




Universal Composable Password Authenticated Key Exchange for the Post-Quantum World

You Lyu^{1,2}, Shengli Liu^{1,2}, and Shuai Han^{2,3}

¹ Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai 200240, China

² State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

³ School of Cyber Science and Engineering, Shanghai Jiao Tong University,
Shanghai 200240, China

{[vergil](mailto:vergil@sjtu.edu.cn), [slliu](mailto:slliu@sjtu.edu.cn), [dalen17](mailto:dalen17@sjtu.edu.cn)}@sjtu.edu.cn

Abstract. In this paper, we construct the *first* password authenticated key exchange (PAKE) scheme from isogenies with Universal Composable (UC) security in the random oracle model (ROM). We also construct the *first* two PAKE schemes with UC security in the quantum random oracle model (QROM), one is based on the learning with error (LWE) assumption, and the other is based on the group-action decisional Diffie-Hellman (GA-DDH) assumption in the isogeny setting.

To obtain our UC-secure PAKE scheme in ROM, we propose a generic construction of PAKE from basic lossy public key encryption (LPKE) and CCA-secure PKE. We also introduce a new variant of LPKE, named extractable LPKE (eLPKE). By replacing the basic LPKE with eLPKE, our generic construction of PAKE achieves UC security in QROM. The LPKE and eLPKE have instantiations not only from LWE but also from GA-DDH, which admit four specific PAKE schemes with UC security in ROM or QROM, based on LWE or GA-DDH.

1 Introduction

Password Authenticated Key Exchange (PAKE) enables two parties (say, a client and a server) who possess a low-entropy password pw to securely establish session keys over public networks. These session keys subsequently facilitate the establishment of secure communication channels. Unlike authenticated key exchange (AKE), which necessitates a Public Key Infrastructure (PKI) to verify the authenticity of public keys, PAKE runs with easily memorable passwords and offers enhanced convenience for deployments and applications.

Security Notions for PAKE: IND vs. UC. There are two primary security notions for PAKE, the game-based security in the Indistinguishability model (IND security) [9] and the simulation-based security under the Universally Composable framework (UC security) [17]. As shown in [17], the UC security in the UC framework implies the IND security. In contrast to the IND model which assumes passwords uniformly distributed over a set, the UC framework permits

arbitrary correlations and distributions for passwords and guarantees security amidst composition with arbitrary protocols and hence is a better security model.

PAKE with Post-Quantum Security. There are quite a few IND-secure PAKE schemes constructed from post-quantum assumptions, including lattice-based [11,26,37,25,20] and isogeny-based [3] ones.

As for UC security, there exists a generic construction for PAKE from Oblivious Transfer (OT) [16]. However, achieving UC security for PAKE requires the underlying OT protocol have adaptive UC security even in absence of authenticated channels. As far as we know, up to now such OT only has instantiations from number-theoretic assumptions like CDH or Factoring, so no instantiation from post-quantum assumptions via the OT approach is known for PAKE. There also exist frameworks of constructing PAKE from Hash Proof System (HPS) (e.g., [2]). To apply the HPS framework, one needs an HPS for language consisting of pairs of messages and commitments/ciphertexts (m, c) , but as far as we know there is no suitable HPS from lattice/isogenies serving for UC-secure PAKE. For example, in the isogeny setting, the existing commitment/encryption schemes (e.g., [18], Section 7.1 in [12]) need to *hash* a set element to mask the message using XOR, i.e., $c = m \oplus H(\text{set element})$, which destroys the algebraic structures and makes it hard to build HPS for (m, c) of this form. The group-action/isogeny-based HPS proposed in [5] is for the DDH-type language $(x_0, x_1, s \star x_0, s \star x_1)$, which is inherently different from the one needed in the HPS framework and can hardly yield PAKE from group actions/isogenies. A feasible approach to PAKE from post-quantum assumptions is making use of Encrypted Key Exchange [10] (EKE) and resorting to the Ideal Cipher Model (ICM) to achieve UC security. This EKE approach results in two UC-secure PAKE schemes [33,8] both of which are lattice-based. However, the ICM has two limitations and this leads to two questions.

- It is unclear how to instantiate ideal cipher from isogenies, as highlighted in [7]. Hence isogeny-based PAKE with UC security is now missing.
Q1: Can we construct UC-secure PAKE from isogeny-based assumptions?
- ICM does not consider quantum access from adversaries and it is unclear how to exploit quantum ICM (QICM) to achieve security against quantum algorithms. But quantum random oracle model (QROM) [14] takes into account quantum-access adversaries, and is better understood than QICM [24].
Q2: Can we construct PAKE protocols with UC security in QROM ?

Our Contribution. We answer the above two questions in this paper with the following two folds of contributions.

1. We propose a generic construction for UC-secure PAKE in the random oracle model (ROM) from two building blocks, a basic Lossy Public Key Encryption (LPKE) and a CCA-secure PKE.
 - The pivot of our generic construction is Lossy Public Key Encryption (LPKE). We identify properties for basic LPKE so that its integration with hashing function makes UC security possible for PAKE in ROM.

- The instantiations of the basic LPKE and CCA-secure PKE yield two UC-secure PAKE schemes in ROM, one is based on the LWE assumption and the other based on the GA-DDH assumption, which leads to the *first* PAKE scheme with UC security from isogenies.
2. We upgrade UC security from ROM to QROM for our generic PAKE construction, by replacing the “basic LPKE + Hash” with an extractable LPKE.
 - We define extractable LPKE by equipping it with an extracting algorithm and identify its properties to make UC security possible for PAKE in QROM. We also present a generic construction for extractable LPKE.
 - The instantiations of extractable LPKE lead to the *first* two UC-secure PAKE schemes in QROM, one is based on the LWE assumption and the other based on the GA-DDH assumption in the isogeny setting.

Technique Overview. The design principle of our PAKE is to make the underlying PKE associating with passwords. To this end, we introduce a labeled lossy public key encryption LPKE, where the passwords pw are used to derive labels $b := H(pw)$ for LPKE via hash function. The labels are used to generate public keys pk and secret keys sk , i.e., $(pk, sk) \leftarrow \text{LPKE.LKeyGen}(b = H(pw))$, and the encryptions also involve labels, i.e., $c \leftarrow \text{LPKE.LEnc}(pk, b = H(pw), m)$. Moreover, LPKE has an algorithm named $\text{IsLossy}(td, pk, b)$, which can use the trapdoor td to tell whether b is a lossy label for pk . If the output is 0, then b must be a normal label for pk , and hence LPKE works in normal mode, which has correctness and CPA security. Otherwise b must be a lossy label, and hence LPKE works in lossy mode. When applying FO-transformation to LPKE, it has CCA security in normal mode but has pseudo-random ciphertexts in lossy mode.

Together with another CCA-secure PKE scheme PKE, we can design a 3-round PAKE scheme which is high-level shown in Fig. 2.

If client C and server S share the same password pw , then the label used to generate pk/sk and that used in the encryption are consistent. Accordingly, LPKE works in *normal* mode and the successful decryption of c guarantees C and S share $m = r|\sigma|k$. The ciphertext $C = \text{PKE.CEnc}(cpk, pw|pk|c; r)$ can be considered as a proof of server S 's knowledge of pw and the authentication of transcript $pk|c$, so it plays the role of authenticating S . Moreover, client C 's knowledge of σ from the decryption of c proves that C shares the same password with S , and hence the third message σ is able to authenticate C .

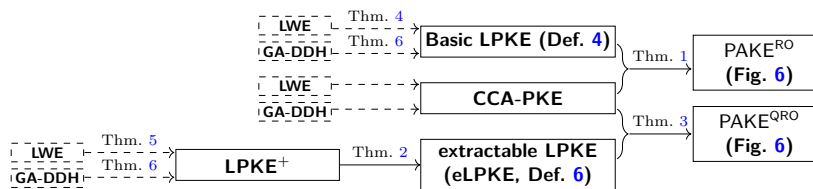


Fig. 1: Schematic overview of our PAKE constructions, where solid arrows “ \rightarrow ” indicate generic constructions and dashed arrows “ \dashrightarrow ” indicate instantiations.

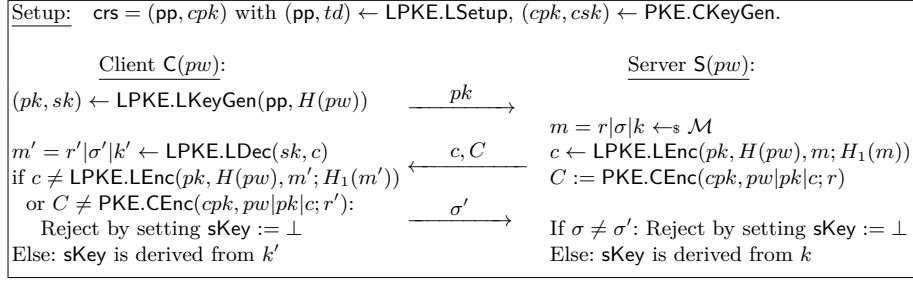


Fig. 2: Our PAKE from LPKE and PKE

UC SECURITY IN ROM. To prove the UC security for our PAKE, we have to construct a simulator Sim , which has no knowledge of password pw but can simulate all the interaction transcripts for both passive attacks and active attacks. The simulations must be indistinguishable to those in the real experiment for an environment, which uses pw to control \mathcal{C} and \mathcal{S} , and manipulates adversaries \mathcal{A} to interfere the interactions between \mathcal{C} and \mathcal{S} . Now we consider how the simulations are implemented according to the type of attacks, passive or active.

Case I: passive attacks. By requiring *pseudo-randomness of public key* for LPKE, simulator Sim can send a random pk as the simulation for the first-round message. The random pk is independent of the random label $H(pw)$ (due to random oracle), so $H(pw)$ is hardly the correct label generating pk . Then $c = \text{LPKE.LEnc}(pk, H(pw), m; H_1(m))$ works in lossy mode and c is pseudo-random due to *pseudo-randomness of ciphertext under lossy labels of LPKE*. Therefore, the simulation of c in the second-round message can be accomplished with a uniformly chosen one. As a result, $m = r|\sigma|k$ are random and independent of c . Then the *CCA security* of PKE implies $C \leftarrow \text{PKE.CEnc}(cpk, pw|pk|c; r) \approx_c C \leftarrow \text{PKE.CEnc}(cpk, 0; r)$. So ciphertext C in the second-round message can be simulated by $C \leftarrow_s \text{PKE.CEnc}(cpk, 0)$. Furthermore, the third-round message σ' can simply be simulated by setting $\sigma' := \sigma$ to keep consistence to the second-round message.

As for active attacks, a critical problem for Sim is that without any knowledge of pw , how to determine whether \mathcal{A} implements attacks with a correct guess of pw . To solve this problem, we resort to the trapdoor td of LPKE, the algorithm IsLossy , random oracle H , and also the secret key of PKE.

Sim will generate the crs of PAKE and hold the trapdoor td of LPKE and the secret key csk of PKE to extract the possible passwords used in active attacks.

- **Password extraction from \tilde{pk} .** For a first-round message \tilde{pk} from \mathcal{A} 's active attack, if \tilde{pk} is associated with some password pw' , then $H(pw')$ must have been queried by \mathcal{A} . Sim just searches all the hash queries and replies under random oracle H to find $(pw', b = H(pw'))$ s.t. $\text{IsLossy}(td, \tilde{pk}, H(pw')) = 0$. If $\text{IsLossy}(td, \tilde{pk}, H(pw')) = 0$ then $H(pw')$ must be a normal label used

to generate \tilde{pk} . Meanwhile, random oracle H makes sure the uniqueness of such password. In this way, the password pw' used by \mathcal{A} for generating \tilde{pk} , if any, is successfully extracted by Sim .

- **Password extraction from (\tilde{c}, \tilde{C}) .** For a second-round message (\tilde{c}, \tilde{C}) from \mathcal{A} 's active attack, Sim will use the secret key csk (w.r.t. the system public key cpk) to decrypt \tilde{C} to extract the encrypted password pw' . The correctness of PKE makes sure that the password pw' used by \mathcal{A} for generating (\tilde{c}, \tilde{C}) , if any, is successfully extracted by Sim .

Case II: active attacks. With help of the above extractions of password pw' , Sim can submit $\text{Testpw}(pw')$ to the ideal functionality of \mathcal{F}_{pake} to decide whether pw' is the correct one. If pw' is correct, then Sim uses the correct pw' to simulate the protocol interactions, just like the real case. If the extracted password pw' is not correct, the simulations are implemented as follows.

- For a first-round message \tilde{pk} from \mathcal{A} 's active attack, if the extracted pw' is not correct, then the random label $H(pw)$ must be lossy for \tilde{pk} . The simulator can simulate the second-round message (c, C) just like the case of passive attacks.
- For a second-round message (\tilde{c}, \tilde{C}) from \mathcal{A} 's active attack,
 - if the extracted pw' is not correct, the simulator just aborts the protocol. In the real case, the correctness of PKE also leads to abort due to an incorrect password.
 - if the extracted pw' is correct, recall that in this case, \tilde{pk} is a random one (simulated in the passive attack), the simulator has no knowledge of sk and cannot decrypt \tilde{c} with LPKE.LDec . To solve this problem, we resort to random oracle $H_1(m')$ to extract the message $m' = r'|\sigma'|k'$ such that $\tilde{c} = \text{LPKE.LEnc}(pk, H(pw), m'; H_1(m'))$, if it exists.

UC SECURITY IN QROM. Note that random oracle H plays an important role in simulator's password extracting from \tilde{pk} . However, H fails in QROM, because the simulating technique of storing and searching queries & hash values in a list does not apply to QRO due to the no-cloning principle.

To achieve UC security in the QROM, our solution is discarding hash function H and directly using passwords pw as labels in the generation of \tilde{pk} and the encryption of m . More importantly, we augment an $\text{Extract}(td, \tilde{pk})$ algorithm to LPKE which can use trapdoor td to extract password pw' from \tilde{pk} directly, free of hash. In this way, LPKE is upgraded to an extractable one, namely eLPKE. The generic construction of PAKE remains the same except that we replace LPKE with eLPKE. The simulator can use $\text{Extract}(td, \tilde{pk})$ to extract pw' and the rest of simulations are almost the same as that in ROM.

We can construct extractable LPKE (eLPKE) from LPKE as follows. Parse the label bit-wise $pw = (pw_1, \dots, pw_\lambda)$. Each bit $pw_i \in \{0, 1\}$ corresponds to two public random tags v_i^0, v_i^1 . Then λ invocations of $(pk_i, sk_i) \leftarrow \text{LPKE.LKeyGen}(v_i^{pw_i})$ result in public key $\mathbf{pk} := (pk_1, \dots, pk_\lambda)$ and secret key $\mathbf{sk} := (sk_1, \dots, sk_\lambda)$. In this way, the password pw can be extracted bit-wisely via testing $\text{IsLossy}(td, pk_i, v_i^0)$

$\stackrel{?}{=} 0$ and $\text{lsLossy}(td, pk_i, v_i^1) \stackrel{?}{=} 0$. For encryption, the plaintext m is divided into λ shares m_1, \dots, m_λ such that $m = m_1 \oplus \dots \oplus m_\lambda$ via (λ, λ) -secret sharing. The ciphertext contains sub-ciphertexts $\{c_i := \text{LPKE.LEnc}(pk_i, m_i)\}_{i \in [\lambda]}$.

There remains a subtlety to be addressed. To justify that the real pk can be replaced by a random one, the security reduction must ensure that the secret key sk is not needed for the decryption of c to get m . To solve this problem, the UC security in ROM also uses the hash list of H_1 , and search the list to find the right $(m, H_1(m))$ by testing the re-encryption relation $c = \text{LPKE.LEnc}(pk, pw, m; H_1(m))$. In QROM, this trick does not apply either. To solve the problem in QROM, we resort to the *on-line extraction* technique [21] so that the simulator can extract m while simulating H_1 in an indistinguishable way. Moreover, in case of active attack (\tilde{c}, \tilde{C}) , after the simulator extracts a correct password pw , it also uses the on-line extraction technique to extract m' .

By instantiating LPKE and eLPKE in our generic construction, we obtain PAKE schemes $\text{PAKE}_{\text{lwe}}^{\text{RO}}$, $\text{PAKE}_{\text{ga}}^{\text{RO}}$, $\text{PAKE}_{\text{lwe}}^{\text{QRO}}$, $\text{PAKE}_{\text{ga}}^{\text{QRO}}$ from LWE in lattice, from GA-DDH in the isogeny setting, in ROM, and in QROM respectively.

The schematic overview of our PAKE constructions is given in Fig. 1.

Comparison. In Table 1, we compare our PAKE schemes with the available schemes based on post-quantum assumptions. Up to now, there are only two PAKE schemes [33,8] with UC security from post-quantum assumptions, both of which are based on ICM. Note that ICM is equivalent to ROM [19]. Therefore, our work admits the *first* UC-secure PAKE schemes from isogenies, both in ROM and QROM, and the *first* UC-secure PAKE schemes from LWE in QROM. The time complexity and communication complexity of our PAKE schemes are given in the Table 2. Our LWE-based PAKE in QROM is not as practical as the MLWE-based PAKE [33,8] in RO/IC, but our PAKE allows quantum access to random oracles from adversaries and avoids the use of IC. Our GA-based PAKE schemes are practical. In particular, the efficiency of our GA-based UC-secure PAKE in ROM is comparable to that of the GA-based IND-secure PAKE in ROM [3].

A recent progress on PAKE [28] shows how to compile PAKE to strong asymmetric PAKE with CSIDH assumption. Applying their compiler, our PAKE from GA-DDH can be upgraded to a strong asymmetric PAKE from isogenies.

2 Preliminaries

If x is defined by y or the value of y is assigned to x , we write $x := y$. For $\mu \in \mathbb{N}$, define $[\mu] := \{1, 2, \dots, \mu\}$. Denote by $x \leftarrow_s \mathcal{X}$ the procedure of sampling x from set \mathcal{X} uniformly at random. We also use “\$” to denote a random variable uniformly chosen from an implicitly known set. Let $|\mathcal{X}|$ denote the number of elements in \mathcal{X} . All our algorithms are probabilistic unless stated otherwise. We use $y \leftarrow \mathcal{A}(x)$ to define the random variable y obtained by executing algorithm \mathcal{A} on input x . We also use $y \leftarrow \mathcal{A}(x; r)$ to make explicit the random coins r used in the probabilistic computation. The notation \approx_s represents statistical indistinguishability, while \approx_c denotes computational indistinguishability. We use

Scheme	Rounds	Security	Model	Assumption	Mutual Authentication	CRS
[11,26]	2	IND	Standard	LWE	No	Yes
[37]	2	IND	RO	LWE	No	Yes
[25]	3	IND	Standard	LWE	Yes	Yes
[20]	3	IND	RO	RLWE	Yes	No
[3]	2	IND	RO	SqInv-GA-StCDH	No	Yes
[16] + [5]	3	IND	Standard	GA-DDH	Yes	Yes
[16] + [30]	3	IND	Standard	LWE	Yes	Yes
[33,8]	2	UC	Ideal Cipher	MLWE	No	No
PAKE _{lwe} ^{RO}	3	UC	RO	LWE	Yes	Yes
PAKE _{ga} ^{RO}	3	UC	RO	GA-DDH	Yes	Yes
PAKE _{lwe} ^{QRO}	3	UC	QRO	LWE	Yes	Yes
PAKE _{ga} ^{QRO}	3	UC	QRO	GA-DDH	Yes	Yes

Table 1: Comparison of PAKE schemes from post-quantum assumptions. In [11,3], a simultaneous flow of two round-messages is counted as 1-round while we count it as 2. “Mutual Authentication” indicates whether the scheme supports mutual explicit authentication. “CRS” means whether the scheme requires a common reference string.

Scheme	Security	Model	Time Complexity	Communication Complexity
[33,8]	UC	Ideal Cipher	$O(\lambda^2)$	$O(\lambda \log \lambda)$
[3]	IND	RO	$O(\lambda) \times \text{GA}$	$O(\lambda^2)$
PAKE _{lwe} ^{RO}	UC	RO	$O(\lambda^4 \log^2 \lambda)$	$O(\lambda^2 \log \lambda)$
PAKE _{ga} ^{RO}	UC	RO	$O(\lambda \log \lambda) \times \text{GA}$	$O(\lambda^2)$
PAKE _{lwe} ^{QRO}	UC	QRO	$O(\lambda^8)$	$O(\lambda^5)$
PAKE _{ga} ^{QRO}	UC	QRO	$O(\lambda \log^2 \lambda) \times \text{GA}$	$O(\lambda^3)$

Table 2: Comparison of PAKE schemes from post-quantum assumptions in terms of time complexity and communication complexity, where GA denotes the time complexity for a single group action operation.

bold lower-case letters to denote column vectors. For a vector \mathbf{v} , we let $\|\mathbf{v}\|$ (resp., $\|\mathbf{v}\|_\infty$) denote its ℓ_2 (resp., ℓ_∞ infinity) norm.

2.1 Hardness Assumptions

Lattice Backgrounds. A q -ary lattice defined with $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is $\Lambda(\mathbf{A}) := \{\mathbf{A}^T \mathbf{s} \mid \mathbf{s} \in \mathbb{Z}_q^n\} + q\mathbb{Z}^m$. The Gaussian function on \mathbb{R}^n centered at \mathbf{c} with parameter s is defined by $\rho_{s,\mathbf{c}}(\mathbf{x}) := e^{-\pi\|\mathbf{x}-\mathbf{c}\|^2/s^2}$. The discrete Gaussian distribution $D_{\Lambda,s,\mathbf{c}}$ over an n -dimensional lattice $\Lambda \subseteq \mathbb{R}^n$ is defined by $D_{\Lambda,s,\mathbf{c}}(\mathbf{x}) := \rho_{s,\mathbf{c}}(\mathbf{x})/\rho_{s,\mathbf{c}}(\Lambda)$ for any lattice vector $\mathbf{x} \in \Lambda$, where $\rho_{s,\mathbf{c}}(\Lambda) := \sum_{\mathbf{z} \in \Lambda} \rho_{s,\mathbf{c}}(\mathbf{z})$. The centered vector \mathbf{c} is often omitted when $\mathbf{c} = \mathbf{0}$. More specifically, we use the notion $D_{\mathbb{Z}^m,r}$ to represent discrete Gaussian distribution over lattice $\Lambda := \mathbb{Z}^m$ centered at $\mathbf{c} := \mathbf{0}$ with parameter r .

Definition 1 (LWE Assumption [32]). Let $n, m, q \in \mathbb{N}$, and χ be a distribution over \mathbb{Z}_q . The $\text{LWE}_{n,q,m,\chi}$ assumption states that the following distributions are computationally indistinguishable: $(\mathbf{A}, \mathbf{A}^T \mathbf{s} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{u})$, where $\mathbf{A} \leftarrow_s \mathbb{Z}_q^{n \times m}$, $\mathbf{e} \leftarrow \chi^m$, $\mathbf{s} \leftarrow_s \mathbb{Z}_q^n$ and $\mathbf{u} \leftarrow_s \mathbb{Z}_q^m$.

Lemma 1 ([4,29]). *There exists a PPT algorithm TrapGen that takes as input positive integers n, q ($q \geq 2$) and a sufficiently large $m = O(n \log q)$, outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a trapdoor matrix $\mathbf{T}_\mathbf{A} \in \mathbb{Z}_q^{m \times m}$ such that \mathbf{A} is statistically close to the uniform distribution, $\mathbf{A} \cdot \mathbf{T}_\mathbf{A} = \mathbf{0}$, and $\|\tilde{\mathbf{T}}_\mathbf{A}\| \leq O(\sqrt{n \log q})$, where $\tilde{\mathbf{T}}_\mathbf{A}$ denotes the Gram-Schmidt orthogonalization of $\mathbf{T}_\mathbf{A}$.*

Lemma 2 ([29, Theorem 5.4]). *There exists a deterministic polynomial-time algorithm Invert that takes as inputs the trapdoor information $\mathbf{T}_\mathbf{A}$ and a vector $\mathbf{v} := \mathbf{A}^T \cdot \mathbf{s} + \mathbf{e}$ with $\mathbf{s} \in \mathbb{Z}_q^n$ and $\|\mathbf{e}\| \leq q/(10 \cdot \sqrt{m})$, and outputs \mathbf{s} and \mathbf{e} .*

Lemma 3 ([22]). *Let n and q be positive integers with q prime, and let $m \geq 2n \log q$. Then for all but a $2q^{-n}$ fraction of all $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and for any $r \geq \omega(\sqrt{\log m})$, the distribution of the syndrome $\mathbf{u} = \mathbf{A}\mathbf{e} \pmod q$ is statistically close to uniform over \mathbb{Z}_q^n , where $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, r}$.*

Cryptographic Group Actions. We will focus on group actions where \mathbb{G} is abelian and the action is regular (See Appendix A for their definitions). We recall the notion of restricted effective group actions (REGA) as follows.

Definition 2 (Restricted Effective Group Action [5]). *A group action $(\mathbb{G}, \mathcal{X}, \star)$ is a restricted effective group action (REGA) if properties 1-5 are satisfied.*

1. *The group \mathbb{G} is generated by a set $\{g_1, \dots, g_n\}$.*
2. *The group \mathbb{G} is finite and $n = \text{poly}(\log |\mathbb{G}|)$.*
3. *The set \mathcal{X} is finite and there exist PPT algorithms for membership testing and for computing unique representation of set element.*
4. *There exists a distinguished element $x_0 \in \mathcal{X}$, called the origin, such that its representation is known.*
5. *There exists an efficient algorithm that given g_i in the generating set and any $x \in \mathcal{X}$, outputs $g_i \star x$ and $g_i^{-1} \star x$ where $i \in [n]$.*

With a REGA, we can use the generating set to approximate the random sampling process of $g \leftarrow_{\mathcal{S}} \mathbb{G}$. The regularity of the $(\mathbb{G}, \mathcal{X}, \star)$ enables an efficient algorithm to sample $x \leftarrow_{\mathcal{S}} \mathcal{X}$ uniformly.

There is a natural generalization of the DDH assumption in REGA settings.

Definition 3 (GA-DDH Assumptions). *Given a restricted effective group action $(\mathbb{G}, \mathcal{X}, \star)$, the Group Action DDH (GA-DDH) assumption states that the following distributions are computationally indistinguishable:*

$$\{x \leftarrow_{\mathcal{S}} \mathcal{X}; s, t \leftarrow_{\mathcal{S}} \mathbb{G} : (x, s \star x, t \star x, (s \cdot t) \star x)\} \approx_c \{x \leftarrow_{\mathcal{S}} \mathcal{X}; s, t, z \leftarrow_{\mathcal{S}} \mathbb{G} : (x, s \star x, t \star x, z \star x)\}.$$

We can instantiate REGA with isogeny-based group actions, like CSIDH. The GA-DDH assumption is believed to hold for CSIDH [18].

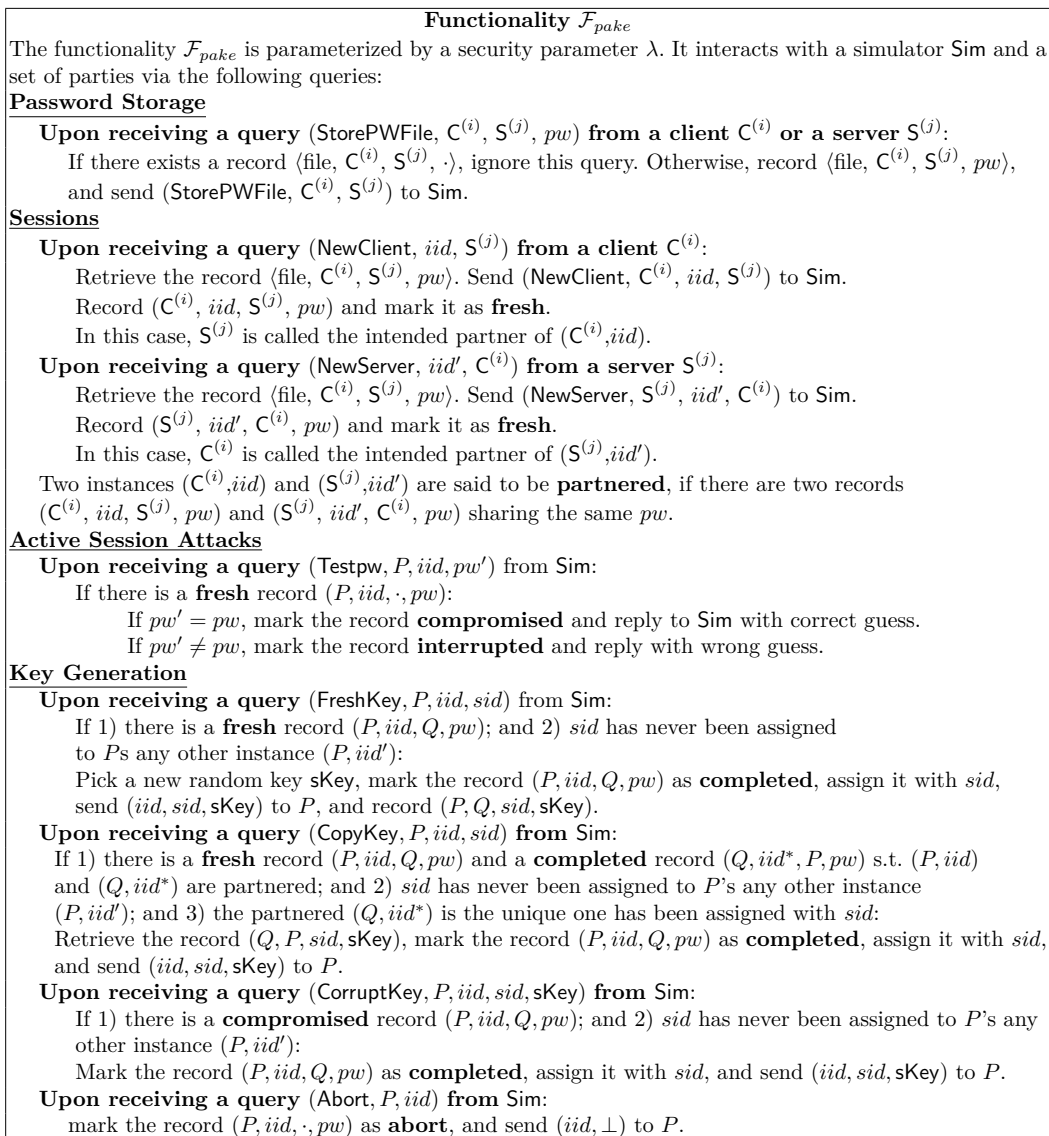
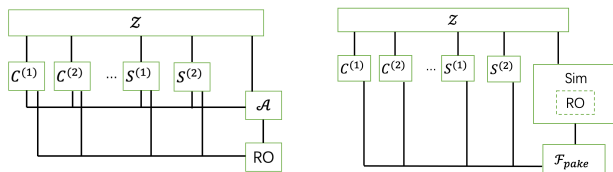
Fig. 3: The ideal functionality \mathcal{F}_{pake} for PAKE.

Fig. 4: The real world execution (left) and the ideal world execution (right).

2.2 UC Framework for PAKE

We present a concise overview of the UC framework for PAKE. Fig. 4 shows the picture of the “real world” execution of a protocol Π and the “ideal world” execution with a simulator Sim . The environment \mathcal{Z} can be considered as higher-level protocols utilizing Π as a sub-protocol (\mathcal{Z} also includes the adversary that is attacking those higher-level protocols). The adversary \mathcal{A} essentially models a completely insecure network and it communicates continuously with \mathcal{Z} . The client/server instances send and receive messages via \mathcal{A} , and \mathcal{A} can do dropping, injecting, and modifying protocol messages at will.

In the real world, the client/server instances are executed as described in the protocol Π . They receive their inputs (passwords in PAKE) from \mathcal{Z} and send their outputs (session key in PAKE) to \mathcal{Z} . Hash function H is modeled as a random oracle. Clients, servers and adversary may directly (quantum) access the random oracle H . However, the environment \mathcal{Z} can only access H indirectly via \mathcal{A} .

In the ideal world, clients/servers are “dummy” parties that pass their passwords directly from \mathcal{Z} to an ideal functionality \mathcal{F}_{pake} , and their outputs directly from \mathcal{F}_{pake} to \mathcal{Z} . We mainly follow the definition of \mathcal{F}_{pake} by Shoup [34], which is a modified version by Canetti et al. [17]. Simulator Sim can communicate with \mathcal{Z} , just as \mathcal{A} did in the real world. Besides, Sim can also interact with \mathcal{F}_{pake} . Sim has two main tasks. Firstly, it must simulate the network transcripts in a way that is indistinguishable from those generated in the real world. Secondly, Sim must provide appropriate inputs to \mathcal{F}_{pake} to obtain outputs (session key) that are indistinguishable from the outputs produced by the client/server instances in the real world.

The ideal functionality \mathcal{F}_{pake} is shown in Fig. 3. We stress that our ideal functionality \mathcal{F}_{pake} supports *mutual authentication*, which indicates mutual explicit authentication. It is mainly captured by the first item and the last item of “Key Generation” in our ideal functionality (Fig. 3), which guarantee that if an adversary makes an unsuccessful password guess on a protocol instance, then when that instance terminates, the corresponding party will receive an abort message. Accordingly, this requires that each party can identify active attacks explicitly in the real-world protocol.

In a nutshell, we say protocol Π securely emulates ideal functionality \mathcal{F}_{pake} if for any efficient adversary \mathcal{A} , there exists an efficient simulator Sim such that no efficient environment \mathcal{Z} can effectively distinguish between the actual execution in the real world and the hypothetical execution in the ideal world.

2.3 ROM vs. QROM

In the Random Oracle Model (ROM), a cryptographic hash function $H : \mathcal{X} \rightarrow \mathcal{Y}$ is idealized as a truly random function $\text{RF} : \mathcal{X} \rightarrow \mathcal{Y}$. And any adversary needs to query H on inputs $x \in \mathcal{X}$ to learn the hash values $H(x)$.

In the quantum world, a quantum algorithm \mathcal{A} can perform superposition queries to the random oracle H , and then oracle H behaves as a unitary operation

$|x\rangle|y\rangle \mapsto |x\rangle|y \oplus H(x)\rangle$. In this case, H becomes a quantum random oracle (QRO). The QRO model supports classical queries x on H , and this can be formalized as setting query register and output register to be $|x\rangle|0\rangle$, and measuring the output register after the unitary operation $|x\rangle|0\rangle \mapsto |x\rangle|0 \oplus H(x)\rangle$.

3 PAKE from Basic LPKE in ROM

In this section we present the definition of basic LPKE and show how to apply the FO-transformation to LPKE to obtain a CCA-secure KEM. Then we show the generic construction of PAKE from basic LPKE and its UC security in ROM.

3.1 Basic Lossy Public Key Encryption (LPKE)

LPKE works in two modes. If label b is a normal one, then LPKE works in normal mode and has correctness and CPA security. If label b is a lossy one, then LPKE works in lossy mode, and the ciphertexts are random. With the trapdoor, algorithm `IsLossy` can decide whether label b is lossy or normal for pk .

Definition 4 (Basic LPKE). *A basic Lossy Public Key Encryption scheme $\text{LPKE} = (\text{LSetup}, \text{LKeyGen}, \text{LEnc}, \text{LDec}, \text{IsLossy})$ consists of five probabilistic algorithms.*

- $\text{LSetup}(1^\lambda)$: *The setup algorithm takes as input the security parameter 1^λ , and outputs a public parameter pp and a trapdoor td . The parameter pp specifies a public key space \mathcal{PK} , a secret key space \mathcal{SK} , a label space \mathcal{T} , a message space \mathcal{M} and a ciphertext space \mathcal{CT} . All the remaining algorithms take pp as input, and we omit it for simplicity.*
- $\text{LKeyGen}(b)$: *The key generation algorithm takes as input a label b and outputs a key pair (pk, sk) .*
- $\text{LEnc}(pk, b, m)$: *The encryption algorithm takes as input a public key pk , a label b and a message m , and outputs a ciphertext c .*
- $\text{LDec}(sk, c)$: *The decryption algorithm takes as input a secret key sk and a ciphertext c , and outputs a message m .*
- $\text{IsLossy}(td, pk, b)$: *The algorithm takes as input a trapdoor td , a public key pk and a label b , and outputs a bit 0 or 1.*

For any $pk \in \mathcal{PK}$ and $b \in \mathcal{T}$, if $\text{IsLossy}(td, pk, b) = 1$, then label b is called a *lossy label* of public key pk . Otherwise, label b is called a *normal label* of pk .

Correctness of Basic LPKE. For all $b \in \mathcal{T}$ and $m \in \mathcal{M}$, it holds that

$$\Pr [\text{LDec}(sk, \text{LEnc}(pk, b, m)) \neq m] \leq \text{negl}(\lambda), \Pr [\text{IsLossy}(td, pk, b) = 1] \leq \text{negl}(\lambda),$$

where $(\text{pp}, td) \leftarrow \text{LSetup}(1^\lambda)$ and $(pk, sk) \leftarrow \text{LKeyGen}(b)$.

A basic LPKE scheme LPKE should satisfy the following properties.

① **Pseudorandomness of Public Key.** For all $b \in \mathcal{T}$, it holds that

$$\left\{ \begin{array}{l} (\text{pp}, td) \leftarrow \text{LSetup} \\ (pk, sk) \leftarrow \text{LKeyGen}(b) \end{array} : (\text{pp}, pk) \right\} \approx_c \left\{ \begin{array}{l} (\text{pp}, td) \leftarrow \text{LSetup} \\ pk \leftarrow_s \mathcal{PK} \end{array} : (\text{pp}, pk) \right\}.$$

② **Random Ciphertexts under Lossy Labels.** For all admissible adversary \mathcal{A} which outputs (pk, b) such that $\text{IsLossy}(td, pk, b) = 1$, it holds that $(\text{pp}, pk, b, m, c) \approx_s (\text{pp}, pk, b, m, c')$, where $(\text{pp}, td) \leftarrow \text{LSetup}$, $(pk, b) \leftarrow \mathcal{A}(\text{pp})$ s.t. $\text{IsLossy}(td, pk, b) = 1$, $m \leftarrow_s \mathcal{M}$, $c \leftarrow \text{LEnc}(pk, b, m)$ and $c' \leftarrow_s \mathcal{CT}$.

③ **Uniqueness of Normal Labels among Polynomial-Size Set:** For any $Q := \text{poly}(\lambda)$, it holds that

$$\Pr \left[\begin{array}{l} (\text{pp}, td) \leftarrow \text{LSetup} \\ b_1, \dots, b_Q \leftarrow_s \mathcal{T} \end{array} : \begin{array}{l} \exists pk \in \mathcal{PK}, i \neq j \text{ with } i, j \in [Q] \\ \text{IsLossy}(td, pk, b_i) = 0 \wedge \text{IsLossy}(td, pk, b_j) = 0 \end{array} \right] \leq \text{negl}(\lambda).$$

④ **Lossiness of Random Labels.** For all adversary \mathcal{A} , it holds that

$$\Pr \left[(\text{pp}, td) \leftarrow \text{LSetup}; pk \leftarrow \mathcal{A}(\text{pp}); b \leftarrow_s \mathcal{T} : \text{IsLossy}(td, pk, b) = 0 \right] \leq \text{negl}(\lambda).$$

⑤ **Ciphertext Unpredictability under Normal Labels:** For all but a negligible fraction of pp from $(\text{pp}, td) \leftarrow \text{LSetup}(1^\lambda)$, for all $b \in \mathcal{T}$, all $(pk, sk) \leftarrow \text{LKeyGen}(b)$, all messages $m \in \mathcal{M}$ and all ciphertexts $c \in \mathcal{CT}$, it holds that $\Pr [r \leftarrow_s \mathcal{R} : \text{LEnc}(pk, b, m; r) = c] \leq \text{negl}(\lambda)$.

⑥ **CPA Security under Normal Labels:** For all PPT \mathcal{A} , we have $(\text{pp}, pk, b, c_0) \approx_c (\text{pp}, pk, b, c_1)$, where $(\text{pp}, td) \leftarrow \text{LSetup}$, $(b, st) \leftarrow \mathcal{A}(\text{pp})$, $(pk, sk) \leftarrow \text{LKeyGen}(b)$, $(m_0, m_1) \leftarrow \mathcal{A}(st, pk)$, $c_0 \leftarrow \text{LEnc}(pk, b, m_0)$, $c_1 \leftarrow \text{LEnc}(pk, b, m_1)$.

Remark 1. The concept of R -lossy PKE (R -LPKE) was introduced by Boyle et al. [15]. Our basic LPKE can be considered as a special R -LPKE with relation R simply defined as $R(K, t) := (K \stackrel{?}{=} t)$, but our basic LPKE is augmented with IsLossy algorithm and equipped with a different set of properties.

By leveraging the FO-transformation, LPKE under normal label b can be transformed to a labeled KEM scheme, whose encapsulation algorithm is implemented as $c \leftarrow \text{LEnc}(pk, b, m; H_1(m))$ with $m \leftarrow_s \mathcal{M}$ and $K := H_2(m)$. The formal description of the KEM construction is shown in Appendix B. Consequently, the CPA-security of LPKE is upgraded to CCA-security of the KEM, according to [21]. This is shown in Lemma 4. Meanwhile, the property of *random ciphertexts under lossy labels* for LPKE is inherited by that of KEM but degrades to a pseudo-random one, as shown in Lemma 5 with proof in Appendix C.1.

Lemma 4 (CCA Security of KEM [21]). *Suppose that LPKE = (LSetup, LKeyGen, IsLossy, LEnc, LDec) is a basic lossy public key encryption scheme and H_1 and H_2 are two (quantum-accessible) random oracles. For any PPT adversary \mathcal{A} against CCA security of the KEM scheme, it holds that*

$$\text{Adv}_{\text{KEM}}^{\text{CCA-FO}}(\mathcal{A}) := \left| \Pr \left[\text{Exp}_{\text{KEM}}^{\text{CCA-FO-0}} \Rightarrow 1 \right] - \Pr \left[\text{Exp}_{\text{KEM}}^{\text{CCA-FO-1}} \Rightarrow 1 \right] \right| \leq \text{negl}(\lambda),$$

where $\text{Exp}_{\text{KEM}}^{\text{CCA-FO-}\beta}$ is defined in Fig 5.

$\text{Exp}_{\text{KEM}}^{\text{CCA-FO-}\beta} : \quad // \beta \in \{0, 1\}$ $(\text{pp}, td) \leftarrow \text{LSetup}; (b, \text{st}) \leftarrow \mathcal{A}(\text{pp})$ $(pk, sk) \leftarrow \text{LKeyGen}(b)$ $m \leftarrow_{\$} \mathcal{M}, c^* \leftarrow \text{LEnc}(pk, b, m; H_1(m))$ $K_0^* := H_2(m), K_1^* \leftarrow_{\$} \mathcal{K}_{\text{kem}}$ $\beta' \leftarrow \mathcal{A}^{\text{O}_{\text{Dec}}(\cdot)}(\text{st}, pk, c^*, K_0^*)$ $\text{Return } \beta'$	$\text{O}_{\text{Dec}}(c):$ $\text{If } c = c^*: \text{Return } \perp$ $K \leftarrow \text{LDec}(sk, c)$ $\text{Return } K$
--	---

Fig. 5: CCA-security of KEM from LPKE via FO

Lemma 5 (Ciphertext Pseudo-Randomness under Lossy Labels). *Suppose that $\text{LPKE} = (\text{LSetup}, \text{LKeyGen}, \text{lsLossy}, \text{LEnc}, \text{LDec})$ is a basic lossy public key encryption scheme and H_1 and H_2 are two (quantum-accessible) random oracles. For all admissible adversary \mathcal{A} which outputs (pk, b) such that $\text{lsLossy}(td, pk, b) = 1$, it holds that $(\text{pp}, pk, b, c, K) \approx_c (\text{pp}, pk, b, c' \leftarrow_{\$} \mathcal{CT}, K' \leftarrow_{\$} \mathcal{K})$, where $(\text{pp}, td) \leftarrow \text{LSetup}, (pk, b) \leftarrow \mathcal{A}(\text{pp})$ s.t. $\text{lsLossy}(td, pk, b) = 1, m \leftarrow_{\$} \mathcal{M}, c \leftarrow \text{LEnc}(pk, b, m; H_1(m))$, and $K := H_2(m)$.*

3.2 Construction of PAKE from Basic LPKE in ROM

We propose a generic construction of 3-round PAKE scheme PAKE^{RO} from LPKE and PKE. The underlying building blocks are as follows.

- a basic lossy public key encryption scheme $\text{LPKE} = (\text{LSetup}, \text{LKeyGen}, \text{LEnc}, \text{LDec}, \text{lsLossy})$ with message space $\{0, 1\}^\lambda$, label space \mathcal{T} and randomness space \mathcal{R} ;
- a CCA-secure public key encryption scheme $\text{PKE} = (\text{CKeyGen}, \text{CEnc}, \text{CDec})$;
- four hash functions: $H : \mathcal{PW} \rightarrow \mathcal{T}, H_1 : \{0, 1\}^\lambda \rightarrow \mathcal{R}, H_2 : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{3\lambda}, H_3 : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$.

Our generic construction of PAKE from LPKE and PKE is given in Fig. 6.

The UC security for PAKE^{RO} constructed from LPKE in ROM is shown in Theorem 1. To facilitate the proof, we define the concept of *linked to* like [34].

Definition 5 (Linked To). *In a protocol execution, a client (resp. server) instance is “linked to” a server (resp. client) instance at a specific time being if the transcript of the client (resp. server) is consistent to that of the server (resp. client), i.e., the messages received and sent by a party are exactly those messages sent and received by another party.*

Theorem 1. *If LPKE is a basic LPKE scheme, PKE is a CCA-secure PKE scheme, and H, H_1, H_2, H_3 work as random oracles, then the PAKE scheme PAKE^{RO} in Fig. 6 securely emulates $\mathcal{F}_{\text{pake}}$, hence achieving UC security in ROM.*

Proof. The main objective of the proof is to construct a PPT simulator Sim . Sim is designed to have access to the ideal functionality $\mathcal{F}_{\text{pake}}$ and interact with the environment \mathcal{Z} , thereby emulating the real-world PAKE protocol interactions involving the adversary \mathcal{A} , the parties, and the environment \mathcal{Z} . It is important to note that Sim *does not possess any password*.

The complete description of the simulator Sim is provided in Fig. 7.

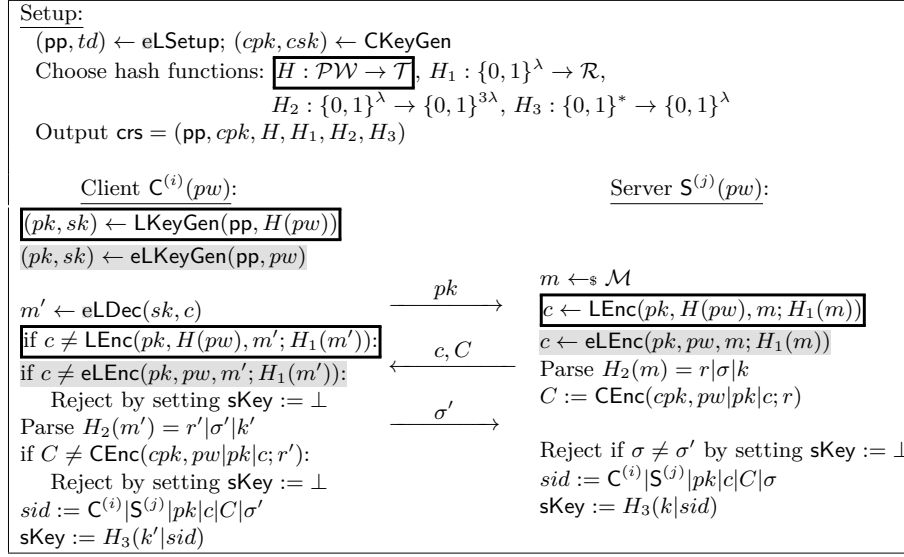


Fig. 6: Construction of PAKE^{RO} (resp. PAKE^{QRO}) from LPKE (resp. eLPKE). `text` only appears in PAKE^{RO} from LPKE and `text` only appears in PAKE^{QRO} from eLPKE .

Let $\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}$ represent the real-world experiment where the environment \mathcal{Z} interacts with the actual parties and adversary \mathcal{A} , while $\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}}$ represents the ideal experiment where \mathcal{Z} interacts with the simulator Sim .

Our goal is to demonstrate that $|\Pr[\mathbf{Real}_{\mathcal{Z}, \mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}} \Rightarrow 1]|$ is negligible by employing a series of games, denoted as Game \mathbf{G}_0 - \mathbf{G}_{11} . In this sequence, \mathbf{G}_0 corresponds to $\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}$, while \mathbf{G}_{11} corresponds to $\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}}$. We aim to show that these adjacent games are indistinguishable from the view of \mathcal{Z} .

Game \mathbf{G}_0 . This is the real experiment $\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}$. In this experiment, \mathcal{Z} initializes a password for each client-server pair, sees the interactions among clients, servers and adversary \mathcal{A} , and also obtains the corresponding session keys of protocol instances. Here \mathcal{A} may implement attacks like view, modify, insert, or drop messages over the network. We have $\Pr[\mathbf{Real}_{\mathcal{Z}, \mathcal{A}} \Rightarrow 1] = \Pr[\mathbf{G}_0 \Rightarrow 1]$.

Game \mathbf{G}_1 (simulations for clients and servers with pw).

In this game, we introduce a simulator Sim who *receives passwords* from \mathcal{Z} . Then it simulates the clients and servers to generate transcripts for instances of the PAKE protocol. With the knowledge of *passwords*, the simulations of the behaviors of all clients and servers are perfect.

Moreover, Sim also simulates random oracles H, H_1, H_2, H_3 by maintaining four separate lists, namely $\mathcal{L}_H, \mathcal{L}_{H_1}, \mathcal{L}_{H_2}, \mathcal{L}_{H_3}$. For example, for a query x on $H(\cdot)$, if $(x, y) \in \mathcal{L}_H$, then Sim will return y as the reply. Otherwise, Sim will choose a random element y , record (x, y) in \mathcal{L}_H , and return y as the reply. By the ideal functionality of random oracles, Sim 's simulations for oracles H, H_1, H_2, H_3 are also perfect. So we have $\Pr[\mathbf{G}_1 \Rightarrow 1] = \Pr[\mathbf{G}_0 \Rightarrow 1]$.

The following games will change the simulations of `Sim` step by step in an indistinguishable way so that `Sim` can arrive at Fig. 7 and accomplish the simulations in $\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}}$ without passwords pw .

Game G_2 (simulation for crs). In G_2 , `Sim` simulates the generation of $\text{crs} = (\text{pp}, \text{cpk}, \dots)$ with $(\text{pp}, \text{td}) \leftarrow \text{LSetup}$ and $(\text{cpk}, \text{csk}) \leftarrow \text{CKeyGen}$, and it also records the trapdoor td of LPKE and the secret key csk of PKE. Clearly, the simulation of crs is perfect, so we have $\Pr[G_2 \Rightarrow 1] = \Pr[G_1 \Rightarrow 1]$.

Game G_3 (simulation of $r|\sigma|k$ for server instances and simulation of the third-round message σ' for client instances in case of passive attacks). In G_3 , simulator `Sim` is the same as in G_2 , except for the simulation of generating $r|\sigma|k$ for server instances and the corresponding simulation for client instances in case of passive attacks.

- For a server instance $(S^{(j)}, iid')$ that is linked to $(C^{(i)}, iid)$ when receiving a first-round message pk , simulator `Sim` will randomly sample $r|\sigma|k \leftarrow \{0, 1\}^{3\lambda}$, rather than computing $r|\sigma|k := H_2(m)$ as did in G_2 . Note that pk must have been generated for the client instance $(C^{(i)}, iid)$ by `Sim`.
- For a client instance $(C^{(i)}, iid)$ that is linked to server instance $(S^{(j)}, iid')$ when receiving a second-round message (c, C) , we know that (c, C) and the corresponding $r|\sigma|k$ must have been generated for $(S^{(j)}, iid')$ by `Sim`. In this case, `Sim` directly sets $\text{sKey} := H_3(k|sid)$ and outputs $\sigma' := \sigma$.

According to Lemma 4, $(pk, c = \text{LEnc}(pk, H(pw), m; H_1(m)), r|\sigma|k := H_2(m))$ works as a CCA-secure KEM, where pk is the public key, $H(pw)$ is the normal label of pk , c is the ciphertext, $m \leftarrow_s \mathcal{M}$ and $H_2(m) = r|\sigma|k$ is the encapsulated key. Then by the CCA-security of KEM we have

$$(pk, c \leftarrow \text{LEnc}(pk, m; H_1(m)), r|\sigma|k := H_2(m)) \approx_c (pk, c \leftarrow \text{LEnc}(pk, m; H_1(m)), \$)$$

for PPT adversaries access to decryption oracle. There are at most ℓ sessions, so hybrid arguments across the ℓ sessions yield.

$$|\Pr[G_3 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \leq \ell \cdot \text{Adv}_{\text{KEM}}^{\text{CCA-FO}}(\mathcal{B}_{\text{KEM}}) \leq \text{negl}(\lambda).$$

Game G_4 (simulation of sKey in case of passive attacks). In G_4 , `Sim` is the same as in G_3 , except for the generation of sKey for client instances and its corresponding sever instances in case of passive attacks.

- For a client instance $(C^{(i)}, iid)$ that is linked to server instance $(S^{(j)}, iid')$, when client $C^{(i)}$ receives a second-round message (c, C) , simulator `Sim` changes the simulation of generating session key sKey for $C^{(i)}$. More precisely, `Sim` will not set $\text{sKey} := H_3(k|sid)$ as did in G_3 , but sample $\text{sKey} \leftarrow_s \{0, 1\}^\lambda$ instead. `Sim` stores sKey for $(C^{(i)}, iid)$. Note that (c, C) must have been generated by `Sim` and $r|\sigma|k$ sampled uniformly by `Sim` for server instance $(S^{(j)}, iid')$. For the simulation of $C^{(i)}$ outputting σ' , `Sim` still outputs $\sigma' := \sigma$ by retrieving σ from $r|\sigma|k$, just like G_3 .

- For a server instance $(S^{(j)}, iid')$ that is linked to $(C^{(i)}, iid)$, when $S^{(j)}$ receives a third-round message σ' , Sim will compare σ' with σ in the transcription (pk, c, C, σ) of instance $(S^{(j)}, iid')$. If $\sigma' = \sigma$, Sim will retrieve sKey stored for $(C^{(i)}, iid)$ and set the same session key sKey for $(S^{(j)}, iid')$.

Note that in the above two cases, $(C^{(i)}, iid)$ and $(S^{(j)}, iid')$ share the same $r|\sigma|k$, where $r|\sigma|k$ is uniformly chosen and independent of the view of \mathcal{A} . The uniformity of k makes sure that \mathcal{A} ever queries $H_3(k|sid)$ for some sid with negligible probability. As long as no query on $H_3(k|\cdot)$ is made by \mathcal{A} , the session key sKey := $H_3(k|\cdot)$ is uniform and independent of other variables in G_3 . In G_4 , the session key sKey is sampled in a random and independent way. Consequently, G_4 and G_3 are the same to \mathcal{Z} , except that \mathcal{A} ever queries $H_3(k|\cdot)$ which happens with negligible probability. So we have $|\Pr[G_4 \Rightarrow 1] - \Pr[G_3 \Rightarrow 1]| \leq \text{negl}(\lambda)$.

Game G_5 (simulation for client instances in case of active attacks). In G_5 , simulator Sim is the same as in G_4 , except that Sim will add a rejection rule in the simulations for client instances in case of active attacks. More precisely,

- For a client instance $(C^{(i)}, iid)$ that is not linked to any server instance $(S^{(j)}, iid')$ when receiving the second-round message (\tilde{c}, \tilde{C}) , we know that (\tilde{c}, \tilde{C}) is NOT generated from any server instance, so it must be forged by adversary \mathcal{A} (with active attacks). Let pw be the password shared between $C^{(i)}$ and $S^{(j)}$ and pk be the public key generated by Sim for instance $(C^{(i)}, iid)$.
Rejection rule: Upon receiving the second-round message (\tilde{c}, \tilde{C}) , Sim first extracts password pw' by invoking $pw'|pk'|c' \leftarrow \text{CDec}(csk, \tilde{C})$. If

$$\underline{pw \neq pw' \text{ or } pk'|c' \neq pk|\tilde{c}}, \quad (\star)$$

then Sim rejects (\tilde{c}, \tilde{C}) by setting sKey := \perp .

G_5 and G_4 differ only when a message (\tilde{c}, \tilde{C}) satisfying (\star) leads to rejection in G_5 but not in G_4 . However, if (\tilde{c}, \tilde{C}) satisfies (\star) then either $pw' \neq pw$ or $pk'|c' \neq pk|\tilde{c}$. If such (\tilde{c}, \tilde{C}) does not lead to rejection in G_4 , then we have $\tilde{C} = \text{CEnc}(cpk, pw|pk|\tilde{c}; r)$ and $pw'|pk'|c' = \text{CDec}(csk, \tilde{C})$, but $pw'|pk'|c' \neq pw|pk|\tilde{c}$, which contradicts to the correctness of PKE. Therefore, (\tilde{c}, \tilde{C}) satisfying (\star) must lead to rejection in G_4 except with negligible probability, and hence we have

$$|\Pr[G_5 \Rightarrow 1] - \Pr[G_4 \Rightarrow 1]| \leq \text{negl}(\lambda).$$

Game G_6 (get rid of sk in simulation for client instances in case of active attacks). In G_6 , Sim is the same as in G_5 , except for the generation of m' during Sim's simulations for client instances in case of active attacks.

- For any client instance $(C^{(i)}, iid)$ that is not linked to any server instance $(S^{(j)}, iid')$ when receiving the second-round message (\tilde{c}, \tilde{C}) , we know that (\tilde{c}, \tilde{C}) is NOT generated from any server instance, so it must be forged by adversary \mathcal{A} . When generating m' during the simulation for instance $(C^{(i)}, iid)$, Sim will not use the decryption algorithm to obtain $m' \leftarrow \text{LDec}(sk, \tilde{c})$ as did in G_5 . Instead, it will check whether $\exists(m', r_H) \in \mathcal{L}_{H_1}$ such that

$\tilde{c} = \text{LEnc}(pk, H(pw), m'; r_H)$, where pw is the password of $C^{(i)}$ and pk is the first-round message generated by Sim for $(C^{(i)}, iid)$. If there exists such pair $(m', r_H) \in \mathcal{L}_{H_1}$, then Sim retrieves m' as the decrypted plaintext. Otherwise Sim rejects (\tilde{c}, \tilde{C}) by setting $\text{sKey} := \perp$.

We consider two cases.

Case I. $\exists(m', r_H) \in \mathcal{L}_{H_1}$ s.t. $\tilde{c} = \text{LEnc}(pk, H(pw), m'; r_H)$. By the correctness of LPKE, we have $\text{LDec}(sk, \tilde{c}) = m'$. Therefore, G_6 results in the same m' as that in G_5 , and thus G_6 and G_5 are indistinguishable to \mathcal{Z} in this case.

Case II. $\nexists(m', r_H) \in \mathcal{L}_{H_1}$ s.t. $\tilde{c} = \text{LEnc}(pk, H(pw), m'; r_H)$. This suggests that adversary \mathcal{A} does not ever query $H_1(m')$ and hence $H_1(m')$ is random. Recall that pk was generated from $\text{LKeyGen}(H(pw))$, so $H(pw)$ is the normal label of pk . Then by the property of *ciphertext unpredictability under normal labels*, $\tilde{c} = \text{LEnc}(pk, H(pw), m'; H_1(m'))$ hardly holds. Accordingly, Sim 's simulation of $C^{(i)}$ will reject with $\text{sKey} := \perp$ in G_5 , except with negligible probability. In G_6 , Sim will terminate the simulation by setting $\text{sKey} := \perp$. Obviously, G_6 and G_5 are identical to \mathcal{Z} except with negligible probability in this case.

Therefore, we have $|\Pr[G_6 \Rightarrow 1] - \Pr[G_5 \Rightarrow 1]| \leq \text{negl}(\lambda)$.

We stress that now in G_6 (and hereafter), Sim 's simulation for client instances does not need the secret key sk of LPKE any more, no matter dealing with active attacks or passive attacks. This helps us to proceed to the next game.

Game G_7 (simulation of generating first-round message pk without pw). In G_7 , Sim is the same as in G_6 , except for Sim 's simulation of generating the first-round message pk for client instances.

- For any client instance $(C^{(i)}, iid)$, when generating the first-round message pk , Sim randomly samples $pk \leftarrow_s \mathcal{PK}$ in G_7 , rather than invoking $(pk, sk) \leftarrow \text{LKeyGen}(H(pw))$ as did in G_6 .

Due to the *pseudo-randomness of public key* of LPKE and by hybrid arguments across the ℓ sessions, we have $|\Pr[G_7 \Rightarrow 1] - \Pr[G_6 \Rightarrow 1]| \leq \text{negl}(\lambda)$.

We note that the reduction proof for the above equation proceeds smoothly since sk is not needed any more in the simulation for client instances.

Game G_8 (simulation of generating c in second-round message). In G_8 , Sim is the same as in G_7 , except for Sim 's simulation of generating c in the second-round message (c, C) for server instances. There are two cases.

Case 1: Passive attacks on Servers. For a server instance $(S^{(j)}, iid')$ that is linked to some client instance $(C^{(i)}, iid)$ when receiving a first-round message pk , simulator Sim will sample c by $c \leftarrow_s \mathcal{CT}$, rather than computing it with $c \leftarrow \text{LEnc}(pk, H(pw), m; H_1(m))$ as did in G_7 .

Note that $r|\sigma|k \leftarrow_s \{0, 1\}^{3\lambda}$ and $C \leftarrow \text{CEnc}(cpk, pw|pk|c; r)$ are still computed in the same way as in G_7 .

Case 2: Active attacks on Servers. For a server instance $(S^{(j)}, iid')$ that is not linked to any client instance when receiving a first-round message pk , we further consider three sub-cases.

- Case 2.1:** $\exists (pw', r_H) \in \mathcal{L}_H$ s.t. $\text{lsLossy}(td, \tilde{pk}, r_H) = 0$. In this case, Sim will compute $c \leftarrow_{\$} \mathcal{CT}, r|\sigma|k \leftarrow_{\$} \{0, 1\}^{3\lambda}$, rather than computing $c \leftarrow \text{LEnc}(\tilde{pk}, H(pw), m; H_1(m))$ and $r|\sigma|k := H_2(m)$ as did in G_7 .
- Case 2.2:** $\exists (pw', r_H) \in \mathcal{L}_H$ s.t. $\text{lsLossy}(td, \tilde{pk}, r_H) = 0$. In this case, Sim extracts this password pw' and checks whether $pw' = pw$ or not.
- If $pw' \neq pw$, Sim simulates (c, C) just like Case 2.1.
 - If $pw' = pw$, Sim computes (c, C) just like G_7 , i.e., $c \leftarrow \text{LEnc}(\tilde{pk}, H(pw), m; H_1(m))$ and $C \leftarrow \text{CEnc}(cpk, pw|pk|c; r)$.
- Case 2.3:** $\exists (pw, r_H), (pw', r'_H) \in \mathcal{L}_H$ s.t. $\text{lsLossy}(td, \tilde{pk}, r_H) = \text{lsLossy}(td, \tilde{pk}, r'_H) = 0$. In this case, Sim just aborts the simulation directly.

In Case 1, pk is random and independent of password pw . Then $H(pw)$ is a random label w.r.t. pk . According to the *lossiness of random labels*, we have $\text{lsLossy}(td, pk, H(pw)) = 1$, i.e., $H(pw)$ is a lossy label of pk . Then according to Lemma 5, the ciphertext $c := \text{LEnc}(pk, H(pw), m; H_1(m))$ is pseudo-random in G_7 . Therefore, we can replace c with a random one as did in G_8 in a computationally indistinguishable way.

In Case 2.1, \mathcal{A} did not ever query $H(pw)$, and hence $H(pw)$ is random to \mathcal{A} . With a similar argument as Case 1, we can also replace c with a random one. Now that c is independent of m , then the randomness of m makes sure that \mathcal{A} hardly ever queries $H_2(m)$. So, $r|\sigma|k := H_2(m)$ is uniform and independent of c .

In Case 2.2, $H(pw')$ is a normal label of pk . If pw' is not the correct password, then $H(pw)$ must be a lossy label to \tilde{pk} except with a negligible probability. The reason is as follows. If $(pw, \cdot) \in \mathcal{L}_H$, then $\text{lsLossy}(td, \tilde{pk}, H(pw)) = 1$ must hold, hence $H(pw)$ is a lossy label of \tilde{pk} . If $(pw, \cdot) \notin \mathcal{L}_H$, then \mathcal{A} did not ever query $H(pw)$, so $H(pw)$ is random to \mathcal{A} . With a similar argument as Case 1, we can replace c with a random one in a computationally indistinguishable way.

Case 2.3 implies $\text{lsLossy}(td, \tilde{pk}, r_H) = \text{lsLossy}(td, \tilde{pk}, r'_H) = 0$, which happens with negligible probability according to the property of *uniqueness of normal labels among polynomial-size set* (among the all the labels stored in \mathcal{L}_H).

Accounting for the above cases, we have $|\Pr[G_8 \Rightarrow 1] - \Pr[G_7 \Rightarrow 1]| \leq \text{negl}(\lambda)$.

Game G_9 (simulation of generating C in the second-round message). In G_9 , Sim is the same as in G_8 , except for Sim's simulation of generating C in the second-round message (c, C) for client instances. We consider the same cases as in G_8 . In Case 1, Case 2.1 and the sub-case $pw' \neq pw$ in Case 2.2, Sim invokes $C \leftarrow \text{CEnc}(cpk, 0; r)$ rather than $C \leftarrow \text{CEnc}(cpk, pw|pk|c; H_1(m))$ as did in G_8 .

Note that r is random and independent of $pw|pk|c$. According to the CCA security of PKE and hybrid arguments over the (at most) ℓ ciphertexts, we have

$$|\Pr[G_9 \Rightarrow 1] - \Pr[G_8 \Rightarrow 1]| \leq \ell \cdot \text{Adv}_{\text{PKE}}^{\text{cca}}(\mathcal{B}_{\text{PKE}}) \leq \text{negl}(\lambda).$$

Game G_{10} (simulation of dealing with the third-round message $\tilde{\sigma}$ for server instances in case of active attacks). In G_{10} , Sim is the same as in G_9 , except for Sim's simulation of generating $s\text{Key}$ upon receiving the third-round message $\tilde{\sigma}$. Consider the same cases defined in G_9 (also G_8). In Case 1, Case

2.1 and the sub-case $pw' \neq pw$ in Case 2.2, Sim sets $\text{sKey} := \perp$ directly in \mathbf{G}_{10} regardless of whether $\sigma = \tilde{\sigma}$ or not.

\mathbf{G}_{10} is the same as \mathbf{G}_9 except that $\sigma = \tilde{\sigma}$ happens in these cases in \mathbf{G}_9 . However, σ is uniformly chosen and independent of other variables, and hence \mathcal{A} can present a correct guess of σ with negligible probability. So we have

$$|\Pr[\mathbf{G}_{10} \Rightarrow 1] - \Pr[\mathbf{G}_9 \Rightarrow 1]| \leq \text{negl}(\lambda).$$

Now Sim does not use pw anymore except for the comparison $pw' \stackrel{?}{=} pw$ in Case 2.2 for server instances and in the rejection rule (\star) for client instances.

Game \mathbf{G}_{11} (Integration of Sim with $\mathcal{F}_{\text{pake}}$). \mathbf{G}_{11} is the same as \mathbf{G}_{10} , except that Sim accesses $\mathcal{F}_{\text{pake}}$ by issuing $\text{Testpw}(pw')$ to decide $pw' \stackrel{?}{=} pw$ for Case 2.2 and (\star) . (The detail of \mathbf{G}_{11} and analysis is shown in Appendix C.2.) Note that $\text{Testpw}(pw')$ and $pw' \stackrel{?}{=} pw$ has the same functionality, so

$$\Pr[\mathbf{G}_{11} \Rightarrow 1] = \Pr[\mathbf{G}_{10} \Rightarrow 1].$$

Now that Sim completely gets rid of pw in the simulation, it finally arrives at Fig. 7 and \mathbf{G}_{11} is exactly $\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}}$.

Finally, by combining all the statements across \mathbf{G}_0 - \mathbf{G}_{11} , we know that

$$|\Pr[\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}] - \Pr[\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}}]| \leq \text{negl}(\lambda). \quad \square$$

Remark 2. For our UC-secure PAKE construction in ROM, it is possible for us to remove the CCA-secure encryption C in the second-round message, resulting in a more efficient PAKE scheme without affecting its UC security. Note that C was originally used to extract password pw in the proof. Benefiting from the RO model, now the simulator can accomplish the extraction of pw without C using a new strategy: the simulator checks if there exists a query $(pw, H(pw))$ and a query $(m, H_1(m))$ s.t. $c = \text{LPKE.Enc}(pk, H(pw), m; H_1(m))$ so as the password pw and the encrypted message m can be extracted from the ciphertext c . However, for our UC-secure PAKE construction in QROM, the new strategy does not work and we still need the CCA encryption of C for the extraction.

4 PAKE from Extractable LPKE in QROM

When considering post-quantum security in QROM, we have to consider a quantum adversary \mathcal{A} making quantum superposition access to the random oracle. The proving technique of keeping hash query lists and searching for all queries in the lists in ROM does not apply any more, since the no-cloning principle makes impossible to maintain a query list.

In order to achieve UC security in QROM, we have to adjust the four hash functions H, H_1, H_2, H_3 in our PAKE construction to avoid keeping hash lists for them in the proof.

- In PAKE^{RO} , hash function H_1 helps LPKE to achieve CCA security as a KEM. According to [21], the FO-transformation works well in QROM.

<p>Initialization</p> <p>Sim maintains $\boxed{\text{lists } \mathcal{L}_H, \mathcal{L}_{H_1}, \mathcal{L}_{H_2}, \mathcal{L}_{H_3}}$ sent, recv (all initialized to be empty) in the simulation</p> <ul style="list-style-type: none"> • $\mathcal{L}_H, \mathcal{L}_{H_1}, \mathcal{L}_{H_2}, \mathcal{L}_{H_3}$: store records to simulate random oracles H, H_1, H_2 and H_3 • sent : store messages sent by client/server instances • recv : store messages received by client/server instances <p>Sim invokes $(cpk, csk) \leftarrow \text{KeyGen}(1^\lambda)$ and $(pp, td') \leftarrow \text{eLSetup}(1^\lambda)$. Sim outputs $\text{crs} := (cpk, pp, H, H_1, H_2, H_3)$ and stores $td := (csk, td')$</p> <p>PAKE Sessions</p> <p>on (NewClient, $C^{(i)}, iid, S^{(j)}$) from \mathcal{F}_{pake}:</p> <p style="padding-left: 20px;">$pk \leftarrow \mathcal{PK}$, sent := sent $\cup \{(C^{(i)}, iid, pk)\}$, send pk from $C^{(i)}$ to \mathcal{A}.</p> <p>on (NewServer, $S^{(j)}, iid', C^{(i)}$) from \mathcal{F}_{pake} and pk from \mathcal{A} as a client message from $C^{(i)}$ to $(S^{(j)}, iid')$:</p> <p style="padding-left: 20px;">recv := recv $\cup \{(S^{(j)}, iid', pk)\}$</p> <p style="padding-left: 20px;">$\boxed{\text{If } \exists!(pw', r) \in \mathcal{L}_H \text{ and } \text{IsLossy}(td, pk, r) \neq 0 : } pw' \leftarrow \text{Extract}(td, pk)$</p> <p style="padding-left: 40px;">ask (Testpw, $S^{(j)}, iid', pw'$) to \mathcal{F}_{pake}</p> <p style="padding-left: 40px;">If \mathcal{F}_{pake} returns “correct guess”:</p> <p style="padding-left: 60px;">$m \leftarrow \{0, 1\}^\lambda$, $\boxed{c \leftarrow \text{LEnc}(pk, H(pw'), m; H_1(m))}$, $c \leftarrow \text{eLEnc}(pk, pw', m; H_1(m))$</p> <p style="padding-left: 60px;">Parse $H_2(m) = r \sigma k$, $C \leftarrow \text{Enc}(cpk, pw' pk c; r)$</p> <p style="padding-left: 60px;">sent := sent $\cup \{(S^{(j)}, iid', (c, C, k, \sigma))\}$</p> <p style="padding-left: 40px;">In other cases: $c \leftarrow \mathcal{CT}, r \sigma k \leftarrow \{0, 1\}^\lambda$, $C \leftarrow \text{CEnc}(cpk, 0; r)$, sent = sent $\cup \{(S^{(j)}, iid', (c, C, k, \sigma))\}$</p> <p style="padding-left: 40px;">Send (c, C) from $S^{(j)}$ to \mathcal{A}</p> <p>on (c, C) from \mathcal{A} as a server message from $S^{(j)}$ to $(C^{(i)}, iid)$:</p> <p style="padding-left: 20px;">recv := recv $\cup \{(C^{(i)}, iid, (c, C))\}$</p> <p style="padding-left: 20px;">If $\exists(iid', pk)$ such that $(C^{(i)}, iid, pk) \in \text{sent} \wedge (S^{(j)}, iid', pk) \in \text{recv} \wedge (S^{(j)}, iid', (c, C, k, \sigma)) \in \text{sent}$:</p> <p style="padding-left: 40px;">sent := sent $\cup \{(C^{(i)}, iid, \sigma)\}$, send σ from $C^{(i)}$ to \mathcal{A}</p> <p style="padding-left: 40px;">$sid := C^{(i)} S^{(j)} pk c C \sigma$, send (FreshKey, $C^{(i)}, iid, sid$) to \mathcal{F}_{pake}, exit</p> <p style="padding-left: 40px;">$pw' pk c' \leftarrow \text{CDec}(sk, C)$</p> <p style="padding-left: 40px;">If $c' \neq c \vee (C^{(i)}, iid, pk) \notin \text{sent}$: send (Abort, $C^{(i)}, iid$) to \mathcal{F}_{pake}, exit</p> <p style="padding-left: 40px;">ask Testpw($C^{(i)}, iid, pw'$). If \mathcal{F}_{pake} returns “wrong guess”: send (Abort, $C^{(i)}, iid$) to \mathcal{F}_{pake}, exit</p> <p style="padding-left: 20px;">$\boxed{\text{If } \exists(m, x) \in \mathcal{L}_{H_1} \text{ such that } c = \text{Enc}(pk, pw', m; x):}$ If $m \leftarrow \mathcal{S.E}(c)$ and $m \neq \perp$:</p> <p style="padding-left: 40px;">Parse $H_2(m) = r \sigma k$ and if $C = \text{Enc}(cpk, pw' pk c'; r)$:</p> <p style="padding-left: 60px;">$sid := C^{(i)} S^{(j)} pk c' C \sigma$, send (CorruptKey, $C^{(i)}, iid, sid, H_3(k sid)$) to \mathcal{F}_{pake}</p> <p style="padding-left: 60px;">sent := sent $\cup \{C^{(i)}, iid, \sigma\}$, send σ from $C^{(i)}$ to \mathcal{A}</p> <p style="padding-left: 40px;">In other cases: send (Abort, $C^{(i)}, iid$) to \mathcal{F}_{pake}, exit</p> <p>on σ from \mathcal{A} as a client message from $C^{(i)}$ to $(S^{(j)}, iid')$:</p> <p style="padding-left: 20px;">If $\exists(iid, pk, c, C)$ such that $\{(C^{(i)}, iid, pk), (C^{(i)}, iid, \sigma)\} \subseteq \text{sent} \wedge (S^{(j)}, iid', pk) \in \text{recv}$ $\wedge \{S^{(j)}, iid', (c, C, \cdot, \cdot)\} \in \text{sent} \wedge \{C^{(i)}, iid, (c, C, \cdot, \cdot)\} \in \text{recv}$:</p> <p style="padding-left: 40px;">$sid := C^{(i)} S^{(j)} pk c C \sigma$, send (CopyKey, $S^{(j)}, iid', sid$) to \mathcal{F}_{pake}</p> <p style="padding-left: 20px;">Else if $\exists k \neq \perp$ such that $(S^{(j)}, iid', (c, C, k, \sigma)) \in \text{sent}$:</p> <p style="padding-left: 40px;">$sid := C^{(i)} S^{(j)} pk c C \sigma$, send (CorruptKey, $S^{(j)}, iid', H_3(k sid)$) to \mathcal{F}_{pake}</p> <p style="padding-left: 20px;">In other cases: send (Abort, $S^{(j)}, iid'$) to \mathcal{F}_{pake}</p>	
<p>On Random Oracles</p> <p>on $H(pw)$ from \mathcal{A}:</p> <p style="padding-left: 20px;">If $\exists(pw, Y) \in \mathcal{L}_H$: return Y</p> <p style="padding-left: 20px;">Else: $Y \leftarrow \mathcal{T}$, $\mathcal{L}_H := \mathcal{L}_H \cup \{(pw, Y)\}$, return Y</p> <p>on $H_1(m)$ from \mathcal{A}:</p> <p style="padding-left: 20px;">If $\exists(m, Y) \in \mathcal{L}_{H_1}$: return Y</p> <p style="padding-left: 20px;">Else: $Y \leftarrow \{0, 1\}^\lambda$, $\mathcal{L}_{H_1} := \mathcal{L}_{H_1} \cup \{(m, Y)\}$, return Y</p> <p>on $H_2(m)$ from \mathcal{A}:</p> <p style="padding-left: 20px;">If $\exists(m, Y) \in \mathcal{L}_{H_1}$: return Y</p> <p style="padding-left: 20px;">Else: $Y \leftarrow \{0, 1\}^{3\lambda}$, $\mathcal{L}_{H_2} := \mathcal{L}_{H_2} \cup \{(m, Y)\}$, return Y</p> <p>on $H_3(m)$ from \mathcal{A}:</p> <p style="padding-left: 20px;">If $\exists(m, Y) \in \mathcal{L}_{H_3}$: return Y</p> <p style="padding-left: 20px;">Else: $Y \leftarrow \{0, 1\}^\lambda$, $\mathcal{L}_{H_3} := \mathcal{L}_{H_3} \cup \{(m, Y)\}$, return Y</p>	<p>On Random Oracles</p> <p>on $H_1(m)$ from \mathcal{A}:</p> <p style="padding-left: 20px;">$Y \leftarrow \mathcal{S.RO}(m)$</p> <p style="padding-left: 20px;">Return Y</p> <p>on $H_2(m)$ from \mathcal{A}:</p> <p style="padding-left: 20px;">$Y := H_2(m)$</p> <p style="padding-left: 20px;">Return Y</p> <p>on $H_3(m)$ from \mathcal{A}:</p> <p style="padding-left: 20px;">$Y := H_3(m)$</p> <p style="padding-left: 20px;">Return Y</p>

Fig. 7: Simulator Sim for PAKE^{RO} and Sim' for PAKE^{QRO} in the ideal world, with $\boxed{\text{text}}$ only appearing in Sim and text only appearing in Sim'.

- In PAKE^{RO} , the hash list of H_1 is also used for decryptions of c without secret key sk . To eliminate hash list of H_1 , we resort to the online-extractability technique due to Don et al. [21]. With this technique, we can construct a simulator $\mathcal{S} = (\mathcal{S}.RO, \mathcal{S}.E)$ who not only extracts the decrypting result m from $c = \text{LEnc}(pk, b, m; H_1(m)) = f_{pk,b}(m, H_1(m))$ with $\mathcal{S}.E$, but also simulates H_1 with $\mathcal{S}.RO$ in QROM. Note that the unpredictability of ciphertexts and the correctness of LPKE ensure not too many y 's satisfy $f_{pk,b}(m, y) = c$, which is a necessary condition for online-extractability to apply.
- In PAKE^{RO} , hash function H_2 and H_3 are used as PRF. Applying the O2H lemma enables us to prove the pseudo-randomness of $r|\sigma|k := H_2(m)$ and $s\text{Key} := H_3(k|sid)$ in QROM, when m and k are uniform.
- In PAKE^{RO} , hash list of H is used to extract password pw from the first-round message pk in the security proof in ROM. However, when formalizing H as a quantum accessible random oracle in $pk = \text{LKeyGen}(H(pw); r)$, an obstacle comes in the way: The O2H lemma does not apply because $\text{LKeyGen}(H(pw); r)$ is not in the form of $H(k|\cdot)$. The online-extractability does not apply either, since the simulator is not able to obtain the randomness r sampled by \mathcal{A} and hence cannot determine the function f to be extracted. To eliminate the obstacle, we discard hash function H and replace the building block LPKE with an “extractable” one named eLPKE. To accomplish password extraction from pk in QROM, eLPKE directly uses pw as its label and uses an extra algorithm `Extract` to extract pw from pk with the help of trapdoor.

In subsec. 4.1, we introduce the concept of extractable LPKE. In subsec. 4.2, we show how to construct extractable LPKE from LPKE. In subsec. 4.3, we show the generic construction of PAKE and prove its UC security in QROM.

4.1 Definition of Extractable LPKE (eLPKE)

When augmenting an extracting algorithm to a basic LPKE, we obtain *extractable LPKE* (eLPKE in short). An eLPKE scheme additionally requires *lossiness of random public keys* and *extractability of the unique normal label*, besides the properties of ①②⑤⑥ as per basic LPKE (see section 3.1).

Definition 6. *An extractable LPKE scheme eLPKE = (eLSetup, eLKeyGen, eLEnc, eLDec, elsLossy, Extract) consists of six algorithms, where (eLSetup, eLKeyGen, eLEnc, eLDec, elsLossy) are defined in the same way as the five algorithms (LSetup, LKeyGen, LEnc, LDec, lsLossy) in LPKE (cf. Def.4). Algorithm Extract is defined below.*

- `Extract(td, pk)` : *The extracting algorithm takes as input a trapdoor td and a public key pk , and outputs a label b .*

Correctness of eLPKE. It has the same correctness requirement as LPKE.

An extractable LPKE scheme eLPKE should satisfy the following properties and security requirements.

- ① **Pseudorandomness of Public Keys.** Same as ① of LPKE. (cf. Def. 4)
- ② **Random Ciphertexts under Lossy Labels.** Same as ② of LPKE.
- ③ **Extractability of the Unique Normal Label.** For every \mathcal{A} , it holds that

$$\Pr \left[\begin{array}{l} (\text{crs}, td) \leftarrow \text{eLSetup} \\ pk \leftarrow \mathcal{A}(\text{crs}); b \leftarrow \text{Extract}(td, pk) : \exists b' \neq b, \text{elsLossy}(td, pk, b') = 0 \end{array} \right] \leq \text{negl}(\lambda).$$

- ④ **Lossiness of Random Public Keys.** For every adversary \mathcal{A} , it holds that

$$\Pr \left[\begin{array}{l} (\text{crs}, td) \leftarrow \text{eLSetup} \\ b \leftarrow \mathcal{A}(\text{crs}), pk \leftarrow_{\$} \mathcal{PK} : \text{elsLossy}(td, pk, b) = 0 \end{array} \right] \leq \text{negl}(\lambda).$$

- ⑤ **Ciphertext Unpredictability under Normal Labels.** Same as ⑤ of LPKE. (cf. Def. 4)
- ⑥ **CPA Security under Normal Label.** Same as ⑥ of LPKE. (cf. Def. 4)

Remark 3. Property ③ means that for adversary's choice of pk , either there exists no normal label for pk or $\text{Extract}(td, pk)$ can extract a unique normal label for pk . In fact, Property ③ along with correctness of eLPKE imply that if $(pk, sk) \leftarrow \text{eLKeyGen}(b)$ then $\text{Extract}(td, pk)$ outputs the unique normal label b of pk .

4.2 Construction of eLPKE from LPKE⁺

Given a basic LPKE = (LSetup, LKeyGen, LEnc, LDec, lsLossy) with message space $\{0, 1\}^\lambda$ and label space \mathcal{T} , we construct an extractable LPKE scheme eLPKE with message space $\{0, 1\}^\lambda$ and label space $\{0, 1\}^\lambda$. See Fig.8 for eLPKE construction.

$\text{eLSetup}(1^\lambda) :$ $(pp', td') \leftarrow \text{LSetup}$ For $i := 1$ to λ : $v_i^0, v_i^1 \leftarrow_{\$} \mathcal{T}$ Return $(pp := (pp', \{v_i^0, v_i^1\}_{i \in [\lambda]}), td := td')$ $\text{eLKeyGen}(b = b_1 \dots b_\lambda \in \{0, 1\}^\lambda) :$ For $i := 1$ to λ : $(pk_i, sk_i) \leftarrow \text{LKeyGen}(v_i^{b_i})$ Return $pk := (pk_1, \dots, pk_\lambda), sk := (sk_1, \dots, sk_\lambda)$ $\text{eLEnc}(pk, b = b_1 \dots b_\lambda \in \{0, 1\}^\lambda, m \in \{0, 1\}^\lambda) :$ Parse $pk := (pk_1, \dots, pk_\lambda)$ For $i := 1$ to $\lambda - 1$: $z_i \leftarrow_{\$} \{0, 1\}^\lambda$ $z_\lambda := m \oplus z_1 \oplus \dots \oplus z_{\lambda-1}$ For $i := 1$ to λ : $c_i \leftarrow \text{LEnc}(pk_i, v_i^{b_i}, z_i)$ Return $c := (c_1, \dots, c_\lambda)$	$\text{eLDec}(sk = (sk_1, \dots, sk_\lambda), c = (c_1, \dots, c_\lambda)) :$ For $i := 1$ to λ : $z_i \leftarrow \text{LDec}(sk_i, c_i)$ Return $m := z_1 \oplus \dots \oplus z_\lambda$ $\text{elsLossy}(td, pk, b = b_1 \dots b_\lambda \in \{0, 1\}^\lambda) :$ Parse $pk = (pk_1, \dots, pk_\lambda)$ For $i := 1$ to λ : If $\text{lsLossy}(td, pk_i, v_i^{b_i}) = 1$: Return 1 Return 0 $\text{Extract}(td, pk = (pk_1, \dots, pk_\lambda)) :$ For $i := 1$ to λ : If $\text{lsLossy}(td, pk_i, v_i^0) = 0$: $b_i := 0$ Else : $b_i := 1$ Return $b := b_1 \dots b_\lambda$
--	---

Fig. 8: Construction of eLPKE from LPKE⁺.

We will prove the property of eLPKE if the underlying basic LPKE not only satisfies property ①②③⑤⑥ (cf. Def. 4) but also two additional properties ④ (cf. Def. 6) and ⑦, where ⑦ is defined below.

- ⑦ **Ciphertext Randomness in case of Random Messages for LPKE.** For all $(\text{pp}, td) \leftarrow \text{LSetup}(1^\lambda)$, every (possibly malformed) public key pk , every label $b \in \mathcal{T}$ (no matter whether $\text{lsLossy}(td, pk, b) = 1$ or not), it holds that $c \approx_s \$$, where $m \leftarrow_s \{0, 1\}^\lambda$ and $c \leftarrow \text{LEnc}(pk, b, m)$.

We designate such a LPKE as LPKE^+ if it satisfies ①②③④⑤⑥⑦.

Theorem 2. *For the construction of eLPKE in Fig. 8, if the underlying LPKE is a LPKE^+ scheme, i.e., it satisfies ①②③④⑤⑥⑦, then the resulting eLPKE scheme has the properties of ①②③④⑤⑥ and supports label space $\{0, 1\}^\lambda$.*

Proof. For eLPKE, its properties of ①④⑤⑥ follow directly from ①④⑤⑥ of the underlying LPKE^+ . Now we prove ②③ for eLPKE.

② **Random Ciphertexts under Lossy Labels:** We aim to show $(\text{pp}, \mathbf{pk}, \mathbf{b}, m, \mathbf{c}) \approx_s (\text{pp}, \mathbf{pk}, \mathbf{b}, m, \$)$, where $(\text{pp}, td) \leftarrow \text{eLSetup}$, $(\mathbf{pk}, \mathbf{b}) \leftarrow \mathcal{A}(\text{pp})$ s.t. $\text{elsLossy}(td, \mathbf{pk}, \mathbf{b}) = 1$, $m \leftarrow_s \{0, 1\}^\lambda$, $z_i \leftarrow_s \{0, 1\}^\lambda$ for $i \in [\lambda - 1]$, $z_\lambda := m \oplus z_1 \oplus \dots \oplus z_{\lambda-1}$, $c_i \leftarrow \text{LEnc}(pk_i, v_i^{b_i}, z_i)$ for $i \in [\lambda]$ and $\mathbf{c} = c_1 | \dots | c_\lambda$. Given $\text{elsLossy}(td, \mathbf{pk}) = 1$, there must exist a position j such that $\text{lsLossy}(td, pk_j, v_j^{b_j}) = 1$. Then we can replace c_j in \mathbf{c} with a random $c_j \leftarrow_s \mathcal{CT}$ by the property of “② random ciphertexts under lossy labels” of the underlying LPKE^+ scheme.

Now m and $\{z_i\}_{i \in [\lambda]}$ can be sampled in an equivalent way: $m \leftarrow_s \{0, 1\}^\lambda$, $z_i \leftarrow_s \{0, 1\}^\lambda$ for $i \in [\lambda] \setminus \{j\}$, and set $z_j := m \oplus \bigoplus_{i=1, i \neq j}^\lambda z_i$. Note that plaintexts $\{z_i\}_{i \in [\lambda] \setminus \{j\}}$ encrypted in $\{c_i\}_{i \in [\lambda] \setminus \{j\}}$ are independent of message m . Therefore, by “⑦ ciphertext randomness in case of random messages” of underlying LPKE^+ scheme, we can replace c_i with a random $c_i \leftarrow_s \mathcal{CT}$ for each $i \in [\lambda] \setminus \{j\}$. Together with the random ciphertext c_j , we arrive at the right side in a statistical indistinguishable way.

③ **Extractability of the Unique Normal Label:** Suppose toward contradiction, there is a \mathbf{b}' such that $\mathbf{b}' \neq \mathbf{b}$ and $\text{elsLossy}(td, \mathbf{pk}, \mathbf{b}') = 0$, where $(\text{pp}, td) \leftarrow \text{eLSetup}$, $pk \leftarrow \mathcal{A}(\text{pp})$, $\mathbf{b} \leftarrow \text{Extract}(td, pk)$.

Given $\mathbf{b}' \neq \mathbf{b}$, there must exist a position $j \in [\lambda]$ s.t. $b'_j \neq b_j$. Given $\text{elsLossy}(td, \mathbf{b}', \mathbf{pk}) = 0$, it holds that $\text{lsLossy}(td, pk_i, v_i^{b'_i}) = 0$ for $i \in [\lambda]$. Now we have $b'_j \neq b_j$ and $\text{lsLossy}(td, pk_j, v_j^{b'_j}) = 0$. According to the specification of $\mathbf{b} \leftarrow \text{Extract}(td, \mathbf{pk})$, we know that $b_j = 0$ if and only if $\text{lsLossy}(td, pk_j, v_j^0) = 0$. We consider two cases according to the value of b_j .

Case 1: $b_j = 0$ (so $b'_j = 1$). In this case, we have $\text{lsLossy}(td, pk_j, v_j^0) = \text{lsLossy}(td, pk_j, v_j^1) = 0$, which contradicts to property ③ of the LPKE^+ .

Case 2: $b_j = 1$ (so $b'_j = 0$). In this case, we have $\text{lsLossy}(td, pk_j, v_j^0) \neq 0$ which contradicts to the fact that $\text{lsLossy}(td, pk_j, v_j^{b'_j}) = \text{lsLossy}(td, pk_j, v_j^0) = 0$. \square

Remark 4. With the FO-transformation, we can construct a KEM scheme KEM' from eLPKE. Note that Lemma 4 and Lemma 5 remain applicable to eLPKE since eLPKE has properties of ②⑤⑥. Therefore, KEM' has both CCA-security and ciphertext pseudo-randomness under lossy labels.

4.3 Construction of PAKE from eLPKE in QROM

Replacing the building block of basic LPKE with eLPKE, we obtain the generic construction of PAKE from eLPKE. The resulting PAKE scheme PAKE^{QRO} is shown in Fig. 6 and its UC security proof in QROM is shown in Theorem 3.

Theorem 3. *If eLPKE is an extractable LPKE scheme, PKE is a CCA-secure PKE, and H_1, H_2, H_3 are quantum-accessible random oracles, then scheme PAKE^{QRO} in Fig. 6 securely emulates $\mathcal{F}_{\text{pake}}$, hence achieving UC security in QROM.*

The proof outline is similar to that of Theorem 1, where $G'_1, G'_2, G'_5, G'_7, G'_9-G'_{11}$ are the same as $G_1, G_2, G_5, G_7, G_9-G_{11}$. Note that the simulator will not keep the hash lists. The differences lie in G'_3, G'_4, G'_6, G'_8 , for which we give a brief overview.

- G'_3 & G'_8 . We do not maintain hash list for $H_2(\cdot)$ to compute $r|\sigma|k$ as did in G_3 & G_8 . Instead, we make use of Lemma 4 & Lemma 5 to argue $r|\sigma|k := H_2(m)$ is pseudo-random.
- G'_4 . We do not maintain hash list for $H_3(\cdot)$ to compute sKey as did in G_4 . Instead, we make use of Lemma 10 to argue $\text{sKey} := H_3(k|\text{sid})$ is pseudorandom.
- G'_6 . For client instances, we have to eliminate the usage of sk of eLPKE so that the first round message pk can be replaced with a random one in G'_7 . We do not use the hash list for $H_1(\cdot)$, as did in G_6 . Instead we resort to the online-extractability technique and make use of the corresponding simulator $\mathcal{S} = (\mathcal{S}.RO, \mathcal{S}.E)$ to simulates random oracle $H_1(\cdot)$ and extract the decryption result of $\text{eLDec}(sk, c)$.
- G'_8 . For server instances receiving the first-round message pk in an active attack, we do not maintain the hash list for $H(\cdot)$ and search the list to find the correct password pw (if exists). Instead we use $\text{Extract}(td, pk)$ to extract pw .

The full description of the proof is shown in Appendix C.3.

5 Instantiations

In our generic PAKE constructions, one building block is a CCA-secure PKE scheme, which can be easily obtained from CPA-secure PKE via FO-transformation in (quantum) ROM [23,36]. For example, by applying FO-transformation to the Regev PKE [32] or ElGamal-like PKE from GA-DDH (Section 7.1 in [12]), we can obtain CCA-secure PKE from the LWE or GA-DDH assumption. Therefore, to obtain specific PAKE schemes, we only consider the instantiations of LPKE.

In subsec. 5.1 and 5.2, we give the instantiations of LPKE and LPKE^+ , from LWE and GA-DDH respectively. The instantiations of CCA-secure PKE and LPKE yield four specific post-quantum UC-secure PAKE schemes, two in ROM and the other two in QROM, as shown in subsec. 5.3.

5.1 LPKE and LPKE⁺ Schemes from LWE

We present LWE-based LPKE scheme LPKE_{lwe} and LPKE⁺ scheme $\text{LPKE}'_{\text{lwe}}$. Before presenting our scheme LPKE_{lwe} , we recall some technical tools including an important algorithm called IsMessy introduced in [22].

- **Statistical distance** $\delta_{q,r}(\mathbf{A}, \mathbf{x})$. Given $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{x} \in \mathbb{Z}_q^m$, define

$$\delta_{q,r}(\mathbf{A}, \mathbf{x}) := \Delta((\mathbf{A}\mathbf{e}, \mathbf{x}^T \mathbf{e}), (\mathbf{u}, u)), \quad (1)$$

where $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, r}$, $\mathbf{u} \leftarrow_s \mathbb{Z}_q^n$, $u \leftarrow_s \mathbb{Z}_q$ and Δ is the statistical distance.

- **Algorithm** $\text{IsMessy}(\mathbf{T}_\mathbf{A}, \mathbf{A}, \mathbf{x})$. It takes as input a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, \mathbf{A} 's trapdoor $\mathbf{T}_\mathbf{A}$, and a vector $\mathbf{x} \in \mathbb{Z}_q^m$ and outputs “messy” or “not sure”.

Lemma 6 (Proposition 7.8 in [30]). *Let $m \geq 2(n+1) \log q$ and let $r \geq \sqrt{qm} \cdot \log^2 m$. Suppose that $(\mathbf{A}, \mathbf{T}_\mathbf{A})$ are generated by TrapGen (c.f. Lemma 1). Then there exists a PPT algorithm $\text{IsMessy}(\mathbf{T}_\mathbf{A}, \mathbf{A}, \mathbf{x})$ satisfying the following statements.*

- (a) *With overwhelming probability over the choice of $\mathbf{A}, \mathbf{T}_\mathbf{A}$, for all but an at most $(1/2\sqrt{q})^m$ fraction of vectors $\mathbf{x} \in \mathbb{Z}_q^m$, $\text{IsMessy}(\mathbf{T}_\mathbf{A}, \mathbf{A}, \mathbf{x})$ outputs “messy” with overwhelming probability (over its own randomness).*
- (b) *There exists $\epsilon(\lambda) = \text{negl}(\lambda)$ such that if $\delta_{q,r}(\mathbf{A}, \mathbf{x}) > 2\epsilon(\lambda)$, then $\text{IsMessy}(\mathbf{T}_\mathbf{A}, \mathbf{A}, \mathbf{x}) = \text{“not sure”}$, with overwhelming probability (over its own randomness). (In other words, if $\text{IsMessy}(\mathbf{T}_\mathbf{A}, \mathbf{A}, \mathbf{x}) = \text{“messy”}$, then $\delta_{q,r}(\mathbf{A}, \mathbf{x}) = \text{negl}$ with overwhelming probability.)*

Our basic LPKE scheme LPKE_{lwe} is described as follows.

LSetup (1^λ):	LDec ($sk = \mathbf{s}, ct = (\mathbf{u}_1 \dots \mathbf{u}_\lambda, c_1 \dots c_\lambda)$):
$(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \text{TrapGen}(1^\lambda)$	For $i := 1$ to λ :
Return $(pp := \mathbf{A}, td := \mathbf{T}_\mathbf{A})$	$d_i := c_i - \mathbf{s}^T \mathbf{u}_i$
LKeyGen ($\mathbf{v} \in \mathbb{Z}_q^m$):	If $d_i \in [q/4, 3q/4]$: $m_i := 1$
$\mathbf{s} \leftarrow_s \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, \sigma}$	Else $m_i := 0$
Return $(pk := \mathbf{A}^T \mathbf{s} + \mathbf{e} - \mathbf{v}, sk := \mathbf{s})$	Return $m := m_1 \dots m_\lambda$
LEnc ($pk, \mathbf{v}, m \in \{0, 1\}^\lambda$):	IsLossy ($td = \mathbf{T}_\mathbf{A}, pk, \mathbf{v}$):
$\mathbf{p} := pk + \mathbf{v}$	If $\text{IsMessy}(\mathbf{T}_\mathbf{A}, \mathbf{A}, pk + \mathbf{v}) = \text{“messy”}$:
For $i := 1$ to λ :	Return 1
$\mathbf{e}_i \leftarrow D_{\mathbb{Z}^m, r}$, $\mathbf{u}_i := \mathbf{A} \mathbf{e}_i$	Else: Return 0
$c_i := \mathbf{p}^T \mathbf{e}_i + m_i \cdot \frac{q}{2}$	
Return $ct := (\mathbf{u}_1 \dots \mathbf{u}_\lambda, c_1 \dots c_\lambda)$	

Fig. 9: The basic LPKE scheme LPKE_{lwe} .

Theorem 4. *If $q > 4r\sigma m \log^2 n$, $m > 2(n+1) \log q$, and $r > \sqrt{qm} \log^2 m$, then LPKE_{lwe} is a basic LPKE scheme satisfying properties ①-⑥ based on $\text{LWE}_{n,q,m,D_{\mathbb{Z}^m,\sigma}}$ assumption.*

Proof. We prove that LPKE_{lwe} has correctness and the corresponding properties. By the tail bound (cf. Lemma 7), $\|\mathbf{e}\| \leq \omega(\sqrt{\log n}) \cdot \sigma\sqrt{m}$ and $\|\mathbf{e}_i\| \leq \omega(\sqrt{\log n}) \cdot r\sqrt{m}$. The correctness follows from $|\mathbf{e}^T \mathbf{e}_i| \leq \|\mathbf{e}\| \|\mathbf{e}_i\| \leq r\sigma m \cdot \omega(\log n) < q/4$.

① **Pseudorandomness of Public Key:** According to Lemma 1, the matrix \mathbf{A} outputted from TrapGen is statistically close to a uniform distribution. Then *pseudorandomness of public key* follows from the LWE assumption.

② **Random Ciphertext under Lossy Labels:** By statement (b) of Lemma 6 and (1), we know $(\mathbf{A}\mathbf{e}_i, \mathbf{p}^T \mathbf{e}_i) \approx_s (\$, \$)$ and independent of $(\mathbf{A}, pk, \mathbf{v})$ for $i \in [\lambda]$. So we have $(\mathbf{A}, pk, \mathbf{v}, m, \{\mathbf{A}\mathbf{e}_i, \mathbf{p}^T \mathbf{e}_i + m_i \cdot \frac{q}{2}\}_{i \in [\lambda]}) \approx_s (\mathbf{A}, pk, \mathbf{v}, m, \{\$, \$\}_{i \in [\lambda]})$.

③ **Uniqueness of Normal Labels among Polynomial-Size Set:** According to statement (a) of Lemma 6, we know that there are $1 - \text{negl}(\lambda)$ fraction of matrices \mathbf{A} (and $\mathbf{T}_{\mathbf{A}}$) such that $\Pr[\mathbf{x} \leftarrow_s \mathbb{Z}_q^m : \text{lsMessy}(\mathbf{T}_{\mathbf{A}}, \mathbf{A}, \mathbf{x}) = \text{“messy”}] \geq 1 - \text{negl}(\lambda)$. Let \mathbf{A} (and $\mathbf{T}_{\mathbf{A}}$) be such a fixed matrix, and define set $\mathcal{S} := \{\mathbf{x} \in \mathbb{Z}_q^m \mid \text{lsMessy}(\mathbf{T}_{\mathbf{A}}, \mathbf{A}, \mathbf{x}) = \text{“messy”}\}$. Then $\Pr[\mathbf{x} \leftarrow_s \mathbb{Z}_q^m : \mathbf{x} \notin \mathcal{S}] \leq (1/2\sqrt{q})^m$.

Further fixing a public key $pk = \mathbf{p} \in \mathbb{Z}_q^m$, we have

$$\Pr[\mathbf{x}_0, \mathbf{x}_1 \leftarrow_s \mathbb{Z}_q^m : \mathbf{p} + \mathbf{x}_0 \notin \mathcal{S} \wedge \mathbf{p} + \mathbf{x}_1 \notin \mathcal{S}] = (\Pr[\mathbf{x} \leftarrow_s \mathbb{Z}_q^m : \mathbf{x} \notin \mathcal{S}])^2 \leq (1/4q)^m.$$

By a union bound over $Q(Q-1)/2 (\leq Q^2)$ possible pairs of $(\mathbf{x}_i, \mathbf{x}_j)$, we have

$$\Pr[\mathbf{x}_1, \dots, \mathbf{x}_Q \leftarrow_s \mathbb{Z}_q^m : \exists i \neq j, i, j \in [Q], \mathbf{p} + \mathbf{x}_i \notin \mathcal{S} \wedge \mathbf{p} + \mathbf{x}_j \notin \mathcal{S}] \leq Q^2 \cdot (1/4q)^m.$$

If $\mathbf{x}_1, \dots, \mathbf{x}_Q \leftarrow_s \mathbb{Z}_q^m$, then with a union bound over all q^m possible $pk = \mathbf{p} \in \mathbb{Z}_q^m$, we have $\Pr[\exists \mathbf{p} \in \mathbb{Z}_q^m, \exists i \neq j, \mathbf{p} + \mathbf{x}_i \notin \mathcal{S} \wedge \mathbf{p} + \mathbf{x}_j \notin \mathcal{S}] \leq Q^2 \cdot (1/4)^m \leq \text{negl}(\lambda)$, so $\Pr[\exists \mathbf{p} \in \mathbb{Z}_q^m, \exists i \neq j, \text{lsLossy}(\mathbf{T}_{\mathbf{A}}, \mathbf{p}, \mathbf{x}_i) = 0 \wedge \text{lsLossy}(\mathbf{T}_{\mathbf{A}}, \mathbf{p}, \mathbf{x}_j) = 0] \leq \text{negl}(\lambda)$.

④ **Lossiness of Random Labels:** It follows directly from (b) of Lemma 6.

⑤ **Ciphertext Unpredictability under Normal Labels:** According to Lemma 3, we know that $\mathbf{A}\mathbf{e}$ is statistically close to a uniform distribution, hence the probability that a ciphertext output from the encryption algorithm collides with a fixed ciphertext should be negligible.

⑥ **CPA Security under Normal Labels:** Indeed, the encryption scheme under a normal label is a variant of the Regev public key encryption scheme [32]. So it naturally inherits the CPA security from the Regev cryptosystem. \square

Remark 5. Our basic LPKE scheme is adapted from the LWE-based dual-mode PKE scheme [30] but with the following differences.

- **Different label space.** The dual-mode PKE [30] only supports a simple label space $\{0, 1\}$, while our LPKE_{lwe} has label space \mathbb{Z}_q^m , which is compatible to the hash value $H(pw)$ of password in the PAKE scheme.
- **Different syntax.** In [30], the CRS consists of two fixed vectors corresponding to label 0 and label 1 respectively. And there are two indistinguishable ways for generating CRS to determine the normal mode or the messy mode. In our LPKE_{lwe} , the CRS only has one mode and the public key together with a label determine the encryption is in a normal mode or a lossy mode.
- **Different security requirements.** Due to the different syntax and different applications, we also have different security requirements for LPKE_{lwe} .

Parameters. Set $n = \Theta(\lambda)$, $m = \Theta(\lambda \log \lambda)$, $q = \Theta(\lambda^5 \log^3 \lambda)$, $r = \Theta(\lambda^{3.5})$, and $\sigma = \Theta(\sqrt{\lambda})$. Such parameters satisfy the requirements in Theorem 4.

LPKE_{lwe} is not a LPKE^+ scheme since it does not satisfy “ $\textcircled{7}$ ciphertext randomness in case of random messages”. Recall that the ciphertext element $c_i = \mathbf{p}^T \mathbf{e}_i + m_i \cdot \frac{q}{2}$. We can consider $k_i := \mathbf{p}^T \mathbf{e}_i \in \mathbb{Z}_q$ as an ephemeral key used to hide the bit $m_i \in \{0, 1\}$. However, the uniformity of one bit $m_i \in \{0, 1\}$ is not sufficient to fully randomize $c_i \in \mathbb{Z}_q$. To solve this problem, we introduce the novel round function $R(\cdot)$ parameterized by T from [11] to result in $k_i := R(\mathbf{p}^T \mathbf{e}_i) \in \{0, 1\}$ so that $c_i := k_i \oplus m_i \in \{0, 1\}$. In this way, property $\textcircled{7}$ is achieved. However, CPA-security is lost since $R(\mathbf{p}^T \mathbf{e}_i)$ is not pseudo-random. To fix that, we add multiple $R(\mathbf{p}^T \mathbf{e}_{ij})$ to get $k_i := \bigoplus_{j \in [\lambda]} R(\mathbf{p}^T \mathbf{e}_{ij})$. By the LWE assumption and leftover hash lemma, $\mathbf{p}^T \mathbf{e}_{ij}$ can be replaced by a uniform element \mathbf{u}_{ij} . Then the rounding function R makes $R(\mathbf{u}_{ij})$ follow a Bernoulli distribution with parameter about $\frac{1}{3}$. By the piling-up lemma (cf. Lemma 8 in Appendix A), $k_i := \bigoplus_{j \in [\lambda]} R(\mathbf{u}_{ij})$ is statistically close to the uniform distribution, thus achieving CPA-security.

LSetup (1^λ) :	LDec ($sk = \mathbf{s}, ct$) :
$(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \text{TrapGen}(1^n, 1^m, q)$	Parse $ct = (\{\mathbf{c}_{ij}\}_{i,j \in [\lambda]}, \{\beta_i\}_{i \in [\lambda]})$.
Return $(pp := \mathbf{A}, td := \mathbf{T}_\mathbf{A})$	For $i := 1$ to λ :
LKeyGen ($\mathbf{v} \in \mathbb{Z}_q^m$) :	$m_i := \bigoplus_{j \in [\lambda]} R(\mathbf{s}^T \mathbf{c}_{ij}) \oplus \beta_i$
$\mathbf{s} \leftarrow_s \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow_s D_{\mathbb{Z}^m, t}$	Return $m = m_1 \dots m_\lambda$
Return $(pk := \mathbf{A}^T \mathbf{s} + \mathbf{e} - \mathbf{v}, sk := \mathbf{s})$	IsLossy ($td = \mathbf{T}_\mathbf{A}, pk, \mathbf{v}$) :
LEnc ($pk, m = m_1 \dots m_\lambda \in \{0, 1\}^\lambda, \mathbf{v}$) :	$\mathbf{p} := pk + \mathbf{v}$
$\mathbf{p} := pk + \mathbf{v}$	$(\mathbf{s}, \mathbf{e}) \leftarrow \text{Invert}(\mathbf{T}_\mathbf{A}, \mathbf{A}, \mathbf{p})$
For $i := 1$ to λ :	$\text{//See Lemma 2 for Invert}$
For $j := 1$ to λ :	If $\ \mathbf{e}\ \leq q/8\sqrt{m}$: Return 0
$\mathbf{e}_{ij} \leftarrow D_{\mathbb{Z}^m, r}$, $\mathbf{c}_{ij} := \mathbf{A} \mathbf{e}_{ij}$	Else: Return 1
$\beta_i := (\bigoplus_{j \in [\lambda]} R(\mathbf{p}^T \mathbf{e}_{ij})) \oplus m_i$	
Return $ct = (\{\mathbf{c}_{ij}\}_{i,j \in [\lambda]}, \{\beta_i\}_{i \in [\lambda]})$	

Fig. 10: LPKE^+ scheme $\text{LPKE}'_{\text{lwe}}$.

Our LPKE^+ scheme $\text{LPKE}'_{\text{lwe}}$ is adapted from [31,11] and shown in Fig. 10. The security proof for $\text{LPKE}'_{\text{lwe}}$ is shown in Theorem 5 in Appendix C.4.

Theorem 5. *If $r > n \log n$, $trm/q = \text{negl}(\lambda)$, $T/q = \text{negl}(\lambda)$, $m = \Theta(n \log n)$, and $\frac{\sqrt{m}}{r} (\frac{nq}{T})^2 < \Theta(\sqrt{m})$, then $\text{LPKE}'_{\text{lwe}}$ in Fig. 10 is a LPKE^+ scheme based on the $\text{LWE}_{n,q,m,D_{\mathbb{Z}^m,t}}$ assumption.*

Parameters. Set $n = \Theta(\lambda)$, $q = 2^\lambda$, $m = \Theta(\lambda^2)$, $t = \tilde{O}(\lambda^{3/2})$, $T = 2^{\frac{2\lambda}{3}}$, $k = \Theta(\lambda)$, and $r = \Omega(\lambda^4 \cdot 2^{\frac{2\lambda}{3}})$. Such parameters satisfy the requirements in Theorem 5.

5.2 LPKE and LPKE⁺ Scheme from Group Actions

The second LPKE scheme LPKE_{ga} is based on restricted effective group actions (REGA). It is adapted from the construction of the dual-mode PKE from group actions in [5]. Our LPKE_{ga} vs. the dual-mode PKE from group actions in [5] is analogous to LPKE_{lwe} vs. the dual-mode PKE from LWE in [30] (See Remark 5).

Let $(\mathbb{G}, \mathcal{X}, \star)$ be an REGA, $\mathcal{H} = \{\mathsf{H} : \mathcal{X}^\ell \rightarrow \{0, 1\}\}$ be a family of pairwise independent universal hash functions. Scheme LPKE_{ga} is described in Fig. 11.

$\text{LSetup}(1^\lambda) :$ $x \leftarrow_s \mathcal{X}, t \leftarrow_s \mathbb{G}, \bar{x} \leftarrow t \star x, \mathsf{H} \leftarrow_s \mathcal{H}$ Return $(\text{pp} := (x, \bar{x}, \mathsf{H}), \text{td} := t)$	$\text{LKeyGen}(\text{pp}, g \in \mathbb{G}) :$ $s \leftarrow_s \mathbb{G}$ Return $(pk := (s \star x, (g \cdot s) \star \bar{x}), sk := s)$
$\text{LEnc}(pk = (y, \bar{y}), g \in \mathbb{G}, m = m_1 \dots m_\lambda \in \{0, 1\}^\lambda) :$ For $i := 1$ to λ : $\mathbf{r}_i = (r_{i1}, \dots, r_{i\ell}) \leftarrow_s \mathbb{G}^\ell$ $\mathbf{b}_i = (b_{i1}, \dots, b_{i\ell}) \leftarrow_s \{0, 1\}^\ell$ For $j := 1$ to ℓ : If $b_{ij} = 0$: $c_{ij} := r_{ij} \star \bar{x}, c'_{ij} := r_{ij} \star (g^{-1} \star \bar{y})$ Else $c_{ij} := r_{ij} \star x, c'_{ij} := r_{ij} \star y$ $\mathbf{c}_i := (c_{i1}, \dots, c_{i\ell}) \in \mathcal{X}^\ell, \mathbf{c}'_i := (c'_{i1}, \dots, c'_{i\ell}) \in \mathcal{X}^\ell$ $ct_i := (\mathbf{c}_i, \mathsf{H}(\mathbf{c}'_i) \oplus m_i)$ Return $ct := (ct_1, \dots, ct_\lambda)$	$\text{LDec}(sk = s, ct) :$ Parse $ct = (\mathbf{c}_1, b_1), \dots, (\mathbf{c}_\lambda, b_\lambda)$ For $i := 1$ to λ : $\mathbf{z}_i := (s \star \mathbf{c}_i) = (s \star c_{i1}, \dots, s \star c_{i\ell})$ $m_i := b_i \oplus \mathsf{H}(\mathbf{z}_i)$ Return $m := m_1 \dots m_\lambda$
$\text{IsLossy}(\text{td} = t \in \mathbb{G}, pk = (y, \bar{y}, g \in \mathbb{G})) :$ If $\bar{y} = (t \cdot g) \star y$: Return 0 Else: Return 1	

Fig. 11: LPKE and LPKE⁺ scheme LPKE_{ga} .

Theorem 6. *If $(\mathbb{G}, \mathcal{X}, \star)$ is a restricted effective group action, then LPKE_{ga} in Fig. 11 is both a LPKE scheme and a LPKE⁺ scheme based on GA-DDH assumption.*

Proof. The proofs are similar to [5], so we only provide a concise overview.

① **Pseudorandomness of Public Keys:** We need to show that

$$(x, t \star x, s \star x, (g \cdot s \cdot t) \star x) \approx_c (x, t \star x, u_1, u_2) \quad (2)$$

for all $g \in \mathbb{G}$ provided by adversary \mathcal{A} , where $s, t \leftarrow_s \mathbb{G}$ and $u_1, u_2 \leftarrow_s \mathcal{X}$. The GA-DDH assumption requires $(x, s \star x, t \star x, (s \cdot t) \star x) \approx_c (x, s \star x, t \star x, z \star x)$, where $s, t, z \leftarrow_s \mathbb{G}$. Let g act on the last term, we have $(x, s \star x, t \star x, g \star ((s \cdot t) \star x)) \approx_c (x, s \star x, t \star x, g \star (z \star x))$. Note that $g \star ((s \cdot t) \star x) = (g \cdot s \cdot t) \star x$, and the uniformity and independence of x and z guarantees that $s \star x$ and $g \star (z \star x)$ are uniformly and independently distributed. So we obtain (2).

② **Random Ciphertexts under Lossy Labels:** Recall that $\text{pp} = (x, \bar{x} = t \star x, \mathsf{H}), pk = (y, \bar{y})$. Given a lossy label $g \in \mathbb{G}$ from \mathcal{A} , we have $\text{IsLossy}(\text{td}, pk, g) = 1$, i.e., $\bar{y} \neq (t \cdot g) \star y$. When writing $\bar{y} = (t \cdot g') \star y$, we have $g \neq g'$. Now the encryption scheme using label g becomes

$$c_j = \begin{cases} (r_j \cdot t) \star x & \text{if } b_j = 0 \\ r_j \star x & \text{if } b_j = 1 \end{cases}, \quad c'_j = \begin{cases} (r_j \cdot (g^{-1} \cdot g') \cdot t) \star y & \text{if } b_j = 0 \\ r_j \star y & \text{if } b_j = 1 \end{cases}.$$

Conditioned on pp, pk, g, c_j , bit b_j is perfectly hidden in c_j thanks to the randomness of r_j . Suppose $y = s \star x$. Then given $(x, t \star x, y, (t \cdot g') \star y, g, m, c_j)$, we know either $c'_j = s \star (g^{-1} \cdot g') \star c_j$ in case of $b_j = 0$ or $c'_j = s \star c_j$ in case of $b_j = 1$. Now that $g \neq g'$, so c'_j has one bit entropy and hence \mathbf{c}' has ℓ bits entropy.

As a result, the vector $\mathbf{c}' = (c'_1, \dots, c'_\ell)$ has ℓ bit entropy and by the leftover hash lemma, $\mathbf{H}(\mathbf{c}')$ is close to uniform distribution, thus hiding m_i statistically.

③ **Uniqueness of Normal Labels among Polynomial-Size Set:** By the regularity of the group action, we have

$$\begin{aligned} & \Pr \left[\begin{array}{l} x \leftarrow_s \mathcal{X}, t \leftarrow_s \mathbb{G}, \bar{x} := t \star x \\ g_1, \dots, g_Q \leftarrow_s \mathbb{G} \end{array} : \begin{array}{l} \exists (y, \bar{y}) \in \mathcal{X} \times \mathcal{X}, i \neq j \\ \bar{y} = (t \cdot g_i) \star y \wedge \bar{y} = (t \cdot g_j) \star y \end{array} \right] \\ &= \Pr [g_1, \dots, g_Q \leftarrow_s \mathbb{G} : \exists i \neq j, g_i = g_j] \leq Q^2/|\mathbb{G}| = \text{negl}(\lambda). \end{aligned}$$

④&④ **Lossiness of Random Labels/Random Public Keys:** By the regularity of the group action, we have $\Pr [g \leftarrow_s \mathbb{G} : \bar{y} = (t \cdot g) \star y] \leq 1/|\mathbb{G}| = \text{negl}(\lambda)$ for any fixed $y, \bar{y} \in \mathcal{X}$ and $t \in G$, and prove ④. For any fixed $g \in \mathbb{G}$ and $t \in \mathbb{G}$, we have $\Pr [y, \bar{y} \leftarrow_s \mathcal{X} : \bar{y} = (t \cdot g) \star y] \leq 1/|\mathcal{X}| = \text{negl}(\lambda)$ that proves ④.

⑤ **Ciphertext Unpredictability under Normal Labels:** For any fixed pp, pk, b, m, c , by the regularity of the group action, we have

$$\Pr [r \leftarrow_s \mathcal{R} : \text{LEnc}(pk, b, m; r) = c] \leq \Pr [r \leftarrow_s \mathbb{G} : r \star x = c] \leq 1/|\mathbb{G}| = \text{negl}(\lambda).$$

⑥ **CPA Security under Normal Labels:** The proof begins with changing public key to a random one by property ①. Then with overwhelming probability, the label is a lossy one of the random public key. Finally, the property of *random ciphertexts under lossy labels* guarantees the CPA security.

⑦ **Ciphertext Randomness in case of Random Messages:** Note that for each component $ct_i = (\mathbf{c}_i, \mathbf{H}(\mathbf{c}'_i) \oplus m_i)$ of the ciphertext, the first part \mathbf{c}_i is uniformly distributed in \mathcal{X} . Moreover, the random message bit m_i is independent of $\mathbf{c}_i, \mathbf{c}'_i$, so $\mathbf{H}(\mathbf{c}'_i) \oplus m$ is uniform and independent of \mathbf{c}_i . \square

Remark 6. When instantiating REGA with CSIDH, any group element $g \in \mathbb{G}$ will be sampled with randomness $r_1 \in \mathcal{R}_1, \dots, r_n \in \mathcal{R}_n$ such that $g = g_1^{r_1} \cdot g_2^{r_2} \cdot \dots \cdot g_n^{r_n}$ and g is described by $g = [r_1, \dots, r_n]$, where $\{g_1, \dots, g_n\}$ is a generating set of \mathbb{G} . In fact, we need to use the exponents r_1, \dots, r_n to implement the group action. In $\text{PAKE}_{\text{ga}}^{\text{RO}}$, the hash function H is implemented with $H(pw) := [r_1, \dots, r_n] \in \mathcal{R}_1 \times \dots \times \mathcal{R}_n$. We stress that knowing the exponents $[r_1, \dots, r_n]$ does not lead to any gain to adversary \mathcal{A} . For example, ① of LPKE_{ga} holds even if $g = [r_1, \dots, r_n]$ is provided or chosen by adversary \mathcal{A} since \mathcal{A} does not know the representation of t and s and the GA-DDH assumption holds. Similarly, for $\text{PAKE}_{\text{ga}}^{\text{QRO}}$, there is no harm to issue the exponent representation of the labels $\{v_i^0 = [r_{i1}^0, \dots, r_{in}^0], v_i^1 = [r_{i1}^1, \dots, r_{in}^1]\}_{i \in [\lambda]}$ in pp to adversary.

5.3 Instantiations of PAKE

With the instantiations of LPKE and LPKE^+ , we obtain four PAKE schemes in ROM and QROM, namely $\text{PAKE}_{\text{lwe}}^{\text{RO}}, \text{PAKE}_{\text{ga}}^{\text{RO}}, \text{PAKE}_{\text{lwe}}^{\text{QRO}}, \text{PAKE}_{\text{ga}}^{\text{QRO}}$.

Corollary 1. *By plugging the LPKE_{lwe} in Fig. 9 (resp. LPKE_{ga} in Fig. 11) scheme in the generic PAKE construction (cf. Fig. 6), we obtain a specific PAKE scheme $\text{PAKE}_{\text{lwe}}^{\text{RO}}$ (resp. $\text{PAKE}_{\text{ga}}^{\text{RO}}$), which UC-realizes the $\mathcal{F}_{\text{pake}}$ functionality based on the LWE (resp. GA-DDH) assumption in ROM.*

Corollary 2. *By plugging the LPKE^+ scheme $\text{LPKE}'_{\text{lwe}}$ in Fig. 10 (resp. LPKE_{ga} in Fig. 11) in the construction of eLPKE in Fig. 8, we obtain eLPKE from the LWE (resp. GA-DDH) assumption in the QROM. Moreover, by plugging the resulting eLPKE scheme in the generic PAKE construction in Fig. 6, we obtain a specific PAKE scheme $\text{PAKE}_{\text{lwe}}^{\text{QRO}}$ (resp. $\text{PAKE}_{\text{ga}}^{\text{QRO}}$), which UC-realizes the $\mathcal{F}_{\text{pake}}$ functionality based on the LWE (resp. GA-DDH) assumption in QROM.*

Acknowledgements. We would like to thank the reviewers for their valuable comments. This work was partially supported by National Natural Science Foundation of China under Grant 61925207 and Grant 62372292, Guangdong Major Project of Basic and Applied Basic Research (2019B030302008), and the National Key R&D Program of China under Grant 2022YFB2701500.

References

1. https://en.wikipedia.org/wiki/Piling-up_lemma
2. Abdalla, M., Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D.: SPHF-friendly non-interactive commitments. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 214–234. Springer, Heidelberg (Dec 2013). https://doi.org/10.1007/978-3-642-42033-7_12
3. Abdalla, M., Eisenhofer, T., Kiltz, E., Kunzweiler, S., Riepel, D.: Password-authenticated key exchange from group actions. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 699–728. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_24
4. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: 28th ACM STOC. pp. 99–108. ACM Press (May 1996). <https://doi.org/10.1145/237814.237838>
5. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 411–439. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_14
6. Ambainis, A., Hamburg, M., Unruh, D.: Quantum security proofs using semi-classical oracles. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 269–295. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_10
7. Azarderakhsh, R., Jao, D., Koziel, B., LeGrow, J.T., Soukharev, V., Taraskin, O.: How not to create an isogeny-based PAKE. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) ACNS 20, Part I. LNCS, vol. 12146, pp. 169–186. Springer, Heidelberg (Oct 2020). https://doi.org/10.1007/978-3-030-57808-4_9
8. Beguinet, H., Chevalier, C., Pointcheval, D., Ricosset, T., Rossi, M.: Get a CAKE: generic transformations from key encapsulation mechanisms to password authenticated key exchanges. In: Tibouchi, M., Wang, X. (eds.) Applied Cryptography and

- Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13906, pp. 516–538. Springer (2023). https://doi.org/10.1007/978-3-031-33491-7_19
9. 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, Heidelberg (May 2000). https://doi.org/10.1007/3-540-45539-6_11
 10. Bellare, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: 1992 IEEE Symposium on Security and Privacy. pp. 72–84. IEEE Computer Society Press (May 1992). <https://doi.org/10.1109/RISP.1992.213269>
 11. Benhamouda, F., Blazy, O., Ducas, L., Quach, W.: Hash proof systems over lattices revisited. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 644–674. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76581-5_22
 12. Beullens, W., Dobson, S., Katsumata, S., Lai, Y.F., Pintore, F.: Group signatures and more from isogenies and lattices: Generic, simple, and efficient. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 95–126. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_4
 13. Bindel, N., Hamburg, M., Hövelmanns, K., Hülsing, A., Persichetti, E.: Tighter proofs of CCA security in the quantum random oracle model. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 61–90. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-36033-7_3
 14. 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, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0_3
 15. Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 89–108. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_7
 16. Canetti, R., Dachman-Soled, D., Vaikuntanathan, V., Wee, H.: Efficient password authenticated key exchange via oblivious transfer. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 449–466. Springer, Heidelberg (May 2012). https://doi.org/10.1007/978-3-642-30057-8_27
 17. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (May 2005). https://doi.org/10.1007/11426639_24
 18. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03332-3_15
 19. Coron, J.S., Patarin, J., Seurin, Y.: The random oracle model and the ideal cipher model are equivalent. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 1–20. Springer, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5_1
 20. Ding, J., Alsayigh, S., Lancrenon, J., RV, S., Snook, M.: Provably secure password authenticated key exchange based on RLWE for the post-quantum world. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 183–204. Springer, Heidelberg (Feb 2017). https://doi.org/10.1007/978-3-319-52153-4_11

21. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Online-extractability in the quantum random-oracle model. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 677–706. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_24
22. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008). <https://doi.org/10.1145/1374376.1374407>
23. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_12
24. Hosoyamada, A., Yasuda, K.: Building quantum-one-way functions from block ciphers: Davies-Meyer and Merkle-Damgård constructions. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part I. LNCS, vol. 11272, pp. 275–304. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03326-2_10
25. Jiang, S., Gong, G., He, J., Nguyen, K., Wang, H.: PAKEs: New framework, new techniques and more efficient lattice-based constructions in the standard model. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 396–427. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45374-9_14
26. Katz, J., Vaikuntanathan, V.: Smooth projective hashing and password-based authenticated key exchange from lattices. In: Matsui, M. (ed.) Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6–10, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5912, pp. 636–652. Springer (2009). https://doi.org/10.1007/978-3-642-10366-7_37
27. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_43
28. McQuoid, I., Xu, J.: An efficient strong asymmetric pake compiler instantiable from group actions. Cryptology ePrint Archive, Paper 2023/1434 (2023), <https://eprint.iacr.org/2023/1434>, <https://eprint.iacr.org/2023/1434>
29. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_41
30. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5_31
31. Quach, W.: UC-secure OT from LWE, revisited. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 192–211. Springer, Heidelberg (Sep 2020). https://doi.org/10.1007/978-3-030-57990-6_10
32. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005). <https://doi.org/10.1145/1060590.1060603>
33. Santos, B.F.D., Gu, Y., Jarecki, S.: Randomized half-ideal cipher on groups with applications to UC (a)PAKE. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 128–156. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_5

34. Shoup, V.: Security analysis of itSPAKE2+. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part III. LNCS, vol. 12552, pp. 31–60. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/978-3-030-64381-2_2
35. Unruh, D.: Revocable quantum timed-release encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 129–146. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_8
36. Zhandry, M.: How to record quantum queries, and applications to quantum indistinguishability. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 239–268. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_9
37. Zhang, J., Yu, Y.: Two-round PAKE from approximate SPH and instantiations from lattices. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 37–67. Springer, Heidelberg (Dec 2017). https://doi.org/10.1007/978-3-319-70700-6_2

Appendix

A More Preliminaries

Given a lattice \mathcal{L} , the smoothing parameter $\eta_\epsilon(\mathcal{L})$ is parameterized by $\epsilon > 0$ and defined as the minimal $s > 0$ such that $\rho_{1/s}(\mathcal{L}^*) \leq 1 + \epsilon$, where \mathcal{L}^* is the dual lattice of \mathcal{L} . Given a matrix \mathbf{A} , define a q -array lattice $\Lambda^\perp(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{x} = 0^n \pmod{q}\}$.

We recall the tail bound about the discrete Gaussian distributions over \mathbb{Z}^m .

Lemma 7 (Tail Bound [27]). *For any $t > 0$, we have $\Pr_{x \leftarrow D_{\mathbb{Z}, \sigma}}[|x| \geq t \cdot \sigma] \leq 2e^{-\frac{t^2}{2}}$ and $\Pr_{\mathbf{x} \leftarrow D_{\mathbb{Z}^m, \sigma}}[\|\mathbf{x}\| \geq \|\mathbf{x}\|_\infty \geq t \cdot \sigma \sqrt{m}] \leq t^m \cdot e^{\frac{m}{2}(1-t^2)}$.*

In particular, for $t \geq \omega(\sqrt{\log \lambda})$, the probability that $|x| \geq t \cdot \sigma$ and $\|\mathbf{x}\|_\infty \geq t \cdot \sigma \sqrt{m}$ is negligible.

Lemma 8 (Piling-up lemma[1]). *For $0 < \mu < 1/2$ and random variables E_1, E_2, \dots, E_ℓ that are i.i.d. to \mathcal{B}_μ we have $\bigoplus_{i=1}^\ell E_i$ follows \mathcal{B}_σ with $\sigma = \frac{1}{2}(1 - (1 - 2\mu)^\ell)$, where \mathcal{B}_x is Bernoulli distribution with parameter x .*

Here we review some definitions of cryptographic group actions from [5].

Definition 7 (Group Action). *A group \mathbb{G} is said to act on a set \mathcal{X} if there is a map $\star : \mathbb{G} \times \mathcal{X} \mapsto \mathcal{X}$ that satisfies the following two properties:*

1. *Identity: If e is the identity element of \mathbb{G} , then for any $x \in \mathcal{X}$, we have $e \star x = x$.*
2. *Compatibility: For any $g, h \in \mathbb{G}$ and any $x \in \mathcal{X}$, we have $(g \cdot h) \star x = g \star (h \star x)$.*

We use the notation $(\mathbb{G}, \mathcal{X}, \star)$ to denote a group action.

Definition 8 (Regular Group Action). *A group action $(\mathbb{G}, \mathcal{X}, \star)$ is said to be regular if it satisfies the following two properties:*

1. *Transitive: for every $x_1, x_2 \in \mathcal{X}$, there exists a group element $g \in \mathbb{G}$ such that $x_2 = g \star x_1$.*
2. *Free: for each group element $g \in \mathbb{G}$, g is the identity if and only if there exists some set element $x \in \mathcal{X}$ such that $x = g \star x$.*

Remark 7. If a group action is regular, then for any $x \in \mathcal{X}$, the map $f_x : g \mapsto g \star x$ defines a bijection between \mathbb{G} and \mathcal{X} .

Now we recall the well-known O2H lemma proposed in [35] and some corollaries. We will use the general version of O2H lemma which is Theorem 3 in [6].

Lemma 9 (O2H Lemma[6]). *Let $S \subseteq X$ be random. Let $G, H : X \rightarrow Y$ be random functions satisfying $\forall x \notin S, G(x) = H(x)$. Let z be a random bitstring. (S, G, H, z may have arbitrary joint distribution.) Let A be quantum oracle algorithm with query number q . Let B^H be an oracle algorithm that on input z does the following: pick $i \leftarrow_{\$} \{1, \dots, q\}$, run $A^H(z)$ until (just before) the i -th query, measure all query input registers in the computational basis, output the set T of measurement outcome. Then,*

$$\left| \Pr [b = 1 : b \leftarrow A^H(z)] - \Pr [b = 1 : b \leftarrow A^G(z)] \right| \leq 2q \sqrt{\Pr [S \cap T \neq \emptyset : T \leftarrow B^H(z)]}.$$

B Construction of Labeled KEM from LPKE via FO Transformation

From a lossy public encryption scheme $\text{LPKE} = (\text{LSetup}, \text{LKeyGen}, \text{IsLossy}, \text{LEnc}, \text{LDec})$, we construct a key encapsulation mechanism $\text{KEM}_{\text{LPKE}}^{\text{FO}} = (\text{Setup}, \text{KeyGen}, \text{Encap}, \text{Decap})$ via FO-transformation. which shares the same $\text{LSetup}, \text{LKeyGen}, \text{IsLossy}$ algorithms with LPKE . Let H_1 and H_2 be two hash functions. Let \mathcal{K}_{kem} be the encapsulation key space.

- $\text{Setup} := \text{LSetup}, \text{KeyGen} := \text{LKeyGen}$.
- $\text{Encap}(b, pk)$: The encapsulation algorithm takes as input a label b and a public key pk . It samples $m \leftarrow_{\$} \mathcal{M}$, computes $c \leftarrow \text{LEnc}(pk, b, m; H_1(m))$, and outputs ciphertext c and encapsulated key $K := H_2(m) \in \mathcal{K}_{\text{kem}}$.
- $\text{Decap}(sk, c)$: The decapsulation algorithm takes as input a secret key sk and ciphertext c . It computes $m \leftarrow \text{LDec}(sk, c)$, checks whether $c = \text{LEnc}(pk, b, m; H_1(m))$. It outputs $K := H_2(m)$ if the check is successful and outputs \perp otherwise.

C Omitted Security Proofs

C.1 Proof of Lemma 5

Proof. We need to show the following via a series of games.

$$(\text{pp}, pk, b, c, K) \approx_c (\text{pp}, pk, b, c' \leftarrow_{\$} \mathcal{CT}, K' \leftarrow_{\$} \mathcal{K}). \quad (3)$$

The proof makes a canonical use of O2H Lemma.

Game 0. In Game 0, adversary \mathcal{A} is given the pp and outputs a public key pk and a label b s.t. $\text{IsLossy}(td, pk, b) = 1$. Then \mathcal{A} is given a pair $(c, K) \leftarrow \text{Encap}(b, pk)$, i.e., $c \leftarrow \text{LEnc}(pk, b, m; H_1(m))$ and $K := H_2(m)$ for $m \leftarrow_{\$} \mathcal{M}$. Meanwhile, \mathcal{A} is also given quantum-access to two random oracles H_1 and H_2 .

Clearly, \mathcal{A} has a view of the left-hand of (3).

Game 1. In Game 1, two new random oracles H'_1 and H'_2 are introduced, and adversary \mathcal{A} can query the random oracles H'_1 and H'_2 instead of H_1 and H_2 . These oracles behave the same as the original oracles H_1 and H_2 for all inputs

except m^* . Upon the query m^* from \mathcal{A} , both H'_1 and H'_2 return randomly chosen values r^* from their own output domains respectively. In Game 1, the ciphertext c is still generated using the randomness $H_1(m^*)$, and the encapsulated key k is still generated using the random oracle $H_2(m^*)$.

In classical random oracle model, Game 1 and Game 0 are indistinguishable if m^* is not queried by adversary \mathcal{A} . However, in quantum random oracle model, \mathcal{A} can perform a superposition query including the information of m^* . So we resort to the O2H Lemma (Lemma 9) and have

$$|\Pr[\mathbf{Game\ 1} \Rightarrow 1] - \Pr[\mathbf{Game\ 0} \Rightarrow 1]| \leq 2q\sqrt{\Pr[m' = m^* \text{ in Game 1.1}]}, \quad (4)$$

where m' and **Game 1.1** are defined in the next game. In fact, **Game 1.1** is introduced for the O2H Lemma to extract the input m' on which H'_1, H'_2 and H_1, H_2 have different functionality.

Game 1.1. Game 1.1 is identical to Game 1, except for the following changes.

In Game 1.1, challenger \mathcal{C} first chooses a random query $j \leftarrow_{\$} [q]$. Upon the j -th query to random oracles H'_1 or H'_2 , \mathcal{C} measures the input register of this query, obtains a value m' and aborts the game. Note that Game 1.1 is introduced for the O2H Lemma and we only care about the probability of event $m' = m^*$. The probability analysis will be deferred to the next game.

Game 1.2. Game 1.2 is the same as Game 1.1, except for the following changes.

In Game 1.2, c and K are generated by $c \leftarrow_{\$} \mathcal{CT}$ and $K \leftarrow_{\$} \mathcal{K}$, rather than $c \leftarrow \text{LEnc}(pk, b, m^*; H_1(m^*))$ and $K := H_2(m^*)$ for $m^* \leftarrow_{\$} \mathcal{M}$. Note that \mathcal{A} can only query random oracles H'_1 and H'_2 in Game 1.2 and $H_1(m^*)$ and $H_2(m^*)$ is uniform and independent of \mathcal{A} 's view. Therefore, $(H_1(m^*), H_2(m^*)) \equiv (r \leftarrow_{\$} \mathcal{R}, K \leftarrow_{\$} \mathcal{K})$, where \equiv denotes identical distribution.

By the property of *random ciphertexts under lossy labels*, c is statistically close to random distribution when $m^* \leftarrow_{\$} \mathcal{M}$. So Game 1.2 is indistinguishable to Game 1.1, and hence

$$|\Pr[m' = m^* \text{ in Game 1.2}] - \Pr[m' = m^* \text{ in Game 1.1}]| \leq \text{negl}(\lambda). \quad (5)$$

We also note that m^* is uniform and independent of other variables in Game 1.2, so we have

$$\Pr[m' = m^* \text{ in Game 1.2}] = 1/|\mathcal{M}| \leq \text{negl}(\lambda). \quad (6)$$

By combining equations (4)(5)(6), we have

$$|\Pr[\mathbf{Game\ 1} \Rightarrow 1] - \Pr[\mathbf{Game\ 0} \Rightarrow 1]| \leq \text{negl}(\lambda).$$

Game 2. Game 2 is almost the same as Game 1.2, except that \mathcal{C} no longer measures the j -th query & abort any more. With a similar argument in Game 1.2, we have

$$|\Pr[\mathbf{Game\ 2} \Rightarrow 1] - \Pr[\mathbf{Game\ 1} \Rightarrow 1]| \leq \text{negl}(\lambda).$$

Note that Game 2 actually provides the right-hand distribution of (3) for \mathcal{A} . By the indistinguishable shifts from Game 0-Game 2, we finish the proof of (3). \square

C.2 Description of Game \mathbf{G}_{11} and Equivalence of \mathbf{G}_{11} and \mathbf{G}_{10}

Game \mathbf{G}_{11} (Integration of Sim with \mathcal{F}_{pake}). \mathbf{G}_{11} is the same as \mathbf{G}_{10} , except that Sim has access to \mathcal{F}_{pake} and uses the replies of \mathcal{F}_{pake} to simulate the generation of session key \mathbf{sKey} . Meanwhile, when \mathcal{Z} sends pw to $\mathbf{C}^{(i)}$ (or $\mathbf{S}^{(j)}$), then $\mathbf{C}^{(i)}$ (or $\mathbf{S}^{(j)}$) sends query (StorePWFile, $\mathbf{C}^{(i)}$, $\mathbf{S}^{(j)}$, pw) or (query (StorePWFile, $\mathbf{S}^{(j)}$, $\mathbf{C}^{(i)}$, pw)) to \mathcal{F}_{pake} . When $\mathbf{C}^{(i)}$ (resp. $\mathbf{S}^{(j)}$) is asked to initialize a new instance iid with $\mathbf{S}^{(j)}$ (resp. $\mathbf{C}^{(i)}$), $\mathbf{C}^{(i)}$ (resp. $\mathbf{S}^{(j)}$) sends query (NewClient, iid , $\mathbf{S}^{(j)}$) (resp. (NewServer, iid' , $\mathbf{C}^{(i)}$)) to \mathcal{F}_{pake} .

We consider the following two cases, covering both passive and active attacks.

Case A: passive attacks. Sim has a different way of generating \mathbf{sKey} .

- To generate session key \mathbf{sKey} for a client instance ($\mathbf{C}^{(i)}$, iid), Sim sends (FreshKey, $\mathbf{C}^{(i)}$, iid , sid) to \mathcal{F}_{pake} , where $sid := \mathbf{C}^{(i)}|\mathbf{S}^{(j)}|pk|c|C|\sigma$. According to the functionality of \mathcal{F}_{pake} , \mathcal{F}_{pake} will choose a random session key \mathbf{sKey} and send (iid , sid , \mathbf{sKey}) to $\mathbf{C}^{(i)}$ and record ($\mathbf{C}^{(i)}$, $\mathbf{S}^{(j)}$, sid , \mathbf{sKey}). Recall in \mathbf{G}_{10} , \mathbf{sKey} is chosen uniformly and stored for ($\mathbf{C}^{(i)}$, iid) by Sim.
- To generate session key \mathbf{sKey} for a server instance ($\mathbf{S}^{(j)}$, iid'), Sim sends (CopyKey, $\mathbf{S}^{(j)}$, iid' , sid) to \mathcal{F}_{pake} , where $sid := \mathbf{C}^{(i)}|\mathbf{S}^{(j)}|pk|c|C|\sigma$. According to the functionality of \mathcal{F}_{pake} , \mathcal{F}_{pake} will retrieve the record ($\mathbf{C}^{(i)}$, $\mathbf{S}^{(j)}$, sid , \mathbf{sKey}) and send (iid' , sid , \mathbf{sKey}) to $\mathbf{S}^{(j)}$. Recall in \mathbf{G}_{10} , \mathbf{sKey} is retrieved from the session key stored for ($\mathbf{C}^{(i)}$, iid) by Sim.

It is easy to see that \mathbf{sKey} follows the uniform distribution and both $\mathbf{C}^{(i)}$ and $\mathbf{S}^{(j)}$ share the same session key \mathbf{sKey} , both in \mathbf{G}_{11} and \mathbf{G}_{10} . Therefore,

$$\Pr[\mathbf{G}_{11} \Rightarrow 1 \text{ in Case A}] = \Pr[\mathbf{G}_{10} \Rightarrow 1 \text{ in Case A}]. \quad (7)$$

Case B: active attacks. Sim does not use pw anymore for the simulation for server instance in Case 2.2 and the simulation of rejection rule (\star) for client instance. Sim will do the simulation in the following way.

Simulation for server instances in Case 2.2. If Case 2.2 happens, i.e., $\exists!(pw', r_H) \in \mathcal{L}_H$ s.t. $\text{IsLossy}(td, \tilde{pk}, r_H) = 0$, Sim first sends (Testpw, $\mathbf{S}^{(j)}$, iid' , pw') to \mathcal{F}_{pake} .

- If \mathcal{F}_{pake} returns “wrong guess”, Sim will compute $c \leftarrow_s \mathcal{CT}$, $r|\sigma|k \leftarrow_s \{0, 1\}^{3\lambda}$.
- If \mathcal{F}_{pake} returns “correct guess”, Sim can extract the true password $pw := pw'$. Then Sim computes (c, C) by $c \leftarrow \text{LEnc}(\tilde{pk}, H(pw), m; H_1(m))$ and $C \leftarrow \text{CEnc}(cpk, pw|\tilde{pk}|c; r)$.

Furthermore, if the server instance later receives a third-round message $\tilde{\sigma}$ satisfying $\tilde{\sigma} = \sigma$, Sim sends query (CorruptKey, $\mathbf{S}^{(j)}$, iid' , $H_3(k|sid)$) to \mathcal{F}_{pake} .

Simulation of rejection rule (\star) for client instances.

Rejection rule: When receiving the second-round message (\tilde{c}, \tilde{C}) , Sim first invokes the decryption algorithm with $pw'|\tilde{pk}'|c' \leftarrow \text{CDec}(csk, \tilde{C})$ and sends

(Testpw, $C^{(i)}, iid, pw'$) to \mathcal{F}_{pake} . If \mathcal{F}_{pake} returns a response of “wrong guess” or $pk'|c' \neq pk|\tilde{c}$ $(\star\star)$

Sim sends (Abort, $C^{(i)}, iid$) to \mathcal{F}_{pake} and terminates the simulation of the instance $(C^{(i)}, iid)$, rather than rejecting (\tilde{c}, \tilde{C}) by setting $sKey := \perp$ as did in G_{10} . Otherwise i.e., $(\star\star)$ does not hold, the simulation of Sim is implemented as follows. If $\exists(m, x) \in \mathcal{L}_{H_1}$ such that $\tilde{c} = \text{Enc}(pk, pw', m; x)$ and $\tilde{C} = \text{Enc}(cpk, pw'|pk|c'; r)$, then Sim sends (CorruptKey, $C^{(i)}, iid, sid, H_3(k|sid)$) to \mathcal{F}_{pake} , where $H_2(m) = r|\sigma|k$ and $sid := C^{(i)}|S^{(j)}|pk|\tilde{c}|\tilde{C}|\sigma$.

Recall that in other cases of active attacks in G_{10} , Sim always rejects the instance with $sKey := \perp$. In contrast, Sim sends (Abort, $C^{(i)}, iid$) or (Abort, $S^{(j)}, iid'$) to \mathcal{F}_{pake} in G_{11} .

Recall that upon a query (Testpw, $C^{(i)}/S^{(j)}, iid, pw'$), \mathcal{F}_{pake} returns “wrong guess” iff $pw' \neq pw$, and returns “correct guess” iff $pw' = pw$.

Meanwhile, in G_{11} upon a query (CorruptKey, $C^{(i)}/S^{(j)}, iid, sid, H_3(k|sid)$), \mathcal{F}_{pake} sets $sKey := H_3(k|sid)$ for instance of $C^{(i)}/S^{(j)}$, while in G_{10} Sim sets $sKey := H_3(k|sid)$ directly.

Moreover, in G_{11} upon a query (Abort, $C^{(i)}/S^{(j)}, iid$), \mathcal{F}_{pake} sets $sKey := \perp$ for instance of $C^{(i)}/S^{(j)}$, while in G_{10} Sim sets $sKey := \perp$ directly.

For Case B, the above analysis shows that \mathcal{Z} has the same view in G_{11} as that in G_{10} . Therefore,

$$\Pr[G_{11} \Rightarrow 1 \text{ in Case B}] = \Pr[G_{10} \Rightarrow 1 \text{ in Case B}]. \quad (8)$$

Consequently, by (7)(8), \mathcal{Z} has the same the view in G_{11} as that in G_{10} , so

$$\Pr[G_{11} \Rightarrow 1] = \Pr[G_{10} \Rightarrow 1].$$

□

C.3 Proof of Theorem 3

Before presenting the formal proof of Theorem 3, we recall the online-extractability technique[21] and a corollary [13] of O2H Lemma.

Definition 9 ([21]). Let $f : \mathcal{X} \times \{0, 1\}^n \rightarrow \mathcal{C}$ be an arbitrary fixed function. Define $\Gamma(f) := \max_{x, c} |\{y | f(x, y) = c\}|$ and $\Gamma'(f) := \max_{x \neq x', y'} |\{y | f(x, y) = f(x', y')\}|$.

Theorem 7 (Summary of Corollary 4.7 in [21]). For any fixed deterministic function $f : \mathcal{X} \times \{0, 1\}^n \rightarrow \mathcal{C}$ and a random oracle RO, there exists an extractable RO-simulator $\mathcal{S} = (\mathcal{S}.RO, \mathcal{S}.E)$ satisfying the following properties.

- $\mathcal{S}.RO$ simulates random oracle RO.
- $\mathcal{S}.E$ extracts element $\hat{x} \in \mathcal{X}$ from element $t \in \mathcal{C}$.

- Let x be a randomized classical value, and W be a quantum register with a state ρ_W^x that depends on x . Let $\delta([x, W]_{\mathcal{G}}, [x, W]_{\mathcal{G}'})$ be the trace distance of the respective density matrices in game \mathcal{G} and in game \mathcal{G}' . For any quantum algorithm \mathcal{A} that first outputs t after $q_1 = \text{poly}(n)$ queries and outputs $x \in \mathcal{X}$ and W after an additional $q_2 = \text{poly}(n)$ queries, if $\Gamma(f)/2^n = \text{negl}(n)$ and $\Gamma(f')/2^n = \text{negl}(n)$, then
 - (a) $\delta([t, x, \text{RO}(x), W]_{\text{Exp}_A^{\text{RO}}}, [t, x, S.\text{RO}(x), W]_{\text{Exp}_A^S}) \leq \text{negl}(n)$,
 - (b) $\Pr [x \neq \hat{x} \wedge f(x, S.\text{RO}(x)) = t \text{ in } \text{Exp}_A^S] \leq \text{negl}(n)$,
 where Exp_A^{RO} and Exp_A^S are described in Fig. 12.

$\text{Exp}_A^{\text{RO}} :$ $(t, st) \leftarrow \mathcal{A}^{\text{RO}}$ $(x, W) \leftarrow \mathcal{A}^{\text{RO}}(st)$ Output (t, x, W)	Exp_A^S $(t, st) \leftarrow \mathcal{A}^{S.\text{RO}}$ $\hat{x} \leftarrow S.E(t)$ $(x, W) \leftarrow \mathcal{A}^{S.\text{RO}}(st)$ Output (t, x, W)
---	--

Fig. 12: The original experiment Exp_A^{RO} executed by \mathcal{A} equipped with RO, and simulated experiment Exp_A^S executed by \mathcal{A} equipped with $S = (S.\text{RO}, S.E)$.

Moreover, we also resort to the following corollary of O2H Lemma [35] to resolve pseudo-randomness of $H_3(k|sid)$.

Lemma 10 (PRF from QROM, Corollary 1 from [13]). *Let $H : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ be a quantum-accessible random oracle. The function $f(k, x) := H(k|x)$ can be used as a quantum-accessible PRF with a key $k \leftarrow_s \mathcal{K}$. More precisely, for any quantum algorithm \mathcal{A} making at most Q queries to H and any number of queries to oracle $f(k, \cdot)$ such that $f(k, x^*)$ is never queried, its advantage satisfies*

$$\text{Adv}_{\text{PRF}}^{\text{ps}}(\mathcal{A}) := \left| \Pr \left[k \leftarrow_s \mathcal{K}, x^* \leftarrow \mathcal{A}^{f(k, \cdot)}; y := f(k, x^*) : \mathcal{A}^{f(k, \cdot)}(x^*, y) \Rightarrow 1 \right] - \Pr \left[k \leftarrow_s \mathcal{K}, x^* \leftarrow \mathcal{A}^{f(k, \cdot)}; y \leftarrow_s \mathcal{Y} : \mathcal{A}^{f(k, \cdot)}(x^*, y) \Rightarrow 1 \right] \right| \leq \frac{2Q}{\sqrt{|\mathcal{K}|}}.$$

Now we are ready to show the security proof of Theorem 3.

Theorem 3. *If eLPKE is an extractable LPKE scheme, PKE is a CCA-secure PKE, and H_1, H_2, H_3 work as quantum-accessible random oracles, then the scheme in Fig. 6 securely emulates $\mathcal{F}_{\text{pake}}$, hence achieving UC security in QROM.*

Proof. The proof outline is similar to that of Theorem 1.

Game \mathcal{G}'_0 . This is the real experiment $\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}$. We have

$$\Pr [\mathbf{Real}_{\mathcal{Z}, \mathcal{A}} \Rightarrow 1] = \Pr [\mathcal{G}'_0 \Rightarrow 1].$$

Game G'_1 (simulations for clients and servers with pw). In this game, we introduce a simulator Sim receives passwords from \mathcal{Z} . Then it simulates the clients and servers to generate transcripts for instances of the PAKE protocol. With the knowledge of passwords, the simulations of the behaviors of all clients and servers are perfect.

Therefore, we have

$$\Pr[G'_1 \Rightarrow 1] = \Pr[G'_0 \Rightarrow 1].$$

Game G'_2 (simulation for crs). In Game G'_2 , simulator Sim' simulates the generation of crs by invoking the setup algorithm. Clearly, the simulation of crs is perfect, so we have

$$\Pr[G'_2 \Rightarrow 1] = \Pr[G'_1 \Rightarrow 1].$$

Game G'_3 (simulation of $r|\sigma|k$ for server instances and simulation of the third-round message σ' for client instances in case of passive attacks).

In Game G'_3 , simulator Sim' is the same as in Game G_3 in the proof of Theorem 1. Note that Lemma 4 is proved in QROM, $(pk, c = \text{eLEnc}(pk, pw, m; H_1(m)), H_2(m))$ works as a CCA-secure KEM where $(pk, sk) \leftarrow \text{eLKeyGen}(pk, pw)$ even if H_1, H_2 are quantum-accessible random oracles. Therefore,

$$|\Pr[G'_3 \Rightarrow 1] - \Pr[G'_2 \Rightarrow 1]| \leq \ell \cdot \text{Adv}_{\text{KEM}}^{\text{CCA-FO}}(\mathcal{B}_{\text{KEM}}) \leq \text{negl}(\lambda).$$

Game G'_4 (simulation of sKey in case of passive attacks). In Game G'_4 , simulator Sim' is the same as in Game G_4 in the proof of Theorem 1. The only difference here is now we use Lemma 10 to show $f_k(x) = H(k|x)$ is a PRF and then $H(k|\text{sid})$ is uniform distributed by the uniformity of k . Therefore,

$$|\Pr[G'_4 \Rightarrow 1] - \Pr[G'_3 \Rightarrow 1]| \leq \ell \cdot \frac{2q}{2^{\lambda/2}} \leq \text{negl}(\lambda).$$

Game G'_5 (simulation for client instances in case of active attacks). In Game G'_5 , simulator Sim' is the same as in Game G_5 in the proof of Theorem 1.

Recall that we introduce a rejection rule in G'_5 : when receiving the second-round message (\tilde{c}, \tilde{C}) , Sim' first invokes the decryption algorithm with $pw'|pk'|c' \leftarrow \text{CDec}(csk, \tilde{C})$. If

$$\underline{pw \neq pw' \text{ or } pk'|c' \neq pk|\tilde{c}}, \quad (\star)$$

then Sim' rejects (\tilde{c}, \tilde{C}) by setting $\text{sKey} := \perp$.

By the correctness of PKE, we have

$$|\Pr[G'_5 \Rightarrow 1] - \Pr[G'_4 \Rightarrow 1]| \leq \text{negl}(\lambda).$$

Game $G'_{5.5}$ (introduce the online-extractable RO simulator \mathcal{S}). In $G'_{5.5}$, Sim' invokes the random oracle simulator $\mathcal{S} = (\mathcal{S}.RO, \mathcal{S}.E)$ as per Theorem 7 to simulate random oracle H_1 and extract the corresponding queries. More precisely, Sim' invokes $\mathcal{S}.RO(x)$ to simulate the random oracle H_1 . Besides, for a client instance $(C^{(i)}, iid)$ that is not linked to any server instance $(S^{(j)}, iid')$

when receiving the second-round message (\tilde{c}, \tilde{C}) , Sim' will additionally invoke $\hat{m} \leftarrow \mathcal{S}.E(\tilde{c})$ to obtain the extracted value \hat{m} .

Obviously, the only difference between $G'_{5.5}$ and G'_5 is the simulation of the random oracle H_1 . Recall that $\tilde{c} = \text{eLEnc}(pk, pw, m; H_1(m))$. We define function $f_{pk, pw}(m; r) := \text{eLEnc}(pk, pw, m; r)$ where $r \in \mathcal{R}$. In $G'_{5.5}$, the invocation of $\mathcal{S}.E(\tilde{c})$ can be regarded as Sim' 's conduction of measurements by interface $\mathcal{S}.E(\tilde{c})$ w.r.t. the function $f_{pk, pw}$.

According to property “⑤*ciphertext unpredictability under normal labels*” and correctness of eLPKE, $\Gamma(f_{pk, pw})/|\mathcal{R}|$ and $\Gamma'(f_{pk, pw})/|\mathcal{R}|$ (cf. Def. 9) are negligible. According to Theorem 7 (a), except with negligible probability, $\mathcal{S}.RO$ perfectly simulates random oracle H_1 , even if $\mathcal{S}.E$ performs the measurement to get \hat{m} . So we have

$$|\Pr [G'_{5.5} \Rightarrow 1] - \Pr [G'_5 \Rightarrow 1]| \leq \text{negl}(\lambda).$$

Game G'_6 (getting rid of sk in simulation for client instances in case of active attacks). In Game G'_6 , simulator Sim' is the same as in Game $G'_{5.5}$, with the exception for the generation of m' during Sim' 's simulations for client instances in case of active attacks.

- For any client instance $(C^{(i)}, iid)$ that is not linked to any server instance $(S^{(j)}, iid')$ when receiving the second-round message (\tilde{c}, \tilde{C}) , we know that (\tilde{c}, \tilde{C}) is NOT generated from any server instance, so it must be forged by adversary \mathcal{A} . When generating m' during the simulation for instance $(C^{(i)}, iid)$, Sim' will not use the decryption algorithm to obtain $m' \leftarrow \text{eLDec}(sk, \tilde{c})$ as did in $G'_{5.5}$. Instead, it will invoke $\hat{m} \leftarrow \mathcal{S}.E(\tilde{c})$ w.r.t. the function $f_{pw, pk}(m; r)$, where pw is the password of $C^{(i)}$ and pk is the first-round message generated by Sim' for $(C^{(i)}, iid)$. If $\hat{m} \neq \perp$, then Sim' sets $m' := \hat{m}$ as the decrypted plaintext. Otherwise Sim' rejects (\tilde{c}, \tilde{C}) by setting $sKey := \perp$.

Let $m' := \text{eLDec}(sk, \tilde{c})$ and $\hat{m} := \mathcal{S}.E(\tilde{c})$. Define bad events as

$$\text{bad}_1 : m' \neq \hat{m} \wedge \text{eLEnc}(pk, pw, m'; H_1(m')) = \tilde{c},$$

$$\text{bad}_2 : m' \neq \hat{m} \wedge \text{eLEnc}(pk, pw, \hat{m}; H_1(\hat{m})) = \tilde{c}.$$

If neither bad_1 nor bad_2 happens, then Game G'_6 is the same as $G'_{5.5}$. According to Theorem 7 (b), bad_1 happens with negligible probability. According to the correctness of eLPKE, bad_2 happens with negligible probability. So we have

$$|\Pr [G'_6 \Rightarrow 1] - \Pr [G'_{5.5} \Rightarrow 1]| \leq \text{negl}(\lambda).$$

We stress that now in G'_6 (and hereafter), Sim' 's simulation for client instances does not need the secret key sk of eLPKE any more, no matter dealing with active attacks or passive attacks. This helps us to proceed to the next game.

Game G'_7 (simulation of generating first-round message pk without pw). In Game G'_7 , simulator Sim' is the same as in Game G'_6 , except for Sim' 's simulation of generating the first-round message pk for client instances.

- For any client instance $(C^{(i)}, iid)$, when generating the first-round message pk , Sim' randomly samples $pk \leftarrow_{\$} \mathcal{PK}$ in G'_7 , rather than invoking $(pk, sk) \leftarrow \text{eLKeyGen}(pw)$ as did in G'_6 .

Due to the “①pseudorandomness of public keys” of eLPKE and by hybrid arguments across the ℓ sessions, we have

$$|\Pr[G'_7 \Rightarrow 1] - \Pr[G'_6 \Rightarrow 1]| \leq \text{negl}(\lambda).$$

Game G'_8 (simulation of generating c in second-round message with the help of Extract algorithm). In Game G'_8 , simulator Sim' is the same as in Game G'_7 , except for Sim' 's simulation of generating c in the second-round message (c, C) for server instances. We consider the following two cases.

Case 1: Passive attacks on Servers. For a server instance $(S^{(j)}, iid')$ that is linked to some client instance $(C^{(i)}, iid)$ when receiving a first-round message pk , simulator Sim' will sample c by $c \leftarrow_{\$} \mathcal{CT}$, rather than computing it with $c \leftarrow \text{eLEnc}(pk, pw, m; H_1(m))$ as did in G'_7 . Note that $r|\sigma|k \leftarrow_{\$} \{0, 1\}^{3\lambda}$ and $C \leftarrow \text{CEnc}(cpk, pw|pk|c; r)$ are still computed in the same way as in G'_7 .

Case 2: Active attacks on Servers. For a server instance $(S^{(j)}, iid')$ that is not linked to any client instance when receiving a first-round message pk , Sim' invokes $pw' \leftarrow \text{Extract}(td, \tilde{pk})$ and checks whether $pw' = pw$ or not.

Case 2.1: $pw' \neq pw$. In this case, Sim' will compute $c \leftarrow_{\$} \mathcal{CT}$, $r|\sigma|k \leftarrow_{\$} \{0, 1\}^{3\lambda}$, rather than computing $c \leftarrow \text{eLEnc}(pk, pw, m; H_1(m))$ and $r|\sigma|k := H_2(m)$ as did in G'_7 .

Case 2.2: $pw' = pw$. In this case, Sim' computes (c, C) just like G'_7 , i.e., $c \leftarrow \text{eLEnc}(pk, pw, m; H_1(m))$ and $C \leftarrow \text{CEnc}(cpk, pw|pk|c; r)$.

In Case 1, pk is random and independent of password pw . According to property “④lossiness of random public keys”, pw is a lossy label under pk . Then according to Lemma 5, the ciphertext $c \leftarrow \text{eLEnc}(pk, pw, m; H_1(m))$ and key $r|\sigma|k := H_2(m)$ are pseudo-random in G'_7 . Therefore, we can replace c and $r|\sigma|k$ with random ones as did in G'_8 in a computationally indistinguishable way.

In Case 2.1, $pw' \neq pw$ implies pw is a lossy label under \tilde{pk} according to “③extractability of the unique normal label” of eLPKE. With a similar argument as Case 1, we can replace c and $r|\sigma|k$ with random ones in a computationally indistinguishable way.

In Case 2.2, simulator Sim' is the same as in Game G'_7 .

By taking into account all the above cases, we have

$$|\Pr[G'_8 \Rightarrow 1] - \Pr[G'_7 \Rightarrow 1]| \leq \text{negl}(\lambda).$$

Game G'_9 (simulation of generating C in the second-round message). In Game G'_9 , simulator Sim' is the same as in Game G'_8 , except for Sim' 's simulation of generating C in the second-round message (c, C) for client instances. We

consider the same cases as in G'_8 . In Case 1 and Case 2.1, Sim' generates C by $C \leftarrow \text{CEnc}(cpk, 0; r)$ rather than $C \leftarrow \text{CEnc}(cpk, pw|pk|c; H_1(m))$ as did in G'_8 .

Note that randomness r is random and independent of $pw|pk|c$. According to the CCA security of PKE and hybrid arguments over the (at most) ℓ ciphertexts, we have

$$|\Pr[G'_9 \Rightarrow 1] - \Pr[G'_8 \Rightarrow 1]| \leq \ell \cdot \text{Adv}_{\text{PKE}}^{\text{cca}}(\mathcal{B}_{\text{PKE}}) \leq \text{negl}(\lambda).$$

Game G'_{10} (simulation of dealing with the third-round message $\tilde{\sigma}$ for server instances in case of active attacks). In Game G'_{10} , simulator Sim' is the same as in Game G'_9 , except for Sim' 's simulation of generating sKey upon receiving the third-round message $\tilde{\sigma}$. We consider the same cases as in G'_9 (and also G'_8). In Case 1 and Case 2.1, Sim' sets $\text{sKey} := \perp$ directly no matter whether $\sigma = \tilde{\sigma}$ or not.

G'_{10} is the same as G'_9 except that $\sigma = \tilde{\sigma}$ happens in these cases in G'_9 . However, σ is uniformly chosen and independent of other variables and hence, \mathcal{A} can present a correct guess of σ with negligible probability. We have

$$|\Pr[G'_{10} \Rightarrow 1] - \Pr[G'_9 \Rightarrow 1]| \leq \text{negl}(\lambda).$$

We stress that Sim does not use pw any more except for the simulation for server instances in Case 2 and the simulation of rejection rule (\star') for client instances.

Game G'_{11} (Integration of Sim' with $\mathcal{F}_{\text{pake}}$). Game G'_{11} is the same as Game G'_{10} , except that Sim' has access to $\mathcal{F}_{\text{pake}}$ and use the replies of $\mathcal{F}_{\text{pake}}$ to simulate the generation of session key sKey . Meanwhile, when \mathcal{Z} sends pw to $C^{(i)}$ (or $S^{(j)}$), then $C^{(i)}$ (or $S^{(j)}$) sends query ($\text{StorePWFile}, C^{(i)}, S^{(j)}, pw$) or (query ($\text{StorePWFile}, S^{(j)}, C^{(i)}, pw$)) to $\mathcal{F}_{\text{pake}}$. When $C^{(i)}$ (resp. $S^{(j)}$) is asked to initialize a new instance iid with $S^{(j)}$ (resp. $C^{(i)}$), $C^{(i)}$ (resp. $S^{(j)}$) sends query ($\text{NewServer}, iid, C^{(i)}$) (resp. ($\text{NewClient}, iid, S^{(j)}$)) to $\mathcal{F}_{\text{pake}}$.

We consider the following two cases, passive attacks and active attacks.

Case A: passive attacks. Sim' has a different way of generating session keys sKey .

- To generate session key sKey for a client instance $(C^{(i)}, iid)$, Sim' sends $(\text{FreshKey}, C^{(i)}, iid, sid)$ to $\mathcal{F}_{\text{pake}}$, where $sid := C^{(i)}|S^{(j)}|pk|c|C|\sigma$. According to the functionality of $\mathcal{F}_{\text{pake}}$, $\mathcal{F}_{\text{pake}}$ will choose a random session key sKey and send (iid, sid, sKey) to $C^{(i)}$ and record $(C^{(i)}, S^{(j)}, sid, \text{sKey})$. Recall in G'_{10} , sKey is chosen uniformly and stored for $(C^{(i)}, iid)$ by Sim' .
- To generate session key sKey for a server instance $(S^{(j)}, iid')$, Sim' sends $(\text{CopyKey}, S^{(j)}, iid', sid)$ to $\mathcal{F}_{\text{pake}}$, where $sid := C^{(i)}|S^{(j)}|pk|c|C|\sigma$. According to the functionality of $\mathcal{F}_{\text{pake}}$, $\mathcal{F}_{\text{pake}}$ will retrieve the record $(C^{(i)}, S^{(j)}, sid, \text{sKey})$ and send (iid', sid, sKey) to $S^{(j)}$. Recall in G'_{10} , sKey is retrieved from the session key stored for $(C^{(i)}, iid)$ by Sim' .

It is easy to see that sKey follows the uniform distribution and both $C^{(i)}$ and $S^{(j)}$ share the same session key sKey , both in G'_{11} and G'_{10} . Therefore,

$$\Pr[G'_{11} \Rightarrow 1 \text{ in Case A}] = \Pr[G'_{10} \Rightarrow 1 \text{ in Case A}]. \quad (9)$$

Case B: active attacks. Sim' does not use pw anymore for the simulation for server instance in Case 2 and the simulation of rejection rule (\star') for client instance. Sim' will do the simulation in the following way.

Simulation for server instances in Case 2. If Case 2 happens, i.e., there exists a server instance $(S^{(j)}, iid')$ that is not linked to any client instance when receiving message \tilde{pk} , Sim' first extracts pw' from \tilde{pk} by Extract algorithm of eLPKE, then it sends $(\text{Testpw}, S^{(j)}, iid', pw')$ to $\mathcal{F}_{\text{pake}}$.

– If $\mathcal{F}_{\text{pake}}$ returns “wrong guess”, Sim' will compute $c \leftarrow_s \mathcal{CT}, r|\sigma|k \leftarrow_s \{0, 1\}^{3\lambda}$.

– If $\mathcal{F}_{\text{pake}}$ returns “correct guess”, Sim' can extract the true password $pw := pw'$. Then Sim' computes (c, C) by $c \leftarrow \text{LEnc}(pk, pw, m; H_1(m))$ and $C \leftarrow \text{CEnc}(cpk, pw|pk|c; r)$.

Furthermore, if the server instance later receives a third-round message $\tilde{\sigma}$ satisfying $\tilde{\sigma} = \sigma$, Sim' sends query $(\text{CorruptKey}, S^{(j)}, iid', H_3(k|sid))$ to $\mathcal{F}_{\text{pake}}$.

Simulation of rejection rule (\star') for client instances.

Rejection rule: When receiving the second-round message (\tilde{c}, \tilde{C}) , Sim' first invokes the decryption algorithm with $pw'|pk'|c' \leftarrow \text{CDec}(esk, \tilde{C})$ and sends $(\text{Testpw}, C^{(i)}, iid, pw')$ to $\mathcal{F}_{\text{pake}}$. If

$$\mathcal{F}_{\text{pake}} \text{ returns a response of “wrong guess” or } pk'|c' \neq pk|\tilde{c} \quad (\star\star')$$

Sim' sends $(\text{Abort}, C^{(i)}, iid)$ to $\mathcal{F}_{\text{pake}}$ and terminates the simulation of the instance $(C^{(i)}, iid)$, rather than rejecting (\tilde{c}, \tilde{C}) by setting $\text{sKey} := \perp$ as did in G'_{10} . Otherwise i.e., $(\star\star')$ does not hold, the simulation of Sim' is implemented as follows. It first executes $\hat{m} \leftarrow \mathcal{S.E}(\tilde{c})$ w.r.t. function $\text{LEnc}_{pk, pw'}(m; r)$. If $\hat{m} \neq \perp$, then Sim' sends $(\text{CorruptKey}, C^{(i)}, iid, sid, H_3(k|sid))$ to $\mathcal{F}_{\text{pake}}$, where $H_2(m) = r|\sigma|k$ and $sid := C^{(i)}|S^{(j)}|pk|\tilde{c}|\sigma$.

Recall that in other cases of active attacks, Sim' always rejects the instance with $\text{sKey} := \perp$ in G'_{10} . In contrast, Sim' sends $(\text{Abort}, C^{(i)}, iid)$ or $(\text{Abort}, S^{(j)}, iid')$ to $\mathcal{F}_{\text{pake}}$ in G'_{11} .

Note that upon a query $(\text{Testpw}, C^{(i)}/S^{(j)}, iid, pw')$, $\mathcal{F}_{\text{pake}}$ returns “wrong guess” if and only if $pw' \neq pw$, and returns “correct guess” if and only if $pw' = pw$.

Meanwhile, in G'_{11} upon a query $(\text{CorruptKey}, C^{(i)}/S^{(j)}, iid, sid, H_3(k|sid))$, $\mathcal{F}_{\text{pake}}$ sets $\text{sKey} := H_3(k|sid)$ for instance of $C^{(i)}/S^{(j)}$, while in G'_{10} Sim' sets $\text{sKey} := H_3(k|sid)$ directly.

Moreover, in G'_{11} upon a query $(\text{Abort}, C^{(i)}/S^{(j)}, iid)$, $\mathcal{F}_{\text{pake}}$ sets $\text{sKey} := \perp$ for instance of $C^{(i)}/S^{(j)}$, while in G'_{10} Sim' sets $\text{sKey} := \perp$ directly.

Therefore, in Case B, the above analysis shows that \mathcal{Z} has same view in G'_{11} as that in G'_{10} . Therefore,

$$\Pr[G'_{11} \Rightarrow 1 \text{ in Case B}] = \Pr[G'_{10} \Rightarrow 1 \text{ in Case B}]. \quad (10)$$

Consequently, by (9)(10), \mathcal{Z} has the same view in G'_{11} as that in G'_{10} , and we have

$$\Pr[G'_{11} \Rightarrow 1] = \Pr[G'_{10} \Rightarrow 1].$$

Now that Sim' completely gets rid of pw in the simulation, G'_{11} is exactly $\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}'}$. By combining those equations across G'_0 - G'_{11} , we know that

$$|\Pr[\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}] - \Pr[\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}'}]| \leq \text{negl}(\lambda).$$

See Fig. 7. for the pseudo-code of the final simulator Sim' in the ideal world. \square

C.4 Proof of Theorem 5

Our LPKE⁺ scheme is adapted from the dual encryption system in [31], but integrated with the novel round function from [11].

Round function in [11]. Recall that Regev's encryption scheme implicitly uses a round function r^\sharp which is defined by

$$r^\sharp(x) := \begin{cases} 1 & \text{if } x \in [-q/4, q/4) \\ 0 & \text{otherwise.} \end{cases} \quad \text{for } x \in [-q/2, q/2].$$

Consider the q -periodic function defined on $[-q/2, q/2]$ by:

$$r^b(x) = \begin{cases} \frac{1}{2T} & \text{if } |x| \leq T \\ 0 & \text{otherwise} \end{cases} \quad \text{for } x \in [-q/2, q/2].$$

Define a new (randomized) rounding function R such that for all $x \in \mathbb{R}$,

$$\Pr[R(x) = 1] := (r^\sharp \odot r^b)(x) := \int_{-q/2}^{q/2} r^\sharp(u) \cdot r^b(x - u) du,$$

where \odot corresponds to the convolution of q -periodic functions.

By Lemma 4.1 and Theorem 4.5 in [11], the rounding function R satisfies the following two properties.

Statistical Correctness: Given a fixed $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, let $r \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A}))$ for some $\epsilon = \text{negl}(n)$. For any $\mathbf{p} = \mathbf{A}^T \mathbf{s} + \mathbf{e}$, where $\|\mathbf{e}\| \leq B = 2t\sqrt{m}$, if $trm/q = \text{negl}(n)$, and $T/q = \text{negl}(n)$, then

$$\Pr_{\mathbf{r} \leftarrow D_{\mathbb{Z}, r}^m} [R(\mathbf{s}^T \mathbf{A} \mathbf{r}) = R(\mathbf{p}^T \mathbf{r})] \geq 1 - \text{negl}(n).$$

Here we note that q must be a super-polynomial to make trm/q and T/q negligible.

Approximate Smoothness: Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with $m = \Theta(n \log q)$, $r \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A}))$ for some $\epsilon = \text{negl}(n)$, and fix $\mathbf{c} \in \mathbb{Z}_q^n$. Let $B' = q/\Theta(\sqrt{m})$. Suppose also that

parameters T, N, δ , and k satisfy $\delta > \frac{q\sqrt{m}}{r}$, $N = \frac{kq}{T}$, and $\delta N^2 < B'$. Then for all $\mathbf{p} \in \mathbb{Z}_q^m$ such that $\text{dis}(\mathbf{p}, \Lambda(\mathbf{A})) \geq B'$, we have

$$\left| \Pr_{\mathbf{r} \leftarrow D_{\mathbb{Z}, r}^n} [R(\mathbf{p}^T \mathbf{r}) = 1 \mid \mathbf{A}^T \mathbf{r} = \mathbf{c}] - 1/2 \right| \leq \frac{1}{6} + \text{negl}(n),$$

where $\text{dis}(\mathbf{p}, \Lambda(\mathbf{A})) := \min_{\mathbf{x} \in \Lambda(\mathbf{A})} \|\mathbf{x} - \mathbf{p}\|$.

The LPKE⁺ scheme. We present the LPKE⁺ scheme from LWE in Fig. 10.

Remark 8. Note that the TrapGen and IsLossy algorithms remain efficient even for super-polynomial LWE modulus q . Additional details are available in Lemma 2.6 and Lemma 3.5 in [31].

Theorem 5. *If $t > n \log n$, $trm/q = \text{negl}(\lambda)$, $T/q = \text{negl}(\lambda)$, $m = \Theta(n \log n)$, $\frac{\sqrt{m}}{r} (\frac{nq}{T})^2 < \Theta(\sqrt{m})$ then The LPKE_{lwe}' scheme in Fig. 10 is a LPKE⁺ scheme based on the LWE_{n,q,m,D_{\mathbb{Z}^m,t}} assumption.*

Proof. We provide a concise overview proof of these properties.

Correctness: According to [11], if $t > n \log n$, $trm/q = \text{negl}(\lambda)$, $T/q = \text{negl}(\lambda)$, then the statistical correctness of round function R holds. Then the correctness of LPKE_{lwe}' follows.

① **Pseudorandomness of Public Key:** This property is directly derived from the LWE assumption.

② **Random Ciphertexts under Lossy Labels:** Recall that $\text{IsLossy}(td, pk = \mathbf{p}, \mathbf{v}) = 1$ implies that $\text{dis}(\mathbf{p}, \Lambda(A)) > q/8\sqrt{m}$. Due to the approximate smoothness of the rounding function R , $R(\mathbf{p}^T \mathbf{e}_{ij})$ is 1/6-close to uniform even given $\mathbf{Ae}_{ij} = \mathbf{c}_{ij}$. Consequently, given $\{\mathbf{Ae}_{ij} = \mathbf{c}_{ij}\}_{j \in [\lambda]}$, the bit $\bigoplus_{j \in [\lambda]} R(\mathbf{p}^T \mathbf{e}_{ij})$ is

statistically close to uniform distribution over $\{0, 1\}$, thus hiding the message $m_i \in \{0, 1\}$ almost perfectly.

③ **Uniqueness of Normal Labels among Polynomial-Size Set:** Suppose there are two random labels, \mathbf{v}_0 and \mathbf{v}_1 , such that $\text{IsLossy}(td, pk, \mathbf{v}_0) = \text{IsLossy}(td, pk, \mathbf{v}_1) = 0$. This implies that both $pk + \mathbf{v}_0$ and $pk + \mathbf{v}_1$ are close to the lattice $\Lambda(\mathbf{A})$ within $q/8\sqrt{m}$. Due to the triangular inequality, the point $\mathbf{v}_0 - \mathbf{v}_1$ is within a distance of $q/4\sqrt{m}$ from $\Lambda(\mathbf{A})$. Importantly, the labels \mathbf{v}_0 and \mathbf{v}_1 are selected uniformly at random, meaning the event that $\mathbf{v}_0 - \mathbf{v}_1$ is close to the lattice $\Lambda(\mathbf{A})$ occurs only with negligible probability. Along with the union bound on $Q = \text{poly}(n)$ random labels, the probability that more than one label satisfy $\text{IsLossy}(td, pk, \mathbf{v}) = 0$ is negligible.

④ **Lossiness of Random Public Keys:** This fact is proven in Lemma 3.3 of [31]. In summary, the set of points within a distance of $q/4$ (using the ℓ_∞ norm) from $\Lambda(\mathbf{A})$ has a size of at most $q^n \cdot (q/2)^m$. Given that $m \geq 2n \log q$, the probability of $\mathbf{p} \in \mathbb{Z}_q^m$ falling within this set is at most q^{-n} .

⑤ **Ciphertext Unpredictability under Normal Labels:** The argument is almost the same as the proof in Section 5.1.

⑥ **CPA Security with Normal Labels:** The argument is almost the same with the proof in Section 5.1.

⑦ **Ciphertext randomness with random message:** Recall that the ciphertext $ct = (\{\mathbf{c}_{ij}\}_{i,j \in [\lambda]}, \{\beta_i\}_{i \in [\lambda]})$. The first part $\{\mathbf{c}_{ij}\}_{i,j \in [\lambda]}$ is uniformly distributed due to the leftover hash lemma. Moreover, the computation of the first part does not involve m , and hence is independent of m . Finally, the m_i is uniformly chosen from $\{0,1\}$ and independent of \mathbf{p} and \mathbf{r}_{ij} , so β_i is also uniform over $\{0,1\}$ and independent of \mathbf{c}_{ij} . Therefore, the ciphertext is uniform as long as the plaintext is. \square

Table of Contents

Universal Composable Password Authenticated Key Exchange for the Post-Quantum World	1
<i>You Lyu^{id}, Shengli Liu^{(✉)id}, and Shuai Han^{id}</i>	
1 Introduction	1
2 Preliminaries	6
2.1 Hardness Assumptions	7
2.2 UC Framework for PAKE	10
2.3 ROM vs. QROM	10
3 PAKE from Basic LPKE in ROM	11
3.1 Basic Lossy Public Key Encryption (LPKE)	11
3.2 Construction of PAKE from Basic LPKE in ROM	13
4 PAKE from Extractable LPKE in QROM	19
4.1 Definition of Extractable LPKE (eLPKE)	21
4.2 Construction of eLPKE from LPKE ⁺	22
4.3 Construction of PAKE from eLPKE in QROM	24
5 Instantiations	24
5.1 LPKE and LPKE ⁺ Schemes from LWE	25
5.2 LPKE and LPKE ⁺ Scheme from Group Actions	28
5.3 Instantiations of PAKE	29
A More Preliminaries	34
B Construction of Labeled KEM from LPKE via FO Transformation	35
C Omitted Security Proofs	35
C.1 Proof of Lemma 5	35
C.2 Description of Game G_{11} and Equivalence of G_{11} and G_{10}	37
C.3 Proof of Theorem 3	38
C.4 Proof of Theorem 5	45