

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

Jintai Ding^{1(✉)}, Saed Alsayigh¹, Jean Lancrenon²,
Saraswathy RV¹, and Michael Snook¹

¹ University of Cincinnati, Cincinnati, USA

jintai.ding@gmail.com

² University of Luxembourg, Luxembourg City, Luxembourg

Abstract. Authenticated Key Exchange (AKE) is a cryptographic scheme with the aim to establish a high-entropy and secret session key over an 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 [19] to construct two lattice-based PAKE protocols enjoying a very simple and elegant design that is a parallel extension of the class of Random Oracle Model (ROM)-based protocols PAK and PPK [13, 41], 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.

Keywords: 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 [18], or HMQV [35]), 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 [7]).

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 [6, 13], and a plethora of protocols have been designed and analyzed (e.g., [1, 14, 31, 34, 41]). The current pool of practical protocols¹ can essentially be classified into two categories: the first we shall call the class of *Random Oracle Model* (ROM)-based PAKEs (such as [3, 6, 9, 14, 15, 30, 41]), and the second, the class of *Common Reference String* (CRS)-based PAKEs (such as [16, 23, 27, 32, 34]). Roughly, the protocols in the first category have very simple and elegant designs, but rely crucially on the ROM [8] for their proofs of security,

¹ Some impractical yet complexity-theoretically efficient protocols have been studied, for theoretical reasons. See e.g. [25, 26, 43].

while the protocols in the second category use sophisticated cryptographic tools² to achieve standard-model security (assuming a CRS is in place³). 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) [44] 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 the 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 [33] relies instead on lattice-type assumptions for its security. Unfortunately, it is CRS-based, and therefore not very efficient.

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 [13, 41] 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 [19], and used in [50] 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

² In particular, they use *universal hash proof systems* [17] over complex languages.

³ 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.

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. [6]. 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, and allows us to build intermediate steps that did not appear in the original proofs for PAK and PPK.

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 with lattice-handling analogues. The 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 [19], 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.

Finally, we created a proof-of-concept implementation of our new PAKE (and the “implicit” version) to demonstrate its efficiency and practicality. This part is moved to Appendix E of the full version due the lack of room.

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 Bellare and Merritt in [9]. 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 [30] 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 [38] and Halevi and Krawczyk [28]. 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. [6] and Boyko et al. [13], building respectively on the AKE models proposed by Bellare et al. in [7] and Shoup in [46]. The model in [6] was further refined by Abdalla et al. in [4]. The notion of universally composable PAKE has also since been introduced by Canetti et al. [16].

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 [2, 3, 6, 13–15, 29, 30, 36, 40, 41]. On one hand, these are mostly two-or-three-pass protocols, with very few group operations. For instance, the explicitly authenticated PAK protocol in [41] 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 assumptions⁴. In 2001, a milestone was reached with the work of Katz et al. [32], 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 [22, 23, 27, 31, 34], all using a CRS. It was further shown in [16] 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. [34] 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 [27] down to 6 and 18 respectively, and those of [34] 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. [25] were the first to answer in the affirmative, but assuming non-concurrent protocol executions. Their work was followed up by Nguyen et al. [43], who found a more efficient construction, but in a weaker model. Later, Jain et al. [26] 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 than even the CRS-based protocols.

⁴ The ROM is one of them; another is the *ideal cipher* model, see [6].

AKE from Lattices. Some work was done to address the problem of finding AKE protocols based on lattice-type assumptions. The protocols in [20, 21, 37] are essentially lattice-based instantiations of generic constructions that use key-encapsulation mechanisms to construct AKEs. In 2012, Ding et al. [19] first proposed simple LWE and RLWE analogues of the unauthenticated Diffie-Hellman protocol. Later there appeared a few variants of Ding’s key exchange [5, 11, 12, 45], with the slight modification that the new rounding technique from [19] for least significant bits was adjusted to work for most significant bits. A true LWE-based AKE was proposed in Zhang et al. [50], where the protocol proposed by Ding et al. [19] was leveraged to build a 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. [33]. 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 *equipped with a universal hash proof system* [17], and the heart of [33] is essentially a lattice-based instantiation of such a scheme.

2 Preliminaries

2.1 Security Model

Here, we review the security model from [6]. 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 \mathcal{U} of *users*, partitioned into two non-empty sets \mathcal{C} of *clients* and \mathcal{S} of *servers*. We also assume some fixed, non-empty dictionary D of size L . Before the game starts, for each $C \in \mathcal{C}$ a password pw_C is drawn uniformly at random from D and assigned to C outside of \mathcal{A} ’s view. For each server $S \in \mathcal{S}$, we set $pw_S := (f(pw_C))_C$, where 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 $\Pi_{\mathcal{U}}^i$, 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 $\Pi_{\mathcal{U}}^i$ may *accept*. When this happens, it holds a *Partner IDentity* (PID) $pid_{\mathcal{U}}^i$, a *Session IDentity* (SID) $sid_{\mathcal{U}}^i$, and a *Session Key* (SK) $sk_{\mathcal{U}}^i$. The PID is the identity of the user that instance believes it is talking to. The SK is what $\Pi_{\mathcal{U}}^i$ 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. Often the SID is defined as the ordered concatenation of messages sent and received by an instance, except possibly the last message. (In our case, we will need to modify this a bit.)

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

- **Send**(\mathcal{U}, i, M): Causes message M to be sent to instance $\Pi_{\mathcal{U}}^i$. 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 $\Pi_{\mathcal{U}}^i$ to accept or terminate.
- **Execute**($\mathcal{C}, i, \mathcal{S}, j$): Causes P to be executed to completion between $\Pi_{\mathcal{C}}^i$ (where $\mathcal{C} \in \mathfrak{C}$) and $\Pi_{\mathcal{S}}^j$ (where $\mathcal{S} \in \mathfrak{S}$) and hands \mathcal{A} the execution's transcript.
- **Reveal**(\mathcal{U}, i): Returns the SK $sk_{\mathcal{U}}^i$ held by $\Pi_{\mathcal{U}}^i$ to \mathcal{A} .
- **Test**(\mathcal{U}, i): For this query to be valid, instance $\Pi_{\mathcal{U}}^i$ 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 $sk_{\mathcal{U}}^i$ 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.
- **Corrupt**(\mathcal{U}): Returns $(f(pw_{\mathcal{C}}))_{\mathcal{C}}$ 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)** $\text{pid}_{\mathcal{U}}^i = \mathcal{V}$ and $\text{pid}_{\mathcal{V}}^j = \mathcal{U}$, **(iv)** $\text{sid}_{\mathcal{U}}^i = \text{sid}_{\mathcal{V}}^j =: \text{sid}$ 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)** $\text{Reveal}(\mathcal{U}, i)$ was queried, **(ii)** a $\text{Reveal}(\mathcal{V}, j)$ was queried, where $\Pi_{\mathcal{V}}^j$ is $\Pi_{\mathcal{U}}^i$'s partner, if it has one, or **(iii)** $\text{Corrupt}(\mathcal{V})$ was queried for some \mathcal{V} before the Test query and a $\text{Send}(\mathcal{U}, i, M)$ 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 $\text{Succ}_{\mathsf{P}}^{\text{ake}}(\mathcal{A})$ be the event:

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

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

$$\text{Adv}_{\mathsf{P}}^{\text{ake}}(\mathcal{A}) = 2\text{Pr}[\text{Succ}_{\mathsf{P}}^{\text{ake}}(\mathcal{A})] - 1$$

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

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 [39]. 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 [50].

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.

We 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 $\|s \cdot t\| \leq \sqrt{n} \cdot \|s\| \cdot \|t\|$ and $\|s \cdot t\|_\infty \leq n \cdot \|s\|_\infty \cdot \|t\|_\infty$.*

Lemma 2 ([24, 42]). *For any real number $\alpha = \omega(\sqrt{\log n})$, we have $\Pr_{\mathbf{x} \leftarrow \chi_\alpha}[\|\mathbf{x}\| > \alpha\sqrt{n}] \leq 2^{-n+1}$.*

We now recall the **Cha** and **Mod₂** functions defined in [50]. We denote $\mathbb{Z}_q = \{-\frac{q-1}{2}, \dots, \frac{q-1}{2}\}$ and consider the set $E := \{-\lfloor \frac{q}{4} \rfloor, \dots, \lfloor \frac{q}{4} \rfloor\}$, 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 is in E and 1 if it is not in E , and that **Mod₂**: $\mathbb{Z}_q \times \{0, 1\} \rightarrow \{0, 1\}$ is defined as:

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

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

Lemma 3 ([50]). *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 **Mod₂**($v+v'$, b) given **Cha**(v) is statistically close to uniform on $\{0, 1\}$.*

Lemma 4 ([50]). *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 $\text{Mod}_2(v, \text{Cha}(v)) = \text{Mod}_2(w, \text{Cha}(v))$.*

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 $v = (v_0, \dots, v_{n-1}) \in R_q$ and binary-vector $\mathbf{b} = (b_0, \dots, b_{n-1}) \in \{0, 1\}^n$, we set $\text{Cha}(v) = (\text{Cha}(v_0), \dots, \text{Cha}(v_{n-1}))$ and $\text{Mod}_2(v, \mathbf{b}) = (\text{Mod}_2(v_0, b_0), \dots, \text{Mod}_2(v_{n-1}, b_{n-1}))$.

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 Sect. 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 $\tau(X, s) = \text{Mod}_2(Xs, \text{Cha}(Xs))$. 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 $\text{Cha}(Xs)$ itself. Formally, let

$$\text{Adv}_{R_q}^{\text{PWE}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr \left[a \leftarrow R_q; s \leftarrow \chi_\beta; X \leftarrow R_q; e \leftarrow \chi_\beta; \right. \\ \left. Y \leftarrow as + 2e; W \leftarrow \text{Cha}(Xs) : \tau(X, s) \in \mathcal{A}(a, X, Y, W) \right]$$

Let $\text{Adv}_{R_q}^{\text{PWE}}(t, N) = \max_{\mathcal{A}} \left\{ \text{Adv}_{R_q}^{\text{PWE}}(\mathcal{A}) \right\}$, 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 , $\text{Adv}_{R_q}^{\text{PWE}}(t, N)$ 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 3. (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 $w = \text{Cha}K$ for some $K \in R_q$ and $\sigma = \text{Mod}_2(K, w)$. The Decision Pairing with Errors problem (DPWE) is to decide whether $K = Xs + 2g$ and $Y = as + 2e$ for some $s, g, \text{ and } e$ 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 4. (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 .

Now we state the reduction theorems without proof due to the lack of the space. Look at Appendix A of the full version for the proof details.

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.

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.

As a result from Theorems 1 and 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 studying the protocols RLWE-PAK and RLWE-PPK, and 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 $H_1: \{0, 1\}^* \rightarrow R_q$ be a hash function, $H_l: \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$ for $l \in \{2, 3\}$ be hash functions for verification of communications, and $H_4: \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$ 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 [50] 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:

Initiation. Client \mathcal{C} randomly samples $s_C, e_C \leftarrow \chi_\beta$, computes $\alpha = as_C + 2e_C$, $\gamma = H_1(pw_C)$ and $m = \alpha + \gamma$ and sends $\langle \mathcal{C}, m \rangle$ to party \mathcal{S} .

Response. Server \mathcal{S} receives $\langle \mathcal{C}, m \rangle$ from party \mathcal{C} and checks that $m \in R_q$; if not, it aborts. Otherwise it computes $\alpha = m + \gamma'$ where $\gamma' = -H_1(pw_C)$. Server \mathcal{S} then randomly samples $s_S, e_S \leftarrow \chi_\beta$ and computes $\mu = as_S + 2e_S$ and $k_S = \alpha \cdot s_S$.

Next, Server \mathcal{S} computes $w = \text{Cha}(k_S) \in \{0, 1\}^n$ and $\sigma = \text{Mod}_2(k_S, w)$. 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 $k_C = s_C \cdot \mu$ and $\sigma = \text{Mod}_2(k_C, w)$. Client \mathcal{C} verifies that $H_2(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$ 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 $sk_C = H_4(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$. It then sends k' back to Server \mathcal{S} , and sets $sid_C = (\mathcal{C}, \mathcal{S}, m, \mu)$.

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 $sk_S = H_4(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma')$. It sets $sid_S = (\mathcal{C}, \mathcal{S}, m, \mu)$ ⁵. Otherwise, \mathcal{S} refuses to compute a session key.

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} , honestly follow the protocol described above. Then, the two will end with the same key with overwhelming probability.*

Proof. To show the correctness of RLWE-PAK, it is sufficient to show that the key material derived at each end verifies $\text{Mod}_2(k_C, \text{Cha}(k_S)) = \text{Mod}_2(k_S, \text{Cha}(k_S))$. By Lemma 4, if k_C and k_S are sufficiently close then we are done. Specifically, if $|k_C - k_S| < q/4$ then both sides have the same value, σ . If we compare the two, we find that $k_C - k_S = 2[e_S s_C - e_C s_S]$. By Lemma 2, each individual e_S, s_C, e_C, s_S

⁵ We purposefully excluded the hint w from the session identifier in order to avoid a trivial bit-flipping attack that makes the proof fail in theory, but otherwise leaves the protocol security unaffected.

term has norm less than $\beta\sqrt{n}$ with overwhelming probability. Applying Lemma 1 and the triangle inequality, we have that $\|k_C - k_S\| \leq 4\beta^2 n^{3/2} < q/4$ with overwhelming probability. Hence $\text{Mod}_2(k_C, \text{Cha}(k_S)) = \text{Mod}_2(k_S, \text{Cha}(k_S))$. \square

4 Proof of Security for RLWE-PAK

Our proof of security follows the one in the PAK suite paper by MacKenzie [41]. We essentially adapt it to our PWE instantiation. The objective is to show that an adversary \mathcal{A} attacking the system is unable to gain any information on the SK of a fresh instance with a greater advantage than through 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 Π_C^i to send the first message to some \mathcal{S} ;
- **SA1** query if it sends some message to a previously unused Π_S^j ;
- **CA1** query if it sends a message to some Π_C^i expecting the second protocol message;
- **SA2** query if it sends some message to a Π_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:

- $\text{testpw}(\mathcal{C}, i, \mathcal{S}, pw, l)$: 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 Π_C^i with input \mathcal{S} and output $< \mathcal{C}, m >$, a CA1 query to Π_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 $H_l(\cdot)$ query or the CA1 query. $\sigma = \text{Mod}_2(k_S, w) = \text{Mod}_2(k_C, w)$, $k_S = \alpha s_S, k_C = \mu s_C$ and $m = \alpha - \gamma'$. The associated value of this event is output of $H_l(\cdot), l \in \{2, 3, 4\}$.
- $\text{testpw}!(\mathcal{C}, i, \mathcal{S}, pw)$: for some w and k a CA1 query with input $< \mu, k, w >$ causes a $\text{testpw}(\mathcal{C}, i, \mathcal{S}, pw, 2)$ event to occur, with associated value k .
- $\text{testpw}(\mathcal{S}, j, \mathcal{C}, pw, l)$: 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 Π_S^j with input $< \mathcal{C}, m >$ and output $< \mu, k, w >$, and an $H_1(pw)$ query returning $-\gamma'$, where $\sigma = \text{Mod}_2(k_S, w) = \text{Mod}_2(k_C, w)$, $k_S = \alpha s_S, k_C = \mu s_C$ and $m = \alpha - \gamma'$. The associated value of this event is output of $H_l(\cdot), l \in \{2, 3, 4\}$ generated by Π_S^j .
- $\text{testpw}!(\mathcal{S}, j, \mathcal{C}, pw)$: a SA2 query to Π_S^j is made with k' , where a $\text{testpw}(\mathcal{S}, j, \mathcal{C}, pw, 3)$ event previously occurred with associated value k' .
- $\text{testpw}^*(\mathcal{S}, j, \mathcal{C}, pw)$: $\text{testpw}(\mathcal{S}, j, \mathcal{C}, pw, l)$ occurs for some $l \in \{2, 3, 4\}$.
- $\text{testpw}(\mathcal{C}, i, \mathcal{S}, j, pw)$: for some $l \in \{2, 3, 4\}$, both a $\text{testpw}(\mathcal{C}, i, \mathcal{S}, pw, l)$ event and a $\text{testpw}(\mathcal{S}, j, \mathcal{C}, pw, l)$ event occur, where Π_C^i is paired with Π_S^j and Π_S^j is paired with Π_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 $Execute(\mathcal{C}, i, \mathcal{S}, j)$ query that generates m and μ and an $H_1(pw)$ query returning $-\gamma' = as_h + 2e_h \in R_q$, where $\sigma = \text{Mod}_2(k_S, w) = \text{Mod}_2(k_C, w)$, $k_S = \alpha s_S$, $k_C = \mu s_C$ and $m = \alpha - \gamma'$.
- $correctpw$: before any $Corrupt$ query, either a $testpw!(\mathcal{C}, i, \mathcal{S}, pw)$ event occurs for some \mathcal{C}, i and \mathcal{S} , or a $testpw^*(\mathcal{S}, j, \mathcal{C}, pw_C)$ event occurs for some \mathcal{S}, j , and \mathcal{C} .
- $correctpwexec$: a $testexecpw(\mathcal{C}, i, \mathcal{S}, j, pw_C)$ event occurs for some $\mathcal{C}, i, \mathcal{S}$, and j .
- $doublepwserver$: before any $Corrupt$ 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 $testpw(\mathcal{C}, i, \mathcal{S}, j, pw_C)$ event occurs, for some $\mathcal{C}, i, \mathcal{S}$, and j .

Theorem 4. *Let $P := RLWE\text{-PAK}$, 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 $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$:*

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

Proof. We study a sequence of protocols - P_0, P_1, \dots, P_7 - with the following properties. First $P_0 = P$ and P_7 is by design only possible to attack using natural online guessing. Secondly, we have

$$Adv_{P_0}^{ake}(\mathcal{A}) \leq Adv_{P_1}^{ake}(\mathcal{A}) + \epsilon_1 \leq \dots \leq Adv_{P_7}^{ake}(\mathcal{A}) + \epsilon_7$$

where $\epsilon_1, \dots, \epsilon_7$ 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. The reader can find the proofs of the claims in Appendix C of the full version.

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 sequence 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 P₁: P₁ is nearly identical to P₀, 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 $H_l(\cdot)$ 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 $H_l(\cdot)$ query. Setting $E = E_1 \vee E_2$ then P₁ is defined as being identical to P₀ except that the protocol halts and the adversary fails when E occurs.

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

$$Adv_{P_0}^{ake}(\mathcal{A}) \leq Adv_{P_1}^{ake}(\mathcal{A}) + \frac{O((n_{se} + n_{ex})(n_{ro} + n_{se} + n_{ex}))}{q^n}$$

Protocol P₂: This protocol is identical to P₁ 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 P₂ 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 $H_l(\cdot)$ 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} ,

$$Adv_{P_1}^{ake}(\mathcal{A}) = Adv_{P_2}^{ake}(\mathcal{A}) + \frac{O(n_{ro})}{q^n} + \frac{O(n_{se})}{2^{\kappa}}$$

Protocol P₃: is identical to P₂ 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,*

$$Adv_{P_2}^{ake}(\mathcal{A}) \leq Adv_{P_3}^{ake}(\mathcal{A}) + Adv_{R_q}^{DRLWE}(t', n_{ro}) + 2Adv_{R_q}^{PWE}(t', n_{ro})$$

Protocol P₄: is identical to P₃ except that if $correctpw$ occurs then the protocol halts and the adversary automatically succeeds. This causes theses changes:

1. In a CA1 query to $\Pi_{\mathcal{C}}^i$, if a $testpw!(\mathcal{C}, i, \mathcal{S}, pw_{\mathcal{C}})$ event occurs and no **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 **Corrupt** query has been made, halt and say the adversary automatically succeeds.

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

$$Adv_{P_3}^{ake}(\mathcal{A}) \leq Adv_{P_4}^{ake}(\mathcal{A})$$

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

Protocol P₅: is identical to P₄ 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 protocol 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 $H_l(\cdot)$ 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,*

$$Adv_{P_4}^{ake}(\mathcal{A}) \leq Adv_{P_5}^{ake}(\mathcal{A}) + 2n_{se}Adv_{R_q}^{PWE}(t', n_{ro})$$

Protocol P₆: is identical to P₅ 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,*

$$Adv_{P_5}^{ake}(\mathcal{A}) \leq Adv_{P_6}^{ake}(\mathcal{A}) + 4Adv_{R_q}^{PWE}(t', n_{ro}^2)$$

Protocol P₇: is identical to P₆ 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 $Corrupt(U)$ queries whether $U \in \mathfrak{S}$ or $U \in \mathfrak{C}$. When a $Corrupt(U)$ query made in the protocol, it is answered using a $Corrupt(U)$ 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} ,

$$Adv_{P_6}^{ake}(\mathcal{A}) = Adv_{P_7}^{ake}(\mathcal{A})$$

Proof. By observation, P₆ and P₇ are perfectly indistinguishable. \square

Now we analyze the advantage of an adversary \mathcal{A} against the protocol P₇. From the definition of P₇, one can easily bounds the probability of adversary \mathcal{A} succeeding in P₇ as the following:

$$Pr(Succ_{P_7}^{ake}(\mathcal{A})) \leq Pr(correctpw) + Pr(Succ_{P_7}^{ake}(\mathcal{A}) \mid \neg correctpw)Pr(\neg correctpw).$$

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 $Pr(Succ_{P_7}^{ake}(\mathcal{A}) \mid \neg correctpw)$. 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 $sk_{\mathcal{U}}^i$ then the probability of success is exactly $\frac{1}{2}$ and to do that we have to examine $Reveal$ and $H_4(\cdot)$ queries.

For the first type, we know by definition of $Reveal(U, i)$ query that there could be no one for the fresh instance $\Pi_{\mathcal{U}}^i$. Also there is no $Reveal(U', j)$ query for the instance $\prod_j^{U'}$ which is partnered with $\Pi_{\mathcal{U}}^i$. Moreover the adversary fails if more than a single client instance and a single server instance accept with the same *sid* by protocol P₁. Thus the output of $Reveal$ queries is independent of $sk_{\mathcal{U}}^i$.

For the second type, from P₄ 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 $\Pi_{\mathcal{U}}^i$ is partnered, $H_4(\cdot)$ query will never reveal $sk_{\mathcal{U}}^i$ by P₅.

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

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

And $\text{Adv}_{\mathcal{P}_7}^{\text{ake}}(\mathcal{A}) \leq \frac{n_{se}}{L}$. The theorem follows from this and the Claims 1, 2, 3, 4, 5, 6 and 7 above. \square

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. 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: $f(\cdot) = (-H_1(\cdot), H_2(\cdot))$.

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

Response. Server \mathcal{S} receives $\langle \mathcal{C}, m \rangle$ 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 $\langle \gamma'_1, \gamma_2 \rangle$. Then compute $\mu = \nu + \gamma_2$ and $k_{\mathcal{S}} = \alpha \cdot s_{\mathcal{S}}$.

Next, Server \mathcal{S} computes $w = \text{Cha}(k_{\mathcal{S}}) \in \{0, 1\}^n$ and $\sigma = \text{Mod}_2(k_{\mathcal{S}}, w)$. 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 $\langle \mu, w \rangle$ 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 $\sigma = \text{Mod}_2(k_{\mathcal{C}}, w)$.

Finally, Client \mathcal{C} derives the session key $sk_{\mathcal{C}} = H_3(\mathcal{C}, \mathcal{S}, m, \mu, \sigma, \gamma'_1)$.

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 [41], and is similar to our proof

for the explicitly authenticated protocol above. Therefore we will not go through the proof details. However we give a sketch of the proof in Appendix D of the full version. We first define some similar events to those in Sect. 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. Then we need to show that an adversary attacking the system is unable to determine the session key of a fresh instance with greater advantage than that in an online dictionary attack.

Theorem 5. *Let $P := \text{RLWE-PPK}$ as described above, 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 $t' = O(t + (n_{ro} + n_{se} + n_{ex})t_{exp})$:*

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

For space limitation reasons, the proof of this theorem and more details regarding the security of RLWE-PPK were moved to Appendix D of the full version.

6 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 Random 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) [10]. 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 [47–49], more work is needed to adapt classical proofs to this setting.

Acknowledgments. Many thanks to the reviewers for their comments and Peter Ryan for the useful discussions. We would also like to thank the NSF for its partial support. Finally, the third author was supported by the National Research Fund, Luxembourg (CORE project aToMS and INTER project Sequoia).

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.) PKC 2015. LNCS, vol. 9020, pp. 332–352. Springer, Berlin (2015). doi:[10.1007/978-3-662-46447-2_15](https://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.) CT-RSA 2008. LNCS, vol. 4964, pp. 335–351. Springer, Berlin (2008). doi:[10.1007/978-3-540-79263-5_22](https://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.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Berlin (2005). doi:[10.1007/978-3-540-30580-4_6](https://doi.org/10.1007/978-3-540-30580-4_6)
5. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange—a new hope. In: 25th USENIX Security Symposium, USENIX Security 16, pp. 327–343 (2016)
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Berlin (2000). doi:[10.1007/3-540-45539-6_11](https://doi.org/10.1007/3-540-45539-6_11)
7. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Berlin (1994). doi:[10.1007/3-540-48329-2_21](https://doi.org/10.1007/3-540-48329-2_21)
8. 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, CCS 1993, pp. 62–73. ACM, New York (1993)
9. Bellare, 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, pp. 72–84, 4–6 May 1992
10. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Berlin (2011). doi:[10.1007/978-3-642-25385-0_3](https://doi.org/10.1007/978-3-642-25385-0_3)
11. Bos, J., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM (2016)
12. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: 2015 IEEE Symposium on Security and Privacy (SP), pp. 553–570. IEEE (2015)
13. Boyko, V., MacKenzie, P., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Berlin (2000). doi:[10.1007/3-540-45539-6_12](https://doi.org/10.1007/3-540-45539-6_12)
14. Bresson, E., Chevassut, O., Pointcheval, D.: Security proofs for an efficient password-based key exchange. In: ACM Conference on Computer and Communications Security. ACM (2003)

15. Bresson, E., Chevassut, O., Pointcheval, D.: New security results on encrypted key exchange. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 145–158. Springer, Berlin (2004). doi:[10.1007/978-3-540-24632-9_11](https://doi.org/10.1007/978-3-540-24632-9_11)
16. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Berlin (2005). doi:[10.1007/11426639_24](https://doi.org/10.1007/11426639_24)
17. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Berlin (2002). doi:[10.1007/3-540-46035-7_4](https://doi.org/10.1007/3-540-46035-7_4)
18. Diffie, W., Van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. *Des. Codes Crypt.* **2**, 107–125 (1992)
19. Ding, J., Xie, X., Lin, X.: A simple provably secure key exchange scheme based on the learning with errors problem. *Cryptology ePrint Archive*, Report 2012/688 (2012)
20. 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.) PKC 2012. LNCS, vol. 7293, pp. 467–484. Springer, Berlin (2012). doi:[10.1007/978-3-642-30057-8_28](https://doi.org/10.1007/978-3-642-30057-8_28)
21. 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, ASIA CCS 2013, pp. 83–94. ACM, New York (2013)
22. Gennaro, R.: Faster and shorter password-authenticated key exchange. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 589–606. Springer, Berlin (2008). doi:[10.1007/978-3-540-78524-8_32](https://doi.org/10.1007/978-3-540-78524-8_32)
23. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Berlin (2003). doi:[10.1007/3-540-39200-9_33](https://doi.org/10.1007/3-540-39200-9_33)
24. 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, STOC 2008, pp. 197–206. ACM, New York (2008)
25. Goldreich, O., Lindell, Y.: Session-key generation using human passwords only. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 408–432. Springer, Berlin (2001). doi:[10.1007/3-540-44647-8_24](https://doi.org/10.1007/3-540-44647-8_24)
26. Goyal, V., Jain, A., Ostrovsky, R.: Password-authenticated session-key generation on the internet in the plain model. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 277–294. Springer, Berlin (2010). doi:[10.1007/978-3-642-14623-7_15](https://doi.org/10.1007/978-3-642-14623-7_15)
27. 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, CCS 2010, pp. 516–525. ACM, New York (2010)
28. Halevi, S., Krawczyk, H.: Public-key cryptography and password protocols. *ACM Trans. Inf. Syst. Secur.* **2**, 230–268 (1999)
29. Hao, F., Ryan, P.: J-PAKE: authenticated key exchange without PKI. In: Gavrillova, M.L., Tan, C.J.K., Moreno, E.D. (eds.) Transactions on Computational Science XI. LNCS, vol. 6480, pp. 192–206. Springer, Berlin (2010). doi:[10.1007/978-3-642-17697-5_10](https://doi.org/10.1007/978-3-642-17697-5_10)
30. Jablon, D.P.: Strong password-only authenticated key exchange. *ACM SIGCOMM Comput. Commun. Rev.* **5**, 5–26 (1996)

31. Jiang, S., Gong, G.: Password based key exchange with mutual authentication. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 267–279. Springer, Berlin (2004). doi:[10.1007/978-3-540-30564-4_19](https://doi.org/10.1007/978-3-540-30564-4_19)
32. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Berlin (2001). doi:[10.1007/3-540-44987-6_29](https://doi.org/10.1007/3-540-44987-6_29)
33. Katz, J., Vaikuntanathan, V.: Smooth projective hashing and password-based authenticated key exchange from lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 636–652. Springer, Berlin (2009). doi:[10.1007/978-3-642-10366-7_37](https://doi.org/10.1007/978-3-642-10366-7_37)
34. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer, Berlin (2011). doi:[10.1007/978-3-642-19571-6_18](https://doi.org/10.1007/978-3-642-19571-6_18)
35. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Berlin (2005). doi:[10.1007/11535218_33](https://doi.org/10.1007/11535218_33)
36. Kwon, T.: Authentication and key agreement via memorable password. In: ISOC Network and Distributed System Security Symposium (2001)
37. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Berlin (2011). doi:[10.1007/978-3-642-19074-2_21](https://doi.org/10.1007/978-3-642-19074-2_21)
38. Lucks, S.: Open key exchange: how to defeat dictionary attacks without encrypting public keys. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 79–90. Springer, Berlin (1998). doi:[10.1007/BFb0028161](https://doi.org/10.1007/BFb0028161)
39. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Berlin (2010). doi:[10.1007/978-3-642-13190-5_1](https://doi.org/10.1007/978-3-642-13190-5_1)
40. MacKenzie, P.: On the Security of the SPEKE Password-Authenticated Key Exchange Protocol. Cryptology ePrint Archive, Report 2001/057 (2001)
41. MacKenzie, P.: The PAK Suite: Protocols for Password-Authenticated Key Exchange. DIMACS Technical report 2002-46 (2002). p. 7
42. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. SIAM J. Comput. **37**, 267–302 (2007)
43. Nguyen, M.-H., Vadhan, S.: Simpler session-key generation from short random passwords. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 428–445. Springer, Berlin (2004). doi:[10.1007/978-3-540-24638-1_24](https://doi.org/10.1007/978-3-540-24638-1_24)
44. NSA: Commercial national security algorithm suite (2015). <https://www.iad.gov/iad/programs/iad-initiatives/cnsa-suite.cfm>
45. Peikert, C.: Lattice cryptography for the internet. In: Mosca, M. (ed.) PQCrypto 2014. LNCS, vol. 8772, pp. 197–219. Springer, Cham (2014). doi:[10.1007/978-3-319-11659-4_12](https://doi.org/10.1007/978-3-319-11659-4_12)
46. Shoup, V.: On Formal Models for Secure Key Exchange. Cryptology ePrint Archive, Report 1999/012 (1999)
47. Unruh, D.: Quantum position verification in the random oracle model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 1–18. Springer, Berlin (2014). doi:[10.1007/978-3-662-44381-1_1](https://doi.org/10.1007/978-3-662-44381-1_1)
48. Unruh, D.: Revocable quantum timed-release encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 129–146. Springer, Berlin (2014). doi:[10.1007/978-3-642-55220-5_8](https://doi.org/10.1007/978-3-642-55220-5_8)

49. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 758–775. Springer, Berlin (2012). doi:[10.1007/978-3-642-32009-5_44](https://doi.org/10.1007/978-3-642-32009-5_44)
50. Zhang, J., Zhang, Z., Ding, J., Snook, M., Dagdelen, Ö.: Authenticated key exchange from ideal lattices. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 719–751. Springer, Berlin (2015). doi:[10.1007/978-3-662-46803-6_24](https://doi.org/10.1007/978-3-662-46803-6_24)