The security of Kyber's FO-transform

Manuel Barbosa[1] and Andreas Hülsing[2]

[1] University of Porto (FCUP) and INESC TEC, Portugal
[2] Eindhoven University of Technology
mbb@fc.up.pt, andreas@huelsing.net

Abstract. In this short note we give another direct proof for the variant of the FO transform used by Kyber in the QROM. At PKC'23 Maram & Xagawa gave the first direct proof which does not require the indirection via FO with explicit rejection, thereby avoiding either a non-tight bound, or the necessity to analyze the failure probability in a new setting. However, on the downside their proof produces a bound that incurs an additive collision bound term. We explore a different approach for a direct proof, which results in a simpler argument closer to prior proofs, but a slightly worse bound.

1 Introduction

Kyber is the winning KEM of the NIST post-quantum standardization project and will be standardized as the post-quantum KEM of choice. Like most post-quantum KEMs, the design of Kyber first defines a CPA-secure PKE, which is then used to instantiate the Fujisaki-Okamoto transform [FO99] and obtain a CCA-secure KEM. However, the transform used by Kyber is not the actual FO transform as analyzed, for example, in [HHK17, SXY18, JZC$^+$18, HKSU20, JZM19a, JZM19b, BHH$^+$19].

Figure 1 shows the variant of the FO construction described in [HHK17] that is closest to the Kyber construction. We call this $\mathsf{FO}^{\not\perp}$ and obtain it by combining the T and the $\mathsf{U}^{\not\perp}$ transforms. Figure 2 shows the Kyber construction, which we call $\mathsf{FO}^{\mathsf{Kyber}}$. The Kyber transform $\mathsf{FO}^{\mathsf{Kyber}}$ deviates from the original transform $\mathsf{FO}^{\not\perp}$ in two steps of the encapsulation procedure and the corresponding parts of the decapsulation. The first step of the FO-transform (called T-transform in [HHK17]) is to derandomize the PKE encryption scheme. For this, the encryption randomness is replaced by the hash of the message $r \leftarrow G(m)$. G is modeled as a random oracle in security proofs for the FO transformation. Kyber extends the output of G to include, not only encryption randomness r, but also a *pre-key* $\tilde{K}$ (see $(\tilde{K}, r) \leftarrow H_1(m, H_3(\mathrm{pk}))$ in line 2 of Encaps). We note that, in the security proof, one can ignore the $H_3(\mathrm{pk})$ input and H_1 can be seen as $H_1(x) := H_1'(x) \| G(x)$, where H_1' and G are random oracles. This is because there is only one public key in the CCA game (hashing the public key is relevant when considering multi-instance attacks) and any random oracle that maps to a product space can be split into two independent random oracles over the same input type. Indeed, once this conceptual change is introduced, the standard T transform is recovered and one can focus on the other modifications as applying to the $\mathsf{U}^{\not\perp}$ transform, which do have an impact on the security proof.

Consider the key derivation step (line 4 in Encaps of Figure 2). Kyber uses $\tilde{K} = (H_1'(m))_1$ in place of m and $H_2(c)$, the hash of the ciphertext, instead of the ciphertext c. These two modifications broke existing proofs for the $\mathsf{U}^{\not\perp}$ transform and created complications in attempts to give a direct proof that would follow along the lines of previous proofs for FO with implicit rejection. The main issue is that the proofs for implicit rejection commonly use an approach that dates back to at least the seminal work introducing the QROM [BDF$^+$11], which answers decryption queries

This work was funded by an NWO VIDI grant (Project No. VI.Vidi.193.066). Date: May 25, 2023

```
Keygen():
1  $(\mathrm{pk}, \mathrm{sk}_{\mathsf{P}}) \leftarrow \mathrm{Keygen}_{\mathsf{P}}()$
2  $\mathrm{prfk} \xleftarrow{\$} \mathcal{K}_{\mathsf{F}}$
3  $\mathrm{sk} \leftarrow (\mathrm{sk}_{\mathsf{P}}, \mathrm{prfk})$
4  return $(\mathrm{pk}, \mathrm{sk})$
```

```
Encaps(pk):
1  $m \xleftarrow{\$} \mathcal{M}$
2  $r \leftarrow G(m)$
3  $c \leftarrow \mathrm{Encr}(\mathrm{pk}, m; r)$
4  $K \leftarrow H(m, c)$
5  return $(K, c)$
```

```
Decaps(sk, c):
1  parse $\mathrm{sk} = (\mathrm{sk}_{\mathsf{P}}, \mathrm{prfk})$
2  $m' \leftarrow \mathrm{Decr}(\mathrm{sk}_{\mathsf{P}}, c)$
3  if $m' = \perp$:
4      return $\mathsf{F}(\mathrm{prfk}, c)$
5  else if $\mathrm{Encr}(\mathrm{pk}, m'; G(m')) \neq c$:
6      return $\mathsf{F}(\mathrm{prfk}, c)$
7  else: return $H(m', c)$
```

Fig. 1. Transform $\mathsf{FO}^{\not\perp}(\mathsf{P}, \mathsf{F}, G, H) := (\mathrm{Keygen}, \mathrm{Encaps}, \mathrm{Decaps})$.

by manipulating the random oracle used for key derivation (c.f., line 4 of Encaps in Figure 1) in a smart way. Namely, the oracle is transformed into $H(m, c) := H'(\mathrm{Encr}(\mathrm{pk}, m; G(m)))$ for a random oracle H' that the adversary is not given direct access to, using the fact that the ciphertext can be deterministically computed from message m. This way, they can simulate the corresponding decapsulation oracle $\mathrm{Decaps}'(c) := H'(c)$ without having knowledge of the secret key.

This strategy is not directly applicable to $\mathsf{FO}^{\mathsf{Kyber}}$, first of all, because the key derivation step only gets the pre-key $\tilde{K} = (H_1(m))_1$, rather than m itself. This issue was observed and is solved by [GMP22] who notice that H_1 is length-preserving, i.e., it is n-to-n. This means that in the proof we can start by replacing H_1, modelled as a random oracle, by a random permutation H_1'.[3]. Once this transformation is introduced, one can first invert the function $H_1' := (H_1)_1$ and set $H(m, c) := H'(\mathrm{Encr}(\mathrm{pk}, {H'}_1^{-1}(\tilde{K}); G(m)))$. However, the authors of [GMP22] did not present a complete proof for $\mathsf{FO}^{\mathsf{Kyber}}$, referring to the hashed ciphertext in the key derivation step as the remaining obstacle. While the authors do not explicitly state what the issue is, their proof strategy relies on a technique that uses a plaintext checking oracle — an oracle PCO that given a plaintext-ciphertext pair (m, c) returns true if m is the decryption of c — which requires knowledge of (m, c). Indeed, as H_2 is compressing, one cannot recreate (m, c) given $(\tilde{K}, H_2(c))$. In this note we show that there is no problem because the alternative proof strategy in [BHH+19] does not require the explicit construction of a PCO. As already conjectured in [GMP22] this does add an additive collision term to the bound, but otherwise works the same.

An alternative approach to prove the same result was given in [MX23]. There, the authors reduce the IND-CCA security of $\mathsf{FO}^{\perp}$ to that of $\mathsf{FO}^{\mathsf{Kyber}}$ in an elegant way. Their approach avoids struggling with implementing the decapsulation oracle, by using the oracle of $\mathsf{FO}^{\perp}$ in their simulation. Nevertheless, their proof also has to consider the possibility of H_2 collisions which allows to detect the simulation. Hence, they also get a collision term. However, they can avoid the second collision term that our bound contains, which is caused by a proof step that replaces H_1' by a random permutation.

Implications for Kyber. Incurring a collision bound term in the security bound for Kyber seems unavoidable when using the modified FO transform without resorting to a different failure notion. This means, that the latest parameters of Kyber for levels III and V which instantiate H_2 with SHA3-256 cannot be justified by the proven bound when using the modified FO transform. The recent proposal by the Kyber team to switch to the standard FO transform therefore seems a good choice from a provable security point of view.

[3] This kind of trick was first suggested in the QROM by Unruh [Unr15]

```
Keygen():
1  $(\mathrm{pk}, \mathrm{sk}_\mathsf{P}) \leftarrow \mathrm{Keygen}_\mathsf{P}()$
2  $\mathrm{prfk} \xleftarrow{\$} \mathcal{K}_\mathsf{F}$
3  $\mathrm{sk} \leftarrow (\mathrm{sk}_\mathsf{P}, \mathrm{pk}, H_3(\mathrm{pk}), \mathrm{prfk})$
4  return $(\mathrm{pk}, \mathrm{sk})$

Encaps(pk):
1  $m \xleftarrow{\$} \mathcal{M}$
2  $(\tilde{K}, r) \leftarrow H_1(m, H_3(\mathrm{pk}))$
3  $c \leftarrow \mathrm{Encr}(\mathrm{pk}, m; r)$
4  $K \leftarrow H(\tilde{K}, H_2(c))$
5  return $(K, c)$

Decaps(sk, c):
1  parse $\mathrm{sk} = (\mathrm{sk}_\mathsf{P}, \mathrm{pk}, h_\mathrm{pk}, \mathrm{prfk})$
2  $m' \leftarrow \mathrm{Decr}(\mathrm{sk}_\mathsf{P}, c)$
3  if $m' = \perp$:
4      return $\mathsf{F}(\mathrm{prfk}, H_2(c))$
5  else if $\mathrm{Encr}(\mathrm{pk}, m'; (H_1(m', h_\mathrm{pk}))_2) \neq c$:
6      return $\mathsf{F}(\mathrm{prfk}, H_2(c))$
7  else: return $H((H_1(m'))_1, H_2(c))$
```

Fig. 2. Transform $\mathsf{FO}^{\mathsf{Kyber}}(\mathsf{P}, \mathsf{F}, H, H_1, H_2) := (\mathrm{Keygen}, \mathrm{Encaps}, \mathrm{Decaps})$. Modifications in blue.

1.1 OW-CPA dPKE $\overset{\text{QROM}}{\Rightarrow}$ IND-CCA $\mathsf{U}^{\mathsf{Kyber}}(\mathsf{P}, \mathsf{F}, H, H_1', H_2)$

We now give a security analysis of the $\mathsf{U}^{\not\perp}$ part of $\mathsf{FO}^{\mathsf{Kyber}}$ (see Figure 4), which we obtain by considering the split of H_1 discussed above, i.e., $H_1(x) := H_1'(x)||G(x)$. The full proof of the Kyber CCA KEM then follows from instantiating $\mathsf{U}^{\mathsf{Kyber}}$ with the output of the standard T transform, which in turn is instantiated with the Kyber IND-CPA-secure PKE ($\mathsf{P}_{\mathsf{Kyber}}$). Recall that the standard T transform yields a deterministic encryption scheme $\mathsf{P} := \mathsf{T}(\mathsf{P}_{\mathsf{Kyber}}, G)$, which is shown to be a OW-CPA secure dPKE in [BHH+19]).

The proof of $\mathsf{U}^{\mathsf{Kyber}}$ follows very closely the proof in [BHH+19], with minor presentation changes for readability. The proof makes use of a correctness notion called Find Failing Ciphertexts (FFC). Moreover, we require the underlying dPKE to be ϵ-injective as also defined in [BHH+19]. We recall the two definitions here.

Definition 1 (Finding Failing Ciphertext). *The find-failing-ciphertexts experiment* (FFC) *is shown in Figure 3. The* FFC*-advantage of an adversary $\mathcal{A}$ is defined by*

$$\mathrm{Adv}_\mathsf{P}^{\mathsf{FFC}}(\mathcal{A}) := \Pr[\mathrm{Expt}_\mathsf{P}^{\mathsf{FFC}}(\mathcal{A}) \to 1].$$

```
Expt_P^FFC(A):
1  $H \xleftarrow{\$} \mathcal{H}$
2  $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{Keygen}()$
3  $L \leftarrow \mathcal{A}^H(\mathrm{pk})$
4  return $[\exists \mathrm{m} \in \mathcal{M}, c \in L: \ \mathrm{Encr}(\mathrm{pk}, \mathrm{m}) = c \ \wedge \ \mathrm{Decr}(\mathrm{sk}, c) \neq \mathrm{m}]$
```

Fig. 3. FFC experiment on a dPKE P. The instantiation of H generalizes to any number of random oracles, including zero.

Definition 2 (Injectivity of PKEs). *A dPKE* $\mathsf{P} = (\mathrm{Keygen}, \mathrm{Encr}, \mathrm{Decr})$ *is ϵ-injective if*

$$\Pr\left[\mathrm{Encr}(\mathrm{pk}, m) \textit{ is not injective} : (\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{Keygen}(), H \xleftarrow{\$} \mathcal{H}\right] \leq \epsilon.$$

We say P *is injective if $\epsilon = 0$. We say that an rPKE is injective if for all public keys* pk*, all $m \neq m'$ and all coins r, r', we have* $\mathrm{Encr}(\mathrm{pk}, m, r) \neq \mathrm{Encr}(\mathrm{pk}, m', r')$.

The main technical novelty in [BHH+19] was the following one-way to hiding lemma which is used in the proof:

Lemma 1 (Double-sided O2H [BHH⁺19, Lemma 5]). *Let $G, H : X \to Y$ be random functions, let z be a random value, and let $S \subset X$ be a random set such that $\forall x \notin S, G(x) = H(x)$. (G, H, S, z) may have arbitrary joint distribution. Let $\mathcal{A}^H$ be a quantum oracle algorithm. Let $f : X \to W \subseteq \{0,1\}^n$ be any function, and let $f(S)$ denote the image of S under f. Let Ev be an arbitrary classical event.*

We will define another quantum oracle algorithm $\mathcal{B}^{G,H}(z)$. This $\mathcal{B}$ runs in about the same amount of time as $\mathcal{A}$, but when $\mathcal{A}$ queries H, $\mathcal{B}$ queries both G and H, and also runs f twice. Let

$$P_{\text{left}} := \Pr[\mathsf{Ev} : \mathcal{A}^H(z)],\ P_{\text{right}} := \Pr[\mathsf{Ev} : \mathcal{A}^G(z)],\ P_{\text{extract}} := \Pr[\mathcal{B}^{G,H}(z) \in f(S)].$$

If $f(S) = \{w^*\}$ is a single element, then $\mathcal{B}$ will only return $\perp$ or w^*, and furthermore

$$|P_{\text{left}} - P_{\text{right}}| \leq 2\sqrt{P_{\text{extract}}} \qquad \text{and} \qquad \left|\sqrt{P_{\text{left}}} - \sqrt{P_{\text{right}}}\right| \leq 2\sqrt{P_{\text{extract}}}.$$

```
Keygen():
1  (pk, sk_P) ← Keygen_P()
2  prfk ←$ K_F
3  sk ← (sk_P, prfk)
4  return (pk, sk)

Encaps(pk):
1  m ←$ M
2  K̃ ← H'_1(m)
3  c ← Encr(pk, m)
4  K ← H(K̃, H_2(c))
5  return (K, c)

Decaps(sk, c):
1  parse sk = (sk_P, prfk)
2  m' ← Decr(sk_P, c)
3  if m' = ⊥:
4      return F(prfk, H_2(c))
5  else if Encr(pk, m') ≠ c:
6      return F(prfk, H_2(c))
7  else
8      return H(H'_1(m'), H_2(c))
```

Fig. 4. Transform $\mathsf{U}^{\mathsf{Kyber}}(\mathsf{P}, \mathsf{F}, H, H_1', H_2) := (\text{Keygen}, \text{Encaps}, \text{Decaps})$.

The following theorem which states that breaking the $\mathsf{IND\text{-}CCA}$ security of $\mathsf{U}^{\mathsf{Kyber}}(\mathsf{P}, \mathsf{F}, H, H_1', H_2)$ requires either breaking the $\mathsf{OW\text{-}CPA}$ security of P, causing a decapsulation failure, breaking the PRF property of F used in implicit rejection, noticing when H_1' is replaced by a random permutation, or finding a collision in H_2. For this, P has to be an ϵ-injective dPKE as in Definition 2.

Theorem 1. *Let $H : \mathcal{M} \times \mathcal{C}_H \to \mathcal{K}$, $H_1' : \mathcal{M} \to \mathcal{M}$, and $H_2 : \mathcal{C} \to \mathcal{C}_H$ be quantum-accessible random oracles. Let $\mathsf{F} : \mathcal{K}_\mathsf{F} \times \mathcal{C}_H \to \mathcal{K}$ be a PRF. Let P be an ϵ-injective* dPKE *which is independent of H, H_1', H_2. Let $\mathcal{A}$ be an $\mathsf{IND\text{-}CCA}$ KEM adversary against $\mathsf{U}^{\mathsf{Kyber}}(\mathsf{P}, \mathsf{F}, H, H_1', H_2)$, and suppose that $\mathcal{A}$ makes at most q_{dec} decryption queries, and q_1 queries to H_1'. Then we can construct four adversaries running in about the same time and resources as $\mathcal{A}$:*

- *a* $\mathsf{OW\text{-}CPA}$*-adversary $\mathcal{B}$ against* P
- *a* PRF*-adversary $\mathcal{B}_1$ against* F
- FFC*-adversaries $\mathcal{B}_5$ and $\mathcal{B}_9$ against* P*, returning a list of at most q_{dec} ciphertexts*

such that

$$\begin{aligned}\text{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{U}^{\mathsf{Kyber}}(\mathsf{P})}(\mathcal{A}) \ \leq 2 \cdot \Bigg(&\sqrt{\text{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{P}}(\mathcal{B})} + \text{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathcal{B}_1) + \text{Adv}^{\mathsf{FFC}}_{\mathsf{P}}(\mathcal{B}_5) + \text{Adv}^{\mathsf{FFC}}_{\mathsf{P}}(\mathcal{B}_9) + 2\epsilon \\ &+ \frac{C(q_1 + q_{\text{dec}} + 1)^3}{|\mathcal{M}|} + \frac{C(q_2 + q_{\text{dec}} + 1)^3}{|\mathcal{C}_H|}\Bigg),\end{aligned}$$

where C is the universal constant in the collision finding bound of [Zha15]

This bound assumes access to further random functions and random permutations. Both can be efficiently simulated using quantum secure PRFs and PRPs, adding a respective additive distinguishing term. The main difference to the original bound in [BHH+19] comprises the collision terms caused by collisions in H_2 and the switch from random function to random permutation for H_1' during the proof. Additional multiplicative factors in the reductions to FFC and ϵ-injectivity are due to a slightly modified game sequence that explicitly introduces (and sometimes removes) explicit aborts when bad events are detected. In the original proof this was avoided in order to optimize the bound, but here this allows a modular analysis of the various bad events and, in particular, the self-contained bounding of collision events.

Before we present the proof, we should also comment on the impact of the collision terms on the choice of security parameters for Kyber. Each term depends on the number of queries made by the adversary to the corresponding hash function, q_1 and q_2. These parameters can only be bound by the security parameter, as the H_1' and H_2 oracles model local computation. These terms also seem to be unavoidable using current proof techniques, and they reflect a potential entropy loss due to the use of additional hashing steps that may impact the security level. For this reason, it would be advisable to remove these additional hash computations from the U^{Kyber} transformation and revert to a standard form of FO.

Proof. The proof is constructed using a sequence of games. We use W_i to denote the event that the adversary wins, i.e., guesses the secret bit correctly, in Game i and the game returns 1.

Game 0 ($\mathsf{IND\text{-}CCA}$). *This is the original* $\mathsf{IND\text{-}CCA}$ *game against* $\mathsf{U}^{\mathsf{Kyber}}(\mathsf{P}, \mathsf{F}, H, H_1', H_2)$.

By definition we have $\mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{U}^{\mathsf{Kyber}}(\mathsf{P})}(\mathcal{A}) = \left|W_0 - \frac{1}{2}\right|$.

Game 1 (PRF is random). *Game 1 is the same as Game 0, except we replace* $\mathsf{F}(\mathrm{prfk}, \cdot)$ *with a random function* $R \xleftarrow{\$} \mathcal{K}^{\mathcal{C}_H}$.

We construct a PRF-adversary $\mathcal{B}_1$ which simulates Game 1 replacing calls to $\mathsf{F}(\mathrm{prfk}, \cdot)$ by calls to its oracle, runs $\mathcal{A}$, and outputs 1 if $\mathcal{A}$ wins and 0 otherwise. Now, by construction

$$\Pr_{k \xleftarrow{\$} \mathcal{K}}\left[\mathcal{B}_1^{\mathsf{F}(k,\cdot)} = 1\right] = W_0 \qquad \Pr_{R \xleftarrow{\$} \mathcal{K}^{\mathcal{C}_{H_2}}}\left[\mathcal{B}_1^{R(\cdot)} = 1\right] = W_1\,.$$

Hence,

$$|W_1 - W_0| = \mathrm{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(\mathcal{B}_1).$$

Game 2 (Replace H_1' by random permutation π). *Game 2 is the same as Game 1, but the game implements H_1' using a random permutation π for which it also knows the inverse π^{-1}.*

If $\mathcal{A}$ could change its behavior from one game to another, we could use $\mathcal{A}$ to distinguish random functions from random permutations. Indeed, one can construct an adversary $\mathcal{C}_2$ that, given oracle access to a function $\mathsf{F} : \mathcal{M} \to \mathcal{M}$ that is sampled uniformly at random from either all functions or a permutation π sampled uniformly at random from the set of all permutations with the respective domain and co-domain, simply sets H_1' as its own oracle and runs $\mathcal{A}$ according to the rules of Game 2. $\mathcal{C}_2$ returns 1 if A wins. If interacting with a random function, we perfectly simulate Game 1, otherwise, Game 2. By [Zha15], there exists a constant C such that the advantage of any q query quantum adversary in distinguishing a random function from a random permutation over $\mathcal{M}$ is upper-bounded by $Cq^3/|\mathcal{M}|$, hence,

$$|W_2 - W_1| \leq \mathrm{Adv}^{\mathrm{PRF\text{-}PRP}}_{H_1'}(\mathcal{C}_2) \leq \frac{C(q_1 + q_{\mathrm{dec}} + 1)^3}{|\mathcal{M}|}.$$

In the next games we will introduce changes to Decaps. We show Decaps_i for Game i in Figure 5 for clarity, as well as the change in H introduced in Game 6.

Game 3 (Exclude collisions in H_2 during decapsulation). *In this game we modify the decapsulation oracle once more. Game 3 is the same as Game 2, but the game keeps a list of all the decapsulation queries and their hashes $(c_i, H_2(c_i))$. On every new query c, the game checks for event* Bad_3*: there exists a pair $(c_i, H_2(c))$ in the list with $c \neq c_i$. If so, then the game aborts outputting a random bit.*

```
$\mathrm{Decaps}_2(\mathrm{sk}, c)$:

parse $\mathrm{sk} = (\mathrm{sk_P}, \mathrm{prfk})$
$m' \leftarrow \mathrm{Decr}(\mathrm{sk_P}, c)$
if $m' = \perp$:
    return $R(H_2(c))$
else if $\mathrm{Encr}(\mathrm{pk}, m') \neq c$:
    return $R(H_2(c))$
else
    return $H(\pi(m'), H_2(c))$
```

```
$\mathrm{Decaps}_3(\mathrm{sk}, c)$:

Append $(c, H_2(c))$ to $L$
if $\mathsf{Bad}_3$ then abort
parse $\mathrm{sk} = (\mathrm{sk_P}, \mathrm{prfk})$
$m' \leftarrow \mathrm{Decr}(\mathrm{sk_P}, c)$
if $m' = \perp$:
    return $R(H_2(c))$
else if $\mathrm{Encr}(\mathrm{pk}, m') \neq c$:
    return $R(H_2(c))$
else
    return $H(\pi(m'), H_2(c))$
```

```
$\mathrm{Decaps}_4(\mathrm{sk}, c)$:

Append $(c, H_2(c))$ to $L$
if $\mathsf{Bad}_3$ then abort
parse $\mathrm{sk} = (\mathrm{sk_P}, \mathrm{prfk})$
$m' \leftarrow \mathrm{Decr}(\mathrm{sk_P}, c)$
if $m' = \perp$:
    return $R'(c)$
else if $\mathrm{Encr}(\mathrm{pk}, m') \neq c$:
    return $R'(c)$
else
    return $H(\pi(m'), H_2(c))$
```

```
$H_6(\bar{K}, h_c)$:

$c' \leftarrow \mathrm{Encr}(\mathrm{pk}, \pi^{-1}(\bar{K}))$
if $H_2(c') = h_c$
    return $R'(c')$
Else return $H(\bar{K}, h_c)$
```

```
$\mathrm{Decaps}_7(c)$:

if $\mathsf{Bad}_5$ then abort
Append $(c, H_2(c))$ to $L$
if $\mathsf{Bad}_3$ then abort
return $R'(c)$
```

Fig. 5. Decapsulation oracle in Games 2 to 7. H_6 is H as reprogrammed in Game 6.

As decapsulation queries are classical, the notion of identical until Bad_3 is well defined. Furthermore, we construct an algorithm $\mathcal{C}_3$ that uses $\mathcal{A}$ to find collisions in H_2 whenever the games diverge. By [Zha15], there exists a constant C such that the probability of any q query quantum adversary in finding a collision in a function with co-domain $\mathcal{C}_H$ is upper-bounded by $Cq^3/|\mathcal{C}_H|$. Hence,

$$|W_3 - W_2| \leq \Pr[\ \mathsf{Bad}_3 : \mathit{Game}\ 3\ ] = \mathrm{Adv}^{\mathrm{Col}}_{H_2}(\mathcal{C}_3) \leq \frac{C(q_2 + q_{\mathrm{dec}} + 1)^3}{|\mathcal{C}_H|}.$$

One important observation at this point is that, on inputs $c \neq c'$ the Decapsulation oracle never places two queries to R such that $H_2(c) = H_2(c')$.

Game 4 (Replace $R(H_2(c))$ by random $R'(c)$). *Game 4 is the same as Game 3, but the game samples a fresh random function $R' \xleftarrow{\$} \mathcal{K}^{\mathcal{C}}$. And replaces all calls $R(H_2(c))$ in* Decaps *by $R'(c)$.*

Given that $\mathcal{A}$ only has indirect access to R via Decaps, and the game aborts upon colliding Decaps queries under H_2, this change cannot be noticed by $\mathcal{A}$ and we have $W_4 = W_3$.

Game 5 (Bad on fail or non-injective pk). *Let* Fail *be the event that the adversary submitted a ciphertext to decapsulation $D(c)$ which fails to decrypt because $c = \mathrm{Encr}(\mathrm{pk}, \mathrm{m})$ for some* m, *but $\mathrm{Decr}(\mathrm{sk}, c) = \mathrm{m}' \neq \mathrm{m}$ and $c \neq \mathrm{Encr}(\mathrm{pk}, \mathrm{m}')$. Let* NonInj *be the event that $\mathrm{Encr}(\mathrm{pk}, \cdot)$ is not injective, and let $\mathsf{Bad}_2 := \mathsf{Fail} \vee \mathsf{NonInj}$. In Game 5 and onward, if Bad_5 occurs then the game aborts outputting a random bit.*

Again, Bad_5 is a well-defined classical event and does not depend on H, H_1', H_2 (even though it might not be possible to determine efficiently whether it occurred). Let $\mathcal{B}_5$ be the algorithm which, given a public key pk, simulates Game 5 for $\mathcal{A}$ and outputs a list L of all of $\mathcal{A}$'s decapsulation queries. Then $\mathcal{B}_5$ is a FFC-adversary against P which runs in about the same time as $\mathcal{A}$ and succeeds whenever Fail occurred. Consequently,

$$|W_5 - W_4| \leq \Pr[\ \mathsf{Bad}_5\ ] \leq \Pr[\ \mathsf{Fail}\ ] + \Pr[\ \mathsf{NonInj}\ ] = \mathrm{Adv}_{\mathsf{P}}^{\mathsf{FFC}}(\mathcal{B}_5) + \epsilon.$$

Game 6 (Reprogram $H(\tilde{K}, x)$ to $R'(\mathrm{Encr}(\mathrm{pk}, \pi^{-1}(\tilde{K})))$. *Game 6 is the same as Game 5, but the game now reprograms $H(\tilde{K}, h_c)$ as follows: if $h_c = H_2(\mathrm{Encr}(\mathrm{pk}, \pi^{-1}(\tilde{K})))$ then return $R'(\mathrm{Encr}(\mathrm{pk}, \pi^{-1}(\tilde{K}))$ else return $H(\tilde{K}, h_c)$.*

We now show that this change has no impact on the adversary's view, so $W_6 = W_5$. This is the case because:

– For each $\tilde{K}$, only a single value $H(\tilde{K}, H_2(\mathrm{Encr}(\mathrm{pk}, \pi^{-1}(\tilde{K}))))$ is changed.
– This value is uniformly random because R' is uniformly random.
– This value is independent of all values of $H(\tilde{K}', h_c)$ for $\tilde{K}' \neq \tilde{K}$ because π is a permutation (thereby injective) and $\mathrm{Encr}(\mathrm{pk}, \cdot)$ is injective (as no abort occurred, and note that the input to R' is the full ciphertext).
– The previous observations imply that the distribution of the modified points will look independent when observed using only information collected from H. Indeed, the adversary could only detect this modification if it indirectly observes that the Decapsulation oracle is now returning correlated values caused by reprogramming H using R'.
– We know that reprogrammings use values $R'(c)$ corresponding to valid ciphertexts, meaning that $c = \mathrm{Encr}(\mathrm{pk}, m')$, for some m'. On the other hand, the decapsulation oracle only calls $R'(c')$ for rejected ciphertexts c', i.e. ones where $\mathrm{Decr}(\mathrm{sk}, c') = \perp$ or $c' \neq \mathrm{Encr}(\mathrm{pk}, \mathrm{Decr}(\mathrm{sk}, c'))$. A valid ciphertext being rejected and passed to R' in Decapsulation is excluded by Bad_2. This means that direct calls to R' by the Decapsulation oracle do not help the adversary detect reprogrammings.
– It remains the possibility that the decapsulation of a non-rejected (and hence valid) ciphertext uses a value of R' via the reprogrammed H that could also be used in decapsulating a rejected ciphertext. However, this is again excluded by the above observation that all reprogrammed values correspond to valid ciphertexts.

Game 7 (Decapsulation oracle returns $R'(c)$). *Game 7 is the same as Game 6, but the decapsulation oracle simply returns $R'(c)$ for all ciphertexts (other than the challenge).*

This modification changes nothing. In fact, the decapsulation oracle was already using R' on all execution paths in Game 6: The original decapsulation returns either $H(\tilde{K}, H_2(c))$ with $c = \mathrm{Encr}(\mathrm{pk}, \mathrm{Decr}(\mathrm{sk}, c))$ or $\mathsf{F}(\mathrm{prfk}, H_2(c))$, but both of those have been changed to return $R'(c)$. Therefore $W_7 = W_6$. As of this game, the private key is not used anymore.

Game 8 (Change shared secret encrypted by challenge ciphertext). *In Game 8, the shared secret is changed to a uniformly random value r. More in detail, if $b = 1$, then for all inputs $(\tilde{K}, h_c)$ such that $\mathrm{Encr}(\mathrm{pk}, \pi^{-1}(\tilde{K})) = c^*$ and $h_c = H_2(c^*)$, the oracle $H(\tilde{K}, h_c)$ is reprogrammed to return r. If $b = 0$, then H is not reprogrammed.*

If $\mathrm{Encr}(\mathrm{pk}, \cdot)$ is injective, which is the case if the game does not abort, then this is the same distribution as Game 7. Therefore $W_8 = W_7$.

Game 9 (Remove aborts on Fail). *In this game, we stop aborting due to event* NonInj *introduced in Game 5.*

Following the same up to bad reasoning, let $\mathcal{B}_9$ be the algorithm which, given a public key pk, simulates Game 9 for $\mathcal{A}$ and outputs a list L of all of $\mathcal{A}$'s decapsulation queries. Note that, in order to be efficient, $\mathcal{B}_9$ cannot simulate the check for NonInj, so we get again:

$$|W_9 - W_8| \leq \mathrm{Adv}_{\mathsf{P}}^{\mathsf{FFC}}(\mathcal{B}_9) + \epsilon.$$

It remains to bound $\mathcal{A}$'s advantage in Game 9. Excluding the detection of the non-injectivity of $\mathrm{Encr}(\mathrm{pk}, \cdot)$, the game still runs in about the same time as $\mathcal{A}$.[4]

Let us condition our analysis on $\mathrm{Encr}(\mathrm{pk}, \cdot)$ being injective, so that the oracle H is reprogrammed only at m^*. Then the $b = 0$ and $b = 1$ cases are now distinguished by a single return value from the H oracle. Hence, we can consider two oracles H and $H' := H[m^* \to r]$ as required by Lemma 1. Then Lemma 1, states that there is an algorithm $\mathcal{B}$, running in about the same time as $\mathcal{A}$, such that for all such pk*:

$$|\Pr[\mathrm{Win}|b = 1 \wedge \mathrm{pk} = \mathrm{pk}^*] - \Pr[\mathrm{Lose}|b = 0 \wedge \mathrm{pk} = \mathrm{pk}^*]| \leq 2\sqrt{\Pr[\mathcal{B}(\mathrm{pk}^*, c) \to m^*]}.$$

Here, c is defined according to the distribution of the challenge ciphertext in the CCA experiment. Note also that we are checking on both sides for the same classical event: the adversary returning 1. The same inequality holds if $\mathrm{Encr}(\mathrm{pk}, \cdot)$ is not injective, for then the adversary wins with probability $1/2$ and the left-hand side is zero. (The algorithm $\mathcal{B}$ still runs with the same efficiency in that case; it just might not return m^*.)

The inequality also holds in expectation over pk by Jensen's inequality:

$$\begin{aligned}
\mathrm{E}&\left[2\sqrt{\Pr[\mathcal{B}(\mathrm{pk}, c) \to m^*]} : (\mathrm{pk}, \mathrm{sk}) \xleftarrow{\$} \mathrm{Keygen}_{\mathsf{P}}()\right] \\
&\leq 2\sqrt{\mathrm{E}\left[\Pr[\mathcal{B}(\mathrm{pk}, c) \to m^*] : (\mathrm{pk}, \mathrm{sk}) \xleftarrow{\$} \mathrm{Keygen}_{\mathsf{P}}()\right]} \\
&= 2\sqrt{\mathrm{Adv}_{\mathsf{P}}^{\mathsf{OW\text{-}CPA}}(\mathcal{B})}
\end{aligned}$$

so that

$$|\Pr[\mathrm{Win} : b = 0] - \Pr[\mathrm{Lose} : b = 1]| \leq 2\sqrt{\mathrm{Adv}_{\mathsf{P}}^{\mathsf{OW\text{-}CPA}}(\mathcal{B})}.$$

and finally $\left|w_9 - \frac{1}{2}\right| \leq \sqrt{\mathrm{Adv}_{\mathsf{P}}^{\mathsf{OW\text{-}CPA}}(\mathcal{B})}$.

Summing up the differences in the previous games, we have

$$\left|w_0 - \frac{1}{2}\right| \leq \sqrt{\mathrm{Adv}_{\mathsf{P}}^{\mathsf{OW\text{-}CPA}}(\mathcal{B})} + \mathrm{Adv}_{\mathsf{P}}^{\mathsf{FFC}}(\mathcal{B}_9) + \mathrm{Adv}_{\mathsf{P}}^{\mathsf{FFC}}(\mathcal{B}_5) + 2\epsilon + \mathrm{Adv}_{\mathsf{F}}^{\mathsf{PRF}}(\mathcal{B}_1) +$$

$$\frac{C(q_1 + q_{\mathrm{dec}} + 1)^3}{|\mathcal{M}|} + \frac{C(q_2 + q_{\mathrm{dec}} + 1)^3}{|\mathcal{C}_H|}$$

The proof of Theorem 1 follows from the definition of CCA advantage. □

[4] Recall that we take it as a given that π and the various random oracles can be efficiently simulated, possibly at the cost of relying on additional computational assumptions.

References

BDF+11. Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Heidelberg, December 2011. `doi:10.1007/978-3-642-25385-0_3`.

BHH+19. Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 61–90. Springer, Heidelberg, December 2019. `doi:10.1007/978-3-030-36033-7_3`.

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

GMP22. Paul Grubbs, Varun Maram, and Kenneth G. Paterson. Anonymous, robust post-quantum public key encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 402–432. Springer, 2022. `doi:10.1007/978-3-031-07082-2\_15`.

HHK17. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Heidelberg, November 2017. `doi:10.1007/978-3-319-70500-2_12`.

HKSU20. Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. Generic authenticated key exchange in the quantum random oracle model. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 389–422. Springer, Heidelberg, May 2020. `doi:10.1007/978-3-030-45388-6_14`.

JZC+18. Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 96–125. Springer, Heidelberg, August 2018. `doi:10.1007/978-3-319-96878-0_4`.

JZM19a. Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. Key encapsulation mechanism with explicit rejection in the quantum random oracle model. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 618–645. Springer, Heidelberg, April 2019. `doi:10.1007/978-3-030-17259-6_21`.

JZM19b. Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. Tighter security proofs for generic key encapsulation mechanism in the quantum random oracle model. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 227–248. Springer, Heidelberg, 2019. `doi:10.1007/978-3-030-25510-7_13`.

MX23. Varun Maram and Keita Xagawa. Post-quantum anonymity of Kyber. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 3–35. Springer, Heidelberg, May 2023. `doi:10.1007/978-3-031-31368-4_1`.

SXY18. Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 520–551. Springer, Heidelberg, April / May 2018. `doi:10.1007/978-3-319-78372-7_17`.

Unr15. Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 755–784. Springer, Heidelberg, April 2015. `doi:10.1007/978-3-662-46803-6_25`.

Zha15. Mark Zhandry. A note on the quantum collision and set equality problems. *Quantum Info. Comput.*, 15(7–8):557–567, May 2015.