Tight Multi-challenge Security Reductions for Key Encapsulation Mechanisms

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

[1] École Polytechnique Fédérale de Lausanne, Switzerland
lewis.glabush@epfl.ch
[2] Eindhoven University of Technology, The Netherlands
kathrin@hoevelmanns.net
[3] University of Waterloo, Canada
dstebila@uwaterloo.ca

Abstract. A key encapsulation mechanism (KEM) allows two parties to establish a shared secret key using only public communication. For post-quantum KEMs, the most widespread approach is to design a passively secure public-key encryption (PKE) scheme and then apply the Fujisaki–Okamoto (FO) transform that turns any such PKE scheme into an IND-CCA secure KEM. While the base security requirement for KEMs is typically IND-CCA security, adversaries in practice can sometimes observe and attack many public keys and/or ciphertexts, which is referred to as multi-challenge security. FO does not necessarily guarantee multi-challenge security: for example, FrodoKEM, a Round 3 alternate in NIST's post-quantum project, used FO to achieve IND-CCA security, but was subsequently shown to be vulnerable to attackers that can target multiple ciphertexts. To avert this multi-ciphertext attack, the FrodoKEM team added a salt to the encapsulation procedure and proved that this does not degrade (single-ciphertext) IND-CCA security. The formal analysis of whether this indeed averts multi-ciphertext attacks, however, was left open, which we address in this work.
Firstly, we formalize FrodoKEM's approach as a new variant of the FO transform, called the *salted* FO transform. Secondly, we give tight reductions from multi-challenge security of the resulting KEM to multi-challenge security of the underlying public key encryption scheme, in both the random oracle model (ROM) and the quantum-accessible ROM (QROM). Together these results justify the multi-ciphertext security of the salted FrodoKEM scheme, and can also be used generically by other schemes requiring multi-ciphertext security.

Keywords: Fujisaki–Okamoto transform; key exchange; quantum random oracle model (QROM); post-quantum cryptography; multi-challenge security; FrodoKEM

K.H. was supported by an NWO VENI grant (Project No. VI.Veni.222.397). D.S was supported by Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery grant RGPIN-2022-03187 and NSERC Alliance grant ALLRP 578463-22.

1 Introduction

The Fujisaki–Okamoto (FO) transform [FO99, FO13] converts a weakly secure public-key encryption scheme into an $\mathsf{IND\text{-}CCA}$-secure public-key encryption scheme. In the context of post-quantum cryptography, its adaptations for key encapsulation mechanisms (KEMs) given in [Den03a, HHK17] received renewed attention and by now have become the de-facto 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 (amongst others) on some variant of FO.

IND-CCA vs. multi-challenge security. The required security goal for KEMs during the NIST PQC process was $\mathsf{IND\text{-}CCA}$ 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 FO meets this goal by developing more sophisticated formalisms to capture quantum attackers. It can be argued, however, that $\mathsf{IND\text{-}CCA}$ security alone might not be enough in practice, where 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 guarantee the security of the exchanged keys even if adversaries can observe and attack many public keys and/or ciphertexts: this is *multi-challenge security*. However, multi-challenge security is not a security goal that is automatically achieved by the FO transform. This was recently exemplified by an attack on the NIST PQC Round 3 alternate $\mathsf{FrodoKEM}$, which could recover the shared secret of one of a large (but not unfeasibly large) amount of collected ciphertexts. The attack made use of the fact that the KEM was built by applying FO to a public key encryption (PKE) scheme whose message space was small enough to permit the attack. While this attack formally falls out of the scope of $\mathsf{IND\text{-}CCA}$ security, it nonetheless exposes an important vulnerability, which motivates the study into techniques for establishing multi-challenge 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 some encryption randomness r, instead it computes

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

where G is a hash function, modelled 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.)

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

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

for a randomly chosen message m. Decaps likewise, will return $\mathsf{H}(m)$ 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 $\mathbf{ss}$ – $\mathsf{U}^{\perp}_{m}$ 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-challenge security refers to attack models parameterized by the number of users n_{u}, and the number of challenge ciphertexts n_{c}. A multi-ciphertext attack refers to the case that $n_{\mathsf{u}} = 1$, while $n_{\mathsf{c}} > 1$. Likewise, multi-user security refers to the scenario where $n_{\mathsf{u}} > 1$ and $n_{\mathsf{c}} = n_{\mathsf{u}}$. A general multi-challenge attack may have $n_{\mathsf{u}} > 1$ and $n_{\mathsf{c}} > n_{\mathsf{u}}$.

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 exemplified by FrodoKEM-640 as proposed in [NAB+20]. This multi-ciphertext attack was identified privately by NIST [NIS21] and publicly by Bernstein [Ber22], which led to an update of FrodoKEM [ABD+23]. The same sort of attack was identified by NIST against HQC [AAB+22].

Suppose an attacker $\mathcal{A}$ collected n_{c} many 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 reasonable amount of challenges (think: 2^{64}), success is very likely: $\mathcal{A}$ can prepare n_{c} ciphertexts $c'_1, \ldots, c'_{n_{\mathsf{c}}}$ on their own by picking uniformly random messages and encrypting them under the user's public key . 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 $c'_1, \ldots, c'_{n_{\mathsf{c}}}$, $\mathcal{A}$ 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 $c'_1, \ldots, c'_{n_\mathsf{c}}$. Over the n_c many challenge ciphertexts, the probability of $\mathcal{A}$ sampling a collision is thus

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

Assuming that $\mathcal{A}$ was given up to $n_\mathsf{c} = 2^{64}$ many challenges and that $\mathcal{A}$ can feasibly compute $N = 2^{64}$ many encryptions on its own, $\mathcal{A}$ is able to find a collision with almost-certain probability and thus completely breaks the multi-challenge security of KEM.

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. The $\mathsf{FrodoKEM}$ update [ABD+23] thus presented an FO variant, called the salted Fujisaki–Okamoto (SFO) transform, that aimed to mitigate collisions based on too-small message spaces by increasing the search space via salting: SFO takes the T-transformed encryption scheme and slightly alters how the encryption randomness is derived. Instead of only hashing the message, SFO additionally hashes a uniformly random salt of length $\mathsf{len}_\mathsf{salt}$ (which is then communicated along with the ciphertext). While the Frodo team's 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 achieved multi-challenge security.

1.1 Our contributions

In this work, we prove that the salted Fujisaki–Okamoto (SFO) transform produces a KEM achieving multi-challenge security against chosen ciphertext attacks, assuming multi-challenge security against chosen plaintext in the underlying PKE. We show that for suitably chosen parameters—such as those used in $\mathsf{FrodoKEM}$ and HQC—the SFO transform generically achieves multi-challenge security since the probability of critical collisions significantly decreases.

We show 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-user-multi-challenge security for n_u many users and n_c many ciphertexts via security notions we denote by $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ and $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$. We base $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$ security of the SFO-constructed KEM on the $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ security of its underlying encryption scheme. The resulting security bounds contains a term that reflects *collision attacks*: let n_c be the number of challenge ciphertexts an attacker is given access to, and let q_RO represent the number of ROM queries. We now look at the success probability of an attacker $\mathcal{A}$ that aims to craft valid ciphertexts that collide with a challenge. This proves to depend

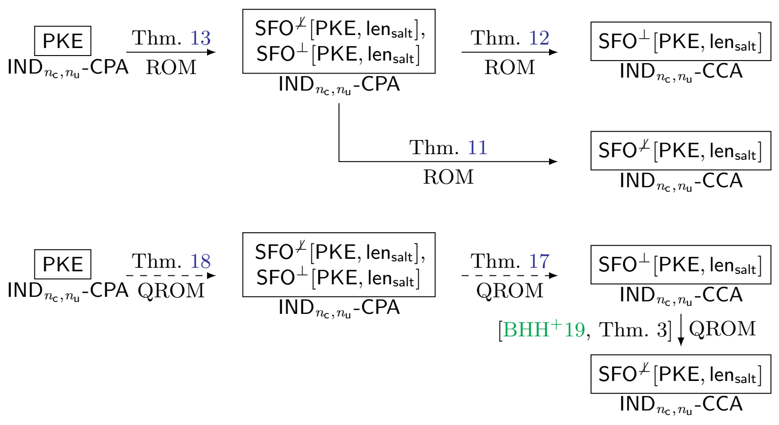
PKE
IND$_{n_c,n_u}$-CPA
Thm. 13
ROM
SFO$^{\not\perp}$[PKE, len$_{salt}$], SFO$^{\perp}$[PKE, len$_{salt}$]
IND$_{n_c,n_u}$-CPA
Thm. 12
ROM
SFO$^{\perp}$[PKE, len$_{salt}$]
IND$_{n_c,n_u}$-CCA
Thm. 11
ROM
SFO$^{\not\perp}$[PKE, len$_{salt}$]
IND$_{n_c,n_u}$-CCA
PKE
IND$_{n_c,n_u}$-CPA
Thm. 18
QROM
SFO$^{\not\perp}$[PKE, len$_{salt}$], SFO$^{\perp}$[PKE, len$_{salt}$]
IND$_{n_c,n_u}$-CPA
Thm. 17
QROM
SFO$^{\perp}$[PKE, len$_{salt}$]
IND$_{n_c,n_u}$-CCA
[BHH+19, Thm. 3]
QROM
SFO$^{\not\perp}$[PKE, len$_{salt}$]
IND$_{n_c,n_u}$-CCA

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

on q_{RO} for probabilistic schemes: for such schemes, FO constructions tie the encryption randomness to the message by setting it to $r := \mathsf{G}(m)$. It is thus prohibitively hard to craft valid ciphertexts without a query to the random oracle that models G. For the standard FO transform, the probability of $\mathcal{A}$ successfully crafting a valid, colliding ciphertext would be upper-bounded by

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

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

$$\frac{n_{\mathsf{c}}^2}{|\mathcal{M}|2^{\mathsf{len_{salt}}}} + \frac{2n_{\mathsf{c}}q_{\mathsf{RO}}}{|\mathfrak{L}_S||\mathcal{M}|} \quad , \tag{2}$$

where $\mathfrak{L}_S$ is the set of salts that were sampled when creating the challenges. ($|\mathfrak{L}_S|$ can be smaller than n_{c} since a salt might be sampled more than once.) 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 attacker advantage in Equation (1) against an FO-transformed scheme becomes close to 1. On the other hand, an SFO-transformed scheme can still tightly achieve multi-ciphertext security: taking $\mathsf{len_{salt}} = 64$, and noting that $|\mathfrak{L}_S| \approx n_{\mathsf{c}}$, the attacker advantage against an SFO-transformed scheme in Equation (2) remains small.

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, but the QROM results are not.

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

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 $(\mathsf{pk}, \mathsf{sk})$, with pk also defines 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 $\mathbf{ss}$. The decapsulation algorithm Decaps, on input sk and an encapsulation c, outputs either $\mathbf{ss} = \mathsf{Decaps}(\mathsf{sk}, c)$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* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with message space* $\mathcal{M}$ *is called* δ*-correct if*

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

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

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

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

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 PKE's and KEM's, except that we take the maximum probability over all $j \in [n_{\mathsf{u}}]$ [DHK+21a].

```
$\mathsf{T.Gen}$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}$
$\mathsf{sk}' \leftarrow (\mathsf{sk}, \mathsf{pk})$
return $(\mathsf{pk}, \mathsf{sk}')$

$\mathsf{T.Enc}(\mathsf{pk}, m)$:
$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; \mathsf{G}(m))$
return $c$

$\mathsf{T.Dec}(\mathsf{sk}' = (\mathsf{pk}, \mathsf{sk}), c)$:
$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
if $m' = \perp$ or $c \neq \mathsf{T.Enc}(\mathsf{pk}, m')$
    return $\perp$
else
    return $m'$
```

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

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

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

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

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 $\mathsf{U}^{\not\perp}$ 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 KEM* $\mathsf{KEM}^{\not\perp} = \mathsf{U}^{\not\perp}[\mathsf{PKE}, \mathsf{H}]$ *is defined in Figure 3.*

Definition 6 (Augmented $\mathsf{U}^{\perp}$ 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 KEM* $\mathsf{KEM}^{\perp} = \mathsf{U}^{\perp}[\mathsf{PKE}, \mathsf{H}]$ *is defined in Figure 4.*

```
$\mathsf{KEM}^{\not\perp}.\mathsf{Gen}()$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{Gen}()$
$s \leftarrow_\$ \mathcal{M}$
$\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$
return $(\mathsf{pk}, \mathsf{sk}')$
```

```
$\mathsf{KEM}^{\not\perp}.\mathsf{Encaps}(\mathsf{pk}, m)$
$m \leftarrow_\$ \mathcal{M}$
$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$
ss $\leftarrow \mathsf{H}(\mathsf{pk}, m, c)$
return $(K, c)$
```

```
$\mathsf{KEM}^{\not\perp}.\mathsf{Decaps}(\mathsf{sk}, c)$
parse $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$
$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
if $m' = \perp$
  return $\mathsf{H}(\mathsf{pk}, s, c)$
return $\mathsf{H}(\mathsf{pk}, m', c)$
```

Fig. 3: KEM $\mathsf{KEM}^{\not\perp}$ constructed by the augmented $\mathsf{U}^{\not\perp}$ transform. The only difference from the original $\mathsf{U}^{\not\perp}$ transform is that when deriving the session key, we additionally hash in pk.

```
$\mathsf{KEM}^{\perp}.\mathsf{Decaps}(\mathsf{sk}, c)$
parse $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$
$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
if $m' = \perp$
  return $\perp$
return $\mathsf{H}(\mathsf{pk}, m', c)$
```

Fig. 4: Decapsulation algorithm for the KEM $\mathsf{KEM}^{\perp}$ constructed by the augmented $\mathsf{U}^{\perp}$ transform; the key generation and encapsulation algorithms are the same as in Figure 3. The only difference from the original $\mathsf{U}^{\not\perp}$ transform is that when deriving the session key, we additionally hash in pk.

2.3 Multi-challenge security notions

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

Definition 7 (Multi-challenge IND-CPA security ($\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$) 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 $\mathrm{Exp}_{\mathsf{PKE}}^{\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{CPA}}(\mathcal{A})$ shown in Fig. 5. The $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ advantage function of an adversary $\mathcal{A}$ against PKE is*

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

Definition 8 (Multi-challenge IND-CCA security($\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$) 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 $\mathrm{Exp}_{\mathsf{KEM}}^{\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{CCA}}(\mathcal{A})$ shown in Fig. 6. The $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$ advantage function of an adversary $\mathcal{A}$ against KEM is*

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

```
$\mathrm{Exp}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A})$
01 $b \leftarrow_\$ \{0,1\}$
02 for $j = 1, \ldots, n_u$
03     $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow_\$ \mathsf{Gen}()$
04 $\vec{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_{n_u})$
05 $b' \leftarrow \mathcal{A}^{\textsc{Chall}}(\vec{pk})$
06 return $[\![b = b']\!]$

$\textsc{Chall}(j, m_0, m_1)$    // At most $n_c$ queries
07 return $\mathsf{Enc}(\mathsf{pk}_j, m_b)$
```

Fig. 5: Multi-challenge security experiment (IND_{n_c,n_u}-CPA) against a public key encryption scheme PKE, with n_u users and n_c ciphertexts.

```
$\mathrm{Exp}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{A})$
01 $b \leftarrow_\$ \{0,1\}$
02 for $j = 1, \ldots, n_u$
03     $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow_\$ \mathsf{Gen}()$
04 $\vec{\mathsf{pk}} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_{n_u})$
05 $b' \leftarrow \mathcal{A}^{\textsc{Decaps},\textsc{Chall}}(\vec{pk})$
06 return $[\![b = b']\!]$

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

$\textsc{Decaps}(j, c \notin \mathfrak{L}_{C_j})$
11 return $\mathsf{Decaps}(\mathsf{sk}_j, c)$
```

Fig. 6: Multi-challenge security experiment (IND_{n_c,n_u}-CCA) against a key encapsulation mechanism KEM, with n_u users and n_c encapsulations.

Omitting the oracle DECAPS on line 05 in Figure 6 yields IND_{n_c,n_u}-CPA security for a KEM.

Prefix 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+21a] 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 $\mathsf{ID} : \mathcal{PK} \to \{0,1\}^\ell$. 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 raises a flag COLL and aborts if there are colliding public key identifiers, namely if there are indices $i \neq j$ with $\mathsf{F}(\mathsf{pk}_i) = \mathsf{F}(\mathsf{pk}_j)$. By the birthday bound, the probability of COLL is $n_u^2/2^\ell$. Provided that flag COLL is not raised, 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 in a bound that only differs from the original one by the term $n_u^2/2^\ell$.

3 The Salted FO transform

In this section, we formalize the salting countermeasure introduced in [ABD+23] to thwart pre-computation attacks on multi-ciphertext security. The countermeasure introduces a uniformly random salt in the encryption process, so that the change 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 9 (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* $\mathsf{len_{salt}}$ *be a non-negative integer. To* PKE*,* G *and* $\mathsf{len_{salt}}$ *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. 7.*

```
ST[PKE, G].Enc_1(pk, $m$)
    $\mathsf{salt} \leftarrow_\$ \{0,1\}^{\mathsf{len_{salt}}}$
    $r \leftarrow \mathsf{G}(\mathsf{pk}, m \| \mathsf{salt})$
    $c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m; r)$
    return $c \| \mathsf{salt}$

ST[PKE, G].Dec_1($sk$, $c \| \mathsf{salt}$)
    $m' \leftarrow \mathsf{PKE.Dec}(sk, c)$
    $r' \leftarrow \mathsf{G}(pk, m \| \mathsf{salt})$
    if $m' = \perp$ or $\mathsf{PKE.Enc}(\mathsf{pk}, m'; r') \neq c$
        return $\perp$
    else return $m'$
```

Fig. 7: 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 10 ($\mathsf{SFO}^{\not\perp}$ and $\mathsf{SFO}^\perp$ transforms). *For a public key encryption scheme* PKE*, a salt length parameter* $\mathsf{len_{salt}}$*, and hash functions* G*,* H*, the salted FO transforms yields the KEMs*

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

with algorithms described in Figure 8 and Figure 9 respectively.

```
$\mathsf{KEM}^{\not\perp}.\mathsf{Gen}()$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}()$
$s \leftarrow\!\!\$\ \mathcal{M}$
$\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$
return $(\mathsf{pk}, \mathsf{sk}')$
```

```
$\mathsf{KEM}^{\not\perp}.\mathsf{Encaps}(\mathsf{pk}, m)$
$\mathsf{salt} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$
$r \leftarrow \mathsf{G}(\mathsf{pk}, m \| \mathsf{salt})$
$c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m; r)$
$\mathsf{ss} \leftarrow \mathsf{H}(\mathsf{pk}, m, c)$
return $(\mathsf{ss}, c \| \mathsf{salt})$
```

```
$\mathsf{KEM}^{\not\perp}.\mathsf{Decaps}(\mathsf{sk}, c \| \mathsf{salt})$
parse $\mathsf{sk}' = (\mathsf{sk}, s)$
$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$r \leftarrow \mathsf{G}(\mathsf{pk}, m \| \mathsf{salt})$
$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$
if $m' = \perp$ or $c \neq c'$
  return $\mathsf{H}(\mathsf{pk}, s, c)$
return $\mathsf{H}(\mathsf{pk}, m', c)$
```

Fig. 8: KEM $\mathsf{KEM}^{\not\perp}$ constructed by the salted FO transform $\mathsf{SFO}^{\not\perp}$.

```
$\mathsf{KEM}^{\perp}.\mathsf{Gen}()$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}()$
return $(\mathsf{pk}, \mathsf{sk})$
```

```
$\mathsf{KEM}^{\perp}.\mathsf{Encaps}(\mathsf{pk}, m)$
$\mathsf{salt} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$
$r \leftarrow \mathsf{G}(\mathsf{pk}, m \| \mathsf{salt})$
$c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m; r)$
$\mathsf{ss} \leftarrow \mathsf{H}(\mathsf{pk}, m, c)$
return $(\mathsf{ss}, c \| \mathsf{salt})$
```

```
$\mathsf{KEM}^{\perp}.\mathsf{Decaps}(\mathsf{sk}, c \| \mathsf{salt})$
parse $\mathsf{sk}' = (\mathsf{sk}, s)$
$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
$r \leftarrow \mathsf{G}(\mathsf{pk}, m \| \mathsf{salt})$
$c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$
if $m' = \perp$ or $c \neq c'$
  return $\perp$
return $\mathsf{H}(\mathsf{pk}, m', c)$
```

Fig. 9: KEM $\mathsf{KEM}^{\perp}$ constructed by the salted FO transform $\mathsf{SFO}^{\perp}$.

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 $\mathsf{U}_m^{\perp}$ or $\mathsf{U}_m^{\not\perp}$ from [HHK17].

Bindel et al. [BHH+19] show that in the single-challenge setting, using $\mathsf{H}(m)$ as a KEM key is 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

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

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 $\mathsf{H}_2(m) \leftarrow \mathsf{H}_1(m, \mathsf{Enc}(\mathsf{pk}, m))$.

This argument does not immediately extend to the multi-challenge setting, as there would be u public keys to consider. In fact, $\mathsf{H}(m)$ is in no way connected to the public key of any user, while $\mathsf{H}(m, c \leftarrow \mathsf{Enc}(\mathsf{pk}_j, m)$ is. However, if the public key hash is included, the argument does hold, with the simulation being

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

The difference being that now the adversary is required to have knowledge of the public key in either case.

4 Security proofs in the ROM

We prove $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$ security for KEM's built from the SFO transform, applied to $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ KEM's. 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 KEM's. Combining, we obtain $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$ security bounds. Then we prove that the SFO transform tightly preserves $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ security.

4.1 Simulating the decapsulation oracle in the ROM: $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-CPA to $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-CCA

In the following theorems, we use known techniques to simulate decapsulation oracles with either implicit or explicit rejection. These techniques have been used in [Den03b, HHK17, DHK+21a, HHM22, HM24], usually in conjunction with other proofs 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 the 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.

Theorem 11 (Simulation of $\textsc{Decaps}^{\not\perp}$). *Let* $\mathsf{KEM}^{\not\perp} = \mathsf{SFO}^{\not\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$. *For any* $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$ *adversary* $\mathcal{B}$ *against* $\mathsf{KEM}^{\not\perp}$, *issuing at most* q_H *queries to* H, q_G *queries to* G *and* q_D *queries to* $\textsc{Decaps}^{\not\perp}$, *there exists an* $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ *adversary* $\mathcal{A}$ *against* $\mathsf{KEM}^{\not\perp}$ *such that*

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

Proof. Let G and $\textsc{Chall}$ be defined as in $\mathbf{G}_0$ in the proof of theorem 1. Furthermore, let $\mathfrak{L}_{C_j}$ denote the list of challenge ciphertexts for a user j. Consider the sequence of games shown in Figure 11.

$\mathbf{G}_0$ is the IND_{n_c,n_u}-CCA-game against $\mathsf{KEM}^{\not\perp}$ and so

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

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

$$\begin{tikzcd}
\underset{\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{CPA}}{\boxed{\mathsf{PKE}}} \arrow[r, "\text{Thm. 13}", "\text{ROM}"'] & \underset{\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{CPA}}{\boxed{\begin{array}{c}\mathsf{SFO}^{\not\perp}[\mathsf{PKE},\mathsf{len}_\mathsf{salt}],\\ \mathsf{SFO}^{\perp}[\mathsf{PKE},\mathsf{len}_\mathsf{salt}]\end{array}}} \arrow[r, "\text{Thm. 12}", "\text{ROM}"'] \arrow[dr, rounded corners, "\text{Thm. 11}", "\text{ROM}"'] & \underset{\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{CCA}}{\boxed{\mathsf{SFO}^{\perp}[\mathsf{PKE},\mathsf{len}_\mathsf{salt}]}} \\
 & & \underset{\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-}\mathsf{CCA}}{\boxed{\mathsf{SFO}^{\not\perp}[\mathsf{PKE},\mathsf{len}_\mathsf{salt}]}}
\end{tikzcd}$$

Fig. 10: Summary of our results on the salted Fujisaki-Okamoto (SFO) transform in the ROM.

```
$\mathsf{H}(\mathsf{pk}_j, m, c\|\mathsf{salt})$
01 if $\exists$ss s.t. $(m, c\|\mathsf{salt}, \mathbf{ss}) \in \mathfrak{L}_{\mathsf{H}_j}$
02   return ss
03 ss $\leftarrow\$ \mathcal{K}$
04 if $m = s_j$                                   // $\mathbf{G}_1$-$\mathbf{G}_2$
05   QUERY = true; abort                         // $\mathbf{G}_1$-$\mathbf{G}_2$
06 if $\mathsf{Enc}_1(\mathsf{pk}_j, m) = c$                        // $\mathbf{G}_2$
07   if $\exists$ss$'$ s.t. $(c\|\mathsf{salt}, \mathbf{ss}') \in \mathfrak{L}_{D_j}$   // $\mathbf{G}_2$
08     ss := ss$'$                                 // $\mathbf{G}_2$
09   else                                        // $\mathbf{G}_2$
10     $\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c\|\mathsf{salt}, \mathbf{ss}\}$            // $\mathbf{G}_2$
11 $\mathfrak{L}_{\mathsf{H}_j} \leftarrow \mathfrak{L}_{\mathsf{H}_j} \cup \{(m, c\|\mathsf{salt}, \mathbf{ss})\}$
12 return ss

$\text{DECAPS}^{\not\perp}(j, c\|\mathsf{salt})$
13 if $c \in \mathfrak{L}_{C_j}$ abort
14 $m' \leftarrow \mathsf{Dec}(sk_j', c\|\mathsf{salt})$                      // $\mathbf{G}_0$-$\mathbf{G}_1$
15 if $m' = \perp$ or
   $\mathsf{Enc}(\mathsf{pk}_j, m'; \mathsf{G}(\mathsf{pk}_j, m'\|\mathsf{salt}) \neq c$      // $\mathbf{G}_0$-$\mathbf{G}_1$
16   ss $\leftarrow \mathsf{H}(\mathsf{pk}_j, s_j, c\|\mathsf{salt})$                  // $\mathbf{G}_0$
17   ss $\leftarrow \mathsf{H}'(\mathsf{pk}_j, s_j, c\|\mathsf{salt})$                 // $\mathbf{G}_1$
18 if $m' = s_j$                                    // $\mathbf{G}_1$
19   ss $\leftarrow \mathsf{H}'(\mathsf{pk}_j, m', c\|\mathsf{salt})$                 // $\mathbf{G}_1$
20 ss $\leftarrow \mathsf{H}(\mathsf{pk}_j, m', c\|\mathsf{salt})$                 // $\mathbf{G}_0$-$\mathbf{G}_1$
21 if $\exists$ss s.t. $(c\|\mathsf{salt}, \mathbf{ss}) \in \mathfrak{L}_{D_j}$             // $\mathbf{G}_2$
22   return ss                                     // $\mathbf{G}_2$
23 ss $\leftarrow\$ \mathcal{K}$                                       // $\mathbf{G}_2$
24 $\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c\|\mathsf{salt}, \mathbf{ss}\}$
25 return ss
```

Fig. 11: Simulation of a decapsulation oracle with implicit rejection for proof of Theorem 11.

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 $\text{DECAPS}^{\not\perp}$ 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 $\mathcal{B}$ queries G, with a j and $m\|\mathsf{salt}$ such that $m \neq \mathsf{Dec}(sk_j, \mathsf{Enc}(\mathsf{pk}_j, m; \mathsf{G}(m\|\mathsf{salt})))$, by $\delta(n_\mathsf{u})$-correctness, we have that

$$\Pr[\mathsf{CORR}] \leq (q_\mathsf{G} + q_\mathsf{H} + q_D) \cdot \delta(n_\mathsf{u})$$

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

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

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

– **Case 2:** $m \neq \perp$ and $c \neq c'$. In this case, $\text{Decaps}^{\not\perp}$ will output a uniformly random secret in either game. However, in $\mathbf{G}_2$ a query $\mathsf{H}(\mathsf{pk}_j, m, c\|\mathsf{salt})$ would return the same value as $\text{Decaps}^{\not\perp}(j, c\|\mathsf{salt})$, where in $\mathbf{G}_1$ $\text{Decaps}^{\not\perp}(j, c\|\mathsf{salt})$ would output $\mathsf{H}'(\mathsf{pk}_j, c\|\mathsf{salt})$, while $\mathsf{H}(\mathsf{pk}_j, m, c\|\mathsf{salt})$ 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 $\text{Decaps}^{\not\perp}$ is $\mathsf{H}(\mathsf{pk}_j, m', c\|\mathsf{salt})$. In $\mathbf{G}_2$ the output of $\text{Decaps}^{\not\perp}(j, c\|\mathsf{salt})$ is $\mathsf{H}(\mathsf{pk}_j, m, c\|\mathsf{salt})$ if $\mathsf{H}(\mathsf{pk}_j, m, c\|\mathsf{salt})$. Hence the output of the two games will only be different if $m \neq m'$.

By the difference lemma

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

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

Theorem 12 (Simulation of $\text{Decaps}^{\perp}$). *Let* $\mathsf{KEM}^{\perp} = \mathsf{SFO}^{\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$. *For any* $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-$\mathsf{CCA}$ *adversary* $\mathcal{B}$ *against* $\mathsf{KEM}^{\perp}$*, issuing at most* q_{H} *queries to* H*,* q_{G} *queries to* G *and* q_D *queries to* $\text{Decaps}^{\perp}$*, there exists an* $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-$\mathsf{CPA}$ *adversary* $\mathcal{A}$ *against* $\mathsf{KEM}^{\perp}$ *such that*

$$\text{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CCA}}_{\mathsf{KEM}^{\perp}}(\mathcal{B}) \leq \text{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CPA}}_{\mathsf{KEM}^{\perp}}(\mathcal{A}) + q_D 2^{-\gamma} + (q_{\mathsf{G}} + q_{\mathsf{H}} + q_D) \cdot \delta(n_{\mathsf{u}})$$

Proof. Let G and Chall be defined as in $\mathbf{G}_0$ in the proof of theorem 1. Furthermore, let $\mathfrak{L}_{C_j}$ denote the list of challenge ciphertexts for a user j. Consider the sequence of games shown in Figure 12

$\mathbf{G}_0$ is the $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-$\mathsf{CCA}$-game against $\mathsf{KEM}^{\not\perp}$ and so

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

In $\mathbf{G}_1$ we patch the random oracles so that Decaps and H always return the same thing. Even in the event of an correctness error, the view of the adversary is identical, since $\text{Decaps}^{\perp}$ implicitly makes a CVO query in line 13, and will return $\perp$ on a correctness error, as in $\mathbf{G}_2$

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

In $\mathbf{G}_2$, we modify $\text{Decaps}^{\perp}$, so that instead of checking that $m' \in \mathcal{M}$, it instead checks that $\mathsf{G}(m'\|\mathsf{salt})$ had been queried, and then $\mathsf{Enc}(\mathsf{pk}_j, m'; \mathsf{G}(m'\|\mathsf{salt})) = c'$. The two games proceed identically, unless $\mathcal{B}$ was able to produce a ciphertext c such that $c = \mathsf{Enc}(\mathsf{pk}_j, m'; \mathsf{G}(m'))$ without querying $\mathsf{G}(m')$. In this case in $\mathbf{G}_2$ $\text{Decaps}^{\perp}(j, c\|\mathsf{salt})$ would return $\perp$, while in $\mathbf{G}_1$ $\text{Decaps}^{\perp}(\mathbf{ss}, c\|\mathsf{salt})$ would return $\mathsf{H}(\mathsf{pk}_j, m', c\|\mathsf{salt})$.

This implies that $\mathcal{B}$ found $r \neq \mathsf{G}(m'\|\mathsf{salt})$, such that $\mathsf{Enc}(\mathsf{pk}, m; \mathsf{G}(m'\|\mathsf{salt})) = \mathsf{Enc}(\mathsf{pk}, m; r)$. For a single decapsulation query, this occurs with probability $2^{-\gamma}$,

```
$\underline{\mathsf{H}(\mathsf{pk}_j, m, c\|\mathsf{salt})}$
if $\exists \mathbf{ss}$ s.t. $(m, c\|\mathsf{salt}, \mathbf{ss}) \in \mathfrak{L}_{\mathsf{H}_j}$
  return ss
$\mathbf{ss} \leftarrow\!\!\$\ \mathcal{K}$
if $\mathsf{Enc}_1(\mathsf{pk}_j, m) = c$                              // $\mathbf{G}_1$-$\mathbf{G}_3$
  if $\exists \mathbf{ss}'$ s.t. $(c\|\mathsf{salt}, \mathbf{ss}') \in \mathfrak{L}_{D_j}$      // $\mathbf{G}_1$-$\mathbf{G}_3$
    $\mathbf{ss} := \mathbf{ss}'$                                        // $\mathbf{G}_1$-$\mathbf{G}_3$
  else                                                  // $\mathbf{G}_1$-$\mathbf{G}_3$
    $\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c\|\mathsf{salt}, \mathbf{ss}\}$              // $\mathbf{G}_1$-$\mathbf{G}_3$
$\mathfrak{L}_{\mathsf{H}_j} \leftarrow \mathfrak{L}_{\mathsf{H}_j} \cup \{(m, c\|\mathsf{salt}, \mathbf{ss})\}$
return ss
```

```
$\underline{\textsc{Decaps}^{\perp}(j, c\|\mathsf{salt})}$
if $c \in \mathfrak{L}_{C_j}$ abort
$m' \leftarrow \mathsf{Dec}(sk_j', c\|\mathsf{salt})$                  // $\mathbf{G}_0$-$\mathbf{G}_2$
if $m' = \perp$ or
$\mathsf{Enc}(\mathsf{pk}_j, m'; \mathsf{G}(\mathsf{pk}_j, m'\|\mathsf{salt}) \neq c$ // $\mathbf{G}_0$-$\mathbf{G}_1$
  $\mathbf{ss} \leftarrow \perp$                                  // $\mathbf{G}_0$-$\mathbf{G}_1$
$\mathbf{ss} \leftarrow \mathsf{H}(\mathsf{pk}_j, m', c\|\mathsf{salt})$                   // $\mathbf{G}_0$
if $\nexists (m', r) \in \mathfrak{L}_{\mathsf{G}_j}$ s.t.
  $\mathsf{Enc}(\mathsf{pk}, m; r\|\mathsf{salt}) = c$                    // $\mathbf{G}_2$
  return $\perp$                                      // $\mathbf{G}_2$
if $\nexists (m, r) \in \mathfrak{L}_{\mathsf{G}_j}$ s.t.
  $\mathsf{Enc}(\mathsf{pk}, m; r\|\mathsf{salt}) = c$                    // $\mathbf{G}_3$
  return $\perp$                                      // $\mathbf{G}_3$
if $\exists \mathbf{ss}$ s.t. $(c\|\mathsf{salt}, \mathbf{ss}) \in \mathfrak{L}_{D_j}$ // $\mathbf{G}_1$-$\mathbf{G}_3$
  return ss                                  // $\mathbf{G}_1$-$\mathbf{G}_3$
$\mathbf{ss} \leftarrow\!\!\$\ \mathcal{K}$                                   // $\mathbf{G}_1$-$\mathbf{G}_3$
$\mathfrak{L}_{D_j} \leftarrow \mathfrak{L}_{D_j} \cup \{c\|\mathsf{salt}, \mathbf{ss}\}$
return ss
```

Fig. 12: Simulation of a decapsulation oracle with explicit rejection for proof of Theorem 12.

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 $\mathbf{G}_4$ we remove all dependence on the secret key, by no longer checking decryption c during decapsulation. Instead, we check for queries to $\mathsf{G}(\mathsf{pk}_j, -)$ 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 Theorem 11, , as there are $q_{\mathsf{G}} + q_D + q_{\mathsf{H}}$ implicit and explicit queries to G we have

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

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

4.2 Tight passive security of SFO

In Theorem 13 below we show that SFO *tightly* turns multi-challenge security of PKE into (for now passive) multi-challenge security of the salted KEM.

Bookkeeping salts. In the following proofs challenges are produced using an oracle $\textsc{Chall}$ which takes as input an index $j \in [n_{\mathsf{u}}]$. $\mathfrak{L}_S$ denotes the list of distinct salts sampled during challenge oracle queries as, and $\mathfrak{L}_{S_j}$ denotes the list of salts sampled for a specific user j. Likewise, n_{c} denotes the total queries to Chall, while $n_{\mathsf{c}j}$ denotes the number of challenge queries to a specific user - Chall(j).

The expected number of distinct salt values for $n_{\mathsf{c}j}$ queries to Chall(j) is

$$\mathbb{E}\left(|\mathfrak{L}_{S_j}|\right) = 2^{\mathsf{len}_{\mathsf{salt}}}\left(1 - \left(1 - \frac{1}{2^{\mathsf{len}_{\mathsf{salt}}}}\right)^{n_{\mathsf{c}j}}\right).$$[4]

Repeated sampling of salts introduces collisions in $\mathfrak{L}_S$, so that, with high probability, $|\mathfrak{L}_S| < n_{\mathsf{c}}$ when $n_{\mathsf{c}} \geq \sqrt{2^{\mathsf{len}_{\mathsf{salt}}}}$ by the birthday paradox. As $n_{\mathsf{c}j}$ increases, the rate of collisions in $\mathfrak{L}_{S_j}$ grows quadratically. We make use of a simplifying assumption, that

$$\sum_j \frac{n_{\mathsf{c}j}}{|\mathfrak{L}_{S_j}|} \leq \frac{n_{\mathsf{c}}}{|\mathfrak{L}_S|}.$$

One can see that in the case where all challenges were issued to the same user (i.e., there exists a j so that $\mathfrak{L}_S = \mathfrak{L}_{S_j}$), where the rate of salt collisions is maximized.

Theorem 13 (PKE IND_{n_c,n_u}**-CPA** $\xRightarrow{\textbf{ROM}}$ $\mathsf{SFO}^{\not\perp}_{m,c}[\mathsf{PKE}, \mathsf{len}_{\mathsf{salt}}]$ IND_{n_c,n_u}**-CPA).** *Let* PKE *be a public-key encryption scheme, and let* KEM *be the KEM constructed as* KEM := $\mathsf{SFO}^{\not\perp}_{m,c}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$*. If* PKE *is* $\delta(n_{\mathsf{u}})$*-correct, then* KEM *is* $\delta(n_{\mathsf{u}})$*-correct in the random oracle model. Moreover for any* 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* q_D *queries to* Decaps*, there exists an* IND_{n_c,n_u}-CPA *adversary* $\mathcal{A}$ *against* PKE *such that*

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{KEM}}(\mathcal{B}) \leq \frac{n_{\mathsf{c}}^2}{|\mathcal{M}|2^{\mathsf{len}_{\mathsf{salt}}}} + \frac{2n_{\mathsf{c}}(q_{\mathsf{G}} + q_{\mathsf{H}})}{|\mathcal{M}||\mathfrak{L}_S|} + 2 \cdot \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A}) \ ,$$

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

The same bound holds for the KEM $\mathsf{KEM}^{\perp}$:= $\mathsf{SFO}^{\perp}_{m,c}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$ *since they only differ in their decapsulation algorithm (which is not used in the* IND_{n_c,n_u}-CPA *experiment).*

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 $\mathbf{ss}_0 = \mathsf{H}(\mathsf{pk}, m, c)$ if they already queried H on (pk, m, c). To make such queries harder to achieve, we exploit IND_{n_c,n_u}-CPA 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

[4] This uses a generic result from discrete probability: expected number of distinct values when sampling n times, from a set of size N, with replacement. Here $N = 2^{\mathsf{len}_{\mathsf{salt}}}$ and $n = n_{\mathsf{c}j}$.

```
$\mathbf{G}_0$-$\mathbf{G}_2$
01 $b \leftarrow_\$ \{0,1\}$
02 for $j = 1, ..., u$
03   $(\mathsf{pk}_j, sk_j) \leftarrow_\$ \mathsf{Gen}()$
04 $\vec{pk} = (\mathsf{pk}_1, ..., pk_j)$
05 $b' \leftarrow \mathcal{B}^{\textsc{Chall},\mathsf{G},\mathsf{H}}(\vec{pk})$
06 return $[\![b = b']\!]$

$\mathsf{G}(\mathsf{pk}_j, m\|\mathsf{salt})$
07 if $\exists r$ s.t. $(m\|\mathsf{salt}, r) \in \mathfrak{L}_{\mathsf{G}_j}$
08   return $r$
09 if $m\|\mathsf{salt} \in \mathfrak{L}_{M_j}$                    // $\mathbf{G}_2$
10   $\mathsf{CHAL} = \mathbf{true}$; abort                   // $\mathbf{G}_2$
11 $r \leftarrow_\$ \mathcal{R}$
12 $\mathfrak{L}_{\mathsf{G}_j} \leftarrow \mathfrak{L}_{\mathsf{G}_j} \cup \{(m\|\mathsf{salt}, r)\}$
13 return $r$

$\textsc{Chall}(j)$ // at most $n_\mathsf{c}$ queries
14 $(m\|\mathsf{salt}) \leftarrow_\$ \mathcal{M} \times \{0,1\}^{\mathsf{len}_\mathsf{salt}}$          // $\mathbf{G}_0$
15 $r \leftarrow \mathsf{G}(\mathsf{pk}_j, m\|\mathsf{salt})$                    // $\mathbf{G}_0$-$\mathbf{G}_1$
16 $r \leftarrow_\$ \mathcal{R}$                                          // $\mathbf{G}_2$
17 $c = \mathsf{Enc}\left(\mathsf{pk}_j, m; r\right)$
18 if $(m\|\mathsf{salt}) \in \mathfrak{L}_{M_j}$                     // $\mathbf{G}_1 - \mathbf{G}_2$
19   return $(c\|\mathsf{salt}, \mathbf{ss} \leftarrow_\$ \mathcal{K})$          // $\mathbf{G}_1 - \mathbf{G}_2$
20 $\mathfrak{L}_{M_j} \leftarrow \mathfrak{L}_{M_j} \cup \{m\|\mathsf{salt}\}$
21 $\mathbf{ss}_0 = \mathsf{H}\left(\mathsf{pk}_j, m, c\|\mathsf{salt}\right)$               // $\mathbf{G}_0$-$\mathbf{G}_1$
22 $\mathbf{ss}_0 \leftarrow_\$ \mathcal{K}$                                       // $\mathbf{G}_2$
23 $\mathbf{ss}_1 \leftarrow_\$ \mathcal{K}$
24 return $(c\|\mathsf{salt}, \mathbf{ss}_b)$

$\mathsf{H}(\mathsf{pk}_j, m, c\|\mathsf{salt})$
25 if $\exists \mathbf{ss}$ s.t. $(m, c\|\mathsf{salt}, \mathbf{ss}) \in \mathfrak{L}_{\mathsf{H}_j}$
26   return $\mathbf{ss}$
27 if $m\|\mathsf{salt} \in \mathfrak{L}_{M_j}$                              // $\mathbf{G}_2$
28   $\mathsf{CHAL} = \mathbf{true}$; abort                             // $\mathbf{G}_2$
29 $\mathbf{ss} \leftarrow_\$ \mathcal{K}$
30 $\mathfrak{L}_{\mathsf{H}_j} \leftarrow \mathfrak{L}_{\mathsf{H}_j} \cup \{(m, c\|\mathsf{salt}, \mathbf{ss})\}$
31 return $\mathbf{ss}$
```

Fig. 13: Games for the proof of Theorem 13.

obtain $c := \mathsf{Enc}(\mathsf{pk}_j, m_b)$, 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 $r := \mathsf{G}(\mathsf{pk}, m\|\mathsf{salt})$. 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.

Proof. Consider the sequence of games shown in Fig. 13.

Game $\mathbf{G}_0$. $\mathbf{G}_0$ is the original $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$-game against the KEM constructed as $\mathsf{SFO}^{\not\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_\mathsf{salt}]$.

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

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 19. Since the games only differ if such a resampling of message/salt pairs occurs and since the total number of challenge messages is n, the birthday bound yields

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

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

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

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 CHAL.

Bounding CHAL. To this end, we construct an $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ adversary $\mathcal{A}$ against PKE that perfectly simulates $\mathbf{G}_2$ for $\mathcal{B}$ and wins if CHAL 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 CHALL. To simulate CHALL for index 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 $m_1\|\mathsf{salt}$ to $\mathcal{M}_{j1}$.

Then $\mathcal{A}$ queries its CPA challenge oracle on input (j, m_0, m_1) to obtain $c = \mathsf{Enc}(\mathsf{pk}, m_b)$. $\mathcal{A}$ 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 CHALL in $\mathbf{G}_2$.

We will now discuss how $\mathcal{A}$ can win their own game. Intuitively, $\mathcal{A}$ recognizes any query triggering CHAL by looking through the two bookkeeping lists $\mathcal{M}_{j0}$ and $\mathcal{M}_{j1}$ - if CHAL 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 $(\mathsf{pk}_j, m\|\mathsf{salt})$ 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 main remaining issue is that $\mathcal{B}$ might derail $\mathcal{A}$'s guess by querying G on a message $m \in \mathfrak{L}_{M_{j,1-b}}$, so on a message that has nothing to do with the ciphertexts $\mathcal{B}$

received. We denote this event by BADG, that is, we let BADG be the event that $\mathcal{B}$ queries G on any $(\mathsf{pk}_j, m\|\mathsf{salt})$ or H on $(\mathsf{pk}_j, m, c\|\mathsf{salt})$ such that $m\|\mathsf{salt} \in \mathfrak{L}_{M_{j,1-b}}$. Since these messages are completely independent of $\mathcal{B}$'s view and since the right salt must be chosen from the user's overall list of salts $\mathfrak{L}_{S_j}$ to trigger BADG,

$$\Pr[\mathsf{BADG}] = \sum_{j\in[u]} \frac{n_{\mathsf{c}j}\cdot(q_{\mathsf{H}}+q_{\mathsf{G}})}{|\mathcal{M}\|\mathfrak{L}_{S_j}|} \le \frac{n_{\mathsf{c}}\cdot(q_{\mathsf{H}}+q_{\mathsf{G}})}{|\mathcal{M}\|\mathfrak{L}_{S}|}.$$

If CHAL is raised, but BADG is not, then $\mathcal{A}$ accurately detects that a random oracle query containing $m \in \mathcal{M}_{jb}$ triggered CHAL, returns b, and wins. If neither CHAL nor BADG is raised, $\mathcal{A}$ returns a random guess, as is the case when both CHAL and BADG are raised. So, provided BADG is not raised, $\mathcal{A}$ wins the game with probability 1 if CHAL is raised and with probability $\frac{1}{2}$ otherwise. Hence

$$\begin{aligned}\mathrm{Adv}_{\mathsf{PKE}}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CPA}}(\mathcal{A}) + \Pr[\mathsf{BADG}] &\ge \left|\Pr[b=b'|\neg\mathsf{BADG}] - \frac{1}{2}\right| \\ &= \left|\Pr[\mathsf{CHAL}] + \frac{1}{2}\Pr[\neg\mathsf{CHAL}] - \frac{1}{2}\right| \\ &= \frac{1}{2}\Pr[\mathsf{CHAL}].\end{aligned}$$

And so

$$\Pr[\mathsf{CHAL}] \le \frac{2n_{\mathsf{c}}\cdot(q_{\mathsf{H}}+q_{\mathsf{G}}+2q_D)}{|\mathcal{M}\|\mathfrak{L}_S|} + 2\cdot\mathrm{Adv}_{\mathsf{PKE}}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-}\mathsf{CPA}}(\mathcal{A}).$$

Adding up the terms gives the desired result. □

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 Section 5.1.

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 [DFMS21]. 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 eCO.Ext. Intuitively, that additional interface 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. eCO.Ext takes as input a classical target value $t \in T$. Intuitively, 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 eCO.RO and define eCO.Ext relative to a function f for which 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

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

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

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

Effect of eCO.Ext on adversarial behavior. We start by restating helper Theorem [DFMS21, 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 14 (Online extractability (Part of Theorem 4.3 in [DFMS21])). *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 interfaces* 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*

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

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

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

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 Theorem 13: 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 eCO.RO, but only after having applied an additional *'semi-classical'* oracle $O_{\mathcal{S}}^{\mathsf{SC}}$ 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 of F returns 1 is denoted by FIND.

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

Theorem 15 (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*

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

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 16 (Extractable OWTH: Finding independent values [HHM22, Corollary 4]). *Let* eCO, f, GenInp, FIND, *be like in Theorem 15, 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,*

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

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

In this section, we lift our IND-CPA-KEM to IND-CCA-KEM result, Theorem 11, to the quantum-accessible random oracle model.

Theorem 17 (Simulatability of salted $\text{D}\textsc{ecaps}^{\perp}$ 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* $\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}$ *adversary against* $\mathsf{KEM}^{\perp}$, *issuing at most* q_{G} *many queries to* G, q_D *queries to* $\text{D}\textsc{ecaps}^{\not\perp}$, *and with* d *and* w *being the combined query depth/width of* $\mathcal{A}$*'s random oracle queries. Then there exist an* $\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}$ *adversary* $\mathcal{B}$ *against* $\mathsf{KEM}^{\not\perp}$ *and an* $\mathsf{FFP}\text{-}\mathsf{CCA}_u$ *adversary* $\mathcal{C}$ *against* PKE_1 *in the extractable QROM such that*

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

The $\mathsf{FFP}\text{-}\mathsf{CCA}_u$ *game for* PKE_1 *is defined in Fig. 14 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* $\text{D}\textsc{ec}$ *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* $\mathrm{Time}(\mathcal{B}), \mathrm{Time}(\mathcal{C}) = \mathrm{Time}(\mathcal{A}) + O(q_D)$.

Discussion of the bound in Theorem 17. 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' ($\mathsf{FFP}\text{-}\mathsf{CCA}$) advantage against the salted derandomized encryption scheme PKE_1:

```
Game $\mathsf{FFP\text{-}ATK}_{\mathsf{PKE}_1,n_u}$
01 for $j \in [n_u]$
02     $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow\!\!\$\ \mathsf{Gen}()$
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 $(c, \mathsf{salt}) := \mathsf{PKE}_1.\mathsf{Enc}(\mathsf{pk}_j, m)$ for given salt
06 $m' := \mathsf{Dec}(\mathsf{sk}_j, (c, \mathsf{salt}))$
07 return $[[m' \neq m]]$

$\textsc{Dec}(j \in [n_u], (c, \mathsf{salt}))$
08 $m' := \mathsf{PKE}_1.\mathsf{Dec}(\mathsf{sk}_j, (c, \mathsf{salt}))$
09 return $m'$
```

Fig. 14: 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. (4). $\mathsf{O}_{\mathsf{ATK}}$ 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.)

1. We can analyze a softer requirement: according to Theorem 21 which we prove in Section A, there exists an $\mathsf{FFP\text{-}CPA}_u$ attacker $\mathcal{C}'$ such that

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

so it is sufficient to analyze the 'Find-Failing-Plaintext' property of PKE_1 for passive attackers (see Fig. 14). The additional γ-term in the resulting bound occurs only due to our modular proof (combining Theorem 17 with Theorem 21) and can be avoided with a direct proof that immediately reduces to $\mathsf{FFP\text{-}CPA}$.

2. According to Theorem 22 which we also prove in Section A, if PKE is δ-worst-case correct, then we can alternatively bound

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

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 Theorem 17. 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 [DFMS21] 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 Section 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 Theorem 17 in Section A.

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

We now revisit the IND-CPA-PKE to IND-CPA-KEM result, Theorem 13, and lift it to the QROM.

Theorem 18. *Let* PKE *be a public-key encryption scheme, and let* $\mathsf{KEM}^{\not\perp} := \mathsf{SFO}^{\not\perp}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$. *Let* $\mathcal{B}$ *be an adversary against the* $\mathsf{IND}_{n_\mathsf{c}, n_\mathsf{u}}\text{-}\mathsf{CPA}$ *security of* KEM *in the extractable QROM, issuing at most* q *many queries in total to* H *and* G, *with a total depth of* d *and let the bounds* $n_{\mathsf{c}j}$ *of challenges per user* $j \in \{1, \cdots, u\}$ *be known in advance. Furthermore, assume that* $\mathcal{B}$ *does not query* $\mathsf{eCO.Ext}$ *on any of its challenge ciphertexts.*

Then there exists a quantum $\mathsf{IND}_{n_\mathsf{c}, n_\mathsf{u}}\text{-}\mathsf{CPA}$ *adversary* $\mathcal{A}$ *against* PKE *such that*

$$\mathrm{Adv}_{\mathsf{KEM}}^{\mathsf{IND}_{n_\mathsf{c}, n_\mathsf{u}}\text{-}\mathsf{CPA}}(\mathcal{B}) \leq 4\sqrt{d \cdot \mathrm{Adv}_{\mathsf{PKE}}^{\mathsf{IND}_{n_\mathsf{c}, n_\mathsf{u}}\text{-}\mathsf{CPA}}(\mathcal{A})} + \frac{n_\mathsf{c}^2}{|\mathcal{M}| 2^{\mathsf{len}_{\mathsf{salt}}}} + 4\sqrt{d \frac{n_{\mathsf{c}j}\, q}{|\mathcal{M}| |\mathfrak{L}_S|}} \ ,$$

where $n_{\mathsf{c}j}$ *is the number of queries to* Chall *for user index* j.

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 $\mathsf{IND}_{n_\mathsf{c}, n_\mathsf{u}}\text{-}\mathsf{CPA}$ 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 multi-INDCPA game.

Proof. We consider the same sequence of games as in the proof of Theorem 13, except that in this proof, the random oracles in game 2 do not abort. (Since the RO-query-related event CHAL no longer is a well-defined event if the ROs are accessible in superposition.) For convenience, games 0 and 2 are repeated in Fig. 15. Since the reasoning underlying the bounds does not change for quantum attackers up to how we bound CHAL, the bound

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

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 CHAL.

Games $\mathbf{G}_{1'}$ and $\mathbf{G}_{2'}$: presample to prepare quantum counterpart of bounding CHAL. We will still want to construct an $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ 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. 15. 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 $|\Pr[\mathbf{G}_{2'} \Rightarrow 1] - \Pr[\mathbf{G}_{1'} \Rightarrow 1]|$.

```
G_{1'} and G_{2'}
$b \leftarrow_\$ \{0,1\}$
for  $j \in [n_\mathsf{u}]$
    $(pk_j, sk_j) \leftarrow_\$ \mathsf{Gen}()$
$\vec{pk} = (pk_1, ..., pk_j)$
for  $j \in [n_\mathsf{u}]$
    for $i \in [n_{\mathsf{c}j}]$
        $(m_{j,i}, \mathsf{salt}_{j,i}) \leftarrow_\$ \mathcal{M} \times \{0,1\}^{\mathsf{len}_\mathsf{salt}}$
        if $(m_{j,i}, \mathsf{salt}_{j,i}) \in \mathfrak{L}_{M_j}$
            $\mathsf{REPEAT}_{j,i} := \mathsf{true}$
        $\mathfrak{L}_{M_j} := \mathfrak{L}_{M_j} \cup \{m_{j,i} || \mathsf{salt}_{j,i}\}$
        $r_{j,i} := \mathsf{G}(pk_j, m || \mathsf{salt}_{j,i})$                    // $\mathbf{G}_{1'}$
        $c_{j,i} := \mathsf{Enc}\left(pk_j, m_{j,i_j}; r_{j,i_j}\right)$
        $\mathbf{ss}_{0,j,i} = \mathsf{H}\left(pk_j, m_{j,i_j}, c_{j,i} || \mathsf{salt}\right)$      // $\mathbf{G}_{1.5}$
        $(r_{j,i}, \mathbf{ss}_{0,j,i}) \leftarrow_\$ \mathcal{R} \times \mathcal{K}$              // $\mathbf{G}_{2'}$
$b' \leftarrow \mathcal{B}^{\textsc{Chall},\mathsf{G},\mathsf{H}}(\vec{pk})$
return $[\![b = b']\!]$

CHALL(j) // at most $n_{\mathsf{c}j}$ queries
$i_j++$
$c := c_{j,i_j}$
if $\mathsf{REPEAT}_{j,i}$
    return $(c, \mathbf{ss} \leftarrow_\$ \mathcal{K})$
$\mathbf{ss}_0 := \mathbf{ss}_{0,j,i_j}$
$\mathbf{ss}_1 \leftarrow_\$ \mathcal{K}$
return $(c, \mathsf{salt}_{j,i_j}, \mathbf{ss}_b)$
```

Fig. 15: Games for the proof of Theorem 18 with early sampling of all values during game initialization.

Quantum counterpart of bounding CHAL. We still want to construct an $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ 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 Theorem 15, 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 $\mathcal{S} := \mathfrak{L}_{M_j}$, we now puncture on the set $\mathcal{S}'' := \mathfrak{L}''_{M_j}$ 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 $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ adversary $\mathcal{A}$ against PKE: $\mathcal{A}$ pre-computes $\mathcal{S} = \mathfrak{L}_{M_j}$ and $\mathcal{S}'' := \mathfrak{L}''_{M_j}$, 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 occured. We thus obtain

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

Lastly, we bound $\Pr[\mathsf{FIND} : \mathcal{B}^{\mathsf{eCO}^1 \backslash \mathcal{S}''}(inp)]$. In this experiment, the puncturing set $\mathcal{S}$" is independent of the ciphertexts used by the challenge oracle. We can thus apply Corollary 16 to conclude

$$\Pr[\mathsf{FIND} : \mathcal{B}^{\mathsf{eCO}^1 \backslash \mathcal{S}''}(inp)] \leq \sum_{j \in [n_\mathsf{u}]} \frac{n_{\mathsf{c}j} \cdot (q_\mathsf{H} + q_\mathsf{G})}{|\mathcal{M}||\mathfrak{L}_{S_j}|} \leq \frac{n_\mathsf{c} \cdot (q_\mathsf{H} + q_\mathsf{G})}{|\mathcal{M}||\mathfrak{L}_S|}.$$

□

6 Conclusions

Our results show that the SFO transform tightly reduces $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$ KEM security, to the $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ security of the underlying PKE. Our theorems show that this approach indeed mitigate the sort of multi-target attacks identified by NIST against $\mathsf{FrodoKEM}$ and HQC.

6.1 Comparison to existing work

Comparison to hybrid bound. Bellare et al. [BBM00] proved the following bound on multi-challenge IND-CCA security for a generic protocol:

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

Combining this with the bounds in [HHK17] yields the following simplified bounds:

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

$$\mathrm{Adv}_{\mathsf{KEM}^{\perp}}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}} \leq n_u \cdot n_c \cdot \left(2\mathrm{Adv}_{\mathsf{PKE}}^{\mathsf{IND\text{-}CPA}} + q_{\mathsf{RO}} \cdot \delta + \frac{2q_{\mathsf{RO}}}{\mathcal{M}} + q_{\mathsf{RO}}2^{-\gamma}\right)$$

where q_{RO} is the number of random oracle queries (across all oracles).

In comparison applying the hybrid bound to $\mathrm{Adv}_{\mathsf{PKE}}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}$, our analysis yields:

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

$$\mathrm{Adv}_{\mathsf{KEM}^{\perp}}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}} \leq (2n_u \cdot n_c)\mathrm{Adv}_{\mathsf{PKE}}^{\mathsf{IND\text{-}CPA}} + \frac{n_c^2}{|\mathcal{M}|2^{\mathsf{len}_{\mathsf{salt}}}} + \frac{2n_c \cdot q_{\mathsf{RO}}}{|\mathcal{M}||\mathfrak{L}_S|} + q_{\mathsf{RO}} \cdot \delta(n_u) + q_{\mathsf{RO}} \cdot 2^{\gamma}$$

For example, consider $|\mathcal{M}| = 2^{128}$, $\mathsf{len}_{\mathsf{salt}} = 64$, $n_c = 2^{64}, n_u = 2^{32}$ and $q_{\mathsf{RO}} = 2^{64}$. In the hybrid bound, the advantage is dominated by the intermediate terms, as these are multiplied by 2^{96}. These parameters are consistent with, for example FrodoKEM-640 and HQC-128. In our bounds, intermediate terms relating to collision finding are very small, with

$$\frac{n_c^2}{|\mathcal{M}|2^{\mathsf{len}_{\mathsf{salt}}}} = \frac{n_c}{|\mathcal{M}|} = \frac{1}{2^{64}}$$
$$\frac{2n_c \cdot q_{\mathsf{RO}}}{|\mathcal{M}|\mathfrak{L}_S|} \approx \frac{4q_{\mathsf{RO}}}{\mathcal{M}} = \frac{1}{2^{62}}$$

The trivial bound on multi-user correctness is $\delta \leq \delta(n_u) \leq n_u \cdot \delta$, which is of course, much smaller than the term $n_u \cdot n_c \cdot \delta$ obtained by the hybrid bound. There is also strong evidence that for lattice-based schemes, $\delta(n_u) < n_u\delta(1)$ as reasoned in [DHK+21b].

Comparison to FO transform. Duman et al. [DHK+21a] give a bound for IND_{n_c,n_u}-CCA security for FO based KEM's, which is similar to ours, but only covers implicit rejection, and does not include salts. Their bound is

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

For our parameters, the advantage is dominated by the term

$$\frac{n_c^2 + 2n_c \cdot q_{\mathsf{RO}}}{|\mathcal{M}|} = \frac{2^{128} + 2^{129}}{2^{128}}$$

Thus, for small message spaces, (i.e., $|\mathcal{M}| = n_\mathsf{c}^2$, for some feasible n_c), the FO transform results in KEM's that are trivially not $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$ secure, regardless of the $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ security of the underlying PKE. Our SFO transform, on the other hand, tightly reduces security to the $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ security of the underlying PKE.

6.2 Future work

While our results dramatically improve on the hybrid bounds for $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CCA}$ KEM security, we still bound $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ PKE security by a hybrid argument. Bellare et al. [BBM00] show that, in general, one cannot improve on the hybrid bound for $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$-$\mathsf{CPA}$ security. However, for concrete schemes, like $\mathsf{FrodoPKE}$, it may be possible to induce some marginal improvement.

It may also be worth considering a weaker notion of multi-challenge security, where the challenge bit b is sampled independently for each user, and the adversarial goal is to identify an index i for which they can guess the challenge bit. This notion might be more realistic, and better capture the intuition that a multi-user attack breaks security for “one-of-many” user's.

Acknowledgments

We would like to thank Sam Jaques and David Jao for providing helpful comments on the thesis project from which this paper was adapted.

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/round-4-submissions.

ABD+23. 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.

AHU19. Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 269–295, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

BBM00. Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274, Bruges, Belgium, May 14–18, 2000. Springer, Berlin, Heidelberg, Germany.

Ber22. Daniel J. Bernstein. Multi-ciphertext security degradation for lattices. Cryptology ePrint Archive, Report 2022/1580, 2022.

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

BS20. Nina Bindel and John M. Schanck. Decryption failure is more likely after success. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 206–225, Paris, France, April 15–17, 2020. Springer, Cham, Switzerland.

Den03a. Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151, Cirencester, UK, December 16–18, 2003. Springer, Berlin, Heidelberg, Germany.

Den03b. Alexander W. Dent. A designer's guide to KEMs. In Paterson and Kenneth G., editors, *Cryptography and Coding*, pages 133–151, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

DFMS21. Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Online-extractability in the quantum random-oracle model. Cryptology ePrint Archive, Report 2021/280, 2021.

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

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: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 565–598, Beijing, China, April 14–17, 2019. Springer, Cham, Switzerland.

DHK$^+$21a. 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 *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 2722–2737, New York, NY, USA, 2021. Association for Computing Machinery.

DHK$^+$21b. 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: 28th Conference on Computer and Communications Security*, pages 2722–2737, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press.

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, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture*

Notes in Computer Science, pages 3–33, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.

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: 29th Conference on Computer and Communications Security*, pages 979–993, Los Angeles, CA, USA, November 7–11, 2022. ACM Press.

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

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

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

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

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

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

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

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

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`.

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

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

Supplementary material

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

In this section, we prove Theorem 17, our lift of Theorem 11 to the quantum-accessible random oracle model. We first repeat the theorem statement for convenience.

Theorem 19 (Theorem 17: 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* $\mathsf{IND}_{n_\mathsf{c},n_\mathsf{u}}$*-*$\mathsf{CCA}$ *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* $\mathsf{FFP}\text{-}\mathsf{CCA}_u$ *adversary* $\mathcal{C}$ *against* PKE_1 *in the extractable QROM such that*

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

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. 16 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

$$f : ((\mathsf{pk}, m \| \mathsf{salt}), r) \mapsto (\mathsf{Enc}(\mathsf{pk}, m; r), \mathsf{pk}, \mathsf{salt}) \quad , \tag{4}$$

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 $r := \mathsf{G}(\mathsf{pk}, m \| \mathsf{salt})$.

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, after $\mathcal{A}$ finished, we use $\mathsf{eCO.Ext}$ to compute oracle preimages for all

```
$\textsc{Decaps}(j, (c\|\mathsf{salt}) \notin \mathfrak{L}_{C_j})$
$m' := \mathsf{Dec}(\mathsf{sk}_j, c)$
**if** $m' = \bot$
    **return** $K{:=}\bot$
**else**
    $r' := \mathsf{G}(\mathsf{pk}_j, m'\|\mathsf{salt})$
    $c' := \mathsf{Enc}(\mathsf{pk}_j, m'; r')$
    **if** $c \neq c'$
        **return** $\bot$
    **else**
        **return** $\mathsf{H}(\mathsf{pk}, m', c\|\mathsf{salt})$
```

```
$\textsc{Decaps}'(j, (c\|\mathsf{salt}) \notin \mathfrak{L}_{C_j})$
Parse $\mathsf{pk} := \mathsf{pk}_j$
$\hat{m} \leftarrow \mathsf{eCO.Ext}(c, \mathsf{pk}, \mathsf{salt})$
**if** $\hat{m} = \bot$
    **return** $\bot$
**else**
    **return** $\mathsf{H}(\mathsf{pk}, \hat{m}, c\|\mathsf{salt})$
```

```
$\textsc{Decaps}''(j, (c\|\mathsf{salt}) \notin \mathfrak{L}_{C_j})$
Parse $\mathsf{pk} := \mathsf{pk}_j$
$\hat{m} \leftarrow \mathsf{eCO.Ext}(c, \mathsf{pk}, \mathsf{salt})$
$m' := \textsc{Dec}(j, (c\|\mathsf{salt}))$
**if** $\hat{m} \neq \bot$ **and** $\hat{m} \neq m'$
    Store $(j, \hat{m}\|\mathsf{salt})$ in $\mathcal{L}_{\textsc{Fail}}$
**if** $\hat{m} = \bot$
    **return** $\bot$
**else**
    **return** $\mathsf{H}(\mathsf{pk}, \hat{m}, c\|\mathsf{salt})$
```

```
$\textsc{Dec}(j, (c\|\mathsf{salt}))$
$m' := \mathsf{Dec}(\mathsf{sk}_j, c)$
$r' := \mathsf{G}(\mathsf{pk}_j, m', \mathsf{salt})$
**if** $m' = \bot$
    **return** $\bot$
**else**
    **if** $\mathsf{Enc}(\mathsf{pk}_j, m'; r') \neq c$
        **return** $\bot$
    **else**
        **return** $m'$
```

Fig. 16: 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 [DFMS21] (see Section 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. (4).

ciphertexts on which $\textsc{Decaps}$ was queried, i.e., we take each decapsulation query $(j_i, (c_i, \mathsf{salt}_i))$ and compute $\hat{m}_i := \mathsf{eCO.Ext}(c_i, \mathsf{pk}_{j_1}, \mathsf{salt}_i)$.

By property 1 in Lemma 14, 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

$$\mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}^{\bot}_{m}}(\mathcal{A}) = \mathrm{Adv}^{\mathbf{G}}_{0} = \mathrm{Adv}^{\mathbf{G}}_{1} \ . \tag{5}$$

$\mathbf{G}_2$ is like $\mathbf{G}_1$, except that $\hat{m}_i := \mathsf{eCO.ExteCO.Ext}(c_i, \mathsf{pk}_{j_1}, \mathsf{salt}_i)$ 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 Lemma 14): according to that item, each query $8\sqrt{2\Gamma(f)/2^n}$-almost-commutes, where $\{0,1\}^n$ is the random oracle co-domain. For our choice of random oracle, $2^n = |\mathcal{R}|$. For our choice of f,

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

where 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,

$$\left|\mathrm{Adv}_1^{\mathbf{G}} - \mathrm{Adv}_2^{\mathbf{G}}\right| \leq 8\sqrt{2}q_D(q_{\mathsf{G}} + q_D) \cdot 2^{-\gamma/2} \ . \tag{6}$$

$\mathbf{G}_3$ 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

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

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 $\hat{m} := \mathsf{eCO.Ext}(c, \mathsf{pk}_j, \mathsf{salt})$. 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 $\hat{m} := \mathsf{eCO.Ext}(c, \mathsf{pk}_j, \mathsf{salt})$ differs from $m' := \mathsf{Dec}(\mathsf{sk}_j, c)$. 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 $\hat{m} := \mathsf{eCO.Ext}(c, \mathsf{pk}_j, \mathsf{salt})$ in line 12 yielded $\perp$. At the same time, c passed the re-encryption check inside Decaps (line 07), so $\mathsf{Enc}(\mathsf{pk}_j, m', r') = c$ for $m' := \mathsf{Dec}(\mathsf{sk}_j, c)$ and $r' := \mathsf{G}(\mathsf{pk}_j, m', \mathsf{salt})$. 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 Lemma 20 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 $\textsc{Decaps}''$, using its own $\mathsf{FFP\text{-}CCA}$ oracle $\textsc{Dec}$ 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 $\textsc{Decaps}''$ adds a plaintext $\hat{m}$ to $\mathcal{L}_{\text{FAIL}}$, 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 $\mathcal{L}_{\text{FAIL}}$ 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 $\mathsf{FFP\text{-}CCA}$ game.

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

$\mathbf{G}_4$ prepares for fading out the now-redundant internal calls to $\textsc{Decaps}$– note that these calls were not used within the responses of the oracle $\textsc{Decaps}'$ 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 $\textsc{Decaps}$ (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 $\textsc{Decaps}$ invocations are postponed until after $\mathcal{A}$ finishes. With similar reasoning as when stepping from $\mathbf{G}_1$ to $\mathbf{G}_2$, we obtain

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

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

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

We now define $\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}$ 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 $\textsc{Decaps}'$ and forwards

```
$\underline{\textsc{Dec}(j, (c \| \mathsf{salt}))}$
01 $m' := \mathsf{Dec}(\mathsf{sk}_j, c)$
02 $r' := \mathsf{G}(\mathsf{pk}_j, m', \mathsf{salt})$
03 if $m' = \bot$
04   return $\bot$
05 else
06   if $\mathsf{Enc}(\mathsf{pk}_j, m'; r') \neq c$
07     return $\bot$
08   else
09     return $m'$

$\underline{\textsc{Dec}'(j, (c \| \mathsf{salt}))}$
10 $\hat{m} \leftarrow \mathsf{eCO.Ext}(c, \mathsf{pk}_j, \mathsf{salt})$
11 return $m$
```

Fig. 17: Simulation $\textsc{Dec}'$ of decryption oracle $\textsc{Dec}$ for $\mathsf{PKE}_1 := \mathsf{ST}[\mathsf{PKE}, \mathsf{G}]$.

$\mathcal{A}$'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.

$$\mathrm{Adv}_5^{\mathbf{G}} = \mathrm{Adv}_{\mathsf{KEM}_m^{\bot}}^{\mathsf{IND\text{-}CPA}}(\mathcal{B}). \tag{7}$$

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 (Theorem 21) 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 $\textsc{Dec}$ is simulated via simulation $\textsc{Dec}'$ (see Fig. 17). ($\textsc{Dec}'$ uses the same plaintext extraction technique as simulation $\textsc{Decaps}'$ and differs from $\textsc{Dec}$ only in a case very similar to GUESS.)

Lemma 20. *Let PKE be γ-spread, and let GUESS be as defined like 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 $\mathsf{SFO}_{m,c}^{\bot}[\mathsf{PKE}, \mathsf{G}, \mathsf{H}, \mathsf{len}_{\mathsf{salt}}]$, 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. 16), 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 $\hat{m}_i = \bot$ and $m_i \neq \bot$. 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. 17.

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 $\hat{m} = \bot$ but $m \neq \bot$ for that query. Intuitively, this captures that $c = \mathsf{Enc}(\mathsf{pk}_j, m'; r')$ for $m' := \mathsf{Dec}(\mathsf{sk}_j, c)$ and $r' := \mathsf{G}(\mathsf{pk}_j, m', \mathsf{salt})$, but that the query $(\mathsf{pk}_j, m', \mathsf{salt}))$ had not yet been written into the oracle database of G before the re-encryption step. We claim

$$\Pr[\hat{m}_i = \bot \wedge m_i \neq \bot] \leq 2 \cdot 2^{-\gamma}. \tag{8}$$

Once we have proven Eq. (8), 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 $\mathsf{eCO.Ext}$:

$$\begin{aligned}\Pr[\hat{m} = \bot \wedge m' \neq \bot] &\leq \Pr[\hat{m} = \bot \wedge \mathsf{Enc}(m'; \mathsf{eCO.RO}(\mathsf{pk}_j, m', \mathsf{salt})) = c] \\ &= \left\| \mathit{\Pi}_Y^{c,x} O_{XYF} \mathit{\Sigma}_F^{c,\varnothing} \left|\mathsf{pk}_j, m', \mathsf{salt}\right\rangle_X |0\rangle_Y |\psi_i\rangle_{FE} \right\|^2 \quad , \end{aligned} \tag{9}$$

where $|\psi_i\rangle$ denotes the adversary-oracle state right before $\mathcal{A}$ submits the i-th query c, and the projectors $\mathit{\Pi}_Y^{c,x}$ and $\mathit{\Sigma}^{c,\varnothing}$ (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 $x := m'$, and where the projectors are defined relative to $f_{\mathsf{pk}} : (m, r) \mapsto \mathsf{Enc}(\mathsf{pk}, m; r)$ – can be bounded by

$$\left\| \mathit{\Pi}_Y^{c,x} O_{XYF} \mathit{\Sigma}_F^{c,\varnothing} |x\rangle_X |0\rangle_Y |\psi_i\rangle_{FE} \right\| \leq \sqrt{2} \cdot 2^{-\gamma/2} \quad . \tag{10}$$

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 $\mathit{\Gamma}(f_{\mathsf{pk}}) \leq 2^{-\gamma}|\mathcal{R}|$. By adapting the X-register such that it additionally accommodates pk and salt, and by noticing that our projectors are also diagonal and that $\mathit{\Gamma}(f) \leq 2^{-\gamma}|\mathcal{R}|$, we thus obtain the same bound as in Eq. (10) in our adapted setting, and thus Eq. (8) by combining Eq. (10) with Eq. (9).

□

Bound FFP-CCA via FFP-CPA. We now show that we can simplify the FFP-CCA term as indicated below Theorem 17: firstly, we simplify it to its passive counterpart, FFP-CPA.

Theorem 21 (ST [PKE, G] $\mathsf{FFP\text{-}CPA}_u \Rightarrow \mathsf{ST}$ [PKE, 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*

$$\mathrm{Adv}^{\mathsf{FFP\text{-}CCA}_u}_{\mathsf{ST[PKE,G]}}(\mathcal{C}) \leq (q_D + 1) \cdot \mathrm{Adv}^{\mathsf{FFP\text{-}CPA}_u}_{\mathsf{ST[PKE,G]}}(\mathcal{C}') + 12 q_D(q_{\mathsf{G}} + 4q_D)2^{-\gamma/2} \quad . \tag{11}$$

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 DEC provided in the FFP-CCA game can be simulated via oracle DEC$'$ (see Fig. 17). Since DEC$'$ works without the secret key, we can then construct $\mathcal{C}'$ that uses $\mathcal{C}$, simulating $\mathcal{C}$'s decryption oracle via DEC$'$. In a way, this is analogous to Theorem 17 in which we replaced DECAPS by simulation DECAPS$'$. We note the tightness loss: as a passive adversary $\mathcal{C}'$ cannot use DEC to determine at which 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 Theorem 17 – we first 'commute in' the extraction queries that would be needed for DEC$'$, then switch from DEC to DEC$'$, and then omit the redundant internal calls to DEC. Like in the proof of Theorem 17, we have

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

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 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,

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

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 Theorem 17, we now bound it in terms of (statistical) worst-case correctness.

Theorem 22 (PKE δ **(n_{u})-worst-case-correct** $\Rightarrow$ $\mathsf{ST}[\mathsf{PKE},\mathsf{G}]$ $\mathsf{FFP\text{-}CCA}_u$**).** *Let* PKE *be a (randomized)* PKE *scheme that is δ-worst-case-correct, and let $\mathcal{C}$ be an* eQROM *adversary against* $\mathsf{ST}[\mathsf{PKE},\mathsf{G}]$ *in the* $\mathsf{FFP\text{-}CCA}_u$ *game as defined in Fig. 14, issuing at most q_D many decryption queries and at most q many queries to its interface* $\mathsf{eCO.RO}$*. 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 Theorem 22, we thus first recall this theorem.

Theorem 23 (PKE **δ-worst-case-correct** $\Rightarrow$ $\mathsf{T}[\mathsf{PKE},\mathsf{G}]$ $\mathsf{FFP\text{-}CCA}$**).** *Let* PKE *be a (randomized)* PKE *scheme that is δ-worst-case-correct, and let $\mathcal{C}$ be an* $\mathsf{FFP\text{-}CCA}$ *adversary against* $\mathsf{T}[\mathsf{PKE},\mathsf{G}]$ *in the* $\mathsf{eQROM}_{\mathsf{Enc}}$*, issuing at most q_D decryption queries and q many queries to its extQROM oracle interface* $\mathsf{eCO.RO}$*. 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 Theorem 22.. First, we will show that the bound in Theorem 23 also applies for our transformation ST when viewing the salts as part of the message, like in Fig. 14. 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. 18.
2. Apply to $\mathsf{Salt}[\mathsf{PKE}]$ 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. 19.

```
$\mathsf{Enc}_{\mathsf{st}}(pk, m\|\mathsf{salt})$
    $c \leftarrow \mathsf{PKE.Enc}(pk, m)$
    return $c\|\mathsf{salt}$

$\mathsf{Dec}_{\mathsf{st}}(\mathsf{sk}, c\|\mathsf{salt})$
    $m' := \mathsf{PKE.Dec}(\mathsf{sk}, c)$
    return $m'\|\mathsf{salt}$
```

Fig. 18: Salted PKE scheme $\mathsf{Salt}[\mathsf{PKE}] = (\mathsf{Gen},\mathsf{Enc}_{\mathsf{st}},\mathsf{Dec}_{\mathsf{st}})$.

So 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

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

```
$\mathsf{T_{pk}[PKE,G].Enc(pk},m)$
01 $r := \mathsf{G}(\mathsf{pk},m)$
02 $c \leftarrow \mathsf{PKE.Enc(pk},m;r)$
03 return $c$

$\mathsf{T_{pk}[PKE,G].Dec(sk},c)$
04 $m' = \mathsf{PKE.Dec(sk},c)$
05 $r' := \mathsf{G}(\mathsf{pk},m)$
06 if $m' = \perp$ or $\mathsf{PKE.Enc(pk},m';r') \neq c$
07   return $\perp$
08 else return $m'$
```

Fig. 19: Multi-user variant $\mathsf{T_{pk}[PKE,G]}$, deviations from T highlighted in violet.

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

$$\mathrm{Adv}^{\mathsf{FFP\text{-}CCA}}_{\mathsf{T_{pk}[Salt[PKE],G]}}(\mathcal{C}) \leq 10(q+q_D+1)^2 \cdot \delta \ . \tag{12}$$

We address two (minor) obstacles: first, we use transformation $\mathsf{T_{pk}}$ 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 Theorem 24 below). Second, the bound's right-hand side would use the correctness term of $\mathsf{Salt[PKE]}$, 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 $(m\|\mathsf{salt}) \in \mathcal{M} \times \{0,1\}^{\mathsf{len_{salt}}}$, and define the conditional terms

$$\delta(\mathsf{kp},m) := \Pr[\mathsf{Dec(sk},c) \neq m : c \leftarrow \mathsf{Enc(pk},m)]$$

and

$$\delta^{\mathsf{st}}(\mathsf{kp},m\|\mathsf{salt}) := \Pr[\mathsf{Dec_{st}(sk},c\|\mathsf{salt}) \neq m\|\mathsf{salt} : c\|\mathsf{salt} \leftarrow \mathsf{Enc_{st}(pk},m\|\mathsf{salt})] \ .$$

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. (12).

We proceed to capturing the multi-user setting. In [HM24], Theorem 23 was proven by fixing the key pair $(\mathsf{pk},\mathsf{sk})$, showing that conditioned on that key pair,

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

where

$$\delta(\mathsf{pk},\mathsf{sk}) := \max_m \Pr_{r \leftarrow \mathcal{R}}[\mathsf{Dec_{sk}}(\mathsf{Enc_{pk}}(m;r)) \neq m] \ ,$$

and then taking the expectation over the key pair to obtain

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

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

$$\Pr[\mathsf{FFP\text{-}CCA}^{\mathcal{C}}_{\mathsf{ST[PKE,G]}} \Rightarrow 1 \mid (\mathsf{pk}, \mathsf{sk})] \leq 10(q + q_D + 1)^2 \cdot \delta(\mathsf{pk}, \mathsf{sk}) \ . \tag{13}$$

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

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

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 $\mathsf{FFP\text{-}CCA}_u$ 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. (13).)

□

For the sake of completeness, we close the proof of Theorem 22 by taking T-derandomized schemes and showing that in/exclusion of pk during randomness derivation has no influence on their $\mathsf{FFP\text{-}CCA}$ property in the single-user setting.

Theorem 24 ($\mathsf{T_{pk}[PKE, G]}$ FFP-CCA $\Leftrightarrow$ $\mathsf{T[PKE, G]}$ FFP-CCA). *Let* PKE *be a (randomized)* PKE *scheme. Let* $\mathcal{C}$ *be an* $\mathsf{FFP\text{-}CCA}$ *adversary against* $\mathsf{T[PKE, G]}$ *in the* $\mathsf{eQROM_{Enc}}$*. 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* $\mathsf{FFP\text{-}CCA}$ *adversary* $\mathcal{C}'$ *against* $\mathsf{T_{pk}[PKE, G']}$ *in the* eQROM_f *such that*

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

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

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

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 $\mathsf{FFP\text{-}CCA}$ adversary $\mathcal{C}$ against $\mathsf{T[PKE, G]}$ in the $\mathsf{eQROM_{Enc}}$. We construct $\mathsf{FFP\text{-}CCA}$ adversary $\mathcal{C}'$ against $\mathsf{T_{pk}[PKE, G']}$ 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 $|\psi\rangle_{\mathcal{PK}\times\mathcal{M}} := |\mathsf{pk}^*\rangle_{\mathcal{PK}} \otimes |\varphi\rangle_{\mathcal{M}}$ 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 $r := \mathsf{G}'(\mathsf{pk}^*, m)$ is identical to the re-encryption check with $r := \mathsf{G}(m)$. 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 $\mathsf{FFP\text{-}CCA}$ 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 $\mathsf{FFP\text{-}CCA}$ adversary $\mathcal{D}$ against $\mathsf{T}_{\mathsf{pk}}[\mathsf{PKE}, \mathsf{G}']$ in the eQROM_f. We construct $\mathsf{FFP\text{-}CCA}$ adversary $\mathsf{FFP\text{-}CCA}$ 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 25 (**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 $v := \mathsf{RO}(x)$ is exactly as secure as a counterpart C_{var} of C that replaces $v := \mathsf{RO}(x)$ by $v := \mathsf{RO}(\mathsf{var}, x)$, 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 $f' := (f_{\mathsf{var}}(x, y), \mathsf{var})$ 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 $\mathsf{RO} : X \rightarrow T$, simulate $\mathsf{RO}' : X \times \mathit{Var} \rightarrow T$ via $\mathsf{RO}'(x, \mathsf{var}') := \mathsf{RO}(x)$ 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'.