The security of Kyber’s FO-transform

Manuel Barbosa and Andreas Hülsing

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 $\text{FO}_{\perp}$ and obtain it by combining the $T$ and the $U_{\perp}$ transforms. Figure 2 shows the Kyber construction, which we call FO$^{\text{Kyber}}$. The Kyber transform FO$^{\text{Kyber}}$ deviates from the original transform FO$^{T}$ 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 $(\tilde{K}, r) \leftarrow H_1(m, H_3(pk))$ in line 2 of Encaps). We note that, in the security proof, one can ignore the $H_3(pk)$ 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 $U_{\perp}$ 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 $U_{\perp}$ 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
Indeed, as ciphertext pair \((\text{relies on a technique that uses a plaintext checking oracle — an oracle PCO that given a plaintext-remaining obstacle. While the authors do not explicitly state what the issue is, their proof strategy FO complete proof for Keygen():

\begin{align*}
\text{Keygen():} & \quad 1 \quad (pk, sk_p) \leftarrow \text{Keygen}_p() \\
& \quad 2 \quad \text{prfk} \leftarrow \text{K}_\text{prf} \\
& \quad 3 \quad \text{sk} \leftarrow (sk_p, \text{prfk}) \\
& \quad 4 \quad \text{return } (pk, sk)
\end{align*}

\begin{align*}
\text{Encaps(pk):} & \quad 1 \quad m \leftarrow \mathcal{M} \\
& \quad 2 \quad r \leftarrow G(m) \\
& \quad 3 \quad c \leftarrow \text{Encr}(pk, m; r) \\
& \quad 4 \quad K \leftarrow H(m, c) \\
& \quad 5 \quad \text{return } (K, c)
\end{align*}

\begin{align*}
\text{Decaps(sk, c):} & \quad 1 \quad \text{parse } sk = (sk_0, \text{prfk}) \\
& \quad 2 \quad m' \leftarrow \text{Decr}(sk_0, c) \\
& \quad 3 \quad \text{if } m' = \bot: \\
& \quad 4 \quad \text{return } F(\text{prfk}, c) \\
& \quad 5 \quad \text{else if } \text{Encr}(pk, m'; G(m')) \neq c: \\
& \quad 6 \quad \text{return } F(\text{prfk}, c) \\
& \quad 7 \quad \text{else: return } H(m', c)
\end{align*}

\textbf{Fig. 1.} Transform FO\(^{\text{rel}}\)(P, F, G, H) := (Keygen, Encaps, 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 \(H(m, c) := H'(\text{Encr}(pk, m; G(m)))\) 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 \(\text{Decaps'}(c) := H'(c)\) without having knowledge of the secret key.

This strategy is not directly applicable to FO\(^{\text{Kyber}}\), first of all, because the key derivation step only gets the pre-key \(K_1 = (H_1(m))_1\), rather than \(m\) itself. This issue was observed and is solved by \([\text{GMP22}]\) 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 \(H' := (H_1)_1\) and set \(H(m, c) := H'(\text{Encr}(pk, H_1^{-1}(K); G(m)))\). However, the authors of \([\text{GMP22}]\) did not present a complete proof for FO\(^{\text{Kyber}}\), 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 \((K, H_2(c))\). In this note we show that there is no problem because the alternative proof strategy in \([\text{BHH}+19]\) does not require the explicit construction of a PCO. As already conjectured in \([\text{GMP22}]\) 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 \([\text{MX23}]\). There, the authors reduce the IND-CCA security of FO\(^{\perp}\) to that of FO\(^{\text{Kyber}}\) in an elegant way. Their approach avoids struggling with implementing the decapsulation oracle, by using the oracle of 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 \([\text{Unr15}]\).
Keygen():
1. \((pk, sk) \leftarrow \text{Keygen}()\)
2. prfk \leftarrow K\_f
3. \(sk \leftarrow (sk\_P, pk, H_3(pk), \text{prfk})\)
4. return \((pk, sk)\)

Encaps(pk):
1. \(m \leftarrow M\)
2. \((K, r) \leftarrow H_1(m, H_3(pk))\)
3. \(c \leftarrow \text{Encr}(pk, m; r)\)
4. \(K \leftarrow H(K, H_2(c))\)
5. return \((K, c)\)

Decaps(sk, c):
1. parse sk = \((sk\_P, pk, h_{pk}, \text{prfk})\)
2. \(m' \leftarrow \text{Decr}(sk\_P, c)\)
3. if \(m' = \bot\):
4. return \(F(\text{prfk}, H_2(c))\)
5. else if \(\text{Encr}(pk, m'; (H_1(m', h_{pk}))_2) \neq c:\)
6. return \(F(\text{prfk}, H_2(c))\)
7. else: return \(H((H_1(m'))_1, H_2(c))\)

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

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

We now give a security analysis of the \(\text{U} \neq \bot\) part of \(\text{FO}\_\text{Kyber}\) (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 \(\text{U}\_\text{Kyber}\) with the output of the standard \(T\) transform, which in turn is instantiated with the Kyber IND-CPA-secure PKE \((P\_\text{Kyber})\). Recall that the standard \(T\) transform yields a deterministic encryption scheme \(P := T(P\_\text{Kyber}, G)\), which is shown to be a OW-CPA secure dPKE in \([BHH+19]\).

The proof of \(\text{U}\_\text{Kyber}\) 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 \(\epsilon\)-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 \(A\) is defined by

\[
\text{Adv}^\text{FFC}_P(A) := \Pr[\text{Expt}^\text{FFC}_P(A) \rightarrow 1].
\]

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 \(P = (\text{Keygen}, \text{Encr}, \text{Decr})\) is \(\epsilon\)-injective if

\[
\Pr[\text{Encr}(pk, m) \text{ is not injective} : (pk, sk) \leftarrow \text{Keygen()}, H \leftarrow \mathcal{H}] \leq \epsilon.
\]

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 \(\text{Encr}(pk, m, r) \neq \text{Encr}(pk, m', r')\).

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 \(G, H : X \rightarrow Y\) 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 \(A^H\) be a quantum oracle algorithm. Let \(f : X \rightarrow W \subset \{0,1\}^n\) be any function, and let \(f(S)\) denote the image of \(S\) under \(f\). Let \(\mathcal{E_v}\) 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 \(A\), but when \(A\) queries \(H, H\) random oracles, let \(F_{\text{Keygen}}()\):

\[
\begin{align*}
\text{Keygen}(): & \quad (\text{pk}, \text{sk}) \leftarrow \text{Keygen}_{\text{p}}() \\
\text{Encaps}(\text{pk}): & \quad m \leftarrow M \\
& \quad \tilde{K} \leftarrow H_1^2(m) \\
& \quad c \leftarrow \text{Encr}(\text{pk}, m) \\
& \quad K \leftarrow H(\tilde{K}, H_2(c)) \\
& \quad \text{return } (K, c) \\
\text{Decaps}(\text{sk}, c): & \quad \text{parse } \text{sk} = (\text{sk}, \text{prfk}) \\
& \quad m' \leftarrow \text{Decr}(\text{sk}, c) \\
& \quad \text{if } m' = \bot: \\
& \quad \quad \text{return } F(\text{prfk}, H_2(c)) \\
& \quad \quad \text{else if } \text{Encr}(\text{pk}, m') \neq c: \\
& \quad \quad \quad \text{return } F(\text{prfk}, H_2(c)) \\
& \quad \quad \text{else } \\
& \quad \quad \quad \text{return } H(H_1^2(m'), H_2(c))
\end{align*}
\]

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

The following theorem which states that breaking the IND-CCA security of \(U_{\text{Kyber}}(P, F, H, H_1', H_2)\) requires either breaking the OW-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 \(\epsilon\)-injective dPKE as in Definition 2.

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

- a OW-CPA-adversary \(B\) against \(P\)
- a PRF-adversary \(B_1\) against \(F\)
- FFC-adversaries \(B_5\) and \(B_9\) against \(P\), returning a list of at most \(q_{\text{dec}}\) ciphertexts

such that

\[
\text{Adv}_{\text{IND-CCA}}^{U_{\text{Kyber}}(P)}(A) \leq 2 \cdot \left( \sqrt{\text{Adv}_{\text{OW-CPA}}^P(B)} + \text{Adv}_{\text{PRF}}^F(B_1) + \text{Adv}_{\text{FFC}}^P(B_5) + \text{Adv}_{\text{FFC}}^P(B_9) + 2\epsilon \\
+ \frac{C(q_1 + q_{\text{dec}} + 1)^3}{|\mathcal{M}|} + \frac{C(q_2 + q_{\text{dec}} + 1)^3}{|C_H|} \right),
\]

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 \cite{BHH19} 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 \(\epsilon\)-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 (IND-CCA).** This is the original IND-CCA game against \(U^{Kyber}(P, F, H, H'_1, H_2)\).

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

**Game 1 (PRF is random).** Game 1 is the same as Game 0, except we replace \(F(prf_k, \cdot)\) with a random function \(R \leftarrow_{K} \mathcal{C}^H\).

We construct a PRF-adversary \(B_1\) which simulates Game 1 replacing calls to \(F(prf_k, \cdot)\) by calls to its oracle, runs \(A\), and outputs 1 if \(A\) wins and 0 otherwise. Now, by construction

\[
\begin{align*}
\Pr_{k \leftarrow_{K}} \left[ B_1^{F(k, \cdot)} = 1 \right] &= W_0 \\
\Pr_{R \leftarrow_{K^{CH}}} \left[ B_1^{R(\cdot)} = 1 \right] &= W_1.
\end{align*}
\]

Hence,

\[
|W_1 - W_0| = \text{Adv}^\text{PRF}_{F}(B_1).
\]

**Game 2 (Replace \(H'_1\) by random permutation \(\pi\)).** Game 2 is the same as Game 1, but the game implements \(H'_1\) using a random permutation \(\pi\) for which it also knows the inverse \(\pi^{-1}\).

If \(A\) could change its behavior from one game to another, we could use \(A\) to distinguish random functions from random permutations. Indeed, one can construct an adversary \(C_2\) that, given oracle access to a function \(F : \mathcal{M} \rightarrow \mathcal{M}\) that is sampled uniformly at random from either all functions or a permutation \(\pi\) 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 \(A\) according to the rules of Game 2. \(C_2\) returns 1 if \(A\) wins. If interacting with a random function, we perfectly simulate Game 1, otherwise, Game 2. By \cite{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,

\[
|W_2 - W_1| \leq \text{Adv}^{\text{PRF-PRP}}_{H'_1}(C_2) \leq \frac{C(q_1 + q_{\text{dec}} + 1)^3}{|\mathcal{M}|}.
\]
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 $\text{Bad}_3$: 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.

<table>
<thead>
<tr>
<th>Decaps$_2$(sk, c):</th>
<th>Decaps$_3$(sk, c):</th>
<th>Decaps$_4$(sk, c):</th>
<th>$H_6(\bar{K}, h_c)$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 parse sk = (sk$_p$, prfk)</td>
<td>1 Append $(c, H_2(c))$ to $L$</td>
<td>1 Append $(c, H_2(c))$ to $L$</td>
<td>1 $c' \leftarrow \text{Encr}(\bar{K}, \pi^{-1}(\bar{K}))$</td>
</tr>
<tr>
<td>2 $m' \leftarrow \text{Decr}(sk_p, c)$</td>
<td>2 if $\text{Bad}_3$ then abort</td>
<td>2 if $\text{Bad}_3$ then abort</td>
<td>2 if $H_2(c') = h_c$</td>
</tr>
<tr>
<td>3 if $m' = \perp$:</td>
<td>3 parse sk = (sk$_p$, prfk)</td>
<td>3 parse sk = (sk$_p$, prfk)</td>
<td>3 return $R'(c')$</td>
</tr>
<tr>
<td>4 return $R(H_2(c))$</td>
<td>4 $m' \leftarrow \text{Decr}(sk_p, c)$</td>
<td>5 if $m' = \perp$:</td>
<td>4 Else return $H(\bar{K}, h_c)$</td>
</tr>
<tr>
<td>5 else if $\text{Encr}(pk, m') \neq c$:</td>
<td>6 return $R(H_2(c))$</td>
<td>6 return $R'(c)$</td>
<td>Decaps$_5$(c):</td>
</tr>
<tr>
<td>6 return $R(H_2(c))$</td>
<td>7 else if $\text{Encr}(pk, m') \neq c$:</td>
<td>7 else if $\text{Encr}(pk, m') \neq c$:</td>
<td>1 if $\text{Bad}_5$ then abort</td>
</tr>
<tr>
<td>8 return $H(\pi(m'), H_2(c))$</td>
<td>8 return $R(H_2(c))$</td>
<td>8 return $R'(c)$</td>
<td>2 Append $(c, H_2(c))$ to $L$</td>
</tr>
<tr>
<td>9 else</td>
<td>9 else</td>
<td>9 else</td>
<td>3 if $\text{Bad}_5$ then abort</td>
</tr>
<tr>
<td>10 return $H(\pi(m'), H_2(c))$</td>
<td>10 return $H(\pi(m'), H_2(c))$</td>
<td>10 return $H(\pi(m'), H_2(c))$</td>
<td>4 return $R'(c)$</td>
</tr>
</tbody>
</table>

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 $\text{Bad}_5$ is well defined. Furthermore, we construct an algorithm $C_3$ that uses $A$ to find collisions in $H_2$ whenever the games diverge. By \cite{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 $C_H$ is upper-bounded by $Cq^3/|C_H|$. Hence,

$$|W_3 - W_2| \leq \Pr[\text{Bad}_3 : \text{Game 3}] = \text{Adv}^{\text{Col}}_{H_2}(C_3) \leq \frac{C(q_2 + q_{\text{dec}} + 1)^3}{|C_H|}.$$

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 $R' \leftarrow \mathcal{K}^c$. And replaces all calls $R(H_2(c))$ in Decaps by $R'(c)$.

Given that $A$ 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 $A$ and we have $W_4 = W_3$.

**Game 5 (Bad on fail or non-injective pk).** Let $\text{Fail}$ be the event that the adversary submitted a ciphertext to decapsulation $D(c)$ which fails to decrypt because $c = \text{Encr}(pk, m)$ for some $m$, but $\text{Decr}(sk, c) = m' \neq m$ and $c \neq \text{Encr}(pk, m')$. Let $\text{NonInj}$ be the event that $\text{Encr}(pk, \cdot)$ is not injective, and let $\text{Bad}_5 := \text{Fail} \lor \text{NonInj}$. In Game 5 and onward, if $\text{Bad}_5$ occurs then the game aborts outputting a random bit.

Again, $\text{Bad}_5$ 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 $B_5$ be the algorithm which,
given a public key $pk$, simulates Game 5 for $A$ and outputs a list $L$ of all of $A$’s decapsulation queries. Then $B_5$ is a FFC-adversary against $P$ which runs in about the same time as $A$ and succeeds whenever $\text{Fail}$ occurred. Consequently,

$$|W_5 - W_4| \leq \Pr[\text{Bad}_5] \leq \Pr[\text{Fail}] + \Pr[\text{NonInj}] = \text{Adv}^\text{FFC}_P(B_5) + \epsilon.$$

**Game 6 (Reprogram $H(\tilde{K}, x)$ to $R'(\text{Encr}(pk, \pi^{-1}(\tilde{K})))$).** Game 6 is the same as Game 5, but the game now reprograms $H(\tilde{K}, h_c)$ as follows: if $h_c = H_2(\text{Encr}(pk, \pi^{-1}(\tilde{K})))$ then return $R'(\text{Encr}(pk, \pi^{-1}(\tilde{K})))$ 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 $H(\tilde{K}, H_2(\text{Encr}(pk, \pi^{-1}(\tilde{K}))))$ 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 $\pi$ is a permutation (thereby injective) and $\text{Encr}(pk, \cdot)$ 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 $c = \text{Encr}(pk, m')$, for some $m'$. On the other hand, the decapsulation oracle only calls $R'(c')$ for rejected ciphertexts $c'$, i.e. ones where $\text{Decr}(sk, c') = \bot$ or $c' \neq \text{Encr}(pk, \text{Decr}(sk, c'))$. A valid ciphertext being rejected and passed to $R'$ in Decapsulation is excluded by $\text{Bad}_2$. 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 $c = \text{Encr}(pk, \text{Decr}(sk, c))$ or $F(\text{prfk}, H_2(c))$, 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 $\text{Encr}(pk, \pi^{-1}(\tilde{K})) = c^*$ 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 $\text{Encr}(pk, \cdot)$ 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 $B_9$ 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, $B_9$ cannot simulate the check for NonInj, so we get again:

$$|W_9 - W_8| \leq \text{Adv}^{\text{FFC}}_P(B_9) + \epsilon.$$  

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

Let us condition our analysis on Encr$(pk, \cdot)$ 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 $B$, running in about the same time as $\mathcal{A}$, such that for all such $pk^*$:

$$|\Pr[\text{Win} : b = 1 \land pk = pk^*] − \Pr[\text{Lose} : b = 0 \land pk = pk^*]| \leq 2\sqrt{\Pr[B(pk^*, c) \to m^*]}.$$  

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 Encr$(pk, \cdot)$ is not injective, for then the adversary wins with probability $1/2$ and the left-hand side is zero. (The algorithm $B$ 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:

$$\mathbb{E} \left[ 2\sqrt{\Pr[B(pk, c) \to m^*] : (pk, sk) \leftarrow \text{Keygen}_P()} \right] \leq 2\sqrt{\mathbb{E} \left[ \Pr[B(pk, c) \to m^*] : (pk, sk) \leftarrow \text{Keygen}_P() \right]} = 2\sqrt{\text{Adv}^{\text{OW-CPA}}_P(B)}$$

so that

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

and finally $w_9 - \frac{1}{2} \leq \sqrt{\text{Adv}^{\text{OW-CPA}}_P(B)}$.

Summing up the differences in the previous games, we have

$$w_0 - \frac{1}{2} \leq \sqrt{\text{Adv}^{\text{OW-CPA}}_P(B) + \text{Adv}^{\text{FFC}}_P(B_9) + \text{Adv}^{\text{FFC}}_P(B_5) + 2\epsilon + \text{Adv}^{\text{PRF}}_F(B_1) + \frac{C(q_1 + q_{\text{dec}} + 1)^3}{|\mathcal{M}|} + \frac{C(q_2 + q_{\text{dec}} + 1)^3}{|\mathcal{C}_H|}}.$$  

The proof of Theorem 1 follows from the definition of CCA advantage. \[\square\]

\[^4\text{Recall that we take it as a given that } \pi \text{ and the various random oracles can be efficiently simulated, possibly at the cost of relying on additional computational assumptions.}\]
References


