

# Tighter Security for Generic Authenticated Key Exchange in the QROM\*

Jiaxin Pan <sup>1</sup>

Benedikt Wagner <sup>2,3</sup>

Runzhi Zeng <sup>1</sup>

September 14, 2023

<sup>1</sup> Department of Mathematical Sciences,  
NTNU – Norwegian University of Science and Technology, Trondheim, Norway

[jiaxin.pan@ntnu.no](mailto:jiaxin.pan@ntnu.no), [runzhi.zeng@ntnu.no](mailto:runzhi.zeng@ntnu.no)

<sup>2</sup> CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

[benedikt.wagner@cispa.de](mailto:benedikt.wagner@cispa.de)

<sup>3</sup> Saarland University, Saarbrücken, Germany

## Abstract

We give a tighter security proof for authenticated key exchange (AKE) protocols that are generically constructed from key encapsulation mechanisms (KEMs) in the quantum random oracle model (QROM). Previous works (Hövelmanns et al., PKC 2020) gave reductions for such a KEM-based AKE protocol in the QROM to the underlying primitives with square-root loss and a security loss in the number of users and total sessions. Our proof is much tighter and does not have square-root loss. Namely, it only loses a factor depending on the number of users, not on the number of sessions.

Our main enabler is a new variant of lossy encryption which we call parameter lossy encryption. In this variant, there are not only lossy public keys but also lossy system parameters. This allows us to embed a computational assumption into the system parameters, and the lossy public keys are statistically close to the normal public keys. Combining with the Fujisaki-Okamoto transformation, we obtain the *first* tightly IND-CCA secure KEM in the QROM in a multi-user (without corruption), multi-challenge setting.

Finally, we show that a multi-user, multi-challenge KEM implies a square-root-tight and session-tight AKE protocol in the QROM. By implementing the parameter lossy encryption tightly from lattices, we obtain the *first* square-root-tight and session-tight AKE from lattices in the QROM.

**Keywords:** Authenticated key exchange, key encapsulation mechanism, quantum random oracle model, tight security, lattices

---

\*Supported by the Research Council of Norway under Project No. 324235.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Our Contributions . . . . .	3
1.2	More Related Work . . . . .	4
<b>2</b>	<b>Preliminaries</b>	<b>5</b>
2.1	Quantum Random Oracle Model . . . . .	5
2.2	Background about Lattices . . . . .	6
<b>3</b>	<b>Parameter Lossy Encryption</b>	<b>7</b>
3.1	Parameter Lossy Encryption . . . . .	8
3.2	Parameter Lossy Encryption from Lattices . . . . .	8
3.3	Lossy Encryption from Lattices . . . . .	11
<b>4</b>	<b>CCA Secure KEMs from (Parameter) Lossy Encryption</b>	<b>12</b>
4.1	MC-IND-CCA Secure KEM from Lossy Encryption . . . . .	12
4.2	MUC-IND-CCA Secure KEM from Parameter Lossy Encryption . . . . .	13
<b>5</b>	<b>Security Model for AKE</b>	<b>20</b>
<b>6</b>	<b>Session-tight AKE protocol</b>	<b>23</b>

# 1 Introduction

Authenticated key exchange (AKE) is a fundamental primitive in cryptography. An AKE allows to establish a session key between two users. In combination with symmetric-key primitives, this allows to establish a secure channel. Many well-known AKE protocols (such as SIGMA [25] and HMQV [26]) are constructed based on the Diffie-Hellman assumption. Contrary to that, we focus on quantum-safe AKE in this paper.

KEM/PKE-BASED AKE. It is known that AKE protocols can be constructed generically from key encapsulation mechanisms (KEMs) or public-key encryption (PKE) (e.g., [9, 10, 19]). In particular, a quantum-safe AKE can be constructed from a quantum-safe KEM. One main advantage of such KEM-based AKE protocols is that they do not require any digital signature to authenticate the protocol transcripts explicitly. Considering the (in)efficiency of quantum-safe signature schemes, this avoids a significant overhead.

AKE IN THE QROM AND ITS NON-TIGHTNESS. The well-established random oracle model (ROM) [4] idealizes hash functions and is used to prove the security of many practical cryptographic protocols, including the aforementioned generic KEM-based AKE protocols. For quantum adversaries, however, it is more realistic to assume that they can run an “offline” primitive such as a hash function in a quantum manner. To model this, the quantum (accessible) random oracle model (QROM) has been introduced in [7]. In the QROM, a quantum adversary can query the random oracle on arbitrary superpositions. This makes it difficult to use many of the proof techniques applied in the classical ROM. In addition, it introduces a large security loss. We take the existing KEM-based AKE protocol [19] in the QROM as an example. Its security bound is<sup>1</sup>

$$\Theta(S^2 + S \cdot N) \cdot \left( \varepsilon_{\text{IND-CPA}} + \sqrt{Q \cdot \varepsilon_{\text{IND-CPA}}} \right), \quad (1)$$

where  $S$ ,  $N$ , and  $Q$  are the numbers of total sessions, users, and random oracle queries, respectively, and  $\varepsilon_{\text{IND-CPA}}$  is the advantage of breaking the underlying IND-CPA secure PKE. This is the only known bound in the QROM. Regarding the level of IND-CPA security, especially the square-root loss (i.e., the term  $\sqrt{\varepsilon_{\text{IND-CPA}}}$ ) is undesirable. This square-root loss results from the use of the so-called oneway-to-hiding strategy in the QROM [2]. In practice, the PKE would be implemented by a standardized scheme with a 128-bit security guarantee. Even without counting other non-tight terms, the resulting AKE is only guaranteed to have 64-bit security, which is not a reasonable security margin. Even worse, for today’s applications, it is easy to have  $S = 2^{30}$  and  $N = 2^{30}$ . Hence, the security bound given by Equation (1) provides almost no security guarantee given such a PKE implementation.

In this paper, our goal is to minimize the security loss of AKE protocols in the QROM. We emphasize that there is no known tightly secure AKE protocol in the QROM, and most tightly secure AKE protocols (e.g., [14, 20, 15]) are based on variants of Diffie-Hellman assumptions, which are not quantum-safe.

## 1.1 Our Contributions

We propose a tighter proof for KEM-based AKE protocols in the QROM. Our proof does not have square-root loss and is tight with respect to the number of total sessions. Assuming a multi-challenge IND-CCA secure (MC-CCA) KEM (with advantage denoted as  $\varepsilon_{\text{MC-CCA}}$ ) and a multi-user, multi-challenge IND-CCA (MUC-CCA) secure KEM (with advantage denoted as  $\varepsilon_{\text{MUC-CCA}}$ ), our security bound for AKE in the QROM is

$$\Theta(N) \cdot \varepsilon_{\text{MC-CCA}} + \Theta(1) \cdot \varepsilon_{\text{MUC-CCA}}. \quad (2)$$

The concrete bound is given in Theorem 6.1. Here, the multi-user security provides an adversary with multiple users’ public keys but does not allow corruption for any of the corresponding secret keys. The multi-challenge security allows an adversary to ask for multiple challenge ciphertexts under any user.

We also show that MC-CCA and MUC-CCA can be efficiently achieved either tightly or almost tightly<sup>2</sup> from the Decisional Learning with Errors (LWE) assumption. In combination, our bound for the resulting AKE protocol is

$$\Theta(N) \cdot \Theta(\lambda) \cdot \varepsilon_{\text{LWE}} + \Theta(\lambda) \cdot \varepsilon_{\text{LWE}}, \quad (3)$$

<sup>1</sup>For all security bounds in this section, we ignore all additive and negligible statistical terms.

<sup>2</sup>This is a relaxed tightness notion from [8] where security loss is at most linear in the security parameter  $\lambda$ .

where  $\lambda$  is the security parameter, and  $\varepsilon_{\text{LWE}}$  is the advantage against the LWE assumption (cf. Corollary 6.2). Our AKE model is essentially the Bellare-Rogaway model [5], and additionally, it captures the key-compromise-impersonation (KCI) attacks.

PARAMETER LOSSY ENCRYPTION. Our technical tool is a more expressive and fine-grained variant of lossy encryption which we call parameter lossy encryption (PLE). (Slightly) different from the original notion of lossy encryption [16], the PLE has a system parameter that is shared among many users in the system, and each user has an independent public key. Both public keys and system parameters have a lossy mode. Under such lossy parameters and lossy public keys, ciphertexts statistically hide the encrypted messages. This enables a tight security proof as follows: Under the normal parameters, lossy public keys are statistically close to the normal ones. Further, lossy parameters are computationally indistinguishable from normal parameters. In combination, this allows us to switch from the normal to the lossy setting with a security loss that is independent of the number of keys.

TIGHT SECURITY IN THE QROM FROM PLE. Separating the system parameter from public keys can improve efficiency, since multiple users can share the same system parameter, instead of generating an independent parameter that is in a user’s public key. This can largely improve the communication complexity of a KEM-based AKE, where an initiator will generate an ephemeral public key and send it to the responder (cf. Figure 13). Moreover, separating the system parameters is important for tightness. For instance, a PLE scheme immediately implies a multi-user, multi-challenge IND-CPA KEM tightly without random oracles, while the (original) lossy encryption can only tightly imply multi-challenge IND-CPA KEM. This is because the original lossy encryption requires computational assumptions to switch user public keys to lossy one-by-one, which introduces a security loss linear in the number of users. More importantly, the aforementioned fine-grained separation is very useful to remove the square-root loss in the QROM. When we apply the Fujisaki-Okamoto transformation [11, 17] to achieve IND-CCA security, the only step that needs computational assumptions is switching normal system parameters to lossy ones, and all the other proof steps are merely statistical. The parameter-switching step does not involve random oracles. When the oneway-to-hiding lemma [2] is used, the square root function is only applied on a purely statistical term and does not affect the security loss with respect to computational assumptions. Hence, this gives us the *first* tightly secure multi-user, multi-challenge IND-CCA KEM in the QROM, which solves the open problem in [19] about a root-tight proof of IND-CCA security. We note that the work of Pan and Zeng [32] tightly implied a PKE with the same security in the classical ROM, yet it is not clear how to transform it in the QROM, since they used a lot of reprogramming.

PARAMETER LOSSY ENCRYPTION FROM LATTICES. Finally, we propose a tight construction of PLE from the LWE assumption. Our construction extends the dual Regev encryption [33, 13] with lossy LWE matrices [27]. Combining with the aforementioned generic constructions, we obtain

- the first lattice-based AKE protocol in the QROM that does not have square-root security loss and is tightly secure with respect to the number of total sessions. It is not tight with respect to the number of users;
- the first tightly IND-CCA secure lattice-based KEM in the multi-user, multi-challenge setting and in the QROM.

Both results provide new insights on minimizing the security loss in the QROM, namely, PLE is a useful tool to tighten security loss in the QROM. It may be useful for future applications.

OPEN PROBLEMS. We view avoiding the square-root loss and loss concerning the number of total sessions as an important step towards tightly secure AKE in the QROM. It would be interesting to extend our techniques to construct a tightly secure AKE in the QROM. Another interesting open problem is how to construct our parameter lossy encryption from other quantum-safe assumptions, e.g., module-LWE.

## 1.2 More Related Work

The work of Fujioka et al. (FSXY) [9] constructed AKE generically from KEMs in the standard model. One may think that it is secure in the QROM with the same proof. However, as pointed out by Hövelmanns et al. [19], FSXY has two major drawbacks: First, it requires perfect correctness, which makes it hard to instantiate with lattices. Second, it lacks simplicity, making it overly complicated and very inefficient. Moreover, the security loss of FSXY is  $\Theta(N^2S)$  which is much larger than ours. Another work on AKE protocols in the QROM is due to Xue et al. [36] which constructed AKE from

commutative supersingular isogenies. Similar to the work of Hövelmanns et al., it contains square-root-loss and depends on both the number of users and sessions. Hence, the work of Hövelmanns et al. is the most representative for our discussion. We note a very recent work on lattice-based tightly secure AKE [30] in the classical ROM, but extending it to the QROM is not trivial, since it seems difficult to extend the programming techniques of [30] to the QROM.

## 2 Preliminaries

For an integer  $n$ , we define the notation  $[n] := \{1, \dots, n\}$ . Let  $\mathcal{X}$  and  $\mathcal{Y}$  be two finite sets. The notation  $x \xleftarrow{\$} \mathcal{X}$  denotes sampling an element  $x$  from  $\mathcal{X}$  uniformly at random. Let  $\mathcal{A}$  be an algorithm. If  $\mathcal{A}$  is probabilistic, then  $y \leftarrow \mathcal{A}(x)$  means that the variable  $y$  is assigned to the output of  $\mathcal{A}$  on input  $x$ . If  $\mathcal{A}$  is deterministic, then we may write  $y := \mathcal{A}(x)$ . We write  $\mathcal{A}^{\mathcal{O}}$  to indicate that  $\mathcal{A}$  has classical access to oracle  $\mathcal{O}$ , and  $\mathcal{A}^{|\mathcal{O}\rangle}$  to indicate that  $\mathcal{A}$  has quantum access to oracle  $\mathcal{O}$ . All algorithms (including adversaries) in this paper are probabilistic polynomial-time (PPT), unless we state it otherwise. We use code-based games [6] to define and prove security. We implicitly assume that Boolean flags are initialized to false, numerical types are initialized to 0, sets and ordered lists are initialized to  $\emptyset$ , and strings are initialized to the empty string  $\epsilon$ . The notation  $\Pr[\mathbf{G}^{\mathcal{A}} \Rightarrow 1]$  denotes the probability that the final output  $\mathbf{G}^{\mathcal{A}}$  of game  $\mathbf{G}$  running an adversary  $\mathcal{A}$  is 1. Let  $\text{Ev}$  be an (classical) event. We write  $\Pr[\text{Ev} : \mathbf{G}]$  to denote the probability that  $\text{Ev}$  occurs during the game  $\mathbf{G}$ . In our security notions throughout the paper, we let  $N, S$  be numbers of users and challenges, respectively, which are assumed to be polynomial in  $\lambda$ .

### 2.1 Quantum Random Oracle Model

In the quantum random oracle model (QROM), some hash functions are modelled as publicly quantum-accessible random oracles (see [7] for more details). Unlike the classical random oracle model, the efficient reduction algorithm in the QROM cannot use lazy sampling to simulate quantum random oracles (QROs). In this paper, we do not specify the way for reduction algorithms to simulate QROs. Following the convention in [21, 22, 24, 34], we assume that reduction algorithms (i.e., game simulators) have access to some internal quantum random oracles (which can be instantiated by quantum-secure pseudo-random functions or real-world hash functions [24, 34]). Lemma 2.1 gives a probabilistic bound for an adversary  $\mathcal{A}$  (at most  $q$  queries to  $|\mathcal{O}\rangle$ ) to distinguish whether it is interacting with random oracle  $\mathcal{O}_0$  or interacting with random oracle  $\mathcal{O}_1$ , where  $\mathcal{O}_0 \setminus \mathcal{S} = \mathcal{O}_1 \setminus \mathcal{S}$ . If  $\mathcal{A}$  can distinguish, then Lemma 2.1 states that there exists an PPT reduction EXT that randomly measures  $\mathcal{A}$ 's QRO queries and outputs an element  $x \in \mathcal{S}$ .

**Lemma 2.1** (OW2H, probabilities [2]). *Let  $\mathcal{X}, \mathcal{Y}$ , and  $\mathcal{S} \subseteq \mathcal{X}$  be sets. Let  $\mathcal{O}_0, \mathcal{O}_1 : \mathcal{X} \rightarrow \mathcal{Y}$  be random functions satisfying  $\forall x \notin \mathcal{S}, \mathcal{O}_0(x) = \mathcal{O}_1(x)$ . Let  $\text{inp}$  be some bitstring.  $(\mathcal{S}, \mathcal{O}_0, \mathcal{O}_1, \text{inp})$  may have arbitrary joint distribution. Let  $\mathcal{A}$  be an adversary issuing at most  $q$  quantum-superposition queries to random oracle and, on input  $\text{inp}$ , it outputs either 0 or 1. Let  $\text{EXT}^{\mathcal{O}}$  ( $\mathcal{O} = \mathcal{O}_0$  or  $\mathcal{O}_1$ ) be a quantum algorithm that on input  $\text{inp}$  does the following: It picks  $i^* \xleftarrow{\$} [q]$ , runs  $\mathcal{A}^{|\mathcal{O}\rangle}(\text{inp})$  until  $i^*$ th query (denoted as  $|\phi\rangle$ ) to  $\mathcal{O}$ , and returns  $x' := \text{MEASURE}(|\phi\rangle)$ . Then we have*

$$\begin{aligned} & \left| \Pr[1 \leftarrow \mathcal{A}^{|\mathcal{O}_0\rangle}(\text{inp})] - \Pr[1 \leftarrow \mathcal{A}^{|\mathcal{O}_1\rangle}(\text{inp})] \right| \\ & \leq 2q \sqrt{\Pr[x' \in \mathcal{S} : x' \leftarrow \text{EXT}^{\mathcal{A}, |\mathcal{O}_1\rangle}(\text{inp})]} \end{aligned}$$

We consider a special case of Lemma 2.1. Let  $\mathcal{S}$  in Lemma 2.1 be a randomly generated set and independent of  $\text{inp}$ . Then we have the following corollary. The proof is straight-forward since the  $\mathcal{S}$  is independently random, the probability that EXT finds an element in  $\mathcal{S}$  is the uniform probability  $\frac{|\mathcal{S}|}{|\mathcal{X}|}$ .

**Corollary 2.2** *With the same notations and assumptions in Lemma 2.1, if  $\mathcal{S}$  is random set generated at independently and uniformly random, then we have*

$$\left| \Pr[1 \leftarrow \mathcal{A}^{|\mathcal{O}_0\rangle}(\text{inp})] - \Pr[1 \leftarrow \mathcal{A}^{|\mathcal{O}_1\rangle}(\text{inp})] \right| \leq 2q \sqrt{|\mathcal{S}|/|\mathcal{X}|}$$

Lemma 2.3 gives a probabilistic bound for an adversary (has quantum access to oracles) to distinguish  $h(k, \cdot)$  and  $h'$ , where  $k$  is secret,  $h$  and  $h'$  are QRO and have the same image. When the image set is large enough, the adversary cannot distinguish these two oracles, unless it ‘‘queries’’ the oracle on  $k$ .

<p><b>Game</b> <math>\text{GDPB}_{\lambda,b}^A</math></p> <p>01 <math>(\lambda_x)_{x \in \mathcal{X}} \leftarrow \mathcal{A}</math></p> <p>02 <b>if</b> <math>\exists x \in \mathcal{X}</math> s.t. <math>\lambda_x &gt; \lambda</math>: <b>return</b> 0</p> <p>03 <b>if</b> <math>b = 0</math></p> <p>04     Define <math>F := 0</math></p> <p>05 <b>else for</b> <math>x \in \mathcal{X}</math></p> <p>06     <math>F(x) \leftarrow B_{\lambda_x}</math></p> <p>07 <math>b' \leftarrow \mathcal{A}^F</math></p> <p>08 <b>return</b> <math>b'</math></p>
---

Figure 1: Game  $\text{GDPB}_{\lambda,b}^A$  used in Lemma 2.4.

**Lemma 2.3** ([34]). *Let  $s$  be an integer. Let  $h : \{0, 1\}^s \times \mathcal{X} \rightarrow \mathcal{Y}$  and  $h' : \mathcal{X} \rightarrow \mathcal{Y}$  be two independent random oracles. If an unbounded time quantum adversary  $\mathcal{A}$  that queries  $H$  at most  $q_H$  times, then we have*

$$|\Pr[1 \leftarrow \mathcal{A}^{(h), |h(k, \cdot)}(\cdot) \mid k \leftarrow \{0, 1\}^s] - \Pr[1 \leftarrow \mathcal{A}^{(h), |h'}(\cdot)]| \leq 2q_H \cdot 2^{-s/2}$$

We also need the following lemma to handle PKE schemes with imperfect correctness (Definition 3.1). Let  $B_\lambda$  be the Bernoulli distribution (i.e.,  $\Pr[b = 1] = \lambda$  for the bit  $b \leftarrow B_\lambda$ ). Roughly speaking, for any unbounded and quantum adversary  $\mathcal{A}$ , Lemma 2.4 bounds  $\mathcal{A}$ 's advantage in distinguishing whether it is interacting with a constant function or a function that follows the Bernoulli distribution  $B_\lambda$ . We call such a distinguishing problem as Generic quantum Distinguishing Problem with Bounded probabilities (GDPB).

**Lemma 2.4** (GDPB [19]). *Let  $\mathcal{X}$  be a finite set, and let  $\lambda \in [0, 1]$ . Then, for any unbounded and quantum algorithm  $\mathcal{A}$  issuing at most  $q$  quantum queries,*

$$\left| \Pr[\text{GDPB}_{\lambda,0}^A \Rightarrow 1] - \Pr[\text{GDPB}_{\lambda,1}^A \Rightarrow 1] \right| \leq 8(q+1)^2 \lambda,$$

where games  $\text{GDPB}_{\lambda,b}^A$  are defined in Figure 1.

## 2.2 Background about Lattices

In this section, we recall the LWE assumption and some well-known facts about Gaussians [29, 12], and the lossy LWE technique and a generalized leftover hash lemma [1, 23]. First, we recall the LWE assumption.

**Definition 2.5** (LWE Assumption). *Let  $n, m$  be positive integers,  $q$  be a prime. Let  $\chi$  be a distribution over  $\mathbb{Z}$ . All of these are implicitly parameterized by the security parameter  $\lambda$ . We say that the  $\text{LWE}_{n,m,q,\chi}$  assumption holds, if for any algorithm  $\mathcal{B}$ , the following advantage is negligible in  $\lambda$ :*

$$\begin{aligned} \text{Adv}^{\text{LWE}_{n,m,q,\chi}}(\mathcal{B}) := & \left| \Pr[\mathcal{B}(\mathbf{A}, \mathbf{b}) = 1 \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^m] \right. \\ & \left. - \Pr[\mathcal{B}(\mathbf{A}, \mathbf{A}^\top \mathbf{s} + \mathbf{e}) = 1 \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m] \right|. \end{aligned}$$

Let  $s > 0$ . We define the discrete Gaussian distribution over  $\mathbb{Z}$  with parameter  $s$ , denoted by  $D_{\mathbb{Z},s}$  to be the distribution proportional to  $\rho_s(\mathbf{x}) := \exp(-\pi \|\mathbf{x}\|^2 / s^2)$ , restricted to  $\mathbb{Z}$ . Next, we recall well-known regularity lemmas and tail bounds, following [29, 12].

**Lemma 2.6** *Consider natural numbers  $n, m \in \mathbb{N}$  and a prime  $q$  at least polynomial in  $n$ . Assume  $m \geq 2n \log q$  and  $s \geq \omega(\sqrt{\log m})$ . Then, the following distributions have negligible statistical distance:*

$$\{(\mathbf{A}, \mathbf{Ae}) \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \mathbf{e} \leftarrow D_{\mathbb{Z},s}^m\} \text{ and } \{(\mathbf{A}, \mathbf{b}) \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^n\}.$$

**Lemma 2.7** *For any  $s \geq \omega(\sqrt{\log m})$ , and  $\mathbf{x} \leftarrow D_{\mathbb{Z},s}^m$ , the probability that  $\|\mathbf{x}\| > s\sqrt{m}$  is at most  $2^{-m+1}$ .*

We also make use of the lossy LWE technique. For that, we require the following lemmas from [1, 23]. The lemmas make use of the so called ‘‘smooth average min-entropy’’  $\tilde{H}_\infty(\cdot \mid \cdot)$  [23].

**Lemma 2.8** Consider positive integers  $n, t, m, q, g$ , and  $\beta, s' > 0$  and a distribution  $\chi$  over  $\mathbb{Z}$  such that  $s' \geq \beta q n m$  and  $\Pr[|x| \geq \beta q \mid x \leftarrow \chi] \leq \text{negl}(\lambda)$ . Assume  $\mathbf{s}$  is uniformly distributed over  $[-g, g]^n$ , and  $\mathbf{e}$  is distributed according to  $D_{\mathbb{Z}, \mathbb{Z}}^s$ . Let  $\mathbf{B} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{n \times t}$ ,  $\mathbf{C} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{t \times m}$ ,  $\mathbf{D} \leftarrow \chi^{n \times m}$  and set  $\mathbf{A} := \mathbf{B}\mathbf{C} + \mathbf{D}$ . Then, for any  $\epsilon \geq 2^{-\lambda}$ , we have

$$\tilde{H}_{\infty}^{\epsilon}(\mathbf{s} \mid \mathbf{A}^{\top} \mathbf{s} + \mathbf{e}) \geq n \log(2g + 1) - (t + 2\lambda) \log q - \text{negl}(\lambda).$$

**Lemma 2.9** Let  $\mathcal{H} := \{h_k : \mathcal{X} \rightarrow \mathcal{Y}\}_k$  be a universal family of hash functions. Assume that the keys  $k$  of  $\mathcal{H}$  are distributed according to some distribution  $K$ . Further, let  $U$  denote a random variable distributed uniformly over  $\mathcal{Y}$  and  $X$  be any random variable with values in  $\mathcal{X}$  and  $I$  be any random variable. Let  $\epsilon \geq 0$ . With these assumptions, the statistical distance between  $(K, h_K(X), I)$  and  $(K, U, I)$  is upper bounded by

$$2\epsilon + \frac{1}{2} \sqrt{2^{-\tilde{H}_{\infty}^{\epsilon}(X \mid I)} \cdot |\mathcal{Y}|}.$$

### 3 Parameter Lossy Encryption

In this section, we focus on public key encryption. Formally, a public key encryption (PKE) scheme PKE consists of four algorithms (**Setup**, **KG**, **Enc**, **Dec**) and a message space  $\mathcal{M}$  that is assumed to be efficiently recognizable. The algorithms work as follows:

- The setup algorithm **Setup**, on input the security parameter  $\lambda$ , outputs system parameters  $\text{par}$ .
- The key generation algorithm **KG**, on input the parameter  $\text{par}$ , outputs a public and secret key pair  $(\text{pk}, \text{sk})$ .
- The encryption algorithm **Enc**, on input  $\text{pk}$  and a message  $m \in \mathcal{M}$ , outputs a ciphertext  $c \in \mathcal{C}$ .
- The decryption algorithm **Dec**, on input  $\text{sk}$  and a ciphertext  $c$ , outputs a message  $m' \in \mathcal{M}$  or a rejection symbol  $\perp \notin \mathcal{M}$ .

**Definition 3.1** (Correctness of PKE). A PKE scheme  $\text{PKE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$  with message space  $\mathcal{M}$  is  $(1 - \delta)$ -correct if

$$\mathbb{E}_{(\text{pk}, \text{sk}) \leftarrow \text{KG}} \left[ \max_{m \in \mathcal{M}} \Pr[\text{Dec}(\text{sk}, c) \neq m : c \leftarrow \text{Enc}(\text{pk}, m)] \right] \leq \delta,$$

where the expectation is taken over  $\text{par} \leftarrow \text{Setup}(\lambda)$ ,  $(\text{pk}, \text{sk}) \leftarrow \text{KG}(\text{par})$  and randomness of **Enc**. Here  $\delta := \delta(\lambda)$  is related to the security parameter  $\lambda$ .

For technical reasons, we also need a bound on the probability that two public keys collide.

**Definition 3.2** (Collision Probability of Key Generation). We define the collision probability of **KG** of PKE as

$$\eta_{\text{PKE}} := \max \left[ \Pr[\text{pk}_0 = \text{pk}_1 : (\text{pk}_0, \text{sk}_0) \leftarrow \text{KG}(\text{par}), (\text{pk}_1, \text{sk}_1) \leftarrow \text{KG}(\text{par})] \right],$$

where the maximum is taken over all  $\text{pk}_0, \text{pk}_1$ .

We can assume that  $\eta_{\text{PKE}}$  is negligible, as otherwise an adversary would have non-negligible probability of sampling a secret key for a given public key, which would imply that the scheme is insecure for any reasonable notion.

**LOSSY ENCRYPTION.** We recall the notion of lossy encryption [3, 16, 18]. In lossy encryption schemes, there are two modes of the public keys. Public keys in the real mode work as defined above. On the other hand, if we encrypt a plaintext using a public key in lossy mode, the ciphertext statistically hides the plaintext. Real and lossy public keys should be computationally indistinguishable. Unlike the lossy encryption in [3, 18], we do not require openness here.

**Definition 3.3** (Lossy Encryption). Let  $\text{PKE} := (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$  be a PKE scheme with message space  $\mathcal{M}'$ . PKE is *lossy* if there is an algorithm LKG such that the following properties hold:

- PKE is correct according to Definition 3.1.

- *Key Indistinguishability*: We say PKE satisfies key indistinguishability if for any algorithm  $\mathcal{B}$ , the advantage function

$$\text{Adv}_{\text{PKE}}^{\text{ind-key}}(\mathcal{B}) := |\Pr[\mathcal{B}(\text{par}, \text{pk}) \Rightarrow 1] - \Pr[\mathcal{B}(\text{par}, \text{lpk}) \Rightarrow 1]|$$

is negligible, where the probability is taken over  $\text{par} \leftarrow \text{Setup}(\lambda)$ ,  $(\text{pk}, \text{sk}) \leftarrow \text{KG}(\text{par})$ , and  $\text{lpk} \leftarrow \text{LKG}(\text{par})$ .

- *Lossiness*: For any arbitrary messages  $m, m' \in \mathcal{M}'$ , the statistical distance between the following distributions  $D$  and  $D'$  is at most  $\epsilon^{\text{lo}}$ , where  $\epsilon^{\text{lo}}$  is negligible:

$$D := \left\{ (\text{par}, \text{lpk}, c) \left| \begin{array}{l} \text{par} \leftarrow \text{Setup}(\lambda), \text{lpk} \leftarrow \text{LKG}(\text{par}) \\ c \leftarrow \text{Enc}(\text{lpk}, m) \end{array} \right. \right\},$$

$$D' := \left\{ (\text{par}, \text{lpk}, c) \left| \begin{array}{l} \text{par} \leftarrow \text{Setup}(\lambda), \text{lpk} \leftarrow \text{LKG}(\text{par}) \\ c \leftarrow \text{Enc}(\text{lpk}, m') \end{array} \right. \right\}.$$

We refer to  $\epsilon^{\text{lo}}$  as the lossiness of PKE.

We give a lattice-based lossy encryption in Section 3.3. The construction is essentially the Regev encryption scheme [33].

### 3.1 Parameter Lossy Encryption

We now extend the lossiness notion to a multi-user notion, where the global system parameters are also allowed to have a lossy mode. We call this new notion parameter lossy encryption.

**Definition 3.4** (Parameter Lossy Encryption). Let  $\text{PKE} := (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$  be a PKE scheme with message space  $\mathcal{M}'$ . PKE is *parameter lossy* if there are algorithms LSetup and LKG such that the following properties hold:

- PKE is correct according to Definition 3.1.
- *Parameter-Key Indistinguishability*: We say PKE satisfies parameter-key indistinguishability if for any PPT algorithm  $\mathcal{B}$ , the advantage function

$$\text{Adv}_{\text{PKE}}^{\text{ind-par-key}}(\mathcal{B}) := |\Pr[\mathcal{B}(\text{par}, \text{pk}_1, \dots, \text{pk}_N) \Rightarrow 1] - \Pr[\mathcal{B}(\text{lpar}, \text{lpk}_1, \dots, \text{lpk}_N) \Rightarrow 1]|$$

is negligible, where  $N$  denotes the number of users, and the first probability is taken over the experiment  $\text{par} \leftarrow \text{Setup}(\lambda)$ ,  $(\text{pk}_1, \text{sk}_1) \leftarrow \text{KG}(\text{par}), \dots, (\text{pk}_N, \text{sk}_N) \leftarrow \text{KG}(\text{par})$  and the second one is taken over  $\text{lpar} \leftarrow \text{LSetup}(\lambda)$ ,  $\text{lpk}_1 \leftarrow \text{LKG}(\text{lpar}), \dots, \text{lpk}_N \leftarrow \text{LKG}(\text{lpar})$ .

- *Lossiness*: For any arbitrary messages  $m, m' \in \mathcal{M}'$ , the statistical distance between the following distributions  $D$  and  $D'$  is at most  $\epsilon^{\text{lo}}$ , where  $\epsilon^{\text{lo}}$  is negligible:

$$D := \left\{ (\text{lpar}, \text{lpk}, c) \left| \begin{array}{l} \text{lpar} \leftarrow \text{LSetup}(\lambda), \text{lpk} \leftarrow \text{LKG}(\text{lpar}) \\ c \leftarrow \text{Enc}(\text{lpk}, m) \end{array} \right. \right\},$$

$$D' := \left\{ (\text{lpar}, \text{lpk}, c) \left| \begin{array}{l} \text{lpar} \leftarrow \text{LSetup}(\lambda), \text{lpk} \leftarrow \text{LKG}(\text{lpar}) \\ c \leftarrow \text{Enc}(\text{lpk}, m') \end{array} \right. \right\}.$$

We refer to  $\epsilon^{\text{lo}}$  as the lossiness of PKE.

### 3.2 Parameter Lossy Encryption from Lattices

We construct a parameter lossy encryption scheme from the (Decisional) Learning With Errors (LWE) assumption. Essentially, in our encryption we extend the (dual) Regev scheme [33, 13] with a lossy mode of system parameters.

**SCHEME.** Our scheme has message space  $\{0, 1\}^\ell$ . As common for lattice-based encryption schemes, a message  $m \in \{0, 1\}^\ell$  has to be encoded on encryption and decoded on decryption. More precisely, we define the following algorithms and use them in our scheme:

- Algorithm  $\text{Encode}(m)$  computes a vector  $\mathbf{m}^\top \in \mathbb{Z}_q^\ell$ . The  $i^{\text{th}}$  coordinate of  $\mathbf{m}^\top$  is given as  $\lfloor q/2 \rfloor \cdot m_i$  for each  $i \in [\ell]$ .
- Algorithm  $\text{Decode}(\mathbf{m}^\top)$  computes a message  $m \in \{0, 1\}^\ell$  by componentwise rounding. That is, for all  $i \in [\ell]$ , it sets  $m_i = 0$  if  $\mathbf{m}_i$  is closer to 0 than to  $\lfloor q/2 \rfloor$ . Otherwise, it sets  $m_i = 1$ .

Further, our scheme makes use of parameters  $n, m, q, t, g \in \mathbb{N}, s, s', s'' \in \mathbb{R}, s, s', s'' > 0$  satisfying the following conditions:

- $n = \Theta(\lambda)$ ,  $q$  prime
- $m \geq 2n \log q$  (for Lemma 2.6)
- $s, s' \geq \omega(\sqrt{\log m})$  (for Lemmata 2.6 and 2.7)
- $ss'm \leq q/4$  (for correctness)
- $s' \geq gn^2m$  (for Lemma 2.8, we choose  $\beta q = n$ )
- $n \log(2g + 1) - (t + 2\lambda) \log q - \text{negl}(\lambda) \geq \lambda \log q + \Omega(n)$  (for Lemma 2.9)

For example, a (very conservative) parameter setting that satisfies all these conditions for a given  $\lambda$  is

$$\begin{array}{llll} n := 56\lambda & n^6 < q \leq n^7, & g := \sqrt{n}, & s := \sqrt{n}, \\ t := \lambda, & m := 2n \log q, & s' := n^{2.5}m, & s'' := \sqrt{n}. \end{array}$$

Formally, we present our scheme in Figure 2.

<b>Setup(<math>\lambda</math>)</b> 01 $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ 02 <b>return</b> $\text{par} := \mathbf{A}$	<b>KG(<math>\text{par} = \mathbf{A}</math>)</b> 07 $\text{sk} := \mathbf{X} \leftarrow D_{\mathbb{Z}, s}^{m \times \ell}$ 08 $\text{pk} := \mathbf{Y} := \mathbf{A}\mathbf{X}$ 09 <b>return</b> $(\text{pk}, \text{sk})$	<b>Enc(<math>\text{pk} = \mathbf{Y}, m</math>)</b> 12 $\mathbf{s} \xleftarrow{\$} [-g, g]^n, \mathbf{e} \leftarrow D_{\mathbb{Z}, s'}^m$ 13 $\mathbf{m}^\top := \text{Encode}(m)$ 14 $\mathbf{c}^\top := \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$ 15 $\mathbf{v}^\top := \mathbf{s}^\top \mathbf{Y} + \mathbf{m}^\top$ 16 <b>return</b> $c := (\mathbf{c}^\top, \mathbf{v}^\top)$
<b>LSetup(<math>\lambda</math>)</b> 03 $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times t}, \mathbf{C} \xleftarrow{\$} \mathbb{Z}_q^{t \times m}$ 04 $\mathbf{D} \leftarrow D_{\mathbb{Z}, s''}^{n \times m}$ 05 $\mathbf{A} := \mathbf{B}\mathbf{C} + \mathbf{D}$ 06 <b>return</b> $\text{lpar} := \mathbf{A}$	<b>LKG(<math>\text{lpar} = \mathbf{A}</math>)</b> 10 $\text{lpk} := \mathbf{Y} \xleftarrow{\$} \mathbb{Z}_q^{n \times \ell}$ 11 <b>return</b> $\text{lpk}$	<b>Dec(<math>\text{sk} = \mathbf{x}, c = (\mathbf{c}^\top, \mathbf{v}^\top)</math>)</b> 17 $\mathbf{m}^\top := \mathbf{v}^\top - \mathbf{c}^\top \mathbf{X}$ 18 <b>return</b> $\text{Decode}(\mathbf{m}^\top)$

Figure 2: The parameter lossy encryption scheme  $\text{PKE} := (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$  from the LWE assumption with algorithms  $\text{LSetup}$  and  $\text{LKG}$ .

ANALYSIS. We show correctness, parameter-key indistinguishability, and lossiness. The proof of correctness follows standard arguments [33].

**Lemma 3.5** *The scheme PKE in Figure 2 is  $(1 - \delta)$ -correct, for negligible  $\delta$ .*

*Proof.* Let  $\text{sk} = \mathbf{X} \leftarrow D_{\mathbb{Z}, s}^{m \times \ell}$  and  $\text{pk} = \mathbf{Y} = \mathbf{A}\mathbf{X}$  be a pair of public key and secret key. Consider a message  $m \in \{0, 1\}^\ell$  and an honestly computed ciphertext  $c := (\mathbf{c}^\top, \mathbf{v}^\top)$  for  $m$ . We have  $\mathbf{c}^\top = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top$  and  $\mathbf{v}^\top := \mathbf{s}^\top \mathbf{Y} + \mathbf{m}^\top$ . Now, consider  $\mathbf{m}^\top$  computed during the decryption algorithm. We have

$$\mathbf{m}^\top = \mathbf{v}^\top - \mathbf{c}^\top \mathbf{X} = \mathbf{m}^\top - \mathbf{e}^\top \mathbf{X}.$$

Thus, one can see that decryption recovers  $m$  if each coordinate of  $\mathbf{e}^\top \mathbf{X}$  has absolute value less than  $q/4$ . Fix such a coordinate, say the  $i$ th, and call it  $z$ . Except with negligible probability (see Lemma 2.7), we have that  $\|\mathbf{e}\| \leq s'\sqrt{m}$ , and the  $i$ th column  $\mathbf{x}$  of  $\mathbf{X}$  satisfies  $\|\mathbf{x}\| \leq s\sqrt{m}$ . Thus, we have

$$|z| = |\mathbf{e}^\top \mathbf{x}| \leq \|\mathbf{e}\| \|\mathbf{x}\| \leq ss'm < q/4.$$

except with negligible probability. □

**Lemma 3.6** *If the  $\text{LWE}_{t, m, q, D_{\mathbb{Z}, s''}}$  assumption holds, then the scheme PKE with algorithms  $\text{LSetup}$  and  $\text{LKG}$  as presented in Figure 2 satisfies parameter-key indistinguishability. Namely, for any adversary  $\mathcal{A}$ , there is an algorithm  $\mathcal{B}$  such that the running time of  $\mathcal{B}$  is about that of  $\mathcal{A}$  and*

$$\text{Adv}_{\text{PKE}}^{\text{ind-par-key}}(\mathcal{A}) \leq n \cdot \text{Adv}_{\text{LWE}_{t, m, q, D_{\mathbb{Z}, s''}}}(\mathcal{B}) + \text{negl}(\lambda).$$

*Proof.* To show parameter-key indistinguishability, we need to argue that the distributions of (1) parameters and keys output by **Setup** and **KG** and (2) parameters and keys output by **LSetup** and **LKG** are computationally indistinguishable. We show this using a sequence of hybrid distributions. Namely, we start with distribution  $D_1$ , which is the distribution output by **Setup** and **KG**, namely

$$D_1 := \left\{ (\mathbf{A}, \mathbf{Y}_1, \dots, \mathbf{Y}_N) \mid \begin{array}{l} \mathbf{A} \leftarrow_{\mathbb{S}} \mathbb{Z}_q^{n \times m}, \\ \forall i \in [N] : \mathbf{X}_i \leftarrow D_{\mathbb{Z}, s}^{m \times \ell}, \mathbf{Y}_i := \mathbf{A} \mathbf{X}_i \end{array} \right\}.$$

Now, we argue that the distribution

$$D_2 := \left\{ (\mathbf{A}, \mathbf{Y}_1, \dots, \mathbf{Y}_N) \mid \begin{array}{l} \mathbf{A} \leftarrow_{\mathbb{S}} \mathbb{Z}_q^{n \times m}, \\ \forall i \in [N] : \mathbf{Y}_i \leftarrow_{\mathbb{S}} \mathbb{Z}_q^{n \times \ell} \end{array} \right\}$$

is statistically close to  $D_1$ . This can easily be seen using  $\ell \cdot N$  applications of Lemma 2.6. Next, using  $n$  applications of the  $\text{LWE}_{t, m, q, D_{\mathbb{Z}, s''}}$  assumption (one per row of  $\mathbf{A}$ ), we see that the distribution

$$D_3 := \left\{ (\mathbf{A}, \mathbf{Y}_1, \dots, \mathbf{Y}_N) \mid \begin{array}{l} \mathbf{B} \leftarrow_{\mathbb{S}} \mathbb{Z}_q^{n \times t}, \mathbf{C} \leftarrow_{\mathbb{S}} \mathbb{Z}_q^{t \times m}, \mathbf{D} \leftarrow D_{\mathbb{Z}, s''}^{n \times m} \\ \mathbf{A} := \mathbf{B} \mathbf{C} + \mathbf{D}, \\ \forall i \in [N] : \mathbf{Y}_i \leftarrow_{\mathbb{S}} \mathbb{Z}_q^{n \times \ell} \end{array} \right\}$$

is computationally indistinguishable from  $D_2$ . Finally, observe that  $D_3$  is exactly the distribution of parameters and keys output by **LSetup** and **LKG**.  $\square$

**Lemma 3.7** *The scheme PKE with algorithms **LSetup** and **LKG** as presented in Figure 2 satisfies lossiness.*

*Proof.* Fix two arbitrary messages  $m, m' \in \{0, 1\}^\ell$ . According to the definition of lossiness, and the specification of scheme PKE and algorithms **LSetup** and **LKG**, we need to argue that the following distributions  $D$  and  $D'$  are statistically close:

$$D := \left\{ (\mathbf{A}, \mathbf{Y}, \mathbf{c}^\top, \mathbf{v}^\top) \mid \begin{array}{l} \mathbf{B} \leftarrow_{\mathbb{S}} \mathbb{Z}_q^{n \times t}, \mathbf{C} \leftarrow_{\mathbb{S}} \mathbb{Z}_q^{t \times m}, \mathbf{D} \leftarrow D_{\mathbb{Z}, s''}^{n \times m}, \\ \mathbf{A} := \mathbf{B} \mathbf{C} + \mathbf{D}, \mathbf{Y} \leftarrow_{\mathbb{S}} \mathbb{Z}_q^{n \times \ell}, \\ \mathbf{c}^\top := \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top, \mathbf{v}^\top := \mathbf{s}^\top \mathbf{Y} + \text{Encode}(m) \end{array} \right\},$$

$$D' := \left\{ (\mathbf{A}, \mathbf{Y}, \mathbf{c}^\top, \mathbf{v}^\top) \mid \begin{array}{l} \mathbf{B} \leftarrow_{\mathbb{S}} \mathbb{Z}_q^{n \times t}, \mathbf{C} \leftarrow_{\mathbb{S}} \mathbb{Z}_q^{t \times m}, \mathbf{D} \leftarrow D_{\mathbb{Z}, s''}^{n \times m}, \\ \mathbf{A} := \mathbf{B} \mathbf{C} + \mathbf{D}, \mathbf{Y} \leftarrow_{\mathbb{S}} \mathbb{Z}_q^{n \times \ell}, \\ \mathbf{c}^\top := \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top, \mathbf{v}^\top := \mathbf{s}^\top \mathbf{Y} + \text{Encode}(m') \end{array} \right\}.$$

Observe that it is sufficient to argue that  $\mathbf{s}^\top \mathbf{Y}$  is statistically close to uniform over  $\mathbb{Z}_q^\ell$ , given  $\mathbf{A}, \mathbf{Y}, \mathbf{c}^\top$  as in  $D$  and  $D'$ . To do this, we make use of Lemma 2.9. Namely, we consider the hash function family  $\mathbf{s} \mapsto \mathbf{s}^\top \mathbf{Y}$  parameterized by  $\mathbf{Y}$ . As  $\mathbf{Y}$  is sampled uniformly at random in distributions  $D$  and  $D'$ , and  $q$  is a prime, this family is universal. Next, we claim that  $\mathbf{s}$  has a lot of entropy given  $\mathbf{c}^\top$ . Precisely, we use Lemma 2.8 and derive

$$\begin{aligned} \tilde{H}_\infty^\epsilon(\mathbf{s} \mid \mathbf{c}) &\geq n \log(2g + 1) - (t + 2\lambda) \log q - \text{negl}(\lambda) \\ &\geq \lambda \log q + \Omega(n), \end{aligned}$$

where the first inequality follows from Lemma 2.8, and the last inequality follows from our assumptions on parameters. Now that the lower bound on the entropy of  $\mathbf{s}$  is established, we use Lemma 2.9 with  $\epsilon = 2^{-\lambda}$  and  $\mathcal{Y} := \mathbb{Z}_q^\lambda$ , and get that the statistical distance between  $\mathbf{s}^\top \mathbf{Y}$  and uniform, given  $\mathbf{A}, \mathbf{Y}, \mathbf{c}^\top$ , is at most

$$2\epsilon + \frac{1}{2} \sqrt{2^{-\tilde{H}_\infty^\epsilon(\mathbf{s} \mid \mathbf{c})} \cdot |\mathcal{Y}|} \leq 2^{-\lambda+1} + \frac{1}{2} \sqrt{2^{-\lambda \log q - \Omega(n) + \lambda \log q}} \leq \text{negl}(\lambda),$$

which finishes the proof.  $\square$

### 3.3 Lossy Encryption from Lattices

We present a simple construction of lossy encryption from lattices. The construction is essentially Regev's public key encryption scheme [33]. Formally, the public key encryption  $\text{PKE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$  and algorithm  $\text{LKG}$  for message space  $\{0, 1\}^\ell$  is given in Figure 3. For our description, we rely on algorithms  $\text{Encode}$  and  $\text{Decode}$  introduced in Section 3.2. It makes use of parameters  $n, m, q \in \mathbb{N}, s, s' \in \mathbb{R}, s, s' > 0$ , that should satisfy the following conditions

- $n = \Theta(\lambda)$ ,  $q$  prime
- $m \geq 2(n + \ell) \log q$  (for Lemma 2.6)
- $s, s' \geq \omega(\sqrt{\log m})$  (for Lemmata 2.6 and 2.7)
- $ss'm \leq q/4$  (for correctness)

An example non-optimized instantiation for a given security parameter  $\lambda$  and message length  $\ell = n$  is  $n := \lambda$ ,  $n^3 < q \leq n^4$ ,  $m := 4n \log q$ , and  $s := s' := \log m$ .

<p><u>Setup(<math>1^\lambda</math>)</u></p> <p>01 <b>return</b> <math>\text{par} := \mathbf{A} \xleftarrow{\\$} \mathbb{Z}_q^{n \times m}</math></p> <p><u>KG(<math>\text{par} = \mathbf{A}</math>)</u></p> <p>02 <math>\text{sk} := \mathbf{S} \xleftarrow{\\$} \mathbb{Z}_q^{n \times \ell}</math>, <math>\mathbf{E} \leftarrow D_{\mathbb{Z}, s}^{m \times \ell}</math></p> <p>03 <math>\text{pk} := \mathbf{Y} := \mathbf{S}^\top \mathbf{A} + \mathbf{E}^\top \in \mathbb{Z}_q^{\ell \times m}</math></p> <p>04 <b>return</b> <math>(\text{pk}, \text{sk})</math></p> <p><u>LKG(<math>\text{par} = \mathbf{A}</math>)</u></p> <p>05 <b>return</b> <math>\text{lpk} := \mathbf{Y} \xleftarrow{\\$} \mathbb{Z}_q^{\ell \times m}</math></p>	<p><u>Enc(<math>\text{pk} = \mathbf{Y}, m</math>)</u></p> <p>06 <math>\mathbf{x} \leftarrow D_{\mathbb{Z}, s'}^m</math></p> <p>07 <math>\mathbf{c} := \mathbf{A}\mathbf{x}</math></p> <p>08 <math>\mathbf{v} := \mathbf{Y}\mathbf{x} + \text{Encode}(m)^\top</math></p> <p>09 <b>return</b> <math>c := (\mathbf{c}, \mathbf{v})</math></p> <p><u>Dec(<math>\text{sk} = \mathbf{S}, c = (\mathbf{c}, \mathbf{v})</math>)</u></p> <p>10 <math>\mathbf{m} := \mathbf{v} - \mathbf{S}^\top \mathbf{c}</math></p> <p>11 <b>return</b> <math>\text{Decode}(\mathbf{m}^\top)</math></p>
--	--

Figure 3: The lossy PKE scheme  $\text{PKE} := (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$  from the LWE assumption with algorithm  $\text{KG}$ .

We now turn to the analysis of PKE. We show correctness, key indistinguishability, and lossiness.

**Lemma 3.8** *The scheme PKE in Figure 3 is  $(1 - \delta)$ -correct, for negligible  $\delta$ .*

*Proof.* The proof is standard [33, 13]. One can easily see that decryption works as long as  $|\mathbf{e}^\top \mathbf{x}| < q/4$  for any column  $\mathbf{e}$  of  $\mathbf{E}$ . By Lemma 2.7 and our assumption about  $s, s', m$ , and  $q$ , we have

$$|\mathbf{e}^\top \mathbf{x}| \leq \|\mathbf{e}\| \|\mathbf{x}\| \leq ss'm < q/4.$$

with overwhelming probability. □

**Lemma 3.9** *If the  $\text{LWE}_{n, m, q, D_{\mathbb{Z}, s}}$  assumption holds, then the scheme PKE with algorithm LKG as presented in Figure 3 satisfies key indistinguishability. Namely, for any algorithm  $\mathcal{A}$ , there is an algorithm  $\mathcal{B}$  such that the running time of  $\mathcal{B}$  is about that of  $\mathcal{A}$  and*

$$\text{Adv}_{\text{PKE}}^{\text{ind-key}}(\mathcal{A}) \leq \ell \cdot \text{Adv}_{\text{LWE}_{n, m, q, D_{\mathbb{Z}, s}}}(\mathcal{B})$$

*Proof.* The statement follows directly from the LWE assumption, applied to each row of matrix  $\mathbf{Y}$ . □

**Lemma 3.10** *The scheme PKE with algorithm LKG as presented in Figure 3 satisfies lossiness.*

*Proof.* Fix two arbitrary messages  $m, m' \in \{0, 1\}^\ell$ . Now, according to the definition of lossiness and the specification of the scheme, we have to argue that the distributions  $D$  and  $D'$  are statistically close, where  $D$  and  $D'$  are given as

$$D := \left\{ (\mathbf{A}, \mathbf{Y}, \mathbf{c}, \mathbf{v}) \left| \begin{array}{l} \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \mathbf{Y} \xleftarrow{\$} \mathbb{Z}_q^{\ell \times m} \\ \mathbf{c} := \mathbf{A}\mathbf{x}, \mathbf{v} := \mathbf{Y}\mathbf{x} + \text{Encode}(m)^\top \end{array} \right. \right\},$$

$$D' := \left\{ (\mathbf{A}, \mathbf{Y}, \mathbf{c}, \mathbf{v}) \mid \begin{array}{l} \mathbf{A} \xleftarrow{s} \mathbb{Z}_q^{n \times m}, \mathbf{Y} \xleftarrow{s} \mathbb{Z}_q^{\ell \times m} \\ \mathbf{c} := \mathbf{A}\mathbf{x}, \mathbf{v} := \mathbf{Y}\mathbf{x} + \text{Encode}(m')^\top \end{array} \right\}.$$

It is sufficient that in both distributions the term

$$\begin{bmatrix} \mathbf{A} \\ \mathbf{Y} \end{bmatrix} \mathbf{x}$$

is statistically close to uniform. This is guaranteed by Lemma 2.6.  $\square$

## 4 CCA Secure KEMs from (Parameter) Lossy Encryption

In this section, we construct two KEM schemes  $\text{KEM}_1$  and  $\text{KEM}_2$  from lossy encryption and parameter lossy encryption, respectively. The schemes  $\text{KEM}_1$  and  $\text{KEM}_2$  have tight multi-challenge, and tight multi-user multi-challenge security, respectively, and will be used in the construction of our AKE protocol in Section 6. Before we describe the schemes in Sections 4.1 and 4.2, we recall the formal definition of KEMs and define the security notions of interest.

**DEFINITIONS.** We recall the syntax and security definitions of a KEM. A KEM  $\text{KEM}$  consists of four algorithms ( $\text{Setup}$ ,  $\text{KGen}$ ,  $\text{Encaps}$ ,  $\text{Decaps}$ ) and a key space  $\mathcal{K}$  that is assumed to be efficiently recognizable. The algorithms work as follows:

- The setup algorithm  $\text{Setup}$ , on input the security parameter  $\lambda$ , outputs system parameters  $\text{par}$ .
- The key generation algorithm  $\text{KGen}$ , on input the parameter  $\text{par}$ , outputs a public and secret key pair  $(\text{pk}, \text{sk})$ .
- The encapsulation algorithm  $\text{Encaps}$ , on input  $\text{pk}$ , outputs a ciphertext  $e$  and a key  $K \in \mathcal{K}$ .
- The decapsulation algorithm  $\text{Decaps}$ , on input  $\text{sk}$  and a ciphertext  $e$ , outputs a key  $K \in \mathcal{K}$  or a rejection symbol  $\perp \notin \mathcal{K}$ .

In this paper, we use MC-IND-CCA secure KEM and MUC-IND-CCA secure KEM to construct AKE protocols.

**Definition 4.1** (MC-IND-CCA Security of KEM). Let  $\text{KEM} = (\text{Setup}, \text{KGen}, \text{Encaps}, \text{Decaps})$  be a KEM. We say that  $\text{KEM}$  is MC-IND-CCA secure, if for any algorithm  $\mathcal{A}$ , the advantage

$$\begin{aligned} \text{Adv}_{\text{KEM}}^{\text{MC-IND-CCA}}(\mathcal{A}) := & \left| \Pr[\text{MC-IND-CCA}_{\text{KEM},0}^{\mathcal{A}}(\lambda) \Rightarrow 1] \right. \\ & \left. - \Pr[\text{MC-IND-CCA}_{\text{KEM},1}^{\mathcal{A}}(\lambda) \Rightarrow 1] \right| \end{aligned}$$

is negligible in  $\lambda$ , where games  $\text{MC-IND-CCA}_{\text{KEM},b}^{\mathcal{A}}(\lambda)$  for  $b \in \{0, 1\}$  are specified in Figure 4.

**Definition 4.2** (MUC-IND-CCA Security of KEM). Let  $\text{KEM} = (\text{Setup}, \text{KGen}, \text{Encaps}, \text{Decaps})$  be a KEM. We say that  $\text{KEM}$  is MUC-IND-CCA secure, if for any algorithm  $\mathcal{A}$ , the advantage

$$\begin{aligned} \text{Adv}_{\text{KEM}}^{\text{MUC-IND-CCA}}(\mathcal{A}) := & \left| \Pr[\text{MUC-IND-CCA}_{\text{KEM},0}^{\mathcal{A}}(\lambda) \Rightarrow 1] \right. \\ & \left. - \Pr[\text{MUC-IND-CCA}_{\text{KEM},1}^{\mathcal{A}}(\lambda) \Rightarrow 1] \right| \end{aligned}$$

is negligible in  $\lambda$ , where games  $\text{MUC-IND-CCA}_{\text{KEM},b}^{\mathcal{A}}(\lambda)$  for  $b \in \{0, 1\}$  are specified in Figure 4.

### 4.1 MC-IND-CCA Secure KEM from Lossy Encryption

Let  $\text{PKE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$  be a lossy encryption scheme with message space  $\mathcal{M}'$ , randomness space  $\mathcal{R}'$ , and ciphertext space  $\mathcal{C}'$ . Let  $s$  be an integer and  $\mathcal{K}$  be a key space. Let  $\text{H}: \mathcal{M}' \times \mathcal{C}' \rightarrow \mathcal{K}$ ,  $\text{H}': \{0, 1\}^s \times \mathcal{C}' \rightarrow \{0, 1\}^s$ , and  $\text{G}: \mathcal{M}' \rightarrow \mathcal{R}'$  be random oracles. Our KEM scheme  $\text{KEM}_1$  with KEM key space  $\mathcal{K}$  is shown in Figure 5.

$\text{KEM}_1$  has the same structure as the modular Fujisaki-Okamoto transformation  $\text{FO}^\times[\text{PKE}, \text{G}, \text{H}]$  from [21, 19], but its underlying PKE is a lossy encryption scheme. Theorem 4.3 shows that, if PKE is a lossy encryption, then  $\text{KEM}_1$  is a tightly IND-CCA secure KEM in the multi-challenge setting (Definition 4.1) in the QROM.

Game MC-IND-CCA <sub>KEM,b</sub> <sup>A</sup> ( $\lambda$ )	Game MUC-IND-CCA <sub>KEM,b</sub> <sup>A</sup> ( $\lambda$ )	Oracle DEC( $e$ )
01 $\text{par} \leftarrow \text{Setup}(\lambda)$	09 $\text{par} \leftarrow \text{Setup}(\lambda)$	20 <b>if</b> $e \in \mathbf{e}$
02 $(\text{pk}, \text{sk}) \leftarrow \text{KGen}(\text{par})$	10 <b>for</b> $j \in [N]$	21 <b>return</b> $\perp$
03 <b>for</b> $i \in [S]$	11 $(\text{pk}_j, \text{sk}_j) \leftarrow \text{KGen}(\text{par})$	22 $K := \text{Decaps}(\text{sk}, e)$
04 $(e, K) \leftarrow \text{Encaps}(\text{pk})$	12 <b>for</b> $i \in [S]$	23 <b>return</b> $K$
05 $\mathbf{e}[i] := e, \mathbf{K}_0[i] := K$	13 $(e, K) \leftarrow \text{Encaps}(\text{pk}_j)$	<b>Oracle</b> DEC <sub>mu</sub> ( $j, e$ )
06 $\mathbf{K}_1[i] \stackrel{\$}{\leftarrow} \mathcal{K}$	14 $\mathbf{e}[j, i] := e$	24 <b>if</b> $e \in \mathbf{e}[j, \cdot]$
07 $b' \leftarrow \mathcal{A}^{\text{DEC}}(\text{par}, \text{pk}, \mathbf{e}, \mathbf{K}_b)$	15 $\mathbf{K}_0[j, i] := K$	25 <b>return</b> $\perp$
08 <b>return</b> $b'$	16 $\mathbf{K}_1[j, i] \stackrel{\$}{\leftarrow} \mathcal{K}$	26 $K := \text{Decaps}(\text{sk}_j, e)$
	17 $\text{pk}[j] := \text{pk}_j$	27 <b>return</b> $K$
	18 $b' \leftarrow \mathcal{A}^{\text{DEC}_{\text{mu}}}(\text{par}, \text{pk}, \mathbf{e}, \mathbf{K}_b)$	
	19 <b>return</b> $b'$	

Figure 4: Games MC-IND-CCA<sub>KEM,b</sub><sup>A</sup> and MUC-IND-CCA<sub>KEM,b</sub><sup>A</sup> for a KEM  $\text{KEM} = (\text{Setup}, \text{KGen}, \text{Encaps}, \text{Decaps})$ . In DEC<sub>mu</sub>,  $\mathbf{e}[j, \cdot]$  is the list  $(\mathbf{e}[j, 1], \dots, \mathbf{e}[j, S])$ .

KGen <sub>1</sub> (par)	Encaps <sub>1</sub> (pk)	Decaps <sub>1</sub> ((sk, k), e)
01 $(\text{pk}, \text{sk}) \leftarrow \text{KG}(\text{par})$	06 $r \stackrel{\$}{\leftarrow} \mathcal{M}', R := \text{G}(r)$	10 $r' := \text{Dec}(\text{sk}, e)$
02 $k \stackrel{\$}{\leftarrow} \mathcal{M}'$	07 $e := \text{Enc}(\text{pk}, r; R)$	11 <b>if</b> $r' = \perp \vee e \neq \text{Enc}(\text{pk}, r'; \text{G}(r'))$
03 $\text{pk}' := \text{pk}$	08 $K := \text{H}(r, e)$	12 $K := \text{H}'(k, e)$
04 $\text{sk}' := (\text{sk}, k)$	09 <b>return</b> $(e, K)$	13 <b>else</b> $K := \text{H}(r', e)$
05 <b>return</b> $(\text{pk}', \text{sk}')$		14 <b>return</b> $K$

Figure 5: The KEM scheme  $\text{KEM}_1 = (\text{Setup} := \text{Setup}, \text{KGen}_1, \text{Encaps}_1, \text{Decaps}_1)$  based on a lossy encryption scheme  $\text{PKE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$ , where  $\text{par} \leftarrow \text{Setup}(\lambda)$ .  $\text{KEM}_1$  has implicit rejection property, namely, the decryption algorithm returns a pseudorandom KEM key if the input ciphertext is invalid.

**Theorem 4.3** *Let  $S$  be the number of challenge ciphertexts. If PKE is a  $(1 - \delta)$ -correct lossy encryption (Definition 3.3) with lossiness  $\epsilon_{\text{PKE}}^{\text{lo}}$  and  $\text{H}', \text{G}$ , and  $\text{H}$  are modeled as quantum random oracles, then for any quantum adversary  $\mathcal{A}$ , there exists an adversary  $\mathcal{B}$  such that the running time of  $\mathcal{A}$  is about that of  $\mathcal{B}$  and*

$$\begin{aligned}
\text{Adv}_{\text{KEM}_1}^{\text{MC-IND-CCA}}(\mathcal{A}) &\leq 4\text{Adv}_{\text{PKE}}^{\text{ind-key}}(\mathcal{B}) + S^2 \left( \frac{1}{|\mathcal{M}'|} + \frac{1}{|\mathcal{K}|} + \frac{1}{2^s} \right) \\
&\quad + \frac{S + S^2}{|\mathcal{R}'|} + 48(1 + (q_{\text{H}} + q_{\text{G}} + 2q_{\text{DEC}} + S)^2)\delta \\
&\quad + 4(q_{\text{G}} + q_{\text{H}}) \sqrt{S \cdot \epsilon_{\text{PKE}}^{\text{lo}} + \frac{S}{|\mathcal{M}'|}} + 4q_{\text{H}'} \cdot 2^{-s/2},
\end{aligned}$$

where  $q_{\text{H}'}, q_{\text{G}}, q_{\text{H}}$ , and  $q_{\text{DEC}}$  are the numbers of  $\mathcal{A}$ 's queries to  $\text{H}', \text{G}, \text{H}$ , and  $\text{DEC}$ , respectively.

The proof of Theorem 4.3 is the almost identical to the one of Theorem 4.4, except that Theorem 4.3 deals with only one user and uses the key indistinguishability of lossy encryption (cf. Definition 3.3) instead of the parameter-key indistinguishability. By letting  $N := 1$  in the proof of Theorem 4.4, all arguments can be adapted to the proof of Theorem 4.3. Thus, we refer the reader to the proof of Theorem 4.4.

## 4.2 MUC-IND-CCA Secure KEM from Parameter Lossy Encryption

Let  $\text{PKE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$  be a parameter lossy encryption scheme with public key space  $\mathcal{PK}'$ , message space  $\mathcal{M}'$ , randomness space  $\mathcal{R}'$ , and ciphertext space  $\mathcal{C}'$ . Let  $s$  be an integer and  $\mathcal{K}$  be a key space. Let  $\text{H}: \mathcal{PK}' \times \mathcal{M}' \times \mathcal{C}' \rightarrow \mathcal{K}$ ,  $\text{H}': \mathcal{PK}' \times \{0, 1\}^s \times \mathcal{C}' \rightarrow \{0, 1\}^s$ , and  $\text{G}: \mathcal{PK}' \times \mathcal{M}' \rightarrow \mathcal{R}'$  be random oracles. Fix  $\text{par} \leftarrow \text{Setup}(\lambda)$  and our KEM scheme  $\text{KEM}_2$  with KEM key space  $\mathcal{K}$  is defined as in Figure 6.

$\text{KEM}_2$  has two differences compared to the modular Fujisaki-Okamoto transformation  $\text{FO}^{\mathcal{X}}[\text{PKE}, \text{G}, \text{H}]$  from [21, 19]. The first one is that we include a user public key into the hash function. We suppose that

KGen <sub>2</sub> (par)	Encaps <sub>2</sub> (pk)	Decaps <sub>2</sub> ((sk, k), e)
01 (pk, sk) ← KG(par)	06 $r \xleftarrow{\$} \mathcal{M}'$	11 $r' := \text{Dec}(\text{sk}, e)$
02 $k \xleftarrow{\$} \{0, 1\}^s$	07 $R := \text{G}(\text{pk}, r)$	12 <b>if</b> $r' = \perp$
03 $\text{pk}' := \text{pk}$	08 $e := \text{Enc}(\text{pk}, r; R)$	13 $\quad \forall e \neq \text{Enc}(\text{pk}, r'; \text{G}(\text{pk}, r'))$
04 $\text{sk}' := (\text{sk}, k)$	09 $K := \text{H}(\text{pk}, r, e)$	14 $\quad K := \text{H}'(\text{pk}, k, e)$
05 <b>return</b> (pk', sk')	10 <b>return</b> (e, K)	15 <b>else</b> $K := \text{H}(\text{pk}, r', e)$
		16 <b>return</b> K

Figure 6: KEM scheme  $\text{KEM}_2 = (\text{Setup}, \text{KGen}_2, \text{Encaps}_2, \text{Decaps}_2)$  based on a parameter lossy encryption  $\text{PKE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$ , where  $\text{par} \leftarrow \text{Setup}(\lambda)$ .  $\text{KEM}_2$  has implicit rejection property, namely, the decryption algorithm returns a pseudorandom KEM key if the input ciphertext is invalid.

this change is necessary for tightness in the multi-user setting. The second difference is that our security requirement on the underlying PKE is parameter lossy. More precisely, we show in Theorem 4.4 that, if PKE is a parameter lossy encryption, then  $\text{KEM}_2$  is a tightly IND-CCA secure KEM in the multi-user and multi-challenge setting (Definition 4.2) in the QROM.

**Theorem 4.4** *Let  $N$  be the number of users and let  $S$  be the number of challenge ciphertexts. If PKE is a  $(1 - \delta)$ -correct parameter lossy encryption (Definition 3.4) with lossiness  $\epsilon_{\text{PKE}}^{\text{lo}}$  and  $\text{H}'$ ,  $\text{G}$ , and  $\text{H}$  are modeled as quantum random oracles, then for any quantum adversary  $\mathcal{A}$ , there exists an adversary  $\mathcal{B}$  such that the running time of  $\mathcal{A}$  is about that of  $\mathcal{B}$  and*

$$\begin{aligned} \text{Adv}_{\text{KEM}_2}^{\text{MUC-IND-CCA}}(\mathcal{A}) &\leq 4\text{Adv}_{\text{PKE}}^{\text{ind-par-key}}(\mathcal{B}) + 48N(1 + (q_{\text{H}} + q_{\text{G}} + 2q_{\text{DEC}} + S)^2)\delta \\ &\quad + \frac{NS + N^2S^2}{|\mathcal{R}'|} + N^2S^2 \left( \frac{1}{|\mathcal{M}'|} + \frac{1}{|\mathcal{K}|} + \frac{1}{2^s} + \eta_{\text{PKE}} \right) \\ &\quad + 4(q_{\text{G}} + q_{\text{H}}) \sqrt{NS \cdot \epsilon_{\text{PKE}}^{\text{lo}} + \frac{NS}{|\mathcal{M}'|}} + 4Nq_{\text{H}'} \cdot 2^{-s/2}, \end{aligned}$$

where  $q_{\text{H}'}$ ,  $q_{\text{G}}$ ,  $q_{\text{H}}$ , and  $q_{\text{DEC}}$  are the numbers of  $\mathcal{A}$ 's queries to  $\text{H}'$ ,  $\text{G}$ ,  $\text{H}$ , and  $\text{DEC}_{\text{mu}}$ , respectively.  $\eta_{\text{PKE}}$  is the collision probability of  $\text{KG}$  (Definition 3.2).

*Proof of Theorem 4.4.* We prove the theorem via a sequence of games, formally given in Figure 7. Following [21, 34, 19], we assume that the game has access to some internal quantum random oracles (QROs) which are used to simulate the QROs accessed by the adversary. Namely, let  $h', h'_{\text{pk}_1}, \dots, h'_{\text{pk}_N} : \mathcal{C}' \rightarrow \mathcal{K}$  be internal QROs used to simulate  $\text{H}'$ ,  $h, h_{\text{pk}_1}, \dots, h_{\text{pk}_N} : \mathcal{C}' \rightarrow \mathcal{K}$  be internal QROs used to simulate  $\text{H}$ , and  $g, g'_{\text{pk}_1}, \dots, g'_{\text{pk}_N} : \mathcal{M}' \rightarrow \mathcal{R}'$  be internal QROs used to simulate  $\text{G}$ . Such internal QROs can be simulated by several ways [34], e.g., using  $2q$ -wise independent hash function (if the adversary queries the QRO at most  $q$  times) [37]. For sake of simplicity, during all our security games, we implicitly exclude collisions of users' public keys  $\text{pk}_i$ 's and secret keys  $k_i$ 's for implicit rejection and the collisions of the PKE messages  $r_{j,i}$ 's, randomnesses  $R_{j,i}$ 's, and KEM keys  $K_{j,i}$ . Excluding such collisions will add

$$N^2S^2 \left( \frac{1}{|\mathcal{M}'|} + \frac{1}{|\mathcal{K}|} + \frac{1}{|\mathcal{R}'|} + \frac{1}{2^s} + \eta_{\text{PKE}} \right)$$

to the final bound. In  $\mathbf{G}_0$ , we use  $g, h'$ , and  $h$  to simulate  $\text{G}, \text{H}', \text{H}$ , respectively. This game is equivalent to  $\text{MUC-IND-CCA}_{\text{KEM}_2,0}^{\text{A}}$  game (Definition 4.2), so we have

$$\Pr \left[ \text{MUC-IND-CCA}_{\text{KEM}_2,0}^{\text{A}} \Rightarrow 1 \right] = \Pr \left[ \mathbf{G}_0^{\text{A}} \Rightarrow 1 \right].$$

$\mathbf{G}_1$ : If  $\mathcal{A}$  queries  $\text{DEC}_{\text{mu}}$  on  $(j, e)$  that  $e$  is invalid, then  $\text{DEC}_{\text{mu}}$  returns  $h'_{\text{pk}_j}(e)$  instead of  $\text{H}'(\text{pk}_j, k_j, e)$ . We use Lemma 2.3 to bound the difference. Concretely, we apply Lemma 2.3 for any user  $j \in [N]$ , by viewing  $\text{H}'(\text{pk}_j, \cdot)$  as oracle  $h$  in Lemma 2.3, and  $h'_{\text{pk}_j}$  as oracle  $h'$  in Lemma 2.3. Thus, we have

$$\left| \Pr \left[ \mathbf{G}_0^{\text{A}} \Rightarrow 1 \right] - \Pr \left[ \mathbf{G}_1^{\text{A}} \Rightarrow 1 \right] \right| \leq 2Nq_{\text{H}'} \cdot 2^{-s/2}.$$

<b>Game <math>\mathbf{G}_0\text{-}\mathbf{G}_9</math></b>	<b>Oracle <math>\text{DEC}_{\text{mu}}(j, e)</math></b>
01 $\text{par} \leftarrow \text{Setup}(\lambda)$	20 <b>if</b> $e \in \mathbf{e}[j, \cdot]$
02 $\text{par} := \text{lpar} \leftarrow \text{LSetup}(\lambda)$ <span style="float: right;">// <math>\mathbf{G}_6\text{-}\mathbf{G}_9</math></span>	21 <b>return</b> $\perp$
03 <b>for</b> $j \in [N]$	22 $r' := \text{Dec}(\text{sk}_j, e)$
04 $(\text{pk}_j, (\text{sk}_j, k_j)) \leftarrow \text{KG}(\text{par})$	23 <b>if</b> $r' = \perp \vee c \neq \text{Enc}(\text{pk}_j, r'; \mathbf{G}(\text{pk}_j, r'))$
05 $(\text{lpk}_j, \text{lsk}_j) \leftarrow \text{LKG}(\text{par})$ <span style="float: right;">// <math>\mathbf{G}_6\text{-}\mathbf{G}_7</math></span>	24 $K := \text{H}'(\text{pk}_j, k_j, e)$
06 $(\text{pk}_j, \text{sk}_j) := (\text{lpk}_j, \text{lsk}_j)$ <span style="float: right;">// <math>\mathbf{G}_6\text{-}\mathbf{G}_7</math></span>	25 $K := \text{h}'_{\text{pk}_j}(e)$ <span style="float: right;">// <math>\mathbf{G}_1\text{-}\mathbf{G}_2</math></span>
07 <b>for</b> $i \in [S]$	26 <b>else</b>
08 $r_{j,i} \xleftarrow{\$} \mathcal{M}'$	27 $K := \text{H}(\text{pk}_j, r', e)$
09 $R_{j,i} := \mathbf{G}(\text{pk}_j, r_{j,i})$	28 $K := \text{h}_{\text{pk}_j}(e)$ <span style="float: right;">// <math>\mathbf{G}_3</math></span>
10 $R_{j,i} \xleftarrow{\$} \mathcal{R}'$ <span style="float: right;">// <math>\mathbf{G}_7\text{-}\mathbf{G}_9</math></span>	29 $K := \text{h}_{\text{pk}_j}(e)$ <span style="float: right;">// <math>\mathbf{G}_4\text{-}\mathbf{G}_9</math></span>
11 $e := \text{Enc}(\text{pk}_j, r_{j,i}; R_{j,i})$	30 <b>return</b> $K$
12 $K_{j,i} := \text{H}(\text{pk}_j, r_{j,i}, e)$	<b>Oracle <math>\mathbf{G}(\text{pk}, r)</math></b>
13 $K_{j,i} := \text{h}_{\text{pk}_j}(e)$ <span style="float: right;">// <math>\mathbf{G}_3\text{-}\mathbf{G}_9</math></span>	31 <b>if</b> $\text{pk} \in \mathbf{pk}$ <span style="float: right;">// <math>\mathbf{G}_2\text{-}\mathbf{G}_4, \mathbf{G}_8\text{-}\mathbf{G}_9</math></span>
14 $K_{j,i} \xleftarrow{\$} \mathcal{K}$ <span style="float: right;">// <math>\mathbf{G}_9</math></span>	32 <b>return</b> $\mathbf{g}'_{\text{pk}}(r)$ <span style="float: right;">// <math>\mathbf{G}_2\text{-}\mathbf{G}_4, \mathbf{G}_8\text{-}\mathbf{G}_9</math></span>
15 $\mathbf{e}[j, i] := e, \mathbf{K}[j, i] := K_{j,i}$	33 <b>return</b> $\mathbf{g}(\text{pk}, r)$
16 $\mathbf{pk}[j] := \text{pk}_j$	<b>Oracle <math>\text{H}(\text{pk}, r, e)</math></b>
17 $b' \leftarrow \mathcal{A}^{\text{DEC}_{\text{mu}}, \text{H}, \text{H}', \text{G}}(\text{par}, \mathbf{pk}, \mathbf{e}, \mathbf{K})$	34 <b>if</b> $\text{pk} \in \mathbf{pk}$ <span style="float: right;">// <math>\mathbf{G}_3\text{-}\mathbf{G}_9</math></span>
18 <b>return</b> $b'$	35 $\wedge e = \text{Enc}(\text{pk}, r; \mathbf{G}(\text{pk}, r))$ <span style="float: right;">// <math>\mathbf{G}_3\text{-}\mathbf{G}_9</math></span>
<b>Oracle <math>\text{H}'(\text{pk}, k, e)</math></b>	36 <b>return</b> $\text{h}_{\text{pk}}(e)$ <span style="float: right;">// <math>\mathbf{G}_3\text{-}\mathbf{G}_9</math></span>
19 <b>return</b> $\text{h}'(\text{pk}, k, e)$	37 <b>return</b> $\text{h}(\text{pk}, r, e)$

Figure 7: Games sequence  $\mathbf{G}_0\text{-}\mathbf{G}_9$  in the proof of Theorem 4.4. Highlighted lines are only executed in the corresponding games.

$\mathbf{G}_2$ : The image set of  $\mathbf{G}(\text{pk}_j, \cdot)$  is restricted to be the set only containing “good” randomnesses of  $\text{pk}_j$ . Namely, for  $j \in [N]$ , we define the set

$$\mathcal{R}'_{\text{bad}}(\text{pk}_j, \text{sk}_j, r) := \{R' \in \mathcal{R}' \mid \text{Dec}(\text{sk}, \text{Enc}(\text{pk}, r; R')) \neq r\}$$

which denotes the “bad” randomness with respect to  $(\text{pk}_j, \text{sk}_j)$  and  $r$ . And we similarly define the “good” randomness set as  $\mathcal{R}'_{\text{good}}(\text{pk}_j, \text{sk}_j, r) := \mathcal{R}' \setminus \mathcal{R}'_{\text{bad}}(\text{pk}_j, \text{sk}_j, r)$ . and let  $\mathbf{g}'_{\text{pk}_j} : \mathcal{M}' \rightarrow \mathcal{R}'$  be a quantum-accessible random oracle such that for any  $r \in \mathcal{M}'$ ,  $\mathbf{g}'_{\text{pk}_j}(r)$  is sampled uniformly from  $\mathcal{R}'_{\text{good}}(\text{pk}_j, \text{sk}_j, r)$ .

We use Lemma 2.4 to bound the probability difference between  $\mathbf{G}_1$  and  $\mathbf{G}_2$ . The proof method here is similar to the one in [19, Theorem 2]. We define

$$\begin{aligned} \delta(\text{pk}_j, \text{sk}_j, r) &:= |\mathcal{R}'_{\text{bad}}(\text{pk}_j, \text{sk}_j, r)| / |\mathcal{R}'|, \\ \delta(\text{pk}_j, \text{sk}_j) &:= \max_{r \in \mathcal{M}'} \delta(\text{pk}_j, \text{sk}_j, r), \end{aligned}$$

and by these notations, if PKE is  $(1 - \delta)$ -correct, then  $\delta = \mathbb{E}[\delta(\text{pk}_j, \text{sk}_j)]$  where the expectation is taken over  $(\text{pk}, \text{sk}) \leftarrow \text{KG}$ .

Here we construct unbounded adversaries  $\mathcal{B}_j$  for  $1 \leq j \leq N$  that run in game  $\text{GDPB}_{\delta, b}$  ( $b = 0$  or  $b = 1$ ). For any such  $j$ ,  $\mathcal{B}_j$  first generates  $(\text{pk}_i, \text{sk}_i, k_i)$  for  $i \in [N]$  as in  $\mathbf{G}_1$  and picks a random function  $f$  (domain and range will be clear later), and then it sets  $\lambda_r := \delta(\text{pk}_j, \text{sk}_j, r)$  for all  $r \in \mathcal{M}'$  and outputs  $(\lambda_r)_{r \in \mathcal{M}'}$ .

Then,  $\mathcal{B}_j$  has quantum access to a function  $\text{F}$  (provided by  $\text{GDPB}_{\delta, b}$ ). It sets up the oracle  $\mathbf{G}(\text{pk}, \cdot)$  such that

$$\mathbf{G}'(\text{pk}, r) = \begin{cases} \text{Samp}(\mathcal{R}'_{\text{good}}(\text{pk}, \text{sk}, r); f(\text{pk}, r)) & \text{if } \text{pk} \in \{\text{pk}_1, \dots, \text{pk}_{j-1}\}, \\ \text{Samp}(\mathcal{R}'_{\text{good}}(\text{pk}_j, \text{sk}_j, r); f(\text{pk}_j, r)), & \text{if } \text{pk} = \text{pk}_j \wedge \text{F}(r) = 0 \\ \text{Samp}(\mathcal{R}'_{\text{bad}}(\text{pk}_j, \text{sk}_j, r); f(\text{pk}_j, r)), & \text{if } \text{pk} = \text{pk}_j \wedge \text{F}(r) = 1 \\ \mathbf{g}(\text{pk}, r), & \text{Otherwise} \end{cases}$$

and uses such  $\mathbf{G}'$  to simulate  $\mathbf{G}_1$  for  $\mathcal{A}$  (namely, it replaces  $\mathbf{G}(\text{pk}_j, \cdot)$  by  $\mathbf{G}'(\text{pk}_j, \cdot)$ , and other oracles like  $\text{H}$  and  $\text{DEC}$  are the same as in  $\mathbf{G}_1$ ) and outputs  $\mathcal{A}$ 's final output. Here  $\text{Samp}$  is a sampling process and  $f$  is

used to generate randomness for **Samp** (so that it can sample elements from a set uniformly at random). Since  $\mathcal{B}_j$  is unbounded, it can construct such  $f$  and **Samp**.

If  $\mathcal{B}_j$  is playing  $\text{GDPB}_{\delta,0}$ , then  $F(r)$  always outputs 0, and then  $G'(\text{pk}_j, r) = \mathbf{g}'_{\text{pk}_j}(r)$  in  $\mathbf{G}_2$ . If  $\mathcal{B}_j$  is playing  $\text{GDPB}_{\delta,1}$ , then  $F(r)$  outputs 1 with probability  $\delta(\text{pk}_j, \text{sk}_j, r)$  and then  $G'(\text{pk}_j, r)$  is distributed identically with  $G(\text{pk}_j, r)$  (and  $\mathbf{g}(\text{pk}_j, r)$ ) in  $\mathbf{G}_1$ .

We further let  $\text{Hyb}_j$  for  $0 \leq j \leq N$  be a hybrid game which is almost the same as  $\mathbf{G}_1$  except that, for users  $j+1$  to  $N$ , we use  $\mathbf{g}'_{\text{pk}_{j+1}}, \dots, \mathbf{g}'_{\text{pk}_N}$  to simulate  $G(\text{pk}_{j+1}, \cdot), \dots, G(\text{pk}_N, \cdot)$ , respectively. By definition,  $\text{Hyb}_0 = \mathbf{G}_2$  and  $\text{Hyb}_N = \mathbf{G}_1$ . By the construction of  $\mathcal{B}_j$ , if  $\mathcal{B}_j$  plays  $\text{GDPB}_{\delta,1}$ , then it simulates  $\text{Hyb}_{j-1}$  for  $\mathcal{A}$ . If  $\mathcal{B}_j$  plays  $\text{GDPB}_{\delta,0}$ , then it simulates  $\text{Hyb}_j$  for  $\mathcal{A}$ . We can use  $\mathcal{B}_j$  for each  $j \in [N]$  described above to bound the probabilities difference between  $\text{Hyb}_{j-1}$  and  $\text{Hyb}_j$ . Namely, using Lemma 2.4, we have

$$\begin{aligned} |\Pr[\mathbf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1]| &\leq |\Pr[\text{Hyb}_N^{\mathcal{A}} \Rightarrow 1] - \Pr[\text{Hyb}_0^{\mathcal{A}} \Rightarrow 1]| \\ &\leq \sum_{j=1}^N |\Pr[\text{Hyb}_j^{\mathcal{A}} \Rightarrow 1] - \Pr[\text{Hyb}_{j-1}^{\mathcal{A}} \Rightarrow 1]| \\ &\leq \sum_{j=1}^N \left| \Pr[\text{GDPB}_{\delta,0}^{\mathcal{B}_j} \Rightarrow 1] - \Pr[\text{GDPB}_{\delta,1}^{\mathcal{B}_j} \Rightarrow 1] \right| \\ &\leq N \cdot (\delta + 8(q_{\text{H}} + q_{\text{G}} + 2q_{\text{DEC}} + S)^2 \delta) \\ &= 8N(1 + (q_{\text{H}} + q_{\text{G}} + 2q_{\text{DEC}} + S)^2) \delta \end{aligned}$$

by Lemma 2.4 and  $\mathcal{B}_j$  issuing  $(q_{\text{H}} + q_{\text{G}} + 2q_{\text{DEC}} + S)$  queries to  $F$ . The additional  $\delta$  appears in the next to last equation is from the probability that a bad key pair with no good randomness [35, 28]. For simplicity, we add the error bound  $\delta$  here and exclude the event that **KG** outputs such bad key pair.

$\mathbf{G}_3$ : In this game, we start to get rid of the secret key by using the “encrypt-then-hash” technique [21, 34, 19]. When  $\mathcal{A}$  queries  $H(\text{pk}, r, e)$  where  $\text{pk} = \text{pk}_j$  for some  $j \in [N]$  and  $e = \text{Enc}(\text{pk}_j, r; G(\text{pk}_j, r))$ , instead of returning  $h(\text{pk}, r, e)$ , the game returns  $h_{\text{pk}_j}(e)$  (see Items 34 to 36). For consistency, we also change the generation of challenge KEM keys (in Item 13) and  $\text{DEC}_{\text{mu}}$  (in Item 28), since if  $e = \text{Enc}(\text{pk}_j, r; G(\text{pk}_j, r))$  then  $H(\text{pk}_j, r, e) = h_{\text{pk}_j}(e)$ .

We claim that  $\mathcal{A}$ 's views in  $\mathbf{G}_2$  and  $\mathbf{G}_3$  are the same. This is because, starting from  $\mathbf{G}_2$ ,  $G(\text{pk}_j, \cdot)$  always uses “good” randomness, which implies that the map  $\text{Enc}(\text{pk}_j, \cdot; G(\text{pk}_j, \cdot))$  is injective and thus  $H(\text{pk}_j, \cdot, \text{Enc}(\text{pk}_j, \cdot; G(\text{pk}_j, \cdot)))$  behaves as a random oracle. We have

$$\Pr[\mathbf{G}_2^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathbf{G}_3^{\mathcal{A}} \Rightarrow 1].$$

$\mathbf{G}_4$ : We change  $\text{DEC}_{\text{mu}}$  such that, on query  $(j, e)$ , it always returns  $h_{\text{pk}_j}(e)$  regardless of the validity of  $e$  (Item 29). We argue that this change does not affect  $\mathcal{A}$ 's view: On query  $(j, e)$ , if  $e$  is a valid ciphertext with respect to  $\text{pk}_j$ , then  $\text{DEC}_{\text{mu}}$  returns  $h_{\text{pk}_j}(e)$  in both two games; If  $e$  is invalid (its decryption is  $\perp$  or it cannot pass the re-encryption checking), then in  $\mathbf{G}_3$ ,  $\text{DEC}$  returns  $h'_{\text{pk}_j}(e)$ , which is an independently random key ( $h'_{\text{pk}_j}$  is an internal RO). Moreover, if  $e$  is invalid,  $h_{\text{pk}_j}(e)$  is also independently random by the definition of  $H$  ( $\mathcal{A}$  cannot learn  $h_{\text{pk}_j}(e)$  from  $H$  when  $e$  is invalid). Therefore, when  $e$  is invalid with respect to  $\text{pk}_j$ ,  $h_{\text{pk}_j}(e)$  has the same distribution with  $h'_{\text{pk}_j}(e)$ , which means that the modification made by  $\mathbf{G}_4$  does not change  $\mathcal{A}$ 's view. We have

$$\Pr[\mathbf{G}_3^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathbf{G}_4^{\mathcal{A}} \Rightarrow 1].$$

$\mathbf{G}_5$ : We switch back to using  $\mathbf{g}$  to simulate  $G$  instead of using  $(\mathbf{g}'_{\text{pk}_1}, \dots, \mathbf{g}'_{\text{pk}_N})$ . Similar to the gamehop from  $\mathbf{G}_1$  to  $\mathbf{G}_2$ , we have

$$|\Pr[\mathbf{G}_4^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_5^{\mathcal{A}} \Rightarrow 1]| \leq 8N(1 + (q_{\text{H}} + q_{\text{G}} + 2q_{\text{DEC}} + S)^2) \delta.$$

Observe that in  $\mathbf{G}_5$ , we do not need to use  $\text{sk}_j$  to simulate  $\text{DEC}_{\text{mu}}(j, \cdot)$  (where  $j \in [N]$ ). From  $\mathbf{G}_6$  to  $\mathbf{G}_8$ , we start to use the properties of parameter lossy encryption PKE to finish the proof.

<b>Reduction</b> $\mathcal{B}(\text{par}, \text{pk}_1, \dots, \text{pk}_N)$	<b>Oracle</b> $\text{DEC}_{\text{mu}}(j, e)$
01 <b>for</b> $j \in [N]$	13 <b>if</b> $e \in \mathbf{e}$ : <b>return</b> $\perp$
02 $k_j \leftarrow \{0, 1\}^s$	14 <b>return</b> $K := \text{h}_{\text{pk}_j}(e)$
03 <b>for</b> $i \in [S]$	<b>Oracle</b> $\text{G}(\text{pk}, r)$
04 $r_{j,i} \xleftarrow{\$} \mathcal{M}'$	15 <b>return</b> $\text{g}(\text{pk}, r)$
05 $R_{j,i} := \text{G}(\text{pk}_j, r_{j,i})$	<b>Oracle</b> $\text{H}(\text{pk}, r, e)$
06 $e := \text{Enc}(\text{pk}_j, r_{j,i}; R_{j,i})$	16 <b>if</b> $\text{pk} \in \mathbf{pk} \wedge e = \text{Enc}(\text{pk}, r; \text{G}(\text{pk}, r))$
07 $K := \text{h}_{\text{pk}_j}(e)$	17 <b>return</b> $\text{h}_{\text{pk}}(e)$
08 $\mathbf{e}[j, i] := e$	18 <b>return</b> $\text{h}(\text{pk}, r, e)$
09 $\mathbf{K}[j, i] := K$	<b>Oracle</b> $\text{H}'(\text{pk}, k, e)$
10 $\mathbf{pk}[j] := \text{pk}_j$	19 <b>return</b> $\text{h}'(\text{pk}, k, e)$
11 $b' \leftarrow \mathcal{A}^{\text{DEC}_{\text{mu}}, \text{H}, \text{H}', \text{G}}(\text{par}, \mathbf{pk}, \mathbf{e}, \mathbf{K})$	
12 <b>return</b> $b'$	

Figure 8: Adversary  $\mathcal{B}$  in bounding  $\mathbf{G}_5$  and  $\mathbf{G}_6$ .

$\mathbf{G}_6$ : We switch the parameter and public keys of PKE to the lossy mode, namely, the parameter  $\text{par}$  in  $\mathbf{G}_6$  is generated by LSetup (Item 02) and the public keys in  $\mathbf{G}_6$  are generated by LKG (Items 05 to 06).

We construct a reduction  $\mathcal{B}$  against the parameter-key indistinguishability of PKE in Figure 8.  $\mathcal{B}$ 's input  $(\text{par}, \text{pk}_1, \dots, \text{pk}_N)$  is from Setup and KG, then  $\mathcal{B}$  perfectly simulates  $\mathbf{G}_5$  for  $\mathcal{A}$ . If  $(\text{par}, \text{pk}_1, \dots, \text{pk}_N)$  is from LSetup and LKG, then  $\mathcal{B}$  perfectly simulates  $\mathbf{G}_6$  for  $\mathcal{A}$ . Moreover,  $\mathcal{B}$  outputs  $\mathcal{A}$ 's final output. So, we have

$$|\Pr[\mathbf{G}_5^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_6^{\mathcal{A}} \Rightarrow 1]| \leq \text{Adv}_{\text{PKE}}^{\text{ind-par-key}}(\mathcal{B}).$$

$\mathbf{G}_7$ : The randomness  $R_{j,i}$  of challenge ciphertext  $\mathbf{e}[j, i]$  is generated by independently uniform sampling from  $\mathcal{R}'$  instead of by using  $\text{G}$  (see Item 10).

In  $\mathbf{G}_6$ , we always have  $R_{j,i} = \text{G}(\text{pk}_j, r_{j,i})$  for all  $(j, i) \in [N] \times [S]$ , while in  $\mathbf{G}_7$ ,  $R_{j,i}$ 's are independent of  $\text{G}$ . Despite these  $r_{j,i}$ 's and  $R_{j,i}$ 's, oracle  $\text{G}$  behaves the same in  $\mathbf{G}_6$  and  $\mathbf{G}_7$ . Let  $\mathcal{O}_0$  be the oracle  $\text{G}$  in  $\mathbf{G}_6$  and let  $\mathcal{O}_1$  be the oracle  $\text{G}$  in  $\mathbf{G}_7$ , then we have  $\mathcal{O}_0 \setminus \mathcal{S} = \mathcal{O}_1 \setminus \mathcal{S}$ , where  $\mathcal{S}$  is defined as follows:

$$\mathcal{S} := \{(\text{pk}_1, r_{1,1}), (\text{pk}_1, r_{1,2}), \dots, (\text{pk}_j, r_{j,i}), \dots, (\text{pk}_N, r_{N,S})\}, \text{ and } |\mathcal{S}| = NS$$

We use Lemma 2.1 to bound the difference between  $\mathbf{G}_6$  with  $\mathbf{G}_7$ . By this lemma, there is an algorithm EXT captures the probability that  $\mathcal{A}$  “learns”  $r_{j,i} \in \mathcal{S}$ . However, this probability cannot be directly bounded since  $e_{j,i}$  is still related to  $r_{j,i}$ . To deal with it, we use delayed analysis. Looking ahead, we will firstly switch all challenge ciphertexts to other challenge ciphertexts that are independent of  $r_{j,i}$ 's (by using the lossiness of PKE) so that  $r_{j,i}$ 's are independently and uniformly random in  $\mathcal{A}$ 's view, so we can bound the winning probability of EXT and thus can bound  $|\Pr[\mathbf{G}_6^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_7^{\mathcal{A}} \Rightarrow 1]|$ . For readability, we continue the proof of Theorem 4.4 and leave these arguments as a lemma which will be proved later.

**Lemma 4.5** *With notations and assumptions from  $\mathbf{G}_6$  and  $\mathbf{G}_7$  in the proof of Theorem 4.4, we have*

$$|\Pr[\mathbf{G}_6^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_7^{\mathcal{A}} \Rightarrow 1]| \leq 2(q_G + q_H + NS) \sqrt{NS \cdot \epsilon_{\text{PKE}}^{\text{lo}} + NS/|\mathcal{M}'|}.$$

$\mathbf{G}_8$ : We switch the parameter and public keys of PKE to the normal mode (namely, parameter and public keys are generated by Setup and KG). Moreover, we restrict the image of  $\text{G}(\text{pk}_j, \cdot)$  to be the set only containing “good” randomnesses of  $\text{pk}_j$  (as we did in  $\mathbf{G}_2$ ). Similar to the game hops from  $\mathbf{G}_5$  to  $\mathbf{G}_6$  and from  $\mathbf{G}_1$  to  $\mathbf{G}_2$ , there exists an adversary  $\mathcal{B}$  such that

$$\begin{aligned} & |\Pr[\mathbf{G}_7^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_8^{\mathcal{A}} \Rightarrow 1]| \\ & \leq \text{Adv}_{\text{PKE}}^{\text{ind-par-key}}(\mathcal{B}) + 8N(1 + (q_H + q_G + 2q_{\text{DEC}} + S)^2)\delta. \end{aligned}$$

$\mathbf{G}_9$ : We change the generation of challenge KEM keys  $K_{j,i}$ 's. In this game, we generate  $K_{j,i} \xleftarrow{\$} \mathcal{K}$  instead of  $K_{j,i} := \text{h}_{\text{pk}_j}(e_{j,i})$  where  $e_{j,i} = \mathbf{e}[j, i]$ . Since  $\text{h}_{\text{pk}_j}$ 's are internal QROs,  $\mathcal{A}$  cannot trivially detect this modification.

We claim that this modification does not change  $\mathcal{A}$ 's view except with negligible probability. Here we firstly analyze the information about  $h_{pk_j}(e_{j,i})$  for  $(j,i) \in [N] \times [S]$  that  $\mathcal{A}$  can learn from its oracle queries.

- Oracles G and H': These two oracles do not reveal any information about  $h_{pk_j}(e_{j,i})$  since they are independent to each other in both  $\mathbf{G}_7$  and  $\mathbf{G}_8$ .
- Oracle DEC: If  $\mathcal{A}$  queries  $\text{DEC}_{\text{mu}}(j, e_{j,i})$  for  $(j,i) \in [N] \times [S]$ , then by the definitions of  $\text{DEC}_{\text{mu}}$  in  $\mathbf{G}_7$  and  $\mathbf{G}_8$ , the oracle always returns  $\perp$ ; Otherwise,  $\text{DEC}_{\text{mu}}$  returns a key that is independent to  $h_{pk_j}(e_{j,1}), \dots, h_{pk_j}(e_{j,S})$  (since  $h_{pk_j}$  is a QRO). So, DEC does not reveal any information about  $h_{pk_j}(e_{j,i})$ .
- Oracle H:  $\mathcal{A}$  learns  $h_{pk_j}(e_{j,i})$  if it queries  $\text{H}(pk_j, r, e_{j,i})$  such that  $e_{j,i} = \text{Enc}(pk_j, r; R)$  where  $R = \text{G}(pk_j, r)$ . Since in  $\mathbf{G}_8$ , we already restricted  $\text{G}(pk_j, \cdot)$  to always output good randomness,  $\text{Enc}(pk_j, \cdot; \text{G}(pk_j, \cdot))$  is injective and  $e_{j,i} = \text{Enc}(pk_j, r; R)$  means that  $(r, R) = (r_{j,i}, R_{j,i})$ . Since  $\text{G}(pk_j, \cdot)$  is random oracle and  $R_{j,i}$  is sampled at uniformly random from  $\mathcal{R}'$ , we have

$$\Pr [R_{j,i} = \text{G}(pk_j, r_{j,i})] = 1/|\mathcal{R}'|.$$

Since there are  $NS$  randomnesses, by a union bound, we have

$$|\Pr [\mathbf{G}_8^{\mathcal{A}} \Rightarrow 1] - \Pr [\mathbf{G}_9^{\mathcal{A}} \Rightarrow 1]| \leq NS/|\mathcal{R}'|.$$

In  $\mathbf{G}_9$ , the KEM keys  $K_{j,i}$ 's are generated at independently and uniformly random. We can undo the modifications made in  $\mathbf{G}_8, \dots, \mathbf{G}_1$  to achieve the game  $\text{MUC-IND-CCA}_{\text{KEM},1}^{\mathcal{A}}$ . We have

$$\begin{aligned} & |\Pr[\mathbf{G}_9^{\mathcal{A}} \Rightarrow 1] - \Pr[\text{MUC-IND-CCA}_{\text{KEM},1}^{\mathcal{A}} \Rightarrow 1]| \\ & \leq 24N(1 + (q_{\text{H}} + q_{\text{G}} + 2q_{\text{DEC}} + S)^2)\delta + 2Nq_{\text{H}'} \cdot 2^{-s/2} \\ & \quad + 2(q_{\text{G}} + q_{\text{H}} + NS)\sqrt{NS \cdot \epsilon_{\text{PKE}}^{\text{lo}} + NS/|\mathcal{M}'|} + 2\text{Adv}_{\text{PKE}}^{\text{ind-par-key}}(\mathcal{B}). \end{aligned}$$

Combining all the probability differences in the games sequence, we have

$$\begin{aligned} & |\Pr[\text{MUC-IND-CCA}_{\text{KEM},0}^{\mathcal{A}} \Rightarrow 1] - \Pr[\text{MUC-IND-CCA}_{\text{KEM},1}^{\mathcal{A}} \Rightarrow 1]| \\ & \leq 4\text{Adv}_{\text{PKE}}^{\text{ind-par-key}}(\mathcal{B}) + 48N(1 + (q_{\text{H}} + q_{\text{G}} + 2q_{\text{DEC}} + S)^2)\delta \\ & \quad + \frac{NS + N^2S^2}{|\mathcal{R}'|} + N^2S^2\left(\frac{1}{|\mathcal{M}'|} + \frac{1}{|\mathcal{K}|} + \frac{1}{2^s} + \eta_{\text{PKE}}\right) \\ & \quad + 4(q_{\text{G}} + q_{\text{H}} + NS)\sqrt{NS \cdot \epsilon_{\text{PKE}}^{\text{lo}} + NS/|\mathcal{M}'|} + 4Nq_{\text{H}'} \cdot 2^{-s/2}, \end{aligned}$$

as stated in Theorem 4.4. □

*Proof of Lemma 4.5.* In the gamehop from  $\mathbf{G}_6$  to  $\mathbf{G}_7$  in the proof of Theorem 4.4 in Section 4.2, we argued that if  $\mathcal{A}$  plays  $\mathbf{G}_6$  then  $\mathcal{A}$  is interacting with the oracle  $\mathcal{O}_0$ , and if  $\mathcal{A}$  plays  $\mathbf{G}_7$  then  $\mathcal{A}$  is interacting with oracle  $\mathcal{O}_1$ , where  $\mathcal{O}_0$  is the oracle G in  $\mathbf{G}_6$ ,  $\mathcal{O}_1$  is the oracle G in  $\mathbf{G}_7$ ,  $\mathcal{O}_0 \setminus \mathcal{S} = \mathcal{O}_1 \setminus \mathcal{S}$ , and  $\mathcal{S}$  is defined as follows:

$$\mathcal{S} := \{(pk_1, r_{1,1}), (pk_1, r_{1,2}), \dots, (pk_j, r_{j,i}), \dots, (pk_N, r_{N,S})\}.$$

We can view  $\mathcal{O}_0$  and  $\mathcal{O}_1$  as follows:

$$\mathcal{O}_0(pk, r) = \begin{cases} R_{j,i}, & \text{if } \exists(j,i) \in [N] \times [S] \\ & \text{s.t. } (pk, r) = (pk_{j,i}, r_{j,i}), \quad \mathcal{O}_1(pk, r) = g(pk, r). \\ g(pk, r), & \text{Otherwise} \end{cases}$$

Therefore, we can also view the game environment of  $\mathbf{G}_6$  is the same as the one of  $\mathbf{G}_7$ , except that the oracles G in these two games are different. That is, we can view  $R_{j,i}$ 's in  $\mathbf{G}_6$  are also generated by independently and uniformly sampling, but then G is set up such that  $\text{G}(pk_j, r_{j,i}) := R_{j,i}$ . While in  $\mathbf{G}_7$ ,

<p><b>Game <math>\mathbf{G}'_7</math> and <math>\mathbf{G}''_7</math></b></p> <pre> 01 lpar <math>\leftarrow</math> LSetup(<math>\lambda</math>), par := lpar 02 for <math>j \in [N]</math> 03   (lpk<math>_j</math>, lsk<math>_j</math>) <math>\leftarrow</math> LKG(par) 04   (pk<math>_j</math>, sk<math>_j</math>) := (lpk<math>_j</math>, lsk<math>_j</math>) 05   for <math>i \in [S]</math> 06     <math>r_{j,i} \xleftarrow{\\$} \mathcal{M}'</math>, <math>R_{j,i} \xleftarrow{\\$} \mathcal{R}'</math> 07     <math>e := \text{Enc}(\text{pk}_j, r_{j,i}; R_{j,i})</math> 08     <math>r'_{j,i} \xleftarrow{\\$} \mathcal{M}'</math>, <math>R'_{j,i} \xleftarrow{\\$} \mathcal{R}'</math> 09     <math>e := \text{Enc}(\text{pk}_j, r'_{j,i}; R'_{j,i})</math> 10     <math>\mathbf{e}[j, i] := e</math>, <math>\mathbf{K}[j, i] := \text{h}_{\text{pk}_j}(e)</math> 11   pk[j] := pk<math>_j</math> 12 (pk, r) <math>\leftarrow</math> EXT<sup>A,  G , DEC<sub>mu</sub>,  H ,  H' </sup>(par, pk, e, K) 13 return (pk, r) </pre> <p><b>Oracle <math>\mathbf{H}'(\text{pk}, k, e)</math></b></p> <pre> 14 return h'(pk, k, e) </pre>	<p><b>Oracle <math>\text{DEC}_{\text{mu}}(j, e)</math></b></p> <pre> 15 if <math>e \in \mathbf{e}[j, \cdot]</math> 16   return <math>\perp</math> 17 <math>K := \text{h}_{\text{pk}_j}(e)</math> 18 return K </pre> <p><b>Oracle <math>\mathbf{G}(\text{pk}, r)</math></b></p> <pre> 19 return g(pk, r) </pre> <p><b>Oracle <math>\mathbf{H}(\text{pk}, r, e)</math></b></p> <pre> 20 if pk <math>\in</math> pk 21   <math>\Delta e = \text{Enc}(\text{pk}, r; \mathbf{G}(\text{pk}, r))</math> 22   return h<sub>pk</sub>(e) 23 return h(pk, r, e) </pre>
--	--

Figure 9: Games  $\mathbf{G}'_7$  and  $\mathbf{G}''_7$  in the proof of Lemma 4.5. Highlighted lines are only executed in the corresponding games.

we do not change  $\mathbf{G}$ . So,  $\mathbf{G}_6^{\mathcal{A}}$  is equivalent to  $\mathcal{A}$  plays  $\mathbf{G}_7$  but the oracle  $\mathbf{G}$  it interacts with is  $\mathcal{O}_0$ . And thus we have

$$\begin{aligned}
& |\Pr[\mathbf{G}_6^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_7^{\mathcal{A}} \Rightarrow 1]| \\
&= |\Pr[1 \leftarrow \mathcal{A}^{\mathcal{O}_0} : \mathbf{G}_7] - \Pr[1 \leftarrow \mathcal{A}^{\mathcal{O}_1} : \mathbf{G}_7]|.
\end{aligned}$$

Here we ignore other oracles that  $\mathcal{A}$  can access, since such oracles,  $\text{DEC}$ ,  $\mathbf{H}'$ , and  $\mathbf{H}$ , are either independent of  $\mathbf{G}$  or can be simulated by querying  $\mathbf{G}$ .

In  $\mathbf{G}_7$ ,  $\mathcal{A}$  issues at most  $(q_{\mathbf{G}} + q_{\mathbf{H}})$  queries to  $\mathbf{G}$ . By using Lemma 2.1, there exists  $\text{EXT}$  such that

$$\begin{aligned}
& |\Pr[1 \leftarrow \mathcal{A}^{\mathcal{O}_0} : \mathbf{G}_7] - \Pr[1 \leftarrow \mathcal{A}^{\mathcal{O}_1} : \mathbf{G}_7]| \\
&\leq 2(q_{\mathbf{G}} + q_{\mathbf{H}}) \sqrt{\Pr[(\text{pk}, r) \in \mathcal{S} : (\text{pk}, r) \leftarrow \text{EXT in } \mathbf{G}'_7]}
\end{aligned}$$

where  $\mathbf{G}'_7$  is defined in Figure 9 and  $(\text{pk}, r) \in \mathcal{S}$  means that there exists  $(j, i) \in [N] \times [S]$  such that  $(\text{pk}, r) = (\text{pk}_{j,i}, r_{j,i})$  (i.e.,  $\text{EXT}$  finds out one of  $r_{j,i}$ 's in  $\mathbf{G}'_7$ ).  $\mathbf{G}'_7$  has identical structure with  $\mathbf{G}_7$ , and the only difference is that  $\mathbf{G}'_7$  is defined for  $\text{EXT}$ , since by definitions in Lemma 2.1,  $\text{EXT}$  plays the same game with  $\mathcal{A}$  and it randomly measures  $\mathcal{A}$ 's QRO queries and outputs the measurement outcome.

Here we bound  $\Pr[(\text{pk}, r) \in \mathcal{S} : (\text{pk}, r) \leftarrow \text{EXT in } \mathbf{G}'_7]$ . We use an auxiliary game  $\mathbf{G}''_7$ , which is almost the same as  $\mathbf{G}'_7$  except that the challenge ciphertexts  $e_{j,i}$ 's are generated using  $r'_{j,i}$ 's and  $R'_{j,i}$ 's, respectively, which are independent of  $r_{j,i}$ . By the lossiness of PKE and a simple hybrid argument, we have

$$\begin{aligned}
& |\Pr[(\text{pk}, r) \in \mathcal{S} : (\text{pk}, r) \leftarrow \text{EXT in } \mathbf{G}'_7] \\
&- \Pr[(\text{pk}, r) \in \mathcal{S} : (\text{pk}, r) \leftarrow \text{EXT in } \mathbf{G}''_7]| \leq NS \cdot \epsilon_{\text{PKE}}^{\text{lo}}.
\end{aligned}$$

In  $\mathbf{G}''_7$ ,  $r_{j,i}$ 's are independent of the view of  $\mathcal{A}$  (and thus independent of  $\text{EXT}$ ), so we have

$$\Pr[(\text{pk}, r) \in \mathcal{S} : (\text{pk}, r) \leftarrow \text{EXT in } \mathbf{G}''_7] \leq \frac{NS}{|\mathcal{M}'|}.$$

Therefore, we have

$$|\Pr[\mathbf{G}_6^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_7^{\mathcal{A}} \Rightarrow 1]| \leq 2(q_{\mathbf{G}} + q_{\mathbf{H}}) \sqrt{NS \cdot \epsilon_{\text{PKE}}^{\text{lo}} + NS/|\mathcal{M}'|},$$

as stated in Lemma 4.5.  $\square$

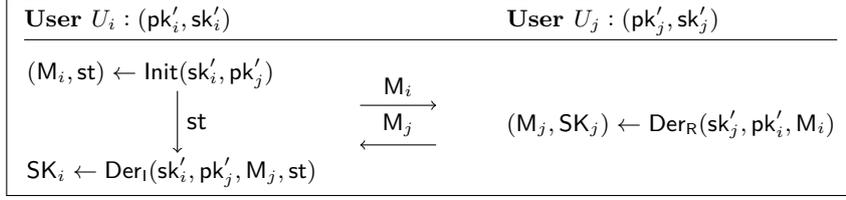


Figure 10: Illustration for a two-pass AKE protocol execution between user  $U_i$  and  $U_j$ .

## 5 Security Model for AKE

A two-message AKE protocol AKE consists of five algorithms  $\text{Setup}'$ ,  $\text{KG}'$ ,  $\text{Init}$ ,  $\text{Der}_R$ , and  $\text{Der}_I$ . The setup algorithm  $\text{Setup}'$ , on input security parameter  $1^\lambda$ , outputs global AKE system parameters  $\text{par}'$ . For sake of simplicity, we ignore the input  $\lambda$  and just write  $\text{par}' \leftarrow \text{Setup}'$ .  $\text{KG}'$  takes the system parameters  $\text{par}'$  as input and outputs a key pair  $(\text{pk}', \text{sk}')$ . A user in an AKE protocol runs  $\text{KG}'$  to generate a long-term key pair for itself.

Algorithms  $\text{Init}$ ,  $\text{Der}_R$ , and  $\text{Der}_I$  are used to establish AKE sessions between users. Let  $U_i$  and  $U_j$  be two users with long-term key pairs  $(\text{pk}'_i, \text{sk}'_i)$  and  $(\text{pk}'_j, \text{sk}'_j)$ , respectively. Figure 10 shows how  $U_i$ , (as initiator) shares an AKE session key with  $U_j$  (as responder). To initialize the session with  $U_j$ ,  $U_i$  runs the session initialization algorithm  $\text{Init}$ , which takes  $\text{sk}'_i, \text{pk}'_j$  as inputs and outputs a protocol message  $M_i$  and session state  $\text{st}$ , and then  $U_i$  sends  $M_i$  to  $U_j$  and keeps  $\text{st}$  locally. On receiving  $M_i$ ,  $U_j$  runs the responders derivation algorithm  $\text{Der}_R$ , which takes  $\text{sk}'_j, \text{pk}'_i$ , and the received message  $M_i$  as inputs, to generate a response  $M_j$  and a session key  $\text{SK}_j$ .  $U_j$  sends  $M_j$  to  $U_i$ . Finally, on receiving  $M_j$ ,  $U_i$  runs the initiators derivation algorithm  $\text{Der}_I$  which inputs  $\text{sk}'_i, \text{pk}'_j$ , the received message  $M_j$ , and the local session state  $\text{st}$  generated before, to generate a session key  $\text{SK}_i$ . In two-message AKE protocols, the responder does not need to save session state since it can compute the session key right after receiving the initiator's message.

**AKE SECURITY MODEL.** Following [20], we define a game-based AKE security model using pseudocode. This model is a weaker version of the weak-forward-secrecy model in [20] that it does not consider the state-reveal attack and considers only one TEST query. Our motivation of considering such a model is to focus on the standard security for AKE, such as security against key-compromise-impersonation (KCI) attacks and weak forward secrecy. With state reveals, our security loss has an additional linear factor on the number of sessions, but no square-root loss, which still improves the bound of Hövelmanns et al. [19]. We stress that even in this weaker model the analysis of KEM-based AKE in the QROM of Hövelmanns et al. still has a square-root-loss. For more details, please refer to Remark 6.3.

In this AKE model, we consider  $N$  users  $U_1, \dots, U_N$  in the security game  $\text{wFS-KCI}_{\text{AKE}, b}$  shown in Figure 11, where each user  $U_i (i \in [N])$  has a honestly generated long-term key pair  $(\text{pk}'_i, \text{sk}'_i)$ . Each user may have multiple sessions at the same time. In this model, each session between two users has a unique session identification number  $\text{sID}$ , and it also has the following variables defined relative to  $\text{sID}$ :

- $\text{Init}[\text{sID}] \in [N]$  denotes the initiator of the session.
- $\text{Resp}[\text{sID}] \in [N]$  denotes the responder of the session.
- $\text{Type}[\text{sID}] \in \{\text{"In"}, \text{"Re"}\}$  denotes the session is owned by the initiator or by the responder.
- $\text{Used}[\text{sID}]$  denotes whether the session  $\text{sID}$  was used if  $\text{Type}[\text{sID}] = \text{"In"}$ .
- $\text{I}[\text{sID}]$  denotes the messages that was computed by the initiator.
- $\text{R}[\text{sID}]$  denotes the messages that was computed by the responder.
- $\text{SK}[\text{sID}]$  denotes the session key of the session.

In the security games  $\text{wFS-KCI}_{\text{AKE}, b}$  of this model,  $\mathcal{A}$  is given access to oracles  $\text{SESSION}_I, \text{SESSION}_R$ , and  $\text{DER}_I$ . These oracles are used to capture the adversary's ability to control the channel, in the security game as described in Figure 11, adversary  $\mathcal{A}$  is given access to oracles  $\text{SESSION}_I, \text{SESSION}_R$ , and  $\text{DER}_I$ .  $\text{SESSION}_I$  creates a session owned by the initiator, and  $\text{SESSION}_R$  creates a session owned by the responder, which are different sessions. More precisely,  $\mathcal{A}$  queries  $\text{SESSION}_I(i, j)$  to activate a session between  $U_i$  (as initiator) with  $U_j$  (as responder) and gets  $(\text{sID}, M_i)$ , where  $\text{sID}$  is the session ID of this session and  $M_i$  is  $U_i$ 's initiator protocol message (generated from  $\text{Init}$ ). The query  $\text{SESSION}_R(i, j, M)$  captures the process that  $\mathcal{A}$  sends  $M$  to  $U_j$ , activates a session between  $U_i$  (as initiator) with  $U_j$  (as responder), and gets the session ID  $\text{sID}$  and the responded protocol message  $M_j$  of this session. To complete a session  $\text{sID}$  owned

<p><b>Game wFS-KCI<sub>AKE,b</sub></b></p> <p>01 <math>\text{cnt} := 0, \mathcal{L}_{\text{test}} := \emptyset</math></p> <p>02 <math>\text{par}' \leftarrow \text{Setup}'(1^\lambda)</math></p> <p>03 <b>for</b> <math>i \in [N] : (\text{pk}'_i, \text{sk}'_i) \leftarrow \text{KG}'(\text{par}')</math></p> <p>04 <math>O_1 := (\text{SESSION}_I, \text{DER}_I, \text{SESSION}_R)</math></p> <p>05 <math>O_2 := (\text{COR}, \text{REV}, \text{TEST})</math></p> <p>06 <math>b' \leftarrow \mathcal{A}^{O_1, O_2}(\text{par}, (\text{pk}'_i)_{i \in [N]})</math></p> <p>07 <b>if</b> <math>\text{Fresh}(\text{sID}^*) = 0 \vee \text{Valid}(\text{sID}^*) = 0</math>:</p> <p>08     <b>return</b> 0</p> <p>09 <b>return</b> <math>b'</math></p> <p><b>Oracle TEST</b>(sID)                     // Only one query</p> <p>10 <math>\text{sID}^* := \text{sID}</math></p> <p>11 <math>\text{SK}_0 := \text{SK}[\text{sID}], \text{SK}_1 \xleftarrow{\\$} \mathcal{SK}</math></p> <p>12 <b>return</b> <math>\text{SK}_b</math></p> <p><b>Oracle REV</b>(sID)</p> <p>13 <math>\text{revSK}[\text{sID}] := 1</math></p> <p>14 <b>return</b> <math>\text{SK}[\text{sID}]</math></p> <p><b>Oracle COR</b>(<math>i</math>)</p> <p>15 <math>\text{Cor}[i] := 1</math></p> <p>16 <b>return</b> <math>\text{sk}'_i</math></p>	<p><b>Oracle SESSION<sub>I</sub></b>((<math>i, j</math>) <math>\in [N]^2</math>)</p> <p>17 <math>\text{cnt} := \text{cnt} + 1, \text{sID} := \text{cnt}</math></p> <p>18 <math>(\text{Init}[\text{sID}], \text{Resp}[\text{sID}]) := (i, j)</math></p> <p>19 <math>\text{Type}[\text{sID}] := \text{"In"}</math></p> <p>20 <math>(M_i, \text{st}) := \text{Init}(\text{sk}_i, \text{pk}_j, \text{par}')</math></p> <p>21 <math>(\text{I}[\text{sID}], \text{St}[\text{sID}]) := (M_i, \text{st})</math></p> <p>22 <b>return</b> <math>(\text{sID}, M_i)</math></p> <p><b>Oracle DER<sub>I</sub></b>(sID, M)</p> <p>23 <b>if</b> <math>\text{Used}[\text{sID}] = 1 \vee \text{St}[\text{sID}] = \perp</math></p> <p>24     <math>\forall \text{SK}[\text{sID}] \neq \perp : \text{return } \perp</math></p> <p>25 <math>\text{Used}[\text{sID}] := 1, \text{st} := \text{St}[\text{sID}]</math></p> <p>26 <math>(i, j) := (\text{Init}[\text{sID}], \text{Resp}[\text{sID}])</math></p> <p>27 <math>\text{SK} := \text{Der}_I(\text{sk}'_i, \text{pk}'_j, M, \text{st})</math></p> <p>28 <math>(\text{R}[\text{sID}], \text{SK}[\text{sID}]) := (M, \text{SK})</math></p> <p>29 <b>return</b> 1</p> <p><b>Oracle SESSION<sub>R</sub></b>((<math>i, j</math>) <math>\in [N]^2, M</math>)</p> <p>30 <math>\text{cnt} := \text{cnt} + 1, \text{sID} := \text{cnt}</math></p> <p>31 <math>(\text{Init}[\text{sID}], \text{Resp}[\text{sID}]) := (i, j)</math></p> <p>32 <math>\text{Type}[\text{sID}] := \text{"Re"}</math></p> <p>33 <math>(M_j, \text{SK}) := \text{Der}_R(\text{sk}'_i, \text{pk}'_j, M)</math></p> <p>34 <math>(\text{I}[\text{sID}], \text{R}[\text{sID}]) := (M, M_j)</math></p> <p>35 <math>\text{SK}[\text{sID}] := \text{SK}</math></p> <p>36 <b>return</b> <math>(\text{sID}, M_j)</math></p>
--	--

Figure 11: The games wFS-KCI<sub>AKE,b</sub> ( $b \in \{0, 1\}$ ) for AKE = (Setup', KG', Init, Der<sub>R</sub>, Der<sub>I</sub>).

by an initiator,  $\mathcal{A}$  queries  $\text{DER}_I(\text{sID}, M)$ , which captures the process that  $\mathcal{A}$  sends the response  $M$  to the initiator of the session. Moreover,  $\text{DER}_I$  and  $\text{SESSION}_R$  may output  $\perp$  to indicate that the session does not generate a session key.

In wFS-KCI<sub>AKE,b</sub>,  $\mathcal{A}$  can forward messages between sessions honestly, or modify messages to launch some attacks (e.g., man-in-the-middle attack or replaying attack). For two sessions  $\text{sID}$  and  $\text{sID}'$ , We define two relationships:

**Definition 5.1** ((Partially) Matching Session). We say sessions  $\text{sID}$  and  $\text{sID}'$  match if the same users are involved (*i.e.*,  $(\text{Init}[\text{sID}], \text{Resp}[\text{sID}]) = (\text{Init}[\text{sID}'], \text{Resp}[\text{sID}'])$ ), the messages sent and received are the same ( $(\text{I}[\text{sID}], \text{R}[\text{sID}]) = (\text{I}[\text{sID}'], \text{R}[\text{sID}'])$ ), and they are of different types  $\text{Type}[\text{sID}] \neq \text{Type}[\text{sID}']$ .

We say  $\text{sID}$  is partially matching to  $\text{sID}'$  if  $\text{Type}[\text{sID}] = \text{"In"}, \text{Type}[\text{sID}'] = \text{"Re"}$ , and the initial messages are the same ( $\text{I}[\text{sID}] = \text{I}[\text{sID}']$ ).

Sessions  $\text{sID}$  and  $\text{sID}'$  match means that  $\mathcal{A}$  honestly delivers the protocol message between  $\text{sID}$  with  $\text{sID}'$  without any modification. If  $\text{Type}[\text{sID}] = \text{"In"}$ ,  $\text{sID}$  partially matches  $\text{sID}'$  means that  $\mathcal{A}$  honestly sends the protocol message generated from  $\text{sID}$  to  $\text{sID}'$ .  $\text{sID}$  does not have matching session means that  $\mathcal{A}$  may tamper with the session.

Furthermore,  $\mathcal{A}$  has access to oracles COR and REV to reveal secret information. By querying COR( $i$ ),  $\mathcal{A}$  can get the long-term secret key of user  $U_i$ , and by querying REV(sID),  $\mathcal{A}$  can obtain the session key of session  $\text{sID}$ . The following variables are used to keep track of which queries the adversary made and which secret information is revealed. If the variable equals 1, then it means the corresponding secret information is revealed.

- $\text{Cor}[i]$  denotes whether the long-term secret key of user  $U_i$  was revealed.
- $\text{revSK}[\text{sID}]$  denotes whether the session key of session  $\text{sID}$  was revealed.

Finally,  $\mathcal{A}$  is given access to oracle TEST which will return either the session key of the given session or a uniformly random key independent of the given session.  $\mathcal{A}$  can only issue once query to TEST. A

	$\text{Cor}[i^*]$	$\text{Cor}[j^*]$	$\text{Type}[sID^*]$	$ \mathfrak{M}(sID^*) $	$ \mathfrak{P}(sID^*) $
$\mathcal{A}$ gets (Initiator, Responder), attack type					
(1) (long-term, long-term), wFS	-	-	-	1	-
(2) (long-term, long-term), wFS	-	-	“Re”	0	1
(3) (long-term, -), KCI	-	<b>F</b>	“In”	0	0
(4) (-, long-term), KCI	<b>F</b>	-	“Re”	0	0

Table 1: List of valid attack types against two-message AKE protocols in the  $\text{wFS-KCI}_{\text{AKE},b}$  game (in Figure 11).  $(i^*, j^*) = (\text{Init}[sID^*], \text{Resp}[sID^*])$ . An attack is regarded as an AND conjunction of variables with specified values as shown in the each line, where **F** means “false”, “-” means that the variable can take arbitrary value (can be viewed as “true”), and “n/a” means that there is no (partially) matching session exists. The sets  $\mathfrak{M}(sID^*)$  and  $\mathfrak{P}(sID^*)$  are defined in Valid procedure in Figure 12. Here “0” and “1” are numbers instead of boolean values.

session  $sID$  that has been queried to TEST is called as a test session, and we denote the test session as  $sID^*$ .  $\mathcal{A}$  wins if it can distinguish the keys output by TEST are either the actual session keys of the test session or independent random keys, i.e., distinguish whether it plays  $\text{wFS-KCI}_{\text{AKE},0}$  or  $\text{wFS-KCI}_{\text{AKE},1}$ .

<b>Alg Fresh</b> ( $sID^*$ )	
01 $(i^*, j^*) := (\text{Init}[sID^*], \text{Resp}[sID^*])$	
02 $\mathfrak{M}(sID^*) := \{sID \mid (\text{Init}[sID], \text{Resp}[sID]) = (i^*, j^*)$ $\wedge (\text{I}[sID], \text{R}[sID]) = (\text{I}[sID^*], \text{R}[sID^*])$ $\wedge \text{Type}[sID] \neq \text{Type}[sID^*]\}$	// Matching session(s) of $sID^*$
03 <b>if</b> $\text{revSK}[sID^*] \vee (\exists sID \in \mathfrak{M}(sID^*) : \text{revSK}[sID] = 1)$ : <b>return</b> 0	// $\mathcal{A}$ trivially learned the test session's key
04 <b>if</b> $\exists sID \in \mathfrak{M}(sID^*)$ s.t. $sID \in \mathcal{L}_{\text{test}}$ : <b>return</b> 0	// $\mathcal{A}$ also tested a matching session
05 <b>return</b> 1	
<b>Alg Valid</b> ( $sID^*$ )	
06 $(i^*, j^*) := (\text{Init}[sID^*], \text{Resp}[sID^*])$	
07 $\mathfrak{M}(sID^*) := \{sID \mid (\text{Init}[sID], \text{Resp}[sID]) = (i^*, j^*)$ $\wedge (\text{I}[sID], \text{R}[sID]) = (\text{I}[sID^*], \text{R}[sID^*])$ $\wedge \text{Type}[sID] \neq \text{Type}[sID^*]\}$	// Matching session(s) of $sID^*$
08 $\mathfrak{P}(sID^*) := \{sID \mid \text{I}[sID] = \text{I}[sID^*]$ $\wedge \text{Type}[sID] \neq \text{Type}[sID^*]$ $\wedge \text{Type}[sID] = \text{“In”}\}$	// Partially matching session(s) of $sID^*$
09 <b>if</b> $ \mathfrak{P}(sID^*)  > 1 \vee  \mathfrak{M}(sID^*)  > 1$ : <b>return</b> 1	
10 <b>if</b> the attack type of $sID^* \in$ Table 1: <b>return</b> 1	
11 <b>else return</b> 0	

Figure 12: Algorithms to check the validity and freshness of the test session  $sID^*$ . Item 09 forces a secure AKE protocol not to have more than one (partially) matching sessions.

To avoid trivial attack, we define *freshness* and *validity*. In Figure 12, we define two processes to determine if a given session ID  $sID^*$  is fresh and valid. Roughly speaking,  $sID^*$  is fresh if its session key is not revealed and its matching sessions are not revealed and not been tested, and  $sID^*$  is valid if it is fresh and the attack  $\mathcal{A}$  performed on  $sID^*$  is defined in this AKE model. We capture all valid attacks in Table 1.

**Definition 5.2** (Freshness and Validity). Let  $sID^*$  be a session. (1)  $sID^*$  is fresh if  $\text{Fresh}(sID^*)$  outputs 1. (2)  $sID^*$  is valid if  $\text{Valid}(sID^*)$  outputs 1. Fresh and Valid are given in Figure 12.

In Table 1, all attacks are defined using the boolean variables that indicates which queries the adversary made. This table is obtained from considering all possible attacks and excluding all trivial or

redundant attacks and state attacks in the full attack table in [20]. Informally, the attacks defined in Table 1 capture weak forward secrecy (wFS), and key compromise impersonation (KCI). The man-in-the-middle attack and replaying attack are captured. Moreover, if  $\mathcal{A}$  is able to create more than one (partially) matching session to a test session, then it is considered to be insecure in this model.

In this model, we require that the test session is fresh and valid. The adversary wins if it distinguishes the session keys from uniformly random keys which it obtains through queries to the Test oracle.

**Definition 5.3** (Key Indistinguishability of AKE). Let  $\text{AKE} := (\text{Setup}', \text{KG}', \text{Init}, \text{Der}_R, \text{Der}_I)$  be an AKE protocol and consider the games  $\text{wFS-KCI}_{\text{AKE}, b}^A$  for  $b \in \{0, 1\}$  defined in Figure 11. We say AKE is wFS-KCI secure, if for all adversaries  $\mathcal{A}$ , the following advantage is negligible:

$$\text{Adv}_{\text{AKE}}^{\text{wFS-KCI}}(\mathcal{A}) := \left| \Pr \left[ \text{wFS-KCI}_{\text{AKE}, 0}^A(\lambda) \Rightarrow 1 \right] - \Pr \left[ \text{wFS-KCI}_{\text{AKE}, 1}^A(\lambda) \Rightarrow 1 \right] \right|$$

## 6 Session-tight AKE protocol

Let  $\text{KEM}_1$  and  $\text{KEM}_2$  be two KEM schemes with KEM key spaces  $\mathcal{K}_1$  and  $\mathcal{K}_2$ , respectively. We construct our two-message AKE protocol  $\text{AKE} = (\text{Setup}', \text{KG}', \text{Init}, \text{Der}_R, \text{Der}_I)$  as shown in Figure 13, where  $\mathcal{SK}$  is the session key space of AKE and  $\text{H}: \{0, 1\}^* \rightarrow \mathcal{SK}$  is a hash function which is used to derive the session key.

<p><b>Alg Setup'(<math>\lambda</math>)</b></p> <p>01 <math>\text{par} \leftarrow \text{Setup}_1(\lambda)</math></p> <p>02 <math>\tilde{\text{par}} \leftarrow \text{Setup}_2(\lambda)</math></p> <p>03 <b>return</b> <math>\text{par}' := (\text{par}, \tilde{\text{par}})</math></p> <p><b>Alg KG'(<math>\text{par}, \tilde{\text{par}}</math>)</b></p> <p>04 <math>(\text{pk}, \text{sk}) \leftarrow \text{KGen}_1(\text{par})</math></p> <p>05 <b>return</b> <math>(\text{pk}, \text{sk})</math></p> <p><b>Alg Der<sub>R</sub>(<math>\text{sk}_j, \text{pk}_i, (\tilde{\text{pk}}, \text{ct}_j)</math>)</b></p> <p>06 <math>K_j := \text{Decaps}_1(\text{sk}_j, \text{ct}_j)</math></p> <p>07 <math>(\tilde{\text{ct}}, \tilde{K}) \leftarrow \text{Encaps}_2(\text{pk})</math></p> <p>08 <math>(\text{ct}_i, K_i) \leftarrow \text{Encaps}_1(\text{pk}_i)</math></p> <p>09 <math>\text{ctxt} := (\text{pk}_i, \text{pk}_j, \tilde{\text{pk}}, \text{ct}_i, \text{ct}_j, \tilde{\text{ct}})</math></p> <p>10 <math>\text{SK} := \text{H}(\text{ctxt}, K_i, K_j, \tilde{K})</math></p> <p>11 <b>return</b> <math>((\tilde{\text{ct}}, \text{ct}_i), \text{SK})</math></p>	<p><b>Alg Init(<math>\text{sk}_i, \text{pk}_j, (\text{par}, \tilde{\text{par}})</math>)</b></p> <p>12 <math>(\tilde{\text{pk}}, \tilde{\text{sk}}) \leftarrow \text{KGen}_2(\tilde{\text{par}})</math></p> <p>13 <math>(\text{ct}_j, K_j) \leftarrow \text{Encaps}_1(\text{pk}_j)</math></p> <p>14 <math>\text{st} := (\tilde{\text{pk}}, \tilde{\text{sk}}, \text{ct}_j, K_j)</math></p> <p>15 <b>return</b> <math>((\tilde{\text{pk}}, \text{ct}_j), \text{st})</math></p> <p><b>Alg Der<sub>I</sub>(<math>\text{sk}_i, \text{pk}_j, (\tilde{\text{ct}}, \text{ct}_i), \text{st}</math>)</b></p> <p>16 <b>let</b> <math>(\tilde{\text{pk}}, \tilde{\text{sk}}, \text{ct}_j, K_j) := \text{st}</math></p> <p>17 <math>\tilde{K} := \text{Decaps}_2(\tilde{\text{sk}}, \tilde{\text{ct}})</math></p> <p>18 <math>K_i := \text{Decaps}_1(\text{sk}_i, \text{ct}_i)</math></p> <p>19 <math>\text{ctxt} := (\text{pk}_i, \text{pk}_j, \tilde{\text{pk}}, \text{ct}_i, \text{ct}_j, \tilde{\text{ct}})</math></p> <p>20 <math>\text{SK} := \text{H}(\text{ctxt}, K_i, K_j, \tilde{K})</math></p> <p>21 <b>return</b> SK</p>
---	---

Figure 13: Our AKE protocol AKE which is based on KEM schemes  $\text{KEM}_1 = (\text{Setup}_1, \text{KGen}_1, \text{Encaps}_1, \text{Decaps}_1)$  and  $\text{KEM}_2 = (\text{Setup}_2, \text{KGen}_2, \text{Encaps}_2, \text{Decaps}_2)$ .

**Theorem 6.1** Let  $N$  be the number of users and  $S$  be the number of total sessions in game wFS-KCI. If  $\text{KEM}_1$  is a MC-IND-CCA secure KEM and  $\text{KEM}_2$  is a MUC-IND-CCA secure KEM and  $\text{H}$  is modeled as a quantum random oracle, then for any quantum adversary  $\mathcal{A}$  against AKE, there exists quantum adversaries  $\mathcal{B}_1$  and  $\mathcal{B}_2$  such that the running time of  $\mathcal{B}_1$  and  $\mathcal{B}_2$  about that of  $\mathcal{A}$  and

$$\begin{aligned} \text{Adv}_{\text{AKE}}^{\text{wFS-KCI}}(\mathcal{A}) \leq & N\eta_{\text{KEM}_1} + \frac{4q_{\text{H}}\sqrt{S}N}{\sqrt{|\mathcal{K}_1|}} + \frac{4q_{\text{H}}\sqrt{S}}{\sqrt{|\mathcal{K}_2|}} \\ & + 4 \cdot \text{Adv}_{\text{KEM}_2}^{\text{MUC-IND-CCA}}(\mathcal{B}_2) + 4N \cdot \text{Adv}_{\text{KEM}_1}^{\text{MC-IND-CCA}}(\mathcal{B}_1), \end{aligned}$$

where  $q_{\text{H}}$  is the number of queries to  $\text{H}$  and  $\eta_{\text{KEM}_1}$  is the public key collision probability of  $\text{KEM}_1$ .

Combining the results from this section with the results from Sections 3.2, 3.3, 4.1 and 4.2, we obtain the following corollary

**Corollary 6.2** *There is an AKE scheme AKE, such that for any quantum adversary  $\mathcal{A}$  against AKE, there is an algorithm  $\mathcal{B}$  such that the running time of  $\mathcal{B}$  is about that of  $\mathcal{A}$  and*

$$\begin{aligned} \text{Adv}_{\text{AKE}}^{\text{wFS-KCI}}(\mathcal{A}) &\leq 16k \cdot \text{Adv}^{\text{LWE}_{t,m,q,D_{z,s''}}}(\mathcal{B}) \\ &\quad + 16N\ell \cdot \text{Adv}^{\text{LWE}_{k',m,q,D_{z,s}}}(\mathcal{B}) + \text{negl}(\lambda), \end{aligned}$$

where  $k = \Theta(\lambda)$ ,  $k' = \Theta(\lambda)$ ,  $\ell = \Theta(\lambda)$ ,  $t = \Theta(\lambda)$ ,  $m = o(\lambda^2)$  and  $s, s'' >$  denote appropriate parameters and  $\text{negl}(\lambda)$  denotes a negligible statistical term.

*Remark 6.3 (Session State Reveal).* Our AKE model does not allow an adversary to reveal session states as in [19]. Considering SessionStateReveal, our security bound is no longer session-tight, since we cannot simulate the session states in a session-tight manner, given only MC-CCA or MUC-CCA security. In order to embed challenges, the security reduction has to guess which session will be tested by adversaries in advance. Hence, the bound will be

$$\varepsilon_{\text{AKE}} \leq \Theta(NS) \cdot \Theta(\lambda) \cdot \varepsilon_{\text{LWE}},$$

which does not contain square-root loss. It still improves the bound of Hövelmanns et al. [19] which has square-root loss on  $\varepsilon_{\text{LWE}}$  (cf. Equation (1)).

*Proof of Theorem 6.1.* First, we assume that all users in the AKE game have different key pairs and all the messages output by the oracles are different. This will add  $\eta_{\text{KEM}_1}$  to the final bound. Since  $\text{KEM}_1$  and  $\text{KEM}_2$  are multi-challenge IND-CCA and multi-user-challenge IND-CCA secure, respectively, the probability that two different executions of  $\text{Encaps}_1$ ,  $\text{KGen}_2$ , or  $\text{Encaps}_2$  have the same output is negligible (and such probability is already considered in their multi-user-challenge or multi-challenge definitions). So, assuming different executions of  $\text{SESSION}_I$  and  $\text{SESSION}_R$  will output different protocol messages will not influence our final bound. Moreover, by this assumption, it is impossible for a session to have more than one matching or partially matching session.

To bound  $\text{Adv}_{\text{AKE}}^{\text{wFS-KCI}}(\mathcal{A})$ , we split up the event that the adversary wins into four cases. Let  $\mathbf{G}_{x,b}$  be a game that is the same as  $\text{wFS-KCI}_{\text{AKE},b}^A(\lambda)$  except that the test session  $\text{sID}^*$  is of type  $(x)$  (for  $x \in \{1, 2, 3, 4\}$ , (see Table 1). That is,

$$\Pr[\mathbf{G}_{x,b} \Rightarrow 1] = \Pr[\text{wFS-KCI}_{\text{AKE},b}^A(\lambda) \Rightarrow 1 \wedge \text{sID}^* \text{ is of type } (x)],$$

and thus we have

$$\begin{aligned} \text{Adv}_{\text{AKE}}^{\text{wFS-KCI}}(\mathcal{A}) &= \left| \Pr[\mathbf{G}_{x,0} \Rightarrow 1] - \Pr[\mathbf{G}_{x,1} \Rightarrow 1] \right| \\ &\leq \sum_{x=1}^4 |\Pr[\mathbf{G}_{x,0} \Rightarrow 1] - \Pr[\mathbf{G}_{x,1} \Rightarrow 1]|. \end{aligned}$$

Now, we can construct a security reduction according to the type of  $\text{sID}^*$ . Lemmata 6.4 and 6.5 bound  $|\Pr[\mathbf{G}_{x,0} \Rightarrow 1] - \Pr[\mathbf{G}_{x,1} \Rightarrow 1]|$  for  $x \in \{1, 2, 3, 4\}$ . These lemmas will be proved later.

**Lemma 6.4** *With notations and assumptions in the proof of Theorem 6.1, there exists an adversary  $\mathcal{B}_2$  such that its running time is about if  $\mathcal{A}$  and*

$$\begin{aligned} |\Pr[\mathbf{G}_{1,0}^A \Rightarrow 1] - \Pr[\mathbf{G}_{1,1}^A \Rightarrow 1]| &\leq 2\text{Adv}_{\text{KEM}_2}^{\text{MUC-IND-CCA}}(\mathcal{B}_2) + \frac{2q_H\sqrt{S}}{\sqrt{|\mathcal{K}_2|}}, \\ |\Pr[\mathbf{G}_{2,0}^A \Rightarrow 1] - \Pr[\mathbf{G}_{2,1}^A \Rightarrow 1]| &\leq 2\text{Adv}_{\text{KEM}_2}^{\text{MUC-IND-CCA}}(\mathcal{B}_2) + \frac{2q_H\sqrt{S}}{\sqrt{|\mathcal{K}_2|}}. \end{aligned}$$

**Lemma 6.5** *With notations and assumptions in the proof of Theorem 6.1, there exists an adversary  $\mathcal{B}_1$  such that its running time is about if  $\mathcal{A}$  and*

$$\begin{aligned} |\Pr[\mathbf{G}_{3,0}^A \Rightarrow 1] - \Pr[\mathbf{G}_{3,1}^A \Rightarrow 1]| &\leq 2N\text{Adv}_{\text{KEM}_1}^{\text{MC-IND-CCA}}(\mathcal{B}_1) + \frac{2q_H N\sqrt{S}}{\sqrt{|\mathcal{K}_1|}}, \\ |\Pr[\mathbf{G}_{4,0}^A \Rightarrow 1] - \Pr[\mathbf{G}_{4,1}^A \Rightarrow 1]| &\leq 2N\text{Adv}_{\text{KEM}_1}^{\text{MC-IND-CCA}}(\mathcal{B}_1) + \frac{2q_H N\sqrt{S}}{\sqrt{|\mathcal{K}_1|}}. \end{aligned}$$

Combining these lemmas, we have

$$\begin{aligned} \text{Adv}_{\text{AKE}}^{\text{wFS-KCI}}(\mathcal{A}) &= \left| \Pr \left[ \text{wFS-KCI}_{\text{AKE},0}^{\mathcal{A}} \Rightarrow 1 \right] - \Pr \left[ \text{wFS-KCI}_{\text{AKE},1}^{\mathcal{A}} \Rightarrow 1 \right] \right| \\ &\leq N\eta_{\text{KEM}_1} + \frac{4q_{\text{H}}N\sqrt{S}}{\sqrt{|\mathcal{K}_1|}} + \frac{4q_{\text{H}}\sqrt{S}}{\sqrt{|\mathcal{K}_2|}} \\ &\quad + 4\text{Adv}_{\text{KEM}_2}^{\text{MUC-IND-CCA}}(\mathcal{B}_2) + 4N \cdot \text{Adv}_{\text{KEM}_1}^{\text{MC-IND-CCA}}(\mathcal{B}_1) \end{aligned}$$

as stated in Theorem 6.1.  $\square$

*Proof of Lemma 6.4.* We bound type (1), i.e.  $|\Pr[\mathbf{G}_{1,0}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_{1,1}^{\mathcal{A}} \Rightarrow 1]|$ . The proof for type (2), i.e.  $|\Pr[\mathbf{G}_{2,0}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_{2,1}^{\mathcal{A}} \Rightarrow 1]|$ , is identical as the one of type (1). To prove the bound, we give a game sequence  $\mathbf{G}_{1-0,b}$ ,  $\mathbf{G}_{1-1,b}$ , and  $\mathbf{G}_{1-2,b}$  in Figure 14. Game  $\mathbf{G}_{1-0,b}$  is the same as  $\mathbf{G}_{1,b}$ , and we have

$$\Pr[\mathbf{G}_{1,b}^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathbf{G}_{1-0,b}^{\mathcal{A}} \Rightarrow 1] \text{ for both } b \in \{0,1\}.$$

<b>Game <math>\mathbf{G}_{1-0,b}</math>-<math>\mathbf{G}_{1-2,b}</math> (<math>b \in \{0,1\}</math>)</b>	<b>Oracle <math>\text{SESSION}_I((i,j) \in [N]^2)</math></b>
01 $\mathcal{L}_2 := \emptyset$	28 $\text{cnt} := \text{cnt} + 1, \text{sID} := \text{cnt}$
02 $\text{cnt} := 0, \text{sID}^* := \emptyset$	29 $(\text{Init}[\text{sID}], \text{Resp}[\text{sID}]) := (i, j)$
03 $\text{par} \leftarrow \text{Setup}_1(\lambda)$	30 $\text{Type}[\text{sID}] := \text{"In"}$
04 $\text{par} \leftarrow \text{Setup}_2(\lambda)$	31 $(\text{pk}, \text{sk}) \leftarrow \text{KGen}_2(\text{par})$
05 $\text{par}' := (\text{par}, \text{par})$	32 $\mathcal{L}_2 := \mathcal{L}_2 \cup \{(\text{pk}, \perp, \perp)\}$ <span style="float: right;"><math>\// \mathbf{G}_{1-1,b}</math>-<math>\mathbf{G}_{1-2,b}</math></span>
06 <b>for</b> $t \in [N]$ :	33 $(\text{ct}_j, K_j) \leftarrow \text{Encaps}_1(\text{pk}_j)$
07 $(\text{pk}_t, \text{sk}_t) \leftarrow \text{KGen}_1(\text{par})$	34 $(\text{pk}, \tilde{\text{sk}}) \leftarrow \text{KGen}_2(\text{par})$
08 $O_1 := (\text{SESSION}_I, \text{DER}_I, \text{SESSION}_R)$	35 $\text{st} := (\text{pk}, \tilde{\text{sk}}, \text{ct}_j, K_j), M_i := (\text{pk}, \text{ct}_j)$
09 $O_2 := (\text{COR}, \text{REV}, \text{TEST})$	36 $(\text{I}[\text{sID}], \text{St}[\text{sID}]) := (M_i, \text{st})$
10 $b' \leftarrow \mathcal{A}^{O_1, O_2, \text{IH}}(\text{par}', (\text{pk}_t)_{t \in [N]})$	37 <b>return</b> $(\text{sID}, M_i)$
11 <b>if</b> $\text{Fresh}(\text{sID}^*) = 0 \vee \text{Valid}(\text{sID}^*) = 0$	<b>Oracle <math>\text{SESSION}_R((i,j) \in [N]^2, M)</math></b>
12 $\wedge$ $\text{sID}^*$ is not type (1).	38 $\text{cnt} := \text{cnt} + 1, \text{sID} := \text{cnt}$
13 <b>return</b> 0	39 $(\text{Init}[\text{sID}], \text{Resp}[\text{sID}]) := (i, j)$
14 <b>return</b> $b'$	40 $\text{Type}[\text{sID}] := \text{"Re"}$
<b>Oracle <math>\text{DER}_I(\text{sID}, M)</math></b>	41 <b>let</b> $(\text{pk}, \text{ct}_j) := M$
15 <b>if</b> $\text{Used}[\text{sID}] = 1 \vee \text{St}[\text{sID}] = \perp$	42 $(\tilde{\text{ct}}, \tilde{K}) \leftarrow \text{Encaps}_2(\text{pk})$
16 $\vee \text{SK}[\text{sID}] \neq \perp$ : <b>return</b> $\perp$	43 <b>if</b> $(\text{pk}, \perp, \perp) \in \mathcal{L}_2$ <span style="float: right;"><math>\// \mathbf{G}_{1-1,b}</math>-<math>\mathbf{G}_{1-2,b}</math></span>
17 $\text{Used}[\text{sID}] := 1, \text{st} := \text{St}[\text{sID}]$	44 $\tilde{K}_j \leftarrow \mathcal{K}_2$ <span style="float: right;"><math>\// \mathbf{G}_{1-2,b}</math></span>
18 $(i, j) := (\text{Init}[\text{sID}], \text{Resp}[\text{sID}])$	45 $\mathcal{L}_2 := \mathcal{L}_2 \cup \{(\text{pk}, \tilde{\text{ct}}, \tilde{K})\}$ <span style="float: right;"><math>\// \mathbf{G}_{1-1,b}</math>-<math>\mathbf{G}_{1-2,b}</math></span>
19 <b>let</b> $(\tilde{\text{ct}}, \text{ct}_i) := M$	46 $K_j := \text{Decaps}_1(\text{sk}_j, \text{ct}_j)$
20 <b>let</b> $(\text{pk}, \text{sk}, \text{ct}_j, K_j) := \text{st}$	47 $\text{ctxt} := (\text{pk}_i, \text{pk}_j, \text{pk}, \text{ct}_i, \text{ct}_j, \tilde{\text{ct}})$
21 $\tilde{K} := \text{Decaps}_2(\tilde{\text{sk}}, \tilde{\text{ct}})$	48 $\text{SK} := \text{H}(\text{ctxt}, K_i, K_j, \tilde{K})$
22 <b>if</b> $\exists K$ s.t. $(\text{pk}, \tilde{\text{ct}}, K) \in \mathcal{L}_2$ <span style="float: right;"><math>\// \mathbf{G}_{1-1,b}</math>-<math>\mathbf{G}_{1-2,b}</math></span>	49 $\text{SK}[\text{sID}] := \text{SK}, M_j := (\tilde{\text{ct}}, \text{ct}_i)$
23 $\tilde{K} := K$ <span style="float: right;"><math>\// \mathbf{G}_{1-1,b}</math>-<math>\mathbf{G}_{1-2,b}</math></span>	50 $(\text{I}[\text{sID}], \text{R}[\text{sID}]) := (M, M_j)$
24 $\text{ctxt} := (\text{pk}_i, \text{pk}_j, \tilde{\text{pk}}, \text{ct}_i, \text{ct}_j, \tilde{\text{ct}})$	51 <b>return</b> $(\text{sID}, M_j)$
25 $\text{SK} := \text{H}(\text{ctxt}, K_i, K_j, \tilde{K})$	
26 $(\text{R}[\text{sID}], \text{SK}[\text{sID}]) := (M, \text{SK})$	
27 <b>return</b> 1	

Figure 14: Games in proving Lemma 6.4. Oracles in  $O_2$  are the same as in  $\text{wFS-KCI}_{\text{AKE},b}$ . The QRO H is simulated in the same way as in the proof of Theorem 4.4.

$\mathbf{G}_{1-1,b}$ : The game first initializes a list  $\mathcal{L}_2$  that will be used to store triples  $(\text{pk}, \tilde{\text{ct}}, \tilde{K})$  of  $\text{KEM}_2$  generated in  $\text{SESSION}_I$  and  $\text{SESSION}_R$ . Specifically, it maintains  $\mathcal{L}_2$  as follows:

- In  $\text{SESSION}_I(i, j)$ , the game simulator records  $(\text{pk}, \perp, \perp)$  in  $\mathcal{L}_2$ .
- In  $\text{SESSION}_R(i, j, (\text{pk}, \text{ct}_j))$ , the game simulator records the tuple  $(\text{pk}, \tilde{\text{ct}}, \tilde{K})$  in  $\mathcal{L}_2$  if  $\text{pk}$  is generated from  $\text{SESSION}_I$  (i.e., generated by the game simulator).

- In  $\text{DER}_1(\text{sID}, (\tilde{\text{ct}}, \text{ct}_i))$ , the game simulator gets the decryption of  $\tilde{\text{ct}}$  from  $\mathcal{L}_2$  (without decrypting) if its corresponding KEM key is recorded in the list.

This modification does not change  $\mathcal{A}$ 's view. If  $\tilde{\text{pk}}$  is generated from  $\text{SESSION}_I$  and  $\tilde{\text{ct}}$  is generated from  $\text{SESSION}_R$ , then the game simulator knows the corresponding KEM key of  $\tilde{\text{ct}}$ .  $\mathcal{L}_2$  is used to record such KEM keys. Therefore, these modifications are conceptual, we have

$$\Pr [\mathbf{G}_{1-0,b}^A \Rightarrow 1] = \Pr [\mathbf{G}_{1-1,b}^A \Rightarrow 1] \text{ for both } b \in \{0, 1\}.$$

$\mathbf{G}_{1-2,b}$ : We switch the KEM keys generated by  $\text{KEM}_2$  to be independently and uniformly random. Namely, in  $\text{SESSION}_R(i, j, (\tilde{\text{pk}}, \text{ct}_i))$ , if  $\tilde{\text{pk}}$  is generated from  $\text{SESSION}_I$  (i.e.,  $(\tilde{\text{pk}}, \perp, \perp) \in \mathcal{L}_2$ ), we sample  $\tilde{K}$  uniformly at random. Intuitively, this change will not influence the consistency of the computation of session keys, since in  $\mathbf{G}_{1,b}$ , all KEM keys of  $\text{KEM}_2$  generated by the game can be found in the  $\mathcal{L}_2$ , and the adversary cannot get the corresponding  $\tilde{\text{sk}}$ .

<b>Reduction</b> $\mathcal{B}_2^{\text{DEC}_{\text{mu}}}(\tilde{\text{par}}, \text{pk}, \text{c}, \mathbf{K})$	<b>Oracle</b> $\text{SESSION}_I((i, j) \in [N]^2)$
01 $\mathcal{L}_1 := \emptyset$	26 $\text{cnt} := \text{cnt} + 1, \text{sID} := \text{cnt}$
02 $\text{cnt} := 0, \text{sID}^* := \perp$	27 $(\text{Init}[\text{sID}], \text{Resp}[\text{sID}]) := (i, j)$
03 $\text{par} \leftarrow \text{Setup}_1(\lambda), \text{par}' := (\text{par}, \tilde{\text{par}})$	28 $\text{Type}[\text{sID}] := \text{"In"}$
04 <b>for</b> $t \in [N]$ :	29 $\tilde{\text{pk}} := \text{pk}[\text{sID}]$
05 $(\text{pk}_t, \text{sk}_t) \leftarrow \text{KGen}_1(\text{par})$	30 $\mathcal{L}_2 := \mathcal{L}_2 \cup \{(\tilde{\text{pk}}, \perp, \perp)\}$
06 $O_1 := (\text{SESSION}_I, \text{DER}_1, \text{SESSION}_R)$	31 $(\text{ct}_j, K_j) \leftarrow \text{Encaps}_1(\text{pk}_j)$
07 $O_2 := (\text{COR}, \text{REV}, \text{TEST})$	32 $\text{st} := (\tilde{\text{pk}}, \tilde{\text{sk}}, \text{ct}_j, K_j), M_i := (\tilde{\text{pk}}, \text{ct}_j)$
08 $b' \leftarrow \mathcal{A}^{O_1, O_2, \text{H}}(\text{par}', (\text{pk}_t)_{t \in [N]})$	33 $(\text{I}[\text{sID}], \text{St}[\text{sID}]) := (M_i, \text{st})$
09 <b>if</b> $\text{Fresh}(\text{sID}^*) = 0 \vee \text{Valid}(\text{sID}^*) = 0$	34 <b>return</b> $(\text{sID}, M_i)$
10 <b>return</b> 0	<b>Oracle</b> $\text{SESSION}_R((i, j) \in [N]^2, M)$
11 <b>return</b> $b'$	35 $\text{cnt} := \text{cnt} + 1, \text{sID} := \text{cnt}$
<b>Oracle</b> $\text{DER}_1(\text{sID}, M)$	36 $(\text{Init}[\text{sID}], \text{Resp}[\text{sID}]) := (i, j)$
12 <b>if</b> $\text{Used}[\text{sID}] = 1 \vee \text{St}[\text{sID}] = \perp$	37 $\text{Type}[\text{sID}] := \text{"Re"}$
13 $\vee \text{SK}[\text{sID}] \neq \perp$ : <b>return</b> $\perp$	38 <b>let</b> $(\text{pk}, \text{ct}_j) := M$
14 $\text{Used}[\text{sID}] := 1, \text{st} := \text{St}[\text{sID}]$	39 $(\tilde{\text{ct}}, \tilde{K}) \leftarrow \text{Encaps}_2(\tilde{\text{pk}})$
15 $(i, j) := (\text{Init}[\text{sID}], \text{Resp}[\text{sID}])$	40 <b>if</b> $(\tilde{\text{pk}}, \perp, \perp) \in \mathcal{L}_2$
16 <b>let</b> $(\tilde{\text{ct}}, \text{ct}_i) := M$	41 <b>Let</b> $t \in [S]$ s.t. $\tilde{\text{pk}} = \text{pk}[t]$
17 <b>let</b> $(\tilde{\text{pk}}, \perp, \text{ct}_j, K_j) := \text{st}$	42 $(\tilde{\text{ct}}, \tilde{K}) := (\text{c}[t, \text{sID}], \mathbf{K}[t, \text{sID}])$
18 <b>if</b> $\exists K$ s.t. $(\text{pk}, \tilde{\text{ct}}, K) \in \mathcal{L}_2$	43 $\mathcal{L}_2 := \mathcal{L}_2 \cup \{(\tilde{\text{pk}}, \tilde{\text{ct}}, \tilde{K})\}$
19 $\tilde{K} := K$	44 $K_j := \text{Decaps}_1(\text{sk}_j, \text{ct}_j)$
20 <b>else</b> $\tilde{K} := \text{DEC}_{\text{mu}}(\text{sID}, \tilde{\text{ct}})$	45 $(\text{ct}_i, K_i) \leftarrow \text{Encaps}_1(\text{pk}_i)$
21 $K_i := \text{Decaps}_1(\text{sk}_i, \text{ct}_i)$	46 $\text{ctxt} := (\text{pk}_i, \text{pk}_j, \tilde{\text{pk}}, \text{ct}_i, \text{ct}_j, \tilde{\text{ct}})$
22 $\text{ctxt} := (\text{pk}_i, \text{pk}_j, \tilde{\text{pk}}, \text{ct}_i, \text{ct}_j, \tilde{\text{ct}})$	47 $\text{SK} := \text{H}(\text{ctxt}, K_i, K_j, \tilde{K})$
23 $\text{SK} := \text{H}(\text{ctxt}, K_i, K_j, \tilde{K})$	48 $\text{SK}[\text{sID}] := \text{SK}, M_j := (\tilde{\text{ct}}, \text{ct}_i)$
24 $(\text{R}[\text{sID}], \text{SK}[\text{sID}]) := (M, \text{SK})$	49 $(\text{I}[\text{sID}], \text{R}[\text{sID}]) := (M, M_j)$
25 <b>return</b> 1	50 <b>return</b> $(\text{sID}, M_j)$

Figure 15: The reduction in the proof of Lemma 6.4. The highlighted codes show how the reduction embeds the challenges into the AKE sessions. Oracles  $O_2$  and  $\text{H}$  are simulated in the same way with that in Figure 14.

More formally, we use MUC-IND-CCA security of  $\text{KEM}_2$  to argue that  $\mathcal{A}$  cannot detect this change. To this end, we construct a reduction (against  $\text{KEM}_2$ ), which works as follows: It plays the MUC-IND-CCA game with  $S$  users and  $S$  challenge ciphertexts per users ( $S$  is the number of session in the AKE game). It embeds the challenge public keys in  $\text{SESSION}_I$  and embed the challenge ciphertexts in  $\text{SESSION}_R$ . This reduction  $\mathcal{B}_2$  is formally given Figure 15. The triple recorded in  $\mathcal{L}_2$  are all from the inputs of  $\mathcal{B}_2$ . When simulating  $\text{DER}_1$ , if  $(\tilde{\text{pk}}, \tilde{\text{ct}}, \tilde{K}) \notin \mathcal{L}_2$ , then  $\tilde{\text{ct}}$  is not a challenge ciphertext respect to  $\tilde{\text{pk}}$ , and  $\mathcal{B}_2$  can query  $\text{DEC}_{\text{mu}}$  to decrypt  $\tilde{\text{ct}}$ . If  $\mathcal{B}_2$  plays  $\text{MUC-IND-CCA}_{\text{KEM}_2, 0}$ , then it perfectly simulates  $\mathbf{G}_{1-1,b}$ , and if it plays  $\text{MUC-IND-CCA}_{\text{KEM}_2, 1}$ , then it perfectly simulates  $\mathbf{G}_{1-2,b}$ . Therefore, we have

$$|\Pr [\mathbf{G}_{1-1,b}^A \Rightarrow 1] - \Pr [\mathbf{G}_{1-2,b}^A \Rightarrow 1]| \leq \text{Adv}_{\text{KEM}_2}^{\text{MUC-IND-CCA}}(\mathcal{B}_2).$$

We argue that  $\mathbf{G}_{1-2,0}$  is equivalent to  $\mathbf{G}_{1-2,1}$ , except with a negligible probability. Let  $(\mathbf{pk}_i, \mathbf{pk}_j, \widetilde{\mathbf{pk}}, \mathbf{ct}_i, \mathbf{ct}_j, \widetilde{\mathbf{ct}}, K_i, K_j, \widetilde{K})$  be the hash input of  $\mathbf{sID}^*$ . Since  $\mathbf{sID}^*$  is of type (1), then by definition,  $\mathbf{sID}^*$  has a unique matching session, which means that  $(\widetilde{\mathbf{pk}}, \widetilde{\mathbf{ct}}, \widetilde{K})$  is generated by the game and thus  $\widetilde{K}$  is independently and uniformly random. Then, by Corollary 2.2, if  $\mathcal{A}$  queries  $\mathbf{H}$  at most  $q_{\mathbf{H}}$  times, except with  $\frac{2q_{\mathbf{H}}\sqrt{S}}{\sqrt{|\mathcal{K}_2|}}$  (there are at most  $S$  session keys), the session key of  $\mathbf{sID}^*$  generated in  $\mathbf{G}_{1-2,0}$  is indistinguishable from the one in  $\mathbf{G}_{1-2,1}$ , i.e.

$$|\Pr[\mathbf{G}_{1-2,0}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_{1-2,1}^{\mathcal{A}} \Rightarrow 1]| \leq \frac{2q_{\mathbf{H}}\sqrt{S}}{\sqrt{|\mathcal{K}_2|}},$$

and in conclusion, we have

$$|\Pr[\mathbf{G}_{1,0}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_{1,1}^{\mathcal{A}} \Rightarrow 1]| \leq 2\text{Adv}_{\text{KEM}_2}^{\text{MUC-IND-CCA}}(\mathcal{B}_2) + \frac{2q_{\mathbf{H}}\sqrt{S}}{\sqrt{|\mathcal{K}_2|}}.$$

The same arguments can be used to bound  $|\Pr[\mathbf{G}_{2,0}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_{2,1}^{\mathcal{A}} \Rightarrow 1]|$ , and we have

$$|\Pr[\mathbf{G}_{2,0}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_{2,1}^{\mathcal{A}} \Rightarrow 1]| \leq 2\text{Adv}_{\text{KEM}_2}^{\text{MUC-IND-CCA}}(\mathcal{B}_2) + \frac{2q_{\mathbf{H}}\sqrt{S}}{\sqrt{|\mathcal{K}_2|}}.$$

□

*Proof of Lemma 6.5.* We firstly bound type (3), i.e.  $|\Pr[\mathbf{G}_{3,0}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_{3,1}^{\mathcal{A}} \Rightarrow 1]|$ . The proof for type (4), i.e. bounding  $|\Pr[\mathbf{G}_{4,0}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{G}_{4,1}^{\mathcal{A}} \Rightarrow 1]|$  is the same as the one of type (3). We provide a sequence of games  $\mathbf{G}_{3-0,b}$ ,  $\mathbf{G}_{3-1,b}$ , and  $\mathbf{G}_{3-2,b}$  in Figure 16. Game  $\mathbf{G}_{3-0,b}$  is the same as  $\mathbf{G}_{3,b}$ , and we have

$$\Pr[\mathbf{G}_{3-0,b}^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathbf{G}_{3,b}^{\mathcal{A}} \Rightarrow 1] \text{ for both } b \in \{0, 1\}.$$

$\mathbf{G}_{3-1,b}$ : In this game, we guess which user will be targeted by the type (3) attack by  $\mathcal{A}$ . Namely, we modify the game as follows:

1. When initializing the game environment, the game samples  $t^* \xleftarrow{\$} [N]$  and initializes a list  $\mathcal{L}_1$ , which will be used to store some ciphertext-key pairs of  $\text{KEM}_1$  generated in  $\text{SESSION}_I$  or  $\text{DER}_I$ .
2. Moreover, after  $\mathcal{A}$  outputs  $b'$ , if  $\text{Resp}[\mathbf{sID}^*] \neq t^*$ , then the game aborts. This abort condition means that the game expects that the responder of test session  $\mathbf{sID}^*$  is user  $t^*$ , and it also means that  $\mathcal{A}$  cannot corrupt user  $t^*$  since the game requires  $\mathbf{sID}^*$  to be valid (in type (3) attack,  $\text{Cor}[j^*]$  cannot be corrupted, where  $j^* = \text{Resp}[\mathbf{sID}^*]$ ).
3. In  $\text{SESSION}_I$ , if the  $\text{KEM}_1$  ciphertexts and keys are generated using  $\mathbf{pk}_{t^*}$ , the game records such ciphertext-key pairs in  $\mathcal{L}_1$ . Moreover, in  $\text{SESSION}_R$ , the game uses  $\mathcal{L}_1$  to recover the  $\text{KEM}$  keys of  $\mathbf{ct}$  if  $\mathbf{ct}$  is recorded in  $\mathcal{L}_1$ . This modification is similar to the one in  $\mathbf{G}_{1-1,b}$  (in Figure 14), except that here  $\mathcal{L}_1$  only stores the ciphertext-key pairs generated using  $\mathbf{pk}_{t^*}$  in  $\text{SESSION}_I$ .

$\mathcal{A}$  cannot detect these modifications unless it triggers the abort conditions described above. Since  $t^*$  is sampled uniformly at random and the view of  $\mathcal{A}$  is independent of  $t^*$  until a potential abort, we have

$$\Pr[\mathbf{G}_{3-0,b}^{\mathcal{A}} \Rightarrow 1] = N \cdot \Pr[\mathbf{G}_{3-1,b}^{\mathcal{A}} \Rightarrow 1] \text{ for both } b \in \{0, 1\}.$$

$\mathbf{G}_{3-2,b}$ : In  $\text{SESSION}_I$ , if the responder is user  $t^*$ , then we switch the  $\text{KEM}$  keys to be independently and uniformly random, instead of being generated using  $\text{Encaps}_1(\mathbf{pk}_{t^*})$ .

Formally, we use MC-IND-CCA security of  $\text{KEM}_1$  to argue that  $\mathcal{A}$  cannot detect this change. Namely, we construct a reduction  $\mathcal{B}_1$  in Figure 17. It embeds the challenge public key  $\mathbf{pk}^*$  into the user  $t^*$ 's public key. The ciphertext-key pairs recorded in  $\mathcal{L}_1$  are all from the inputs of  $\mathcal{B}_1$ . In the reduction, if  $\mathcal{A}$  corrupts user  $t^*$ , then the reduction aborts. This does not change the winning probability of  $\mathcal{A}$  since by the definitions of  $\mathbf{G}_{3-1}$  and  $\mathbf{G}_{3-2}$ , corrupting user  $t^*$  makes  $\mathbf{sID}^*$  become invalid. Further, reduction  $\mathcal{B}_1$  uses the decryption oracle provided by  $\text{KEM}_1$  to decrypt the ciphertext respect to  $\mathbf{pk}_{t^*}$ . If  $\mathcal{B}_1$  plays

<b>Game <math>\mathbf{G}_{3-0,b}-\mathbf{G}_{3-2,b}</math> (<math>b \in \{0,1\}</math>)</b>	<b>Oracle <math>\text{SESSION}_I((i,j) \in [N]^2)</math></b>
01 $t^* \xleftarrow{\$} [N], \mathcal{L}_1 := \emptyset$	27 $\text{cnt} := \text{cnt} + 1, \text{sID} := \text{cnt}$
02 $\text{cnt} := 0, \text{sID}^* := \emptyset$	28 $(\text{Init}[\text{sID}], \text{Resp}[\text{sID}]) := (i, j)$
03 $\text{par} \leftarrow \text{Setup}_1(\lambda), \widetilde{\text{par}} \leftarrow \text{Setup}_2(\lambda)$	29 $\text{Type}[\text{sID}] := \text{"In"}$
04 $\text{par}' := (\text{par}, \widetilde{\text{par}})$	30 $(\widetilde{\text{pk}}, \widetilde{\text{sk}}) \leftarrow \text{KGen}_2(\widetilde{\text{par}})$
05 <b>for</b> $t \in [N]$ :	31 $(\text{ct}_j, K_j) \leftarrow \text{Encaps}_1(\text{pk}_j)$
06 $(\text{pk}_t, \text{sk}_t) \leftarrow \text{KGen}_1(\text{par})$	32 <b>if</b> $j = t^*$
07 $O_1 := (\text{SESSION}_I, \text{DER}_I, \text{SESSION}_R)$	33 $K_j \leftarrow \mathcal{K}_1$
08 $O_2 := (\text{COR}, \text{REV}, \text{TEST})$	34 $\mathcal{L}_1 := \mathcal{L}_1 \cup \{(\text{ct}_j, K_j)\}$
09 $b' \leftarrow \mathcal{A}^{O_1, O_2,  H }(\text{par}', (\text{pk}_t)_{t \in [N]})$	35 $(\widetilde{\text{pk}}, \widetilde{\text{sk}}) \leftarrow \text{KGen}_2(\widetilde{\text{par}})$
10 <b>if</b> $\text{Fresh}(\text{sID}^*) = 0 \vee \text{Valid}(\text{sID}^*) = 0$ :	36 $\text{st} := (\widetilde{\text{pk}}, \widetilde{\text{sk}}, \text{ct}_j, K_j), M_i := (\widetilde{\text{pk}}, \text{ct}_j)$
11 <b>return</b> 0	37 $(\text{I}[\text{sID}], \text{St}[\text{sID}]) := (M_i, \text{st})$
12 <b>if</b> $\text{Resp}[\text{sID}^*] \neq t^*$	38 <b>return</b> $(\text{sID}, M_i)$
13 <b>abort</b>	<b>Oracle <math>\text{SESSION}_R((i,j) \in [N]^2, M)</math></b>
14 <b>return</b> $b'$	39 $\text{cnt} := \text{cnt} + 1, \text{sID} := \text{cnt}$
<b>Oracle <math>\text{DER}_I(\text{sID}, M)</math></b>	40 $(\text{Init}[\text{sID}], \text{Resp}[\text{sID}]) := (i, j)$
15 <b>if</b> $\text{Used}[\text{sID}] = 1 \vee \text{St}[\text{sID}] = \perp$	41 $\text{Type}[\text{sID}] := \text{"Re"}$
16 $\vee \text{SK}[\text{sID}] \neq \perp$ : <b>return</b> $\perp$	42 <b>let</b> $(\widetilde{\text{pk}}, \text{ct}_j) := M$
17 $\text{Used}[\text{sID}] := 1, \text{st} := \text{St}[\text{sID}]$	43 $(\widetilde{\text{ct}}, \widetilde{K}) \leftarrow \text{Encaps}_2(\widetilde{\text{pk}})$
18 $(i, j) := (\text{Init}[\text{sID}], \text{Resp}[\text{sID}])$	44 $K_j := \text{Decaps}_1(\text{sk}_j, \text{ct}_j)$
19 <b>let</b> $(\widetilde{\text{ct}}, \text{ct}_i) := M$	45 <b>if</b> $j = t^*$
20 <b>let</b> $(\widetilde{\text{pk}}, \widetilde{\text{sk}}, \text{ct}_j, K_j) := \text{st}$	46 $\wedge \exists K$ s.t. $(\text{ct}_j, K) \in \mathcal{L}_1$
21 $\widetilde{K} := \text{Decaps}_2(\widetilde{\text{sk}}, \widetilde{\text{ct}})$	47 $K_j := K$
22 $K_i := \text{Decaps}_1(\text{sk}_i, \text{ct}_i)$	48 $(\text{ct}_i, K_i) \leftarrow \text{Encaps}_1(\text{pk}_i)$
23 $\text{ctxt} := (\text{pk}_i, \text{pk}_j, \widetilde{\text{pk}}, \text{ct}_i, \text{ct}_j, \widetilde{\text{ct}})$	49 $\text{ctxt} := (\text{pk}_i, \text{pk}_j, \text{pk}, \text{ct}_i, \text{ct}_j, \widetilde{\text{ct}})$
24 $\text{SK} := \text{H}(\text{ctxt}, K_i, K_j, \widetilde{K})$	50 $\text{SK} := \text{H}(\text{ctxt}, K_i, K_j, \widetilde{K})$
25 $(\text{R}[\text{sID}], \text{SK}[\text{sID}]) := (M, \text{SK})$	51 $\text{SK}[\text{sID}] := \text{SK}, M_j := (\widetilde{\text{ct}}, \text{ct}_i)$
26 <b>return</b> 1	52 $(\text{I}[\text{sID}], \text{R}[\text{sID}]) := (M, M_j)$
	53 <b>return</b> $(\text{sID}, M_j)$

Figure 16: The games sequence in the proof of Lemma 6.5. Oracles in  $O_2$  are defined the same as in  $\text{wFS-KCl}_{\text{AKE},b}$ . The QRO  $H$  is simulated in the same way with that in the proof of Theorem 4.4.

$\text{MC-IND-CCA}_{\text{KEM}_1,0}$ , then it perfectly simulates  $\mathbf{G}_{3-1}$ , and if it plays  $\text{MC-IND-CCA}_{\text{KEM}_1,0}$ , then it perfectly simulates  $\mathbf{G}_{3-2}$ . Therefore, we have

$$\Pr[\mathbf{G}_{3-1,b}^A \Rightarrow 1] - \Pr[\mathbf{G}_{3-2,b}^A \Rightarrow 1] \leq \text{Adv}_{\text{KEM}_1}^{\text{MC-IND-CCA}}(\mathcal{B}_1).$$

Lastly, we argue that  $\mathbf{G}_{3-2,0}$  is equivalent to  $\mathbf{G}_{3-2,1}$  except with a negligible probability. Let  $(\text{pk}_i, \text{pk}_j, \widetilde{\text{pk}}, \text{ct}_i, \text{ct}_j, \widetilde{\text{ct}}, K_i, K_j, \widetilde{K})$  be the hash input of  $\text{sID}^*$ . Since  $\text{sID}^*$  is of type (3), then by definitions in Table 1 and in games  $\mathbf{G}_{3-2,b}$ , user  $t^*$  is the responder of  $\text{sID}^*$ , which means that  $j = t^*$  and  $(\text{ct}_j, K_j)$  is generated in  $\text{SESSION}_I(i, j)$ , and thus the key  $K_j$  is independently and uniformly random. By Corollary 2.2, if  $\mathcal{A}$  queries  $H$  at most  $q_H$  times, except with  $2q_H \sqrt{\frac{S}{|\mathcal{K}_1|}}$ , the session key of  $\text{sID}^*$  generated in  $\mathbf{G}_{3-2,0}$  is indistinguishable from the one in  $\mathbf{G}_{3-2,1}$ , i.e.,

$$|\Pr[\mathbf{G}_{3-2,0}^A \Rightarrow 1] - \Pr[\mathbf{G}_{3-2,1}^A \Rightarrow 1]| \leq 2q_H \sqrt{\frac{|S|}{|\mathcal{K}_1|}},$$

and in conclusion, we have

$$|\Pr[\mathbf{G}_{3,0}^A \Rightarrow 1] - \Pr[\mathbf{G}_{3,1}^A \Rightarrow 1]| \leq 2N \text{Adv}_{\text{KEM}_1}^{\text{MC-IND-CCA}}(\mathcal{B}_1) + \frac{2Nq_H \sqrt{S}}{\sqrt{|\mathcal{K}_1|}}.$$

Observe that the same arguments can be used to bound  $|\Pr[\mathbf{G}_{4,0}^A \Rightarrow 1] - \Pr[\mathbf{G}_{4,1}^A \Rightarrow 1]|$ , and we have

$$|\Pr[\mathbf{G}_{4,0}^A \Rightarrow 1] - \Pr[\mathbf{G}_{4,1}^A \Rightarrow 1]| \leq 2N \text{Adv}_{\text{KEM}_1}^{\text{MC-IND-CCA}}(\mathcal{B}_1) + \frac{2Nq_H \sqrt{S}}{\sqrt{|\mathcal{K}_1|}}.$$

<p><b>Reduction</b> <math>\mathcal{B}_1^{\text{DEC}}(\text{par}, \text{pk}^*, \text{c}, \mathbf{K})</math></p> <pre> 01 <math>t^* \xleftarrow{\\$} [N], \mathcal{L}_1 := \emptyset</math> 02 <math>\text{cnt} := 0, \text{sID}^* := \emptyset</math> 03 <math>\widetilde{\text{par}} \leftarrow \text{Setup}_2(\lambda)</math> 04 <math>\text{par}' := (\text{par}, \widetilde{\text{par}}), \text{pk}_{t^*} := \text{pk}^*</math> 05 <b>for</b> <math>t \in [N] \setminus \{t^*\}</math> : 06   <math>(\text{pk}_t, \text{sk}_t) \leftarrow \text{KGen}_1(\text{par})</math> 07 <math>O_1 := (\text{SESSION}_I, \text{DER}_I, \text{SESSION}_R)</math> 08 <math>O_2 := (\text{COR}, \text{REV}, \text{TEST})</math> 09 <math>b' \leftarrow \mathcal{A}^{O_1, O_2, \text{H}}(\text{par}', (\text{pk}_t)_{t \in [N]})</math> 10 <b>if</b> <math>\text{Fresh}(\text{sID}^*) = 0 \vee \text{Valid}(\text{sID}^*) = 0</math> 11   <b>return</b> 0 12 <b>if</b> <math>\text{Resp}[\text{sID}^*] \neq t^*</math>: <b>return</b> 0 13 <b>return</b> <math>b'</math> </pre> <p><b>Oracle</b> <math>\text{DER}_I(\text{sID}, M)</math></p> <pre> 14 <b>if</b> <math>\text{Used}[\text{sID}] = 1 \vee \text{St}[\text{sID}] = \perp</math> 15   <math>\vee \text{SK}[\text{sID}] \neq \perp</math>: <b>return</b> <math>\perp</math> 16 <math>\text{Used}[\text{sID}] := 1, \text{st} := \text{St}[\text{sID}]</math> 17 <math>(i, j) := (\text{Init}[\text{sID}], \text{Resp}[\text{sID}])</math> 18 <b>let</b> <math>(\widetilde{\text{ct}}, \widetilde{\text{ct}}_i) := M</math> 19 <b>let</b> <math>(\text{pk}, \text{sk}, \text{ct}_j, K_j) := \text{st}</math> 20 <math>\widetilde{K} := \text{Decaps}_2(\text{sk}, \widetilde{\text{ct}})</math> 21 <b>if</b> <math>i = t^*</math>: <math>K_i := \text{DEC}(\widetilde{\text{ct}}_i)</math> 22 <b>else</b> <math>K_i := \text{Decaps}_1(\text{sk}_i, \widetilde{\text{ct}}_i)</math> 23 <math>\text{ctxt} := (\text{pk}_i, \text{pk}_j, \widetilde{\text{pk}}, \text{ct}_i, \text{ct}_j, \widetilde{\text{ct}})</math> 24 <math>\text{SK} := \text{H}(\text{ctxt}, K_i, K_j, \widetilde{K})</math> 25 <math>(\text{R}[\text{sID}], \text{SK}[\text{sID}]) := (M, \text{SK})</math> 26 <b>return</b> 1 </pre> <p><b>Oracle</b> <math>\text{COR}(i)</math></p> <pre> 27 <b>if</b> <math>i = t^*</math>: abort the game and output 0 28 <math>\text{Cor}[i] := 1</math> 29 <b>return</b> <math>\text{sk}'_i</math> </pre>	<p><b>Oracle</b> <math>\text{SESSION}_I((i, j) \in [N]^2)</math></p> <pre> 30 <math>\text{cnt} := \text{cnt} + 1, \text{sID} := \text{cnt}</math> 31 <math>(\text{Init}[\text{sID}], \text{Resp}[\text{sID}]) := (i, j)</math> 32 <math>\text{Type}[\text{sID}] := \text{"In"}</math> 33 <math>(\text{pk}, \text{sk}) \leftarrow \text{KGen}_2(\widetilde{\text{par}})</math> 34 <math>(\text{ct}_j, K_j) \leftarrow \text{Encaps}_1(\text{pk}_j)</math> 35 <b>if</b> <math>j = t^*</math> 36   <math>(\text{ct}_j, K_j) := (\text{c}[\text{sID}], \mathbf{K}[\text{sID}])</math> 37   <math>\mathcal{L}_1 := \mathcal{L}_1 \cup \{(\text{ct}_j, K_j)\}</math> 38 <math>(\widetilde{\text{pk}}, \widetilde{\text{sk}}) \leftarrow \text{KGen}_2(\widetilde{\text{par}})</math> 39 <math>\text{st} := (\widetilde{\text{pk}}, \widetilde{\text{sk}}, \text{ct}_j, K_j), M_i := (\widetilde{\text{pk}}, \text{ct}_j)</math> 40 <math>(\text{I}[\text{sID}], \text{St}[\text{sID}]) := (M_i, \text{st})</math> 41 <b>return</b> <math>(\text{sID}, M_i)</math> </pre> <p><b>Oracle</b> <math>\text{SESSION}_R((i, j) \in [N]^2, M)</math></p> <pre> 42 <math>\text{cnt} := \text{cnt} + 1, \text{sID} := \text{cnt}</math> 43 <math>(\text{Init}[\text{sID}], \text{Resp}[\text{sID}]) := (i, j)</math> 44 <math>\text{Type}[\text{sID}] := \text{"Re"}</math> 45 <b>let</b> <math>(\widetilde{\text{pk}}, \text{ct}_j) := M</math> 46 <math>(\widetilde{\text{ct}}, \widetilde{K}) \leftarrow \text{Encaps}_2(\widetilde{\text{pk}})</math> 47 <b>if</b> <math>j = t^*</math> 48   <b>if</b> <math>\exists K</math> s.t. <math>(\text{ct}_j, K) \in \mathcal{L}_1</math> 49     <math>K_j := K</math> 50   <b>else</b> <math>K_j := \text{DEC}(\widetilde{\text{ct}}_j)</math> 51 <b>else</b> <math>K_j := \text{Decaps}_1(\text{sk}_j, \text{ct}_j)</math> 52 <math>(\text{ct}_i, K_i) \leftarrow \text{Encaps}_1(\text{pk}_i)</math> 53 <math>\text{ctxt} := (\text{pk}_i, \text{pk}_j, \widetilde{\text{pk}}, \text{ct}_i, \text{ct}_j, \widetilde{\text{ct}})</math> 54 <math>\text{SK} := \text{H}(\text{ctxt}, K_i, K_j, \widetilde{K})</math> 55 <math>\text{SK}[\text{sID}] := \text{SK}, M_j := (\widetilde{\text{ct}}, \text{ct}_i)</math> 56 <math>(\text{I}[\text{sID}], \text{R}[\text{sID}]) := (M, M_j)</math> 57 <b>return</b> <math>(\text{sID}, M_j)</math> </pre>
---	--

Figure 17: The reduction in the proof of Lemma 6.5. The highlighted codes show how the reduction embeds the challenges into the AKE sessions. Oracles  $O_2$  and  $\text{H}$  are simulated in the same way with that in Figure 16. □

## References

- [1] Alwen, J., Krenn, S., Pietrzak, K., Wichs, D.: Learning with rounding, revisited - new reduction, properties and applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 57–74. Springer, Heidelberg (Aug 2013) (Cited on page 6.)
- [2] 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) (Cited on page 3, 4, 5.)
- [3] Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (Apr 2009) (Cited on page 7.)
- [4] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993) (Cited on page 3.)

- [5] Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (Aug 1994) (Cited on page 4.)
- [6] Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (May / Jun 2006) (Cited on page 5.)
- [7] 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) (Cited on page 3, 5.)
- [8] Chen, J., Wee, H.: Fully, (almost) tightly secure IBE and dual system groups. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 435–460. Springer, Heidelberg (Aug 2013) (Cited on page 3.)
- [9] Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 467–484. Springer, Heidelberg (May 2012) (Cited on page 3, 4.)
- [10] Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In: Chen, K., Xie, Q., Qiu, W., Li, N., Tzeng, W.G. (eds.) ASIACCS 13. pp. 83–94. ACM Press (May 2013) (Cited on page 3.)
- [11] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology* 26(1), 80–101 (Jan 2013) (Cited on page 4.)
- [12] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. *Cryptology ePrint Archive, Report 2007/432* (2007), <https://eprint.iacr.org/2007/432> (Cited on page 6.)
- [13] 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) (Cited on page 4, 8, 11.)
- [14] Gjøsteen, K., Jager, T.: Practical and tightly-secure digital signatures and authenticated key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 95–125. Springer, Heidelberg (Aug 2018) (Cited on page 3.)
- [15] Han, S., Jager, T., Kiltz, E., Liu, S., Pan, J., Riepel, D., Schäge, S.: Authenticated key exchange and signatures with tight security in the standard model. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 670–700. Springer, Heidelberg, Virtual Event (Aug 2021) (Cited on page 3.)
- [16] Hemenway, B., Libert, B., Ostrovsky, R., Vergnaud, D.: Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 70–88. Springer, Heidelberg (Dec 2011) (Cited on page 4, 7.)
- [17] 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) (Cited on page 4.)
- [18] Hofheinz, D., Jager, T., Rupp, A.: Public-key encryption with simulation-based selective-opening security and compact ciphertexts. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 146–168. Springer, Heidelberg (Oct / Nov 2016) (Cited on page 7.)
- [19] Hövelmanns, K., Kiltz, E., Schäge, S., Unruh, D.: Generic authenticated key exchange in the quantum random oracle model. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 389–422. Springer, Heidelberg (May 2020) (Cited on page 3, 4, 6, 12, 13, 14, 15, 16, 20, 24.)

- [20] Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 117–146. Springer, Heidelberg (Oct 2021) (Cited on page 3, 20, 23.)
- [21] Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 96–125. Springer, Heidelberg (Aug 2018) (Cited on page 5, 12, 13, 14, 16.)
- [22] Jiang, H., Zhang, Z., Ma, Z.: Key encapsulation mechanism with explicit rejection in the quantum random oracle model. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 618–645. Springer, Heidelberg (Apr 2019) (Cited on page 5.)
- [23] Katsumata, S., Yamada, S., Yamakawa, T.: Tighter security proofs for GPV-IBE in the quantum random oracle model. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 253–282. Springer, Heidelberg (Dec 2018) (Cited on page 6.)
- [24] Kiltz, E., Lyubashevsky, V., Schaffner, C.: A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 552–586. Springer, Heidelberg (Apr / May 2018) (Cited on page 5.)
- [25] Krawczyk, H.: SIGMA: The “SIGn-and-MAc” approach to authenticated Diffie-Hellman and its use in the IKE protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (Aug 2003) (Cited on page 3.)
- [26] Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (Aug 2005) (Cited on page 3.)
- [27] Libert, B., Sakzad, A., Stehlé, D., Steinfeld, R.: All-but-many lossy trapdoor functions and selective opening chosen-ciphertext security from LWE. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 332–364. Springer, Heidelberg (Aug 2017) (Cited on page 4.)
- [28] Liu, X., Wang, M.: QCCA-secure generic key encapsulation mechanism with tighter security in the quantum random oracle model. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 3–26. Springer, Heidelberg (May 2021) (Cited on page 16.)
- [29] Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. In: 45th FOCS. pp. 372–381. IEEE Computer Society Press (Oct 2004) (Cited on page 6.)
- [30] Pan, J., Wagner, B., Zeng, R.: Lattice-based authenticated key exchange with tight security. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023. pp. 616–647. Springer Nature Switzerland, Cham (2023) (Cited on page 5.)
- [31] Pan, J., Wagner, B., Zeng, R.: Tighter security for generic authenticated key exchange in the qrom. Cryptology ePrint Archive (2023) (Cited on page .)
- [32] Pan, J., Zeng, R.: Compact and tightly selective-opening secure public-key encryption schemes. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part III. LNCS, vol. 13793, pp. 363–393. Springer, Heidelberg (Dec 2022) (Cited on page 4.)
- [33] 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) (Cited on page 4, 8, 9, 11.)
- [34] Saito, T., Xagawa, K., Yamakawa, T.: Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 520–551. Springer, Heidelberg (Apr / May 2018) (Cited on page 5, 6, 14, 16.)
- [35] Unruh, D.: Post-quantum verification of Fujisaki-Okamoto. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 321–352. Springer, Heidelberg (Dec 2020) (Cited on page 16.)

- [36] Xue, H., Au, M.H., Yang, R., Liang, B., Jiang, H.: Compact authenticated key exchange in the quantum random oracle model. Cryptology ePrint Archive, Report 2020/1282 (2020), <https://eprint.iacr.org/2020/1282> (Cited on page 4.)
- [37] Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 758–775. Springer, Heidelberg (Aug 2012) (Cited on page 14.)