Treating dishonest ciphertexts in post-quantum KEMs – explicit vs. implicit rejection in the FO transform

Kathrin Hövelmanns and Mikhail Kudinov

Eindhoven University of Technology, Eindhoven, Netherlands
`kathrin@hoevelmanns.net, mishel.kudinov@gmail.com`

Abstract. We revisit a basic building block in the endeavor to migrate to post-quantum secure cryptography, Key Encapsulation Mechanisms (KEMs). KEMs enable the establishment of a shared secret key, using only public communication. When targeting chosen-ciphertext security against quantum attackers, the go-to method is to design a Public-Key Encryption (PKE) scheme and then apply a variant of the PKE-to-KEM conversion known as the Fujisaki-Okamoto (FO) transform, which we revisit in this work. Intuitively, FO ensures chosen-ciphertext security by rejecting dishonest messages. This comes in two flavors – the KEM could reject by returning 'explicit' failure symbol $\perp$ or instead by returning a pseudo-random key ('implicit' reject). During the NIST post-quantum standardization process, designers chose implicit rejection, likely due to the availability of security proofs against quantum attackers. On the other hand, implicit rejection introduces complexity and can easily deteriorate into explicit rejection in practice. While it was proven [BHH$^+$19] that implicit rejection is not less secure than explicit rejection, the other direction was less clear. This is relevant because the available security proofs against quantum attackers still leave things to be desired. When envisioning future improvements, due to, e.g., advancements in quantum proof techniques, having to treat both variants separately creates unnecessary overhead.
In this work, we thus re-evaluate the relationship between the two design approaches and address the so-far unexplored direction: in the classical random oracle model, we show that explicit rejection is not less secure than implicit rejection, up to a rare edge case. This, however, uses the observability of random oracle queries. To lift the proof into the quantum world, we make use of the extractable QROM (eQROM). As an alternative that works without the eQROM, we give an indirect proof that involves a new necessity statement on the involved PKE scheme.
Keywords: Post-quantum, Public-key encryption, Key Encapsulation, Fujisaki-Okamoto transform, QROM, NIST.

1 Introduction

Post-quantum cryptography is a critical topic, as the development of quantum computing technology poses a significant threat to the security of the currently deployed cryptographic algorithms. This threat necessitates to explore and implement new cryptographic techniques and protocols that are capable of withstanding potential quantum-based attacks. The two most basic building blocks are digital signatures and Key Encapsulation Mechanisms (KEMs), which were recently targeted in the NIST PQC standardization process [NIS]. This work is concerned with KEMs, more specifically, we revisit the Fujisaki-Okamoto (FO) transform [FO99,FO13]. Dent [Den03, Table 5] and later [HHK17] gave FO adaptations for KEMs, which by now have become the de-facto standard for building post-quantum secure KEMs. Notably, all KEM submissions to the NIST PQC standardization process

K.H. was supported by an NWO VENI grant (Project No. VI.Veni.222.397). Date: January 14, 2025

which made it to later rounds used some variant of FO [NIS]. An important contribution in [HHK17] was that the paper provided a security proof against quantum attackers, by analyzing the FO transform in the Quantum Random Oracle model (QROM).

Intuitively, FO ensures chosen-ciphertext security by rejecting dishonest messages: when asked to compute the key corresponding to a received ciphertext, the KEM will first perform a sanity check (sometimes called 're-encryption check') on the received ciphertext and only derive the key if the ciphertext passes the check. Otherwise, the KEM rejects. Rejection comes in two flavors – an FO-KEM could reject by returning 'explicit' failure symbol $\perp$ ('explicit' reject, sometimes denoted as $\mathsf{FO}^{\perp}$) or instead by returning a pseudo-random key ('implicit' reject, sometimes denoted as $\mathsf{FO}^{\not\perp}$). During the NIST post-quantum standardization process, designers chose implicit rejection, likely because it was proven secure against quantum attackers much earlier [HHK17], and with follow-up proofs allowing for better parameters [SXY18, JZC+18, BHH+19, HKSU20, KSS+20] than the ones for explicit rejection [JZM19, DFMS22] at least until more recent results [HHM22, HM24]. These two more recent results allowed parameters close to the ones for implicit rejection – by making use of an emerging new quantum proof technique, the extractable Quantum Random Oracle [DFMS22] (eQROM), a better security bound proof could be given.

With the currently best known security bounds being relatively close for both variants, the question arises how much their security actually differs. Before making the case why this question might be relevant, we survey what is known: it is known [BHH+19] that implicit rejection is at least as secure as explicit rejection, so designers of an explicitly rejecting KEM can freely switch to implicit rejection without deteriorating security. (Intuitively, this might feel natural since implicit rejection simply replaces the explicit failure symbol with something that at least has the format of a proper key, so in a sense, the attacker might learn less from such responses.) Implicit rejection, however, introduces some complexity: it involves more hashing, and unless special care is taken, practice can turn implicit rejects into explicit ones, e.g., due to timing information, implementations raising (and leaking) internal flags, or because the larger infrastructure that uses the KEM aborts connections once it notices that the key is improper. If designers want to be on the safe side and account for such potential deterioration, they might indeed want to design their KEM such that even its explicit rejection counterpart has a security proof.

The other direction of the question – whether explicit rejection is as secure as implicit rejection – so far was not formally analyzed. As a result, it is not completely clear what a KEM actually gains in terms of security (if anything at all) when choosing implicit over explicit rejection, which might be dissatisfying for designers that want to choose their concrete KEM design. Additionally, this missing link might create unnecessary work in the future: neither variant has an easily applicable security proof against quantum attackers that is 'tight' – if designers want to base the KEM's security on the security of the underlying PKE scheme, using a standard notion of PKE security, the parameters need to be chosen much larger to mitigate quantum attacks than when only aiming to mitigate classical attacks. This lack of tightness stems from how the proofs argue key indistinguishability – to do so, they use a quantum proof technique called One-Way-To-Hiding that (so far) incurs a security loss. The closeness of currently known bounds does not rule out that future advancements in quantum proof techniques can tighten up one variant, but not the other, which would unveil one variant as significantly more secure than the other. Even if both variants can be tightened up, having to treat both variants separately creates unnecessary overhead. We note that previous work [BHH+19] gave a result for explicit rejection with key confirmation, i.e., for another KEM variant that includes an additional hash into the ciphertext, but that the proof crucially relies on this additional hash and that the design thus suffers from additional computation and communication overhead.

Our contribution. For FO-KEMs that are built from probabilistic schemes with sufficient entropy, we will show that explicit rejection is as secure against classical attackers as implicit rejection, up to one edge case that in practice will be very rare. For schemes with imperfect correctness, i.e., for schemes where decrypting sometimes does not return the originating plaintext, it can be bounded by computational correctness notions. We decided to give a more fine-grained analysis (see Definition 9) to enable better bounds and to capture how rare this edge case will be in practice. Notably, the edge case does not occur at all if the involved encryption scheme is perfectly correct, i.e., if decrypting always returns the originating plaintext.

We stress that our approach only applies to KEMs that are built from probabilistic schemes and that follow the FO paradigm of computing encryptions as $c := \mathsf{Enc}(\mathsf{pk}, m; r)$ with randomness $r := \mathsf{Hash}(m)$, like, e.g., Kyber. This is because the proof crucially exploits that the randomness-generating hash function Hash is modeled as a random oracle.

Our proof does not directly carry over to the quantum setting – in the classical setting, we model the involved hash functions as random oracles and the proof exploits that all adversarial oracle queries can be observed. In the quantum-accessible ROM, we thus offer two alternative approaches: the first one makes use of the more recent QROM techniques, in particular of the extractable QROM [DFMS22]. The extractable QROM is a compressed oracle [Zha19] that provides an additional 'extraction' interface Ext. Essentially, Ext can serve as a QROM counterpart to book-keeping random oracle queries. While we will argue that attackers do not gain anything from interacting with Ext, security proofs heavily benefit from it. In our case, Ext enables a tight reduction between the two rejection methods. More specifically, given an adversary against an explicitly rejecting FO-KEM, one can again construct an adversary against the implicitly rejecting counterpart, with the proof resembling its classical counterpart up to how the relevant oracle queries are found. We also briefly discuss a second, alternative approach that dispenses with the extractable QROM, but comes at the price of being far less tight. This approach combines already known relations with a new necessity statement on the involved PKE scheme: we use that if the implicitly rejecting KEM is IND-CCA secure, then the involved PKE scheme *must* be Oneway-secure. Consequentially [HHM22, Theorem 9], the explicitly rejecting KEM then is secure as well.

Organization of this paper. After recalling the necessary definitions/notation in Section 2 as well as – for the reader's convenience – relevant previous results in Section 3, we start by showing the result for classical attackers, giving a reduction in the (classical) ROM in Section 4. Section 5 uses the extractable QROM to lift the proof to the quantum world. We finish by giving the non-tight, but extQROM-less indirect proof in Section 6.

2 Preliminaries

In this section we recall the necessary background from previous literature that we will use throughout the paper: we recall Fujisaki-Okamoto transformation, definitions and security notions for PKE schemes and KEMs, and the extractable QROM.

2.1 Public-Key Encryption

A public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms and a finite message space $\mathcal{M}$. The key generation algorithm Gen outputs a key pair $(\mathsf{pk}, \mathsf{sk})$, with pk defining a randomness space $\mathcal{R} = \mathcal{R}(\mathsf{pk})$. The encryption algorithm Enc, on input pk and a message $m \in \mathcal{M}$, produces an encryption $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ of m under the public key pk. If necessary, we explicitly specify the used randomness of encryption by writing $c := \mathsf{Enc}(\mathsf{pk}, m; r)$, where $r \leftarrow_\$ \mathcal{R}$ and $\mathcal{R}$ denotes the randomness space. The decryption algorithm Dec, on input sk and a ciphertext c, yields either a message $m = \mathsf{Dec}(\mathsf{sk}, c) \in \mathcal{M}$ or a special symbol $\perp \notin \mathcal{M}$ to show that c is not a valid ciphertext.

2.2 Fujisaki-Okamoto transformation

This subsection recalls the definition of the two FO variants $\mathsf{FO}_m^{\perp}$ and $\mathsf{FO}_m^{\not\perp}$. To a public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$, randomness space $\mathcal{R}$, and hash functions $\mathsf{G} : \mathcal{M} \to \mathcal{R}$ and $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$, we associate the two KEMs (either with rejection type $\perp$ or $\not\perp$):

$$\mathsf{KEM}_m^{\perp/\not\perp} := \mathsf{FO}_m^{\perp/\not\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}] := (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps}) \ ,$$

with the respective algorithms given in Algorithm 1 and Algorithm 2. (To reject in line 6, $\mathsf{FO}_m^{\not\perp}$ uses a seed $\mathsf{sk}.\mathit{seed}$ that was randomly chosen from the message space during key generation and is stored alongside the secret key for PKE.)

Algorithm 1: $\mathsf{Encaps}(\mathsf{pk})$
```
1 $m \leftarrow_\$ \mathcal{M}$
2 $c = \mathsf{Enc}(\mathsf{pk}; m; \mathsf{G}(m))$
3 $K = \mathsf{H}(m)$
```

Algorithm 2: $\mathsf{Decaps}(\mathsf{sk}, c)$
```
1 $m' = \mathsf{Dec}(\mathsf{sk}, c)$
2 if $m' = \perp$ or $c \neq \mathsf{Enc}(pk, m', \mathsf{G}(m'))$ then
3     if $type = \perp$ then
4         return $\perp$
5     if $type = \not\perp$ then
6         return $\mathsf{H}(c, \mathsf{sk}.seed)$
7 else
8     return $K = \mathsf{H}(m')$
```

The two KEMs use the underlying scheme PKE in a derandomized way by using $\mathsf{G}(m)$ as the encryption randomness (see line 2 of Algorithm 1) and check during decapsulation whether the decrypted plaintext does re-encrypt to the ciphertext (see line 2 of Algorithm 2). This building block of $\mathsf{FO}_m^{\perp/\not\perp}$, i.e., derandomizing PKE and introducing a re-encryption check, was formalized in [HHK17] as the following transformation T, whose result we denote by $\mathsf{PKE}^{\mathsf{G}}$:

$$\mathsf{PKE}^{\mathsf{G}} := \mathsf{T}[\mathsf{PKE}, \mathsf{G}] := \left(\mathsf{Gen}, \mathsf{Enc}^{\mathsf{G}}, \mathsf{Dec}^{\mathsf{G}}\right) ,$$

with the respective algorithms given in Algorithm 3 and Algorithm 4 in Figure 1.

Algorithm 3: $\mathsf{Enc}^{\mathsf{G}}(\mathsf{pk}, m)$
```
1 $m \leftarrow_\$ \mathcal{M}$
2 $c = \mathsf{Enc}(\mathsf{pk}; m; \mathsf{G}(m))$
3 return $c$
```

Algorithm 4: $\mathsf{Dec}^{\mathsf{G}}(\mathsf{sk}, c)$
```
1 $m' = \mathsf{Dec}(\mathsf{sk}, c)$
2 if $m' = \perp$ or
    $c \neq \mathsf{Enc}(pk, m', \mathsf{G}(m'))$ then
3     return $\perp$
4 else
5     return $m'$
```

Fig. 1: Derandomized PKE scheme $\mathsf{PKE}^{\mathsf{G}} := \mathsf{T}[\mathsf{PKE}, \mathsf{G}] := (\mathsf{Gen}, \mathsf{Enc}^{\mathsf{G}}, \mathsf{Dec}^{\mathsf{G}})$.

Following the notation established in [HHK17], we denote the transformation from a deterministic dPKE to a KEM with corresponding rejection mechanism by $\mathrm{U}_m^{\not\perp}$ $\left(\mathrm{U}_m^{\perp}\right)$. Starting from a deterministic encryption scheme dPKE and a hash function H, we build a key encapsulation mechanism $\mathsf{KEM}^{\not\perp} :=$ $\mathrm{U}_m^{\not\perp}[\mathsf{dPKE}, \mathsf{H}]$ with 'implicit' rejection by defining

$$\mathsf{Encaps}(pk) := (c \leftarrow \mathsf{Enc}(pk, m), K := \mathsf{H}(m)) \enspace ,$$

where m is picked at random from the message space, and

$$\mathsf{Decaps}^{\not\perp}(\mathsf{sk}, c) = \begin{cases} \mathsf{H}(m) & m \neq \perp \\ \mathsf{H}(c, \mathsf{sk}.seed) & m = \perp \end{cases}$$

where $m := \mathsf{Dec}(\mathsf{sk}, c)$ and $\mathsf{sk}.seed$ is a random seed which is contained in sk.

We also define the 'explicit rejection' counterpart $\mathsf{KEM}^{\perp} = \mathrm{U}_m^{\perp}\left[\mathsf{PKE}^{\mathsf{G}}, \mathsf{H}\right]$ which differs from $\mathsf{KEM}^{\not\perp}$ only in decapsulation:

$$\mathsf{Decaps}^{\perp}(\mathsf{sk}, c) = \begin{cases} \mathsf{H}(m) & m \neq \perp \\ \perp & m = \perp \end{cases}$$

where $m := \mathsf{Dec}(\mathsf{sk}, c)$.

2.3 Security notions for Public-Key Encryption schemes

In the security analysis of FO-KEMs, it is usually necessary to utilize notions of **correctness**. This captures a particular class of chosen-ciphertext attacks, where attackers found a failing ciphertext c, meaning c decrypts to a different message than its originating one, and uses this to obtain some leakage on the secret key. There are two approaches to handling such correctness errors. The first one is information-theoretic.

Definition 1 (Correctness (inf. theor.) [HHK17]). *We call a public-key encryption scheme* PKE *δ-correct if*

$$\mathbf{E}\left[\max_{m \in \mathcal{M}} \Pr[\mathsf{Dec}(\mathsf{sk}, c) \neq m | c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)]\right] \leq \delta.$$

where the expectation is taken over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$.

In the security analysis of FO-KEMs, it is usually necessary to use a correctness term for the derandomized version of PKE.

Definition 2 (Deterministic Correctness (inf. theor.) [HHK17]). *We call the deterministic public-key encryption scheme* $\mathsf{PKE}^{\mathsf{G}}$ *δ_1-correct if*

$$\mathbf{E}\left[\max_{m \in \mathcal{M}} \Pr[\mathsf{Dec}^{\mathsf{G}}(\mathsf{sk}, c) \neq m | c \leftarrow \mathsf{Enc}(\mathsf{pk}, m, \mathsf{G}(m))]\right] \leq \delta_1.$$

where the expectation is taken over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$.

According to Theorem 3.1 [HHK17] the correctness of a $\mathsf{PKE}^{\mathsf{G}}$ can be related to correctness of PKE as $\delta_1 = q_G \cdot \delta$, where q_G is the number of queries to G in the classical ROM. In the QROM, δ_1 is bounded by $\delta_1 \leq 8(q_G + 1)^2 \cdot \delta$ [HHK17, Lemma 4.3] .

This reflects that even a possibly unbounded adversary with access to the key pair cannot create a failing ciphertext with a probability bigger than δ_1. A computational (game-based) approach to capturing this was introduced in [HHM22].

Definition 3 (FFP-ATK). *Let* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a deterministic public-key encryption scheme. For* $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$*, we define* FFP-ATK *games as in Algorithm 5, where*

$$\mathrm{O}_{\mathsf{ATK}} := \begin{cases} - & \mathsf{ATK} = \mathsf{CPA} \\ \mathsf{oDecrypt} & \mathsf{ATK} = \mathsf{CCA} \end{cases}$$

We define the FFP-ATK *advantage function of an adversary* $\mathcal{A}$ *against PKE as*

$$\mathrm{Adv}_{\mathsf{PKE}}^{\mathsf{FFP-ATK}}(\mathcal{A}) := \Pr\left[\mathsf{FFP-ATK}_{\mathsf{PKE}}^{\mathcal{A}} \Rightarrow 1\right].$$

Although the computational approach might give a tighter bound, we will at times stick to the somewhat simpler information-theoretic notion.

Algorithm 5: FFP-ATK
```
1 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$
2 $m \leftarrow \mathcal{A}^{\mathrm{O}_{\mathsf{ATK}}, \mathsf{G}}(\mathsf{pk})$
3 $c \leftarrow \mathsf{Enc}^{\mathsf{G}}(\mathsf{pk}, m)$
4 $m' = \mathsf{Dec}^{\mathsf{G}}(sk, c)$
5 return $[\![m' \neq m]\!]$
```

Algorithm 6: oDecrypt(c)
```
1 $m = \mathsf{Dec}(sk, c)$
2 return $m$
```

Definition 4 (Injectivity of PKE schemes [BHH+19]). *A* $\mathsf{dPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* ϵ*-injective if*

$$\Pr[\mathsf{Enc}(\mathsf{pk}, m) \textit{ is not injective } : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(), H \leftarrow \mathcal{H}] \leq \epsilon$$

We say dPKE *is injective if* $\epsilon = 0$*. We say that an probabilistic PKE scheme is injective if for all public keys pk, all* $m \neq m'$ *and all coins* r, r'*, we have* $\mathsf{Enc}(\mathsf{pk}, m, r) \neq \mathsf{Enc}(\mathsf{pk}, m', r')$*.*

In the security analysis of FO-KEMs, it will also be necessary to capture that adversaries might be able to create valid ciphertexts without knowing the respective plaintext. This will involve the following notion about the entropy (or 'spreadness') of the PKE scheme.

Definition 5 (γ-spreadness [HHM22]). *We say that* PKE *is* γ*-spread iff for all key pairs* $(\mathsf{pk}, \mathsf{sk}) \in \mathrm{supp}(\mathsf{Gen})$ *and all messages* $m \in \mathcal{M}$ *it holds that*

$$\max_{c \in \mathcal{C}} \Pr[\mathsf{Enc}(\mathsf{pk}, m) = c] \leq 2^{-\gamma}$$

where the probability is taken over the internal randomness Enc*.*

Lastly, we define one-wayness under different models (OW-ATK) for PKE schemes.

Definition 6 (OW-ATK). *Let* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public-key encryption scheme with message space* $\mathcal{M}$*. We now define three security notions for public-key encryption: One-Wayness under Chosen Plaintext Attacks (*OW-CPA*), One-Wayness under Plaintext Checking Attacks (*OW-PCA*), and One-Wayness under Plaintext and Validity Checking Attacks (*OW-PCVA*). We define these games as in Algorithm 7 and a corresponding advantage function of an adversary* $\mathcal{A}$ *against* PKE *as*

$$\mathrm{Adv}_{\mathsf{PKE}}^{\mathsf{OW\text{-}ATK}}(\mathcal{A}) = \Pr\left[\mathsf{OW\text{-}ATK}_{\mathsf{PKE}}^{\mathcal{A}} \Rightarrow 1\right]$$

Where a corresponding oracle will be defined as

$$\mathrm{O}_{\mathsf{ATK}} := \begin{cases} - & \mathsf{ATK} = \mathsf{CPA} \\ \mathrm{Pco}(\cdot, \cdot) & \mathsf{ATK} = \mathsf{PCA} \\ \mathrm{Cvo}(\cdot) & \mathsf{ATK} = \mathsf{VA} \\ \mathrm{Pco}(\cdot, \cdot), \mathrm{Cvo}(\cdot) & \mathsf{ATK} = \mathsf{PCVA} \end{cases}$$

Algorithm 7: ATK-CPA

```
1 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$
2 $m^* \leftarrow_\$ \mathcal{M}$
3 $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m^*)$
4 $m' \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{ATK}}}(\mathsf{pk}, c^*)$
5 return $[\![m' = m^*]\!]$
```

Algorithm 8: ATK oracles

```
1 $\mathrm{Pco}(m \in \mathcal{M}, c)$ :
2 return $[\![\mathsf{Dec}(sk, c) = m]\!]$

3 $\mathrm{Cvo}(c \neq c^*)$ :
4 $m = \mathsf{Dec}(\mathsf{sk}, c)$
5 return $[\![m \in \mathcal{M}]\!]$
```

2.4 Security notions for Key Encapsulation Mechanisms

Here, we define security notions for key encapsulation: Indistinguishability under Chosen Plaintext Attacks (IND-CPA) and Chosen Ciphertext Attacks (IND-CCA).

Definition 7 (IND-CPA). *We define the* IND-CPA *game as in Algorithm 9 and the* IND-CPA *advantage function of an adversary $\mathcal{A}$ (with binary output) against* KEM *as*

$$\mathrm{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CPA}}(\mathcal{A}) :=| \Pr\left[\mathsf{IND\text{-}CPA}_{\mathsf{KEM}}^{\mathcal{A}} \Rightarrow 1\right] - 1/2.$$

Definition 8 (IND-CCA). *We define the* IND-CCA *game as in Algorithm 10 and the* IND-CCA *advantage function of an adversary $\mathcal{A}$ (with binary output) against KEM as*

$$\mathrm{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA}}(\mathcal{A}) :=| \Pr\left[\mathsf{IND\text{-}CCA}_{\mathsf{KEM}}^{\mathcal{A}} \Rightarrow 1\right] - 1/2.$$

Algorithm 9: IND-CPA

```
1 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$
2 $b \leftarrow_\$ \{0, 1\}$
3 $(K_0^*, c^*) \leftarrow \mathsf{Encaps}(\mathsf{pk})$
4 $K_1^* \leftarrow_\$ \mathcal{K}$
5 $b' \leftarrow \mathcal{A}(c^*, K_b^*)$
6 return $[\![b' = b]\!]$
```

Algorithm 10: IND-CCA

```
1 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$
2 $b \leftarrow_\$ \{0, 1\}$
3 $(K_0^*, c^*) \leftarrow \mathsf{Encaps}(\mathsf{pk})$
4 $K_1^* \leftarrow_\$ \mathcal{K}$
5 $b' \leftarrow \mathcal{A}^{\mathsf{Decaps}}(c^*, K_b^*)$
6 return $[\![b' = b]\!]$
```

In [HHK17] Theorem 3.4, the authors showed a reduction of IND-CCA of $\mathsf{KEM}^{\not\perp}$ security to OW-PCA security of $\mathsf{PKE}^{\mathsf{G}}$ in ROM. While in Theorem 3.3 they showed a reduction of IND-CCA of $\mathsf{KEM}^{\perp}$ security to OW-PCVA security of $\mathsf{PKE}^{\mathsf{G}}$ in ROM.

2.5 Extractable Compressed Random Oracle

In this section, we want to describe extractable Compressed Oracle (eCO) [DFMS22]. In our context, it was used in the proofs of [HHM22], which allowed the authors to simulate decapsulation queries and produce a tighter bound. This model is an extension of Zhandry's Compressed Random Oracle [Zha19]. eCO implements two interfaces; one is used for quantum-accessible random oracle, and the second one is used for extraction queries. Let's look at them in more detail.

The random oracle interface is usually denoted as eCO.RO while the extraction interface is denoted as eCO.Ext. The extraction mechanism depends on some function f so we can sometimes write $\mathsf{eCO.Ext}_f$ if f is not obvious from the context. The extraction query must be classical and consist of some target value t. The response simulates a quantum measurement that "collapses" the Oracle database, allowing it to yield a specific outcome x. After the measurement, the database is in a state where all the values for $\mathsf{eCO.RO}(x)$ collapse to the values y that satisfy the equation $f(x, y) = t$. Moreover, x is the smallest input in the database that, with some corresponding value y, satisfies the function f.

Now, we present the formal definitions of eCO from [DFMS22, HM24]. We also do not go into detail about the efficiency of eCO implementation. An interested reader can look further details up in [DFMS22].

The simulator eCO for a random function O: $\{0,1\}^m \to \{0,1\}^n$ is a stateful oracle with a state stored in a quantum register $D = D_{0^m} \dots D_{1^m}$, where for each input value $x \in \{0,1\}^m$, register D_x has $n+1$ qubits used to store superpositions of n-bit output strings y, encoded as $0y$, and an additional symbol $\perp$, encoded as 10^n. We adopt the convention that an operator expecting n input qubits acts on the last n qubits when applied to one of the registers D_x. The compressed oracle has the following three components.

1. The initial state of the oracle, $|\phi\rangle = |\perp\rangle^{2^m}$
2. A quantum query with query input register X and output register Y is answered using the oracle unitary O_{XYD} defined by
$$O_{XYD}|x\rangle_X = |x\rangle_X \otimes \left(F_{D_x} \mathrm{CNOT}^{\otimes n}_{D_x:Y} F_{D_x}\right),$$
where $F|\perp\rangle = |\phi_0\rangle, F|\phi_0\rangle = |\perp\rangle$ and $F|\psi\rangle = |\psi\rangle$ for all $|\psi\rangle$ such that $\langle \psi \mid \perp\rangle = \langle \psi \mid \phi_0\rangle = 0$, with $|\phi_0\rangle = |+\rangle^{\otimes n}$ being the uniform superposition.
3. A recovery algorithm that recovers a standard QRO O: apply $F^{\otimes 2^m}$ to D and measure it to obtain the function table of O.

The extraction interface works the following way. Given a random oracle $\mathrm{O} : \{0,1\}^m \to \{0,1\}^n$, let $f : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^\ell$ be a function. We define a family of measurements $(\mathcal{M}^t)_{t \in \{0,1\}^\ell}$. The measurement $\mathcal{M}^t$ has measurement projectors $\{\Sigma^{t,x}\}_{x \in \{0,1\}^m \cup \{\emptyset\}}$ defined as follows. For $x \in \{0,1\}^m$, the projector selects the case where D_x is the first (in lexicographical order) register that contains y such that $f(x,y) = t$, i.e.

$$\Sigma^{t,x} = \bigotimes_{x' < x} \bar{\Pi}^{t,x'}_{D'_x} \otimes \Pi^{t,x}_{D_x}, \quad \text{with} \quad \Pi^{t,x} = \sum_{\substack{y \in \{0,1\}^n: \\ f(x,y)=t}} |y\rangle\langle y|$$

and $\bar{\Pi} = \mathbb{1} - \Pi$. The remaining projector corresponds to the case where no register contains such a y, i.e.

$$\Sigma^{t,\emptyset} = \bigotimes_{x' \in \{0,1\}^m} \bar{\Pi}^{t,x'}_{D'_x}$$

eCO is initialized with the initial state of the compressed oracle. eCO.RO is quantum-accessible, eCO.Ext is a classical oracle interface that, on input t, applies $\mathcal{M}^t$ to eCO's internal state.

The main lemma used in our proof declares how extraction queries affect RO behavior. To define it, we first need to introduce some notions.

- $R_{f,t}(x,y) :\Leftrightarrow f(x,y) = t$.
- $\Gamma_R := \max_x |\{y \mid R(x,y)\}|$.
- $\Gamma(f) = \max_t \Gamma_{R_{f,t}}$.

Lemma 1 (Part of theorem 3.4 in [DFMS22], formulated in [HHM22]). *The extractable RO simulator* eCO *described above, with interfaces* eCO.RO *and* eCO.Ext*, satisfies the following properties.*

1. *If* $\mathsf{eCO.Ext}$ *is unused,* eCO *is perfectly indistinguishable from a random oracle.*
2. *Any two subsequent independent queries to* $\mathsf{eCO.RO}$ *commute. In particular, two subsequent classical* $\mathsf{eCO.RO}$*-queries with the same input x give identical responses.*
3. *Any two subsequent independent queries to* $\mathsf{eCO.Ext}$ *commute. In particular, two subsequent* $\mathsf{eCO.Ext}$ *queries with the same input t give identical responses.*
4. *Any two subsequent independent queries to* $\mathsf{eCO.Ext}$ *and* $\mathsf{eCO.RO}$ $8\sqrt{2\Gamma(f)/2^n}$*-almost-commute.*

3 Known Results

In this section, we revise related results from previous works for the reader's convenience. We begin with the implication result mentioned in the introduction, stating that implicit rejection is at least as secure as explicit rejection.

Theorem 1 (Explicit → implicit [BHH+19, Theorem 3]). *Let* $\mathsf{PKE}^{\mathsf{G}}$ *be a derandomized* PKE*. Let $\mathcal{A}$ be an* $\mathsf{IND\text{-}CCA}$ *adversary against* $\mathrm{U}_m^{\not\perp}(\mathsf{PKE}^{\mathsf{G}}, \mathrm{F}, H)$*. Then there is an* $\mathsf{IND\text{-}CCA}$ *adversary $\mathcal{B}$ against* $\mathrm{U}_m^{\perp}(\mathsf{PKE}^{\mathsf{G}}, H)$*, running in about the same time and resources as $\mathcal{B}$, such that*

$$\mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathrm{U}_m^{\not\perp}(\mathsf{PKE}^{\mathsf{G}},\mathrm{F},H)}(\mathcal{A}) = \mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathrm{U}_m^{\perp}(\mathsf{PKE}^{\mathsf{G}},H)}(\mathcal{B})$$

Algorithm 11: $\mathsf{Enc}_1(\mathsf{pk}, m)$
```
1 $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$
2 $t \leftarrow H_t(m)$
3 return $(c, t)$
```

Algorithm 12: $\mathsf{Dec}_1(\mathsf{sk}, (c, t))$
```
1 $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 if $H_t(m') \neq t$ then
3     return $\perp$
4 return $m'$
```

Fig. 2: PKE scheme $C(\mathsf{PKE}, H_t, \tau)$: PKE with added key confirmation of length τ.

As already pointed out in the introduction, there also exists a proof in the other direction for explicitly rejecting KEMs that additionally add key confirmation to the ciphertexts. We now recall the construction and the result. Let τ be the number of bits used for the key-confirmation tag. To PKE scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and random oracle $H_t : \mathcal{M} \to \{0,1\}^\tau$, [BHH+19] associated the transformed scheme $C(\mathsf{PKE}, H_t, \tau) = (\mathsf{Gen}, \mathsf{Enc}_1, \mathsf{Dec}_1)$ that adds key confirmation to PKE, formally defined in Algorithm 11 and Algorithm 12 in Figure 2. We note that the security bound for the resulting KEM (see below) contains a term relative to τ and to q_{Decaps}, the number of decapsulation queries. In practice, this means that the ciphertext's key confirmation portion must be sufficiently long to rule out chosen-ciphertext attacks.

Theorem 2 (Implicit → explicit with key confirmation [BHH+19, Theorem 4]). *Let* dPKE *be an ϵ-injective derandomized* PKE*. Consider the* KEM $\mathrm{K}_1 := \mathrm{U}_m^{\perp}(C(\mathsf{dPKE}, H_t, \tau), H_s)$ *obtained from* dPKE *applying the C-transform with random oracle* $H_t : \mathcal{M} \to \{0,1\}^\tau$ *and the* $\mathrm{U}_m^{\perp}$*-transform with independent random oracle* $H_s : \mathcal{M} \to \{0,1\}^\varsigma$*. Let* $\mathrm{K}_2 := \mathrm{U}_m^{\not\perp}(\mathsf{dPKE}, \mathrm{F}, H)$ *be the KEM obtained from* dPKE *applying the* $\mathrm{U}_m^{\not\perp}$*-transform with random oracle* $H : \mathcal{M} \to \{0,1\}^{\varsigma+\tau}$*.*

If $\mathcal{A}$ is an $\mathsf{IND\text{-}CCA}$*-adversary against* K_1 *which makes* q_{Decaps} *decapsulation queries, then it is also an* $\mathsf{IND\text{-}CCA}$*-adversary against* K_2*, and there is a PRF adversary $\mathcal{B}$ against F which uses about the same time and resources as $\mathcal{A}$, such that:*

$$\mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathrm{K}_1}(\mathcal{A}) \leq 2 \cdot \mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathrm{K}_2}(\mathcal{A}) + \frac{q_{\mathsf{Decaps}}}{2^{\tau-1}} + 2 \cdot \mathrm{Adv}^{\mathrm{PRF}}_{\mathrm{F}}(\mathcal{B}) + 2\epsilon$$

For our indirect QROM proof in Section 6, we will use the following modular result from [HHM22]). Its first step bases $\mathsf{IND\text{-}CCA}$ security of explicitly rejecting FO-KEMs $\mathsf{KEM}^{\perp} := \mathsf{FO}_m^{\perp}[\mathsf{PKE}]$ on their passive security (i.e., $\mathsf{IND\text{-}CPA}$), together with the correctness of the derandomized scheme $\mathsf{PKE}^{\mathsf{G}}$ (which is captured via a computational 'Find Failing Plaintext' notion, $\mathsf{FFP\text{-}CCA}$) and gamma-spreadness of the underlying scheme PKE. Note that according to [HHM22, Remark 1], this theorem also holds for the implicitly rejecting variant $\mathsf{FO}_m^{\not\perp}[\mathsf{PKE}]$.

Theorem 3 **($\mathsf{FO}_m^{\perp}[\mathsf{PKE}]$ $\mathsf{IND\text{-}CPA}$ and $\mathsf{PKE}^{\mathsf{G}}$ $\mathsf{FFP\text{-}CCA}$ $\overset{\text{eQROM}}{\Rightarrow}$ $\mathsf{FO}_m^{\perp}[\mathsf{PKE}]$ $\mathsf{IND\text{-}CCA}$ [HHM22, Theorem 4]).** *Let PKE be a (randomized) PKE that is γ-spread, and δ-correct, and* $\mathsf{KEM}^{\perp} := \mathsf{FO}_m^{\perp}[\mathsf{PKE},\mathsf{G},\mathsf{H}]$. *Let $\mathcal{A}$ be an* $\mathsf{IND\text{-}CCA}$ *adversary (in the QROM) against* $\mathsf{KEM}^{\perp}$*, making at most q_D many queries to its decapsulation oracle* ODECAPS*, and making $q_{\mathsf{G}}, q_{\mathsf{H}}$ queries to its respective random oracles. Let furthermore d and w be the combined query depth and query width of $\mathcal{A}$'s random oracle queries. Then there exist an* $\mathsf{IND\text{-}CPA}$*-KEM adversary $\tilde{\mathcal{A}}$ and an FFP-CCA adversary $\mathcal{B}$ against* $\mathsf{PKE}^{\mathsf{G}}$*, both in the* $\mathrm{eQROM}_{\mathrm{Enc}}$*, such that*

$$\mathrm{Adv}_{\mathsf{KEM}^{\perp}}^{\mathsf{IND\text{-}CCA}}(\mathcal{A}) \leq \mathrm{Adv}_{\mathsf{KEM}^{\perp}}^{\mathsf{IND\text{-}CPA}}(\tilde{\mathcal{A}}) + \mathrm{Adv}_{\mathsf{PKE}^{\mathsf{G}}}^{\mathsf{FFP\text{-}CCA}}(\mathcal{B}) + 12q_D\,(q_{\mathsf{G}} + 4q_D) \cdot 2^{-\gamma/2} \ .$$

The adversary $\tilde{\mathcal{A}}$ makes $q_{\mathsf{G}} + q_{\mathsf{H}} + q_D$ queries to $\mathsf{eCO.RO}$ *with a combined depth of $d + q_D$ and a combined width of w, and q_D queries to* $\mathsf{eCO.Ext}$*. Here, eCO.RO simulates* $\mathsf{G} \times \mathsf{H}$*. The adversary $\mathcal{B}$ makes q_D many queries to* $\mathsf{oDecrypt}$ *and* $\mathsf{eCO.Ext}$ *and q_{G} queries to* $\mathsf{eCO.RO}$*, and neither $\tilde{\mathcal{A}}$ nor $\mathcal{B}$ query* $\mathsf{eCO.Ext}$ *on the challenge ciphertext. The running times of the adversaries $\tilde{\mathcal{A}}$ and $\mathcal{B}$ are bounded as* $\mathrm{Time}(\tilde{\mathcal{A}}) = \mathrm{Time}(\mathcal{A}) + O(q_D)$ *and* $\mathrm{Time}(B) = \mathrm{Time}(\mathcal{A}) + O(q_D)$

The second step of the modular result picks up the threat and bases the passive security of $\mathsf{KEM}^{\perp} = \mathsf{FO}_m^{\perp}[\mathsf{PKE}]$ on $\mathsf{OW\text{-}CPA}$ security of the underlying PKE scheme. Note that this theorem also holds for the implicitly rejecting variant $\mathsf{FO}_m^{\not\perp}[\mathsf{PKE}]$ because the $\mathsf{IND\text{-}CPA}$ advantage against $\mathsf{FO}_m^{\perp}[\mathsf{PKE}]$ and $\mathsf{FO}_m^{\not\perp}[\mathsf{PKE}]$ is identical. (With the only difference being in how the KEMs reject invalid ciphertexts, the passive $\mathsf{IND\text{-}CPA}$ game is the same for both variants.)

Theorem 4 **(PKE $\mathsf{OW\text{-}CPA}$ $\overset{\text{eQROM}}{\Rightarrow}$ $\mathsf{FO}_m^{\perp}[\mathsf{PKE}]$ $\mathsf{IND\text{-}CPA}$ [HHM22, Theorem 9]).** *For any* $\mathsf{IND\text{-}CPA}$ *adversary $\mathcal{A}$ against* $\mathsf{KEM}^{\perp} := \mathsf{FO}_m^{\perp}[\mathsf{PKE},\mathsf{G},\mathsf{H}]$ *in the* $\mathrm{eQROM}_{\mathrm{Enc}}$ *that issues q many queries to eCO.RO in total, with a query depth (width) of $d(w)$, and q_E many queries to* $\mathsf{eCO.Ext}$*, where none of them is with its challenge ciphertext. there furthermore exists an OW-CPA adversary* $\mathcal{B}_{\mathsf{OW\text{-}CPA}}$ *such that*

$$\mathrm{Adv}_{\mathsf{KEM}^{\perp}}^{\mathsf{IND\text{-}CPA}}(\mathcal{A}) \leq 8d \cdot \sqrt{w \cdot \mathrm{Adv}_{\mathsf{PKE}}^{\mathsf{OW\text{-}CPA}}(\mathcal{B}_{\mathsf{OW\text{-}CPA}})}$$

The running time and quantum memory footprint of $\mathcal{B}_{\mathsf{OW\text{-}CPA}}$ satisfy $\mathrm{Time}(\mathcal{B}_{\mathsf{OW\text{-}CPA}}) = \mathrm{Time}(\mathcal{A}) + \mathrm{Time}(\mathsf{eCO}, q, q_E)$ *and* $\mathrm{QMem}(\mathcal{B}_{\mathsf{OW\text{-}CPA}}) = \mathrm{Time}(\mathcal{A}) + \mathrm{QMem}(\mathsf{eCO}, q, q_E)$

When combining the two results, the KEM's $\mathsf{IND\text{-}CCA}$ security can be based on $\mathsf{OW\text{-}CPA}$ security and gamma-spreadness of the underlying probabilistic PKE scheme PKE, but this still involves a computational correctness notion (the $\mathsf{FFP\text{-}CCA}$ term) in the extractable QROM. Since our indirect proof aims to dispense with the extractable QROM altogether, we will additionally make use of a result in [HM24] that bounds the $\mathsf{FFP\text{-}CCA}$ term, using δ-correctness of the underlying PKE scheme PKE (which is in the standard model):

Theorem 5 **(PKE δ-correct $\overset{\text{eQROM}}{\Rightarrow}$ $\mathsf{PKE}^{\mathsf{G}}$ $\mathsf{FFP\text{-}CCA}$ [HM24, Theorem 3]).** *Let* PKE *be a (randomized)* PKE *scheme that is δ-correct, and let $\mathcal{C}$ be an FFP-CCA adversary $\mathcal{C}$ against* $\mathsf{PKE}^{\mathsf{G}}$ *in the* $\mathrm{eQROM}_{\mathrm{Enc}}$*, issuing at most q_D decryption queries and q_G many queries to its eQROM oracle interface eCO.RO. Then*

$$\mathrm{Adv}_{\mathsf{PKE}^{\mathsf{G}}}^{\mathsf{FFP\text{-}CCA}}(\mathcal{C}) \leq 10\,(q_G + q_D + 1)^2\,\delta.$$

Combining the three results above, we can bound the security of $\mathsf{KEM}^{\perp}$ using δ-correctness, γ-spreadness and one-wayness of the underlying PKE scheme.

Corollary 1 **($\mathsf{OW\text{-}CPA} \rightarrow \mathsf{IND\text{-}CCA}$).** *Let PKE be a (randomized) PKE that is γ-spread and δ-correct, and let $\mathsf{KEM}^{\perp} := \mathsf{FO}_m^{\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}]$. Let $\mathcal{A}$ be an $\mathsf{IND\text{-}CCA}$ adversary against $\mathsf{KEM}^{\perp}$ in the QROM, making at most q_D many queries to its decapsulation oracle* $\mathrm{O}\textsc{Decaps}$*, and making $q_{\mathsf{G}}, q_{\mathsf{H}}$ queries to its respective random oracles. Let furthermore d and w be the combined query depth and query width of $\mathcal{A}$'s random oracle queries. Then there exist an $\mathsf{OW\text{-}CPA}$ adversary $\mathcal{B}$ against the underlying PKE such that*

$$\mathrm{Adv}_{\mathsf{KEM}^{\perp}}^{\mathsf{IND\text{-}CCA}}(\mathcal{A}) \leq \; 8(d+q_D) \cdot \sqrt{w \cdot \mathrm{Adv}_{\mathsf{PKE}}^{\mathsf{OW\text{-}CPA}}(\mathcal{B})} + 10\left(q_G + q_D + 1\right)^2 \delta + 12 q_D \left(q_{\mathsf{G}} + 4q_D\right) \cdot 2^{-\gamma/2} \; .$$

The running time and quantum memory footprint of $\mathcal{B}$ satisfy $\mathrm{Time}(\mathcal{B}) = \mathrm{Time}(\mathcal{A}) + O(q_D) + \mathrm{Time}(\mathsf{eCO}, q_{\mathsf{G}} + q_{\mathsf{H}} + q_D, q_D)$ *and* $\mathrm{QMem}(\mathcal{B}) = \mathrm{QMem}(\mathcal{A}) + \mathrm{QMem}(\mathsf{eCO}, q_{\mathsf{G}} + q_{\mathsf{H}} + q_D, q_D)$.

4 Implicit $\rightarrow$ explicit: reduction in the (classical) ROM

Before capturing quantum adversaries, we first start with a classical proof in the ROM. This will serve as a template for the subsequent quantum proof and help to build a better intuition. We want to relate the security of $\mathsf{KEM}^{\perp} := \mathsf{FO}_m^{\perp}[\mathsf{PKE}]$ to the security of $\mathsf{KEM}^{\not\perp} := \mathsf{FO}_m^{\not\perp}[\mathsf{PKE}]$. To do so, we will want to use the adversary against $\mathsf{KEM}^{\perp}$ to attack $\mathsf{KEM}^{\not\perp}$ by means of a reduction. This reduction needs to simulate decapsulation responses with explicit rejection, itself having access to implicitly rejecting decapsulations. At a first glance, it might not be obvious how our simulation can notice when it needs to reject a ciphertext, since the available oracle always returns keys that are in principle of the right format. We will build on techniques from [HHM22], where the authors implemented a simulation of the decapsulation oracle that worked without access to any decapsulation oracle at all, by instead investigating the posed random oracle queries. We will use this as an initial idea, but at the same time also utilize the extra capabilities given through the decapsulation oracle for $\mathsf{KEM}^{\not\perp}$, which allows us to give a more fine-grained security reduction.

Our reduction will fail to convince the attacker in one particular edge case, which is a particular sub-case of failing plaintexts in which the attacker did not only find a plaintext m whose (derandomized) encryption will not decrypt to m, but even a failing plaintext m that created a form of collision, meaning the decryption result m' and m will encrypt to the same ciphertext. Clearly, this edge case can be ruled out altogether if PKE has no correctness errors at all or if the encryption of PKE is injective. To bound the edge case computationally, we introduce the following computational notion, Find Failing Plaintext With Collision (FFPC). We have two types of this notion: under chosen ciphertext attack and under chosen plaintext attack. We will interpret the meaning of this sub-case after the formal definition.

Definition 9 **(FFPC).** *Let $\mathsf{PKE}^{\mathsf{G}} = (\mathsf{Gen}, \mathsf{Enc}^{\mathsf{G}}, \mathsf{Dec}^{\mathsf{G}})$ be the derandomized version of a public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as defined in Figure 1. For chosen ciphertext attack (CCA), we define an $\mathsf{FFPC\text{-}CCA}$ game for $\mathsf{PKE}^{\mathsf{G}}$ as in Algorithm 13, where the $\mathsf{FFPC\text{-}ATK}$ advantage functions of an adversary $\mathcal{A}$ against $\mathsf{PKE}^{\mathsf{G}}$ as*

$$\mathrm{Adv}_{\mathsf{PKE}^{\mathsf{G}}}^{\mathsf{FFPC\text{-}CCA}}(\mathcal{A}) := \Pr\left[\mathsf{FFPC\text{-}CCA}_{\mathsf{PKE}^{\mathsf{G}}}^{\mathcal{A}} \Rightarrow 1\right].$$

Algorithm 13: FFPC-CCA

```
1 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$
2 $m \leftarrow \mathcal{A}^{\mathsf{oDecrypt},\mathsf{G}}(\mathsf{pk})$
3 $c \leftarrow \mathsf{Enc}^{\mathsf{G}}(\mathsf{pk}, m)$
4 $m' = \mathsf{Dec}^{\mathsf{G}}(sk, c)$
5 return $[\![m' \neq m \wedge \mathsf{Enc}^{\mathsf{G}}(\mathsf{pk}, m') = \mathsf{Enc}^{\mathsf{G}}(\mathsf{pk}, m)]\!]$
```

Algorithm 14: oDecrypt(c)

```
1 $m = \mathsf{Dec}^{\mathsf{G}}(sk, c)$
2 return $m$
```

Interpretation of FFPC**.** One might wonder if this definition is necessary (in the sense that it captures an attack), or if it reflects a proof artifact. Intuitively, it seems that this case should not matter in practice: this is a case where the ciphertext undeservedly passes the full sanity check made during decapsulation, since it decrypts to a message and since the re-encryption also checks out. Consequently, there's no difference in how the two decapsulation variants respond to the ciphertext. On the other hand, it is not clear how the security proof below could be adapted to dispenses with the case. We note that FFPC-CCA can be trivially bounded by the less fine-grained correctness notion FFP-CCA– any attacker who successfully found a failing plaintext with collision obviously found a failing plaintext, thus succeeding in the FFP-CCA game. For perfectly correct schemes, this cannot happen at all, hence the FFPC-CCA advantage always will be 0 for such schemes. On the other hand, even for imperfectly correct schemes, this is a subset of failing ciphertexts that seem to be very rare.

The main result of this section indicates that FFPC-CCA indeed fully captures the difference between the two KEMs, if there is any at all: assuming that the KEMs use suitable key length and that the underlying encryption algorithm carries enough entropy (γ-spreadness), which is always needed for their security, the bound in Theorem 6 shows that the only case where one can attack the explicit KEM, but not the implicit one, indeed is the edge case captured via FFPC-CCA (which completely vanishes for perfectly correct schemes.)

Theorem 6 (Classical ROM: Implicit → Explicit). *Let PKE be a (randomized) PKE scheme that is γ-spread and let $\mathsf{KEM}^{\perp} := \mathsf{FO}_m^{\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}]$, generating keys of length n. Let $\mathcal{A}$ be an* IND-CCA *adversary (in the ROM) against $\mathsf{KEM}^{\perp}$, making at most q_D many queries to its decapsulation oracle* oDecaps. *Then there exist an* IND-CCA *adversary $\tilde{\mathcal{A}}$ against $\mathsf{KEM}^{\not\perp} := \mathsf{FO}_m^{\not\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}]$ and an* FFPC-CCA *adversary $\mathcal{B}$ against $\mathsf{PKE}^{\mathsf{G}}$ such that*

$$\mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}^{\perp}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}^{\not\perp}}(\tilde{\mathcal{A}}) + \mathrm{Adv}^{\mathsf{FFPC\text{-}CCA}}_{\mathsf{PKE}^{\mathsf{G}}}(\mathcal{B}) + q_D \cdot 2^{-\gamma} + (q_D \cdot q_{\mathsf{H}}) \cdot 2^{-n} \quad ,$$

Adversary $\tilde{\mathcal{A}}$ makes q_{G} queries to G and $q_{\mathsf{H}} + q_D$ queries to H, *adversary $\mathcal{B}$ makes q_{G} queries to* G *and q_D decryption queries, and both adversaries run in about the time of $\mathcal{A}$.*

Proof. Consider adversary $\mathcal{A}$ against $\mathsf{KEM}^{\perp}$. We now define the IND-CCA adversary $\tilde{\mathcal{A}}$ against $\mathsf{KEM}^{\not\perp}$, which has access to the implicitly rejecting decapsulation oracle $\text{oDecaps}_{\not\perp}$ (Algorithm 15) and simulates the explicitly rejecting decapsulation oracle oDecaps (Algorithm 16) to $\mathcal{A}$ via the simulation $\text{oDecaps}'_{\perp}$ below (Algorithm 18).

Algorithm 15: $\textsc{oDecaps}_{\not\perp}(\mathsf{sk}, c)$

```
1 $m' = \mathsf{Dec}(\mathsf{sk}, c)$
2 if $m' = \perp$ then
3     return $\mathsf{H}(c, \mathsf{sk}.seed)$
4 else
5     $c' = \mathsf{Enc}(\mathsf{pk}, m', \mathsf{G}(m'))$ if $c' \neq c$ then
6         return $\mathsf{H}(c, \mathsf{sk}.seed)$
7     else
8         return $\mathsf{H}(m')$
```

Algorithm 16: $\textsc{oDecaps}_{\perp}(\mathsf{sk}, c)$

```
1 $m' = \mathsf{Dec}(\mathsf{sk}, c)$
2 if $m' = \perp$ then
3     return $\perp$
4 else
5     $c' = \mathsf{Enc}(\mathsf{pk}, m', \mathsf{G}(m'))$ if $c' \neq c$ then
6         return $\perp$
7     else
8         return $\mathsf{H}(m')$
```

$\tilde{\mathcal{A}}$ runs $b' \leftarrow \mathcal{A}^{\mathsf{G}', H, \textsc{oDecaps}'_{\perp}}$ and returns its output b'. Here, G' (see Algorithm 17) is like random oracle G, except that it additionally book-keeps queried plaintexts and their encryptions. Simulation $\textsc{oDecaps}'_{\perp}$ does not decrypt and perform the re-encryption check: instead of decrypting, it will look for suitable plaintexts in the random oracle query list. (Following the notation in [HHM22], we denote this by $m = \mathcal{L}_G^{-1}(c)$.) Instead of performing the re-encryption check, $\textsc{oDecaps}'_{\perp}$ uses the implicitly rejecting decapsulation oracle $\textsc{oDecaps}_{\not\perp}$ (Algorithm 15) to identify invalid ciphertexts.

Algorithm 17: $\mathsf{G}'(m)$

```
1 $r = \mathsf{G}(m)$
2 $c = \mathsf{Enc}(\mathsf{pk}, m; r)$
3 $\mathcal{L}_G = \mathcal{L}_G \cup (m, c)$
4 return $r$
```

Algorithm 18: $\textsc{oDecaps}'_{\perp}(c)$

```
1 $m = \mathcal{L}_G^{-1}(c)$
2 if $m = \perp$ then
3     return $\perp$
4 $K' = \textsc{oDecaps}_{\not\perp}(\mathsf{sk}, c)$
5 if $\mathsf{H}(m) \neq K'$ then
6     return $K = \perp$
7 else
8     return $K = \mathsf{H}(m)$
```

We note that unless $\tilde{\mathcal{A}}$'s simulation $\textsc{oDecaps}'_{\perp}$ fails to emulate $\textsc{oDecaps}$, $\tilde{\mathcal{A}}$ perfectly simulates the game to $\mathcal{A}$ and wins if $\mathcal{A}$ wins. Let DIFF be the event that $\mathcal{A}$ makes a decryption query c for which $\textsc{oDecaps}'_{\perp}$ fails to emulate $\textsc{oDecaps}$, meaning $\textsc{oDecaps}(\mathsf{sk}, c) \neq \textsc{oDecaps}'_{\perp}(c)$. We bound

$$\begin{aligned}
\frac{1}{2} + \mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}^{\perp}} &= \Pr[\mathcal{A} \text{ wins}] \\
&= \Pr[\mathcal{A} \text{ wins} \wedge \neg\mathsf{DIFF}] + \Pr[\mathcal{A} \text{ wins} \wedge \mathsf{DIFF}] \\
&= \Pr[\tilde{\mathcal{A}} \text{ wins} \wedge \neg\mathsf{DIFF}] + \Pr[\mathcal{A} \text{ wins} \wedge \mathsf{DIFF}] \\
&\leq \Pr[\tilde{\mathcal{A}} \text{ wins}] + \Pr[\mathsf{DIFF}] \\
&= \frac{1}{2} + \mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}^{\not\perp}}(\tilde{\mathcal{A}}) + \Pr[\mathsf{DIFF}] \ .
\end{aligned}$$

To analyze the probability of event DIFF, we make a case distinction:

1. Original oracle $\textsc{oDecaps}_{\perp}(\mathsf{sk}, c)$ rejects, and simulation $\textsc{oDecaps}'_{\perp}(c)$ doesn't.
2. Neither oracle rejects, but the return values differ.
3. Simulation $\textsc{oDecaps}'_{\perp}(c)$ rejects, and original oracle $\textsc{oDecaps}_{\perp}(\mathsf{sk}, c)$ doesn't.

To analyze the first case, we notice that if the original oracle $\textsc{oDecaps}_\perp$ rejects, then so does its implicitly rejecting counterpart $\textsc{oDecaps}_{\not\perp}$, by returning $K' = H(c, \mathsf{sk}.\mathit{seed})$. If the simulation $\textsc{oDecaps}'_\perp$ does not reject, that means that $\textsc{oDecaps}'_\perp$ found a message m such that $\mathsf{H}(m) = H(c, \mathsf{sk}.\mathit{seed})$. In other words, the attacker found a collision, which in the random oracle model happens with probability $\frac{1}{2^n}$ per query. After q_D many decapsulation queries, we have set up q_D possible targets. We can search for the solutions for these target by querying H, which gives an upper bound on the complexity: $(q_D \cdot q_\mathsf{H})/2^n$.

The second case is impossible: in the case where $\textsc{oDecaps}_\perp(\mathsf{sk}, c)$ does not reject, neither does $\textsc{oDecaps}_{\not\perp}(\mathsf{sk}, c)$, and both oracles would return $\mathsf{H}(m)$. Since we are in the case where $\textsc{oDecaps}'_\perp$ also does not reject, $\textsc{oDecaps}'_\perp$ returns $K' = \textsc{oDecaps}_{\not\perp}(\mathsf{sk}, c)$, which is the return value of $\textsc{oDecaps}_\perp(\mathsf{sk}, c)$.

In the third case, the simulated oracle $\textsc{oDecaps}'_\perp$ rejects, which can happen in two different ways: a) it can be that no pre-image was found ($\mathcal{L}_G^{-1}(c) = \perp$), or that b) $\textsc{oDecaps}_{\not\perp}(\mathsf{sk}, c) \neq \mathsf{H}(m)$ for the collected pre-image $m = \mathcal{L}_G^{-1}(c)$. To capture sub-case a), we follow [HHM22]. The adversary has to find a ciphertext c that passes the re-encryption check, meaning c is indeed of the form $c = \mathsf{Enc}(\mathsf{pk}, m, \mathsf{G}'(m))$ for some message m, without $\mathcal{A}$ having queried G' on m yet. Intuitively, the attacker guessed the right encryption randomness. Accordingly, [HHM22] denote this case as GUESS and bound it using γ, the spreadness of the PKE scheme: due to [HHM22, Lemma 1], we can bound the probability of this event as

$$\Pr[\mathsf{GUESS}] \leq \mathsf{q_D} \cdot 2^{-\gamma} \ ,$$

where q_D is the number of queries to the decapsulation oracle.

It remains to capture sub-case b), meaning $\textsc{oDecaps}_{\not\perp}(\mathsf{sk}, c) \neq \mathsf{H}(m)$ for the collected pre-image $m = \mathcal{L}_G^{-1}(c)$. Since we are in the case that the original decapsulation oracle does not reject, we know that c was sorted as valid, i.e., that $m' := \mathsf{Dec}(\mathsf{sk}, c) \neq \perp$ and that $\mathsf{Enc}(\mathsf{pk}, m', \mathsf{G}'(m')) = c$. In that case, $K' := \textsc{oDecaps}_{\not\perp}(\mathsf{sk}, c)$ is not a key that resulted from implicit rejection and it thus must be that $K' = \mathsf{H}(m')$, where $m' = \mathsf{Dec}(\mathsf{sk}, c)$. For $\mathsf{H}(m)$ and K' to differ, it must be that $m \neq m'$ and c thus fails to decrypt. In conclusion, the attacker must have posed a query m to G whose encryption $c = \mathsf{Enc}(\mathsf{pk}, m, \mathsf{G}(m))$ fails to decrypt, but nonetheless it holds that $\mathsf{Enc}(\mathsf{pk}, m', \mathsf{G}'(m')) = c$ for $m' = \mathsf{Dec}(\mathsf{sk}, c)$. In other words, the attacker found a solution for the FFPC-CCA game. We bound this via an FFPC-CCA adversary $\mathcal{B}$ against PKE^G: $\mathcal{B}$ runs $\mathcal{A}^{\mathsf{G}', \mathsf{H}, \textsc{oDecaps}''}$, where $\textsc{oDecaps}''$ (Algorithm 19) works exactly as $\textsc{oDecaps}'_\perp$, except that it first checks for plaintexts solving the FFPC-CCA game (line 3). We note that $\mathcal{B}$'s simulation of $\textsc{oDecaps}'_\perp$ is perfect – for the subroutine $\textsc{oDecaps}_{\not\perp}$, it uses a rejection seed which it randomly sampled at the beginning of the game, and $\mathcal{B}$ uses its own decryption oracle oDecrypt (Algorithm 14) for PKE^G to decrypt c. As soon as $\textsc{oDecaps}''$ hits line 4, $\mathcal{B}$ returns the pre-image m, thus winning its game. If $\mathcal{A}$ finishes without line 4 ever being hit, $\mathcal{B}$ returns $\perp$. Since line 4 is hit whenever sub-case b) occurs, we can finish by collecting the probabilities for all three cases:

$$\Pr[\mathsf{DIFF}] \leq (q_D + q_\mathsf{H})/2^n + q_D \cdot 2^{-\gamma} + \mathrm{Adv}^{\mathsf{FFPC\text{-}CCA}}_{\mathsf{PKE}^\mathsf{G}}(\mathcal{B}) \ .$$

□

Algorithm 19: $\textsc{oDecaps}''_{\perp}(c)$

```
1  $m = \mathcal{L}_G^{-1}(c)$
2  $m' = \mathsf{oDecrypt}(c)$
3  if $m \neq \perp \wedge m' \neq \perp \wedge \mathsf{Enc}^{\mathsf{G}}(\mathsf{pk}, m) = \mathsf{Enc}^{\mathsf{G}}(\mathsf{pk}, m')$ then
4      Abort $\mathcal{A}$ and return $m$ to game
5  if $m = \perp$ then
6      return $\perp$
7  if $m' = \perp$ then
8      $K' := \mathsf{H}(c, seed)$
9  else
10     $c' = \mathsf{Enc}(\mathsf{pk}, m', \mathsf{G}(m'))$
11     if $c' \neq c$ then
12         $K' := \mathsf{H}(c, seed)$
13 if $\mathsf{H}(m) \neq K'$ then
14     return $\perp$
15 else
16     return $\mathsf{H}(m)$
```

5 Implicit → explicit: reduction in the eQROM

This section lifts Section 4 into the quantum setting. Before giving the result, we motivate why this cannot be done in the 'normal' quantum-accessible ROM, by summarizing the crucial parts of the previous section. Given an IND-CCA adversary against $\mathsf{KEM}^{\perp} := \mathsf{FO}_m^{\perp}[\mathsf{PKE}]$, we want to break $\mathsf{KEM}^{\not\perp} := \mathsf{FO}_m^{\not\perp}[\mathsf{PKE}]$, hoping to be able to utilize the decapsulation oracle for $\mathsf{KEM}^{\not\perp}$. This means that the reduction needs to identify implicit rejections. In the classical ROM, we utilized that attackers can only generate valid ciphertexts by querying the randomness-generating random oracle G. (This is assuming that the scheme is sufficiently spread and that the proper encryption randomness for $\mathsf{Enc}^{\mathsf{G}}$ hence cannot simply be guessed.) Unfortunately, in the QROM we cannot observe random oracle queries, thus not being able to distinguish implicit rejections from an ordinary response. This is were we utilize the power of the extractable QROM – essentially, the extraction interface Ext allows us to look into the random oracle queries. The extractable QROM was already utilized for KEMs in [DFMS22, HHM22]. With Theorem 7 below, we now show that we can combine this utilization strategy with the template for our new result in Section 4.

We note that while we gained more power through interface Ext when doing the security proof, an attacker would not benefit from access to Ext: Ext mostly gives information about the adversary's own queries; thus, the attacker does not learn much that is new. This intuition is reflected in the subsequent Theorem 8, in which we show that the best currently known QROM bounds for $\mathsf{KEM}^{\perp}$ (recalled as Theorem 3 in Section 3) also apply in the eQROM. (The only obtainable extra information would come from extractions of valid ciphertexts which the attacker created without first querying the randomness-generating oracle G, but this is already being captured via γ-spreadness anyways.)

Theorem 7 (Implicit secure in eQROM → Explicit secure in QROM). *Let PKE be a (randomized) PKE that is γ-spread, and* $\mathsf{KEM}^{\perp} := \mathsf{FO}_m^{\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}]$. *Let $\mathcal{A}$ be an* $\mathsf{IND\text{-}CCA}$ *adversary (in the QROM) against* $\mathsf{KEM}^{\perp}$*, making at most q_D many queries to its decapsulation oracle* $\textsc{oDecaps}$*,* *and making $q_{\mathsf{G}}, q_{\mathsf{H}}$ queries to its respective random oracles. Let furthermore d and w be the combined query depth and query width of A's random oracle queries. Then there exist an* IND-CCA *adversary $\tilde{\mathcal{A}}$ against $\mathsf{KEM}_m^{\not\perp} := \mathrm{FO}_m^{\not\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}]$ and an* FFPC-CCA *adversary $\mathcal{B}$ against $\mathsf{PKE}^{\mathsf{G}}$, where* G *is modelled in the* $\mathrm{eQROM}_{\mathrm{Enc}}$ *in both cases, such that*

$$\mathrm{Adv}_{\mathsf{KEM}^{\perp}}^{\mathsf{IND\text{-}CCA}}(\mathcal{A}) \leq \mathrm{Adv}_{\mathsf{KEM}_m^{\not\perp}}^{\mathsf{IND\text{-}CCA}}(\tilde{\mathcal{A}}) + \mathrm{Adv}_{\mathsf{PKE}^{\mathsf{G}}}^{\mathsf{FFPC\text{-}CCA}}(\mathcal{B}) + 8\sqrt{2}q_D\,(q_{\mathsf{G}} + 3q_D) \cdot 2^{-\gamma/2} + (q_D \cdot q_{\mathsf{H}})/2^n \ .$$

The main proof idea is the same is in the previous classical proof. However, we have to restructure the games to utilize the eQROM, where we use a strategy similar to the one in [HHM22]. Note that we will have to bound FFPC-CCA in the eQROM. As discussed before the classical proof, FFPC-CCA is a sub-problem of FFP-CCA, a straightforward (but less fine-grained) bound on FFPC-CCA can thus be obtained via Theorem 5.

Proof. We prove this theorem via a sequence of games.

Game $\mathbf{G_0}$ is $\mathsf{IND\text{-}CCA}_{\mathsf{KEM}^{\perp}}(\mathcal{A})$.

Game $\mathbf{G_1}$ is like **Game $\mathbf{G_0}$**, except for two modifications: the quantum-accessible random oracle G is replaced by G′ as defined in Algorithm 20 (i.e., it is simulated using an $\mathrm{eQROM}_{\mathrm{Enc}}$). As a preparation for the next game, we additionally introduce after each oDecaps query a query to $\text{oDecaps}_{\not\perp}$ whose results however will not be used anywhere. According to property 1 in [DFMS22, Thm. 4.3] (which we recalled as Lemma 1), G′ perfectly simulates G until the first query to eCO.Ext, and since we have not made any extraction queries yet, we have

$$\mathrm{Adv}_{\mathsf{KEM}_m^{\perp}}^{\mathsf{IND\text{-}CCA}}(\mathcal{A}) = \mathrm{Adv}^{\textbf{Game } \mathbf{G_0}} = \mathrm{Adv}^{\textbf{Game } \mathbf{G_1}}.$$

Game $\mathbf{G_2}$ is like **Game $\mathbf{G_1}$**, except after the adversary has finished, we compute the oracle pre-images for all ciphertexts on which oDecaps was queried. I.e., we compute $\hat{m}_i = \mathsf{eCO.Ext}(c_i)$ for all $i = 1, \dots, q_D$, where c_i is the input to the adversary's i-th decapsulation query. Again, G′ perfectly simulates G until the first query to eCO.Ext, and the first such query occurs only after $\mathcal{A}$ finished, we have

$$\mathrm{Adv}^{\textbf{Game } \mathbf{G_2}} = \mathrm{Adv}^{\textbf{Game } \mathbf{G_1}}.$$

Algorithm 20: G', input registers X, Y

```
1 Apply $\mathsf{eCO.RO}_{XYD}$
2 return registers $XY$
```

Game $\mathbf{G_3}$ is like **Game $\mathbf{G_2}$**, except that $\hat{m}_i = \mathsf{eCO.Ext}(c_i)$ is computed right after the query to implicitly decapsulation oracle $\text{oDecaps}_{\not\perp}$. Note that **Game $\mathbf{G_3}$** can be obtained from **Game $\mathbf{G_2}$** by first swapping the call to eCO.Ext that produces $\hat{m}_1$ with all eCO.RO and eCO.Ext calls that happen after the decapsulation query for c_1 (including all RO calls performed by both decapsulation oracles), then continuing with $\hat{m}_2$ and so on. We will now use that eCO.RO and eCO.Ext $8\sqrt{2\Gamma(f)/|\mathcal{R}|}$-almost-commute (see Lemma 1). Since $\Gamma(\mathsf{Enc}(\cdot,\cdot)) = 2^{-\gamma} \cdot |\mathcal{R}|$ for a γ-spread PKE scheme, we have

$$|\mathrm{Adv}^{\textbf{Game } \mathbf{G_2}} - \mathrm{Adv}^{\textbf{Game } \mathbf{G_3}}| \leq 8\sqrt{2}q_D(q_G + 2 \cdot q_D) \cdot 2^{\gamma/2}.$$

Now that we are equipped with a method of finding pre-images, we can deploy a strategy that is similar to the one in the previous classical proof.

Game $\mathbf{G_4}$ is the same as **Game $\mathbf{G_3}$**, except that $\mathcal{A}$ is run with access to the simulated oracle $\text{oDecaps}'_\bot$ (Algorithm 21) instead of the original oracle oDecaps. Like in the classical proof, the simulation uses $\text{oDecaps}_{\not\bot}$ (Algorithm 22) as a subroutine. We also note that the simulation $\text{oDecaps}'_\bot$ issues the same number of G' queries as oDecaps does. Note that all the queries needed for simulations in $\text{oDecaps}'_\bot$ are already done in **Game $\mathbf{G_3}$** but the adversary was not given the access to the results of that queries.

Algorithm 21: $\text{oDecaps}'_\bot(c)$

```
1 $K' = \text{oDecaps}_{\not\bot}(\mathsf{sk}, c)$
2 $m = \mathsf{eCO.Ext}(c)$
3 if $m = \bot$ then
4     return $K = \bot$
5 if $H(m) \neq K'$ then
6     return $K = \bot$
7 else
8     return $K = H(m)$
```

Algorithm 22: $\text{oDecaps}_{\not\bot}(\mathsf{sk}, c)$

```
1 $m' = \mathsf{Dec}(\mathsf{sk}, c)$
2 if $m' = \bot$ then
3     return $H(c, \mathsf{sk}.seed)$
4 else
5     $c' = \mathsf{Enc}(\mathsf{pk}, m', G(m'))$ if $c' \neq c$ then
6         return $H(c, \mathsf{sk}.seed)$
7     else
8         return $H(m)$
```

Like in the proof of the classical counterpart, Theorem 6, we again analyze when the simulation differs from the original oracle: let DIFF be the event that $\mathcal{A}$ makes a decryption query c in **Game $\mathbf{G_3}$** such that $\text{oDecaps}(c) \neq \text{oDecaps}'(c)$. With the same probability analysis as before,

$$\frac{1}{2} + \text{Adv}^{\textbf{Game } \mathbf{G_3}}(\mathcal{A}) \leq \frac{1}{2} + \text{Adv}^{\textbf{Game } \mathbf{G_4}}(\mathcal{A}) + \Pr[\mathsf{DIFF}] \ .$$

To analyze the probability of event DIFF, we make the same case distinction:

1. Original oracle $\text{oDecaps}_\bot(\mathsf{sk}, c)$ rejects, and simulation $\text{oDecaps}'_\bot(c)$ doesn't.
2. Neither oracle rejects, but the return values differ. (This case is impossible, as argued in the classical proof.)
3. Simulation $\text{oDecaps}'_\bot(c)$ rejects, and original oracle $\text{oDecaps}_\bot(\mathsf{sk}, c)$ doesn't.

To analyze case 1, we perform exactly the same reasoning as before, only that we now have to consider quantum access to the random oracles: if the original oracle $\text{oDecaps}_\bot$ rejects, then so does its implicitly rejecting counterpart $\text{oDecaps}_{\not\bot}$, by returning $K' = H(c, \mathsf{sk}.seed)$. If the simulation $\text{oDecaps}'_\bot$ does not reject, that means that $\text{oDecaps}'_\bot$ found a message m such that $\mathsf{H}(m) = H(c, \mathsf{sk}.seed)$. In other words, the attacker found a collision, which in the quantum-accessible random oracle model happens with probability $\frac{1}{2^n}$. After q_D many decapsulation queries, we have set up q_D possible targets. We can search for the solutions for these target by querying H, which for the quantum case gives a quadratic speed-up and results in an upper bound: $(q_D \cdot q_{\mathsf{H}}^2)/2^n$.

For case 2 the same argument as in the classical proof of Theorem 6 is applicable. Returning different values that are not rejects is impossible.

In case 3, the simulation $\text{oDecaps}'_\bot$ rejects. Again, this can happen in two different ways: a) it can be that $m = \mathsf{eCO.Ext}(c) = \bot$ (no pre-image was found), or that b) $\text{oDecaps}_{\not\bot}(\mathsf{sk}, c) \neq \mathsf{H}(m)$ for the collected pre-image $m = \mathsf{eCO.Ext}(c)$. Sub-case a) can only happen if $\mathsf{Enc}(\mathsf{pk}, m', G(m')) \neq c$ for the decrypted plaintext $m' := \mathsf{Dec}(sk, c)$ – otherwise, the database will have at least one record that will satisfy the extraction query because the subroutine $\text{oDecaps}_{\not\bot}$ performed the re-encryption check and thus wrote m' into the oracle database. But if $\mathsf{Enc}(\mathsf{pk}, m', G(m')) \neq c$, we have a contradiction with the prerequisites of case 3 since the original oracle $\text{oDecaps}(c)$ will also reject (due to the failing re-encryption check). Hence, this sub-case is impossible.

It remains to capture sub-case b). As reasoned in the classical proof, this means that a pre-image m was collected for which $c = \mathsf{Enc}(\mathsf{pk}, m; \mathsf{G}(m))$ fails to decrypt (meaning $m \neq m'$ for $m' := \mathsf{Dec}(sk, c)$), but nonetheless it holds that $\mathsf{Enc}(\mathsf{pk}, m', \mathsf{G}'(m')) = c$. Again, this is a solution for the FFPC-CCA game, with the only difference to the classical proof being how the pre-image was collected. So we can again bound this case using FFPC-CCA– we only need to adapt the way in which the classical reduction $\mathcal{B}$ collects its pre-images (line 1 in Algorithm 19). Concretely, this means replacing $m = \mathcal{L}_G^{-1}(c)$ by $m = \mathsf{eCO.Ext}(c)$. We note that the reduction plays against $\mathsf{PKE}^{\mathsf{G}}$, thus it could not simply redefine G to be extractable itself. By considering game FFPC-CCA in the extractable QROM, we ensure that the reduction has the necessary access to interface Ext. Collecting the probabilities yields the desired bound.

Game $\mathbf{G_5}$ is like **Game $\mathbf{G_4}$**, except that we move the oDecaps queries that we are not using anymore to the very end (after the $\mathcal{A}$ finishes). The reasoning here is the same as in the hop from $\mathbf{G_2}$ to $\mathbf{G_3}$. As a result we have the following bound

$$|\mathrm{Adv}^{\textbf{Game } \mathbf{G_4}} - \mathrm{Adv}^{\textbf{Game } \mathbf{G_5}}| \leq 8\sqrt{2}q_D^2 \cdot 2^{\gamma/2}.$$

Game $\mathbf{G_6}$ is the same as **Game $\mathbf{G_5}$** but we drop the oDecaps queries. Since they are made after the oracle $\mathcal{A}$ has already finished the work they do not affect the probability of success of the adversary. Hence

$$\mathrm{Adv}^{\textbf{Game } \mathbf{G_6}} = \mathrm{Adv}^{\textbf{Game } \mathbf{G_5}}.$$

The advantage in Game 6 is bounded by the IND-CCA security of the $\mathsf{KEM}_m^{\not\perp}$. Combining all the bounds we get the claimed result. □

Impact of adversarial access to eCO.Ext on security of FO-KEMs. As argued at the beginning of this section, attackers would not substantially benefit from access to Ext. We now make this intuition formal with Theorem 8 below, stating that the best currently known QROM bounds for $\mathsf{KEM}^{\perp}$ ([HHM22, Theorem 4], here recalled as Theorem 3) also apply in the eQROM: the theorem's proof still goes through even if the IND-CCA attacker has additional access to eCO.Ext.

Theorem 8 (IND-CCA security in the eQROM). *Let PKE be a (randomized) PKE that is γ-spread and δ-correct, and let* $\mathsf{KEM}^{\perp} := \mathsf{FO}_m^{\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}]$. *Let $\mathcal{A}$ be an* IND-CCA *adversary (in the eQROM) against* $\mathsf{KEM}^{\perp}$, *making at most q_D many queries to its decapsulation oracle* oDecaps, *and making $q_{\mathsf{G}}, q_{\mathsf{H}}$ queries to its respective random oracles and q_E queries to the extraction interface of oracle* G. *Let furthermore d and w be the combined query depth and query width of A's random oracle queries. Then there exist an* IND-CPA-*KEM adversary $\tilde{\mathcal{A}}$ in the* $\mathrm{eQROM}_{\mathrm{Enc}}$, *such that*

$$\mathrm{Adv}_{\mathsf{KEM}^{\perp}}^{\mathsf{IND\text{-}CCA}}(\mathcal{A}) \leq \mathrm{Adv}_{\mathsf{KEM}^{\perp}}^{\mathsf{IND\text{-}CPA}}(\tilde{\mathcal{A}}) + 10\,(q_G + q_D + 1)^2\,\delta + 12q_D\,(q_{\mathsf{G}} + q_E + 4q_D) \cdot 2^{-\gamma/2}\ .$$

This bound also applies for the implicitly rejecting counterpart $\mathsf{KEM}^{\not\perp} := \mathsf{FO}_m^{\not\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}]$ *(recall [HHM22, Remark 1]).*

In other words, when allowing the attacker to perform extractions, security degrades by the term $12q_Dq_E \cdot 2^{-\gamma/2}$. Since γ already needs to be reasonably large for plain IND-CCA security, we expect that this additional term will be reasonably small for practical instantiations.

Proof. Our proof slightly adjusts the proof of [HHM22, Theorem 4] (here recalled as Theorem 3). The main technique of the proof in Theorem 3 is to show that the decapsulation oracle can be simulated to $\mathcal{A}$ without using the secret key, by suitably using the extraction oracle eCO.Ext. The only technical difference between the setting of Theorem 3 and Theorem 8 is the following: in Theorem 3, only the game has (internal) access to eCO.Ext, whereas in Theorem 8, the game also provides this interface to

$\mathcal{A}$ by forwarding $\mathcal{A}$'s extraction queries. We thus only need to show that $\mathcal{A}$'s queries do not interfere with the simulation $\textsc{oDecaps}'$ of $\textsc{oDecaps}$ given in the proof of Theorem 3. We thus first recall the simulation $\textsc{oDecaps}'$: on a given input ciphertext c, $\textsc{oDecaps}'$ queries the extraction oracle on c to obtain a plaintext m and then returns $H(m)$. (If the extractor returned $\perp$, then so does $\textsc{oDecaps}'$.) In the proof, the switch from $\textsc{oDecaps}$ to $\textsc{oDecaps}'$ is done through several steps:

1. Introduce extraction queries, but only after the fact: after $\mathcal{A}$ finished, extractions are performed for all ciphertexts on which $\textsc{oDecaps}$ was queried. (This does not change the behaviour of $\mathcal{A}$.)
2. Move these extraction queries from the very end to the place where they would help define $\textsc{oDecaps}'$, so perform them immediately when $\mathcal{A}$ issues the respective decapsulation query. The additional measurements performed during extraction change the random oracle database and thus introduce a disruption term relative to PKE's γ-spreadness – using the almost-commuting property (see item 4 in Lemma 1), the disruption is bound by $8\sqrt{2}q_D(q_G + q_D) \cdot 2^{-\gamma/2}$.
3. Use $\textsc{oDecaps}'$ instead of $\textsc{oDecaps}$. However, still perform a muted internal call to $\textsc{oDecaps}$ during $\textsc{oDecaps}'$ (for technical reasons). The switch from outputting $\textsc{oDecaps}(c)$ to $\textsc{oDecaps}'(c)$ introduces a correctness-related term, $\mathrm{Adv}^{\mathsf{FFP\text{-}CCA}}_{\mathsf{PKE}^{\mathsf{G}}}(\mathcal{B})$, which is upper-bounded by $10\left(q_G + q_D + 1\right)^2 \delta$ according to Theorem 5.
4. Delay these internal calls to $\textsc{oDecaps}$ until after $\mathcal{A}$ finished. Technically, this means swapping the $\mathsf{eCO.RO}$ calls within $\textsc{oDecaps}$ with all extraction queries performed by the game. This introduce the second disruption term relative to PKE's γ-spreadness – using the almost-commuting property again, the disruption is bound by $8\sqrt{2}q_D^2 \cdot 2^{-\gamma/2}$.
5. After that, omitting the $\textsc{oDecaps}$ calls entirely does not change the behaviour of $\mathcal{A}$, the game thus effectively switched from $\textsc{oDecaps}$ to $\textsc{oDecaps}'$. Lastly, the two disruption terms are merged/simplified to $12q_D\left(q_{\mathsf{G}} + 4q_D\right) \cdot 2^{-\gamma/2}$.

The only steps affected by adversarial extractions are steps 2 and 4. It thus remains to show that adversarial extraction queries 'play nicely' with these two steps.

Considering step 2, we note that the disruptions stem from the adversarial calls to the random oracle, so every adversarial call to $\mathsf{eCO.RO}$, but since these perfectly commute with the adversarial extractions (see Lemma 1), the bound for step 2 is unchanged.

In step 4, we note that the same overall reasoning still holds – we still swap the $\mathsf{eCO.RO}$ calls within $\textsc{oDecaps}$ with all extraction queries performed by the game, only that these extraction queries now also contain the additional q_E many ones that are issued by $\mathcal{A}$. Consequently, we bound the commuting error for this second step by $8\sqrt{2}q_D(q_D + q_E) \cdot 2^{\gamma/2}$.

Lastly, we merge/simplify the two terms to $12q_D\left(q_{\mathsf{G}} + q_E + 4q_D\right) \cdot 2^{-\gamma/2}$.

6 Implicit → explicit: indirect proof in the QROM (without extractions)

In this section, we re-investigate the security relation in the simpler QROM, that is, with a proof in the compressed oracle without the additional extraction interface. To do so, we will unbox the FO transformation: according to Corollary 1 in Section 3, we can (non-tightly) base INDCCA security of $\mathsf{KEM}^{\perp} := \mathsf{FO}^{\perp}_{m}[\mathsf{PKE}]$ on $\mathsf{OW\text{-}CPA}$ security of the underlying PKE, in the standard model. Concretely, the theorem states that any INDCCA attacker $\mathcal{A}$ on $\mathsf{KEM}^{\perp}$ can be turned into an $\mathsf{OW\text{-}CPA}$ attacker $\mathcal{B}$ on PKE. Using that adversary $\mathcal{B}$, we can build an INDCCA adversary $\mathcal{C}$ against the implicitly rejecting $\mathsf{KEM}^{\not\perp} := \mathsf{FO}^{\not\perp}_{m}[\mathsf{PKE}]$. Hence, we obtain the following (non-tight) theorem.

Theorem 9 (Implicit → Explicit). *Let PKE be a (randomized) PKE that is γ-spread and δ-correct, and let $\mathsf{KEM}^{\perp} := \mathsf{FO}^{\perp}_{m}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}]$. Let $\mathcal{A}$ be an $\mathsf{IND\text{-}CCA}$ adversary (in the QROM) against $\mathsf{KEM}^{\perp}$, making at most q_D many queries to its decapsulation oracle $\textsc{oDecaps}$, and making $q_{\mathsf{G}}, q_{\mathsf{H}}$ queries to its respective random oracles. Let furthermore d and w be the combined query depth* *and query width of $\mathcal{A}$'s random oracle queries. Then there exists an* $\mathsf{IND\text{-}CCA}$ *adversary* $\mathcal{C}$ *against* $\mathsf{KEM}^{\not\perp} := \mathsf{FO}_m^{\not\perp}[\mathsf{PKE}, \mathsf{G}', \mathsf{H}']$ *such that*

$$\mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}^{\perp}}(\mathcal{A}) \leq 8(d+q_D)\cdot\sqrt{w\cdot 4\cdot \mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}^{\not\perp}}(\mathcal{C})} + 10\left(q_G+q_D+1\right)^2\delta + 12q_D\left(q_{\mathsf{G}}+4q_D\right)\cdot 2^{-\gamma/2}$$

The running time and quantum memory footprint of $\mathcal{C}$ *satisfy* $\mathrm{Time}(\mathcal{C}) = \mathrm{Time}(\mathcal{A}) + O(q_D) + \mathrm{Time}(\mathsf{eCO}, q, q_E) + O(1)$ *and* $\mathrm{QMem}(\mathcal{A}) + \mathrm{QMem}(\mathsf{eCO}, q, q_E)$ *for* $q = q_{\mathsf{G}} + q_{\mathsf{H}} + q_D$ *and* $q_E = q_D$*, and where* $O(1)$ *hides a single query to* H *and the comparison of the hash value with the challenge* K_b^**.*

Proof. According to Corollary 1, there exists an $\mathsf{OW\text{-}CPA}$ adversary $\mathcal{B}$ against the underlying PKE, running $\mathcal{A}$ as a subroutine for which it simulates the random oracles and the decapsulation oracle, such that

$$\mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}^{\perp}}(\mathcal{A}) \leq\ 8(d+q_D)\cdot\sqrt{w\cdot \mathrm{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B})} + 10\left(q_G+q_D+1\right)^2\delta + 12q_D\left(q_{\mathsf{G}}+4q_D\right)\cdot 2^{-\gamma/2}\ ,$$

with footprints $\mathrm{Time}(\mathcal{B}) = \mathrm{Time}(\mathcal{A}) + O(q_D) + \mathrm{Time}(\mathsf{eCO}, q, q_E)$ and $\mathrm{QMem}(\mathcal{B}) = \mathrm{QMem}(\mathcal{A}) + \mathrm{QMem}(\mathsf{eCO}, q, q_E)$ for $q = q_{\mathsf{G}} + q_{\mathsf{H}} + q_D$ and $q_E = q_D$.

We will now argue that we can turn $\mathsf{OW\text{-}CPA}$ adversary $\mathcal{B}$ into an $\mathsf{IND\text{-}CCA}$ adversary $\mathcal{C}$ against $\mathsf{KEM}^{\not\perp}$: note that adversary $\mathcal{B}$ is playing against the underlying (non-derandomized) PKE scheme that does not involve random oracle G, and expects a challenge ciphertext that uses uniform randomness. While the $\mathsf{IND\text{-}CCA}$ game for $\mathsf{KEM}^{\not\perp}$ creates its challenge ciphertext for $\mathcal{C}$ by computing $c^* = \mathsf{Enc}(\mathsf{pk}, m^*; \mathsf{G}(m^*))$, using randomness generated via random oracle G, adversary $\mathcal{B}$ has no access to the independent oracle G and thus could not distinguish the KEM challenge ciphertext $c^* = \mathsf{Enc}(\mathsf{pk}, m^*; \mathsf{G}(m^*))$ from the ciphertext it expects.

Hence, given a challenge (c^*, K_b^*) for $\mathsf{KEM}^{\not\perp}$, $\mathcal{C}$ can run $\mathcal{B}$ on c^* to get m^*. Note that $\mathcal{B}$ does not have access to the random oracles of $\mathsf{KEM}^{\not\perp}$ and does its own simulations of the respective oracles for $\mathcal{A}$. The simulation can be done with different techniques, for example q-wise independent functions [Zha12]. The important aspect is that the adversary $\mathcal{B}$ does not need the access to the random oracles of the $\mathsf{IND\text{-}CCA}$ game for $\mathsf{KEM}^{\not\perp}$. After obtaining $\mathcal{B}$'s one-way solution m, $\mathcal{C}$ queries the key derivation oracle H' to obtain the corresponding key $K = \mathsf{H}'(m)$. Assuming $\mathcal{B}$'s guess was right, then $\mathcal{C}$ can determine the $\mathsf{IND\text{-}CCA}$ bit: if $K = K_b^*$, then $b = 0$, otherwise $b = 1$. Lets break it down. In case $\mathcal{C}$ get a challenge (c^*, K_1^*) we run $\mathcal{B}$ on c^* and get m^* with probability equal to $\mathrm{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B})$. We compute $H'(m^*) = K$. Since K_1^* is random, the probability $K = K_1^*$ is $1/2^n$ otherwise we get a different value. In case $\mathcal{C}$ get a challenge (c^*, K_0^*) there will be no mistake if $\mathcal{B}$ succeeds in finding the correct plaintext. If $\mathcal{B}$ fails then $\mathcal{C}$ always returns 1. As a result we have

$$\begin{aligned}\mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}^{\not\perp}}(\mathcal{C}) &= \Pr[b=0]\cdot\Pr[\mathcal{C}\to 0|b=0] + \Pr[b=1]\cdot\Pr[\mathcal{C}\to 1|b=1] - \frac{1}{2} = \\ &= \mathrm{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B})\cdot\frac{1}{2} + (\mathrm{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B})\,(1-1/2^n) + (1-\mathrm{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B})))\cdot\frac{1}{2} - \frac{1}{2} = \\ &= \mathrm{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B})\,(1-1/2^n)\cdot\frac{1}{2}\end{aligned}$$

Hence,

$$\mathrm{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B}) \leq \frac{2^{n+1}}{2^n-1}\mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}^{\not\perp}}(\mathcal{C}) \leq 4\cdot\mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}^{\not\perp}}(\mathcal{C})\ .$$

with footprints $\mathrm{Time}(\mathcal{C}) = \mathrm{Time}(\mathcal{B}) + O(1)$ and $\mathrm{QMem}(\mathcal{C}) = \mathrm{QMem}(\mathcal{B}) + O(1)$, where $O(1)$ hides a single query to H and the comparison of the hash value with the challenge K_b^*. This concludes the proof. □

References

BHH$^+$19. Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 61–90, Nuremberg, Germany, December 1–5, 2019. Springer, Cham, Switzerland.

Den03. Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151, Cirencester, UK, December 16–18, 2003. Springer, Berlin, Heidelberg, Germany.

DFMS22. Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Online-extractability in the quantum random-oracle model. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 677–706, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

FO99. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Berlin, Heidelberg, Germany.

FO13. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.

HHK17. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.

HHM22. Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Failing gracefully: Decryption failures and the Fujisaki-Okamoto transform. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 414–443, Taipei, Taiwan, December 5–9, 2022. Springer, Cham, Switzerland.

HKSU20. Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. Generic authenticated key exchange in the quantum random oracle model. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 389–422, Edinburgh, UK, May 4–7, 2020. Springer, Cham, Switzerland.

HM24. Kathrin Hövelmanns and Christian Majenz. A note on failing gracefully: Completing the picture for explicitly rejecting fujisaki-okamoto transforms using worst-case correctness. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 245–265, Oxford, UK, June 12–14, 2024. Springer, Cham, Switzerland.

JZC$^+$18. Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 96–125, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

JZM19. Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. Key encapsulation mechanism with explicit rejection in the quantum random oracle model. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 618–645, Beijing, China, April 14–17, 2019. Springer, Cham, Switzerland.

KSS$^+$20. Veronika Kuchta, Amin Sakzad, Damien Stehlé, Ron Steinfeld, and Shifeng Sun. Measure-rewind-measure: Tighter quantum random oracle model proofs for one-way to hiding and CCA security. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 703–728, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.

NIS. NIST. Post-quantum cryptography. `https://csrc.nist.gov/projects/post-quantum-cryptography`. Accessed: 2024-10-17.

SXY18. Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 520–551, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

Zha12. Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 758–775, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Heidelberg, Germany.

Zha19. Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 239–268, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.