2

Protocol	Initiation and Initiator Finish (in ms)	Response and Responder Finish (in ms)
RLWE-PAK	4.005	4.639
RLWE-PPK	3.740	4.359

The timings presented in the table are the average of 10000 executions. The operations mainly contributing to the time taken by the protocol are the sampling and multiplication operations on both sides of the protocol. As noted earlier, we have used multiplication with FFT for improved performance. The sampling technique used in the protocol is the sampling procedure in [42] and as noted in [48], the Discrete Gaussian approximates the continuous Gaussian extremely well when $\beta \geq$ 3.2. This proof of concept implementation is a preliminary implementation to show that the protocol presented in this paper can be practical. We believe that this implementation can be improved further with new optimizations.

7 Conclusions

We have proposed two new explicitly and implicitly authenticated PAKE protocols. Our protocols are similar to PAK and PPK; however they are based on the Ring Learning with Errors problem. Though our construction is very similar to the classical construction, the security proof is subtle and intricate and it requires novel techniques. We provide a full proof of security of the new protocols in the Radom Oracle Model. We also provide a proof of concept implementation and implementation results show our protocols are practical and efficient.

In the proof, we make use of the ROM, which models hash functions as random functions. Our proof is a classical proof of security, and may not hold against a quantum adversary. Against such adversaries, one natural extension of the ROM is to allow the queries to be in *quantum superposition*; this is known as the Quantum Random Oracle Model (QROM) [9]. Unfortunately, many tricks that can be used in the ROM are hard to apply in the QROM. Therefore we leave proving the security of our protocols in the QROM as future work. Although there are some developing proof techniques in the QROM [47, 45, 46], more work is needed to adapt classical proofs to this setting.

References

1. Abdalla, M., Benhamouda, F., MacKenzie, P.: Security of the J-PAKE Password-Authenticated Key Exchange Protocol. In: 2015 IEEE Symposium on Security and Privacy (2015)
2. Abdalla, M., Benhamouda, F., Pointcheval, D.: Public-key encryption indistinguishable under plaintext-checkable attacks. In: Katz, J. (ed.) Public-Key Cryptography – PKC 2015, Lecture Notes in Computer Science, vol. 9020, pp. 332–352. Springer Berlin Heidelberg (2015), http://dx.doi.org/10.1007/978-3-662-46447-2_15
3. Abdalla, M., Catalano, D., Chevalier, C., Pointcheval, D.: Efficient two-party password-based key exchange protocols in the uc framework. In: Malkin, T. (ed.) Topics in Cryptology CT-RSA 2008, Lecture Notes in Computer Science, vol. 4964, pp. 335–351. Springer Berlin Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-79263-5_22
4. Abdalla, M., Fouque, P.A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) Public Key Cryptography - PKC 2005, Lecture Notes in Computer Science, vol. 3386, pp. 65–84. Springer Berlin Heidelberg (2005)
5. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure Against Dictionary Attacks. In: Preneel, B. (ed.) Advances in Cryptology – EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer (2000)
6. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) Advances in Cryptology – CRYPTO '93. LNCS, vol. 773, pp. 232–249. Springer (1993)
7. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security. pp. 62–73. CCS '93, ACM, New York, NY, USA (1993), http://doi.acm.org/10.1145/168588.168596
8. Bellovin, S.M., Merritt, M.: Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In: 1992 IEEE Computer Society Symposium on Research in Security and Privacy, May 4-6, 1992. pp. 72–84 (1992)
9. Boneh, D., zgr Dagdelen, Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 7073, pp. 41–69. Springer (2011)
10. 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: Security and Privacy (SP), 2015 IEEE Symposium on. pp. 553–570. IEEE (2015)
11. Boyko, V., MacKenzie, P.D., Patel, S.: Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In: Preneel, B. (ed.) Advances in Cryptology – EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer (2000)

12. Bresson, E., Chevassut, O., Pointcheval, D.: Security Proofs for an Efficient Password-based Key Exchange. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) ACM Conference on Computer and Communications Security. pp. 241–250. ACM (2003)
13. Bresson, E., Chevassut, O., Pointcheval, D.: New Security Results on Encrypted Key Exchange. In: Bao, F., Deng, R.H., Zhou, J. (eds.) Public Key Cryptography. LNCS, vol. 2947, pp. 145–158. Springer (2004)
14. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally Composable Password-Based Key Exchange. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer (2005)
15. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology. pp. 45–64. EUROCRYPT '02, Springer-Verlag, London, UK, UK (2002), http://dl.acm.org/citation.cfm?id=647087.715842
16. Diffie, W., Van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Des. Codes Cryptography 2(2), 107–125 (Jun 1992), http://dx.doi.org/10.1007/BF00124891
17. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) Public Key Cryptography PKC 2012, Lecture Notes in Computer Science, vol. 7293, pp. 467–484. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-30057-8_28
18. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. pp. 83–94. ASIA CCS '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2484313.2484323
19. Gennaro, R.: Faster and shorter password-authenticated key exchange. In: Canetti, R. (ed.) Theory of Cryptography, Lecture Notes in Computer Science, vol. 4948, pp. 589–606. Springer Berlin Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-78524-8_32
20. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) Advances in Cryptology EUROCRYPT 2003, Lecture Notes in Computer Science, vol. 2656, pp. 524–543. Springer Berlin Heidelberg (2003)
21. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th annual ACM symposium on Theory of computing. pp. 197–206. STOC '08, ACM, New York, NY, USA (2008)
22. Goldreich, O., Lindell, Y.: Session-key generation using human passwords only. In: Kilian, J. (ed.) Advances in Cryptology CRYPTO 2001, Lecture Notes in Computer Science, vol. 2139, pp. 408–432. Springer Berlin Heidelberg (2001), http://dx.doi.org/10.1007/3-540-44647-8_24
23. Goyal, V., Jain, A., Ostrovsky, R.: Password-authenticated session-key generation on the internet in the plain model. In: Proceedings of the 30th Annual Conference on Advances in Cryptology. pp. 277–294. CRYPTO'10, Springer-Verlag, Berlin, Heidelberg (2010), http://dl.acm.org/citation.cfm?id=1881412.1881432
24. Groce, A., Katz, J.: A new framework for efficient password-based authenticated key exchange. In: Proceedings of the 17th ACM Conference on Computer and Communications Security. pp. 516–525. CCS '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1866307.1866365
25. Halevi, S., Krawczyk, H.: Public-key cryptography and password protocols. ACM Trans. Inf. Syst. Secur. 2(3), 230–268 (Aug 1999), http://doi.acm.org/10.1145/322510.322514
26. Hao, F., Ryan, P.: J-pake: Authenticated key exchange without pki. In: Gavrilova, M., Tan, C., Moreno, E. (eds.) Transactions on Computational Science XI, Lecture Notes in Computer Science, vol. 6480, pp. 192–206. Springer Berlin Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-17697-5_10
27. Jablon, D.P.: Strong Password-Only Authenticated Key Exchange. ACM SIGCOMM Computer Communication Review 26(5), 5–26 (1996)
28. Jiang, S., Gong, G.: Password based key exchange with mutual authentication. In: Handschuh, H., Hasan, M. (eds.) Selected Areas in Cryptography, Lecture Notes in Computer Science, vol. 3357, pp. 267–279. Springer Berlin Heidelberg (2005)

29. Jintai Ding, Xiang Xie, X.L.: A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688 (2012), http://eprint.iacr.org/
30. Katz, J., Ostrovsky, R., Yung, M.: Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In: Pfitzmann, B. (ed.) Advances in Cryptology – EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer (2001)
31. Katz, J., Vaikuntanathan, V.: Smooth projective hashing and password-based authenticated key exchange from lattices. In: Matsui, M. (ed.) Advances in Cryptology ASIACRYPT 2009, Lecture Notes in Computer Science, vol. 5912, pp. 636–652. Springer Berlin Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-10366-7_37
32. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) Theory of Cryptography, Lecture Notes in Computer Science, vol. 6597, pp. 293–310. Springer Berlin Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-19571-6_18
33. Krawczyk, H.: Hmqv: A high-performance secure diffie-hellman protocol. In: Shoup, V. (ed.) Advances in Cryptology CRYPTO 2005, Lecture Notes in Computer Science, vol. 3621, pp. 546–566. Springer Berlin Heidelberg (2005), http://dx.doi.org/10.1007/11535218_33
34. Kwon, T.: Authentication and key agreement via memorable password. In: ISOC Network and Distributed System Security Symposium (2001)
35. Lucks, S.: Open key exchange: How to defeat dictionary attacks without encrypting public keys. In: Proceedings of the 5th International Workshop on Security Protocols. pp. 79–90. Springer-Verlag, London, UK, UK (1998), http://dl.acm.org/citation.cfm?id=647215.720526
36. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) Advances in Cryptology EUROCRYPT 2010, Lecture Notes in Computer Science, vol. 6110, pp. 1–23. Springer Berlin Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-13190-5_1
37. MacKenzie, P.: On the Security of the SPEKE Password-Authenticated Key Exchange Protocol. Cryptology ePrint Archive, Report 2001/057 (2001), http://eprint.iacr.org/2001/057
38. MacKenzie, P.: The PAK Suite: Protocols for Password-Authenticated Key Exchange. DIMACS Technical Report 2002-46 (2002), (Page 7)
39. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. 37, 267–302 (April 2007)
40. Nguyen, M.H., Vadhan, S.: Simpler session-key generation from short random passwords. In: Naor, M. (ed.) Theory of Cryptography, Lecture Notes in Computer Science, vol. 2951, pp. 428–445. Springer Berlin Heidelberg (2004), http://dx.doi.org/10.1007/978-3-540-24638-1_24
41. NSA: (2015), https://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml
42. Peikert, C.: An efficient and parallel gaussian sampler for lattices. Cryptology ePrint Archive, Report 2010/088 (2010), http://eprint.iacr.org/
43. Peikert, C.: Lattice cryptography for the internet. In: Mosca, M. (ed.) Post-Quantum Cryptography, Lecture Notes in Computer Science, vol. 8772, pp. 197–219. Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-11659-4_12
44. Shoup, V.: On Formal Models for Secure Key Exchange. Cryptology ePrint Archive, Report 1999/012 (1999), http://eprint.iacr.org/1999/012
45. Unruh, D.: Quantum position verification in the random oracle model. In: CRYPTO. pp. 1–18. Springer (2014)
46. Unruh, D.: Revocable quantum timed-release encryption. In: EUROCRYPT. pp. 129–146. Springer (2014)
47. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In: Advances in Cryptology - Crypto 2012. Lecture Notes in Computer Science, vol. 7417, pp. 758–775. Springer (2012)
48. Zhang, J., Zhang, Z., Ding, J., Snook, M., Dagdelen, z.: Authenticated key exchange from ideal lattices. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015, Lecture Notes in Computer Science, vol. 9057, pp. 719–751. Springer Berlin Heidelberg (2015), http://dx.doi.org/10.1007/978-3-662-46803-6_24