IACR Communications in Cryptology
ISSN 3006-5496, Vol. 3, No. 1, 44 pages.

https://doi.org/10.62056/a63zl83y6

Check for updates

On The Multi-target Security of Post-Quantum Key Encapsulation Mechanisms

Lewis Glabush[1], Kathrin Hövelmanns[a,2] and Douglas Stebila[b,3]

[1] EPFL, Lausanne, Switzerland
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
[3] University of Waterloo, Waterloo, Canada

Abstract. Practical deployments of key encapsulation mechanisms (KEMs) may entail large servers each using their public keys to communicate with potentially millions of clients simultaneously. While the standard IND-CCA security definition for KEMs considers only a single challenge public key and single challenge ciphertext, it can be relevant to consider *multi-target* scenarios where the adversary aims to break one of many challenge ciphertexts, for one of many challenge public keys. Many post-quantum KEMs have been built by applying the Fujisaki–Okamoto (FO) transform to a public key encryption (PKE) scheme. Although the FO transform incurs only a few bits of security loss for the standard, single-challenge IND-CCA property, this does not hold in the multi-target setting. Attacks have been identified against standards-track FO-based KEMs with 128-bit message spaces (FrodoKEM-640 and HQC-128) which become feasible if the adversary is given many challenge ciphertexts (say, 2^{64}). These attacks exploit the deterministic encryption induced by the FO transform which allows the IND-CCA experiment to be reduced to a search problem on the message space, which in some cases may not be large enough to avoid collisions between pre-computation and challenge values. A cost effective way to amplify the hardness of this search problem is to add a random but public salt during encapsulation. While revised versions of FrodoKEM and HQC have used salts, there has been no proof showing that salting provides multi-ciphertext security. In this work, we formally analyze a salted variant of the Fujisaki–Okamoto transform, in the classical and quantum random oracle model (ROM); for the classical ROM, we show that multi-target IND-CCA security of the resulting KEM tightly reduces to the multi-target IND-CPA security of the underlying PKE. Our results imply that, for FrodoKEM and HQC at the 128-bit security level, replacing the FO transform with the salted variant can recover 62 bits of multi-target security, at the cost of a very small overhead increase.

Keywords: Fujisaki–Okamoto transform · key exchange · quantum random oracle model (QROM) · post-quantum cryptography · multi-target security · FrodoKEM · HQC · ML-KEM

1 Introduction

The Fujisaki–Okamoto (FO) transform [FO99, FO13] converts a weakly secure public key encryption scheme into an IND-CCA-secure public key encryption scheme. In the context of post-quantum cryptography, its adaptations for key encapsulation mechanisms (KEMs) given in [Den03, HHK17] received renewed attention and by now have become the de-facto

E-mail: lewis.glabush@epfl.ch (Lewis Glabush), kathrin@hoevelmanns.net (Kathrin Hövelmanns), dstebila@uwaterloo.ca (Douglas Stebila)

[a]K.H. was supported by NWO VENI grant (Project No. VI.Veni.222.397).
[b]D.S. was supported by NSERC grants RGPIN-2022-03187 and ALLRP 578463-22.

Received: 2026-02-03 Accepted: 2026-04-22

standard for designing KEMs. Notably, all KEM submissions to the NIST Post-Quantum Cryptography standardization process which made it to later rounds used some variant of FO. Given that communications security protocols like TLS need to perform key exchanges, and that the best-studied post-quantum replacements so far are KEMs, it can be envisioned that the future security of such protocols will be based (among others) on some variant of FO.

IND-CCA vs. multi-target security. The required security goal for KEMs during the NIST PQC process was IND-CCA security in the presence of quantum attackers. Within the last few years, the community made huge progress [HHK17, BHH+19, SXY18, JZC+18, HKSU20, JZM19, DFMS22, HHM22, HM24] in analyzing whether the FO-transform meets this goal by developing more sophisticated formalisms to capture quantum attackers. It can be argued, however, that IND-CCA security alone is not enough in practice, when attackers can observe client-server interactions over a long period of time and then exploit the large collection of public keys and ciphertexts that amounted during these interactions. It would hence be desirable to bound the security of the exchanged keys even if adversaries can observe and attack many public keys and/or ciphertexts: this is *multi-target security*. Of particular concern in the multi-target setting are generic collision finding attacks, such as those identified against the NIST PQC candidates FrodoKEM and HQC demonstrate that IND-CCA security may sufficiently degrade in the multi-target setting to enable real security threats. The attacks make use of the fact that the KEMs are built by applying FO to a public key encryption (PKE), leading to a ciphertext space that is the same size as the message space. While this attack formally falls out of the scope of IND-CCA security, it nonetheless exposes an important vulnerability, which motivates the study into techniques for establishing multi-target security in KEMs.

The Fujisaki–Okamoto transform. A modern way of understanding the FO-transform is the modular approach of [HHK17], in which the FO transformation for KEMs was dissected into two separate steps: a pre-transformation T converting a probabilistic PKE scheme into a deterministic one; and a transform U converting a PKE to a KEM. The combination is denoted $\mathsf{FO} := \mathsf{U} \circ \mathsf{T}$.

- $\mathsf{T} : \mathsf{PKE} \to \mathsf{DPKE}$: this transform modifies the probabilistic encryption algorithm so that, rather than computing $c \leftarrow \mathsf{Enc}(pk, m; r)$ for uniformly encryption randomness r, instead it computes

$$c \leftarrow \mathsf{Enc}_1(pk, m) \leftarrow \mathsf{Enc}(pk, m; \mathsf{G}(m)),$$

  where G is a hash function, modeled as a random oracle during security proofs. Decryption is also modified by introducing a *re-encryption check*: $\mathsf{Dec}_1(sk, c)$ still first computes $m' \leftarrow \mathsf{Dec}(sk, c)$, but after that, it checks whether $\mathsf{Enc}(pk, m'; \mathsf{G}(m')) = c$ and only returns m' if it does. (Otherwise, Dec_1 rejects.)

- $\mathsf{U} : \mathsf{PKE} \to \mathsf{KEM}$: this transform builds an IND-CCA-secure KEM from a PKE by letting

$$(c \leftarrow \mathsf{Enc}(pk, m) \leftarrow \mathsf{Encaps}(pk), \mathbf{ss} \leftarrow \mathsf{H}(m, c))$$

  for a randomly chosen message m. Decaps likewise, will return $\mathsf{H}(m, c)$ unless c fails to decrypt. Two variants of U are given in [HHK17], called $\mathsf{U}^{\perp}$ and $\mathsf{U}^{\not\perp}$; with superscripts $\perp$ and $\not\perp$ describing how invalid ciphertexts are handled: if c fails to decrypt, $\mathsf{U}^{\perp}$ will return a dedicated error symbol $\perp$, whereas $\mathsf{U}^{\not\perp}$ will return a pseudorandom value of the same length as an honestly generated key. These variants can be further subdivided based on which values are included as hash inputs for **ss**: $\mathsf{U}^{\perp}_{m}$ and $\mathsf{U}^{\not\perp}_{m}$ only hash m instead of m, c. Follow-up work [BHH+19] proved that

security is unaffected by the choice between the two hash input options, assuming the re-encryption check is included (so when the step is explicitly added to U or when U simply is combined with T).

Multi-user and multi-ciphertext security. Multi-target security refers to attack models parameterized by the number of users n_{u}, and the number of challenge ciphertexts n_{c}. A multi-ciphertext (sometimes also called multi-challenge) attack refers to the case with $n_{\mathsf{u}} = 1$ and $n_{\mathsf{c}} \geq 1$, with n_{c} total challenge ciphertexts. Likewise, multi-user (sometimes called multi-key) security refers to the scenario with $n_{\mathsf{u}} \geq 1$ and $n_{\mathsf{c}} = 1$, with n_{u} total challenge ciphertexts. A multi-target attack may have both $n_{\mathsf{u}} \geq 1$ and $n_{\mathsf{c}} \geq 1$, with $n_{\mathsf{u}} n_{\mathsf{c}}$ total challenge ciphertexts.

Multi-ciphertext attack on KEMs with "small" message space. We now revisit a generic attack on any $\mathsf{KEM} := \mathsf{FO}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}]$, built from a PKE scheme with comparably small message space. For the sake of giving an example, we pick size 2^{128}, which is the case for both FrodoKEM-640 and HQC-128, proposed in [NAB+20, AAB+22]. This multi-ciphertext attack was identified against FrodoKEM-640 privately by NIST [NIS21] and publicly by Bernstein [Ber22], which led to an update of FrodoKEM [ABD+23a]. The same type of attack was identified by NIST against HQC [AAB+22]. Suppose an attacker $\mathcal{A}$ collected n_{c} many challenge ciphertexts $c_1, \ldots, c_{n_{\mathsf{c}}}$ belonging to a single user, and their attack goal is only to distinguish just one out of the n_{c} many corresponding keys from random. (Here, the choice between real and random is universal: for all challenges, the attacker either always sees a real key or always sees an independent random value.) For message space size 2^{128} and a large but not infeasible number of challenge ciphertexts (for example 2^{64}), success is very likely. The adversary $\mathcal{A}$, given n_{c} challenge ciphertexts $c_1, \ldots, c_{n_{\mathsf{c}}}$, also prepares N ciphertexts of their own, by sampling random messages and encrypting them deterministically. If there is any intersection between the set of challenge ciphertexts $c_1, \ldots, c_{n_{\mathsf{c}}}$ and $\mathcal{A}$'s own precomputed set $c'_1, \ldots, c'_N$, $\mathcal{A}$ can easily win the game: since the intersection must stem from having picked a message m that was also used by one of the observed c_i, $\mathcal{A}$ can compute the corresponding key $\mathbf{ss} = \mathsf{H}(m, c_i)$ and thus immediately can tell this challenge apart from random. To estimate the probability with which this kind of collision will happen, we note that for each challenge ciphertext c_i, there is approximately an $N/2^{128}$ chance that c_i used the same message as one of $\mathcal{A}$'s ciphertexts $c'_1, \ldots, c'_N$. Over the n_{c} many challenge ciphertexts, the probability of $\mathcal{A}$ sampling a collision is thus

$$\Pr[\mathsf{COLL}] \approx \frac{N n_{\mathsf{c}}}{2^{128}}.$$

If $\mathcal{A}$ is given, say, $n_{\mathsf{c}} = 2^{64}$ many challenges, then, by preparing $N = 2^{64}$ ciphertexts of their own, $\mathcal{A}$ will find a collision with constant probability and thus break multi-ciphertext security. With a smaller bound (think 2^{32}) on the number of challenges for a single user, we observe linear security degradation, essentially dropping bits of multi-ciphertext security proportional to n_{c}.

Mitigation of multi-ciphertext attacks via salted FO. As seen above, for small message spaces, FO-based KEMs will be susceptible to multi-ciphertext attacks. But depending on the concrete PKE scheme at hand, increasing the message space size might render it prohibitively inefficient. For example, increasing the message space in FrodoKEM requires increasing the LWE matrix size, substantially impacting communication sizes. The FrodoKEM update [ABD+23a] thus modified their FO transform to include a uniformly random, public salt, that aimed to mitigate collisions based on too-small message spaces by increasing the search space via salting. Instead of only hashing the message and ciphertext, this modified transform additionally hashes a uniformly random salt of length $\mathsf{len}_{\mathsf{salt}}$ (which is then communicated along with the ciphertext). While the FrodoKEM teams updated

specification provided a proof that this tweak does not degrade IND-CCA security, it did not give a proof that the salted version actually improves *multi-target* security.

Our contributions. Our contribution is a study of transforms to achieve multi-target security for post-quantum KEMs. We formalize the salted Fujisaki–Okamoto (SFO) transform and prove that it yields KEMs for which multi-target IND-CCA security follows from multi-target IND-CPA security of the underlying PKE. Concretely, using the generic (unsalted) FO transform on a PKE with 128-bit message space will cause a loss of $\log_2 n_{\mathsf{c}}$ bits of security (i.e., going from 128 bits of security to just 64, for 2^{64} challenge ciphertexts). In our salted transform SFO, however, the reduction incurs a loss of security of around just 2 bits, which is consistent with the loss incurred in the single challenge setting. We give results for two variants of the SFO-transform, one that rejects invalid ciphertexts implicitly and one that does so explicitly, in both the random oracle model (ROM) and the quantum-accessible random oracle model (QROM). To do so, we capture multi-target security for n_{u} many users and n_{c} many ciphertexts-per-user via security notions we denote by $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CPA and $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CCA. For the standard FO transform (with public key hashing), the probability of $\mathcal{A}$, that makes q_{RO} random oracle queries, successfully crafting a valid, colliding ciphertext would be upper-bounded by

$$\frac{2q_{\mathsf{RO}}(n_{\mathsf{u}}n_{\mathsf{c}}) + (n_{\mathsf{u}}n_{\mathsf{c}}^2)}{|\mathcal{M}|} \quad . \tag{1}$$

For the SFO transform, each of the n_{c} challenges samples its own independent salt. With this change, we find that $\mathcal{A}$'s success probability is upper-bounded by

$$\frac{n_{\mathsf{u}}n_{\mathsf{c}}^2}{|\mathcal{M}|2^{\mathsf{len}_{\mathsf{salt}}}} + \frac{2tq_{\mathsf{RO}}}{|\mathcal{M}|} \quad , \tag{2}$$

where t is a small constant (see Sect. D). Considering the same concrete attacker resources mentioned in the attack section above, namely $n_{\mathsf{c}} = 2^{64}$ challenge ciphertexts and message space of size $|\mathcal{M}| = 2^{128}$, the advantage of an adversary making 2^{64} random oracle queries in Eq. (1) against an FO-transformed scheme becomes close to 1. On the other hand, in an SFO-transformed scheme, an adversary making 2^{64} queries will have advantage around 2^{-62}, which is within the target security level. We also show that the salted Fujisaki–Okamoto is secure even in the QROM regardless of its rejection mode, by deploying recently developing QROM techniques. Our results in the ROM are tight, meaning the success probability and run time of any $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CCA adversary against the KEM are close to those of some $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CPA adversary against the PKE. In the QROM, our bounds are non-tight, but in line with the state-of-the-art for quantum security proofs. This non-tightness of the reduction in the QROM is consistent with other FO literature [HHK17,DHK+21]. However, the incorporation of salting induces an even more significant improvement in the security in the QROM than it does in the ROM, precisely because it inflates the hardness of an induced search problem (see Sect. 6).

Additionally, in Sect. E we enumerate, describe, and patch several minor bugs in existing literature on the FO transform.

Existing results on multi-target security. Bellare et al. [BBM00] proved the following bound on multi-target IND-CCA security for a generic protocol:

$$\mathrm{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CCA}} \leq n_{\mathsf{u}} \cdot n_{\mathsf{c}} \cdot \mathrm{Adv}^{\mathsf{IND}\text{-}\mathsf{CCA}} \quad .$$

For FO-based KEMs, significant improvement over the hybrid argument is possible, by reducing $\mathrm{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CCA}}$ to $\mathrm{Adv}^{\mathsf{IND}\text{-}\mathsf{CPA}}$, and then applying the hybrid argument only to $\mathrm{Adv}^{\mathsf{IND}\text{-}\mathsf{CPA}}$. Indeed, Duman et al. [DHK+21] give a bound for $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CCA security for

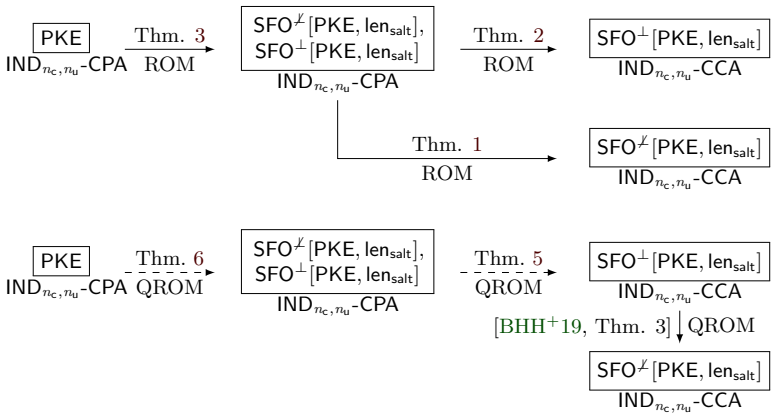

Figure 1: Summary of our results on the salted Fujisaki–Okamoto (SFO) transform. Top: Tight results in the classical random oracle model (Sect. 4). Bottom: Non-tight results in the quantum random oracle model (Sect. 5).

FO-based KEMs with implicit rejection. Adapted to our notation, and slightly different security model (see Remark 1), their Theorem 3.1 yields

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}^{\not\perp}} \leq 2\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}} + \frac{n_u n_c^2 + 2n_c \cdot q_{\mathsf{RO}}}{|\mathcal{M}|} + q_{\mathsf{RO}} \cdot \delta(n_u) \ .$$

These results were motivated by Crystals-Kyber, for which $|\mathcal{M}| = 2^{256}$ across all security levels. For $|\mathcal{M}| = 2^{192}$ or smaller, as in FrodoKEM-640, FrodoKEM-976, HQC-128, HQC-192, and for $n_c \geq 2^{64}$ this bound is not tight, since $n_c q_{\mathsf{RO}}$ may exceed 2^{128}. One may wonder about scaling up the message space in HQC-128 and FrodoKEM-640, in order to apply existing bounds. This would achieve a similar multi-target security bound, as yielded by the SFO transform. However, at least for FrodoKEM, the salting technique has a small impact (less than 1%) on runtime and communication size, whereas doubling message size would come at substantial performance cost [ABD+23a].

Technical challenges. The techniques surrounding the FO transform are highly non-trivial, but mostly well-understood by now. Along many axes, incorporating salts does not significantly change them. However, we encountered many challenges, a few of which we now enumerate.

1. Capturing collision-based attacks is delicate: An adversary aiming to find a collision that induces a vulnerability can exploit all salts that had been used during challenge generation. Handling this requires additional statistical reasoning. Furthermore, irrespective of the incorporation of salting, lifting the proof techniques for the FO-transform into the multi-target setting is quite technical. While this has been done for the (plain) FO transform in [DHK+21], that work only covers implicitly rejecting KEMs. We also cover explicitly rejecting KEMs, which makes the QROM analysis a bit more involved and novel. [DHK+21] also contains some minor errors (see Sect. E) that we patched in our analysis.

2. Incorporating salts must be done carefully, or it can cause unintended consequences in the resulting KEM. E.g., KEMs often aim for implicit rejection, i.e., they reject invalid ciphertexts by returning a pseudorandom key, in order to hide the rejection event. HQC has incorporated salting in order to mitigate the multi-ciphertext attacks mentioned by NIST, but in a previous version, the taken approach made it easy to

distinguish implicit rejection keys from honest keys (which defeats the purpose of implicit rejection.) This issue was identified by Saarinen [Saa25].

3. Finally, there is the challenge of constructing modular multi-target security bounds. The motivation for this is that some KEMs, such as FrodoKEM, are modeled to sample certain values from some idealized distribution (e.g. a discrete Gaussian), but in implementation sample instead from an *approximate* distribution. To capture the impact of this approximation, a full security proof will employ a *distribution substitution* step. This requires a modular analysis, such as [HHK17], only lifted to the multi-target setting, and incorporating salts (see Sect. A for a full explanation). We are not aware of any work that provides a modular analysis of the Fujisaki–Okamoto transform in the multi-target setting. Our modular bounds in Sect. A use an extension of [HHK17, Lemma 2.3]. However, [HHK17, Lemma 2.3] contained two bugs. One of which was patched in a PhD thesis [Höv21], but is not widely known. The other is patched only in Sect. E.

2 Preliminaries

In this section we recall important definitions for correctness of public key encryption schemes and KEMs, as well as previous formulations of the FO transform.

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 (pk, 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 $c \leftarrow \mathsf{Enc}(pk, m)$ of m under the public key pk. If necessary, we explicitly specify the used randomness of encryption by writing $c = \mathsf{Enc}(pk, m; r)$, where $r \leftarrow_{\$} \mathcal{R}$. The decryption algorithm Dec, on input sk and a ciphertext c, yields either a message $m = \mathsf{Dec}(sk, c) \in \mathcal{M}$ or a special symbol $\perp \notin \mathcal{M}$ to show that c is not a valid ciphertext.

A key encapsulation mechanism $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ consists of three algorithms and a finite message space $\mathcal{M}$. The key generation algorithm Gen outputs a key pair (pk, sk), with pk also defining a finite key space $\mathcal{K}$. The encapsulation algorithm Encaps, on input pk, produces a tuple $(c, \mathbf{ss})$ where c is said to be an encapsulation of the key ss. The decapsulation algorithm Decaps, on input sk and an encapsulation c, outputs either $\mathbf{ss} = \mathsf{Decaps}(sk, c)$or a special symbol $\perp \notin \mathcal{M}$ to show that c is not a valid encapsulation.

Correctness notions. Certain public key encryption schemes, for example those based on lattice problems, exhibit correctness errors. These occur when encrypting a message $m \in \mathcal{M}$ and then decrypting the result does not return m. There are several works [DGJ+19, BS20, DRV20, FKK+22] showing how to attack schemes based on correctness errors. Hofheinz, Hövelmanns, and Kiltz [HHK17] developed a statistical notion of correctness that is relevant to security proofs of modular FO transforms, which we recall here:

Definition 1 (δ-correctness of a PKE)**.** A public key encryption scheme PKE with message space $\mathcal{M}$ is called δ-correct if

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

where the expectation is taken over $(pk, sk) \leftarrow_{\$} \mathsf{Gen}()$ and the probability is taken over the internal coins of Enc.

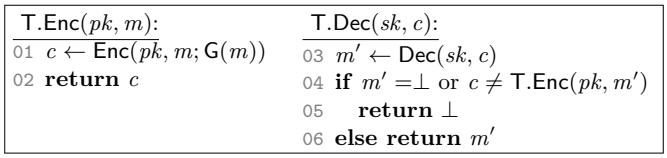

Figure 2: Algorithms of $\mathsf{T}[\mathsf{PKE}, \mathsf{G}]$.

Definition 2 (δ-correctness of a KEM)**.** A key encapsulation mechanism KEM with message space $\mathcal{M}$ is called δ-correct if

$$\Pr[\mathsf{Decaps}(sk, c) \neq \mathbf{ss} | (c, \mathbf{ss}) \leftarrow \mathsf{Encaps}(\mathsf{pk})] \leq \delta \ ,$$

where the probability is taken over the internal coins of Gen and Encaps.

Multi-user correctness. To capture settings with n_{u} many users, we define a corresponding multi-user correctness term $\delta(n_{\mathsf{u}})$ defined almost identically to δ-correctness, for both PKEs and KEMs, except that we take the maximum probability over all $j \in [n_{\mathsf{u}}]$ [DHK+21].

Definition 3 (γ-spreadness)**.** We say that PKE is γ-spread iff for all key pairs $(pk, sk) \in \mathrm{supp}(\mathsf{Gen})$ and all messages $m \in \mathcal{M}$ it holds that

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

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

2.2 The Fujisaki–Okamoto Transform

In this section, we recall the definition of the FO transform as the composition of the two following transformations:

- the de-randomizing T-*transform* that additionally adds a re-encryption check to the decryption procedure; and
- augmented PKE-to-KEM U-*transforms* that derive session keys from a randomly chosen message m, which they encrypt using PKE. The two variants of U vary in their responses to invalid ciphertexts ($\mathsf{U}^{\perp}$ returns $\perp$, while $\mathsf{U}^{\not\perp}$ returns pseudo-random values).

Definition 4 (The T-transform)**.** Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and G be a hash function $\mathsf{G} : \mathcal{M} \to \mathcal{R}$. The transformed deterministic PKE $\mathsf{T}[\mathsf{PKE}, \mathsf{G}]$ is defined in Fig. 2

We recall two variants of the PKE to KEM transformation used in the FO transform. We augment the transform of [HHK17], by including the public key in the inputs to the hash function H when preparing shared secrets, which is done for many KEMs in practice.

Definition 5 (Augmented $\mathsf{U}^{\not\perp}$ and $\mathsf{U}^{\perp}$ transform)**.** Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme, and let H be a hash function. The transformed KEMs $\mathsf{KEM}^{\not\perp} = \mathsf{U}^{\not\perp}[\mathsf{PKE}, \mathsf{H}]$ and $\mathsf{KEM}^{\perp} = \mathsf{U}^{\perp}[\mathsf{PKE}, \mathsf{H}]$ are defined in Fig. 3.

$\underline{\mathsf{KEM}^{\not\perp}.\mathsf{Gen}()}$
01 $(pk, sk) \leftarrow\!\!\$\ \mathsf{Gen}()$
02 $s \leftarrow\!\!\$\ \mathcal{M}$
03 $\mathsf{sk}' \leftarrow (sk, s)$
04 **return** (pk, sk')

$\underline{\mathsf{KEM}.\mathsf{Encaps}(pk)}$
05 $m \leftarrow\!\!\$\ \mathcal{M}$
06 $c \leftarrow \mathsf{Enc}(pk, m)$
07 $\mathsf{ss} \leftarrow \mathsf{H}(pk, m, c)$
08 **return** (ss, c)

$\underline{\mathsf{KEM}^{\not\perp}.\mathsf{Decaps}(\mathsf{sk}', c)}$
09 parse $\mathsf{sk}' \leftarrow (sk, s)$
10 $m' \leftarrow \mathsf{Dec}(sk, c)$
11 **if** $m' = \perp$
12 **return** $\mathsf{H}(pk, s, c)$
13 **return** $\mathsf{H}(pk, m', c)$

$\underline{\mathsf{KEM}^{\perp}.\mathsf{Decaps}(sk, c)}$
14 $m' \leftarrow \mathsf{Dec}(sk, c)$
15 **if** $m' = \perp$
16 **return** $\perp$
17 **return** $\mathsf{H}(pk, m', c)$

Figure 3: $\mathsf{KEM}^{\not\perp}$ (implicitly rejecting) and $\mathsf{KEM}^{\perp}$ (explicitly rejection) constructed by the augmented $\mathsf{U}^{\not\perp}$ and $\mathsf{U}^{\perp}$ transforms to a PKE respectively. The only difference from the original $\mathsf{U}^{\not\perp}$ and $\mathsf{U}^{\perp}$ transforms is that when deriving the session key, we additionally hash in pk. Note that for explicit rejection, the key generation algorithm inherited from the underlying PKE, and that Encaps is the same for either rejection mode.

2.3 Multi-target security notions

We now adapt the relevant standard notions for PKE schemes and KEMs to the multi-target (multi-user, multi-ciphertext) setting. The definitions thus are relative to n_{c}, the number of challenge-ciphertexts-per-user, and n_{u}, the number of users.

Definition 6 (multi-target IND-CPA security ($\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CPA) for PKE)**.** Let PKE be a public key encryption scheme, let n_{u} and n_{c} be positive integers, and let $\mathcal{A}$ be an adversary in the experiment $\mathrm{Exp}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A})$ shown in Fig. 4. The $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CPA *advantage function of an adversary* $\mathcal{A}$ *against* PKE is

$$\mathrm{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A}) := \left|\Pr\left[\mathrm{Exp}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2}\right| \ .$$

$\underline{\mathrm{Exp}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A})}$
01 $b \leftarrow\!\!\$\ \{0,1\}$
02 **for** $j = 1, ..., n_{\mathsf{u}}$
03 $(pk_j, sk_j) \leftarrow\!\!\$\ \mathsf{Gen}()$
04 $\vec{pk} = (pk_1, ..., pk_{n_{\mathsf{u}}})$
05 $b' \leftarrow \mathcal{A}^{\textsc{Chall}_1,\ldots,\textsc{Chall}_{n_{\mathsf{u}}}}(\vec{pk})$
06 **return** $[\![b = b']\!]$

$\underline{\textsc{Chall}_j(m_0, m_1)}$ //At most n_{c} queries
07 **return** $\mathsf{Enc}(pk_j, m_b)$

Figure 4: multi-target security experiment ($\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CPA) against a public key encryption scheme PKE, with n_{u} users and n_{c} ciphertexts-per-user.

Definition 7 (multi-target IND-CCA security ($\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CCA) for KEM)**.** Let KEM be a key encapsulation mechanism, let n_{u} and n_{c} be positive integers, and let $\mathcal{A}$ be an adversary in the experiment $\mathrm{Exp}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{A})$ shown in Fig. 5. The $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CCA *advantage function of an adversary* $\mathcal{A}$ *against* KEM is

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

Omitting the oracle Decaps on line 05 in Fig. 5 yields $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CPA security for a KEM.

Remark 1. The security definition of [DHK+21] differs slightly from ours in that they use a single challenge oracle, which takes an index as input. Their model allows the adversary

$\underline{\mathrm{Exp}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{A})}$
01 $b \leftarrow\!\!\$\ \{0,1\}$
02 **for** $j = 1, ..., n_u$
03 $\quad (pk_j, sk_j) \leftarrow\!\!\$\ \mathsf{Gen}()$
04 $\vec{pk} = (pk_1, ..., pk_{n_u})$
05 $b' \leftarrow \mathcal{A}^{\textsc{Decaps},\textsc{Chall}_1,...,\textsc{Chall}_{n_u}}(\vec{pk})$
06 **return** $[\![b = b']\!]$

$\underline{\textsc{Chall}_j}$ // At most n_c queries
07 $(c, \mathbf{ss}_0) \leftarrow\!\!\$\ \mathsf{Encaps}(pk_j)$
08 $\mathbf{ss}_1 \leftarrow\!\!\$\ \mathcal{K}$
09 $\mathfrak{L}_{C_j} \leftarrow \mathfrak{L}_{C_j} \cup \{c\}$
10 **return** $(c, \mathbf{ss}_b)$

$\underline{\textsc{Decaps}(j, c \notin \mathfrak{L}_{C_j})}$
11 **return** $\mathsf{Decaps}(sk_j, c)$

Figure 5: multi-target security experiment (IND_{n_c,n_u}-CCA) against a key encapsulation mechanism KEM, with n_u users and n_c encapsulations.

to concentrate all queries on a single user. Our model aligns with that of [BBM00], where challenge queries are restricted per-user, and the total number of challenges is bounded by $n_u \cdot n_c$. We believe that this notion better captures realistic attack scenarios, and yields cleaner analysis.

Another consideration in modeling multi-target security is whether b should be global, or sampled independently for each user. In the latter scenario, the adversary wins if they provide an index, and output associated challenge bit at that index. In [HS21] Heum and Stam consider the impact of opting for global bit vs. bit-per-user security notions and recommend the former, as it is proven to be strictly stronger.

Public key hashing. To mitigate multi-user attacks, a common security measure is to tie ciphertexts and their decapsulations to the user's public key by including pk into the input to the hash function used to derive encryption randomness and session keys. However, hashing all of pk can be a costly measure, especially in lattice-based schemes where public keys are large matrices, rather than short bit strings. In many cases, it may already be enough to hash only a bit string that uniquely identifies the public key: Duman et al. [DHK+21] show that, for an identifying function with sufficient entropy, one can get the same security level, with a significant decrease in overhead, with experimental results showing that overhead could be reduced by over 30% for certain lattice-based KEMs. This amounts to hashing a small, unpredictable part of pk, by using an identifying function $\mathsf{ID} : pk \to \{0,1\}^\ell$. This technique is known as *prefix hashing*.

Results on a transformation that use the full public key can be adapted to a transformation that uses prefix hashing as follows. Insert before the first game-hop a new game that aborts if there are colliding public key identifiers, namely if there are indices $i \neq j$ with $\mathsf{ID}(pk_i) = \mathsf{ID}(pk_j)$. By the birthday bound, the probability of a collision is $n_u^2/2^\ell$. Provided that there are no collisions, each public key has a unique identifier. The rest of the security analysis can thus proceed as in the full-key-hashing case and results incurs only an additional term $n_u^2/2^\ell$.

3 The Salted FO transform

In this section, we formalize the salting countermeasure introduced in [ABD+23a] to thwart collision attacks. The countermeasure introduces a uniformly random salt in the encryption process, so that the chance of a collision between the pre-computed ciphertexts and the challenge ciphertexts depends not only on the message space size $|\mathcal{M}|$ and the number of collected ciphertexts n_c, but also on the length of the uniformly random salt. We formalize this approach as a modified T-transform which we call the *salted* T*-transform* (the ST-transform).

Definition 8 (ST-transform)**.** Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption

scheme, let G be a hash function, and let $\mathsf{len}_{\mathsf{salt}}$ be a non-negative integer. To PKE, G and $\mathsf{len}_{\mathsf{salt}}$ we associate public key encryption scheme

$$\mathsf{ST}[\mathsf{PKE}, \mathsf{G}] = \mathsf{PKE}_1 = (\mathsf{Gen}, \mathsf{Enc}_1, \mathsf{Dec}_1) \ ,$$

with algorithms Enc_1 and Dec_1 defined in Fig. 6.

$\mathsf{ST}[\mathsf{PKE}, \mathsf{G}].\mathsf{Enc}_1(pk, m)$	$\mathsf{ST}[\mathsf{PKE}, \mathsf{G}].\mathsf{Dec}_1(sk, c\|\|\mathsf{salt})$
01 $\mathsf{salt} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$	05 $m' \leftarrow \mathsf{PKE.Dec}(sk, c)$
02 $r \leftarrow \mathsf{G}(pk, m\|\|\mathsf{salt})$	06 $r' \leftarrow \mathsf{G}(pk, m'\|\|\mathsf{salt})$
03 $c \leftarrow \mathsf{PKE.Enc}(pk, m; r)$	07 **if** $m' = \perp$ **or** $\mathsf{PKE.Enc}(pk, m'; r') \neq c$
04 **return** $c\|\|\mathsf{salt}$	08 **return** $\perp$
	09 **else return** m'

Figure 6: Public key encryption scheme $\mathsf{ST}[\mathsf{PKE}, \mathsf{G}]$ constructed by the salted T-transform. ST deviates from [HHK17]'s FO T-transform in two ways: to increase the search space, ST includes salts; and ST binds ciphertexts to their users by additionally tying the encryption randomness to pk.

The resulting salted KEMs. The FO-transform is usually obtained by composing the T-transform with one of [HHK17]'s PKE-to-KEM transformations $\mathsf{U} \in \{\mathsf{U}^{\perp}, \mathsf{U}^{\not\perp}\}$. Similarly, we obtain the salted FO transformations by combining U with the salted T-transform ST.

Definition 9 ($\mathsf{SFO}^{\not\perp}$ and $\mathsf{SFO}^{\perp}$ transforms)**.** For a public key encryption scheme PKE, a salt length parameter $\mathsf{len}_{\mathsf{salt}}$, and hash functions G, H, the salted FO transforms yield the KEMs

$$\mathsf{KEM}^{\not\perp} := \mathsf{SFO}^{\not\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}] \quad \text{and} \quad \mathsf{KEM}^{\perp} := \mathsf{SFO}^{\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$$

with algorithms described in Fig. 7 respectively.

Remark 2. Certain results of this paper, such as Thm. 3 refer to KEMs such as $\mathsf{KEM} := \mathsf{SFO}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$ with no rejection mode specified. In those cases, the result holds regardless of which rejection mode is used.

Other possible variants. One could also define a version of the SFO transform, which does not include the ciphertext as input to H during encapsulation and decapsulation. This would entail composing the ST-transform with either $\mathsf{U}^{\perp}_{m}$ or $\mathsf{U}^{\not\perp}_{m}$ from [HHK17]. Bindel et al. [BHH+19] show that in the single-challenge setting, deriving a KEM key as $\mathsf{H}(m)$ equivalent to using $\mathsf{H}(m, c)$ in that the $\mathsf{IND\text{-}CCA}$ security of either KEM is equivalent. To briefly summarize, the argument provided there is that for $\mathsf{KEM}_1 = \mathsf{U}_{m,c}[\mathsf{PKE}, \mathsf{H}_1]$ and $\mathsf{KEM}_2 = \mathsf{U}_{m}[\mathsf{PKE}, \mathsf{H}_2]$, an adversary against the $\mathsf{IND\text{-}CCA}$ security of KEM_2 can perfectly simulate the $\mathsf{IND\text{-}CCA}$ game for KEM_1 by sampling a new random oracle H and setting

$$\mathsf{H}_1(m, c) = \begin{cases} \mathsf{H}_2(m), & \text{if } c = \mathsf{Enc}(pk, m), \\ \mathsf{H}(m, c), & \text{otherwise.} \end{cases}$$

In the other direction, an adversary against the $\mathsf{IND\text{-}CCA}$ security of KEM_1 can simulate the $\mathsf{IND\text{-}CCA}$ game for KEM_2 by letting $\mathsf{H}_2(m) \leftarrow \mathsf{H}_1(m, \mathsf{Enc}(pk, m))$.

This equivalence does not extend to the multi-user (or, for that matter, multi-target) setting, as there would be n_{u} public keys to consider. In fact, $\mathsf{H}(m)$ is in no way connected

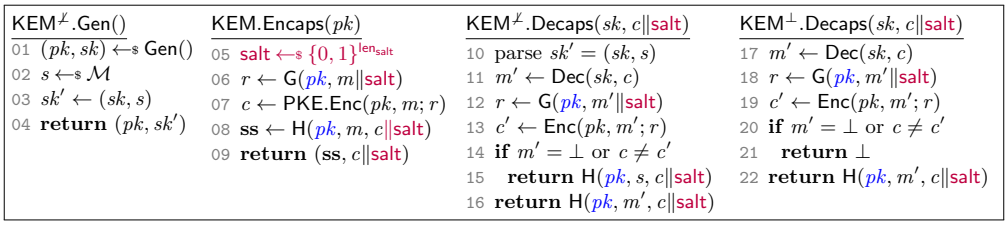

Figure 7: KEM$^{\not\perp}$ from the salted FO transforms SFO$^{\not\perp}$ and SFO$^{\perp}$. Note that for explicit rejection, the key generation algorithm is inherited from the underlying PKE, and that Encaps is the same for either rejection mode.

to the public key of any user, while $\mathsf{H}(m, c \leftarrow \text{Enc}(pk_j, m))$ is. However, if the public key hash is included, a similar argument does hold, with the simulation being

$$\mathsf{H}_1(pk_j, m, c) = \begin{cases} \mathsf{H}_2(pk_j, m), & \text{if } c = \text{Enc}(pk_j, m), \\ \mathsf{H}(pk_j, m, c), & \text{otherwise.} \end{cases}$$

The difference being that now the adversary is required to have knowledge of the public key in either case. Our work focuses on the SFO variant which hashes both m and c, since this is consistent with FrodoKEM and HQC.

4 Security proofs in the ROM

In this section, we prove IND$_{n_c,n_u}$-CCA security for KEMs built from the SFO transform. We break up our results into smaller theorems, for the purpose of making them easier to follow. First, we use known methods to show how to simulate a decapsulation oracle, for implicit and explicit rejection KEMs. This shows that if SFO achieves IND$_{n_c,n_u}$-CPA security, then it also achieves IND$_{n_c,n_u}$-CCA security. These results are largely unaffected by the incorporation of salting, and could be derived from existing literature, especially [DHK+21], by an informed reader. Thus, we omit the proofs from the main body of our work, and instead include them in Sect. B. Then we prove that the SFO transform tightly preserves IND$_{n_c,n_u}$-CPA security in transforming a PKE to a KEM, where the salting introduces more novel aspects.

4.1 Simulating the decapsulation oracle in the ROM: IND$_{n_c,n_u}$-CPA to IND$_{n_c,n_u}$-CCA

In the following theorems, we show that one can simulate decapsulation oracles with either implicit or explicit rejection for (S)FO-based KEMs. These techniques have been used in [Den03, HHK17, DHK+21, HHM22, HM24], usually in conjunction with other proof steps, to provide a monolithic reduction between IND-CCA security of a KEM, to IND-CPA security of a PKE. Intuitively, we apply a series of modifications to the decapsulation oracle, which remove dependence on the secret key, and bound the changes that occur at each step. We then apply a patching technique to ensure that the outputs of H, the random oracle used for producing KEM keys, will match the output of Decaps. The full proofs can be found in Sect. B.

Theorem 1 (Simulation of DECAPS$^{\not\perp}$). *Let* KEM$^{\not\perp}$ = SFO$^{\not\perp}$[PKE, G, H, len$_{\text{salt}}$]. *For any* IND$_{n_c,n_u}$-CCA *adversary* $\mathcal{B}$ *against* KEM$^{\not\perp}$, *issuing at most* q_{H} *queries to* H, *and* q_D *queries*

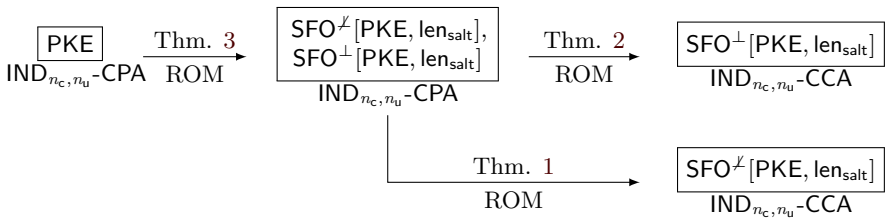

Figure 8: Summary of our results on the salted Fujisaki–Okamoto (SFO) transform in the ROM.

to DECAPS$^{\not\perp}$*, there exists an* IND$_{n_c,n_u}$-CPA *adversary* $\mathcal{A}$ *against* KEM$^{\not\perp}$ *such that*

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}^{\not\perp}}(\mathcal{B}) \le \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{KEM}^{\not\perp}}(\mathcal{A}) + \frac{q_{\mathsf{H}}}{|\mathcal{M}|} + (q_{\mathsf{H}} + q_D) \cdot \delta(n_u)$$

Theorem 2 (Simulation of DECAPS$^{\perp}$)**.** *Let* KEM$^{\perp}$ = SFO$^{\perp}$[PKE, G, H, len$_{salt}$]*. For any* IND$_{n_c,n_u}$-CCA *adversary* $\mathcal{B}$ *against* KEM$^{\perp}$*, issuing at most* q_{H} *queries to* H*, and* q_D *queries to* DECAPS$^{\perp}$*, there exists an* IND$_{n_c,n_u}$-CPA *adversary* $\mathcal{A}$ *against* KEM$^{\perp}$ *such that*

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}^{\perp}}(\mathcal{B}) \le \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{KEM}^{\perp}}(\mathcal{A}) + q_D 2^{-\gamma} + (q_{\mathsf{H}} + q_D) \cdot \delta(n_u)$$

4.2 Tight passive security of SFO

In Thm. 3 below we show that SFO *tightly* preserves multi-target security of PKE into (now passive) multi-target security of the salted KEM.

Theorem 3 (PKE IND$_{n_c,n_u}$-CPA $\xRightarrow{\text{ROM}}$ SFO[PKE, len$_{salt}$] IND$_{n_c,n_u}$-CPA)**.** *Let* PKE *be a public key encryption scheme, and let* KEM := SFO[PKE, G, H, len$_{salt}$]*. If* PKE *is* $\delta(n_u)$*-correct, then* KEM *is* $\delta(n_u)$*-correct in the random oracle model. Let the number of queries to* CHALL$_j$ *be bounded by* n_c *and* $\mathfrak{L}_{S_j}$ *represent the set of salt values sampled during those queries. Consider a* IND$_{n_c,n_u}$-CPA *adversary* $\mathcal{B}$ *against* KEM*, issuing at most* q_{H} *queries to* H*,* q_{G} *queries to* G*, and let* $q = (q_{\mathsf{H}} + q_{\mathsf{G}})$*. Suppose that no* salt *appears in any* $\mathfrak{L}_{M_j}$ *more than* t *times. There exists an* IND$_{n_c,n_u}$-CPA *adversary* $\mathcal{A}$ *against* PKE *such that*

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{KEM}}(\mathcal{B}) \le \frac{n_u n_c (n_c - 1)}{|\mathcal{M}| \cdot 2^{\mathsf{len}_{salt}}} + \frac{2t(q_{\mathsf{H}} + q_{\mathsf{G}})}{|\mathcal{M}|} + 2 \cdot \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A}) \ ,$$

Furthermore $\mathcal{A}$ *and* $\mathcal{B}$ *have similar run-time.*

The correctness bound is trivial. To summarize the security proof: instead of sending the proper challenge KEM keys $\mathbf{ss}_0$ in the case $b = 0$, we want to always send a random key, leaving $\mathcal{B}$ with randomly guessing b as its only option. $\mathcal{B}$ could only notice this replacement of $\mathbf{ss}_0 = \mathsf{H}(pk_j, m, c)$ if they already queried H on (pk_j, m, c). To make such queries harder to achieve, we exploit IND$_{n_c,n_u}$-CPA security of PKE: we can replace the encryptions of the proper challenge seeds m with encryptions of independent messages, after which $\mathcal{B}$ would have to query H on messages about which it has no information at all. The IND$_{n_c,n_u}$-CPA adversary $\mathcal{A}$ simulates the KEM challenge oracle by sampling a message/salt pair (m_0, salt) and an additional message m_1, uses their PKE challenge oracle to obtain $c \leftarrow \mathsf{Enc}(pk_j, m_b)$, samples a uniformly random key and returns $(c, \mathbf{ss})$. What we glossed over so far is that $\mathcal{A}$ obtains encryptions of the plain encryption scheme PKE, whereas $\mathcal{B}$ expects ST-encryptions, so encryptions using the specific fixed randomness $r \leftarrow \mathsf{G}(pk_j, m \| \mathsf{salt})$. To show that this specific randomness is indistinguishable from uniform, we thus argue that queries to G involving $(m \| \mathsf{salt})$ are unlikely, which is captured with the technique already used for H.

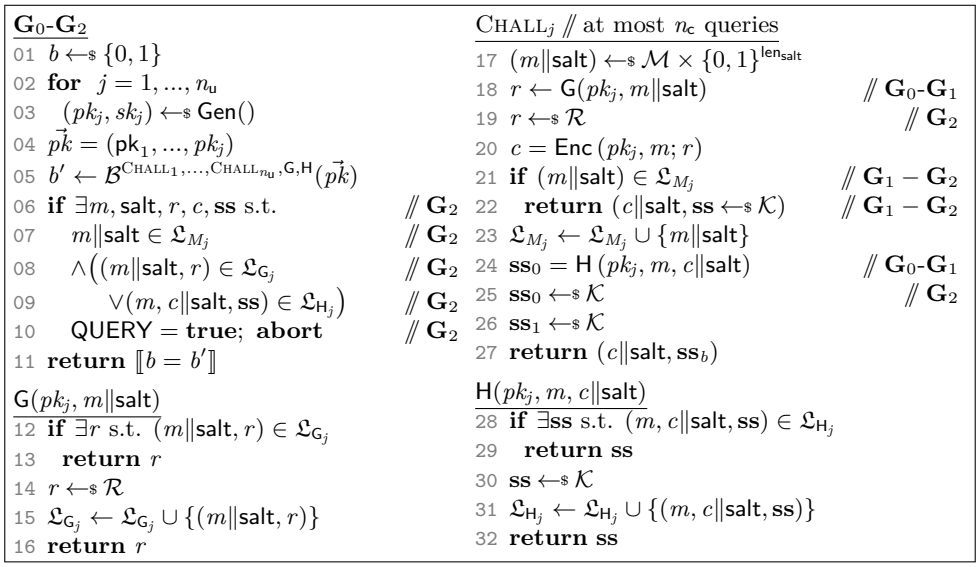

Figure 9: Games for the proof of Thm. 3.

Proof. Consider the sequence of games shown in Fig. 9.

Game $\mathbf{G}_0$. $\mathbf{G}_0$ is the original IND_{n_c,n_u}-CPA-game against the KEM constructed as $\mathsf{SFO}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$.

$$\left|\Pr[\mathbf{G}_0 \Rightarrow 1] - \frac{1}{2}\right| = \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-CPA}}_{\mathsf{KEM}}(\mathcal{B})$$

Game $\mathbf{G}_1$: capture challenge repetitions. As a first preparation step, $\mathbf{G}_1$ deals with repeated message-salt tuples: If a user sampled the same message-salt tuple twice, they would also return the same KEM key twice when $b = 0$, but not when $b = 1$. This would trivially allow the adversary to determine b. We thus first ensure that the challenge response $\mathbf{ss}_0$ is not given out a second time even if the respective message-salt tuple was repeatedly sampled by the user, see line 22. Since the games only differ if such a resampling of message/salt pairs occurs for any user, and since the total number of challenge messages for each user is n_c, the birthday bound yields

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq \frac{n_u n_c (n_c - 1)}{|\mathcal{M}| \cdot 2^{\mathsf{len}_{\mathsf{salt}}}}.$$

Game $\mathbf{G}_2$: randomize challenges. In $\mathbf{G}_2$, we randomize our challenges both with respect to encryption randomness and 'honest' session key: we change challenge oracle $\textsc{Chall}_j$ so that it samples the encryption randomness r at random, rather than setting $r \leftarrow \mathsf{G}(pk_j, m\|\mathsf{salt})$, and such that it also randomizes the 'honest' session key $\mathbf{ss}_0 = \mathsf{H}(pk_j, m, c\|\mathsf{salt})$. The KEM adversary $\mathcal{B}$ will not notice these changes unless they accessed $\mathsf{G}(pk_j, m\|\mathsf{salt})$ or $\mathsf{H}(pk_j, m, c\|\mathsf{salt})$. To capture this, we raise a flag QUERY and abort if $\mathcal{B}$ poses such a query: we abort if $\mathcal{B}$ queries G on a tuple $(pk_j, m\|\mathsf{salt})$ for which $m\|\mathsf{salt}$ previously was stored in the challenge seed list $\mathfrak{L}_{M_j}$. We also abort if the adversary queries H on a tuple $(pk_j, m, c\|\mathsf{salt})$ for which $m\|\mathsf{salt}$ previously was stored in the challenge seed list $\mathfrak{L}_{M_j}$. Since the games do not differ unless QUERY occurs,

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \Pr[\mathsf{QUERY}] \ .$$

Since $\mathbf{G}_2$ provides $\mathcal{B}$ with random shared secrets regardless of the bit b, rendering $\mathcal{B}$'s view completely independent of b,

$$|\Pr[\mathbf{G}_2 \Rightarrow 1]| = \frac{1}{2} \ .$$

It remains to bound QUERY.

Bounding QUERY. To this end, we construct an IND_{n_c,n_u}-CPA adversary $\mathcal{A}$ against PKE that perfectly simulates $\mathbf{G}_2$ for $\mathcal{B}$ and wins if QUERY is raised. Adversary $\mathcal{A}$ forwards its input vector of public keys to $\mathcal{B}$ and simulates the random oracles according to $\mathbf{G}_2$. For each user index j, $\mathcal{A}$ also initializes two empty lists $\mathcal{M}_{j0}$ and $\mathcal{M}_{j1}$ which it will use for bookkeeping during its simulation of $\textsc{Chall}_j$. To simulate $\textsc{Chall}_j$, $\mathcal{A}$ samples $\mathsf{salt} \leftarrow \{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$ and stores salt in a 'user's salts' list $\mathfrak{L}_{S_j}$. Then $\mathcal{A}$ samples $(m_0, m_1) \leftarrow \mathcal{M}^2$ and stores $m_0\|\mathsf{salt}$ in $\mathcal{M}_{j_0}$, and $m_1\|\mathsf{salt}$ in $\mathcal{M}_{j1}$.

Then $\mathcal{A}$ queries its CPA challenge oracle on input (m_0, m_1) to obtain $c = \mathsf{Enc}(pk_j, m_b)$. $\mathcal{A}$ samples a random $\mathbf{ss} \leftarrow_{\$} \mathcal{K}$ and returns $(c\|\mathsf{salt}, K)$ to $\mathcal{B}$. Regardless of $\mathcal{A}$'s challenge bit, this output perfectly matches the output of $\textsc{Chall}_j$ in $\mathbf{G}_2$.

We will now discuss how $\mathcal{A}$ can win their own game. Intuitively, $\mathcal{A}$ looks for adversarial queries that intersect with either $\mathfrak{L}_{M_{j,0}}$ or $\mathfrak{L}_{M_{j,1}}$. If such an intersection occurs in $\mathfrak{L}_{M_{j,b'}}$, $\mathcal{A}$ returns b'. If QUERY occurs, then the message included in this query is found in $\mathcal{M}_{jb}$ for $\mathcal{A}$'s challenge bit b. $\mathcal{A}$ will thus return b if one of $\mathcal{B}$'s random oracle queries $(pk_j, m\|\mathsf{salt})$ included a message m stored in $\mathcal{M}_{jb}$. If no query was made that can be found in either of the two lists, $\mathcal{A}$ returns a random guess. If queries were made with respect to both lists, $\mathcal{A}$ will also return a random guess. The remaining issue is that $\mathcal{B}$ might derail $\mathcal{A}$'s guess by querying G or H on pk_j and $m \in \mathfrak{L}_{M_{j,1-b}}$, so on a message that has nothing to do with the ciphertexts $\mathcal{B}$ received. We denote this event by COLLISION, that is, we let COLLISION be the event that $\mathcal{B}$ queries G on any $(pk_j, m\|\mathsf{salt})$ or H on $(pk_j, m, c\|\mathsf{salt})$ such that $m\|\mathsf{salt} \in \mathfrak{L}_{M_{j,1-b}}$. If QUERY is raised, but COLLISION is not, then $\mathcal{A}$ accurately detects that a random oracle query containing $m \in \mathcal{M}_{jb}$ triggered QUERY, returns b, and wins. If neither QUERY nor COLLISION is raised, $\mathcal{A}$ returns a random guess, as is the case when both QUERY and COLLISION are raised. So, provided COLLISION is not raised, $\mathcal{A}$ wins the game with probability 1 if QUERY is raised and with probability $\frac{1}{2}$ otherwise. Hence

$$\begin{aligned}
\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A}) + \Pr[\mathsf{COLLISION}] &\geq \left|\Pr[b = b' | \neg\mathsf{COLLISION}] - \frac{1}{2}\right| \\
&= \left|\Pr[\mathsf{QUERY}] + \frac{1}{2}\Pr[\neg\mathsf{QUERY}] - \frac{1}{2}\right| \\
&= \frac{1}{2}\Pr[\mathsf{QUERY}].
\end{aligned}$$

For a tuple $(pk_j, m\|\mathsf{salt})$ we have

$$\Pr[(m\|\mathsf{salt}) \in \mathfrak{L}_{M_{j,1-b}}] = \begin{cases} \frac{n_c}{|\mathcal{M}||\mathfrak{L}_{S_j}|} & \textbf{if } \mathsf{salt} \in \mathfrak{L}_{S_j} \\ 0 & \textbf{otherwise} \end{cases}$$

We define q_j to be the total number of queries to H or G which include pk_j. Thus $\sum_{j\in[n_u]} q_j \leq q_{\mathsf{H}} + q_{\mathsf{G}}$. Let $\mathfrak{L}_{M_{j,1-b}}[\mathsf{salt}] = \{m \in \mathcal{M} | (m\|\mathsf{salt}) \in \mathfrak{L}_{M_{j,1-b}}\}$. For a query $(pk_j, m'\|\mathsf{salt}')$, there are at most t values in $\mathfrak{L}_{M_{j,1-b}}$ with which it could collide. Thus we have

$$\Pr[(m'\|\mathsf{salt}') \in \mathfrak{L}_{M_{j,1-b}}] \leq \frac{t}{|\mathcal{M}|}.$$

Taking a union bound over all q_j queries made with index j, the probability that any query

made by $\mathcal{A}$ collide with some $(m\|\mathsf{salt}) \in \mathfrak{L}_{M_{j,1-b}}$ is bounded by $\frac{t \cdot q_j}{|\mathcal{M}|}$. Then we have

$$\Pr[\mathsf{COLLISION}] \leq \sum_{j \in [n_u]} \frac{t \cdot q_j}{|\mathcal{M}|} \leq \frac{t \cdot (q_\mathsf{H} + q_\mathsf{G})}{|\mathcal{M}|}.$$

□

Remark 3 (A Note on HQC). In HQC, the transform used differs slightly from our SFO transform, insofar as the salt is not used in the session key derivation, but only in encryption. This does not significantly impact the security proof. To adapt the proof of Thm. 3 to this setting, one must modify the $\mathbf{G}_2$ so that the flag QUERY is raised whenever there exists m, salt such that $(m\|\mathsf{salt}) \in \mathfrak{L}_{M_j}$ and either:

1. $(pk_j, m\|\mathsf{salt})$ was queried to G;
2. or (pk_j, m, c) was queried to H, and $c = \mathsf{Enc}(pk_j, m; \mathsf{G}(m\|\mathsf{salt}))$.

Aside from this small change, the proof would be identical.

5 The Salted FO transform in the QROM

Before adapting our two ROM proof steps to the QROM in Sections 5.2 and 5.3, we first collect some necessary helper theorems about the QROM in Sect. 5.1. A summary of the results can be found in Fig. 1.

5.1 Online-extractable QROMs and One-Way To Hiding (OWtH)

We will use the extractable QROM variant [HHM22] of OWtH (extOWtH). This variant integrates semi-classical OWtH [AHU19] into the extractable QROM framework developed in [DFMS22]. Before giving the respective theorems, we recollect some intuition and contextualization for the reader's convenience.

Extractable OWtH. We use the adaptation that lifted OWtH into the extractable QROM framework because the extractable QROM framework allows almost-classical reasoning. (In our application, it helps with simulating the decapsulation oracle in a way that is comparably close to our ROM simulation.) This framework models a quantum-accessible random oracle $\mathsf{O} : X \to Y$ as a compressed oracle eCO with random oracle interface $\mathsf{eCO.RO}$, plus an additional 'extraction' oracle interface $\mathsf{eCO.Ext}$. Intuitively, that additional interface $\mathsf{eCO.Ext}$ can be used as a replacement for query book-keeping. It is defined relative to a function $f : X \times Y = T$ that maps the domain X of a random oracle O and its co-domain Y to some other 'target' set T. $\mathsf{eCO.Ext}$ takes as input a classical target value $t \in T$. Intuitively, $\mathsf{eCO.Ext}$ performs a quantum analogue of going through the random oracle queries that were issued so far and then returning a query x such that $f(x, \mathsf{O}(x)) = t$, if such an x exists. (For our purposes, we will model the randomness-generating oracle G as an extractable QROM $\mathsf{eCO.RO}$ and define $\mathsf{eCO.Ext}$ relative to a function f for which $\mathsf{eCO.Ext}$ helps with identifying the plaintexts of ciphertexts.)

To that end, it performs suitable measurements on the oracle database: for each target $t \in T$, we define a projective measurement $\mathcal{M}^t$. $\mathcal{M}^t$ measures according to the measurement projectors $\{\Sigma^{t,x}\}_{x \in X} \cup \{\Sigma^{t,\varnothing}\}$ that are defined as follows: for $x \in X$, the projector $\Sigma^{t,x}$ selects the case where D_x is the first register (in lexicographical order) that contains y such that $f(x, y) = t$, i.e., it is defined as

$$\Sigma^{t,x} \leftarrow \bigotimes_{x' < x} \bar{\Pi}^{t,x'}_{D_{x'}} \otimes \Pi^{t,x}_{D_x}, \quad \text{with} \quad \Pi^{t,x} = \sum_{\substack{y \in Y: \\ f(x,y)=t}} |y\rangle\langle y| \quad \text{and} \quad \bar{\Pi} = id - \Pi \ . \tag{3}$$

The remaining projector $\Sigma^{t,\varnothing}$ captures the case where no register contains such a y:

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

Effect of eCO.Ext on adversarial behavior. We start by restating a helper theorem [DFMS22, Thm. 4.3]. Intuitively, the first item states that any quantum-accessible QROM can be replaced by an extractable one, and items 2a-2c express that calls to the extraction interface can be introduced into the run of a game by 'commuting' them into the game, i.e., from after the game (where they have no impact on the adversary whatsoever) to the desired point during the game run, provided that f behaves sufficiently unpredictable. This unpredictability requirement is formalized via value $\Gamma(f)$ in item 2c.

Lemma 1 (Online extractability (Part of Theorem 4.3 in [DFMS22]))**.** *The extractable RO simulator* eCO*, with interfaces* eCO.RO *and* eCO.Ext*, satisfies the following properties.*

1. *If* eCO.Ext *is unused,* eCO *is perfectly indistinguishable from a random oracle.*

2.a *Any two subsequent independent queries to* eCO.RO *commute. In particular, two subsequent* classical eCO.RO*-queries with the same input x give identical responses.*

2.b *Any two subsequent independent queries to* eCO.Ext *commute. In particular, two subsequent* eCO.Ext*-queries with the same input t give identical responses.*

2.c *Any two subsequent independent queries to* eCO.Ext *and* eCO.RO *$8\sqrt{2\Gamma(f)/2^n}$-almost-commute, where $\{0,1\}^n$ is the codomain of the random oracle and*

$$\Gamma(f) \leftarrow \max_{x,t} |\{y \mid f(x,y) = t\}| \ .$$

Furthermore, the total runtime and quantum memory footprint of eCO*, when using the sparse representation of the compressed oracle, are bounded as*

$$\begin{aligned} \mathrm{Time}(\mathsf{eCO}, q_{RO}, q_E) &= O\big(q_{RO} \cdot q_E \cdot \mathrm{Time}[f] + q_{RO}^2\big), \text{ and} \\ \mathrm{QMem}(\mathsf{eCO}, q_{RO}, q_E) &= O\big(q_{RO}\big). \end{aligned}$$

where q_E and q_{RO} are the number of queries to eCO.Ext *and* eCO.RO*, respectively.*

One-Way-To-Hiding (OWtH). When analyzing FO-constructed KEMs in the QROM, it is common to apply OWtH. In our context, OWtH allows a QROM equivalent to the ROM argument done in game 2 of the proof of Thm. 3: there we argued that the encapsulated challenge key $\mathbf{ss} = \mathsf{H}(pk_j, m, c \| \mathsf{salt})$ looks completely random unless the attacker queried H on the challenge plaintext m (and thus broke security of PKE). OWtH allows a quantum analogue of that step. Intuitively, OWtH says 'the adversary posed superposition queries to H with enough amplitude on m such that we can find it'.

In general, OWtH shows that a distinguisher $\mathcal{A}$ cannot distinguish the extractable oracle from one that was reprogrammed on certain inputs, unless $\mathcal{A}$ managed to pose oracle queries with substantial amplitude on one of the reprogrammed positions. To model this, the framework defines a set of reprogrammed positions $\mathcal{S}$ as follows: the input *inp* of the distinguishing algorithm $\mathcal{A}$ is assumed to be classical, i.e., generated by an algorithm GenInp with classical access to the superposition oracle. $\mathcal{S}$ is defined as the set of all random oracle inputs x queried by GenInp. E.g., for input $(c^*, K^*) := (\mathsf{Enc}(\mathsf{pk}, m^*; \mathsf{G}(m^*))), \mathsf{H}(m^*))$, $\mathcal{S}$ is $\{m^*\}$. To model reprogramming, a superposition oracle eCO^0 is used to generate $\mathcal{A}$'s input, but instead of giving $\mathcal{A}$ access to that oracle, $\mathcal{A}$ is given access to a freshly initialized extractable oracle eCO^1. To identify queries with high amplitude on $\mathcal{S}$, the

games use *'punctured'* versions $\mathsf{eCO}\backslash\mathcal{S}$ of the oracles eCO: $\mathsf{eCO}\backslash\mathcal{S}$ behaves according to the random oracle $\mathsf{eCO.RO}$, but only after having applied an additional *'semi-classical'* oracle O_S^{SC} that essentially marks if an element of $\mathcal{S}$ was found in one of the query registers, by performing a suitable measurement. The event that any measurement performed by O_S^{SC} on the query registers returned 1 is denoted by FIND.

We now restate [HHM22, Theorem 6] that relates the distinguishing advantage between eCO^0 and eCO^1 to the probability that FIND occurs.

Theorem 4 (Extractable OWtH: Distinguishing to Finding [HHM22, Theorem 6])**.** *Let eCO^0 and eCO^1 be two extractable superposition oracles from $\mathcal{X}$ to $\mathcal{Y}$, with their respective extraction interfaces defined relative to a function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{T}$. Let GenInp be an algorithm generating a classical input inp, having access to eCO^0. Let $\mathcal{S}$ be the set of elements $x \in \mathcal{X}$ whose oracle values were queried to compute inp, and let $\mathcal{T}_{\mathcal{S}} := \{t \mid \exists x \in \mathcal{S} \text{ s.th. } t = f(x, \mathsf{eCO}^0(x))\}$.*

We define the OWTH distinguishing advantage function of $\mathcal{A}$ as

$$\mathrm{Adv}^{\mathsf{OWTH}}_{\mathsf{eCO},f}(\mathcal{A}) := |\Pr[1 \leftarrow \mathcal{A}^{\mathsf{eCO}^0}(inp)] - \Pr[1 \leftarrow \mathcal{A}^{\mathsf{eCO}^1}(inp)]| \ ,$$

where the probabilities are taken over the coins of GenInp and the internal randomness of $\mathcal{A}$. For any algorithm $\mathcal{A}$ of query depth d with respect to $\mathsf{eCO.RO}$ that never performs an extraction query on any $t \in \mathcal{T}_{\mathcal{S}}$, we have

$$\mathrm{Adv}^{\mathsf{OWTH}}_{\mathsf{eCO},f}(\mathcal{A}) \leq 4 \cdot \sqrt{d \cdot \Pr[\mathsf{FIND} : \mathcal{A}^{\mathsf{eCO}^1\backslash\mathcal{S}}(inp)]} \ .$$

In one part of our proof, the reprogramming set $\mathcal{S} = \{m_1^*, \cdots, m_n^*\}$ will have become completely independent of the attacker's view. There we will use [HHM22, Corollary 4] below which bounds the probability of FIND in this special case.

Corollary 1 (Extractable OWTH: Finding independent values [HHM22, Corollary 4])**.** *Let eCO, f, GenInp, FIND, be like in Thm. 4, for a set $\mathcal{S} = \{x_1, \cdots, x_n\}$ of uniformly chosen elements $x_1, \cdots, x_n$. If $\mathcal{S}$ and inp are independent, then for any algorithm $\mathcal{A}^{\mathsf{eCO}}$ issuing q many queries to $\mathsf{eCO.RO}$ in total,*

$$\Pr[\mathsf{FIND} : \mathcal{A}^{\mathsf{eCO}\backslash\{x\}}(inp)] \leq \frac{4qn}{|X|} \ .$$

5.2 From IND-CPA-KEM to IND-CCA-KEM in the QROM

In this section, we lift our IND_{n_c,n_u}-CPA-KEM to IND_{n_c,n_u}-CCA-KEM result, Thm. 1, to the quantum-accessible random oracle model.

Theorem 5 (Simulatability of salted $\textsc{Decaps}^{\perp}$ in the QROM)**.** *Let PKE be a public key encryption scheme, let $\mathsf{PKE}_1 := \mathsf{ST}[\mathsf{PKE}, \mathsf{G}]$, and let $\mathsf{KEM}^{\perp} = \mathsf{SFO}^{\perp}_{m,c}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$. Let $\mathcal{A}$ be a IND_{n_c,n_u}-CCA adversary against $\mathsf{KEM}^{\perp}$, issuing at most q_{G} many queries to G, q_D queries to $\textsc{Decaps}^{\perp}$, and with d and w being the combined query depth/width of $\mathcal{A}$'s random oracle queries. Then there exist an IND_{n_c,n_u}-CPA adversary $\mathcal{B}$ against $\mathsf{KEM}^{\perp}$ and an $\mathsf{FFP\text{-}CCA}_u$ adversary $\mathcal{C}$ against PKE_1 in the extractable QROM such that*

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}^{\perp}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{KEM}^{\perp}}(\mathcal{B}) + \mathrm{Adv}^{\mathsf{FFP\text{-}CCA}_u}_{\mathsf{PKE}_1}(\mathcal{C}) + 12q_D(q_{\mathsf{G}} + 4qD) \cdot 2^{-\frac{\gamma}{2}}.$$

The $\mathsf{FFP\text{-}CCA}_u$ game for PKE_1 is defined in Fig. 10. Adversary $\mathcal{B}$ 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 $\mathsf{eCO.Ext}$. Adversary $\mathcal{C}$ makes q_D many queries to $\textsc{Dec}$ and $\mathsf{eCO.Ext}$ and q_{G} queries to $\mathsf{eCO.RO}$. Neither $\mathcal{B}$ nor $\mathcal{C}$ query $\mathsf{eCO.Ext}$ on any of the challenge ciphertexts. The running times of $\mathcal{B}$ and $\mathcal{C}$ are bounded as $\mathrm{Time}(\mathcal{B}), \mathrm{Time}(\mathcal{C}) = \mathrm{Time}(\mathcal{A}) + O(q_D)$.

Game $\mathsf{FFP\text{-}ATK}_{\mathsf{PKE}_1,n_u}$

01 **for** $j \in [n_u]$
02 $\quad(pk_j, sk_j) \leftarrow\!\!\$\ \mathsf{Gen}()$
03 $\vec{\mathsf{pk}} = (\mathsf{pk}_1, ..., \mathsf{pk}_{n_u})$
04 $(j, m, \mathsf{salt}) \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{ATK}},\mathsf{eCO}}(\vec{\mathsf{pk}})$
05 $(c, \mathsf{salt}) \leftarrow \mathsf{PKE}_1.\mathsf{Enc}(pk_j, m)$ for given salt
06 $m' := \mathsf{PKE}_1.\mathsf{Dec}(sk_j, (c, \mathsf{salt}))$
07 **return** $[\![m' \neq m]\!]$

$\textsc{Dec}(j \in [n_u], (c, \mathsf{salt}))$
08 $m' \leftarrow \mathsf{PKE}_1.\mathsf{Dec}(sk_j, (c, \mathsf{salt}))$
09 **return** m'

Figure 10: Multi-user 'Find Failing Plaintext' games $\mathsf{FFP\text{-}ATK}_u$ for the salted PKE scheme $\mathsf{PKE}_1 := \mathsf{ST}[\mathsf{PKE}, \mathsf{G}]$, where $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, in the eQROM_f. Random oracle G is modeled as an extractable superposition oracle eCO that provides an additional extraction interface that is described in the paragraph above Eq. (7). $\mathsf{O}_{\mathsf{ATK}}$ is the additional oracle that is available in the respective $\mathsf{IND\text{-}ATK}$ game, so either the decryption oracle or none at all. To prevent confusion, we highlight our slight abuse of notation: we adapted the original $\mathsf{FFP\text{-}ATK}$ definition in a way such that $\mathcal{A}$ is allowed to preselect the salt that $\mathsf{PKE}_1.\mathsf{Enc}$ otherwise would have chosen randomly in line 05. (So in this game, we view the salt as part of the message.)

Discussion of the bound in Thm. 5. Before discussing the proof, we briefly discuss the additional disruption terms: we expect that for many real-world schemes, the additive loss relative to γ is still small enough to be neglected. For HQC and FrodoPKE, the term was calculated in [HHM22].

There are several ways to analyze the 'Find-Failing-Plaintext' ($\mathsf{FFP\text{-}CCA}$) advantage against the salted de-randomized encryption scheme PKE_1:

1. We can analyze a softer requirement: according to Thm. 10 which we prove in Sect. C, there exists an $\mathsf{FFP\text{-}CPA}_u$ attacker $\mathcal{C}'$ such that

$$\mathrm{Adv}^{\mathsf{FFP\text{-}CCA}_u}_{\mathsf{PKE}_1}(\mathcal{C}) \leq (q_D + 1) \cdot \mathrm{Adv}^{\mathsf{FFP\text{-}CPA}_u}_{\mathsf{PKE}_1}(\mathcal{C}') + 12 \cdot q_D(q_{\mathsf{G}} + 4q_D) \cdot 2^{-\frac{\gamma}{2}} \ ,$$

   so it is sufficient to analyze the 'Find-Failing-Plaintext' property of PKE_1 for passive attackers (see Fig. 10). The additional γ-term in the resulting bound occurs only due to our modular proof (combining Thm. 5 with Thm. 10) and can be avoided with a direct proof that immediately reduces to $\mathsf{FFP\text{-}CPA}$.

2. According to Thm. 11 which we also prove in Sect. C, if PKE is δ-worst-case correct, then we can alternatively bound

$$\mathrm{Adv}^{\mathsf{FFP\text{-}CCA}_u}_{\mathsf{PKE}_1}(\mathcal{C}) \leq 10(q + q_D + 1)^2 \cdot \delta(n_u) \ .$$

   We note that the statistical term δ in practice is being estimated via heuristics (as discussed in a footnote in the introduction of [HHM22].

Summary of the proof of Thm. 5. The main idea is to simulate the decapsulation oracle without using the secret key, drawing some inspiration from the proofs of single-instance $\mathsf{IND\text{-}CCA}$ security of the standard FO transform given in [DFMS22] and [HHM22, Theorem 4]. We have to adapt in two ways: first, address multi-instance security instead of single-instance security, and second, adapt to the salted FO transform.

The simulations do not have to use the secret key because we use the extractable QROM formalism described in Sect. 5.1: they have access to the additional extraction interface $\mathsf{eCO.Ext}$, and intuitively, $\mathsf{eCO.Ext}$ allows them to connect queried ciphertexts to their plaintexts, assuming that the plaintext can be found in the oracle database. The

method of simulating introduces two disruption terms: one reflects the simulation going wrong because the ciphertext does not decrypt to its originating plaintext ($\mathsf{FFP\text{-}CCA}_u$). The term related to γ-spreadness reflects that the originating plaintext can not always be found in the oracle database and that using $\mathsf{eCO.Ext}$ inflicts errors on the oracle database. For the sake of completeness, we formally prove Thm. 5 in Sect. C.

Remark 4. To show that one can simulate an implicit rejection oracle in the QROM, we invoke [BHH+19, Theorem 3], with a slight syntactical change (see page 5 for a visual representation). The reasoning used in [BHH+19, Theorem 3], to simulate an implicit rejection oracle $\mathsf{Decaps}^{\not\perp}$, given an explicit rejection $\mathsf{Decaps}^{\perp}$, one handles a ciphertext c, which rejects, by sampling a uniformly random rejection seed s, and returning $\mathsf{H}(s, c)$. With our inclusion of a salt, we should instead say that one handles a ciphertext/salt pair $c\|\mathsf{salt}$, which rejects, by sampling a uniformly random rejection seed s, and returning $\mathsf{H}(s, c\|\mathsf{salt})$.

5.3 From IND-CPA-PKE to IND-CPA-KEM in the QROM

We now revisit the IND_{n_c,n_u}-CPA PKE to IND_{n_c,n_u}-CPA KEM result, Thm. 3, and lift it to the QROM.

Theorem 6 (PKE IND_{n_c,n_u}-CPA $\overset{\text{QROM}}{\Longrightarrow}$ $\mathsf{SFO}[\mathsf{PKE}, \mathsf{len}_{\mathsf{salt}}]$ IND_{n_c,n_u}-CPA). *Let* PKE *be a public key encryption scheme, and let* KEM *be the KEM constructed as* $\mathsf{KEM} := \mathsf{SFO}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$. *If* PKE *is* $\delta(n_u)$*-correct, then* KEM *is* $\delta(n_u)$*-correct in the random oracle model. Let the number of queries to* $\textsc{Chall}_j$ *at index* $j \in [n_u]$ *be bounded by* n_c *and let* $\mathfrak{L}_{S_j}$ *represent the set of salt values sampled during those queries. Consider a* IND_{n_c,n_u}-CPA *adversary* $\mathcal{B}$ *against* KEM*, issuing at most* q_H *and* q_G *many queries in total to* H *and* G *respectively, with a total depth of* d*. Suppose that no* salt *appears in any* $\mathfrak{L}_{M_j}$ *more than* t *times.*

Furthermore, assume that $\mathcal{B}$ *does not query* $\mathsf{eCO.Ext}$ *on any of its challenge ciphertexts. Then there exists a quantum* IND_{n_c,n_u}-CPA *adversary* $\mathcal{A}$ *against* PKE *such that*

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-CPA}}_{\mathsf{KEM}}(\mathcal{B}) \leq 4\sqrt{d \cdot \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-CPA}}_{\mathsf{PKE}}(\mathcal{A})} + \frac{n_u n_c (n_c - 1)}{|\mathcal{M}| \cdot 2^{\mathsf{len}_{\mathsf{salt}}}} + 4\sqrt{\frac{dt(q_H + q_G)}{|\mathcal{M}|}} \ ,$$

Furthermore $\mathcal{A}$ *and* $\mathcal{B}$ *have similar run-time.*

In spirit, the proof proceeds exactly like its classical counterpart: we replace the proper challenge KEM keys $\mathbf{ss}_0$ and the encryption randomness with random, argue that this can only be noticed via a reasonably informative random oracle query, which we make harder to form by replacing the key seeds in question with random (there utilizing IND_{n_c,n_u}-CPA of PKE). The only difference is that the oracle queries now are in superposition and that the respective search bound thus becomes a quantum search bound. To show this bound, we use one-way to hiding - the respective reduction will measure a random oracle query and use the set of results to solve its IND_{n_c,n_u}-CPA game.

Proof. We consider the same sequence of games as in the proof of Thm. 3, except that in this proof, the random oracles in game 2 do not abort. (Since the RO-query-related event QUERY no longer is a well-defined event if the ROs are accessible in superposition.) For convenience, games 1 to 3 are repeated in Fig. 11. Since the reasoning underlying the bounds does not change for quantum attackers up to how we bound QUERY, the bound

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-CCA}}_{\mathsf{KEM}}(\mathcal{B}) \leq \frac{n_u n_c (n_c - 1)}{|\mathcal{M}| \cdot 2^{\mathsf{len}_{\mathsf{salt}}}} + |\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]|$$

still holds and it only remains to bound $|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]|$ in the QROM, which we will do with a quantum counterpart to bounding QUERY.

Games $\mathbf{G}_{1'}$ and $\mathbf{G}_{2'}$: presample to prepare quantum counterpart of bounding QUERY. We will still want to construct an IND_{n_c,n_u}-CPA adversary $\mathcal{A}$ against PKE that bounds $|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]|$. This will involve OWtH. OWtH, however, cannot randomize the values adaptively as the game proceeds, but rather has to do this in advance. We thus introduce adapted games $\mathbf{G}_{1'}$ and $\mathbf{G}_{2'}$ that are exactly like $\mathbf{G}_1/\mathbf{G}_2$, except that the message-salt tuples and all associated values are defined already before adversary $\mathcal{B}$ is run. For convenience, we make this formal with pseudocode in Fig. 11. Since the time of sampling does not change $\mathcal{B}$'s view, $\mathbf{G}_{1'}$ is equivalent to $\mathbf{G}_1$ and game $\mathbf{G}_{2'}$ is equivalent to $\mathbf{G}_2$, meaning these changes do not impact the bound above, and that we can now instead bound $|\Pr[\mathbf{G}_{2'} \Rightarrow 1] - \Pr[\mathbf{G}_{1'} \Rightarrow 1]|$.

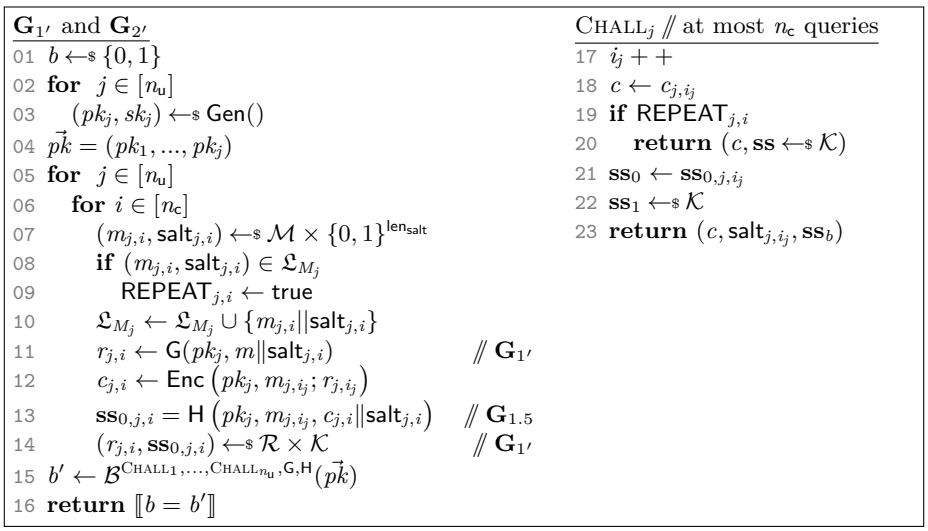

Figure 11: Games for the proof of Thm. 6 with early sampling of all values during game initialization.

Quantum counterpart of bounding QUERY. We still want to construct an IND_{n_c,n_u}-CPA adversary $\mathcal{A}$ against PKE that bounds $|\Pr[\mathbf{G}_{2'} \Rightarrow 1] - \Pr[\mathbf{G}_{1'} \Rightarrow 1]|$. As a first step, we use Thm. 4, according to which

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq 4 \cdot \sqrt{d \cdot \Pr[\mathsf{FIND} : \mathcal{B}^{\mathsf{eCO}^1 \setminus \mathcal{S}}(inp)]} \ ,$$

where $B^{\mathsf{eCO}^1 \setminus \mathcal{S}}$ denotes that $\mathcal{B}$ is run in the version $\mathbf{G}_{2'}$ that punctures G and H on the set $\mathcal{S}$ of challenge message-salt tuples in question, and FIND denotes that the punctured oracles measured a challenge message-salt tuple in $\mathcal{B}$'s oracle queries.

To further bound the right-hand side, we use that we can replace the challenge ciphertexts with encryptions of independent messages. This will make it much harder to create suitable superposition queries and thus significantly decrease the probability of FIND. In more detail, we replace the puncturing set of challenge message-salt tuples: instead of puncturing on the challenge tuples $(m_{j,i}, \mathsf{salt}_{j,i}) \in \mathfrak{L}_{M_j}$ with their respective public keys, so instead of setting $\mathcal{S} \leftarrow \mathfrak{L}_{M_j}$, we now puncture on the set $\mathcal{S}'' \leftarrow \mathfrak{L}''_{M_j}$ in which each message $m_{j,i}$ is replaced by a fresh uniform message $m''_{j,i}$. This switch can be perfectly simulated by the following IND_{n_c,n_u}-CPA adversary $\mathcal{A}$ against PKE: $\mathcal{A}$ pre-computes $\mathcal{S} = \mathfrak{L}_{M_j}$ and $\mathcal{S}'' \leftarrow \mathfrak{L}''_{M_j}$, punctures the oracles on $\mathcal{S}$, and simulates $\mathcal{B}$'s challenge oracle as follows: $\mathcal{A}$

queries their own challenge oracle to obtain an encryption of either the messages m_{j,i_j} or m''_{j,i_j} and returns the encryption together with salt_{j,i_j} and a random key. After running the extractor, it returns 1 iff FIND occurred. We thus obtain

$$\left|\Pr[\mathsf{FIND} : \mathcal{B}^{\mathsf{eCO}^1\setminus\mathcal{S}}(inp)] - \Pr[\mathsf{FIND} : \mathcal{B}^{\mathsf{eCO}^1\setminus\mathcal{S}''}(inp)]\right| \leq \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A}) \ .$$

Lastly, we bound $\Pr[\mathsf{FIND} : \mathcal{B}^{\mathsf{eCO}^1\setminus\mathcal{S}''}(inp)]$. In this experiment, the puncturing set $\mathcal{S}$" is independent of the ciphertexts used by the challenge oracle. As in the classical case, q_j to be the total number of queries to H or G which include pk_j, so that $q_\mathsf{H} + q_\mathsf{G} = \sum_j q_j$. We can thus apply Cor. 1 to conclude

$$\Pr[\mathsf{FIND} : \mathcal{B}^{\mathsf{eCO}^1\setminus\mathcal{S}''}(inp)] \leq \frac{t\cdot(q_\mathsf{H}+q_\mathsf{G})}{|\mathcal{M}|}$$

Adding up all terms gives the desired result. □

6 Discussion

There are two prominent KEMs that incorporate salting into ciphertext, and symmetric key derivation, to mitigate multi-target attacks: HQC and FrodoKEM. Our work identifies these KEMs as the application of the salted Fujisaki–Okamoto transform to an underlying PKE. Our theorems can be used to justify the multi-target security of HQC and FrodoKEM, and may inform parameter selection. We now briefly consider the concrete impact on security level. We also argue that for other KEMs, in particular ML-KEM, there is good reason to consider swapping out the generic FO transform for its salted counterpart, especially when setting parameters against quantum adversaries.

6.1 Concrete Parameters

We can consider concrete parameters to see how much security can be recovered by using the SFO transform. For brevity, we consider only the implicit rejection variant, since the security bounds closely match the explicit rejection variant. We consider parameters motivated by FrodoKEM: $|\mathcal{M}| = 2^{128}$, $\mathsf{len}_\mathsf{salt} = 256$, $n_\mathsf{c} = 2^{64}$, $n_\mathsf{u} = 2^{32}$. Recall that t is a bound on the total number of times the same salt is used during the sampling of challenges for a single public key. For these parameters we find that is $t \leq 4$ with overwhelming probability (see Sect. D).

Comparison to hybrid bound. Combining the basic hybrid argument of [BBM00] with the bounds in [HHK17] yields the following simplified bounds:

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}^\not\perp} \leq n_\mathsf{u}\cdot n_\mathsf{c}\cdot\left(2\mathrm{Adv}^{\mathsf{IND}\text{-}\mathsf{CPA}}_{\mathsf{PKE}} + q_\mathsf{RO}\cdot\delta + \frac{3q_\mathsf{RO}}{|\mathcal{M}|}\right) \tag{4}$$

where q_RO is the number of random oracle queries (across all oracles). In contrast, our analysis using the SFO transform yields:

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}^\not\perp} \leq \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}} + \frac{n_\mathsf{u}n_\mathsf{c}(n_\mathsf{c}-1)}{|\mathcal{M}|2^{\mathsf{len}_\mathsf{salt}}} + \frac{(2t+1)q_\mathsf{RO}}{|\mathcal{M}|} + q_\mathsf{RO}\cdot\delta(n_\mathsf{u}) \ . \tag{5}$$

Applying the hybrid bound (4) to the concrete parameters in this section yields at most 31 bits of multi-target security. Apply our SFO bound (5) yields

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}^\not\perp} \leq \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}} + 2^{-224} + \frac{9q_\mathsf{RO}}{2^{128}} + q_\mathsf{RO}\cdot\delta(2^{32}) \ .$$

Regarding correctness errors: the trivial bound on multi-user correctness is $\delta \leq \delta(n_{\mathsf{u}}) \leq n_{\mathsf{u}} \cdot \delta$, which is of course, much smaller than the term $n_{\mathsf{u}} \cdot n_{\mathsf{c}} \cdot \delta$ included in the hybrid bound. There is also strong evidence that for lattice-based schemes, $\delta(n_{\mathsf{u}}) < n_{\mathsf{u}}\delta(1)$ as reasoned in [DHK+21]. The exact value depends on the PKE itself, and is beyond the scope of this work.

Comparison to the generic FO transform. Duman et al. [DHK+21] give a bound for $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CCA security for FO-based KEMs. Adapted to our notation and security model (where n_{c} is a restriction on challenge ciphertexts-per-public-key and not globally), this bound becomes

$$\mathrm{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-CCA}}_{\mathsf{KEM}^{\not\perp}} \leq 2 \cdot \mathrm{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-CPA}}_{\mathsf{PKE}} + \frac{n_{\mathsf{u}} n_{\mathsf{c}}^2 + (2n_{\mathsf{c}}+1) \cdot q_{\mathsf{RO}}}{|\mathcal{M}|} + q_{\mathsf{RO}} \cdot \delta(n_{\mathsf{u}}) \ . \tag{6}$$

For our explicit parameters, the advantage in (6) is dominated by the term

$$\frac{n_{\mathsf{u}} n_{\mathsf{c}}^2 + (2n_{\mathsf{c}}+1) \cdot q_{\mathsf{RO}}}{|\mathcal{M}|} \approx 1.$$

By comparing this with the second and third terms in our bound (5), we see that salting causes the terms reflecting collision attacks go from dominant to near negligible, and the $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CCA security of the KEM closely matches the $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CPA security of the PKE.

6.2 Is the SFO transform relevant for ML-KEM?

Our previous analysis related to parameters motivated by FrodoKEM. ML-KEM uses 256-bit messages, so it would seem that the collision-finding problem is sufficiently hard even without salting. However, there are two caveats:

- **Targeting 256 bits of security:** ML-KEM has 3 variants, targeting 128, 192, or 256 bits of security, but all of these use 256-bit messages [Nat24]. We note that the parameter n_{c} should not change for the various targeted security levels, as it represents a threshold on the amount of times the same public key will be used for encapsulation in practical deployments. We observe that, if one targets 256 bits of security, collision finding attacks become relevant again, since an adversary generating 2^{128} ciphertexts will find a collision with one of 2^{64} challenge ciphertexts with probability about 2^{-64}. Thus without salting, even ML-KEM-1024 can only achieve 192 bits of multi-target security in the random oracle model.
- **The quantum random oracle model:** in the quantum random oracle, existing security bounds have a quadratic scaling of the bounds associated with collision finding attacks. Whether this scaling is representative of real attacks, or is simply an artifact of existing proofs is not a question we aim to answer in this work. If we take the existing bounds at face value when selecting parameters, even ignoring query depth (which amplifies the advantage), we have a term $4\sqrt{\frac{t q_{\mathsf{RO}}}{|\mathcal{M}|}}$ with salting, or $4\sqrt{\frac{n_{\mathsf{c}} q_{\mathsf{RO}}}{|\mathcal{M}|}}$ without salting. Recall that, for practical choices of $\mathsf{len}_{\mathsf{salt}}$, t is around 4, while n_{c} may be as large as 2^{64}. For, say $q_{\mathsf{RO}} = 2^{40}$, the associated term is 2^{-74} without salting, and 2^{-105} with salting.

6.3 Related and Future work

Our analysis shows that, for a KEM obtained from applying the salted Fujisaki–Okamoto transform to a PKE, the multi-target IND-CCA security level of the KEM closely matches the multi-target IND-CPA security of the PKE. For practical schemes like FrodoKEM-640 and HQC128, our work shows that one can generically recover 62 bits of security in

the multi-target attack setting. However, we do not bound the multi-target IND-CPA security of existing PKEs which underlie prominent KEMs such as HQC, FrodoKEM, and ML-KEM. There are also works that consider the multi-target security degradation of particular schemes, particularly at the PKE level, using cryptanalytic techniques specific to the underlying hard-problem. These include [MD26, Sen11, Ber22].

Acknowledgments

The authors thank Sam Jaques and David Jao for providing helpful comments on the thesis project from which this paper was adapted. We thank Bertram Poettering for a discussion that inspired the paragraph on ML-KEM in our concluding section. Finally, we thank the FrodoKEM team for discussions on the salted FO transform, and Chris Peikert in particular for feedback on earlier versions of proofs.

7 Bibliography

References

[AAB+22] Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jerome Lacan, Jean-Marc Robert, and Pascal Veron. HQC. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-4-submissions.

[ABD+23a] Erdem Alkim, Joppe W. Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. Annex on FrodoKEM updates, 2023. https://frodokem.org/files/FrodoKEM-annex-20230418.pdf. URL: https://frodokem.org/files/FrodoKEM-annex-20230418.pdf.

[ABD+23b] Erdem Alkim, Joppe W. Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM, learning with errors encapsulation preliminary standardization proposal. International Organization for Standardization, 2023. https://frodokem.org/files/FrodoKEM-standard_proposal-20230314.pdf. URL: https://frodokem.org/files/FrodoKEM-standard_proposal-20230314.pdf.

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 269–295. Springer, Cham, August 2019. doi:10.1007/978-3-030-26951-7_10.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, Berlin, Heidelberg, May 2000. doi:10.1007/3-540-45539-6_18.

[Ber22] Daniel J. Bernstein. Multi-ciphertext security degradation for lattices. Cryptology ePrint Archive, Report 2022/1580, 2022. URL: https://eprint.iacr.org/2022/1580.

[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, Part II*, volume 11892 of *LNCS*, pages 61–90. Springer, Cham, December 2019. doi:10.1007/978-3-030-36033-7_3.

[BS20] Nina Bindel and John M. Schanck. Decryption failure is more likely after success. In Jintai Ding and Jean-Pierre Tillich, editors, *PQCrypto 2020*, pages 206–225. Springer, Cham, 2020. doi:10.1007/978-3-030-44223-1_12.

[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 *LNCS*, pages 133–151. Springer, Berlin, Heidelberg, December 2003. doi:10.1007/978-3-540-40974-8_12.

[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, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 677–706. Springer, Cham, May / June 2022. doi:10.1007/978-3-031-07082-2_24.

[DGJ+19] Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede. Decryption failure attacks on IND-CCA secure lattice-based schemes. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 565–598. Springer, Cham, April 2019. doi:10.1007/978-3-030-17259-6_19.

[DHK+21] Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, and Gregor Seiler. Faster lattice-based KEMs via a generic Fujisaki-Okamoto transform using prefix hashing. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2722–2737. ACM Press, November 2021. doi:10.1145/3460120.3484819.

[DRV20] Jan-Pieter D'Anvers, Mélissa Rossi, and Fernando Virdia. (One) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 3–33. Springer, Cham, May 2020. doi:10.1007/978-3-030-45727-3_1.

[FKK+22] Michael Fahr, Hunter Kippen, Andrew Kwong, Thinh Dang, Jacob Lichtinger, Dana Dachman-Soled, Daniel Genkin, Alexander Nelson, Ray A. Perlner, Arkady Yerukhimovich, and Daniel Apon. When frodo flips: End-to-end key recovery on FrodoKEM via rowhammer. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 979–993. ACM Press, November 2022. doi:10.1145/3548606.3560673.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Berlin, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_34.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013. doi:10.1007/s00145-011-9114-1.

[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, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Cham, November 2017. doi:10.1007/978-3-319-70500-2_12.

[HHM22] Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Failing gracefully: Decryption failures and the Fujisaki-Okamoto transform. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 414–443. Springer, Cham, December 2022. doi:10.1007/978-3-031-22972-5_15.

[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, Part II*, volume 12111 of *LNCS*, pages 389–422. Springer, Cham, May 2020. doi:10.1007/978-3-030-45388-6_14.

[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, *PQCrypto 2024, Part II*, pages 245–265. Springer, Cham, June 2024. doi:10.1007/978-3-031-62746-0_11.

[Höv21] Kathrin Hövelmanns. *Generic constructions of quantum-resistant cryptosystems*. doctoralthesis, Ruhr-Universität Bochum, Universitätsbibliothek, 2021. doi:10.13154/294-7758.

[HS21] Hans Heum and Martijn Stam. Tightness subtleties for multi-user PKE notions. In Maura B. Paterson, editor, *18th IMA International Conference on Cryptography and Coding*, volume 13129 of *LNCS*, pages 75–104. Springer, Cham, December 2021. doi:10.1007/978-3-030-92641-0_5.

[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, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 96–125. Springer, Cham, August 2018. doi:10.1007/978-3-319-96878-0_4.

[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, Part II*, volume 11443 of *LNCS*, pages 618–645. Springer, Cham, April 2019. doi:10.1007/978-3-030-17259-6_21.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 239–256. Springer, Berlin, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_14.

[MD26] Alexander May and Gabriel Sá Diogo. Multi-instance security degradation of code-based KEMs. Cryptology ePrint Archive, Paper 2026/517, 2026. URL: https://eprint.iacr.org/2026/517.

[NAB+20] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2020. available

at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[Nat24] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 203, U.S. Department of Commerce, Washington, D.C., 2024. doi:10.6028/NIST.FIPS.203.

[NIS21] NIST PQC Team. Multi-ciphertext attacks, August 2021. Personal communication.

[Pei25] Chris Peikert. Private communication, 2025. April–July.

[Saa25] Markku-Juhani O. Saarinen. IND-CCA2 issue in HQC (latest version). NIST PQC Forum, Google Groups, 2025. Message posted April 2, 2025. URL: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/Wiu4ZQo3fP8/m/t9UTwGO5CAAJ.

[Sen11] Nicolas Sendrier. Decoding one out of many. In Bo-Yin Yang, editor, *PQCrypto 2011*, pages 51–67. Springer, Berlin, Heidelberg, November / December 2011. doi:10.1007/978-3-642-25405-5_4.

[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, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 520–551. Springer, Cham, April / May 2018. doi:10.1007/978-3-319-78372-7_17.

A Modular ROM Bounds

In this section, we provide a modular analysis of the security of the salted Fujisaki–Okamoto transform. This modular analysis is intended to accommodate a distribution substitution step. In FrodoKEM, for example, security is reduced to an LWE problem, where errors are sampled from a discrete Gaussian distribution [ABD+23b]. At the implementation level, this discrete Gaussian will be replaced with an approximate distribution, and a robust security analysis should capture this. A way to handle this distribution substitution is to use Rényi divergence. In short

- Start with a PKE with an idealized distribution.
- Apply the (salted) T transform to yield a deterministic PKE, and prove multi-target one-way security.
- Apply the distribution substitution to the PKE, and bound the difference in advantage using Rényi divergence.
- Apply a variant of the U transform to this deterministic PKE to yield a KEM with multi-target IND-CCA security.

First, we define multi-target one way security, with a plaintext checking oracle.

Definition 10 (OW_{n_c,n_u}-PCA security of $\mathsf{ST}[\mathsf{PKE}]$)**.** Let PKE be a public key encryption scheme, and let n_u, n_c be positive integers, and let $\mathcal{A}$ be an adversary in the $\mathrm{Exp}^{\mathsf{OW}_{n_c,n_u}\text{-}\mathsf{PCA}}_{\mathsf{ST}[\mathsf{PKE}]}(\mathcal{A})$ experiment shown in Fig. 12.

The OW_{n_c,n_u}-PCA advantage of an adversary $\mathcal{A}$ against a PKE is

$$\mathrm{Adv}^{\mathsf{OW}_{n_c,n_u}\text{-}\mathsf{PCA}}_{\mathsf{PKE}}(\mathcal{A}) = \Pr[\mathrm{Exp}^{\mathsf{OW}_{n_c,n_u}\text{-}\mathsf{PCA}}_{\mathsf{PKE}}(\mathcal{A}) \Rightarrow 1]$$

$\underline{\mathrm{Exp}^{\mathsf{OW}_{n_c,n_u}\text{-}\mathsf{PCA}}_{\mathsf{ST}[\mathsf{PKE}]}(\mathcal{A})}$
24 **for** $i = 1, \ldots, n_u$
25 $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\!\$\ \mathsf{Gen}()$
26 $\vec{pk} \leftarrow (\mathsf{pk}_1, ..., \mathsf{pk}_{n_u})$
27 $(j, m') \leftarrow \mathcal{A}^{\mathsf{PCO},\textsc{Chall}}(\vec{pk})$
28 **return** $[\![m' \in \mathfrak{L}_{M_j}]\!]$
$\underline{\mathsf{PCO}(j, m \in \mathcal{M}, c \| \mathsf{salt})}$
29 **return** $[\![\mathsf{Dec}(sk_j, c\|\mathsf{salt}) = m]\!]$

$\underline{\textsc{Chall}_j \;/\!/ \max n_c \text{ queries}}$
30 $m^* \leftarrow\!\!\$\ \mathcal{M}$
31 $\mathsf{salt} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$
32 $\mathfrak{L}_{M_j} \leftarrow \mathfrak{L}_{M_j} \cup \{m^* \| \mathsf{salt}\}$
33 $c^* \leftarrow \mathsf{Enc}(pk_j, m^* \| \mathsf{salt})$
34 $\mathfrak{L}_{C_j} \leftarrow \mathfrak{L}_{C_j} \cup \{c^* \| \mathsf{salt}\}$
35 **return** $(c^* \| \mathsf{salt})$

Figure 12: Experiment for OW_{n_c,n_u}-PCA-security.

Lemma 2 (OW_{n_c,n_u}-PCA security $\implies$ IND_{n_c,n_u}-CCA security.)**.** *Let* PKE *be a public key encryption scheme, and let* n_c *be a positive integer. For any* OW_{n_c,n_u}-PCA *adversary* $\mathcal{A}$ *there exists a* IND_{n_c,n_u}-CCA *adversary* $\mathcal{B}$*, running* $\mathcal{A}$ *as a subroutine, such that*

$$\mathrm{Adv}^{\mathsf{OW}_{n_c,n_u}\text{-}\mathsf{PCA}}_{\mathsf{PKE}} \leq \frac{n_c}{|\mathcal{M}|} + 2\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{PKE}}$$

Proof. We show how an IND_{n_c,n_u}-CPA adversary $\mathcal{A}$ can use a OW_{n_c,n_u}-PCA adversary $\mathcal{B}$ as a black-box subroutine to win their own game. To simulate $\textsc{Chall}_j$ for $\mathcal{B}$, $\mathcal{A}$ samples two random messages m_0 and m_1, initializes two lists $\mathfrak{L}_{M_{j,0}}$ and $\mathfrak{L}_{M_{j,1}}$, appends m_0 to $\mathfrak{L}_{M_{j,0}}$ and m_1 to $\mathfrak{L}_{M_{j,1}}$, and finally queries their own challenge oracle to obtain $c \leftarrow \mathsf{Enc}(pk_j, m_b)$, and passes c to $\mathcal{A}$. Eventually, $\mathcal{A}$ returns (j, m') and wins if $m' \in \mathfrak{L}_{M_{j,b}}$. Thus, $\mathcal{B}$ returns 0 if $m' \in \mathfrak{L}_{M_{j,0}}$, 1 if $m' \in \mathfrak{L}_{M_{j,1}}$, and $b \leftarrow\!\!\$\ \{0,1\}$ otherwise. Observe that

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{PKE}}(\mathcal{B}) = \frac{1}{2}\left(\Pr[m' \in \mathfrak{L}_{M_{j,1}} | b = 1] - \Pr[m' \in \mathfrak{L}_{M_{j,1}} | b = 0]\right)$$

We observe that $\Pr[m' \in \mathfrak{L}_{M_{j,1}} | b = 1] = \mathrm{Adv}^{\mathsf{OW}_{n_c,n_u}\text{-}\mathsf{PCA}}_{\mathsf{PKE}}(\mathcal{B})$. Furthermore, $\mathfrak{L}_{M_{j,0}}$ is independent of the view of $\mathcal{B}$, and thus $\Pr[m' \in \mathfrak{L}_{M_{j,1}} | b = 0] \leq \frac{n_c}{|\mathcal{M}|}$. Thus

$$\mathrm{Adv}^{\mathsf{OW}_{n_c,n_u}\text{-}\mathsf{PCA}}_{\mathsf{PKE}}(\mathcal{B}) \leq 2\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{PKE}}(\mathcal{B}) + \frac{n_c}{|\mathcal{M}|}$$

as desired. □

Theorem 7 (PKE IND_{n_c,n_u}-CPA $\xRightarrow{\text{ROM}}$ $\mathsf{ST}[\mathsf{PKE}]$ OW_{n_c,n_u}-PCA)**.** *Let* PKE *be a public key encryption scheme, and let* DPKE *be the deterministic PKE constructed as* $\mathsf{DPKE} := \mathsf{ST}[\mathsf{PKE}, \mathsf{G}, \mathsf{len}_{\mathsf{salt}}]$*. If* PKE *is* $\delta(n_u)$*-correct then* DPKE *is* $\delta(n_u)$*-correct in the random oracle model. Suppose no salt appears more than* t *times in any* $\mathfrak{L}_{M_j}$*. Moreover for any* OW_{n_c,n_u}-PCA *adversary* $\mathcal{A}$ *against* DPKE*, issuing at most* q *random oracle queries there exists an* IND_{n_c,n_u}-CPA *adversary* $\mathcal{B}$ *against* PKE *such that*

$$\mathrm{Adv}^{\mathsf{OW}_{n_c,n_u}\text{-}\mathsf{PCA}}_{\mathsf{DPKE}}(\mathcal{A}) \leq q \cdot \delta(n_u) + \frac{n_c}{|\mathcal{M}|} + \frac{2tq}{|\mathcal{M}|} + 4\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B})$$

Proof. The game hops used in this result are identical to certain hops appearing either in Thm. 3 and Thm. 1, hence we provide only a proof sketch here. The goal is to show that an IND_{n_c,n_u}-CPA adversary can simulate the OW-CPA for $\mathsf{ST}[\mathsf{PKE}]$ experiment. For this, we need two game hops.

In $\mathbf{G}_1$ we modify the PCO oracle, so that it returns $[\![\mathsf{Enc}(pk_j, m\|\mathsf{salt}) = c]\!]$. The view of the $\mathcal{A}$ is identical, provided no correctness error occurs. Thus, by the same argument as in Thm. 1 we have

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq q \cdot \delta(n_\mathsf{u})$$

In $\mathbf{G}_2$ we introduce a flag QUERY and abort whenever $\mathcal{A}$ queries G on some $(pk_j, m\|\mathsf{salt}) \in \mathfrak{L}_{M_j}$. Furthermore, we replace the random coins used in encryption $\mathsf{G}(m\|\mathsf{salt})$ with a uniformly random value. We have

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \Pr[\mathsf{QUERY}]$$

Furthermore, in $\mathbf{G}_2$, the experiment is completely independent of sk, and thus can be simulated by a $\mathsf{OW}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ adversary $\mathcal{A}'$. By Lem. 2 we have $\mathrm{Adv}^{\mathsf{OW}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{PCA}}_{\mathsf{PKE}}(\mathcal{A}') \leq 2\mathrm{Adv}^{\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A}') + \frac{n_\mathsf{c}}{|M|}$. Finally, we bound QUERY by a $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ adversary $\mathcal{B}'$, as in Thm. 3. $\mathcal{B}$ creates two lists, $\mathfrak{L}_{M_{j,0}}$ and $\mathfrak{L}_{M_{j,1}}$, and simulates $\textsc{Chall}_j$ by sampling $(m_0, m_1) \leftarrow\!\!\$\ \mathcal{M}^2$, samples a uniformly random salt, and gets $c \leftarrow \mathsf{Enc}(pk_j, m_b)$ from their own challenge oracle, and returns $c\|\mathsf{salt}$. Finally, $\mathcal{B}'$ checks if $\mathcal{A}$ ever makes a $(pk_j, m\|\mathsf{salt}) \in \mathfrak{L}_{M_{j,b}}$. This event is exactly QUERY. Furthermore, we define an event $\mathsf{COLLISION}$, to be that $\mathcal{A}$ queries $(pk_j, m\|\mathsf{salt}) \in \mathfrak{L}_{M_{j,1-b}}$. By the same argument as in Thm. 3, we have that

$$\Pr[\mathsf{QUERY}] \leq 2\mathrm{Adv}^{\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B}') + \frac{2tq}{|\mathcal{M}|}$$

Combining $\mathcal{A}'$ and $\mathcal{B}'$ into a single $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ adversary $\mathcal{B}$ yields the desired result. □

Theorem 8 ($\mathsf{ST}[\mathsf{PKE}, \mathsf{G}, \mathsf{len}_\mathsf{salt}]$ $\mathsf{OW}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{PCA} \xrightarrow{\text{ROM}}$ KEM $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$)**.** *Let* PKE *be a public key encryption scheme, and let* DPKE *be the PKE constructed as* $\mathsf{DPKE} := \mathsf{ST}[\mathsf{PKE}, \mathsf{G}, \mathsf{len}_\mathsf{salt}]$ *and* KEM *be the KEM constructed as* $\mathsf{KEM} := \mathsf{SFO}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_\mathsf{salt}]$*. If* PKE *is* $\delta(n_\mathsf{u})$*-correct, then* KEM *is* $\delta(n_\mathsf{u})$*-correct in the random oracle model. Moreover for any* $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$ *adversary* $\mathcal{B}$ *against* KEM*, issuing at most* q *random oracle queries there exists an* $\mathsf{OW}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{PCA}$ *adversary* $\mathcal{A}$ *against* DPKE *such that*

$$\mathrm{Adv}^{\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{B}) \leq \frac{n_\mathsf{u} n_\mathsf{c}(n_\mathsf{c}-1)}{|\mathcal{M}|2^{\mathsf{len}_\mathsf{salt}}} + \frac{q}{|\mathcal{M}|} + \mathrm{Adv}^{\mathsf{OW}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{PCA}}_{\mathsf{DPKE}}(\mathcal{A}) + \delta(n_\mathsf{u})$$

Proof. First, we observe that in the $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$ experiment, if the challenge oracle samples the same message/salt pair twice for a given user, then if $b = 0$, c and $\mathbf{ss}_b$ would match as well, but if $b = 1$, c would match, but $\mathbf{ss}_b$ would be independent of b, leading to a trivial win. Thus, we make a game hop, sampling $\mathbf{ss}_0$ independently whenever $\textsc{Chall}_j$ samples $(m\|\mathsf{salt}) \in \mathfrak{L}_{M_j}$. The probability that a message/salt pair repeats, for any $j \in [n_\mathsf{u}]$ is bounded by

$$\sum_{j \in [n_\mathsf{u}]} \frac{n_c^2}{|\mathcal{M}|} = \frac{n_\mathsf{u} n_\mathsf{c}^2}{|\mathcal{M}|2^{\mathsf{len}_\mathsf{salt}}}$$

In $\mathbf{G}_2$ we patch H and Decaps so that they always return the same value, as in [HHK17, Theorem 3.4]. By the same argument as in [HHK17] we have that

$$\Pr[\mathbf{G}_2 \Rightarrow 1] = \Pr[\mathbf{G}_1 \Rightarrow 1]$$

In $\mathbf{G}_3$ we abort whenever the adversary queries H on pk_j and s_j. Furthermore, we make the output of H perfectly random, when queried on pk_j and s_j, by making use of an internal random oracle H'. The view of the adversary is indistinguishable from $\mathbf{G}_1$, unless the adversary queries $\mathsf{H}(pk_j, s_j, c\|\mathsf{salt})$, since the adversary makes at most q total queries to H we have

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \frac{q}{|\mathcal{M}|}$$

Finally, in $\mathbf{G}_4$ we raise a flag CHAL and abort, if $\mathcal{B}$ ever queries $\mathsf{H}(pk_j, m, c\|\mathsf{salt})$ such that $\mathsf{Dec}(sk_j, c\|\mathsf{salt}) = m$ and $c \in \mathfrak{L}_{C_j}$. Furthermore, we sample $\mathbf{ss}_0$ uniformly at random, so that the view of $\mathcal{A}$ is independent of b and

$$\Pr[\mathbf{G}_4 \Rightarrow 1] = \frac{1}{2}$$

Furthermore, unless CHAL occurs, the view of $\mathcal{A}$ is identical in either game. Thus

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \Pr[\mathsf{CHAL}]$$

To bound $\Pr[\mathsf{CHAL}]$ we consider an $\mathsf{OW}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{PCA}$ adversary $\mathcal{B}$ that wins whenever CHAL is raised. To simulate the $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$ experiment for $\mathcal{A}$ $\mathcal{B}$ responds the $\textsc{Chall}_j$, by sampling random message/salt pairs $(m\|\mathsf{salt})$, and random session key $\mathbf{ss}$, and passing $(\mathsf{Enc}_1(pk_j, m\|\mathsf{salt})\|\mathsf{salt}, \mathbf{ss})$. The decapsulation oracle has been modified to no longer make use of the secret key, so the simulation is perfect (see [HHK17, Theorem 3.4]). $\mathcal{B}$ wins their own game by checking if there exists a query $(pk_j, m', c\|salt)$ with $\mathsf{PCO}(m', c\|\mathsf{salt}) = 1$, and $c \in \mathfrak{L}_{C_j}$ and returning (j, m') if this is the case, and aborts otherwise. In the event that there are multiple such queries, $\mathcal{B}$ simply returns the first one (i.e. smallest such j, and the first such m at that index). Thus $\mathcal{B}$ returns some (j, m') whenever CHAL occurs, and wins their game, as long as m' does not exhibit a correctness error. Thus we have

$$\begin{aligned}\Pr[\mathsf{CHAL}] &\leq \mathrm{Adv}_{\mathsf{ST[PKE]}}(\mathcal{B}) + \Pr[\mathsf{PCO}(j, m', c) = 1 | \mathsf{Enc}(pk_j, m') \neq c] \\ &\leq \mathrm{Adv}_{\mathsf{ST[PKE]}}(\mathcal{B}) + \delta(n_\mathsf{u}).\end{aligned}$$

Thus we have

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \mathrm{Adv}^{\mathsf{OW}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{PCA}}_{\mathsf{ST[PKE]}}(\mathcal{B}) + \delta(n_\mathsf{u}).$$

Adding up all inequalities gives the desired result. □

Finally, we consider how distribution substitution impacts the total security bound of a (salted) KEM.

Definition 11 (Rényi divergence)**.** The Rényi divergence of positive order $\alpha \neq 1$ of a discrete distribution P from a distribution Q is defined as

$$\mathrm{D}_\alpha(P\|Q) = \frac{1}{\alpha - 1} \ln \left(\sum_{x \in \mathrm{supp}\, P} P(x) \left(\frac{P(x)}{Q(x)} \right)^{\alpha - 1} \right) .$$

The probability preservation property of Rényi divergence [LSS14] gives

$$\mathrm{Adv}^{\mathsf{OW}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{PCA}}_{\mathsf{PKE}_P}(\mathcal{A}) \leq \left(\mathrm{Adv}^{\mathsf{OW}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{PCA}}_{\mathsf{PKE}_Q}(\mathcal{A}) \cdot \exp(s \cdot \mathrm{D}_\alpha(P\|Q)) \right)^{1-1/\alpha}$$

Given a concrete distribution substitution, one can compute the Rényi divergence between the two distributions, and apply the probability preservation property to the bound of Thm. 7, then compose that result with the bound from Thm. 8, to get a complete security bound for the KEM.

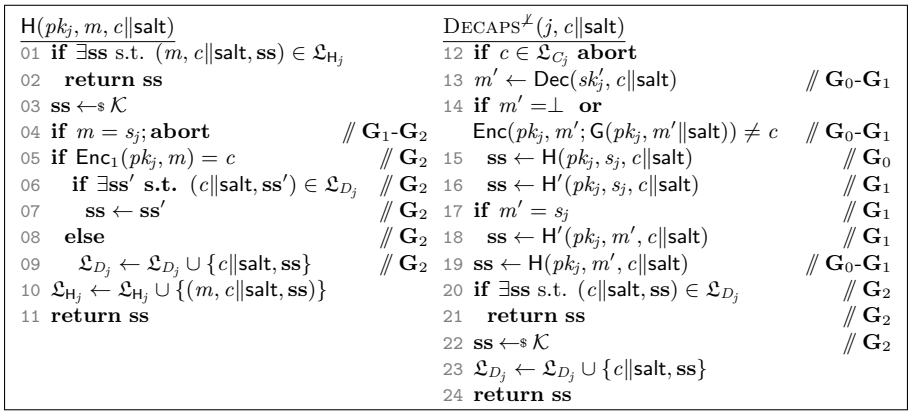

Figure 13: Simulation of a decapsulation oracle with implicit rejection for proof of Thm. 1.

B Simulating Decapsulation Oracles in the Random Oracle Model

In the following theorems, we use known techniques to simulate decapsulation oracles with either implicit or explicit rejection. Intuitively, we apply a series of modifications to the decapsulation oracle, which remove dependence on the secret key, and bound the changes that occur at each step. We then apply a patching technique to ensure that the outputs of H, the random oracle used for producing KEM keys, will match the output of Decaps.

Proof of Thm. 1

Proof. Let G be a random oracle and $\textsc{Chall}_j$ be defined as in Fig. 5. Furthermore, let $\mathfrak{L}_{C_j}$ denote the list of challenge ciphertexts for a user j. Consider the sequence of games shown in Fig. 13.
$\mathbf{G}_0$ is the IND_{n_c,n_u}-CCA-game against $\mathsf{KEM}^{\not\perp}$ and so

$$|\Pr[\mathbf{G}_0 \Rightarrow 1]| = \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}^{\not\perp}}(\mathcal{B})$$

In $\mathbf{G}_1$ we abort whenever H is queried on s_j, and we make the outputs of H perfectly random when an input is rejected by making use of a separate internal random oracle H'. $\mathbf{G}_1$ and $\mathbf{G}_0$ are identical unless the adversary queries H on s_j. Thus

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq \frac{q_{\mathsf{H}}}{|\mathcal{M}|}.$$

In $\mathbf{G}_2$, we remove dependence on the secret key and claim that the oracles H and $\textsc{Decaps}^{\not\perp}$ are patched so that, assuming there is no correctness error, $\mathbf{G}_2$ and $\mathbf{G}_1$ are identical. Hence, if CORR is the event that $\mathcal{B}$ makes a ROM query, with a j and $m\|\mathsf{salt}$ inducing a correctness error. By $\delta(n_u)$-correctness, we have that

$$\Pr[\mathsf{CORR}] \leq (q_{\mathsf{H}} + q_D) \cdot \delta(n_u).$$

Now we show that $\mathbf{G}_2$ and $\mathbf{G}_1$ proceed identically, conditioned on $\neg\mathsf{CORR}$.

Consider a query $\textsc{Decaps}^{\not\perp}(j, c\|\mathsf{salt})$ where $c = \mathsf{Enc}(pk_j, m; \mathsf{G}(pk_j, m\|\mathsf{salt}))$. Let $m' = \mathsf{Dec}(sk_j, c\|\mathsf{salt})$ and $c' = \mathsf{Enc}(pk_j, m'; \mathsf{G}(pk_j, m'\|\mathsf{salt}))$.

- **Case 1:** $m' = \perp$, then in both $\mathbf{G}_2$ and $\mathbf{G}_1$ $\textsc{Decaps}^{\not\perp}$ outputs a uniformly random $\mathbf{ss}$, and H cannot be queried on m.

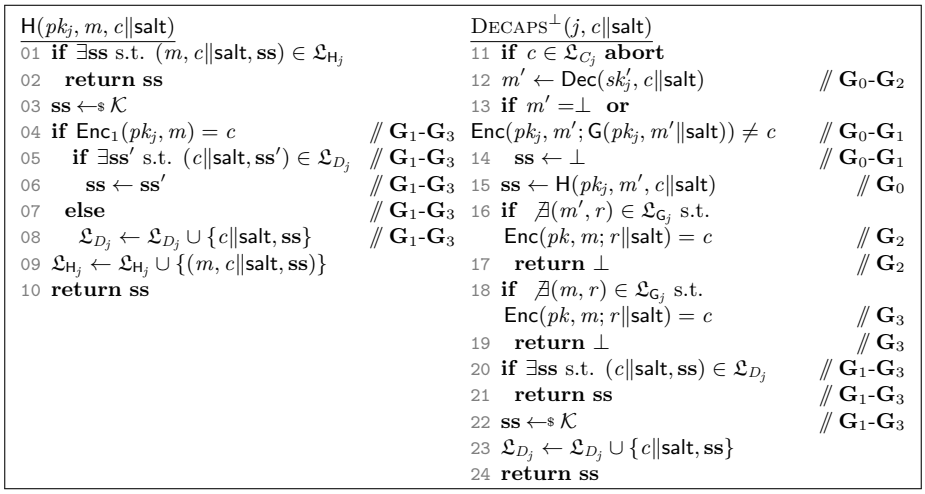

Figure 14: Simulation of a decapsulation oracle with explicit rejection for proof of Thm. 2.

- **Case 2:** $m \neq \perp$ and $c \neq c'$. In this case, $\textsc{Decaps}^{\not\perp}$ will output a uniformly random secret in either game. However, in $\mathbf{G}_2$ a query $\mathsf{H}(pk_j, m, c\|\mathsf{salt})$ would return the same value as $\textsc{Decaps}^{\not\perp}(j, c\|\mathsf{salt})$, where in $\mathbf{G}_1$ $\textsc{Decaps}^{\not\perp}(j, c\|\mathsf{salt})$ would output $\mathsf{H}'(pk_j, c\|\mathsf{salt})$, while $\mathsf{H}(pk_j, m, c\|\mathsf{salt})$ would output an independent random value. This can only happen if m exhibits a correctness error.
- **Case 3:** $m \neq \perp$ and $c = c'$. In $\mathbf{G}_1$ the output of $\textsc{Decaps}^{\not\perp}$ is $\mathsf{H}(pk_j, m', c\|\mathsf{salt})$. In $\mathbf{G}_2$ the output of $\textsc{Decaps}^{\not\perp}(j, c\|\mathsf{salt})$ is $\mathsf{H}(pk_j, m, c\|\mathsf{salt})$ if $\mathsf{H}(pk_j, m, c\|\mathsf{salt})$. Hence the output of the two games will only be different if $m \neq m'$.

By the difference lemma

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq (q_{\mathsf{H}} + q_D) \cdot \delta(n_{\mathsf{u}}).$$

Now, observe that in $\mathbf{G}_2$ $\textsc{Decaps}^{\not\perp}$ has no dependence on sk, and can therefore be simulated by an $\mathsf{IND}_{n_{\mathsf{c}}, n_{\mathsf{u}}}$-$\mathsf{CPA}$ adversary $\mathcal{A}$. □

Proof of Thm. 2

Proof. Let G be a random oracle $\textsc{Chall}$ be defined as Fig. 5. Furthermore, let $\mathfrak{L}_{C_j}$ denote the list of challenge ciphertexts for a user j. Consider the sequence of games shown in Fig. 14

$\mathbf{G}_0$ is the $\mathsf{IND}_{n_{\mathsf{c}}, n_{\mathsf{u}}}$-$\mathsf{CCA}$-game against $\mathsf{KEM}^{\perp}$ and so

$$|\Pr[\mathbf{G}_0 \Rightarrow 1]| = \mathrm{Adv}^{\mathsf{IND}_{n_{\mathsf{c}}, n_{\mathsf{u}}}\text{-}\mathsf{CCA}}_{\mathsf{KEM}^{\perp}}(\mathcal{B})$$

In $\mathbf{G}_1$ we patch the random oracles so that Decaps and H always return the same thing. Even in the event of a correctness error, the view of the adversary is identical, since $\textsc{Decaps}^{\perp}$ will return $\perp$, as in $\mathbf{G}_2$

$$\Pr[\mathbf{G}_1 \Rightarrow 1] = \Pr[\mathbf{G}_0 \Rightarrow 1]$$

In $\mathbf{G}_2$, we modify $\textsc{Decaps}^{\perp}$, so that instead of checking that $m' \in \mathcal{M}$, it instead checks that $\mathsf{G}(m'\|\mathsf{salt})$ had been queried, and then $\mathsf{Enc}(pk_j, m'; \mathsf{G}(m'\|\mathsf{salt})) = c'$. The two games proceed identically, unless $\mathcal{B}$ was able to produce a ciphertext c such that

$c = \mathsf{Enc}(pk_j, m'; \mathsf{G}(m'))$ without querying $\mathsf{G}(m')$. In this case in $\mathbf{G}_2$ $\textsc{Decaps}^{\perp}(j, c\|\mathsf{salt})$ would return $\perp$, while in $\mathbf{G}_1$ $\textsc{Decaps}^{\perp}(\mathbf{ss}, c\|\mathsf{salt})$ would return $\mathsf{H}(pk_j, m', c\|\mathsf{salt})$.

This implies $\mathcal{B}$ found $r \neq \mathsf{G}(m'\|\mathsf{salt})$ such that $\mathsf{Enc}(pk, m; \mathsf{G}(m'\|\mathsf{salt})) = \mathsf{Enc}(pk, m; r)$. For a single decapsulation query, this occurs with probability $2^{-\gamma}$, by the assumption that PKE is γ-spread. For q_D decapsulation queries, we have that

$$\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1] \leq q_D \cdot 2^{-\gamma}$$

In $\mathbf{G}_3$ we remove all dependence on the secret key, by no longer checking decryption c during decapsulation. Instead, we check for queries to $\mathsf{G}(pk_j, -)$ which encrypt to c. This change goes unnoticed, unless the adversary was able to query a message and a ciphertext which exhibit a correctness error. Following the same argument as for $\mathbf{G}_2$ theorem Thm. 1, we have

$$\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1] \leq (q_{\mathsf{H}} + q_D) \cdot \delta(n_{\mathsf{u}})$$

Now, observe that in $\mathbf{G}_3$ $\textsc{Decaps}^{\perp}$ has no dependence on sk_j, and can therefore be simulated by an $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-$\mathsf{CPA}$ adversary $\mathcal{A}$. □

C From $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CPA KEM to $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CCA KEM in the QROM

In this section, we prove Thm. 5, our lift of Thm. 2 to the quantum-accessible random oracle model. We first repeat the theorem statement for convenience.

Theorem 9 (Thm. 5: Simulatability of salted $\textsc{Decaps}^{\perp}$ in the QROM)**.** *Let* PKE *be a public key encryption scheme,* $\mathsf{PKE}_1 := \mathsf{ST}[\mathsf{PKE}, \mathsf{G}]$*, and* $\mathsf{KEM}^{\perp} = \mathsf{SFO}^{\perp}_{m,c}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$*. Let* $\mathcal{A}$ *be a* $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-$\mathsf{CCA}$ *adversary against* $\mathsf{KEM}^{\perp}$*, issuing at most* q_{G} *many queries to* G*,* q_D *queries to* $\textsc{Decaps}^{\not\perp}$*, and with* d *and* w *being the combined query depth/width of* $\mathcal{A}$*'s random oracle queries. Then there exist an* $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-$\mathsf{CPA}$ *adversary* $\mathcal{B}$ *against* $\mathsf{KEM}^{\not\perp}$ *and an* FFP-CCA_u *adversary* $\mathcal{C}$ *against* PKE_1 *in the extractable QROM such that*

$$\mathrm{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CCA}}_{\mathsf{KEM}^{\perp}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CPA}}_{\mathsf{KEM}^{\perp}}(\mathcal{B}) + \mathrm{Adv}^{\mathsf{FFP}\text{-}\mathsf{CCA}_u}_{\mathsf{PKE}_1}(\mathcal{C}) + 12q_D(q_{\mathsf{G}} + 4qD) \cdot 2^{-\frac{\gamma}{2}}.$$

Adversary $\mathcal{B}$ *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* $\mathsf{eCO.Ext}$*. Adversary* $\mathcal{C}$ *makes* q_D *many queries to* $\textsc{Dec}$ *and* $\mathsf{eCO.Ext}$ *and* q_{G} *queries to* $\mathsf{eCO.RO}$*. Neither* $\mathcal{B}$ *nor* $\mathcal{C}$ *query* $\mathsf{eCO.Ext}$ *on any of the challenge ciphertexts. The running times of* $\mathcal{B}$ *and* $\mathcal{C}$ *are bounded as* $\mathrm{Time}(\mathcal{B}), \mathrm{Time}(\mathcal{C}) = \mathrm{Time}(\mathcal{A}) + O(q_D)$*.*

Proof. We give in Fig. 15 below two simulated variants of the decapsulation oracle, $\textsc{Decaps}'$ that will be used by the $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-$\mathsf{CPA}$ reduction $\mathcal{B}$, and a second simulation $\textsc{Decaps}''$ that will be used by the FFP-CCA reduction $\mathcal{C}$. Intuitively, $\textsc{Decaps}''$ additionally notices (and stores) plaintext that trigger a decryption failure.

The simulations extract potential plaintexts by accessing extractor interface $\mathsf{eCO.Ext}$ (see lines 12 and 18). We let this interface extract plaintexts relative to the function

$$f : ((\mathsf{pk}, m\|\mathsf{salt}), r) \mapsto (\mathsf{Enc}(\mathsf{pk}, m; r), \mathsf{pk}, \mathsf{salt}) \ , \tag{7}$$

so $\mathsf{eCO.Ext}(c, \mathsf{pk}, \mathsf{salt})$ either returns $\perp$ or a message m such that $\mathsf{Enc}(\mathsf{pk}, m; r) = c$ for $r \leftarrow \mathsf{G}(\mathsf{pk}, m\|\mathsf{salt})$.

We now prove this theorem via a sequence of games.

$\mathbf{G}_0$ is the $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-$\mathsf{CCA}$ game for $\mathsf{KEM}^{\perp}$.

$\mathbf{G}_1$ is like $\mathbf{G}_0$, except for two modifications: firstly, the quantum-accessible random oracle G is simulated using an extractable quantum random oracle eQRO_f. Secondly,

$\underline{\textsc{Decaps}(j, (c\|\mathsf{salt}) \notin \mathfrak{L}_{C_j})}$
01 $m' \leftarrow \mathsf{Dec}(sk_j, c)$
02 **if** $m' = \bot$
03 **return** $K \leftarrow \bot$
04 **else**
05 $r' \leftarrow \mathsf{G}(pk_j, m' \| \mathsf{salt})$
06 $c' \leftarrow \mathsf{Enc}(pk_j, m'; r')$
07 **if** $c \neq c'$
08 **return** $\bot$
09 **else**
10 **return** $\mathsf{H}(\mathsf{pk}, m', c\|\mathsf{salt})$

$\underline{\textsc{Decaps}'(j, (c\|\mathsf{salt}) \notin \mathfrak{L}_{C_j})}$
11 Parse $\mathsf{pk} \leftarrow pk_j$
12 $\hat{m} \leftarrow \mathsf{eCO.Ext}(c, \mathsf{pk}, \mathsf{salt})$
13 **if** $\hat{m} = \bot$
14 **return** $\bot$
15 **else**
16 **return** $\mathsf{H}(\mathsf{pk}, \hat{m}, c\|\mathsf{salt})$

$\underline{\textsc{Decaps}''(j, (c\|\mathsf{salt}) \notin \mathfrak{L}_{C_j})}$
17 Parse $\mathsf{pk} \leftarrow pk_j$
18 $\hat{m} \leftarrow \mathsf{eCO.Ext}(c, \mathsf{pk}, \mathsf{salt})$
19 $m' \leftarrow \textsc{Dec}(j, (c\|\mathsf{salt}))$
20 **if** $\hat{m} \neq \bot$ **and** $\hat{m} \neq m'$
21 Store $(j, \hat{m}\|\mathsf{salt})$ in $\mathcal{L}_{\text{FAIL}}$
22 **if** $\hat{m} = \bot$
23 **return** $\bot$
24 **else**
25 **return** $\mathsf{H}(\mathsf{pk}, \hat{m}, c\|\mathsf{salt})$

$\underline{\textsc{Dec}(j, (c\|\mathsf{salt}))}$
26 $m' \leftarrow \mathsf{Dec}(sk_j, c)$
27 $r' \leftarrow \mathsf{G}(pk_j, m', \mathsf{salt})$
28 **if** $m' = \bot$
29 **return** $\bot$
30 **else**
31 **if** $\mathsf{Enc}(pk_j, m'; r') \neq c$
32 **return** $\bot$
33 **else**
34 **return** m'

Figure 15: Original multi-instance decapsulation oracle $\textsc{Decaps}$ for $\mathsf{SFO}^{\bot}_{m,c}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$, simulation $\textsc{Decaps}'$, and failing-plaintext-extracting simulation $\textsc{Decaps}''$. The simulations use the extractable QRO simulator eCO from [DFMS22] (see Appendix 5.1), which is assumed to be freshly initialized at the beginning of the security game in which the simulations are being run. Extraction interface $\mathsf{eCO.Ext}$ is defined with respect to function f defined in Eq. (7).

after $\mathcal{A}$ finished, we use $\mathsf{eCO.Ext}$ to compute oracle preimages for all ciphertexts on which $\textsc{Decaps}$ was queried, i.e., we take each decapsulation query $(j_i, (c_i, \mathsf{salt}_i))$ and compute $\hat{m}_i \leftarrow \mathsf{eCO.Ext}(c_i, \mathsf{pk}_{j_1}, \mathsf{salt}_i)$.

By property 1 in Lem. 1, eQRO_f perfectly simulates G until the first query to $\mathsf{eCO.Ext}$, and since the first $\mathsf{eCO.Ext}$-query occurs only after $\mathcal{A}$ finishes, we have

$$\mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}^{\bot}_{m}}(\mathcal{A}) = \mathrm{Adv}^{\mathbf{G}_0} = \mathrm{Adv}^{\mathbf{G}_1} \quad . \tag{8}$$

$\mathbf{G}_2$ is like $\mathbf{G}_1$, except that $\hat{m}_i \leftarrow \mathsf{eCO.Ext}(c_i, \mathsf{pk}_{j_1}, \mathsf{salt}_i)$ is computed right after $\mathcal{A}$ queries $\textsc{Decaps}$ on $(j, (c\|\mathsf{salt})$, meaning we move the extraction queries from the end of the game to within the decapsulation calls. Thus, $\mathbf{G}_2$ is obtained from $\mathbf{G}_1$ as follows: first swap the $\mathsf{eCO.Ext}$ call that produces $\hat{m}_1$ with all random oracle calls that happen after $\mathcal{A}$ posed query c_1, then continue with the $\mathsf{eCO.Ext}$ call that produces $\hat{m}_2$, and so forth. To bound the disruption inflicted by these swaps, we now use that $\mathsf{eCO.RO}$ and $\mathsf{eCO.Ext}$ almost-commute (property 2.a-c of Lem. 1): according to that item, each query $8\sqrt{2\Gamma(f)/2^n}$-almost-commutes, where

- $\{0,1\}^n$ denotes the random oracle co-domain, which for our choice of random oracle is $\mathcal{R}$; and
- $\Gamma(f)$ is the maximal number of elements in $\mathcal{R}$ that could satisfy $Enc(pk, m; r) = c$ for any message m and any ciphertext c. Intuitively, ${}^{\Gamma(f)}/_{2^n} = {}^{\Gamma(f)}/_{|\mathcal{R}|}$ thus represents the maximal probability that sampling a random oracle output $G(m) = r$ will satisfy $\mathsf{Enc}(\mathsf{pk}, m; r) = c$. If PKE happens to be γ-spread, then that probability is upper bounded by $2^{-\gamma}$. In other words, $\Gamma(f)$ is upper bounded by $2^{-\gamma}|\mathcal{R}|$, which we now show in more detail.

More formally, for our choice of random oracle and our choice of f (see Eq. (7)),

$$\begin{aligned}\Gamma(f) &= \max_{(\mathsf{pk},m\|\mathsf{salt}),(c,\mathsf{pk}',\mathsf{salt}')} |\{r \in \mathcal{R} \mid (\mathsf{Enc}(\mathsf{pk},m;r),\mathsf{pk},\mathsf{salt}) = (c,\mathsf{pk}',\mathsf{salt}')\}| \\ &= \max_{(\mathsf{pk},m\|\mathsf{salt}),(c,\mathsf{pk}',\mathsf{salt}')} |\{r \in \mathcal{R} \mid \mathsf{Enc}(\mathsf{pk},m;r) = c \wedge \mathsf{pk} = \mathsf{pk}' \wedge \mathsf{salt} = \mathsf{salt}'\}| \\ &= \max_{\mathsf{pk},m,c} |\{r \in \mathcal{R} \mid \mathsf{Enc}(\mathsf{pk},m;r) = c\}| \\ &\leq 2^{-\gamma}|\mathcal{R}| \ ,\end{aligned}$$

where the second step uses that pk, salt, pk', and salt' have virtually no influence on the to-be-studied subset of $\mathcal{R}$, and the last step uses that we assume PKE to be γ-spread.

Thus, $8\sqrt{2\Gamma(f)/2^n} \leq 8\sqrt{2}\cdot 2^{-\gamma/2}$. For each decapsulation query, we swap the respective $\mathsf{eCO.Ext}$ query with $q_\mathsf{G} + q_D$ many random oracle calls (including calls that happen inside Decaps). Thus,

$$\left|\mathrm{Adv}^{\mathbf{G}_1} - \mathrm{Adv}^{\mathbf{G}_2}\right| \leq 8\sqrt{2}q_D(q_\mathsf{G} + q_D)\cdot 2^{-\gamma/2} \ . \tag{9}$$

$\mathbf{G}_3$ is the same as $\mathbf{G}_2$, except that $\mathcal{A}$ is run with access to the oracle Decaps$'$ instead of Decaps. Since we will only want to study the difference between Decaps$'$ and Decaps in this game, we still let the game also compute Decaps$(j_i,(c_i,\mathsf{salt}_i))$ upon $\mathcal{A}$'s oracle call. (The reason is that Decaps makes internal random oracle queries during reencryption. Omitting them might influence the behavior of $\mathsf{eCO.Ext}$ in subsequent queries and thus create additional disruptions beyond the difference between Decaps$'$ and Decaps.)

Note that unless Decaps$'$ fails to correctly emulate Decaps, the games do not differ at all. Accordingly, let DIFF be the event that $\mathcal{A}$ makes a decapsulation query $(j,(c\|\mathsf{salt}))$ such that Decaps$(j,(c\|\mathsf{salt})) \neq$ Decaps$'(j,(c\|\mathsf{salt}))$. We bound

$$\left|\mathrm{Adv}^{\mathbf{G}_1} - \mathrm{Adv}^{\mathbf{G}_2}\right| \leq \Pr[\mathsf{DIFF}] \ .$$

To analyze the probability of the event DIFF, we note that DIFF contains three cases:

- the original decapsulation oracle Decaps rejects, but the simulation Decaps$'$ does not. The latter means that Decaps$'$ returns Decaps$'(j,(c\|\mathsf{salt})) = \mathsf{H}(\hat{m},c\|\mathsf{salt})$ for $\hat{m} \leftarrow \mathsf{eCO.Ext}(c,pk_j,\mathsf{salt})$. By construction of the oracles, this means that $\hat{m}$ encrypts to c. The rejection of Decaps on the other hand implies that c decrypts to $\perp$ or fails the re-encryption check. Hence, this case occurs only if the c's preimage $\hat{m}$ is a failing plaintext under the j-th key pair.

- Neither oracle rejects, but the return values differ. This can only happen if $\hat{m} \leftarrow \mathsf{eCO.Ext}(c,pk_j,\mathsf{salt})$ differs from $m' \leftarrow \mathsf{Dec}(sk_j,c)$. Like in case 1, this implies that the pre-image $\hat{m}$ is a failing plaintext under the j-th key pair.

- Decaps does not reject, but the simulation Decaps$'$ does. The latter means that $\hat{m} \leftarrow \mathsf{eCO.Ext}(c,pk_j,\mathsf{salt})$ in line 12 yielded $\perp$. At the same time, c passed the re-encryption check inside Decaps (line 07), so $\mathsf{Enc}(pk_j,m',r') = c$ for $m' \leftarrow \mathsf{Dec}(sk_j,c)$ and $r' \leftarrow \mathsf{G}(pk_j,m',\mathsf{salt})$. Intuitively, $\mathcal{A}$ managed to compute a valid encryption without determining the right encryption randomness r' via a respective random oracle query.

We denote the combination of the first two cases by FAIL and the last case by GUESS, yielding

$$\Pr[\mathsf{DIFF}] \leq \Pr[\mathsf{GUESS}] + \Pr[\mathsf{FAIL} \wedge \neg\mathsf{GUESS}] \ ,$$

and now bound the two cases separately. To bound GUESS, we will make use of Lem. 3 below, according to which

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

To bound the probability of $\mathsf{FAIL} \wedge \neg\mathsf{GUESS}$, we define a failure-finding adversary $\mathcal{C}$ against the salted encryption scheme PKE_1: $\mathcal{C}$ forwards its input vector of public keys to $\mathcal{A}$ and runs $\mathcal{A}$ with simulation DECAPS$''$, using its own FFP-CCA oracle DEC to emulate the decryption of ciphertexts. $\mathcal{A}$'s random oracle queries to G are forwarded to $\mathcal{C}$'s extractable superposition oracle, random oracle H can be simulated via a fresh compressed oracle or using a t-wise independent function for sufficiently large t. As soon as DECAPS$''$ adds a plaintext $\hat{m}$ to $\mathcal{L}_{\mathrm{FAIL}}$, together with the involved salt salt and user index j, $\mathcal{C}$ aborts $\mathcal{A}$ and returns $(j, \hat{m}\|\mathsf{salt})$. (If $\mathcal{A}$ finishes and $\mathcal{L}_{\mathrm{FAIL}}$ is still empty, $\mathcal{C}$ returns $\perp$.) $\mathcal{C}$ succeeds if FAIL occurs, but GUESS did not: in that case, a failing plaintext $\hat{m}$ was extracted from the ciphertext that triggered FAIL, and $\mathcal{C}$ recognizes $\hat{m}$ as failing and returns it to the FFP-CCA game.

$$\Pr[\mathsf{FAIL} \wedge \neg\mathsf{GUESS}] \leq \mathrm{Adv}^{\mathsf{FFP\text{-}CCA}_u}_{\mathsf{PKE}_1}(\mathcal{C}) \ .$$

$\mathbf{G}_4$ prepares for fading out the now-redundant internal calls to DECAPS– note that these calls were not used within the responses of the oracle DECAPS$'$ to which $\mathcal{A}$ has access in game 3. These calls were only kept alive so that $\mathbf{G}_3$ could focus on bounding the difference between decapsulation and simulation, so to avoid additional disruptions in the oracle database that would have occurred when omitting DECAPS (and the involved random oracle calls) entirely. Games 4-5 are somewhat-symmetric to Games 0-2 in the sense that they use almost-commutativity to push calls to the end of the game. Concretely, $\mathbf{G}_4$ is defined like $\mathbf{G}_3$, except that the internal DECAPS invocations are postponed until after $\mathcal{A}$ finishes. With similar reasoning as when stepping from $\mathbf{G}_1$ to $\mathbf{G}_2$, we obtain

$$\left|\mathrm{Adv}^{\mathbf{G}_3} - \mathrm{Adv}^{\mathbf{G}_4}\right| \leq 8\sqrt{2}q_D^2 2^{-\gamma/2} \ .$$

Finally, $\mathbf{G}_5$ is like $\mathbf{G}_4$, except that the redundant internal calls to DECAPS are omitted entirely. Since all invocations of DECAPS already happened after the execution of $\mathcal{A}$ in game 4, this omission does not influence $\mathcal{A}$'s success probability and

$$\mathrm{Adv}^{\mathbf{G}_4} = \mathrm{Adv}^{\mathbf{G}_5} \ .$$

We now define IND_{n_c,n_u}-CPA adversary $\mathcal{B}$ against $\mathsf{KEM}^{\not\perp}$ in the eQROM_f, perfectly simulating $\mathbf{G}_5$ to $\mathcal{A}$: $\mathcal{B}$ provides $\mathcal{A}$ with access to DECAPS$'$ and forwards $\mathcal{A}$'s random oracle queries to its own IND_{n_c,n_u}-CPA game. When $\mathcal{A}$ finishes, $\mathcal{B}$ returns its guess b' to its own game.

$$\mathrm{Adv}^{\mathbf{G}_5} = \mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM}^{\perp}_m}(\mathcal{B}). \tag{10}$$

We thus obtain the desired bound by collecting the terms and bounding the collection of γ-terms, using that $q_D 2^{-\gamma} \leq q_D^2 2^{-\gamma/2}$.

□

We now close the proof above by bounding the probability of GUESS. We did not include this into the proof because we will soon (Thm. 10) want to bound a very similar event when simplifying $\mathsf{FFP\text{-}CCA}_{n_u}$ security of the encryption scheme $\mathsf{PKE}_1 := \mathsf{ST}[\mathsf{PKE}, \mathsf{G}]$. To avoid redoing the same argument, we generalize the analysis of GUESS: we also capture attackers against PKE_1 for which the decryption oracle DEC is simulated via simulation DEC$'$ (see Fig. 16). (DEC$'$ uses the same plaintext extraction technique as simulation DECAPS$'$ and differs from DEC only in a case very similar to GUESS.)

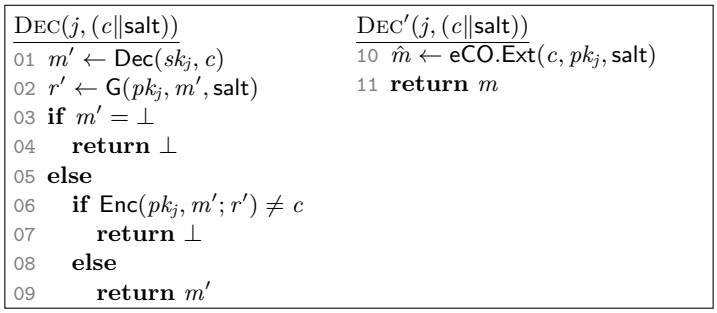

Figure 16: Simulation $\textsc{Dec}'$ of decryption oracle $\textsc{Dec}$ for $\mathsf{PKE}_1 := \mathsf{ST}[\mathsf{PKE}, \mathsf{G}]$.

Lemma 3. *Let* PKE *be* γ*-spread, and let* GUESS *be as defined in the previous proof: let* $\mathcal{A}$ *be an* $\mathsf{eQROM}_{\mathsf{Enc}}$ *adversary with access to random oracles* G, H *and decapsulation oracle* $\textsc{Decaps}$ *for* $\mathsf{SFO}^{\perp}_{m,c}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$*, issuing at most* q_D *many queries to* $\textsc{Decaps}$*. Let* $\mathcal{A}$ *be run with* $\textsc{Decaps}$ *(or* $\textsc{Decaps}'$ *as defined in Fig. 15), and upon each query* c_i*, we first compute* $\hat{m}_i = \textsc{Dec}'(j, (c\|\mathsf{salt}))$ *and then* $m'_i = \textsc{Dec}(j, (c\|\mathsf{salt}))$*. Let* GUESS *be the event that there occurs a query such that* $\hat{m}_i = \perp$ *and* $m_i \neq \perp$*. Then*

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

The same bound applies if we instead consider any $\mathsf{eQROM}_{\mathsf{Enc}}$ *adversary* $\mathcal{A}$ *that expects random oracles* G, H *and a decryption oracle* $\textsc{Dec}$ *for* $\mathsf{ST}[\mathsf{PKE}, \mathsf{G}]$*, issuing at most* q_D *many queries to* $\textsc{Dec}$*, if* $\mathcal{A}$ *has access to* $\textsc{Dec}$ *or* $\textsc{Dec}'$ *as defined in Fig. 16.*

Proof. The proof is very similar to the proof of [HHM22, Lemma 3] that bounded the probability of GUESS for the standard FO-transformation (which does not involve salts), in a single-user setting. The technical difference between [HHM22, Lemma 3] and this lemma is that there, G only hashed m, so neither pk nor salt. Consequently, the extraction interface in [HHM22, Lemma 3] is defined relative to the simpler function $f_{\mathsf{pk}} : (m, r) \mapsto \mathsf{Enc}(\mathsf{pk}, m; r)$, where pk is the public key of the single user. However, the unpredictability of our function f (see previous proof, $\mathbf{G}_2$) boils down to the same γ-term as the function f_{pk}. This explains why we obtain the same bound as in [HHM22, Lemma 3].

We will now bound the probability for a fixed query, so we will fix the i-th query $(j, (c\|\mathsf{salt}))$ and bound the probability that $\hat{m} = \perp$ but $m \neq \perp$ for that query. Intuitively, this captures that $c = \mathsf{Enc}(pk_j, m'; r')$ for $m' := \mathsf{Dec}(sk_j, c)$ and $r' \leftarrow \mathsf{G}(pk_j, m', \mathsf{salt})$, but that the query $(pk_j, m', \mathsf{salt}))$ had not yet been written into the oracle database of G before the re-encryption step. We claim

$$\Pr[\hat{m}_i = \perp \wedge m_i \neq \perp] \leq 2 \cdot 2^{-\gamma}. \tag{11}$$

Once we have proven Eq. (11), the desired bound is obtained by taking the union over all decapsulation queries.

To show prove the claim, we plug in the definitions of the interfaces $\mathsf{eCO.RO}$ and $\mathsf{eCO.Ext}$:

$$\begin{aligned}\Pr[\hat{m} = \perp \wedge m' \neq \perp] &\leq \Pr[\hat{m} = \perp \wedge \mathsf{Enc}(m'; \mathsf{eCO.RO}(pk_j, m', \mathsf{salt})) = c] \\ &= \left\| \Pi_Y^{c,x} O_{XYF} \Sigma_F^{c,\varnothing} \left|pk_j, m', \mathsf{salt}\right\rangle_X \left|0\right\rangle_Y \left|\psi_i\right\rangle_{FE} \right\|^2 , \end{aligned} \tag{12}$$

where $|\psi_i\rangle$ denotes the adversary-oracle state right before $\mathcal{A}$ submits the i-th query c, and the projectors $\Pi_Y^{c,x}$ and $\Sigma^{c,\varnothing}$ (see Eq. (3)) are defined with respect to the function $f : ((\mathsf{pk}, m\|\mathsf{salt}), r) \mapsto (\mathsf{Enc}(\mathsf{pk}, m; r), \mathsf{pk}, \mathsf{salt})$.

In [HHM22, Lemma 3], it was shown that the corresponding term – so the term where G only hashes m, meaning the X-register only contains $x \leftarrow m'$, and where the projectors are defined relative to $f_{\mathsf{pk}} : (m, r) \mapsto \mathsf{Enc}(\mathsf{pk}, m; r)$ – can be bounded by

$$\left\| \Pi_Y^{c,x} O_{XYF} \Sigma_F^{c,\varnothing} \left|x\right\rangle_X \left|0\right\rangle_Y \left|\psi_i\right\rangle_{FE} \right\| \leq \sqrt{2} \cdot 2^{-\gamma/2} \ . \tag{13}$$

The proof used that the chosen projectors are diagonal in the computational basis, plus a commutator bound which in turn used that the predictability term of pk can be bounded by $\Gamma(f_{\mathsf{pk}}) \leq 2^{-\gamma}|\mathcal{R}|$. By adapting the X-register such that it additionally accommodates pk and salt, and by noticing that our projectors are also diagonal and that $\Gamma(f) \leq 2^{-\gamma}|\mathcal{R}|$, we thus obtain the same bound as in Eq. (13) in our adapted setting, and thus Eq. (11) by combining Eq. (13) with Eq. (12).

□

Bound FFP-CCA via FFP-CPA. We now show that we can simplify the FFP-CCA term as indicated below Thm. 5: firstly, we simplify it to its passive counterpart, FFP-CPA.

Theorem 10 ($\mathsf{ST}\,[\mathsf{PKE}, \mathsf{G}\,]\,\mathsf{FFP\text{-}CPA}_u \Rightarrow \mathsf{ST}\,[\mathsf{PKE}, \mathsf{G}\,]\,\mathsf{FFP\text{-}CCA}_u$)**.** *Let* PKE *be a* γ*-spread public key encryption scheme, and let* $\mathcal{C}$ *be an* $\mathsf{FFP\text{-}CCA}_u$ *adversary in the* $\mathsf{eQROM}_{\mathsf{Enc}}$ *against* $\mathsf{ST}[\mathsf{PKE}, \mathsf{G}]$*, issuing at most* q_D *many decryption queries and at most* $q_{\mathsf{eCO.RO}}$ *and* $q_{\mathsf{eCO.Ext}}$ *many queries to the two interfaces* $\mathsf{eCO.RO}$ *and* $\mathsf{eCO.Ext}$*, respectively.*

Then there exist an $\mathsf{FFP\text{-}CPA}_u$ *adversary* $\mathcal{C}$*' in the* $\mathsf{eQROM}_{\mathsf{Enc}}$ *such that*

$$\mathrm{Adv}^{\mathsf{FFP\text{-}CCA}_u}_{\mathsf{ST}[\mathsf{PKE},\mathsf{G}]}(\mathcal{C}) \leq (q_D + 1) \cdot \mathrm{Adv}^{\mathsf{FFP\text{-}CPA}_u}_{\mathsf{ST}[\mathsf{PKE},\mathsf{G}]}(\mathcal{C}') + 12 q_D (q_{\mathsf{G}} + 4 q_D) 2^{-\gamma/2} \ . \tag{14}$$

The adversary $\mathcal{C}$*' makes* $q_{\mathsf{eCO.RO}}$ *queries to* $\mathsf{eCO.RO}$ *and* $q_{\mathsf{eCO.Ext}} + q_D$ *queries to* $\mathsf{eCO.Ext}$*, and its runtime satisfies* $\mathrm{Time}(\mathcal{C}') = \mathrm{Time}(\mathcal{B}) + O(q_D)$*.*

Proof. We will want to show that the decryption oracle $\textsc{Dec}$ provided in the FFP-CCA game can be simulated via oracle $\textsc{Dec}'$ (see Fig. 16). Since $\textsc{Dec}'$ works without the secret key, we can then construct $\mathcal{C}'$ that uses $\mathcal{C}$, simulating $\mathcal{C}$'s decryption oracle via $\textsc{Dec}'$. In a way, this is analogous to Thm. 5 in which we replaced $\textsc{Decaps}$ by simulation $\textsc{Decaps}'$. We note the tightness loss: as a passive adversary $\mathcal{C}'$ cannot use $\textsc{Dec}$ to determine at which $\textsc{Dec}'$ query a failure occurs, $\mathcal{C}'$ instead resorts to guessing the query.

Let $\mathbf{G}_0$ be the $\mathsf{FFP\text{-}CCA}_u$ game, and let games 1-5 be defined based on $\mathbf{G}_0$, reflecting the same changes that we did in the proof of Thm. 5 – we first 'commute in' the extraction queries that would be needed for $\textsc{Dec}'$, then switch from $\textsc{Dec}$ to $\textsc{Dec}'$, and then omit the redundant internal calls to $\textsc{Dec}$. Like in the proof of Thm. 5, we have

$$\mathrm{Adv}^{\mathsf{FFP\text{-}CCA}_u}_{\mathsf{ST}[\mathsf{PKE},\mathsf{G}]}(\mathcal{C}) \leq \mathrm{Adv}^{\mathbf{G}_5} + 12 q_D (q_{\mathsf{G}} + 2 q_D) 2^{-\gamma/2} + \Pr\left[\mathsf{FAIL} \wedge \neg \mathsf{GUESS}\right] \ .$$

Assume without loss of generality that $\mathcal{C}$ makes exactly q_D many queries to its decryption oracle (if it does not, we modify $\mathcal{C}$ by adding a number of useless decryption queries in the end). We now define FFP-CPA adversary $\mathcal{C}'$ as follows: $\mathcal{C}'$ samples $i \leftarrow \{1, ..., q_D + 1\}$ and runs $\mathcal{C}$ until its i-th decryption query $((j_i, (c_i, \mathsf{salt}_i)))$, or until the end if $i = q_D + 1$. If i is smaller then $i = q_D + 1$, $\mathcal{C}'$ returns $\hat{m}_i$, the message that was computed during $\mathcal{C}$'s i-th query to $\textsc{Decaps}'$, together with j_i and salt_i. If $i = q_D + 1$, then $\mathcal{C}'$ simply outputs the output of $\mathcal{C}$. By construction,

$$\mathrm{Adv}^{\mathsf{FFP\text{-}CPA}}_{\mathsf{T}[\mathsf{PKE},\mathsf{G}]}(\mathcal{C}') \geq \frac{1}{q_D + 1} \left(\mathrm{Adv}^{\mathbf{G}_5} + \Pr[\mathsf{FAIL} \wedge \neg \mathsf{GUESS}] \right) \ .$$

Combining the two inequalities yields the desired bound. □

Bound FFP-CCA via δ-correctness. To provide the alternative simplification of the FFP-CCA term as indicated below Thm. 5, we now bound it in terms of (statistical) worst-case correctness.

Theorem 11 (PKE δ (n_{u})-worst-case-correct $\Rightarrow$ ST[PKE, G] FFP-CCA$_u$)**.** *Let* PKE *be a (randomized)* PKE *scheme that is δ-worst-case-correct, and let $\mathcal{C}$ be an* eQROM *adversary against* ST[PKE, G] *in the* FFP-CCA$_u$ *game as defined in Fig. 10, issuing at most q_D many decryption queries and at most q many queries to its interface* eCO.RO*. Then*

$$\mathrm{Adv}^{\mathsf{FFP\text{-}CCA}_u}_{\mathsf{ST}[\mathsf{PKE},\mathsf{G}]}(\mathcal{C}) \leq 10(q + q_D + 1)^2 \cdot \delta(n_{\mathsf{u}}) \ .$$

The proof will use [HM24, Theorem 3] which proved a corresponding single-user bound for the 'standard' derandomization transformation T that is used within 'standard' FO-transforms. Before giving the proof of Thm. 11, we thus first recall this theorem.

Theorem 12 (PKE δ-worst-case-correct $\Rightarrow$ T[PKE, G] FFP-CCA)**.** *Let* PKE *be a (randomized)* PKE *scheme that is δ-worst-case-correct, and let $\mathcal{C}$ be an* FFP-CCA *adversary against* T[PKE, G] *in the* eQROM$_{\mathsf{Enc}}$*, issuing at most q_D decryption queries and q many queries to its extQROM oracle interface* eCO.RO*. Then*

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

Proof of Thm. 11. First, we will show that the bound in Thm. 12 also applies for our transformation ST when viewing the salts as part of the message, like in Fig. 10. Afterwards, we lift the setting from single- to multi-user. We decompose the 'also-output-salt' counterpart to ST into its 'salting' part and a variation of T:

1. Apply to PKE the 'salting' transform Salt which turns PKE into an encryption scheme $\mathsf{Salt}[\mathsf{PKE}] := (\mathsf{Gen}, \mathsf{Enc}_{\mathsf{st}}, \mathsf{Dec}_{\mathsf{st}})$ with message space $\mathcal{M} \times \{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$. $\mathsf{Enc}_{\mathsf{st}}$ simply appends the given salt to the PKE encryption, and $\mathsf{Dec}_{\mathsf{st}}$ appends it to its PKE decryption, see Fig. 17.

2. Apply to Salt[PKE] a multi-user variant T_{pk} of the original T-transform. T_{pk} only differs from T by including pk into the randomness derivation, see Fig. 18.

$\mathsf{Enc}_{\mathsf{st}}(pk, m\|\mathsf{salt})$	$\mathsf{Dec}_{\mathsf{st}}(\mathsf{sk}, c\|\mathsf{salt})$
01 $c \leftarrow \mathsf{PKE.Enc}(pk, m)$	03 $m' \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, c)$
02 **return** $c\|\mathsf{salt}$	04 **return** $m'\|\mathsf{salt}$

Figure 17: Salted PKE scheme $\mathsf{Salt}[\mathsf{PKE}] = (\mathsf{Gen}, \mathsf{Enc}_{\mathsf{st}}, \mathsf{Dec}_{\mathsf{st}})$.

$\mathsf{T}_{\mathsf{pk}}[\mathsf{PKE},\mathsf{G}].\mathsf{Enc}(\mathsf{pk}, m)$	$\mathsf{T}_{\mathsf{pk}}[\mathsf{PKE},\mathsf{G}].\mathsf{Dec}(\mathsf{sk}, c)$
01 $r \leftarrow \mathsf{G}(\mathsf{pk},m)$	04 $m' = \mathsf{PKE.Dec}(\mathsf{sk}, c)$
02 $c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m; r)$	05 $r' \leftarrow \mathsf{G}(\mathsf{pk},m)$
03 **return** c	06 **if** $m' = \perp$ **or** $\mathsf{PKE.Enc}(\mathsf{pk}, m'; r') \neq c$
	07 **return** $\perp$
	08 **else return** m'

Figure 18: Multi-user variant $\mathsf{T}_{\mathsf{pk}}[\mathsf{PKE}, \mathsf{G}]$, deviations from T highlighted in violet.

When viewing salts as part of the message for ST, we have $\mathsf{ST}[\mathsf{PKE},\mathsf{G}] = \mathsf{T}_{\mathsf{pk}}[\mathsf{Salt}[\mathsf{PKE}],\mathsf{G}]$ and thus

$$\mathrm{Adv}^{\text{FFP-CCA}}_{\mathsf{ST}[\mathsf{PKE},\mathsf{G}]}(\mathcal{C}) = \mathrm{Adv}^{\text{FFP-CCA}}_{\mathsf{T}_{\mathsf{pk}}[\mathsf{Salt}[\mathsf{PKE}],\mathsf{G}]}(\mathcal{C}) \ .$$

To finish step 1, we would like to use [HM24, Theorem 3] to argue that

$$\mathrm{Adv}^{\text{FFP-CCA}}_{\mathsf{T}_{\mathsf{pk}}[\mathsf{Salt}[\mathsf{PKE}],\mathsf{G}]}(\mathcal{C}) \leq 10(q+q_D+1)^2 \cdot \delta \ . \tag{15}$$

We address two (minor) obstacles: first, we use transformation T_{pk} instead of T. However, feeding pk into G during randomness derivation has no influence whatsoever on this single-user bound since this additional hash input neither hinders nor eases the search for a message that exhibits decryption failure. (In case this is not obvious, see Thm. 13 below). Second, the bound's right-hand side would use the correctness term of $\mathsf{Salt}[\mathsf{PKE}]$, denoted by δ^{st}, not that of PKE. We thus quickly verify that δ^{st} is upper-bounded by δ: fix any key pair $\mathsf{kp} := (\mathsf{pk},\mathsf{sk}) \in \mathrm{Supp}(\mathsf{Gen})$, any message-salt tuple $(m\|\mathsf{salt}) \in \mathcal{M} \times \{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$, and define the conditional terms

$$\begin{aligned}\delta(\mathsf{kp},m) &= \Pr[\mathsf{Dec}(\mathsf{sk},c) \neq m : c \leftarrow \mathsf{Enc}(\mathsf{pk},m)] \\ \delta^{\mathsf{st}}(\mathsf{kp},m\|\mathsf{salt}) &= \Pr[\mathsf{Dec}_{\mathsf{st}}(\mathsf{sk},c\|\mathsf{salt}) \neq m\|\mathsf{salt} : c\|\mathsf{salt} \leftarrow \mathsf{Enc}_{\mathsf{st}}(\mathsf{pk},m\|\mathsf{salt})] \ .\end{aligned}$$

Plugging this notation in our definition of worst-case correctness, we identify

$$\delta = \mathbb{E}\left[\max_m \delta(\mathsf{kp},m)\right] \text{ and } \delta^{\mathsf{st}} = \mathbb{E}\left[\max_{m\|\mathsf{salt}} \delta^{\mathsf{st}}(\mathsf{kp},m\|\mathsf{salt})\right] \ ,$$

where the expectations are both taken over $\mathsf{kp} \leftarrow\!\!\$\ \mathsf{Gen}$. But the conditional terms coincide: since the salted encryption $(c\|\mathsf{salt})$ of any fixed tuple $(m\|\mathsf{salt})$ fails to decrypt to $(m\|\mathsf{salt})$ iff c fails to decrypt to m,

$$\delta^{\mathsf{st}}(\mathsf{pk},\mathsf{sk},m\|\mathsf{salt}) = \delta(\mathsf{pk},\mathsf{sk},m), \text{ so } \delta^{\mathsf{st}} = \delta$$

and we thus obtain the bound claimed in Eq. (15).

We proceed to capturing the multi-user setting. In [HM24], Thm. 12 was proven by fixing the key pair $(\mathsf{pk},\mathsf{sk})$, showing that conditioned on that key pair,

$$\Pr[\text{FFP-CCA}^{\mathcal{C}}_{\mathsf{T}[\mathsf{PKE},\mathsf{G}]} \Rightarrow 1 \mid (\mathsf{pk},\mathsf{sk})] \leq 10(q+q_D+1)^2 \cdot \delta(\mathsf{pk},\mathsf{sk}) \ ,$$

where

$$\delta(\mathsf{pk},\mathsf{sk}) = \max_m \Pr_{r \leftarrow \mathcal{R}}[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m;r)) \neq m] \ ,$$

and then taking the expectation over the key pair to obtain

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

Using the reasoning above, we adapt each of these steps to accommodate transformation ST instead of T. In particular,

$$\Pr[\text{FFP-CCA}^{\mathcal{C}}_{\mathsf{ST}[\mathsf{PKE},\mathsf{G}]} \Rightarrow 1 \mid (\mathsf{pk},\mathsf{sk})] \leq 10(q+q_D+1)^2 \cdot \delta(\mathsf{pk},\mathsf{sk}) \ . \tag{16}$$

To capture n_{u} many users, we generalize this to

$$\mathrm{Adv}^{\text{FFP-CCA}_u}_{\mathsf{ST}[\mathsf{PKE},\mathsf{G}]}(\mathcal{C}) \leq 10(q+q_D+1)^2 \cdot \delta(n_{\mathsf{u}}) \ ,$$

by sampling n_{u} many key pairs instead of a single one and taking the maximum over the sampled key pairs. (We fix the key pairs and case separate the winning condition of FFP-CCA_u into the cases of $\mathcal{C}$ winning with the j-th key pair – in each case, $\mathcal{C}$'s advantage is upper bounded by its advantage in the case with the 'best' key pair, which in turn is upper bounded by Eq. (16).)

□

For the sake of completeness, we close the proof of Thm. 11 by taking T-derandomized schemes and showing that in/exclusion of pk during randomness derivation has no influence on their FFP-CCA property in the single-user setting.

Theorem 13 ($\mathsf{T}_{\mathsf{pk}}[\mathsf{PKE},\mathsf{G}]$ FFP-CCA $\Leftrightarrow$ $\mathsf{T}[\mathsf{PKE},\mathsf{G}]$ FFP-CCA)**.** *Let* PKE *be a (randomized)* PKE *scheme. Let $\mathcal{C}$ be an* FFP-CCA *adversary against* $\mathsf{T}[\mathsf{PKE},\mathsf{G}]$ *in the* $\mathsf{eQROM}_{\mathsf{Enc}}$*. Additionally, let a second extractor function f be like* Enc*, but adapted to hashed public keys, i.e. defined by* $f : ((\mathsf{pk}, m), r) \mapsto (\mathsf{Enc}(\mathsf{pk}, m; r), \mathsf{pk})$*. Then there exists an* FFP-CCA *adversary $\mathcal{C}'$ against* $\mathsf{T}_{\mathsf{pk}}[\mathsf{PKE},\mathsf{G}']$ *in the* eQROM_f *such that*

$$\mathrm{Adv}^{\mathsf{FFP\text{-}CCA}}_{\mathsf{T}[\mathsf{PKE},\mathsf{G}]}(\mathcal{C}) \leq \mathrm{Adv}^{\mathsf{FFP\text{-}CCA}}_{\mathsf{T}_{\mathsf{pk}}[\mathsf{PKE},\mathsf{G}']}(\mathcal{C}') \ .$$

Vice versa, let $\mathcal{D}$ be an FFP-CCA *adversary against* $\mathsf{T}_{\mathsf{pk}}[\mathsf{PKE},\mathsf{G}']$ *in the* eQROM_f*, where f is defined as above. Then there exists an* FFP-CCA *adversary $\mathcal{D}'$ against* $\mathsf{T}[\mathsf{PKE},\mathsf{G}]$ *in the* $\mathsf{eQROM}_{\mathsf{Enc}}$ *such that*

$$\mathrm{Adv}^{\mathsf{FFP\text{-}CCA}}_{\mathsf{T}_{\mathsf{pk}}[\mathsf{PKE},\mathsf{G}']}(\mathcal{D}) \leq \mathrm{Adv}^{\mathsf{FFP\text{-}CCA}}_{\mathsf{T}[\mathsf{PKE},\mathsf{G}]}(\mathcal{D}') \ .$$

Adversary $\mathcal{C}'$ runs in about the time of $\mathcal{C}$ and issues as many queries to its respective oracles as $\mathcal{C}$ does, and adversary $\mathcal{D}'$ runs in about the time of $\mathcal{D}$ and issues as many queries to its respective oracles as $\mathcal{D}$ does.

Proof. First, consider FFP-CCA adversary $\mathcal{C}$ against $\mathsf{T}[\mathsf{PKE},\mathsf{G}]$ in the $\mathsf{eQROM}_{\mathsf{Enc}}$. We construct FFP-CCA adversary $\mathcal{C}'$ against $\mathsf{T}_{\mathsf{pk}}[\mathsf{PKE},\mathsf{G}']$ as follows: $\mathcal{C}'$ forwards its challenge public key pk^* to $\mathcal{C}$, and at the end, it forwards the output of $\mathcal{C}$ to its own game.

$\mathcal{C}'$ simulates the random oracle G for $\mathcal{C}$ as follows: upon a query $|\varphi\rangle_{\mathcal{M}}$ to G, $\mathcal{C}'$ queries its own oracle G' on $|\psi\rangle_{\mathcal{PK}\times\mathcal{M}} \leftarrow |\mathsf{pk}^*\rangle_{\mathcal{PK}} \otimes |\varphi\rangle_{\mathcal{M}}$ and returns the result to $\mathcal{C}$. This simulation ensures that $\mathsf{G}(m) = \mathsf{G}'(\mathsf{pk}^*, m)$ for all messages, and thereby that the re-encryption check with $r \leftarrow \mathsf{G}'(\mathsf{pk}^*, m)$ is identical to the re-encryption check with $r \leftarrow \mathsf{G}(m)$. This has two consequences: first, if $\mathcal{C}$ wins, then so does $\mathcal{C}'$. Second, the decryption oracle that is provided to $\mathcal{C}'$ perfectly emulates the decryption oracle that $\mathcal{C}$ would expect, $\mathcal{C}'$ thus can simply forward any decryption query to its own decryption oracle.

It remains to describe how $\mathcal{C}'$ can perfectly simulate the extraction interface for $\mathcal{C}$. Whenever $\mathcal{C}$ issues an extraction query c, $\mathcal{C}'$ issues the query (c, pk^*) to its own extraction interface. The result is either $\perp$ or a message m such that $c = \mathsf{Enc}(\mathsf{pk}, m; \mathsf{G}'(\mathsf{pk}^*, m))$. By definition of G, this coincides with the definition of the extraction interface of $\mathcal{C}$.

In conclusion, $\mathcal{C}'$ issues queries to its oracles exactly when $\mathcal{C}$ does, perfectly simulates the FFP-CCA game to $\mathcal{C}$, and wins if $\mathcal{C}$ wins. We conclude

$$\mathrm{Adv}^{\mathsf{FFP\text{-}CCA}}_{\mathsf{T}[\mathsf{PKE},\mathsf{G}]}(\mathcal{C}) \leq \mathrm{Adv}^{\mathsf{FFP\text{-}CCA}}_{\mathsf{T}_{\mathsf{pk}}[\mathsf{PKE},\mathsf{G}']}(\mathcal{C}') \ .$$

For the other direction, we consider FFP-CCA adversary $\mathcal{D}$ against $\mathsf{T}_{\mathsf{pk}}[\mathsf{PKE},\mathsf{G}']$ in the eQROM_f. We construct FFP-CCA adversary FFP-CCA adversary $\mathcal{D}'$ against $\mathsf{T}[\mathsf{PKE},\mathsf{G}]$ in the $\mathsf{eQROM}_{\mathsf{Enc}}$ as follows: again, $\mathcal{D}'$ will forward its challenge public key pk^* to $\mathcal{D}$, and its output to its own game. To be able to simulate G, $\mathcal{D}'$ will prepare and maintain an internal compressed superposition oracle eCO with domain $\mathcal{PK} \times \mathcal{M}$. To answer a query $|\varphi\rangle_{\mathcal{PK}\times\mathcal{M}}$ to G', $\mathcal{D}'$ will use its own oracle G and this additional compressed oracle: on each base state, the return value is

$$O_{\mathsf{G}'}(|\mathsf{pk}, m\rangle \otimes |out\rangle) := \begin{cases} |\mathsf{pk}^*, m\rangle \otimes |out \oplus \mathsf{G}(m)\rangle & \mathsf{pk} = \mathsf{pk}^* \\ |\mathsf{pk}, m\rangle \otimes |out \oplus \mathsf{eCO.RO}(\mathsf{pk}, m)\rangle & \mathsf{pk} \neq \mathsf{pk}^* \end{cases}$$

This simulation again ensures that $\mathsf{G}'(\mathsf{pk}^*, m) = \mathsf{G}(m)$ for all messages. The simulation of the decryption oracle thus again can be done perfectly by forwarding all queries, and $\mathcal{D}'$

again inherits its success from $\mathcal{D}$. To respond to extraction queries (c, pk), $\mathcal{D}'$ will simply execute $\mathsf{eCO.Ext}$ whenever $\mathsf{pk} \neq \mathsf{pk}^*$, and forward the query to its extraction oracle for G when $\mathsf{pk} = \mathsf{pk}^*$.

The 'composed domain-separated' simulation perfectly emulates an extractable superposition oracle – the extraction interface of G' anyways would take its input (t, pk), perform the measurement $M_{t,\mathsf{pk}}$ and return the result. The measurement thus would anyways act on the database of G if $\mathsf{pk} = \mathsf{pk}^*$, and otherwise on the database of eCO. (This can be verified by reordering the registers in the oracle database $D_{\mathsf{G}'}$ of G' in a way such that it can be rewritten as $D_{\mathsf{G}} \otimes D_{\mathsf{eCO}}$.) □

Additional take-away The approach used in the previous proof can serve as a general framework to lift a certain proof technique to the extQROM:

Remark 5 (**Lifting ROM proofs based on domain separation**). In the previous proof, we showed a quite intuitive result: we showed that a construction C that uses an internal computation $v \leftarrow \mathsf{RO}(x)$ is exactly as secure as a counterpart C_{var} of C that replaces $v \leftarrow \mathsf{RO}(x)$ by $v \leftarrow \mathsf{RO}(\mathsf{var}, x)$, provided var is some trivially computable variable.

In our setting, the eQROM extractor interface in the game for C_{var} accounted for this additional hash input, by using the extractor function $f' \leftarrow (f_{\mathsf{var}}(x, y), \mathsf{var})$ with f_{var} being the extractor function in the game for C.

In the random oracle, one would show that security of C implies security of C_{var} via domain separation. (Given $\mathsf{RO} : X \to T$, simulate $\mathsf{RO}' : X \times \mathit{Var} \to T$ via $\mathsf{RO}'(x, \mathsf{var}') \leftarrow \mathsf{RO}(x)$ iff $\mathsf{var} = \mathsf{var}'$, and lazy sampling otherwise.)

To lift this to the extractable QROM, we defined a 'composed domain-separated' simulation: we still viewed RO as $\mathsf{RO}'(-, \mathsf{var}')$ and composed it with an internal extractable QRO RO_{int} with complementary domain $X \times \mathit{Var} \setminus \{\mathsf{var}\}$, by accordingly defining the output of RO' on the base states. It only remained to simulate the extraction interface $\mathit{Extract}'(t, \mathsf{var}')$ of RO', which forwarded the extraction query to either RO or RO'', depending on input value var'.

The 'composed domain-separated' simulation in general perfectly emulates an extractable superposition oracle with the same reasoning as at the end of the previous proof. The technique can thus be applied whenever one want to include or omit a hash input var in a construction, provided that

- var can be computed by a reduction, e.g., because it is public,
- the construction's extractor function f was already parameterized by var anyways, and one is satisfied if security of the 'include-var' construction is defined in the $\mathsf{eQROM}_{f'}$ for the adapted extractor function f'.

D Bounding t

Recall that t is a bound on the total number of times the same salt is used during the sampling of challenges for a single public key. We claim that for practical parameters, we can bound t by some small constant with overwhelming probability. For each $j \in [n_{\mathsf{u}}]$, we sample at most n_{c} independent, uniformly random salt values from $\{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$. Let $X_{j,\mathsf{salt}}$ be a random variable, denoting the number of times salt was sampled for user j. Clearly $X_{j,\mathsf{salt}}$ follows a binomial distribution, with $p = \frac{1}{2^{\mathsf{len}_{\mathsf{salt}}}}$ and $n = n_{\mathsf{c}}$. Then, for $T \in \mathbb{N}$, and $\mathsf{salt} \in \{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$, we have that

$$\Pr[X_{j,\mathsf{salt}} > T] = 1 - \sum_{i=0}^{T} \left(\binom{n_{\mathsf{c}}}{i} \left(\frac{1}{2^{\mathsf{len}_{\mathsf{salt}}}} \right)^i \left(1 - \frac{1}{2^{\mathsf{len}_{\mathsf{salt}}}} \right)^{n_{\mathsf{c}} - i} \right)$$

Taking a union bound over all possible salts we obtain

$$\Pr[\exists \mathsf{salt} \in \{0,1\}^{\mathsf{len}_{\mathsf{salt}}} : X_{j,\mathsf{salt}} > T] \leq 2^{\mathsf{len}_{\mathsf{salt}}} \left(1 - \sum_{i=0}^{T} \left(\binom{n_{\mathsf{c}}}{i} \left(\frac{1}{2^{\mathsf{len}_{\mathsf{salt}}}}\right)^i \left(1 - \frac{1}{2^{\mathsf{len}_{\mathsf{salt}}}}\right)^{n_{\mathsf{c}}-i}\right)\right)$$

Finally, we take a union bound over all $j \in [n_{\mathsf{u}}]$ and obtain

$$\Pr[t > T] \leq n_{\mathsf{u}} 2^{\mathsf{len}_{\mathsf{salt}}} \left(1 - \sum_{i=0}^{T} \left(\binom{n_{\mathsf{c}}}{i} \left(\frac{1}{2^{\mathsf{len}_{\mathsf{salt}}}}\right)^i \left(1 - \frac{1}{2^{\mathsf{len}_{\mathsf{salt}}}}\right)^{n_{\mathsf{c}}-i}\right)\right)$$

For $n_{\mathsf{u}} = 2^{32}$, $n_{\mathsf{c}} = 2^{64}$ and $\mathsf{len}_{\mathsf{salt}} = 128$, we obtain that $\Pr[t > 4] \leq 2^{-166}$.

E A Note on Existing Literature

In this section, we enumerate some bugs and inaccuracies in existing FO literature, which are corrected in our work. The authors of this work are not entirely responsible for *identifying* these bugs, but we are responsible for patching them [1]. In particular, we credit Chris Peikert with identifying the second mentioned bug in [HHK17], as well as the first bug in [DHK+21], in private communication [Pei25].

"A Modular Analysis of the Fujisaki–Okamoto Transform". There are two small bugs in [HHK17, Lemma 2.3], which is recalled below.

> **Lemma 2.3** ([HHK17]). Let PKE be a public key encryption scheme. For any OW-CPA adversary $\mathcal{A}$ there exists a IND-CPA adversary $\mathcal{B}$, running $\mathcal{A}$ as a subroutine, such that
>
> $$\mathrm{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}} \leq \frac{1}{|\mathcal{M}|} + \mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}.$$

This lemma contains two bugs, which are corrected in Lem. 2. The corrected bound would be

$$\mathrm{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}} \leq \frac{1}{|\mathcal{M}|} + 2 \cdot \mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}} + \delta.$$

1. The extra factor of 2 comes from the fact that the IND-CPA advantage function is as defined in [HHK17] is

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

   while the OW-CPA advantage function is defined as

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

   In particular, the former ranges from 0 to 0.5, while the latter ranges from 0 to 1. If we consider a PKE, for which encryption is the identity function, we have a trivial $\mathcal{A}$ such that $\mathrm{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A})$, regardless of $|\mathcal{M}|$.

2. The addition of δ is more subtle, and was also corrected in [Höv21]. Intuitively, in the OW-CPA experiment of [HHK17], the adversary $\mathcal{A}$ is given the public key pk and a challenge ciphertext c^* and outputs m'. The experiment returns 1 if and only if $\mathsf{PCO}(m, c^*) = 1$. PCO is a plaintext checking oracle - it returns whether c^*

[1] The second bug in [HHK17, Lemma 23] was also patched in [Höv21].

decrypts to m, not whether m *encrypts* to c^*. In the event of a decryption failure, these are not the same thing. The reduction between OW-CPA security IND-CPA security, the IND-CPA adversary samples two messages m_0 and m_1, queries their challenge oracle to get $c^* \leftarrow \mathsf{Enc}(pk, m_b)$ and passes c^* to the OW-CPA adversary which outputs m'. $\mathcal{A}$ returns b' if $m' = m'_b$, and a random guess otherwise. Thus we have $\Pr[b = b'] + \delta = \mathrm{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A})$.

A natural question would be whether this additional δ can be avoided by simply changing the OW-CPA experiment so that it returns 1 if and only if m encrypts to c^* under pk, rather than querying the plaintext checking oracle PCO. Indeed, it can, but the δ will surface elsewhere downstream. In particular, the δ would emerge in [HHK17, Theorem 3.3, Theorem 3.4] in relating the flag CHAL to the advantage of a one-way adversary. For a closer look at how this δ emerges, see Thm. 8 or [Höv21, Theorem 2.1.4, Theorem 2.1.5].

"Faster Lattice-Based KEMs via a Generic Fujisaki–Okamoto Transform Using Prefix Hashing". We are only aware of one bug in the proofs of [DHK+21]. It appears in both their Theorem 3.1 and 3.2. For simplicity, we focus on the proof of [DHK+21, Theorem 3.1].

Intuitive summary: the bug has to do with how the flag QUERY is defined. There are essentially two ways to order a sequence of events, which the flag tries to capture, but only one of these orders is captured.

As written, QUERY is raised whenever an adversarial ROM query includes a message m which had *previously* been sampled by the challenge oracle Chal, and is therefore associated with a challenge ciphertext. However, no flag is raised if the challenge oracle samples a new message which collides with an already made adversarial ROM query. In the final game hop of the proof, the authors define a simulation which only breaks in the event of a collision between some set of messages, and the set of adversarial ROM queries. The authors identify this event with the flag QUERY, but they are not the same, as the collision may occur when a new value sampled by the challenge collides with a previous adversarial query, and not the other way around. Our fix can be found in Thm. 3. At first, we had thought that it was necessary to define a new flag within QUERY, but this analysis significantly deteriorated security bounds, and made the QROM analysis a lot messier. We realized that it was sufficient to check for collisions between the two relevant sets only after all queries have been made, and a final output has been returned by the adversary. Thus, we moved the event QUERY outside any of the oracles, and within the main reduction itself.

Aside from this bug in the proof, which had a simple fix, we highlight two technical errors in [DHK+21] (only one of which is relevant for this work.)

1. The first, is a false claim on page 3, that if one assumes the hardness of MLWE such that the number of samples is unlimited, one can show that $\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}} \approx \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}$, where PKE is the underlying PKE in Kyber/ML-KEM. This claim is not relevant to our analysis, since we do not bound the IND_{n_c,n_u}-CPA security of any existing PKEs. Thus, there is nothing to patch in our analysis. This claim was refuted by Bernstein in [Ber22], as well as by Peikert in private communication [Pei25]. The issue is that each encryption will use its own MLWE secret, and so the multiple ciphertexts should be identified by distinct MLWE secrets, and not with distinct samples which use the same MLWE secret.

2. The second error has to do with the comparison to the hybrid bound on page 3. This comparison does not account for the difference in security model between [DHK+21] and [BBM00], which is explained in our Remark 1. The model of [DHK+21] gives the adversary a total budget of q_C challenge queries, which can be distributed among n users adversarially. The model of [BBM00] instead limits *queries-per-user*. The

authors of [DHK+21] claim that by the hybrid argument of [BBM00] one obtains a trivial bound with a security loss linear in both q_C and n. However, the hybrid bound of [BBM00] emerges from a security analysis with nq_C total challenge ciphertexts, whereas the bound of [DHK+21] emerges from a security analysis with just q_C total challenge ciphertexts. If, for example, the adversary concentrated all their queries to a single user, the hybrid bound would only claim a loss linear in q_C, and not in n. This observation was one of the reasons that we opted to use the security model of [BBM00].