FO-like Combiners and Hybrid Post-Quantum Cryptography

Loïs Huguenin-Dumittan and Serge Vaudenay

EPFL, Switzerland
{lois.huguenin-dumittan,serge.vaudenay}@epfl.ch

Abstract. Combining several primitives together to offer greater security is an old idea in cryptography. Recently, this concept has resurfaced as it could be used to improve trust in new Post-Quantum (PQ) schemes and smooth the transition to PQ cryptography. In particular, several ways to combine key exchange mechanisms (KEMs) into a secure hybrid KEM have been proposed. In this work, we observe that most PQ KEMs are built using a variant of the Fujisaki-Okamoto (FO) transform. Thus, we propose several efficient combiners that take OW-CPA public-key encryption schemes (PKEs) and directly build hybrid IND-CCA KEMs. Our constructions are secure in the ROM and QROM and can be seen as generalizations of the FO transform. We also study how the hash functions (ROs) used in our transforms can be combined in order to improve efficiency and security. In a second part, we implement a hybrid KEM using one of our combiners as a proof-of-concept and benchmark it. More precisely, we build a hybrid IND-CCA KEM from the CPA-secure versions of HQC and LAC, two NIST Round 2 PQ proposals. We show that the resulting KEM offers comparable performances to HQC, thus improving security at a small cost. Finally, we discuss which PQ schemes should be combined in order to offer the best efficiency/security trade-off.

1 Introduction

Redundancy is one of the most important concepts of computer science, mostly used to prevent the failure of one component affecting the whole system. In cryptography, the same idea has been used under different terms and in different forms. For instance, increasing the security of DES by performing multiple encryptions was studied by Merkle and Hellman in 1981 [18] and in 2005, Herzberg [13] studied so-called *tolerant* encryption schemes, which remain secure even if one or several of their components are broken. However, the topic became popular in the last few years, following the launch of the post-quantum (PQ) standardization process.

As promising developments have been made in the development of quantum computers, the need for secure post-quantum public-key cryptography (PKC) primitives is pressing. This led the US National Institute of Standards and Technology (NIST) to launch a post-quantum standardization process for public-key

encryption (PKE), key-exchange mechanisms (KEMs) and signatures in 2017. In the second round, 26 proposals were retained and only 7 have been selected for the third round of this process (+8 "alternate candidates").

Most of the assumptions the PQ schemes are based on (e.g. learning with errors, syndrome decoding) have been less extensively studied than their classical counterparts (e.g. factorization, discrete logarithm). Thus, combining several of these schemes into one is considered a sound idea. For example, one could combine both a standard PKE/KEM scheme with a PQ one, and ideally the resulting cryptosystem should be secure as long as one of the underlying schemes is secure. Such systems have been popularized under the term *hybrid* schemes and the way the underlying systems are combined is called a *combiner*. Moreover, if the resulting hybrid scheme is secure as long as one of the underlying systems is secure, the *combiner* is said to be *robust*.

When it comes to PQ cryptography, hybrid schemes have many advantages, such as:

1. Guaranteeing security as long as practical quantum computers do not exist as discussed above.
2. Fulfilling the standards requirement by combining a standardized scheme with another one which is not. This possibility is actively considered by NIST[1].
3. Allowing a smooth transition between classical and PQ cryptography in practice. Hybrid cryptography would allow support of both classical and PQ schemes, allowing compatibility between older and newer systems.
4. Combining multiple PQ schemes together might offer better confidence as most of the problems/assumptions are newer and less studied. Such hybrid schemes would come at the cost of efficiency, however combining two efficient schemes might result in a more efficient scheme than one inefficient one. Such ideas and issues were briefly discussed on the NIST PQ forum[2]. We focus mostly on this application of hybrid systems in this work.

Unfortunately, hybrid schemes do not offer much improvement in terms of theoretical security. Indeed, if both underlying schemes require 2^λ operations to be broken, the hybrid system would be broken in $2^{\lambda+1}$ operations (i.e. we gain only 1 bit of security). In practice however, the security gain might be better, depending on the underlying schemes. Indeed, one might reasonably argue that the probability of a major breakthrough in two different problems believed to be hard by the community is lower than the probability of one (but even more devastating) breakthrough. In any case, while the practical security offered by hybrid cryptosystems obviously depends on many parameters, we think that such schemes offer a greater security boost than what can be deduced from the theoretical bounds only.

[1] https://csrc.nist.gov/Projects/post-quantum-cryptography/faqs

[2] https://groups.google.com/a/list.nist.gov/forum/#!topic/pqc-forum/msRrR13muS4

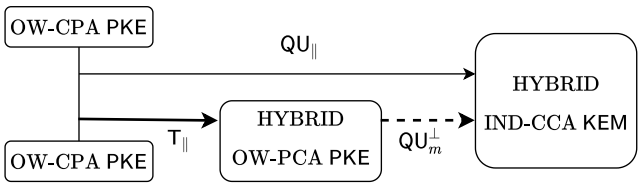


Fig. 1: Solid arrows indicate results implied by our combiners, bold arrows indicate QROM security. The dashed arrow indicates results from Hofheinz et al. [14].

Our contributions

Several authors have considered KEM or signature combiners targeting post-quantum systems in recent years [5,3,11]. However, all the combiners introduced in these papers work in a black-box manner on IND-CCA KEMs. That is, combiners that take two KEMs (or signature schemes) as inputs and output the hybrid construction. Yet, most PQ KEM proposals share a very similar structure: an OW/IND-CPA secure PKE is introduced and then the Fujisaki-Okamato (FO) transform or a variant (e.g. [9,10,14,22]) is applied to give an IND-CCA KEM. Therefore, one could try to directly combine the IND-CPA schemes to give an IND-CCA KEM, hopefully getting better performances. Therefore, we present in this report several hybrid FO-like transforms which combine two OW-CPA PKEs into one IND-CCA KEM. We also generalize these constructions to n schemes (i.e. n PKEs are combined into one KEM).

Compared to previous work, our combiners are simpler as they do not require extra primitives such as special types of PRFs or MACs. As a result, they are slightly more efficient by removing calls to these primitives and by optimizing the use of hash functions. Finally, our combiners follow a different paradigm as they replace FO transforms. Thus, they would likely be implemented in cryptographic libraries directly, whereas previous combiners would likely be implemented in applications/protocol libraries (e.g. openssl). Hence, our constructions offer another approach that might be useful to implementors, for example for optimization or security purposes.

The main disadvantage of FO transforms is that they are only secure in the random oracle model (ROM) and we prove the security of our FO-like hybrid combiner in the ROM as well. However, as all PQ IND-CCA KEM submitted to the NIST process are only proven secure in the ROM, it does not add an extra assumption. We also prove that one of our combiners is secure in the Quantum Random Oracle Model (QROM). The results are summarized in Figure 1.

At a high level, our combiners share the same structure as a system that would apply a robust PKE combiner (e.g. concatenating ciphertexts) followed by a FO-like transform to get a KEM. However, having one scheme for the whole process allows a fine-grained control over the way key derivation and derandomization are performed, in turn offering better flexibility. For instance, we study how one can combine hash functions (i.e. random oracles) s.t. our main FO-like combiner is more efficient or secure. More precisely, we define the properties

the functions g (used to derive random coins in our construction) and h (used to derive the shared key) should have in order for our construction to be secure. Such theoretical analysis is important, as it was demonstrated that Random Oracles in FO transforms are easily misimplemented in practice [1]. Therefore, by presenting generic n-PKEs-to-KEM combiners with detailed security proofs and several examples of ROs combinations, we hope to offer clear flexibility and security guarantees to implementors.

As a proof of concept, we implemented a hybrid KEM based on the IND-CPA version of HQC and LAC, two round 2 proposals to the NIST PQ standardisation process. We call this hybrid KEM `hqc_lac_128` and we report and analyse how this scheme compares to the other round 2 proposals[3]. In particular, we show that the performance of the hybrid scheme is comparable to the performance of the least efficient underlying scheme (i.e. HQC in this case). Moreover, as our combiner is highly parallelizable, our tests show that a parallelized version of `hqc_lac_128` is as efficient as HQC in term of speed, excluding a negligible overhead (mainly due to the creation of an additional thread). We think this demonstrates that using a hybrid PQ system in place of a single PQ scheme may increase significantly the security at a small cost.

Finally, we compute the theoretical performance (based on the data from eBACS [2]) of every possible hybrid scheme based on two PQ IND-CPA schemes that are based on assumptions of a different type (e.g. a lattice-based scheme with a code-based scheme). We discuss the performance of the most efficient ones in two metrics, namely public key/ciphertext size and encapsulation/decapsulation speed. This analysis shows that a given hybrid scheme struggles to perform as well as an efficient non-hybrid one in both metrics.

Related work

Many authors have considered robust combiners for different primitives, like combiners for PKE [6,25], hash functions [7,8], commitment schemes [13], PQ signatures [5], AEAD [20]. Recently, robust combiners for KEM have also been considered by Giacon et al. in [11]. In that work, they propose various robust combiners in the standard model and in the random oracle model that take two IND-CCA KEM and output another IND-CCA KEM. Similarly, Bindel et al. [3] propose similar robust KEM combiners which are secure against quantum adversaries. Our combiners differ from these as we aim at building a monolithic IND-CCA KEM based on several IND-CPA (or OW-CPA) PKEs. In a way, we bypass the intermediate KEM constructions, as many KEMs are based on FO-transformed IND-CPA schemes.

Another related line of work is the construction of Fujisaki-Okamoto (FO)-like transforms [9,10], which have been a hot topic these last years. Several variants meant to be secure in the quantum random oracle (QROM) have been proposed along with tighter security proofs [14,22,21,17]. Our combiner can be

[3] At the time of the tests, round 3 proposals were not announced.

seen as a generalization of a FO-like transform as it takes multiples CPA-secure PKEs and outputs a robust IND-CCA KEM.

2 Notation

Let $\mathcal{A}$ be a randomized algorithm, then we write $b \leftarrow\!\$\, \mathcal{A}$ to indicate b is assigned the value output by $\mathcal{A}$. Similarly, if Ψ (resp. $\mathcal{X}$) is a distribution (resp. a set), then $x \leftarrow\!\$\, \Psi$ (resp. $x \leftarrow\!\$\, \mathcal{X}$) means that x is sampled uniformly at random from Ψ (resp. $\mathcal{X}$). We denote by 1_P the indicator function which returns 1 if the predicate P is fulfilled and 0 otherwise. We write $[n]$ the set $\{0, 1, \ldots, n-1\}$.

Let $\mathcal{A}$ be an algorithm. Then, we write $\mathcal{A} \Rightarrow b$ to denote the event $\mathcal{A}$ *outputs* b. Finally, in an algorithm (or game) **abort** means the algorithm is stopped and "**output** b" means the algorithm is stopped and b is returned.

3 PKC and KEM

We recall several standard definitions in Public-Key Cryptography, namely PKE and KEM.

3.1 Public-Key Encryption scheme

Definition 1 (Public-Key Encryption). *A Public-Key Encryption scheme is composed of four algorithms* $\mathsf{setup}, \mathsf{gen}, \mathsf{enc}, \mathsf{dec}$*:*

- $\mathsf{pp} \leftarrow\!\$\, \mathsf{setup}(1^\lambda)$*: The setup algorithm randomly generates the public parameters* pp *according to a security parameter* λ*.*
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\, \mathsf{gen}(\mathsf{pp})$*: The key generation algorithm takes the public parameters as inputs and outputs the public key* pk *and the secret key* sk*.*
- $\mathsf{ct} \leftarrow\!\$\, \mathsf{enc}(\mathsf{pp}, \mathsf{pk}, \mathsf{pt})$*: The encryption algorithm takes as inputs the public parameters* pp*, the public key* pk *and a plaintext* $\mathsf{pt} \in \mathcal{M}$ *and it outputs a ciphertext* ct*.*
- $\mathsf{pt}' \leftarrow \mathsf{dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$*: The decryption procedure takes as inputs the public parameters* pp*, the secret key* sk *and the ciphertext* $\mathsf{ct} \in \mathcal{C}$ *and it outputs a plaintext* $\mathsf{pt}' \in \mathcal{M} \cup \{\perp\}$*.*

The setup*,* gen *and* enc *are probabilistic algorithms that can be made deterministic by adding random coins as inputs. The decryption procedure is deterministic. Finally, for the sake of simplicity, we omit the public parameters in the inputs from now on.*

Correctness. We define the $\delta(q_H)$-correctness in the random oracle model (ROM) as in [14], using the game CORR defined in Figure 2. We say a PKE scheme is $\delta(q_H)$ correct if for any ppt adversary $\mathcal{A}$ making at most q_H adversary to the random oracle H, we have

$$\Pr[\mathrm{CORR}_{\mathsf{PKE}}(\mathcal{A}) \Rightarrow 1] \leq \delta(q_H, \lambda)$$

$\underline{\mathrm{CORR}_{\mathsf{PKE}}(\mathcal{A})}$
$\mathsf{pp} \leftarrow\$\, \mathsf{setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$\, \mathsf{gen}(\mathsf{pp})$
$\mathsf{pt} \leftarrow \mathcal{A}^H(\mathsf{pk}, \mathsf{sk})$
$\mathsf{ct} \leftarrow\$\, \mathsf{enc}(\mathsf{pk}, \mathsf{pt})$
return $1_{\mathsf{dec}(\mathsf{sk},\mathsf{ct}) \neq \mathsf{pt}}$

Fig. 2: Correctness game.

$\underline{\text{IND-}\mathsf{ATK}^b_{\mathsf{PKE}}(\mathcal{A})}$
$\mathsf{pp} \leftarrow\$\, \mathsf{setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$\, \mathsf{gen}(\mathsf{pp})$
define $\mathsf{ct}^* \leftarrow \emptyset$
$\mathsf{pt}_0, \mathsf{pt}_1 \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{ATK1}}}(\mathsf{pk})$
$\mathsf{ct}^* \leftarrow\$\, \mathsf{enc}(\mathsf{pk}, \mathsf{pt}_b)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{ATK2}}}(\mathsf{pk}, \mathsf{ct}^*)$
return b'

$\underline{\textbf{Oracle } \mathcal{O}^{\mathsf{Dec}}(\mathsf{ct})}$
1 : **if** $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
2 : $\mathsf{pt}' \leftarrow \mathsf{dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$
3 : **return** pt'

Fig. 3: Indistinguishability games.

where λ is the security parameter, we omit it from now on for the sake of simplicity. That is, no adversary can find with probability greater than $\delta(q_H)$ a plaintext such that its encryption does not decrypt to the original plaintext. The correctness δ might depend on the number of queries to the RO, thus it is represented as a function of q_H. The correctness in the standard model is the same except δ is fixed.

Definition 2 (IND-CPA/CCA/CCA1). *We consider the game defined in Figure 3, where the oracles given in each game are defined as in the left of Table 1. A PKE scheme* $\mathsf{PKE} = (\mathsf{setup}, \mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ *is* IND-ATK *for* $\mathsf{ATK} \in \{CPA, CCA, CCA1\}$ *if for any ppt adversary* $\mathcal{A}$ *we have*

$$\mathsf{Adv}^{\text{ind-atk}}_{\mathcal{A},\mathsf{PKE}} = \left|\Pr\left[\text{IND-}\mathsf{ATK}^1_{\mathsf{PKE}}(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[\text{IND-}\mathsf{ATK}^0_{\mathsf{PKE}}(\mathcal{A}) \Rightarrow 1\right]\right| = \mathsf{negl}(\lambda)$$

where $\Pr\left[\text{IND-}\mathsf{ATK}^b_{\mathsf{PKE}}(\mathcal{A}) \Rightarrow 1\right]$ *is the probability that* $\mathcal{A}$ *wins the* IND-$\mathsf{ATK}^b_{\mathsf{PKE}}(\mathcal{A})$ *game defined in Figure 3.*

Plaintext and Validity Checking. We also recall three less common security definitions: One-Wayness under Plaintext-Checking Attacks (OW-PCA)/Validity Checking Attacks (OW-VA)/Plaintext and Validity Checking Attacks (OW-PVCA). These notions are useful when proving the security of FO-like transforms, as shown by Hofheinz et al. [14]. All these notions are weaker than IND-CCA but they model the concept of *reaction attacks*, that is when an adversary can observe whether a decryption is successful or not.

Table 1: Oracles for IND and OW games.

ATK	CPA	CCA1	CCA
$\mathcal{O}^{\mathsf{ATK1}}$	$\perp$	$\mathcal{O}^{\mathsf{Dec}}$	$\mathcal{O}^{\mathsf{Dec}}$
$\mathcal{O}^{\mathsf{ATK2}}$	$\perp$	$\perp$	$\mathcal{O}^{\mathsf{Dec}}$

ATK	CPA	PCA	VCA	PVCA
$\mathcal{O}^{\mathsf{ATK}}$	$\perp$	$\mathcal{O}^{\mathsf{PCO}}$	$\mathcal{O}^{\mathsf{VCO}}$	$\mathcal{O}^{\mathsf{PCO}}, \mathcal{O}^{\mathsf{VCO}}$

$\underline{\text{OW-ATK}_{\mathsf{PKE}}(\mathcal{A})}$

$\mathsf{pp} \leftarrow\$\, \mathsf{setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$\, \mathsf{gen}(\mathsf{pp})$
$\mathsf{pt}^* \leftarrow\$\, \mathcal{M}$
$\mathsf{ct}^* \leftarrow \mathsf{enc}(\mathsf{pk}, \mathsf{pt}^*)$
$\mathsf{pt}' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{ATK}}}(\mathsf{pk}, \mathsf{ct}^*)$
return $1_{\mathsf{pt}'=\mathsf{pt}^*}$

Oracle $\mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, \mathsf{ct})$

1 : $\mathsf{pt}' \leftarrow \mathsf{dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$
2 : **return** $1_{\mathsf{pt}'=\mathsf{pt}}$

Oracle $\mathcal{O}^{\mathsf{VCO}}(\mathsf{ct} \neq \mathsf{ct}^*)$

1 : $\mathsf{pt}' \leftarrow \mathsf{dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$
2 : **return** $1_{\mathsf{pt}' \in \mathcal{M}}$

Fig. 4: One-Wayness games.

Definition 3 (One-Wayness and Plaintext/Validity Checking). *Let $\mathcal{M}$ be the message space,* PKE *a PKE scheme and we consider the games defined in Figure 4 with the different oracles as defined on the right in Table 1. Then,* PKE *is OW-*ATK*, for* $\mathsf{ATK} \in \{CPA, PCA, VCA, PVCA\}$*, if for any ppt adversary $\mathcal{A}$ we have*

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

where $\Pr\left[\text{OW-}\mathsf{ATK}_{\mathsf{PKE}}(\mathcal{A}) \Rightarrow 1\right]$ *is the probability that the adversary wins the* OW-ATK *game.*

3.2 Key Encapsulation Mechanism (KEM)

Definition 4 (Key Encapsulation Mechanism). *A KEM is a tuple of four algorithms* setup, gen, encaps, decaps*:*

- $\mathsf{pp} \leftarrow\$\, \mathsf{setup}(1^\lambda)$*: The setup algorithm takes the security parameter λ as input and outputs the public parameters* pp*.*
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$\, \mathsf{gen}(\mathsf{pp})$*: The key generation algorithm takes as inputs the public parameters and it outputs the public key* pk *and the secret key* sk*.*
- $\mathsf{ct}, K \leftarrow\$\, \mathsf{encaps}(\mathsf{pp}, \mathsf{pk})$*: The encapsulation algorithm takes as inputs the public parameters* pp*, the public key* pk *and it outputs a ciphertext* $\mathsf{ct} \in \mathcal{C}$ *and a key* $K \in \mathcal{K}$*.*
- $K' \leftarrow \mathsf{decaps}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$*: The decapsulation procedure takes as inputs the public parameters* pp*, the secret key* sk *and the ciphertext* $\mathsf{ct} \in \mathcal{C}$ *and it outputs a key K. If the KEM allows explicit rejection, the output is a key* $K \in \mathcal{K}$ *or the rejection symbol $\perp$. If the rejection is implicit, the output is always a key* $K \in \mathcal{K}$*.*

$\underline{\text{IND-}\mathsf{ATK}_{\mathsf{KEM}}(\mathcal{A})}$

$\mathsf{pp} \leftarrow\$\, \mathsf{setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$\, \mathsf{gen}(\mathsf{pp})$
$st \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{ATK1}}}(\mathsf{pk})$
$b \leftarrow\$\, \{0,1\}$
$\mathsf{ct}^*, K_0 \leftarrow\$\, \mathsf{encaps}(\mathsf{pk})$
$K_1 \leftarrow\$\, \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{ATK2}}}(st, \mathsf{pk}, \mathsf{ct}^*, K_b)$
return $1_{b'=b}$

$\underline{\textbf{Oracle } \mathcal{O}^{\mathsf{Dec}}(\mathsf{ct})}$

1 : **if** $\mathsf{ct} = \mathsf{ct}^*$ **then return** $\perp$
2 : $K' \leftarrow \mathsf{decaps}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$
3 : **return** K'

Fig. 5: Indistinguishability games.

The setup, gen *and* encaps *are probabilistic algorithms that can be made deterministic by adding random coins as inputs. The decapsulation function is deterministic. For the sake of simplicity, we omit the public parameters in the inputs from now on.*

Definition 5. *We consider the games defined in Figure 5. The oracles the adversary has access to are defined on the left in Table 1 and $\mathcal{K}$ is the key space. A KEM scheme* $\mathsf{KEM} = (\mathsf{setup}, \mathsf{gen}, \mathsf{encaps}, \mathsf{decaps})$ *is* IND-ATK *if for any ppt adversary $\mathcal{A}$ we have*

$$\mathsf{Adv}^{\text{ind-atk}}_{\mathsf{KEM}}(\mathcal{A}) = \left| \Pr\left[\text{IND} - \mathsf{ATK}_{\mathsf{KEM}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda)$$

where $\Pr\left[\text{IND} - \mathsf{ATK}_{\mathsf{KEM}}(\mathcal{A}) \Rightarrow 1\right]$ *is the probability that $\mathcal{A}$ wins the* IND-$\mathsf{ATK}_{\mathsf{KEM}}(\mathcal{A})$ *game defined in Figure 5.*

4 FO-like transforms

Fujisaki and Okamoto introduced one of the first generic IND-CPA to IND-CCA transforms for PKE [9,10] in 1999 and many have followed (e.g. [16,19]). In recent years, transforms have been a hot topic due to their heavy use in PQ proposals and several variants of the FO transform have been proposed and proven secure in the ROM and in the quantum ROM (QROM, i.e. the adversary can make quantum queries to the RO) [21,22,4]. However, we focus here on results proven by Hofheinz et al. [14], which generalize and decompose FO-like transforms in smaller parts. We recall three transforms from this paper which will be useful to construct and understand our combiners.

The first one is the T transform, which takes an OW/IND-CPA PKE $\mathsf{PKE}' = (\mathsf{setup}', \mathsf{gen}', \mathsf{enc}', \mathsf{dec}')$ and outputs an OW-PVCA PKE scheme $\mathsf{PKE} = (\mathsf{setup}, \mathsf{gen}, \mathsf{enc}, \mathsf{dec})$, it is presented in Figure 6, where $G : \{0,1\}^* \mapsto \{0,1\}^\lambda$ is a hash function modelled as a RO. As both setup and generate functions are the same, we do not present them here. Informally, the transform derandomizes the

$\mathsf{enc}(\mathsf{pk}, \mathsf{pt})$	$\mathsf{dec}(\mathsf{sk}, \mathsf{ct})$
$\mathsf{coins} \leftarrow G(\mathsf{pt})$	$\mathsf{pt}' \leftarrow \mathsf{dec}'(\mathsf{sk}, \mathsf{ct})$
return $\mathsf{enc}'(\mathsf{pk}, \mathsf{pt}; \mathsf{coins})$	**if** $\mathsf{pt}' = \perp$ **or** $\mathsf{enc}(\mathsf{pk}, \mathsf{pt}') \neq \mathsf{ct}$
	return $\perp$
	return pt'

Fig. 6: T transform.

underlying scheme by computing the random coins as the hash of the message. Then, the decryption function checks that the ciphertexts are well-formed by re-encrypting the decrypted message.

Before stating the security result of this transform, we need to introduce the notion of γ-spreadness.

Definition 6 (γ-spreadness). *For any public-key* pk *and plaintext* pt*, we define the min-entropy of* $\mathsf{enc}(\mathsf{pk}, \mathsf{pt})$ *as*

$$\gamma(\mathsf{pk}, \mathsf{pt}) = -\log\left(\max_{\mathsf{ct} \in \mathcal{C}} \Pr\left[\mathsf{ct} = \mathsf{enc}(\mathsf{pk}, \mathsf{pt})\right]\right)$$

where the probability is taken over the randomness of enc*, the logarithm is in base 2 and* $\mathcal{C}$ *is the ciphertext domain. Then, we say that a PKE scheme is* γ*-spread if for any public-key* pk *and plaintext* pt*, we have* $\gamma(\mathsf{pk}, \mathsf{pt}) \geq \gamma$*. This implies that* $\Pr[\mathsf{ct} = \mathsf{enc}(\mathsf{pk}, \mathsf{pt})] \leq 2^{-\gamma}$*.*

Then, the following theorem formally states the security of the T transform.

Theorem 1 (OW-CPA $\overset{\text{ROM}}{\Longrightarrow}$ det. OW-PVCA, Theorem 3.1 [14]). *Let* G *be a hash function modelled as a random oracle,* PKE' *a* γ*-spread and* $\delta(q_G)$*-correct PKE scheme, and* PKE *the resulting PKE after applying* T*. Then, for any OW-PVCA adversary* $\mathcal{A}$ *issuing at most* q_G, q_V *queries to* G *and* $\mathcal{O}^{\mathsf{VCO}}$*, respectively, there exists an OW-CPA adversary* $\mathcal{B}$ *s.t.*

$$\mathsf{Adv}^{\mathrm{ow-pvca}}_{\mathsf{PKE}}(\mathcal{A}) \leq q_G \cdot \delta + q_V \cdot 2^{-\gamma} + (q_G + 1) \cdot \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}'}(\mathcal{B})$$

where the running time and the number of queries of $\mathcal{B}$ *are similar to the ones of* $\mathcal{A}$*.*

We present now the two other transforms called $\mathsf{U}^{\perp}$ and $\mathsf{U}^{\not\perp}$. These transforms convert an OW-PVCA (resp. OW-PCA) PKE into an IND-CCA KEM. The transforms are shown in Figure 7 and the setup algorithm is not detailed. The only difference between both transform is that in $\mathsf{U}^{\not\perp}$ the rejection is implicit (i.e. when an error occurs during decryption a random key is returned instead of the error symbol). The security of these constructions is formally stated in the following theorems.

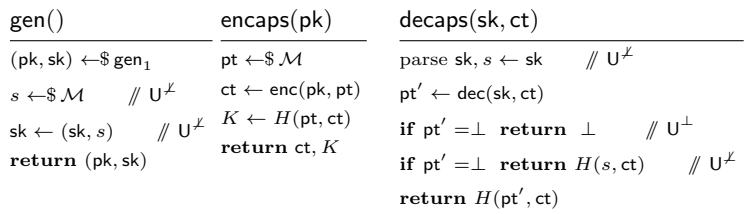


Fig. 7: $\mathsf{U}^{\perp}$ and $\mathsf{U}^{\not\perp}$ transforms from [14].

Theorem 2 (PKE OW-PVCA $\xRightarrow{\text{ROM}}$ KEM IND-CCA, Theorem 3.3 [14]). *Let H be a random oracle, PKE a δ-correct PKE scheme, and KEM the resulting KEM after applying $\mathsf{U}^{\perp}$ on PKE. Then, for any KEM IND-CCA adversary $\mathcal{A}$ issuing at most q_H, q_D queries to H and $\mathcal{O}^{\mathsf{CCA}}$, respectively, there exists an OW-PVCA adversary $\mathcal{B}$ s.t.*

$$\mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{KEM}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{ow-pvca}}_{\mathsf{PKE}}(\mathcal{B})$$

and $\mathcal{B}$ makes at most q_H queries to both checking oracles.

Theorem 3 (PKE OW-PCA $\xRightarrow{\text{ROM}}$ KEM IND-CCA, Theorem 3.4 [14]). *Let H be a random oracle, PKE a δ-correct PKE scheme and KEM the resulting KEM after applying $\mathsf{U}^{\not\perp}$ on PKE. Then, for any KEM IND-CCA adversary $\mathcal{A}$ issuing at most q_H queries to H, there exists an OW-PCA adversary $\mathcal{B}$ s.t.*

$$\mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{KEM}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathcal{B}) + \frac{q_H}{|\mathcal{M}|}$$

where $\mathcal{B}$ makes at most q_H queries to the random oracle and the running time of $\mathcal{B}$ is roughly the same as the one of $\mathcal{A}$.

5 FO-like combiners

We wish to design constructions that take two (or more) IND/OW-CPA schemes instead of one and output an IND-CCA KEM. Compared to black-box combiners, this approach allows for lower-level combiners, which in turn can be more efficient. As more precise examples, we consider KEM combiners proposed by Bindel et al. [3]. These 3 constructions, namely XtM, dualPRF and N are based on special kinds of MAC and PRF. In the XtM combiner, the keys must be split and a tag on the ciphertexts is computed. Similarly, in the dualPRF and N combiners, multiple passes on the keys and ciphertext must be performed to derive the key (see Bindel et al. [3] for more details). All these operations add complexity and/or increase the ciphertext length while being redundant if the underlying KEMs are built using a FO-like transform. Thus, one could hope to remove several superfluous computations and primitives by looking at the actual

$\mathsf{gen}()$

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\$\, \mathsf{gen}_1$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\$\, \mathsf{gen}_2$
$\mathsf{pk} \leftarrow (\mathsf{pk}_1, \mathsf{pk}_2)$
$\mathsf{sk} \leftarrow (\mathsf{sk}_1, \mathsf{sk}_2)$
return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{enc}(\mathsf{pk}, (\mathsf{pt}_1, \mathsf{pt}_2))$

parse $(\mathsf{pk}_1, \mathsf{pk}_2) \leftarrow \mathsf{pk}$
$\mathsf{ct}_1 \leftarrow \mathsf{enc}_1(\mathsf{pk}_1, \mathsf{pt}_1; G(\mathsf{pt}_1))$
$\mathsf{ct}_2 \leftarrow \mathsf{enc}_2(\mathsf{pk}_2, \mathsf{pt}_2; G(\mathsf{pt}_2))$
return $(\mathsf{ct}_1, \mathsf{ct}_2)$

$\mathsf{dec}(\mathsf{sk}, (\mathsf{ct}_1, \mathsf{ct}_2))$

parse $(\mathsf{sk}_1, \mathsf{sk}_2) \leftarrow \mathsf{sk}$
$\mathsf{pt}'_1 \leftarrow \mathsf{dec}_1(\mathsf{sk}_1, \mathsf{ct}_1)$
$\mathsf{pt}'_2 \leftarrow \mathsf{dec}_2(\mathsf{sk}_2, \mathsf{ct}_2)$
if $\mathsf{enc}_1(\mathsf{pk}_1, \mathsf{pt}'_1; G(\mathsf{pt}'_1)) \neq \mathsf{ct}_1$:
  return $\perp$
if $\mathsf{enc}_2(\mathsf{pk}_2, \mathsf{pt}'_2; G(\mathsf{pt}'_2)) \neq \mathsf{ct}_2$:
  return $\perp$
return $(\mathsf{pt}'_1, \mathsf{pt}'_2)$

Fig. 8: $\mathsf{T}_{\|}$ combiner.

implementation of the underlying KEMs. We apply this idea to construct several new combiners, which we call *FO-like combiners*. In addition of not being black-box, these combiners differ from other proposals in the fact that they take several PKEs as inputs and output a KEM.

5.1 $\mathsf{T}_{\|}$ combiner

For our first construction, the idea is to apply twice the T transform of Hofheinz et al. [14] (see Figure 6) to obtain an OW-PCA PKE from two OW-CPA PKEs $\mathsf{PKE}_i = (\mathsf{setup}_i, \mathsf{gen}_i, \mathsf{enc}_i, \mathsf{dec}_i),\ i \in \{1, 2\}$. We call this FO-like combiner $\mathsf{T}_{\|}$ and we present it in Figure 8 (we omit the setup algorithm, which is trivial). Then, one can apply the $\mathsf{U}^{\not\perp}$ transform (see Figure 7) and Theorem 3 to obtain an IND-CCA KEM. The message space $\mathcal{M}$ of the resulting PKE is $\mathcal{M}_1 \times \mathcal{M}_2$ (i.e. the space product of the two message spaces). This construction is actually a useful intermediary step towards a more general OW-CPA to KEM IND-CCA combiner we present in the next section.

The following theorem shows that $\mathsf{T}_{\|}$ is a robust combiner (as long as one of the two underyling PKEs is OW-CPA, the resulting PKE is OW-PCA).

Theorem 4. *Let* PKE *be the PKE resulting from applying* $\mathsf{T}_{\|}$ *on* PKE_1 *and* PKE_2*, which are respectively* δ_1 *and* δ_2 *correct. In addition, let* G *be a hash function modelled as a random oracle. Then, for all ppt OW-PCA adversary* $\mathcal{A}$ *making at most* q_G *queries to* G *and* q_P *queries to the plaintext-checking oracle, there exists adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *such that*

$$\mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathcal{A}) \leq (q_G+q_P)\cdot(\delta_1+\delta_2)+(q_G+1)\cdot\min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathcal{B}_1), \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_2}(\mathcal{B}_2)\}$$

where $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *run in about the same time as* $\mathcal{A}$*.*

Proof. We first show that the trivial PKE combiner C in Figure 9 is a robust OW-PCA combiner. Let $\mathsf{PKE} = \mathsf{C}(\mathsf{PKE}_1, \mathsf{PKE}_2)$ be the PKE resulting from applying C on two PKEs PKE_1 and PKE_2. We show w.l.o.g. that the OW-PCA security of PKE reduces to the OW-PCA security of PKE_1. The OW-PCA game against

$\underline{\mathsf{gen}()}$
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\$\, \mathsf{gen}_1$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\$\, \mathsf{gen}_2$
$\mathsf{pk} \leftarrow (\mathsf{pk}_1, \mathsf{pk}_2)$
$\mathsf{sk} \leftarrow (\mathsf{sk}_1, \mathsf{sk}_2)$
return $(\mathsf{pk}, \mathsf{sk})$

$\underline{\mathsf{enc}(\mathsf{pk}, (\mathsf{pt}_1, \mathsf{pt}_2))}$
parse $(\mathsf{pk}_1, \mathsf{pk}_2) \leftarrow \mathsf{pk}$
$\mathsf{ct}_1 \leftarrow \mathsf{enc}_1(\mathsf{pk}_1, \mathsf{pt}_1)$
$\mathsf{ct}_2 \leftarrow \mathsf{enc}_2(\mathsf{pk}_2, \mathsf{pt}_2)$
return $(\mathsf{ct}_1, \mathsf{ct}_2)$

$\underline{\mathsf{dec}(\mathsf{sk}, (\mathsf{ct}_1, \mathsf{ct}_2))}$
parse $(\mathsf{sk}_1, \mathsf{sk}_2) \leftarrow \mathsf{sk}$
$\mathsf{pt}'_1 \leftarrow \mathsf{dec}_1(\mathsf{sk}_1, \mathsf{ct}_1)$
$\mathsf{pt}'_2 \leftarrow \mathsf{dec}_2(\mathsf{sk}_2, \mathsf{ct}_2)$
return $(\mathsf{pt}'_1, \mathsf{pt}'_2)$

Fig. 9: Trivial PKE combiner C.

$\underline{\text{OW-PCA}_{\mathsf{PKE}}(\mathcal{A})}$
$\mathsf{pp} \leftarrow\$\, \mathsf{setup}(1^\lambda)$
$((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{sk}_1, \mathsf{sk}_2)) \leftarrow\$\, \mathsf{gen}(\mathsf{pp})$
$(\mathsf{pt}^*_1, \mathsf{pt}^*_2) \leftarrow\$\, \mathcal{M}_1 \times \mathcal{M}_2$
$\mathsf{ct}^* \leftarrow \mathsf{enc}(\mathsf{pk}, (\mathsf{pt}^*_1, \mathsf{pt}^*_2))$
$(\mathsf{pt}'_1, \mathsf{pt}'_2) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{PCO}}}(\mathsf{pk}, \mathsf{ct}^*)$
return $1_{(\mathsf{pt}'_1, \mathsf{pt}'_2) = (\mathsf{pt}^*_1, \mathsf{pt}^*_2)}$

$\underline{\textbf{Oracle } \mathcal{O}^{\mathsf{PCO}}((\mathsf{pt}_1, \mathsf{pt}_2), (\mathsf{ct}_1, \mathsf{ct}_2))}$
$\mathsf{pt}'_1 \leftarrow \mathsf{dec}_1(\mathsf{pp}, \mathsf{sk}_1, \mathsf{ct}_1)$
$\mathsf{pt}'_2 \leftarrow \mathsf{dec}_2(\mathsf{pp}, \mathsf{sk}_2, \mathsf{ct}_2)$
return $1_{(\mathsf{pt}_1, \mathsf{pt}_2) = (\mathsf{pt}'_1, \mathsf{pt}'_2)}$

Fig. 10: OW-PCA game against PKE for the proof of Theorem 4.

PKE is presented in Figure 10. One can see that the plaintext-checking oracle can easily be simulated by an adversary having access to a plaintext-checking oracle for PKE_1 and holding the secret key sk_2. Thus, we can easily build an adversary $\mathcal{B}$ against the OW-PCA security of PKE_1. This adversary generates itself $\mathsf{pk}_2, \mathsf{sk}_2, \mathsf{ct}^*_2$, runs $\mathcal{A}$ and simulates perfectly the PCO oracle with its own oracle and sk_2. When $\mathcal{A}$ returns $(\mathsf{pt}'_1, \mathsf{pt}'_2)$, $\mathcal{B}$ returns pt'_1 and wins with at least the same advantage as $\mathcal{A}$. Hence,

$$\mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathcal{A}) \leq \min\{\mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}_1}(\mathcal{B}_1), \mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}_2}(\mathcal{B}_2)\} \ .$$

To conclude, one can just observe that $\mathsf{T}_{\|}(\mathsf{PKE}_1, \mathsf{PKE}_2) = \mathsf{C}(\mathsf{T}(\mathsf{PKE}_1), \mathsf{T}(\mathsf{PKE}_2))$, where T is the OW-CPA to OW-PCA transform from Hofheinz et al. [14]. Therefore, Theorem 3 holds the proof. □

Corollary 1. *Let* KEM *be the KEM resulting from applying* $\mathsf{U}^{\not\perp} \circ \mathsf{T}_{\|}$ *onto two PKE schemes* PKE_1 *and* PKE_2*, which are* δ_1*-correct and* δ_2*-correct, respectively. Then, for any IND-CCA adversary* $\mathcal{A}$ *making at most* q_H *and* q_G *queries to the ROs* H *and* G*, respectively, and* q_D *queries to the decapsulation oracle, there exists OW-CPA adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *such that*

$$\mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{KEM}}(\mathcal{A}) \leq \frac{q_H}{|\mathcal{M}_1||\mathcal{M}_2|} + (q_G + q_D) \cdot (\delta_1 + \delta_2) + (q_G + 1) \cdot \min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathcal{B}_1), \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_2}(\mathcal{B}_2)\}$$

where $\mathcal{M}_i$ *is the message space of* PKE_i *and* $\mathcal{B}_i$ *runs in about the same time as* $\mathcal{A}$*.*

Proof. This is a simple consequence of Theorems 3 and 4. □

$\mathcal{B}_1^{\mathcal{A},\mathcal{O}^{\mathsf{PCO}_1}}(\mathsf{pk}_1, \mathsf{ct}_1^*)$

$\mathsf{pp}_2 \leftarrow\!\$\, \mathsf{setup}_2(1^\lambda)$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\$\, \mathsf{gen}_2(\mathsf{pp}_2)$
$\mathsf{pt}_2^* \leftarrow\!\$\, \mathcal{M}_2$
$\mathsf{ct}_2^* \leftarrow \mathsf{enc}_2(\mathsf{pk}, \mathsf{pt}_2^*)$
$(\mathsf{pt}_1', \mathsf{pt}_2') \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{PCO}}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*))$
return pt_1'

Oracle $\mathcal{O}^{\mathsf{PCO}}((\mathsf{pt}_1, \mathsf{pt}_2), (\mathsf{ct}_1, \mathsf{ct}_2))$

$r \leftarrow \mathcal{O}^{\mathsf{PCO}_1}(\mathsf{pt}_1, \mathsf{ct}_1)$
$\mathsf{pt}_2' \leftarrow \mathsf{dec}_2(\mathsf{pp}, \mathsf{sk}_2, \mathsf{ct}_2)$
return $1_{r=1 \wedge \mathsf{pt}_2 = \mathsf{pt}_2'}$

Fig. 11: OW-CPA adversary for the proof of Theorem 4.

Discussion. Let $\mathsf{UT}_{\|}^{\not\perp}$ be the combiner resulting from composing $\mathsf{U}^{\not\perp}$ and $\mathsf{T}_{\|}$. One could wonder whether combining two PKEs in a trivial way (i.e. encrypting $\mathsf{pt}_1, \mathsf{pt}_2$ as $(\mathsf{enc}_1(\mathsf{pt}_1), \mathsf{enc}_2(\mathsf{pt}_2))$ and decrypting both ciphertexts independently) and then applying a FO-like transform would hold a robust IND-CCA KEM. In fact, this would give a combiner similar to $\mathsf{UT}_{\|}^{\not\perp}$, except the random coins would be split into two parts $(G(\mathsf{pt}_1, \mathsf{pt}_2))_{\lambda_1}$ and $(G(\mathsf{pt}_1, \mathsf{pt}_2))_{\lambda_2}$ for each encryption procedure, where λ_i is the number of coins needed by the encryption of PKE_i. As G is a RO, both shares would be independent and the result would be similar to the coins $G(\mathsf{pt}_i)$ in our $\mathsf{UT}_{\|}^{\not\perp}$ transform. We preferred the latter solution as it is possible to compute the coins in parallel and we think it makes the separation between both sets of coins clear. One could also wonder whether setting the coins to $G(\mathsf{pt}_1, \mathsf{pt}_2)$ would work. This, in turn, creates a correlation between both ciphertexts, which cannot be dealt with in the security proof.

The choice of computing the deterministic coins for ct_i based on σ_i only (instead of σ_1 and σ_2) has positive and negative impacts on the resulting scheme.
Efficiency: Both ciphertexts are totally independent and can be computed in parallel. In turn, this would allow to keep a key share static for a period of time while varying the other one. This could improve consequently the efficiency of hybrid schemes in protocols.
Malleability and misuse resistance: The ciphertexts of the resulting KEM $\mathsf{ct}^* = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ are somewhat malleable. Indeed, it is easy to modify a ciphertext into another one s.t. the decryption is valid. For instance, $\mathsf{ct}' = (\mathsf{ct}_1^*, \mathsf{ct}_2')$, for a valid ct_2', will decapsulate properly to the key $H(\sigma_1^*, \sigma_2', \mathsf{ct}')$. This has no consequence in the ROM as the RO hides perfectly σ_1^*, but this does not necessarily seem a desired property. In particular, due to this malleability effect, the key must be derived as $H(\sigma_1, \sigma_2, \ldots)$ and other KDFs that would seem intuitive lead to security flaw. For instance, computing the key as $H(\sigma_1) \oplus H(\sigma_2)$ in the transform makes a trivial IND-CCA attack possible.

Efficiency. One can see that the main cost of the combiner is to compute two hash values on the two plaintexts (i.e. seeds) and then a hash on the two plaintexts and ciphertexts. This already seems slightly more efficient than the XtM (XOR-then-MAC) combiner proposed by Bindel et al. [3]. Indeed, XtM doubles

$\mathsf{gen}()$

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\$\, \mathsf{gen}_1$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\$\, \mathsf{gen}_2$
$\mathsf{pk} \leftarrow (\mathsf{pk}_1, \mathsf{pk}_2)$
$\mathsf{sk} \leftarrow (\mathsf{sk}_1, \mathsf{sk}_2)$
return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{encaps}(\mathsf{pk})$

parse $(\mathsf{pk}_1, \mathsf{pk}_2) \leftarrow \mathsf{pk}$
$(\sigma_1, \sigma_2) \leftarrow\$\, \mathcal{M}_1 \times \mathcal{M}_2$
$\mathsf{ct}_1 \leftarrow \mathsf{enc}_1(\mathsf{pk}_1, \sigma_1; G(1, \sigma_1, \sigma_2))$
$\mathsf{ct}_2 \leftarrow \mathsf{enc}_2(\mathsf{pk}_2, \sigma_2; G(2, \sigma_1, \sigma_2))$
$K \leftarrow H(\sigma_1 \oplus \sigma_2)$
return $(\mathsf{ct}_1, \mathsf{ct}_2), K$

$\mathsf{decaps}(\mathsf{sk}, (\mathsf{ct}_1, \mathsf{ct}_2))$

parse $(\mathsf{sk}_1, \mathsf{sk}_2) \leftarrow \mathsf{sk}$
$\sigma'_1 \leftarrow \mathsf{dec}_1(\mathsf{sk}_1, \mathsf{ct}_1)$
$\sigma'_2 \leftarrow \mathsf{dec}_2(\mathsf{sk}_2, \mathsf{ct}_2)$
if $\mathsf{enc}_1(\mathsf{pk}_1, \sigma'_1; G(1, \sigma'_1, \sigma'_2)) \neq \mathsf{ct}_1$:
  return $\perp$
if $\mathsf{enc}_2(\mathsf{pk}_2, \sigma'_2; G(2, \sigma'_1, \sigma'_2)) \neq \mathsf{ct}_2$:
  return $\perp$
return $H(\sigma'_1 \oplus \sigma'_2)$

Fig. 12: $\mathsf{UT}_{\|}$ combiner.

the size of the keys returned by the underlying KEMs, split them and compute a MAC on the ciphertexts using two halves of the keys.

Now, as the ciphertexts in post-quantum cryptography can be large (usually a few kilobytes), computing a hash on two ciphertexts can be an expensive operation. Our combiner presented in the next section fixes this drawback.

5.2 $\mathsf{UT}_{\|}$

We now propose an FO-like combiner similar to $\mathsf{T}_{\|}$ that combines two OW-CPA PKEs into an IND-CCA KEM. In a way, we skip the $\mathsf{U}^{\not\perp}$ transform to get directly a KEM. The idea is to encrypt two seeds (i.e. plaintexts) σ_1, σ_2 using the PKE resulting from $\mathsf{T}_{\|}$ and then compute the key as $H(\sigma_1 \oplus \sigma_2)$. However, in order to avoid the malleability issue described in the previous section, the deterministic coins are computed as $G(i, \sigma_1, \sigma_2)$. This links both ciphertexts together and makes tampering one of the two more difficult. Note that in order to compute the XOR, we assume that the seeds σ_i are binary strings or that there exists an efficient and unique encoding of these objects as binary strings. Alternatively, one can take the hash of a plaintext to get a binary seed. All these options are compatible with our combiner and the choice of an approach depends on the underyling PKEs. We present the combiner in Figure 12.

Now, the following theorem formally states the security of the $\mathsf{UT}_{\|}$ combiner.

Theorem 5. *Let* KEM *be the KEM resulting from applying* $\mathsf{UT}_{\|}$ *on* PKE_1 *and* PKE_2*, which are respectively* δ_1 *and* δ_2 *correct, and* γ_1 *and* γ_2*-spread. In addition, let* G *and* H *be hash functions modelled as a random oracle. Then, for all ppt IND-CCA adversary* $\mathcal{A}$ *making at most* q_G, q_H *and* q_D *queries to* G*,*H *and* $\mathcal{O}^{\mathsf{Dec}}$*, respectively, there exists adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{KEM}}(\mathcal{A}) \leq & (q_D + q_G + 1) \cdot (\delta_1 + \delta_2) + q_D \cdot (2^{-\gamma_1} + 2^{-\gamma_2}) \\ & + (q_G + q_H) \cdot \min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathcal{B}_1), \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_2}(\mathcal{B}_2)\}\end{aligned}$$

where $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *run in about the same time as* $\mathcal{A}$ *and make the same number of queries.*

Proof. We first show that if a valid ciphertext is submitted to the decapsulation oracle, then the corresponding plaintexts have been queried to G, and thus one can simulate the decapsulation oracle without the secret key. Then, one can show that the deterministic coins used to compute the challenge ciphertexts look perfectly random until the adversary queries the challenge plaintexts to G. Finally, by the same property of the RO, the challenge key looks perfectly uniform unless the adversary queries $\sigma_1 \oplus \sigma_2$ to H. We proceed by game hopping, the sequence of games is presented in Figure 13. Note that for the sake of simplicity, we consider the IND-CCA game in which the adversary receives the challenge straight away (i.e. no two phases game).

Game Γ^0: This is the original KEM IND-CCA game for the KEM obtained by applying the $\mathsf{UT}_{\|}$ combiner on two PKEs.

Game Γ^1: In this game, we enforce the correctness of the challenge ciphertexts and the ciphertexts that can be computed by the adversary using the RO G. In particular, we abort if the challenge ciphertexts break the correctness property or if any σ_i in a query m to G is of the form (i, σ_1, σ_2) and is s.t. $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; G(m))$ breaks the correctness property. Now, both challenge queries to G made by the game (i.e. $(i, \sigma_1^*, \sigma_2^*)$) are fresh, hence by the property of the RO and the δ_i correctness of the underlying schemes PKE_i, the probability there is a correctness error is at most $\delta_1 + \delta_2$. Then, throughout the game, at most $2q_D$ queries is made to G by the game in the decapsulation oracle (q_D of the form $(1, \sigma_1, \sigma_2)$ and q_D of the form $(2, \sigma_1, \sigma_2)$) and q_G by the adversary. Hence, in the worst case all these queries are fresh and the probability there is a correctness error is upper bounded by $(q_D + q_G + 1) \cdot (\delta_1 + \delta_2)$. Hence, we have

$$|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]| \leq (q_D + q_G + 1) \cdot (\delta_1 + \delta_2) \ .$$

Game Γ^2: We modify the previous game as follows in the decapsulation oracle. We check whether there exist both $((1, \sigma_1, \sigma_2), g_1)$ and $((2, \sigma_1, \sigma_2), g_2)$ in $\mathcal{L}_{\mathcal{A}}$ s.t. $\mathsf{enc}_i(\sigma_i; g_i) = \mathsf{ct}_i$. If this is the case, (let's call this event found) we return $H(\sigma_1 \oplus \sigma_2)$, otherwise we return the key K output by the decapsulation function. Now, if found occur, we return the same key as in game Γ^1. Indeed, by the perfect correctness of the tuples in $\mathcal{L}_{\mathcal{A}}$ enforced in game Γ^1, if we find (σ_1, σ_2) s.t. $\mathsf{enc}_i(\sigma_i; G(i, \sigma_1, \sigma_2)) = \mathsf{ct}_i$, then $\mathsf{dec}_i(\mathsf{sk}_i, \mathsf{ct}_i) = \sigma_i$. Hence, we have, $\Pr[\Gamma^1 \Rightarrow 1] = \Pr[\Gamma^2 \Rightarrow 1]$.

Game Γ^3: We modify the previous game as follows. In the decapsulation oracle, we simply return $\perp$ if found does not occur. Hence, game Γ^2 and Γ^3 differ iff the decapsulation oracle successfully decrypts ct but the adversary did not query to G neither $(1, \sigma_1', \sigma_2')$ or $(2, \sigma_1', \sigma_2')$, where $(\mathsf{enc}_1(\sigma_1'), \mathsf{enc}_2(\sigma_2')) = \mathsf{ct}$ (i.e. at least one tuple is not in $\mathcal{L}_{\mathcal{A}}$). Now, by the perfect correctness of tuples in $\mathcal{L}_{\mathcal{A}}$, this event is equivalent to the decapsulation oracle successfully (i.e. the re-encryption checks pass) recovering the seeds σ_1, σ_2 but either $(1, \sigma_1, \sigma_2)$ or $(2, \sigma_1, \sigma_2)$ or both were not queried to G by the adversary. Let fail be this event and we prove the following lemma.

Lemma 1.

$$\Pr[\mathsf{fail}] \leq q_D \cdot (2^{-\gamma_1} + 2^{-\gamma_2}) \ .$$

Proof. Let fail_k be the event that fail happens at the k-th decapsulation query and $p_k = \Pr[\mathsf{fail}_k]$. By an union bound, it is clear that

$$\Pr[\mathsf{fail}] \leq \sum_{k=1}^{q_D} p_k \ .$$

Now, let's consider an algorithm $\mathcal{B}_k$ as defined in Figure 14. This adversary simulates perfectly the view of the adversary in game Γ^3 until the k-th query. In particular, for each decapsulation query $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$, it checks whether there exists both $((1, \sigma_1, \sigma_2), g_1)$ and $((2, \sigma_1, \sigma_2), g_2)$ in $\mathcal{L}_G$ s.t. $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g_i)$ for $i \in [2]$. We call this condition cond and if it is fulfilled $\mathcal{B}_k$ outputs $H(\sigma_1 \oplus \sigma_2)$, otherwise it outputs $\perp$.

In the k-th decapsulation query, if cond is fulfilled it aborts. Otherwise, it sets i s.t. there is no $((i, \sigma_1, \sigma_2), g_i) \in \mathcal{L}_G$ s.t. $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g_i) = \mathsf{ct}_i$. Note that such an i will be found because cond was not fulfilled. Also, this condition might be fulfilled for both $i = 1$ and $i = 2$. If it the case, the algorithm sequentially performs the remaining of the instructions for both $i = 1$ and $i = 2$. Next, it decrypts ct_1 and ct_2 to both σ_1' and σ_2'. By the definition of i and the perfect correctness of the values σ_i in $\mathcal{L}_G$, we have that $(i, \sigma_1', \sigma_2') \notin \mathcal{L}_G$. In addition, by the perfect corectness of the challenge ciphertexts we have $(\sigma_1', \sigma_2') \neq (\sigma_1^*, \sigma_2^*)$. Finally, $\mathcal{B}_k$ queries $g_i' \leftarrow G(i, \sigma_1', \sigma_2')$ and outputs 1 iff $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g_i') = \mathsf{ct}_i$. Now, as $g_i' = G(i, \sigma_1', \sigma_2')$ was never queried to G, it is sampled uniformly at random and thus $\Pr[\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i'; g_i') = \mathsf{ct}_i] \leq 2^{-\gamma_i}$ by the γ_i-spreadness of PKE_i. In the worst case, the check is performed for both $i = 1$ and $i = 2$ and thus $\Pr[\mathcal{B}_k(\mathcal{A}) \Rightarrow 1] \leq 2^{-\gamma_1} + 2^{-\gamma_2}$. Now, we simply observe that if fail_k occurs, then $\mathcal{B}_k$ perfectly simulates the decapsulation oracle in Γ^2 and Γ^3 in the first $k-1$ queries and it will output 1 by the definition of fail_k. Thus,

$$p_k \leq \Pr[\mathcal{B}_k(\mathcal{A}) \Rightarrow 1] \leq 2^{-\gamma_1} + 2^{-\gamma_2} \ .$$

Taking the union bound on the p_k holds the result. □

By the previous lemma, we have

$$|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^3 \Rightarrow 1]| \leq q_D \cdot (2^{-\gamma_1} + 2^{-\gamma_2}) \ .$$

Game Γ^4: First note that the decapsulation oracle does not use the secret key anymore. Then, in Γ^4, we raise a flag chal^1 and abort if the adversary queries $(i, \sigma_1^*, \sigma_2^*)$. In addition, we raise a flag chal^2 and abort if the adversary queries $\sigma_1^* \oplus \sigma_2^*$ to H. Now, if chal^1 or chal^2 happens, one can break the OW-CPA property of both PKEs. We give the reduction $\mathcal{B}_1$ that breaks the one-wayness of PKE_1 in Figure 15. More precisely, as long as $\mathsf{chal}^1 \cup \mathsf{chal}^2$ does not happen, the adversary cannot distinguish a game where the coins used to compute the challenge ciphertexts are deterministic from a game where the coins are taken

at random. In addition, it cannot distinguish a game where K is random from a game where $K = H(\sigma_1^* \oplus \sigma_2^*)$. Therefore, the probability that $\mathsf{chal}^1 \cup \mathsf{chal}^2$ happens is the same in Γ^4 and in the OW-CPA game played by $\mathcal{B}_1$. Now, if $\mathsf{chal}^1 \cup \mathsf{chal}^2$ happens in a game where the challenge coins and the key are random, one can break the one-wayness of the underlying scheme. Thus, we have

$$|\Pr[\Gamma^3 \Rightarrow 1] - \Pr[\Gamma^4 \Rightarrow 1]| \leq (q_G + q_H) \cdot \min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathcal{B}_1), \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_2}(\mathcal{B}_2)\} \ .$$

Note that we have a factor $q_G + q_H$ because in the reduction we cannot check which query m contains the challenge seeds σ_i^* (if we picked a query to G) or $\sigma_1^* \oplus \sigma_2^*$ (if we picked a query to H). Indeed, in the OW-CPA game, the challenge coins are taken at random and are unknown to the adversary. More details are given in the proof of Theorem 4.

Game Γ^5: Finally, in this last game we replace the challenge key K_0 by a random one. As K_0 and K_1 have the same distribution now, we have $\Pr[\Gamma^5 \Rightarrow 1] = \frac{1}{2}$. In addition, since the adversary cannot query $\sigma_1^* \oplus \sigma_2^*$ anymore, it cannot distinguish between a real key and a random key by the property of the RO H. Hence, we have $|\Pr[\Gamma^4 \Rightarrow 1] - \Pr[\Gamma^5 \Rightarrow 1]| = 0$. Collecting the probabilities and folding the OW-CPA adversaries into one hold the result. □

Generalisation to n PKEs. While the $\mathsf{UT}_{\|}$ combiner presented in Figure 12 takes two PKEs as input, it is straightforward to generalise it to n PKEs. Each of the n ciphertexts will simply be computed as $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; G(i, \sigma_1, \ldots, \sigma_n))$ and the key as $H(\oplus_i^n \sigma_i)$. Then, the security of such a combiner (we call it $\mathsf{UT}^n_{\|}$) can be stated in the following Theorem, which is a generalization of Theorem 5.

Theorem 6. *Let* KEM *be the KEM resulting from applying* $\mathsf{UT}^n_{\|}$ *on* $\mathsf{PKE}_1, \ldots, \mathsf{PKE}_n$*, which are respectively* $\delta_1, \ldots, \delta_n$ *correct, and* $\gamma_1, \ldots, \gamma_n$*-spread. In addition, let* G *and* H *be hash functions modelled as a random oracle. Then, for all ppt IND-CCA adversary* $\mathcal{A}$ *making at most* q_G, q_H *and* q_D *queries to* G*,*H *and* $\mathcal{O}^{\mathsf{Dec}}$*, respectively, there exists adversaries* $\mathcal{B}_1, \ldots, \mathcal{B}_n$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{KEM}}(\mathcal{A}) \leq & (q_D + q_G + 1) \cdot \sum_{i=1}^{n} \delta_i + q_D \cdot \sum_{i=1}^{n} 2^{-\gamma_i} \\ & + (q_G + q_H) \cdot \min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathcal{B}_1), \ldots, \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_n}(\mathcal{B}_n)\}\end{aligned}$$

where $\mathcal{B}_1, \ldots, \mathcal{B}_n$ *run in about the same time as* $\mathcal{A}$ *and make the same number of queries.*

Proof idea. The proof is exactly the same as the one for the security of $\mathsf{UT}_{\|}$ with two PKEs except we consider n schemes. In particular, the probability of having a correctness or spreadness error in some query is upper bounded by $\sum_{i=1}^n \delta_i$ and $\sum_{i=1}^n 2^{-\gamma_i}$, respectively. Also, the reductions $\mathcal{B}_i$ from the OW-CPA of the PKEs still work the same, as an adversary $\mathcal{B}_i$ picks all σ_j^* s.t. $j \neq i$. That

$\Gamma^i(\mathcal{A})$

$\mathsf{pp} \leftarrow\$\, \mathsf{setup}(1^\lambda);\ (\mathsf{pk}, \mathsf{sk}) \leftarrow\$\, \mathsf{gen}(\mathsf{pp})$
$(\sigma_1^*, \sigma_2^*) \leftarrow\$\, \mathcal{M}_1 \times \mathcal{M}_2$
$\mathsf{coins}_1 \leftarrow G(1, \sigma_1^*, \sigma_2^*); \mathsf{coins}_2 \leftarrow G(2, \sigma_1^*, \sigma_2^*)$ // Γ^0-Γ^4
$\mathsf{coins}_1, \mathsf{coins}_2 \leftarrow\$\, \mathcal{R}^2$ // Γ^5
$\mathsf{ct}_1^* \leftarrow \mathsf{enc}_1(\mathsf{pk}_1, \sigma_1^*; \mathsf{coins}_1)$
$\mathsf{ct}_2^* \leftarrow \mathsf{enc}_2(\mathsf{pk}_2, \sigma_2^*; \mathsf{coins}_2)$
if $\exists i \in [2]$ s.t. $\mathsf{dec}_i(\mathsf{sk}_i, \mathsf{ct}_i^*) \neq \sigma_i^*$: **abort** // Γ^1-Γ^5
$b \leftarrow\$\, \{0,1\}$
$K_0 \leftarrow H(\sigma_1^* \oplus \sigma_2^*)$ // Γ^0-Γ^4
$K_0 \leftarrow\$\, \mathcal{K}$ // Γ^5
$K_1 \leftarrow\$\, \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Dec}}}(\mathsf{pk}, (\mathsf{ct}_1^*, \mathsf{ct}_2^*), K_b)$
return $1_{b'=b}$

Oracle $\mathcal{O}^{\mathsf{Dec}}(\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2))$

$\mathsf{flag} \leftarrow$ **false**
if $\mathsf{ct} = \mathsf{ct}^*$ **then return** $\perp$
if $\exists((1, \sigma_1, \sigma_2), g_1) \in \mathcal{L}_\mathcal{A}$ s.t. $\mathsf{enc}_1(\mathsf{pk}_1, \sigma_1; g_1) = \mathsf{ct}_1$
  and $\exists((2, \sigma_1, \sigma_2), g_2) \in \mathcal{L}_\mathcal{A}$
    s.t. $\mathsf{enc}_2(\mathsf{pk}_2, \sigma_2; g_2) = \mathsf{ct}_2$: // Γ^2-Γ^5
  return $H(\sigma_1 \oplus \sigma_2)$ // Γ^2-Γ^5
return $\perp$ // Γ^3-Γ^5
$K' \leftarrow \mathsf{decaps}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$ // Γ^0-Γ^2
return K' // Γ^0-Γ^2

$H(m)$

if $\exists h$ s.t. $(m, h) \in \mathcal{L}_H$:
  return h
if $m = (\sigma_1^* \oplus \sigma_2^*)$: // Γ^4-Γ^5
  $\mathsf{chal}^2 \leftarrow$ **true** // Γ^4-Γ^5
  abort // Γ^4-Γ^5
$h \leftarrow\$\, \{0,1\}^n$
$\mathcal{L}_H \leftarrow \mathcal{L}_H \cup \{(m, h)\}$
return h

$G(m)$

if $\exists g$ s.t. $(m, g) \in \mathcal{L}_G$:
  return g
if $m = (1, \sigma_1^*, \sigma_2^*)$ **or** // Γ^4-Γ^5
    $m = (2, \sigma_1^*, \sigma_2^*)$: // Γ^4-Γ^5
  $\mathsf{chal}^1 \leftarrow$ **true** // Γ^4-Γ^5
  abort // Γ^4-Γ^5
$g \leftarrow\$\, \{0,1\}^n$
$\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m, g)\}$
if parse $m = (i, \sigma_1, \sigma_2)$ succeeds : // Γ^1-Γ^6
  if $\mathsf{dec}_i(\mathsf{sk}_i, \mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g)) \neq \sigma_i$: // Γ^1-Γ^5
    abort // Γ^1-Γ^5
if m queried by $\mathcal{A}$ // Γ^1-Γ^5
  $\mathcal{L}_\mathcal{A} \leftarrow \mathcal{L}_\mathcal{A} \cup \{(m, g)\}$ // Γ^1-Γ^5
return g

Fig. 13: Sequence of games for the proof of Theorem 5.

is, if $(i, \sigma_1^*, \ldots, \sigma_n^*)$ is queried, $\mathcal{B}_i$ can recover σ_i^*, otherwise we can replace the deterministic coins by random ones. Similarly, if $\sigma^* = \oplus_j^n \sigma_j^*$ is queried by the adversary to H, $\mathcal{B}_i$ can recover σ_i^* by computing $\sigma^* \oplus_{j \neq i} \sigma_j^*$. □

Security in the Quantum Random Oracle Model (QROM). As the original FO-transform, our combiner could be made secure in the QROM by adding a hash in the ciphertext, this technique is often called *plaintext confirmation*. For simplicity, here we show that our $\mathsf{T}_\|$ transform generalized to n PKEs is secure in the QROM (it outputs an OW-PCA scheme). We call this transform $\mathsf{T}_\|^n$ and it is detailed in Figure 20 in Appendix A. Then, it suffices to combine the QROM secure $\mathsf{QU}_m^\perp$ transform from Hofheinz et al. [14] to get an IND-CCA secure KEM in the QROM. One can show the following theorem.

Theorem 7. *Let* PKE *be the PKE resulting from applying* $\mathsf{T}_\|^n$ *on* $\mathsf{PKE}_1, \ldots, \mathsf{PKE}_n$*, which are respectively* $\delta_1, \ldots, \delta_n$*-correct. In addition, let*

$\mathcal{B}_k(\mathcal{A})$

$\mathsf{pp} \leftarrow\$\, \mathsf{setup}(1^\lambda);\ (\mathsf{pk}, \mathsf{sk}) \leftarrow\$\, \mathsf{gen}(\mathsf{pp})$
$(\sigma_1^*, \sigma_2^*) \leftarrow\$\, \mathcal{M}_1 \times \mathcal{M}_2$
$\mathsf{coins}_1 \leftarrow G(1, \sigma_1^*, \sigma_2^*); \mathsf{coins}_2 \leftarrow G(2, \sigma_1^*, \sigma_2^*)$
$\mathsf{ct}_1^* \leftarrow \mathsf{enc}_1(\mathsf{pk}_1, \sigma_1^*; \mathsf{coins}_1)$
$\mathsf{ct}_2^* \leftarrow \mathsf{enc}_2(\mathsf{pk}_2, \sigma_2^*; \mathsf{coins}_2)$
if $\exists i \in [2]$ s.t. $\mathsf{dec}_i(\mathsf{sk}_i, \mathsf{ct}_i^*) \neq \sigma_i^*$: **abort**
$b \leftarrow\$\, \{0,1\}$
$K_0 \leftarrow H(\sigma_1^* \oplus \sigma_2^*)$
$K_1 \leftarrow\$\, \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Dec}}}(\mathsf{pk}, (\mathsf{ct}_1^*, \mathsf{ct}_2^*), K_b)$
return $1_{b'=b}$

Oracle $\mathcal{O}^{\mathsf{Dec}}(\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2))$

if $\mathsf{ct} = \mathsf{ct}^*$ **then return** $\perp$
if $\exists((1, \sigma_1, \sigma_2), g_1) \in \mathcal{L}_G$ s.t. $\mathsf{enc}_1(\mathsf{pk}_1, \sigma_1; g_1) = \mathsf{ct}_1$
  and $\exists((2, \sigma_1, \sigma_2), g_2) \in \mathcal{L}_G$
    s.t. $\mathsf{enc}_2(\mathsf{pk}_2, \sigma_2; g_2) = \mathsf{ct}_2$:
  if k-th query : **abort**
  return $H(\sigma_1 \oplus \sigma_2)$
if k-th query :
  $(\sigma_1', \sigma_2') \leftarrow (\mathsf{dec}_1(\mathsf{sk}_1, \mathsf{ct}_1), \mathsf{dec}_2(\mathsf{sk}_2, \mathsf{ct}_2))$
  for i s.t. $\nexists((i, \sigma_1, \sigma_2), g_i) \in \mathcal{L}_G$ s.t. $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g_i) = \mathsf{ct}_i$
    $g_i' \leftarrow G(i, \sigma_1', \sigma_2')$
    if $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_1'; g_i') = \mathsf{ct}_i$: **return** 1
  abort
return $\perp$

$G(m)$

if $\exists g$ s.t. $(m, g) \in \mathcal{L}_G$:
  return g
$g \leftarrow\$\, \{0,1\}^n$
$\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m, g)\}$
if parse $m = (i, \sigma_1, \sigma_2)$ succeeds :
  if $\mathsf{dec}_i(\mathsf{sk}_i, \mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g)) \neq \sigma_i$:
    abort
return g

Fig. 14: Adversary $\mathcal{B}_k$ for the proof of Lemma 1.

G_i be hash functions modelled as (independent) quantum random oracles. Then, for all quantum OW-PCA adversary $\mathcal{A}$ making at most q_G queries to all oracles G_i and q_P queries to the plaintext-checking oracle, there exists adversaries $\mathcal{B}_1, \ldots, \mathcal{B}_n$ such that

$$\mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathcal{A}) \leq \left(8 \cdot (1 + q_G + q_P)^2 + 1\right) \sum_{i \in [n]} \delta_i + (1 + 2q_P + 2q_G) \cdot \sqrt{\min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathcal{B}_1), \ldots, \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_n}(\mathcal{B}_2)\}}$$

where $\mathcal{B}_1, \ldots, \mathcal{B}_n$ run in about the same time as $\mathcal{A}$ and make at most $q_G + q_P$ queries to the QROs.

Proof. The detailed proof of Theorem 7 can be found in Appendix A. □

$\mathcal{B}_1^{\mathcal{A},G}(\mathsf{pk}_1, \mathsf{ct}_1^*)$

$\mathsf{pp}_2 \leftarrow\$\ \mathsf{setup}_2(1^\lambda)$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\$\ \mathsf{gen}_2(\mathsf{pp}_2)$
$\sigma_2^* \leftarrow\$\ \mathcal{M}_2$
$\mathsf{ct}_2^* \leftarrow \mathsf{enc}_2(\mathsf{pk}, \sigma_2^*)$
$K \leftarrow\$\ \mathcal{K}$
run $\mathcal{A}^{\mathcal{O}_2^{\mathsf{Dec}}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), K)$
sample a random query (m, h) from $\mathcal{L}_G \cup \mathcal{L}_H$
if $(m, h) \in \mathcal{L}_G$:
  parse $((i, \mathsf{pt}_1', \sigma_2^*), g) \leftarrow m$
  return pt_1'
if $(m, h) \in \mathcal{L}_H$:
  return $\sigma_2^* \oplus m$

Fig. 15: OW-CPA adversary for the proof of Theorem 5.

This result then implies that $\mathsf{QU}_m^\perp \circ \mathsf{T}_{\|}^n$ is a robust FO-like combiner in the QROM by using Theorem 4.5 of Hofheinz et al. [14]. Note that the proof of Theorem 7 is very similar to the proofs of FO security in the QROM. As a result, the bound could much likely be made tighter using recent QROM techniques (e.g. [4,17,21]). In addition, we conjecture that our main combiner $\mathsf{UT}_{\|}$ could be proven secure in the QROM without the additional hash, using the compressed oracle techniques by Zhandry [24]. We leave these improvements as future work.

5.3 Other combiners

It has been shown that the implementation of ROs in FO-like transforms, in particular in the de-randomization step (i.e. computation of the deterministic coins), is particularly vulnerable to implementation mistakes [1]. Thus, it is of interest to study how these coins can be computed without compromising the security of the resulting scheme. We show in this section how hash functions (i.e. ROs) can be combined s.t. the de-randomization step is secure and efficient. Many combinations of hash functions are possible and we propose a few of those below, offering flexibility to implementors. Finally, we consider using different hash functions to increase the security at no (or very small) cost. This relates to the notion of hash combiner [7,8], which constructs a hash function that fulfils certain security properties as long as one of the underlying hash functions has this property. In our case, we want the hash functions to behave as random oracles, thus we can combine two different functions to make the whole scheme secure as long as *one* of the hash functions is indistinguishable from a RO.

How to combine hash functions. From now on, in order to distinguish (random) functions from random oracles, we denote a function by a small letter and a RO by a capital letter (e.g. $g(x)$ is a function evaluated on x and $G(x)$ is a

$g(i, \sigma_1, \sigma_2)$
$G(\sigma_1 \oplus \sigma_2) \oplus G(i, \sigma_i)$
$G_1(i, \sigma_1) \oplus G_2(i, \sigma_2)$

Table 2: Different g functions, where G, G_i are ROs.

RO queried on x). Note that in our case, the functions are defined using random oracles (e.g. $g(x) := G(1, x) \oplus G(2, x)$). We consider replacing the RO G in our combiners by such a random function g (but still in the ROM).

One can see from the proofs of security of both $\mathsf{T}_{\|}$ and $\mathsf{UT}_{\|}$ that we want the deterministic coins to be indistinguishable from random ones until we can recover the seeds (or plaintexts) from the list of queries. In addition to this property, one also wants the values $g(i, \sigma_1, \sigma_2)$ to be close to uniform. Indeed, in the proof of Theorem 5, we extensively use the fact that the correctness and spreadness property hold with probability at least δ and $2^{-\gamma}$, respectively, even when the coins are not random but computed as $g(i, \sigma_1, \sigma_2)$. Obviously, if the values $g(1, \sigma_1, \sigma_2)$ are not sampled uniformly at random, this may not hold anymore. In other words, we want $g(i, \sigma_1, \sigma_2)$ to be either computable by the adversary using its queries to G or distributed uniformly at random. We developped formal definitions (called *Extractable Random Function (ERF)* and *Indistinguishable unless Queried (IUQ)*) capturing these properties, they are presented in Appendix B. We give two examples of such functions g satisfying these properties in Table 2. Note that these are based on a RO G.

Replacing H. As we did for G, one can also replace the key derivation function H by another random function h. However, as the way we derive the key in the combiner $\mathsf{QU}_{\|}$ (i.e. $H(\sigma_1 \oplus \sigma_2)$) is already efficient, we do not give any example of such function h here. More details can be found in Appendix B.

Then, one can show the following theorem.

Theorem 8 (Informal). *Let g be a function from Table 2. Let $h(\sigma_1, \sigma_2) := H(\sigma_1, \sigma_2)$ or $h(\sigma_1, \sigma_2) := H(\sigma_1) \oplus H(\sigma_2)$. Then, the $\mathsf{UT}_{\|}$ transform where the deterministic coins for encrypting the seed σ_i are computed as $g(i, \sigma_1, \sigma_2)$ instead of $G(i, \sigma_1, \sigma_2)$ and the key is derived as $h(\sigma_1, \sigma_2)$, is still a robust combiner.*

A formal version of this result is proven in Appendix B.3.

Hash combiners. As some of the proposed functions g use more than one hash functions, these functions are themselves hash combiners. Thus, it is of interest to study the robustness of such constructions. That is, if one of the underlying hash functions is broken (i.e. shown not to behave as a RO), is the g function (thus the whole FO-like combiner) still secure? As one of the main security concerns of the use of FO-like transforms is that the proofs are in the ROM, using robust hash combiners may improve the trust in such constructions.

The last function g in Table 2 is actually a robust combiner with respect to the RO property and one of the seeds. That is, $G_1(i, \sigma_1) \oplus G_2(i, \sigma_2)$ is indistinguishable from a RO, even if G_1 (or G_2) is any function. Hence, if we take both $g(i, \sigma_1, \sigma_2) = G_1(i, \sigma_1) \oplus G_2(i, \sigma_2)$ and $h(\sigma_1, \sigma_2) = H_1(\sigma_1) \oplus H_2(\sigma_2)$ in the FO-like combiner, we will obtain a secure KEM as long as G_i **and** H_i **and** PKE_i are secure for some $i \in [2]$.

Proposition 1 (Informal). *Let* $g(i, \sigma_1, \sigma_2) = G_1(i, \sigma_1) \oplus G_2(i, \sigma_2)$ *and* $h(\sigma_1, \sigma_2) = H_1(\sigma_1) \oplus H_2(\sigma_2)$. *We call a tuple* $(G_i, H_i, \mathsf{PKE}_i)$ *secure if* G_i, H_i *are ROs and* PKE_i *is OW-CPA. Let* KEM *be the hybrid KEM resulting from applying* $\mathsf{U}^{\not\perp}_{\parallel}$ *on* PKE_1 *and* PKE_2 *with* g *and* h *to derive the deterministic coins and key, respectively. Then,* KEM *is IND-CCA if* $(G_1, H_1, \mathsf{PKE}_1)$ *or* $(G_2, H_2, \mathsf{PKE}_2)$ *(or both) is secure.*

Proof sketch. We assume w.l.o.g. that the tuple $(G_1, H_1, \mathsf{PKE}_1)$ is secure and G_2, H_2 can be any functions and PKE_2 might not be OW-CPA. In addition, we assume G_1, H_1, G_2, H_2 are mutually independent functions (e.g. this can be implemented by RO separation). The result follows simply from the fact that in the IND-CCA game against KEM, as long as G_1 is a RO, the coins $G_1(i, \sigma_1) \oplus G_2(i, \sigma_2)$ are indistinguishable from uniform unless (i, σ_1) is queried, irrespectively of the value $G_2(i, \sigma_2)$. But in turn such a query would break the OW-CPA assumption on PKE_1 (or happens with negligible probability). The same argument for $h(\sigma_1, \sigma_2) = H_1(\sigma_1) \oplus H_2(\sigma_2)$ holds that the key will always be indistinguishable from uniform if H_1 is a RO and PKE_1 is OW-CPA. □

6 Implementation

As a proof of concept, we implemented a fully PQ hybrid KEM using two IND-CPA proposals that passed to the Round 2 of the standardization process and our combiner. As the main goal of our combiner is to increase the security while still offering good performances, we chose HQC and LAC since

1. LAC is one of the most efficient schemes in term of speed and public-key/ciphertext size but it has been attacked recently in [12]. More generally, it seems LAC is more vulnerable to failure attacks than other schemes and that led this scheme to be dropped for Round 3. Thus, using it along another cryptosystem does not imply a large overhead while preventing a failure attack alone against LAC to break the whole scheme.
2. HQC is a code-based scheme that offers good performance, although the hardness assumption it is based on has not been extensively studied as of yet. Thus, combining it with another efficient scheme might provide more confidence in this scheme at the expense of a small overhead.
3. HQC is code-based while LAC is lattice-based. Therefore, one can hope that any improvement in breaking the assumption of one does not lead to a better cryptanalysis of the other.

Scheme	**SK** (B)	**PK** (B)	**CT** (B)	**KeyGen** (μs)	**Encaps** (μs)	**Decaps** (μs)
`frodo640`	19888	9616	9720	847.553	4650.037	4602.284
`hqc128`	3165	3125	6234	144.166	298.120	528.624
`kyber512`	1632	800	736	154.077	210.857	263.194
`lac128`	1056	544	712	115.308	199.776	311.709
`hqc_lac128`	4221	3669	6882	260.032	484.969	813.452
`hqc_lac128_par`	4221	3669	6882	162.502	315.137	549.516

Table 3: Performance of `hqc_lac128` and `hqc_lac128_par` compared to other schemes. The size of the public/secret key and ciphertext are in bytes. The time for key generation, encapsulation, decapsulation is in microseconds.

6.1 Design choices

We used the reference IND-CPA implementations provided by the authors in the second round for both schemes. Then, we applied our $\mathsf{UT}_{\|}$ combiner. In practice we implemented $G(1,\cdot,\cdot)$ as $\mathsf{SHA256}(\cdot)$, $G(2,\cdot,\cdot)$ as the AES-based expansion function provided by the NIST and $H(\cdot)$ as $\mathsf{SHA512}(\cdot)$. These choices made the implementation easier as we could stick to most of the author's choices. For example, HQC encryption function in the original FO transform is using a seed output by the AES-based expander and our choice of $G(2,\sigma_1,\sigma_2)$ makes it possible to reuse most of the code.

We implemented two versions of the hybrid cryptosystem, a standard version that we are calling `hqc_lac128` and a parallel version denoted by `hqc_lac128_par`, both using the Level 1 (i.e. aiming at 128 bits of classical security) reference implementations of LAC and HQC. The parallel implementation uses the `pthread` library and is implemented without any other optimization. In particular, only the encryption of the seeds is parallelized in the encapsulation function (i.e. the encryption functions of LAC and HQC are called in different threads) and only the decryption and reencryption is parallelized in the decapsulation procedure.

6.2 Results and efficiency

We tested both our hybrid schemes on a laptop running Ubuntu 14.04 with an Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz. The results for our hybrid schemes, the original schemes and reference implementations of two other popular lattice-based schemes (Frodo and Kyber) are reported in Table 3. The sizes are in bytes and the times are given in microseconds (10^{-6}s) and are averaged over 10000 runs. Obviously, the size of the public/secret key and ciphertext are the addition of the corresponding ones in LAC and HQC, except for the ciphertext, which is a bit smaller. This follows from the fact that the ciphertext in HQC contains a confirmation hash that we omit in our FO-like combiner. One can see that compared to a proposal with large keys and ciphertexts (i.e. Frodo), our hybrid compares well. In addition, as LAC produces small outputs, the increase compared to HQC is small. That is, the size of the secret key, public key and ciphertext is increased by roughly 33%, 17% and 10%, respectively.

Considering the speed, the non-optimized hybrid `hqc_lac128` performs slightly better than both LAC and HQC run one after the other. However, all procedures are still much faster than the ones of a slower scheme, like Frodo. On the other hand, the parallelized hybrid `hqc_lac128_par` offers very good performance as one could expect from such a parallelizable design. In particular, we observe only a 13%, 6% and 4% increase of latency compared to HQC, for key generation, encapsulation and decapsulation, respectively. Therefore, `hqc_lac128_par` can perform nearly as good as HQC on systems that offers efficient parallelization, such as laptops or any machine with regularly idle processors.

We give on Figure 16 a visualisation of the performance of `hqc_lac128` compared to other round 2 candidates with security Level 1. Most of the data comes from the SUPERCOP [2] benchmarking system (we picked the results of a test performed on a 2018 Intel Core i7-8809G). All round 2 proposals are represented, except for BIKE, Round 5 and LEDACrypt, which did not have an IND-CCA version benchmarked at the time of the test. We still added the keys and ciphertext sizes of BIKE as they are similar to the ones of HQC.

For the hybrid scheme `hqc_lac128`, we computed the cycles needed for key generation, encapsulation and decapsulation as the sum of the corresponding cycles needed by LAC and HQC. Note that this is a pessimistic approximation as the hybrid system requires less instructions than the sum of both underlying schemes (e.g. we apply some hash functions only once), this is confirmed in practice by the results shown in Table 3. We do not plot the parallelized version `hqc_lac128_par` as the sizes are the same as in `hqc_lac128` and the time is upper bounded by the latter as well.

Analysis. From all three graphs in Figure 16, we can deduce that our hybrid does not perform particularly well compared to other schemes in these metrics. However, one can see that the bottleneck is the use of HQC here. In particular, `hqc_lac128` performs nearly as well as HQC in the metrics considered. This confirm what we wanted to show, that is boosting security by combining a very efficient scheme with one that is less so does not worsen much the performance of the latter one. In other words, if one is willing to use HQC, one can as well use the hybrid `hqc_lac128` for a very small overhead but arguably much better security.

Finally, one can wonder what is the speedup of our combiners compared to existing ones. We take as an example the XtM combiner from Bindel et al. [3], which applies a special kind of MAC to the ciphertexts and keys. It is proposed to implement this primitive as the concatenation (or the XOR) of two standard MACs. This computation is the main overhead compared to our construction and we simulated it as two calls to SHA256 on both ciphertexts and keys. This takes approximately $40\mu s$ on our setup, hence the speedup when considering `hqc_lac128_par` is slightly over 10% for encapsulation. This obviously depends on many factors like hardware, hash functions, parallelization and the underlying schemes. For example, for small ciphertexts the speedup will be negligible while for large ones it will be more important. Finally, we note that PQ schemes are not optimized thus the gain might be more noticeable in the future.

6.3 Other hybrid KEMs.

While `hqc_lac128` is an interesting example of the advantages offered by a PQ hybrid KEM, one might wonder what is the optimal combination of schemes according to some metrics. Using the same data, we computed the theoretical performance of all possible hybrid made of two PKEs based on different assumptions (e.g. code and lattice). We considered the fastest ones in encapsulation/decapsulation and the ones with the smallest public key/ciphertext size. We present some of the most efficient ones according to these metrics in Table 4. The schemes that perform best in term of size were obviously based on SIKE, which has by far the smallest public key/ciphertext sizes but suffer from slow computation. Thus, we chose to show only three of these hybrid in Table 4, in order to offer a broader view of possible schemes. Similarly, we showed only three of the best hybrid schemes based on McEliece and NTS, which have extremely fast encapsulation/decapsulation procedures but have very large public keys. Overall, non-lattice-based schemes are quite slower than lattice-based ones (although some data on BIKE is missing), thus it seems that combining schemes of these two types will not give small public key *and* fast encapsulation/decapsulation. Nevertheless, in Table 4, we also include a lattice/rank-based hybrid scheme for completeness (i.e. `NTRUhps_rqcI`) and a LAC-RSA hybrid KEM as an interesting comparison.

Scheme	**PK** (B)	**CT** (B)	**Encaps** (cycles)	**Decaps** (cycles)
`kyber512_sike`	1 178	1 138	17 652 847	18 817 320
`lac128_sike`	922	1 114	17 677 983	18 871 919
`NTRUhps_sike`	1 077	1 101	17 643 917	18 826 865
`NTRUhps_bike2`	2 171	2 171	-	-
`lightsaber2_bike2`	2 144	2 208	-	-
`lac_bike2`	2 016	2 184	-	-
`NTRUhps_McEliece`	261 819	827	74 361	168 478
`kyber512_McEliece`	261 920	928	83 291	158 933
`lightsaber2_McEliece`	261 792	864	102 172	186 370
`NTRUhps_NTSkem`	320 187	827	140 165	371 082
`kyber512_NTSkem`	320 288	928	123 001	334 107
`lightsaber2_NTSkem`	320 160	864	141 882	361 544
`NTRUhps_rqcI`	1 552	2 389	374 470	1 265 545

Table 4: Selection of efficient hybrid schemes.

We give a visualisation of the performance of these hybrid schemes compared to the NIST proposals (and RSA 2048) in Figure 17. On the first figure, one can easily identify the hybrid schemes based on McEliece and NTS on the right. Then, one can see that the ones based on SIKE have a slightly worse performance than most lattice-based schemes but still give good efficiency in the size metric. In particular, they have a similar performance as `lac_rsa`. Finally, both the hybrid

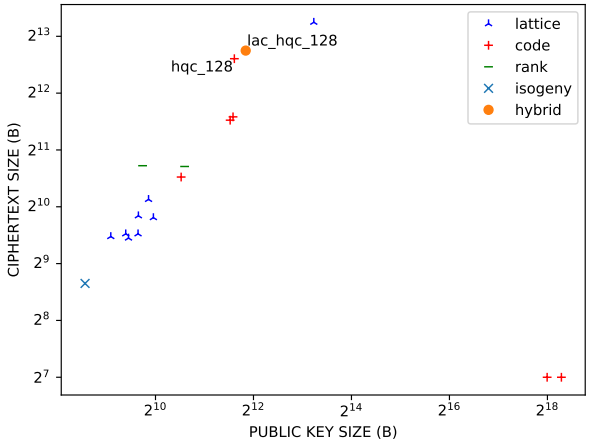
schemes based on BIKE and `NTRUhps_rqcI` have public key and ciphertext sizes that lie between those of the rank-based proposals and some code-based ones.

On the second figure, one can see that hybrid schemes based on SIKE are slow due to the underlying scheme. On the other hand, in term of speed the hybrid systems based on McEliece and NTS offer competitive performance. However, `NTRUhps_rqcI` is the only full PQ hybrid considered that has slightly worse than average performance in all metrics considered (i.e. bandwidth and speed). Interestingly, we see that the decapsulation latency of RSA is one of the worst among the schemes considered, and thus the hybrid `lac_rsa` suffers from slow decapsulation as well.

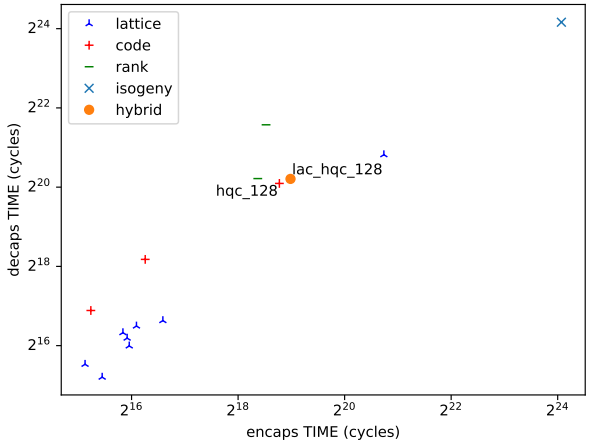
In general, several lattice-based schemes offer good performance in both the chosen metrics. Hence, the hybrid constructions mostly inherits the advantages and disadvantages of the second PKE scheme used in the construction (i.e. isogeny, code or rank-based). Furthermore, one can see from Figure 17 that composing a hybrid KEM from an "extreme" scheme (i.e. a scheme that performs very well in one metric but very badly in another) might not be the best option.

It seems that a better approach would be to combine two schemes based on the same type of assumptions. However, we could loose some security since a breakthrough in breaking one of the assumptions could automatically imply breaking the other one. A more complete study is out of the scope of this paper and we leave it as future research.

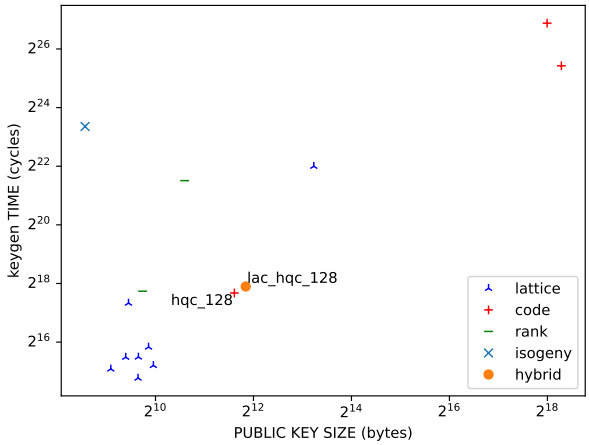
Acknowledgements. Loïs Huguenin-Dumittan is supported by a grant (project № 192364) of the Swiss National Science Foundation (SNSF).



(a) Public Key size vs Cipertext size.

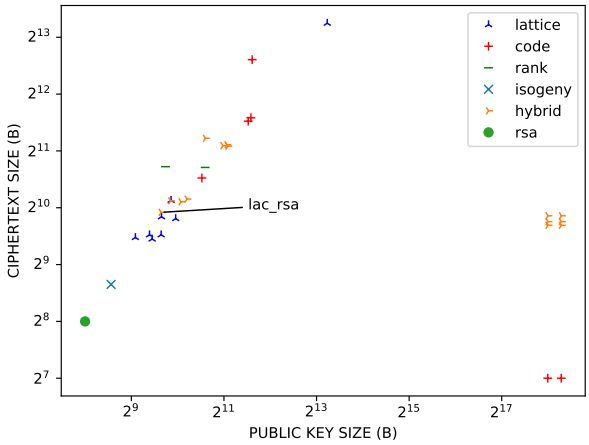


(b) Encapsulation time vs Decapsulation time.

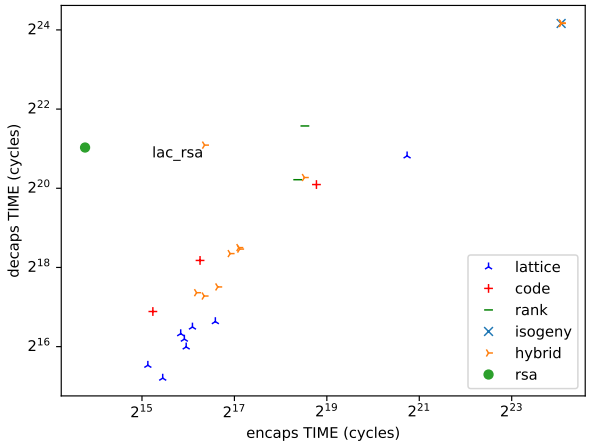


(c) Public Key size vs Key Generation time.

Fig. 16: Visualisation of the performance of `hqc_lac128` compared to several Level 1 implementation of NIST round 2 proposals.



(a) Public Key size vs Cipertext size.



(b) Encapsulation time vs Decapsulation time.

Fig. 17: Visualisation of the performance of different hybrid schemes (see Table 4) compared to several Level 1 implementation of NIST round 2 proposals.

References

1. Bellare, M., Davis, H., Günther, F.: Separate your domains: Nist pqc kems, oracle cloning and read-only indifferentiability. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 3–32. Springer (2020)
2. Bernstein, D.J., (editors), T.L.: eBACS: ECRYPT benchmarking of cryptographic systems (accessed 14 May 2020), https://bench.cr.yp.to
3. Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., Stebila, D.: Hybrid key encapsulation mechanisms and authenticated key exchange. In: Ding, J., Steinwandt, R. (eds.) Post-Quantum Cryptography. pp. 206–226. Springer International Publishing, Cham (2019)
4. Bindel, N., Hamburg, M., Hövelmanns, K., Hülsing, A., Persichetti, E.: Tighter proofs of CCA security in the quantum random oracle model. In: Hofheinz, D., Rosen, A. (eds.) Theory of Cryptography. pp. 61–90. Springer International Publishing, Cham (2019)
5. Bindel, N., Herath, U., McKague, M., Stebila, D.: Transitioning to a quantum-resistant public key infrastructure. Cryptology ePrint Archive, Report 2017/460 (2017), https://eprint.iacr.org/2017/460
6. Dodis, Y., Katz, J.: Chosen-ciphertext security of multiple encryption. In: Theory of Cryptography Conference. pp. 188–209. Springer (2005)
7. Fischlin, M., Lehmann, A.: Multi-property preserving combiners for hash functions. In: Theory of Cryptography Conference. pp. 375–392. Springer (2008)
8. Fischlin, M., Lehmann, A., Pietrzak, K.: Robust multi-property combiners for hash functions revisited. In: International Colloquium on Automata, Languages, and Programming. pp. 655–666. Springer (2008)
9. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Annual International Cryptology Conference. pp. 537–554. Springer (1999)
10. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of Cryptology **26**(1), 80–101 (2013), https://doi.org/10.1007/s00145-011-9114-1
11. Giacon, F., Heuer, F., Poettering, B.: KEM Combiners. Cryptology ePrint Archive, Report 2018/024 (2018), https://eprint.iacr.org/2018/024
12. Guo, Q., Johansson, T., Yang, J.: A novel cca attack using decryption errors against lac. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 82–111. Springer (2019)
13. Herzberg, A.: On tolerant cryptographic constructions. In: Cryptographers' Track at the RSA Conference. pp. 172–190. Springer (2005)
14. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Theory of Cryptography Conference. pp. 341–371. Springer (2017)
15. Huguenin-Dumittan, L., Vaudenay, S.: Fo-like combiners and hybrid post-quantum cryptography. Cryptology ePrint Archive, Report (to appear) (2021)
16. Jean-Sébastien, C., Handschuh, H., Joye, M., Paillier, P., Pointcheval, D., Tymen, C.: GEM: A generic chosen-ciphertext secure encryption method. In: Preneel, B. (ed.) Topics in Cryptology — CT-RSA 2002. pp. 263–276. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
17. Kuchta, V., Sakzad, A., Stehlé, D., Steinfeld, R., Sun, S.F.: Measure-rewind-measure: tighter quantum random oracle model proofs for one-way to hiding and

cca security. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 703–728. Springer (2020)
18. Merkle, R.C., Hellman, M.E.: On the security of multiple encryption. Commun. ACM **24**(7), 465–467 (Jul 1981). https://doi.org/10.1145/358699.358718, https://doi.org/10.1145/358699.358718
19. Okamoto, T., Pointcheval, D.: REACT: Rapid enhanced-security asymmetric cryptosystem transform. In: Naccache, D. (ed.) Topics in Cryptology — CT-RSA 2001. pp. 159–174. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
20. Poettering, B., Rösler, P.: Combiners for aead. IACR Transactions on Symmetric Cryptology pp. 121–143 (2020)
21. Saito, T., Xagawa, K., Yamakawa, T.: Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 520–551. Springer International Publishing, Cham (2018)
22. Targhi, E.E., Unruh, D.: Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In: Theory of Cryptography Conference. pp. 192–216. Springer (2016)
23. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In: Annual Cryptology Conference. pp. 758–775. Springer (2012)
24. Zhandry, M.: How to record quantum queries, and applications to quantum indifferentiability. In: Annual International Cryptology Conference. pp. 239–268. Springer (2019)
25. Zhang, C., Cash, D., Wang, X., Yu, X., Chow, S.S.: Combiners for chosen-ciphertext security. In: International Computing and Combinatorics Conference. pp. 257–268. Springer (2016)

A QROM security

In this Appendix, we briefly recall the notion of QROM before proving the security of the $\mathsf{QU}_m^{\perp} \circ \mathsf{T}_{\|}^n$ combiner in this model.

Reminders and useful lemmas

QROM. In the Quantum Random Oracle Model (QROM), a hash function H is represented as an ideal random function quantumly accessible by the different parties. That is, on a quantum query $\sum_{x,y} \alpha_{x,y}|x,y\rangle$, the QRO $|H\rangle$ returns $\sum_{x,y} \alpha_{x,y}|x,y \oplus H(x)\rangle$, where $H(\cdot)$ is an ideal random function (i.e. a classical random oracle). In general, for a function $H(\cdot)$ and an adversary $\mathcal{A}$, we write $\mathcal{A}^{|H\rangle}$ to say that $\mathcal{A}$ has quantum access to H.

$2q_H$-wise independent functions and QROM. The following result was shown by Zhandry in [23].

Lemma 2. *No adversary limited to q_H quantum queries to an oracle $|H\rangle$ can distinguish between the case where $|H\rangle$ is a QRO and the case where $|H\rangle$ is a $2q_H$-wise independent function.*

The previous lemma implies that in a reduction, one can perfectly and efficiently simulate a QRO by sampling a $2q_H$-wise independent function.

$\mathrm{GSBP}_\lambda(\mathcal{A})$

$(\lambda(x))_{x\in\mathcal{X}} \leftarrow \mathcal{A}$
if $\exists x$ s.t. $\lambda(x) > \lambda$: **return** 0
for $x \in \mathcal{X}$: $F(x) \leftarrow\$ \mathsf{Ber}(\lambda(x))$
$x \leftarrow \mathcal{A}^{|F\rangle}$
return $F(x)$

Fig. 18: GSPB game.

Generic Search Problem with Bounded Probabilities (GSPB). The GSPB problem is defined in [14]. Informally, a quantum and unbounded adversary $\mathcal{A}$ has access to a quantum oracle $|F\rangle$, where $F(x) = 1$ with some probability $\lambda(x) \leq \lambda$. The goal of the adversary is to find x s.t. $F(x) = 1$, by querying $|F\rangle$ q times. The following lemma upper bounds the probability that $\mathcal{A}$ succeeds.

Lemma 3 (Lemma 4.2, [14]). *Let* $\mathsf{GSBP}_\lambda(\mathcal{A})$ *be the game defined in Figure 18, where* $\mathsf{Ber}(\lambda)$ *is the Bernoulli distribution with parameter* λ *(i.e. it outputs* 1 *with prob.* λ*,* 0 *othterwise. Then, for any quantum and unbouded adversary* $\mathcal{A}$ *limited to* q *queries to* $|F\rangle$*, we have*

$$\Pr[GSBP_\lambda(\mathcal{A}) \Rightarrow 1] \leq 8 \cdot \lambda \cdot (q+1)^2 \ .$$

where $F : \mathcal{X} \mapsto \{0,1\}$ *for some set* $\mathcal{X}$*.*

Lemma 3 is useful to upper bound the probability of a correctness error in the game. More precisely, in the security proof of $\mathsf{UT}_{\not\perp}$ (Theorem 5), we upper bound the probability that some query $G(i,\sigma_1,\sigma_2)$ during the game is s.t. $\mathsf{dec}_i(\mathsf{enc}_i(\sigma_i; G(i,\sigma_1,\sigma_2))) \neq \sigma_i$. We can call this event fail_i and we state the following lemma, which is a simple rephrasing of Lemma 4.3 in [14].

Lemma 4. *Let* PKE_i *be a* δ_i*-correct PKE and let* fail_i *be the event defined as* $G_i(\mathsf{pt}_1,\ldots,\mathsf{pt}_n)$ is queried classicaly by the game and $\mathsf{dec}_i(\mathsf{enc}_i(\mathsf{pt}_i; G_i(\mathsf{pt}_1,\ldots,\mathsf{pt}_n))) \neq \mathsf{pt}_i$*. We consider a game* Γ *where at most* q_{G_i} *(quantum or not) queries to* $|G_i\rangle$ *are made. Then,*

$$\Pr[\mathsf{fail}_i] \leq 8 \cdot \delta_i \cdot (q_{G_i}+1)^2 \ .$$

Proof sketch. The proof is the same as Lemma 4.3 in [14]. Briefly, the GSBP problem with $\lambda = \delta$ can be reduced to the problem of triggering fail_i in the game Γ.

(Algorithmic) One-Way to Hiding Lemma (AO2WH). Finally, we briefly recall the highly popular One-Way to Hiding (OW2H) Lemma in the version found in [14]. Note that several improvements of this lemma and of similar proofs have been presented these last few years (e.g. [4,17,21]). Those variants would most likely improve the security bound of Theorem 9 presented below, but for simplicity we stick to one of the simplest version.

Lemma 5 (AOW2H). *Let $\mathcal{A}$ be a quantum adversary making at most q_H queries to the QRO $|H\rangle : \{0,1\}^n \mapsto \{0,1\}^m$ and outputting 0 or 1. Let $\mathsf{Ext}_{q_H}^{|H\rangle}(\mathcal{A})$ be the algorithm in Figure 19. Then, for any algorithm F that does not use $|H\rangle$*

$$\begin{aligned}&\big|\Pr[\mathcal{A}^{|H\rangle}(inp) \Rightarrow 1 | \sigma \leftarrow\!\$\, \{0,1\}^n; inp \leftarrow \mathsf{F}(\sigma, H(\sigma))] \\ &- \Pr[\mathcal{A}^{|H\rangle}(inp) \Rightarrow 1 | (\sigma, K) \leftarrow\!\$\, \{0,1\}^{n+m}; inp \leftarrow \mathsf{F}(\sigma, K)]\big| \\ &\leq 2q_H \sqrt{\Pr[\sigma \leftarrow \mathsf{Ext}^{\mathcal{A},|H\rangle}(inp) | (\sigma, K) \leftarrow\!\$\, \{0,1\}^{n+m}; inp \leftarrow \mathsf{F}(\sigma, K)}} \quad .\end{aligned}$$

$\mathsf{Ext}^{\mathcal{A},|H\rangle}(inp)$

$i \leftarrow\!\$\, [q_H]$

run $\mathcal{A}^{|H\rangle}(inp)$ until i-th query $|\mathsf{QUERY}\rangle = \sum_{x,y} \alpha_{x,y}|x,y\rangle$

$x' \leftarrow$ measure first register of $|\mathsf{QUERY}\rangle$

if $\mathcal{A}$ did not make i queries : **return** $\perp$

return x'

Fig. 19: Extractor Ext for the AOW2H lemma.

Security of FO-like combiner in the QROM

The secure combiner we propose in the QROM is our $\mathsf{T}_{\|}$ construction composed with the QROM secure $\mathsf{QU}_m^{\perp}$ transform from [14]. This simplifies the proof as we only need to prove that $\mathsf{T}_{\|}$ holds a OW-PCA PKE in the QROM, as $\mathsf{QU}_m^{\perp}$ is already known to convert a OW-PCA scheme into an IND-CCA KEM in the Quantum Random Oracle Model. Our main combiner $\mathsf{QU}_{\|}$ with a confirmation hash (i.e. adding $H'(\sigma)$ to the ciphertext) can be proven secure in the QROM in a similar way. We note that the bound could much likely be made tighter using recent QROM techniques (e.g. [4,17,21]). In addition, we conjecture that our main combiner $\mathsf{UT}_{\|}$ could be proven secure in the QROM without the additional hash, using the compressed oracle techniques by Zhandry [24]. We leave these improvements as future work.

In the following theorem, we show that the $\mathsf{T}_{\|}$ combiner generalized to n schemes holds a OW-PCA PKE in the QROM. We call this construction $\mathsf{T}_{\|}^n$ and it is presented in Figure 20. Note that for the sake of the proof, the deterministic coins are computed as $G_i(\cdot)$ instead of $G(i,\cdot)$ as in the original $\mathsf{T}_{\|}$ transform.

Theorem 9. *Let PKE be the PKE resulting from applying $\mathsf{T}_{\|}^n$ on $\mathsf{PKE}_1, \ldots, \mathsf{PKE}_n$, which are respectively $\delta_1, \ldots, \delta_n$-correct. In addition, let G_i be hash functions modelled as (independent) quantum random oracles. Then, for all quantum OW-PCA adversary $\mathcal{A}$ making at most q_G queries to all oracles*

$\mathsf{gen}()$

for $i \in [n]$:
  $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\$\, \mathsf{gen}_i$
$\mathsf{pk} \leftarrow (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$
$\mathsf{sk} \leftarrow (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$
return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{enc}(\mathsf{pk}, (\mathsf{pt}_1, \ldots, \mathsf{pt}_n))$

parse $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n) \leftarrow \mathsf{pk}$
for $i \in [n]$:
  $r_i \leftarrow G_i(\mathsf{pt}_1, \ldots, \mathsf{pt}_n)$;
  $\mathsf{ct}_i \leftarrow \mathsf{enc}_i(\mathsf{pk}_i, \mathsf{pt}_i; r_i)$
return $(\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$

$\mathsf{dec}(\mathsf{sk}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_n))$

parse $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{sk}$
for $i \in [n]$:
  $\mathsf{pt}'_i \leftarrow \mathsf{dec}_i(\mathsf{sk}_i, \mathsf{ct}_i)$
for $i \in [n]$:
  if $\mathsf{enc}_i(\mathsf{pk}_i, \mathsf{pt}'_i; G_i(\mathsf{pt}'_1, \ldots, \mathsf{pt}'_n)) \neq \mathsf{ct}_i$:
  return $\perp$
return $(\mathsf{pt}'_1, \ldots, \mathsf{pt}'_n)$

Fig. 20: $\mathsf{T}^n_{\|}$ combiner.

G_i and q_P queries to the plaintext-checking oracle, there exists adversaries $\mathcal{B}_1, \ldots, \mathcal{B}_n$ such that

$$\mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathcal{A}) \leq \left(8 \cdot (1 + q_G + q_P)^2 + 1\right) \sum_{i \in [n]} \delta_i + (1 + 2q_P + 2q_G) \cdot \sqrt{\min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathcal{B}_1), \ldots, \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_n}(\mathcal{B}_2)\}}$$

where $\mathcal{B}_1, \ldots, \mathcal{B}_n$ run in about the same time as $\mathcal{A}$ and make at most $q_G + q_P$ queries to the QROs.

Proof. We proceed with a sequence of hybrid games detailed in Figure 21. The adversary has access to the n different QROs $|G_1, \ldots, G_n\rangle$ which can be defined as one oracle $|G\rangle = |G_1, \ldots, G_n\rangle$. We assume that the message spaces $\mathcal{M}_i$ are equal to $\{0,1\}^{\ell_i}$ for some integer ℓ_i and that $G_j : \{0,1\}^* \mapsto \{0,1\}^n$.

Game Γ^0: This is the original OW-PCA game in the QROM except we enforce correctness of the challenge ciphertext (i.e. $\mathsf{dec}(\mathsf{pk}, \mathsf{ct}^*) = \mathsf{pt}^*$). As the correctness is broken for ct^* if it is broken for at least one of the components ct^*_i, the probability of that happening is at most $\sum_{i \in [n]} \delta_i$.

Game Γ^1: In this game, we simulate the plaintext-checking oracle by checking whether $\mathsf{enc}_j(\mathsf{pt}_j; G_j(\mathsf{pt}_1, \ldots, \mathsf{pt}_j)) = \mathsf{ct}_j$ for all $j \in [n]$. As seen in the proof of Theorem 4, the simulation is not perfectly iff one of the $(\mathsf{pt}_1, \ldots, \mathsf{pt}_n)$ queried is such that the correctness is broken, i.e. $\mathsf{dec}_j(\mathsf{enc}_j(\mathsf{pt}_j; G_j(\mathsf{pt}_1, \ldots, \mathsf{pt}_n))) \neq \mathsf{pt}_j$ for some $j \in [n]$. This is precisely the fail_j event defined in Lemma 4. Now, the number of quantum and classical (i.e. a quantum query not in superposition) queries made to G_j throughout the game are at most $1 + q_G + q_P$ (i.e. 1 in the computation of the challenge plaintext, q_G by the adversary and q_P for queries to the plaintext-checking oracle). Therefore, we have $\Pr[\mathsf{fail}_j] \leq 8 \cdot (1 + q_G + q_P)^2 \cdot \delta_j$.

Overall, we have

$$|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]| \leq \Pr\left[\cup_{j \in [n]} \mathsf{fail}_j\right] \leq 8 \cdot (1 + q_G + q_P)^2 \cdot \sum_{j \in [n]} \delta_j$$

where the second inequality follows from a union bound.

Game Γ^2: In game Γ^2, we replace the deterministic coins $G_j(\mathsf{pt}_1^*, \ldots, \mathsf{pt}_n^*)$ by random coins $r_j \leftarrow\!\$\, \{0,1\}^r$ for all $j \in [n]$. We can then use the One-Way to Hiding Lemma to upper bound $|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]|$. First, we consider the RO $G := G_1 \otimes \ldots \otimes G_n$ s.t. $G(m) = (G_1(m), \ldots, G_n(m))$ and the function $\mathsf{F}(x, y)$ shown in Figure 22 which outputs $inp = (\mathsf{pk}, \mathsf{ct}^*)$. In addition, let $\mathcal{A}'$ be the adversary that receives inp, run $\mathcal{A}^{\mathcal{O}_1^{\mathsf{PCO}}}(\mathsf{pk}, \mathsf{pt}^*)$ by simulating the plaintext-checking oracle (this is possible since the secret key is not used in $\mathcal{O}_1^{\mathsf{PCO}}$ anymore) and outputs 1 iff $\mathcal{A}$ outputs pt' s.t. $\mathsf{enc}(\mathsf{pk}, \mathsf{pt}') = \mathsf{ct}^*$ (this is equivalent to $\mathsf{pt}' = \mathsf{pt}^*$ by the perfect correctness of the challenge ciphertext). By the AOW2H Lemma (Lemma 5), one can easily see that

$$
\begin{aligned}
&\big|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]\big| = \\
&\big|\Pr[\mathcal{A}'^{|G\rangle}(inp) \Rightarrow 1 | \mathsf{pt}^* \leftarrow\!\$\, \{0,1\}^{\ell_1+\ldots+\ell_n}; inp \leftarrow \mathsf{F}(\mathsf{pt}^*, G(\mathsf{pt}^*))] \\
&- \Pr[\mathcal{A}'^{|G\rangle}(inp) \Rightarrow 1 | (\mathsf{pt}^*, r^*) \leftarrow\!\$\, \{0,1\}^{\ell_1+\ldots+\ell_n+r\cdot n}; inp \leftarrow \mathsf{F}(\mathsf{pt}^*, r^*)]\big| \\
&\leq 2q_{ow2h}\sqrt{\Pr[\mathsf{pt}^* \leftarrow \mathsf{Ext}^{\mathcal{A}', |G\rangle}(inp) | (\mathsf{pt}^*, r^*) \leftarrow\!\$\, \{0,1\}^{\ell_1+\ldots+\ell_n+r\cdot n}; inp \leftarrow \mathsf{F}(\mathsf{pt}^*, r^*)}
\end{aligned}
$$

where Ext is the extractor defined in Figure 19 and q_{ow2h} is the number of queries made by $\mathcal{A}'$ to G, which is q_G to answer $\mathcal{A}$'s queries and q_P to simulate the plaintext-checking oracle (i.e. one can compute the coins $(G_j(\mathsf{pt}_1, \ldots, \mathsf{pt}_n))_{j\in[n]}$ with one quantum query to G). Thus, $q_{ow2h} = (q_P + q_G)$. Now, the probability that the extractor outputs pt^* is precisely the probability that the OW-CPA property of all underlying PKE_i is broken. We provide in Figure 23 an adversary $\mathcal{B}_j$ that breaks the OW-CPA security of any PKE_j given $\mathsf{Ext}^{\mathcal{A}'}$. Thus, we have

$$
\big|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]\big| \leq 2(q_P + q_G)\sqrt{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_j}(\mathcal{B}_j)}
$$

for any $j \in [n]$. Finally, $\Pr[\Gamma^2 \Rightarrow 1]$ is the probability to win the OW-CPA game against any underlying PKE_j. We provide the given adversary $\mathcal{C}_j$ that breaks PKE_j in Figure 23. Hence, $\Pr[\Gamma^2 \Rightarrow 1] \leq \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_j}(\mathcal{C}_j) \leq \sqrt{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_j}(\mathcal{C}_j)}$. Collecting the bounds and folding adversaries hold the result. □

Corollary 2 (Informal). *The $\mathsf{QU}_m \circ \mathsf{T}^n_{\|}$ combiner takes n PKEs and outputs an IND-CCA KEM which is secure as long as one of the underlying PKEs is OW-CPA secure.*

Proof. This is simply a consequence of Theorem 4.5 from [14] and Theorem 9.

B Extractable Random Functions (ERF), Indistinguishable unless Queried (IUQ)

B.1 ERF

Definition 7 (Extractor). *Let g be a random function defined using a random oracle G. An extractor Ext_g for a function g is a ppt deterministic function that*

$\Gamma^i(\mathcal{A})$

$\mathsf{pp} \leftarrow\$\, \mathsf{setup}(1^\lambda)$
$((\mathsf{pk}_1, \ldots, \mathsf{pk}_n), (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)) \leftarrow\$\, \mathsf{gen}(\mathsf{pp})$
$(\mathsf{pt}_1^*, \ldots, \mathsf{pt}_n^*) \leftarrow\$\, \{0,1\}^{\ell_1+\ldots+\ell_n}$
$\mathsf{ct}^* \leftarrow \mathsf{enc}(\mathsf{pk}, (\mathsf{pt}_1^*, \ldots, \mathsf{pt}_n^*))$
for $i \in [n]$: // Γ^2
  $(r_1^*, \ldots, r_n^*) \leftarrow\$\, \{0,1\}^{rn}$ // Γ^2
  $\mathsf{ct}_i^* \leftarrow \mathsf{enc}_i(\mathsf{pk}_i, \mathsf{pt}_i^*; r_i^*)$ // Γ^2
if $\mathsf{dec}(\mathsf{sk}^*, \mathsf{ct}^*) \neq (\mathsf{pt}_1^*, \ldots, \mathsf{pt}_n^*)$:
  abort
$\mathsf{pt}' \leftarrow \mathcal{A}^{|G_1,\ldots,G_n\rangle, \mathcal{O}_0^{\mathsf{PCO}}}(\mathsf{pk}, \mathsf{ct}^*)$ // Γ^0
$\mathsf{pt}' \leftarrow \mathcal{A}^{|G_1,\ldots,G_n\rangle, \mathcal{O}_1^{\mathsf{PCO}}}(\mathsf{pk}, \mathsf{ct}^*)$ // Γ^1-Γ^2
return $1_{\mathsf{pt}'=(\mathsf{pt}_1^*, \ldots, \mathsf{pt}_n^*)}$

Oracle $\mathcal{O}_0^{\mathsf{PCO}}((\mathsf{pt}_1, \ldots, \mathsf{pt}_n), (\mathsf{ct}_1, \ldots, \mathsf{ct}_n))$

$(\mathsf{pt}_1', \ldots, \mathsf{pt}_n') \leftarrow \mathsf{dec}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$
return $1_{(\mathsf{pt}_1, \ldots, \mathsf{pt}_n)=(\mathsf{pt}_1', \ldots, \mathsf{pt}_n')}$

Oracle $\mathcal{O}_1^{\mathsf{PCO}}((\mathsf{pt}_1, \ldots, \mathsf{pt}_n), (\mathsf{ct}_1, \ldots, \mathsf{ct}_n))$

for $j \in [n]$:
  $r_j \leftarrow G_j(\mathsf{pt}_1, \ldots, \mathsf{pt}_n)$
  if $\mathsf{enc}_j(\mathsf{pk}_j, \mathsf{pt}_j; r_j) \neq \mathsf{ct}_j$:
    return 0
return 1

$G_j(m)$

if $\exists g$ s.t. $(m, g) \in \mathcal{L}_{G_j}$:
  return g
$g \leftarrow\$\, \{0,1\}^r$
$\mathcal{L}_{G_j} \leftarrow \mathcal{L}_{G_j} \cup \{(m, g)\}$
return g

Fig. 21: Sequence of games for the proof of Theorem 9.

takes a set of tuples $\mathcal{L}_G = \{(x_i, h_i)\}_{i\in[q_G]}$ defining the event $\{\wedge_{i\in[q_G]} G(x_i) = h_i\}$ and that outputs a set of tuples $\mathcal{E} = \{((i^j, \sigma_1^j, \sigma_2^j), g^j)\}_{j\in[q_E]}$ s.t.

1. *(correctness)* $\Pr[g(i^j, \sigma_1^j, \sigma_2^j) = g^j | \mathcal{L}_G] = 1, \forall j \in [q_E]$.
2. *(initial emptiness)* $\mathsf{Ext}_g(\emptyset) = \emptyset$.
3. *(increasing)* $\mathcal{L}_G \subseteq \mathcal{L}'_G \Rightarrow \mathsf{Ext}_g(\mathcal{L}_G) \subseteq \mathsf{Ext}_g(\mathcal{L}'_G)$.
4. *(initial queries) Let $\mathcal{L}_G^*$ be the set of queries/responses made when computing $g(1, \sigma_1, \sigma_2)$ and $g(2, \sigma_1, \sigma_2)$. Then,*

$$\mathsf{Ext}_g(\mathcal{L}_G^*) = \{((1, \sigma_1, \sigma_2), g(1, \sigma_1, \sigma_2)), ((2, \sigma_1, \sigma_2), g(2, \sigma_1, \sigma_2))\} .$$

   That is, one call to the function $g(i, \sigma_1, \sigma_2)$ (for different i) does not give away any information on other values of g.

Note that the number of tuples output by the extractor q_E is a function of q_G, that is the number of queries made to the RO G. In addition, we define q_E^1 as the maximum number of tuples of the form (i, σ_1, σ_2) with a fixed σ_1 (or σ_2) output by the extractor.

Now, we can define the notion of extractable random functions.

Definition 8 (Extractable Random Function (ERF)). *Let $g : \{0,1\}^* \mapsto \{0,1\}^n$ be a (random) function defined using a random oracle G. Let $\mathcal{J}_{\sigma_1,\sigma_2} = \{((i, \sigma_1', \sigma_2'), g(i, \sigma_1', \sigma_2')) : \sigma_1' \neq \sigma_1, \sigma_2' \neq \sigma_2\}$ be the set of input/output tuples of g for values σ_1' and σ_2' different from σ_1 and σ_2, where each tuple $(x, y) \in \mathcal{J}_{\sigma_1,\sigma_2}$*

$\mathsf{F}(\mathsf{pt}^*, r^*)$

parse $(r_1^*, \ldots, r_n^*) \leftarrow r^*$
parse $(\mathsf{pt}_1^*, \ldots, \mathsf{pt}_n^*) \leftarrow \mathsf{pt}^*$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{gen}$
for $i \in [n]$:
  $\mathsf{ct}_i^* \leftarrow \mathsf{enc}_i(\mathsf{pk}_i, \mathsf{pt}_i^*; r_i^*)$
return $(\mathsf{pk}, \mathsf{ct}^*)$

Fig. 22: Function F for applying the AOW2H Lemma in the proof of Theorem 9.

$\mathcal{B}_1^{\mathsf{Ext}^{\mathcal{A}'}, G}(\mathsf{pk}_j, \mathsf{ct}_j^*)$

for $i \in [n], i \neq j$:
  $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\$ \mathsf{gen}_i$
  $\mathsf{pt}_i^* \leftarrow\$ \{0,1\}^{\ell_i}$
  $\mathsf{ct}_i^* \leftarrow \mathsf{enc}_i(\mathsf{pk}_i, \mathsf{pt}_i^*)$
$\mathsf{pt}' \leftarrow \mathsf{Ext}^{\mathcal{A}'}((\mathsf{pk}_1, \ldots, \mathsf{pk}_n), (\mathsf{ct}_1^*, \ldots, \mathsf{ct}_n^*))$
return pt'_j

$\mathcal{C}_1'^{\mathcal{A}, G}(\mathsf{pk}_j, \mathsf{ct}_j^*)$

for $i \in [n], i \neq j$:
  $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\$ \mathsf{gen}_i$
  $\mathsf{pt}_i^* \leftarrow\$ \{0,1\}^{\ell_i}$
  $\mathsf{ct}_i^* \leftarrow \mathsf{enc}_i(\mathsf{pk}_i, \mathsf{pt}_i^*)$
$\mathsf{pt}' \leftarrow \mathcal{A}^{\mathcal{O}_1^{\mathsf{PCO}}}((\mathsf{pk}_1, \ldots, \mathsf{pk}_n), (\mathsf{ct}_1^*, \ldots, \mathsf{ct}_n^*))$
return pt'_j

Fig. 23: OW-CPA adversaries for the proof of Theorem 9.

defines the event $\{g(x) = y\}$*. Then,* g *is an extractable random function (ERF) if there exists an extractor* Ext_g *s.t. for any* $i, \sigma_1, \sigma_2, y, \mathcal{L}_G$ *and* $\mathcal{J}' \subseteq \mathcal{J}_{\sigma_1,\sigma_2}$ *s.t.* $\Pr[\mathcal{J}', \mathcal{L}_G] > 0$*,*

$$\Pr[g(i,\sigma_1,\sigma_2) = y | \mathcal{L}_G, \mathcal{J}'] = \begin{cases} \frac{1}{2^n}, & \text{if } ((i,\sigma_1,\sigma_2), y) \notin \mathsf{Ext}_g(\mathcal{L}_G) \\ 1, & \text{if } ((i,\sigma_1,\sigma_2), y) \in \mathsf{Ext}_g(\mathcal{L}_G) \\ 0, & \text{if } \exists y' \neq y \text{ s.t. } ((i,\sigma_1,\sigma_2), y) \in \mathsf{Ext}_g(\mathcal{L}_G) \end{cases}$$

In short, this notion captures the fact that either $g(i, \sigma_1, \sigma_2)$ is uniformly distributed, or the extractor can compute it based on the queries made to G. In addition, we require that there is no correlation between different values of g when both inputs are different. Finally, we stress that when a party computes $g(i, \sigma_1, \sigma_2)$, the value of g becomes deterministic. In other words, if we let $\mathcal{L}_G$ be the set of corresponding queries/responses used to compute $g(i, \sigma_1, \sigma_2)$, the list $\mathsf{Ext}_g(\mathcal{L}_G \cup \mathcal{L}'_G)$ will contain $g(i, \sigma_1, \sigma_2)$, for any $\mathcal{L}'_G$.

ROs are* ERF *functions (Example). As an example, we show that ROs are ERF. More precisely, let $g(i, \sigma_1, \sigma_2) = G(i, \sigma_1, \sigma_2)$ as in the $\mathsf{UT}_{\|}$ combiner. Then, we define the extractor Ext_g as a function that takes all tuples of the form $((i, \sigma_1, \sigma_2), h) \in \mathcal{L}_G$ and outputs them. Clearly, if the extractor does not output a given value (i, σ_1, σ_2), then it was not queried to the RO and it is indistinguishable from a uniform value, as requested. Also, by the property of ROs,

$\underline{\mathrm{IUQ}\text{-}^{b}_{h,\mathcal{H},\mathsf{Ext}_h}(\mathcal{A})}$

$\mathsf{chal}^1 \leftarrow \mathbf{false}; \mathsf{chal}^2 \leftarrow \mathbf{false}$
$(\sigma_1, \sigma_2) \leftarrow\$\, \mathcal{M}_1 \times \mathcal{M}_2$
$h_0 \leftarrow\$\, \{0,1\}^n$
$h_1 \leftarrow h(\sigma_1, \sigma_2)$
$b' \leftarrow \mathcal{A}^H(\sigma_1, \sigma_2, h_b)$
return b'

$\underline{H(m)}$

1 : **if** $\exists x$ s.t. $(m,x) \in \mathcal{L}_H$:
2 : **return** x
3 : $x \leftarrow\$\, \{0,1\}^n$
4 : $\mathcal{L}_H \leftarrow \mathcal{L}_H \cup \{(m,x)\}$
5 : **if** $\sigma_1 \in \mathsf{Ext}_h(1, \sigma_2, \mathcal{L}_H)$: $\mathsf{chal}^1 \leftarrow \mathbf{true}$
6 : **if** $\sigma_2 \in \mathsf{Ext}_h(2, \sigma_1, \mathcal{L}_H)$: $\mathsf{chal}^2 \leftarrow \mathbf{true}$
7 : **if** chal^1 **and** chal^2 : **abort**
8 : **return** x

Fig. 24: IUQ game.

a value $G(i, \sigma_1, \sigma_2)$ is mutually independent from any set of values $G(i', \sigma_1', \sigma_2')$ with $\sigma_1 \neq \sigma_1'$ and $\sigma_2 \neq \sigma_2'$. Note also that the maximum number of tuples output by the extractor q_E is upper bounded by q_G.

B.2 IUQ functions

Now we define a weaker assumption than ERF for the hash function h that derives the key in the encapsulation/decapsulation procedures. Indeed, we notice that the only property we need from this function is to look indistinguishable unless one can recover both challenge seeds. This must hold even if the adversary can choose one of the seeds, as in the reduction an adversary attacking PKE_1 can pick σ_2^*. We call such property Indistinguishability unless Queried (IUQ) and define it as follows.

Definition 9 (IUQ functions). *Let $h(\sigma_1, \sigma_2)$ be a (random) function based on a random oracle H with $\sigma_1 \in \mathcal{M}_1$ and $\sigma_2 \in \mathcal{M}_2$. We consider the* IUQ *game defined in Figure 24, where the RO H is defined as shown in the game. Then, if there exists a ppt function $\mathsf{Ext}_h(i, \sigma_i, \mathcal{L}_H)$ s.t. for any ppt adversary $\mathcal{A}$*

$$\mathsf{Adv}^{\mathrm{iuq}}_{h,H,\mathsf{Ext}_h}(\mathcal{A}) = \left|\Pr[\mathrm{IUQ}^1_{h,H,\mathsf{Ext}_h}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{IUQ}^0_{h,H,\mathsf{Ext}_h}(\mathcal{A}) \Rightarrow 1]\right| = 0$$

we say h is IUQ *(Indistinguishable from Uniform unless Queried).*

While looking cumbersome, this definition simply generalizes what we want from the functions that derives the key. Indeed, in the IUQ game, we ask the adversary to distinguish between a uniformly distributed value and $h(\sigma_1, \sigma_2)$ for some random (σ_1, σ_2). However, if there exists some extractor (or parsing function) Ext_h that can recover (σ_1, σ_2) by observing the queries to the random oracles, the game aborts. That captures the fact that either the adversary cannot distinguish, or one can recover the challenge seeds (or plaintexts). Note that the function Ext_h takes the index of the seeds it must recover and the other seed to capture the fact that in a reduction attacking the one-wayness of PKE_1, the adversary can pick σ_2 (and the other way around).

Finally, note that we can ask for perfect indistinguishably (i.e. the probability of distinguishing is 0) because we are still in the ROM. Indeed, we recall that for a RO H, since the value $H(m)$ is taken uniformly at random, the distribution of $H(m)$ is uniform as long as m is not queried to H.

In summary, the definition simply formalizes that one cannot distinguish between uniform values and $h(\sigma_1, \sigma_2)$, unless one can recover (σ_1, σ_2) from the list of queries made to the random oracles.

$\mathsf{H}(\sigma_1 \oplus \sigma_2)$ ***is* IUQ *(Example).*** As an example of a IUQ function, one can consider $h(\sigma_1, \sigma_2) := H(\sigma_1) \oplus H(\sigma_2)$, the function used in the $\mathsf{UT}_{\|}$ combiner. As an extractor, we define $\mathsf{Ext}_h(i, \sigma, \mathcal{L}_H)$ as the function that goes through all tuples $(m, h) \in \mathcal{L}_H$ and outputs m. Now, unless σ_1 and σ_2 are queried, the adversary cannot distinguish a random value from $h(\sigma_1, \sigma_2)$. But if both values are queried, the IUQ game will abort because both lists output by the extractor $\mathsf{Ext}_h(1, \sigma_2, \mathcal{L}_H)$ and $\mathsf{Ext}_h(2, \sigma_1, \mathcal{L}_H)$ will contain σ_1 and σ_2, respectively. In this case, the advantage of IUQ adversary is 0.

B.3 IUQ and ERF in $\mathsf{UT}_{\|}$

Now, based on the IUQ and ERF definition, we prove the following theorem, which states that the $\mathsf{UT}_{\|}$ combiner is still robust if G and H are replaced by ERF and IUQ functions, respectively.

Theorem 10 ($\mathsf{UT}_{\|}$ and ERF/IUQ). *Let $h(\sigma_1, \sigma_2)$ and $g(i, \sigma_1, \sigma_2)$ be a* IUQ*, resp. ERF function, and H, G be the ROs h and g are based on, respectively. In addition, let $q_E(|\mathcal{L}_G|)$, $q_{E_h}(|\mathcal{L}_H|)$ be the maximum number of tuples output by $\mathsf{Ext}_g(\mathcal{L}_G)$, $\mathsf{Ext}_h(\mathcal{L}_H)$, respectively (they are a function of the length of the input). We also let $q_E^1(|\mathcal{L}_G|)$ be the maximal number of tuples with a fixed σ output by $\mathsf{Ext}_g(\mathcal{L}_G)$ (see Definition 7). Finally, let KEM be the hybrid KEM built on top of two OW-CPA PKEs using the $\mathsf{UT}_{\|}$ combiner, where the deterministic coins for encrypting the seed σ_i are computed as $g(i, \sigma_1, \sigma_2)$ instead of $G(i, \sigma_1, \sigma_2)$ and the key is computed as $h(\sigma_1, \sigma_2)$ instead of $H(\sigma_1 \oplus \sigma_2)$. Then, for all ppt IND-CCA adversary $\mathcal{A}$ making at most q_G, q_H and q_D queries to the oracles G,H and $\mathcal{O}^{\mathsf{Dec}}$, respectively, there exists adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\begin{aligned}\mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{PKE}}(\mathcal{A}) \leq & q_E(q_g \cdot 2(q_D + 1) + q_G) \cdot \max\{\delta_1, \delta_2\} \\ & + (q_D + q_E^1(q_G)) \cdot (2^{-\gamma_1} + 2^{-\gamma_2}) \\ & + (q_E^1(q_G) + q_{E_h}(q_H)) \cdot \min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathcal{B}_1), \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_2}(\mathcal{B}_2)\}\end{aligned}$$

where q_g is the number of queries to G needed to evaluate g. All run in about the same time as $\mathcal{A}$ and make the same number of queries.

Proof. The proof is very similar to the proof of security of the $\mathsf{UT}_{\|}$ transform. The only difference is that we use the output of the extractors associated with the ERF/IUQ properties of g/h instead of the list of queries $\mathcal{L}_G, \mathcal{L}_H$. In particular, we argue that if some value is not in these extracted lists, it is uniformly distributed.

Let Ext_g and Ext_h be the functions s.t. g and h are ERF and IUQ, respectively. We also assume that the key space is $\{0,1\}^n$ for some n. We give the sequence of games in Figure 25.

Game Γ^1: In this game, we enforce the correctness of all ciphertexts that can be computed using the function g. In particular, we abort if the challenge ciphertexts break the correctness property or if any (i, σ_1, σ_2) output by Ext_g is s.t. $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g(i, \sigma_1, \sigma_2))$ breaks the correctness property. Let $\mathcal{L}_G$ collect all tuples of query/responses made throughout the game by the adversary and the game itself, and $\mathcal{L}_G^k$ its state after the k-th query to G is made. Then, we can define the set of new tuples output by the extractor at query k as $\mathcal{T}^k = \mathsf{Ext}_g(\mathcal{L}_G^k) \setminus \mathsf{Ext}_g(\mathcal{L}_G^{k-1})$. Hence, when submitting the k-th query m to G, the probability a tuple in the corresponding $\mathcal{T}^k = \mathsf{Ext}_g(\mathcal{L}_G^{k-1} \cup (m, G(m))) \setminus \mathsf{Ext}_g(\mathcal{L}_G^{k-1})$ contains a plaintext that breaks the correctness is

$$
\begin{aligned}
&\Pr\Big[\bigvee_{((i,\sigma_1,\sigma_2),g(i,\sigma_1,\sigma_2))\in\mathcal{T}^k} \mathsf{dec}_i(\mathsf{sk}_i, \mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g(i,\sigma_1,\sigma_2))) \neq \sigma_i | \mathcal{L}_G^{k-1}\Big] \\
&\leq \sum_{((i,\sigma_1,\sigma_2),g(i,\sigma_1,\sigma_2))\in\mathcal{T}^k} \Pr[\mathsf{dec}_i(\mathsf{sk}_i, \mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g(i,\sigma_1,\sigma_2))) \neq \sigma_i | \mathcal{L}_G^{k-1}] \\
&= \sum_{((i,\sigma_1,\sigma_2),g(i,\sigma_1,\sigma_2))\in\mathcal{T}^k} \Pr[\mathsf{dec}_i(\mathsf{sk}_i, \mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; \mathsf{coins})) \neq \sigma_i : \mathsf{coins} \leftarrow\!\!\$\, \{0,1\}^n] \\
&\leq |\mathcal{T}^k| \cdot \max\{\delta_1, \delta_2\}
\end{aligned}
$$

for any query m. The equality follows from the definition of extractable random functions and the last inequality from the δ_i correctness of PKE_i. Then, by a union bound, the probability a correctness error happens for any of the q_E tuples output by Ext_g is upper bounded by $q_E \cdot \max\{\delta_1, \delta_2\}$. Note that q_E is a function of the total number of queries submitted to G, which is $q_g \cdot 2(q_D + 1) + q_G$ in this case, where q_g is the number of queries made to G at each evaluation of g (i.e. in total 2 calls to g for the challenge ciphertexts and for each decapsulation query and q_G queries made by the adversary). Hence, we have

$$
|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]| \leq q_E \cdot \max\{\delta_1, \delta_2\} \ .
$$

Game Γ^2: In this game, we consider $\mathcal{L}_G$, which is the transcript of the queries made to G by any party (i.e. game or adversary). We enforce that at no point the extractor $\mathsf{Ext}_g(\mathcal{L}_G)$ contains a tuple $((1, \sigma_1^*, \sigma_2), g_1')$ with $\sigma_2 \neq \sigma_2^*$ or $((2, \sigma_1, \sigma_2^*), g_2')$ with $\sigma_1 \neq \sigma_1^*$ s.t. $\mathsf{enc}_1(\mathsf{pk}_1, \sigma_1^*; g_1') = \mathsf{ct}_1^*$ or $\mathsf{enc}_2(\mathsf{pk}_2, \sigma_2^*; g_2') = \mathsf{ct}_2^*$, respectively. We proceed as in the previous game and let $\mathcal{T}^k = \mathsf{Ext}_g(\mathcal{L}_G^k) \setminus \mathsf{Ext}_g(\mathcal{L}_G^{k-1})$ where $\mathcal{L}_G^k$ is the state of $\mathcal{L}_G^k$ after the k-th query to G. Finally, let $\mathcal{L}_G^0$ be the state of $\mathcal{L}_G$ after computing the challenge ciphertexts. Then, when submitting the k-th ($k \geq 2$) query m to G, the probability a tuple in $\mathcal{T}^k$ breaks

the first condition is

$$\begin{aligned}
&\Pr[\bigvee_{((1,\sigma_1^*,\sigma_2),g(1,\sigma_1^*,\sigma_2))\in\mathcal{T}^k} \mathsf{enc}_1(\mathsf{pk}_1,\sigma_1^*;g(1,\sigma_1^*,\sigma_2))) = \mathsf{ct}_1^* | \mathcal{L}_G^{k-1}] \\
&\leq \sum_{((1,\sigma_1^*,\sigma_2),g(1,\sigma_1^*,\sigma_2))\in\mathcal{T}^k} \Pr[\mathsf{enc}_1(\mathsf{pk}_1,\sigma_1^*;g(1,\sigma_1^*,\sigma_2))) = \mathsf{ct}_1^* | \mathcal{L}_G^{k-1}] \\
&= \sum_{((1,\sigma_1^*,\sigma_2),g(1,\sigma_1^*,\sigma_2))\in\mathcal{T}^k} \Pr[\mathsf{enc}_1(\mathsf{pk}_1,\sigma_1^*;\mathsf{coins})) = \mathsf{ct}_1^* : \mathsf{coins} \leftarrow\!\!\$\, \{0,1\}^n] \\
&\leq \left|\{((1,\sigma_1^*,\sigma_2),g) \in \mathcal{T}^k\}\right| \cdot 2^{-\gamma_1}
\end{aligned}$$

for any m, where the equality holds by the definition of ERF and the fact that by definition a tuple in $\mathcal{T}^k$ is not in $\mathcal{L}_G^{k-1}$. Now, the equation holds for $k = 1$ as well by the last property of extractors (i.e. $(1,\sigma_1^*,\sigma_2) \notin \mathcal{L}_G^0$ for any $\sigma_2 \neq \sigma_2^*$). Then, it is similar for the second type of failure and the probability it happens for any of the q_E tuples in $\mathsf{Ext}(\mathcal{L}_g)$ is upper bounded by $q_E^1 \cdot (2^{-\gamma_1} + 2^{-\gamma_2})$, where q_E^1 is the maximum number of tuples $g(i,\sigma_1,\sigma_2)$ output by the extractor for a fixed value σ_1 or σ_2. Thus, we have

$$|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]| \leq q_E^1 \cdot (2^{-\gamma_1} + 2^{-\gamma_2}) \ .$$

We also prove the following proposition.

Proposition 2. *Let $\mathsf{ct}_i \neq \mathsf{ct}_i^*$. In game Γ^2, if $\mathcal{A}$ submits $(\mathsf{ct}_1, \mathsf{ct}_2^*)$ or $(\mathsf{ct}_1^*, \mathsf{ct}_2)$ to the decapsulation oracle, the latter either returns $\perp$ or the game aborts.*

Proof. We assume w.l.o.g. that the adversary submits $(\mathsf{ct}_1, \mathsf{ct}_2^*)$. Let $\sigma_1' := \mathsf{dec}_1(\mathsf{sk}_1, \mathsf{ct}_1)$. By the perfect correctness of ct_2^*, $\mathsf{dec}_2(\mathsf{sk}_2, \mathsf{ct}_2^*) = \sigma_2^*$. If $\sigma_1' = \sigma_1^*$, then $\mathsf{enc}_1(\mathsf{pk}_1, \sigma_1'; g(1,\sigma_1',\sigma_2^*)) = \mathsf{ct}_1^* \neq \mathsf{ct}_1$ and the oracle replies $\perp$. Otherwise, in the re-encryption check, the game will compute $g(1,\sigma_1',\sigma_2^*)$ with $\sigma_1' \neq \sigma_1^*$ and thus the extractor will output $((1,\sigma_1',\sigma_2^*), g(1,\sigma_1',\sigma_2^*))$ at some point. By the abort condition in game Γ^2, either $\mathsf{enc}_1(\mathsf{pk}_1, \sigma_1^*; g(1,\sigma_1',\sigma_2^*)) = \mathsf{ct}_1^*$ and the game aborts, or $\mathsf{enc}_1(\mathsf{pk}_1, \sigma_1^*; g(1,\sigma_1',\sigma_2^*)) \neq \mathsf{ct}_1^*$ and the decapsulation oracle outputs $\perp$. □

Game Γ^3: We make nearly the exact same modifications as in game Γ^2 in the proof of Theorem 5. That is, we check whether there exist both $((1,\sigma_1,\sigma_2), g_1)$ and $((2,\sigma_1,\sigma_2), g_2)$ in $\mathsf{Ext}_g(\mathcal{L}_\mathcal{A})$ s.t. $\mathsf{enc}_i(\sigma_i; g_i) = \mathsf{ct}_i$. If this is the case, (let's call this event found) we return $h(\sigma_1,\sigma_2)$, otherwise we return the key K output by the decapsulation function. In other words, this game is the same as Γ^2 of the proof of Theorem 5 except we check for plaintexts in $\mathsf{Ext}_g(\mathcal{L}_\mathcal{A})$ instead of $\mathcal{L}_\mathcal{A}$. Now, if found occurs, we return the same key as in game Γ^1. Indeed, by the perfect correctness of the tuples in $\mathsf{Ext}_g(\mathcal{L}_\mathcal{A}) \subseteq \mathsf{Ext}_g(\mathcal{L}_G)$ enforced in game Γ^1, if we find (σ_1,σ_2) s.t. $\mathsf{enc}_i(\sigma_i; g(i,\sigma_1,\sigma_2)) = \mathsf{ct}_i$, then $\mathsf{dec}_i(\mathsf{sk}_i, \mathsf{ct}_i) = \sigma_i$. Hence, we have, $\Pr[\Gamma^2 \Rightarrow 1] = \Pr[\Gamma^3 \Rightarrow 1]$.

Game Γ^4: We modify the previous game as follows. As before, this game

is the same as game Γ^3 of the proof of Theorem 5 except we replace $\mathcal{L}_{\mathcal{A}}$ by $\mathsf{Ext}_g(\mathcal{L}_{\mathcal{A}})$. In the decapsulation oracle, we simply return $\perp$ if found does not occur. Hence, game Γ^3 and Γ^4 differ iff the decapsulation oracle successfully decrypts ct but the extractor could not find neither $(1, \sigma_1', \sigma_2')$ or $(2, \sigma_1', \sigma_2')$, where $(\mathsf{enc}_1(\sigma_1'), \mathsf{enc}_2(\sigma_2')) = \mathsf{ct}$ (i.e. at least one tuple is not in $\mathsf{Ext}_g(\mathcal{L}_{\mathcal{A}})$). Now, the ciphertexts corresponding to the seeds in $\mathsf{Ext}_g(\mathcal{L}_{\mathcal{A}})$ are perfectly correct. Thus, this event is equivalent to the decapsulation oracle successfully (i.e. the re-encryption checks pass) recovering the seeds σ_1, σ_2 but either $(1, \sigma_1, \sigma_2)$ or $(2, \sigma_1, \sigma_2)$ or both were not recovered by the extractor. Let fail be this event and we prove the following lemma.

Lemma 6.

$$\Pr[\mathsf{fail}] \leq q_D \cdot (2^{-\gamma_1} + 2^{-\gamma_2}) \ .$$

Proof. The proof is nearly the same as the proof of Lemma 1. Let fail_k be the event that fail happens at the k-th decapsulation query and $p_k = \Pr[\mathsf{fail}_k]$. By an union bound, we have

$$\Pr[\mathsf{fail}] \leq \sum_{k=1}^{q_D} p_k \ .$$

Then, we consider an algorithm $\mathcal{B}_k$ defined in Figure 26, which is the same as the one defined in Figure 14 for Lemma 1, except the calls to G are replaced by invocations of g and the checks for values in $\mathcal{L}_G$ by checks in $\mathsf{Ext}_g(\mathcal{L}_G)$. This adversary simulates perfectly the view of $\mathcal{A}$ in game Γ^4 until the k-th query. In particular, for each decapsulation query $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$, it checks whether there exists both $((1, \sigma_1, \sigma_2), g_1)$ and $((2, \sigma_1, \sigma_2), g_2)$ in $\mathsf{Ext}(\mathcal{L}_G)$ s.t. $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g_i)$ for $i \in [2]$. We call this condition cond and if it is fulfilled $\mathcal{B}_k$ outputs $h(\sigma_1, \sigma_2)$, otherwise it outputs $\perp$.

In the k-th decapsulation query, if cond is fulfilled it aborts. Otherwise, it sets i s.t. there is no $((i, \sigma_1, \sigma_2), g_i) \in \mathsf{Ext}(\mathcal{L}_G)$ s.t. $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g_i) = \mathsf{ct}_i$. Next, it decrypts ct_1 and ct_2 to both σ_1' and σ_2'. By the definition of i and the perfect correctness of the values σ_i in $\mathsf{Ext}_g(\mathcal{L}_G)$, we have that $(i, \sigma_1', \sigma_2') \notin \mathsf{Ext}_g(\mathcal{L}_G)$. In addition, by the perfect correctness of the challenge ciphertexts and Proposition 2 we have $\sigma_1' \neq \sigma_1^*$ and $\sigma_2' \neq \sigma_2^*$. Finally, $\mathcal{B}_k$ computes $g_i' \leftarrow g(i, \sigma_1', \sigma_2')$ and outputs 1 iff $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g_i') = \mathsf{ct}_i$. Now, as $g_i' = g(i, \sigma_1', \sigma_2')$ is not in $\mathcal{L}_G$ and $\sigma_1' \neq \sigma_1^*, \sigma_2' \neq \sigma_2^*$, it is sampled uniformly at random. More precisely, if we fix all random coins but the ones used by G, only the responses of G and the challenge ciphertexts (which only depend on $g(i, \sigma_1^*, \sigma_2^*)$) are random. Thus, we have

$$\begin{aligned}
&\Pr[\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i'; g(i, \sigma_1', \sigma_2')) = \mathsf{ct}_i | \mathcal{L}_G, \mathsf{ct}^*] \\
&= \Pr[\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i'; \mathsf{coins}) = \mathsf{ct}_i : \mathsf{coins} \leftarrow\!\!\$\, \{0,1\}^n] \leq 2^{-\gamma_i}
\end{aligned}$$

by the γ_i-spreadness of PKE_i and the definition of ERF. In the worst case, the check is performed for both $i = 1$ and $i = 2$ and thus $\Pr[\mathcal{B}_k(\mathcal{A}) \Rightarrow 1] \leq 2^{-\gamma_1} + 2^{-\gamma_2}$. Now, we simply observe that if fail_k occurs, then $\mathcal{B}_k$ perfectly simulates the decapsulation oracle in Γ^3 and Γ^4 in the first $k-1$ queries and it

will output 1 by the definition of fail_k. Thus,

$$p_k \leq \Pr[\mathcal{B}_k(\mathcal{A}) \Rightarrow 1] \leq 2^{-\gamma_1} + 2^{-\gamma_2} \ .$$

Taking the union bound on the p_k holds the result. □

By the previous Lemma, we have

$$|\Pr[\Gamma^3 \Rightarrow 1] - \Pr[\Gamma^4 \Rightarrow 1]| \leq q_D \cdot (2^{-\gamma_1} + 2^{-\gamma_2}) \ .$$

Game Γ^5: We replace the deterministic coins used in the computation of the challenge ciphertexts by random coins and we abort if the extractor Ext_g outputs a tuple $(i, \sigma_1^*, \sigma_2^*)$ on an input m to the RO G. Let's call this event chal_g. In addition, we replace the key by a random one when $b = 0$ and we raise a flag chal_h when the extractor Ext_h can recover **both** σ_1^* and σ_2^*.

One can see that as long as $\mathsf{chal}_g \cup \mathsf{chal}_h$ does not occur, the adversary cannot distinguish between the coins $g(i, \sigma_1, \sigma_2)$ and random coins, and between a real and random key. Indeed, it means the extractors $\mathsf{Ext}_g, \mathsf{Ext}_h$ cannot recover the values $g(i, \sigma_1^*, \sigma_2^*)$ and σ_1^*, σ_2^*, respectively. By the definition of ERF and IUQ this means that $g(i, \sigma_1^*, \sigma_2^*)$ is uniformly distributed and a random key is indistinguishable from $h(\sigma_1^*, \sigma_2^*)$. Then, if $\mathsf{chal}_g \cup \mathsf{chal}_h$ happens, the adversary can recover the challenge seeds and break the one-wayness properties of both ciphertexts by inspecting the values output by both extractors. We give the OW-CPA adversary $\mathcal{B}_1$ breaking PKE_1 in Figure 27, which wins whenever $\mathsf{chal}_g \cup \mathsf{chal}_h$ happens and it picked the correct extracted value. The adversary $\mathcal{B}_1$ picks the second seed σ_2^* at random and runs the adversary $\mathcal{A}$ with both challenge ciphertexts and a random key K, and it can simulate the decapsulation oracle perfectly as the latter does not use the secret key. Then, if $\mathsf{chal}_g \cup \mathsf{chal}_h$ happens, clearly $(i, \sigma_1^*, \sigma_2^*)$ or σ_1^* will be in the output of the extractors until the end of the game. Thus, $\mathcal{B}_1$ can recover σ_1

1. by looking for a tuple of the form $(i, \sigma_1, \sigma_2 *)$ for some i, σ_1 in the output of Ext_g. There are at most q_E^1 such tuples, where we recall that q_E^1 is the maximum number of tuples of the form (i, σ_1, σ_2) for a fixed σ_1 or σ_2 output by the extractor.
2. by outputting a random value σ_1 in the output of $\mathsf{Ext}_h(1, \sigma_2^*, \mathcal{L}_H)$. There are at most q_{E_h} of these values.

Hence, the probability that $\mathcal{B}_1$ wins is at least $\frac{1}{q_E^1 + q_{E_h}} \Pr[\mathsf{chal}_g \cup \mathsf{chal}_h]$, as it needs to pick the correct tuple/value. On the other hand, as long as $\mathsf{chal}_g \cup \mathsf{chal}_h$ does not happen, both games are indistinguishable. Hence,

$$|\Pr[\Gamma^4 \Rightarrow 1] - \Pr[\Gamma^5 \Rightarrow 1]| \leq (q_E^1 + q_{E_h}) \cdot \min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathcal{B}_1), \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_2}(\mathcal{B}_2)\} \quad .$$

Now, since both K_0 and K_1 are uniformly distributed in Γ^5, $\Pr[G^5 \Rightarrow 1] = \frac{1}{2}$. Collecting the probabilities and folding similar adversaries into one hold the result. Hence, when h is IUQ and g is ERF, Theorem 5 still holds, but with a bound that might be less tight. □

$\Gamma^i(\mathcal{A})$

$\mathsf{pp} \leftarrow\$\, \mathsf{setup}(1^\lambda);\ (\mathsf{pk}, \mathsf{sk}) \leftarrow\$\, \mathsf{gen}(\mathsf{pp})$
$(\sigma_1^*, \sigma_2^*) \leftarrow\$\, \mathcal{M}_1 \times \mathcal{M}_2$
$\mathsf{coins}_1 \leftarrow g(1, \sigma_1^*, \sigma_2^*)$ // Γ^0-Γ^4
$\mathsf{coins}_2 \leftarrow g(2, \sigma_1^*, \sigma_2^*)$ // Γ^0-Γ^4
$\mathsf{coins}_1, \mathsf{coins}_2 \leftarrow\$\, \mathcal{R}^2$ // Γ^5
$\mathsf{ct}_1^* \leftarrow \mathsf{enc}_1(\mathsf{pk}_1, \sigma_1^*; \mathsf{coins}_1)$
$\mathsf{ct}_2^* \leftarrow \mathsf{enc}_2(\mathsf{pk}_2, \sigma_2^*; \mathsf{coins}_2)$
if $\exists i \in [2]$ s.t. $\mathsf{dec}_i(\mathsf{sk}_i, \mathsf{ct}_i^*) \neq \sigma_i^*$: **abort** // Γ^1-Γ^5
$b \leftarrow\$\, \{0,1\}$
$K_0 \leftarrow h(\sigma_1^*, \sigma_2^*)$ // Γ^0-Γ^4
$K_0 \leftarrow\$\, \mathcal{K}$ // Γ^5
$K_1 \leftarrow\$\, \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Dec}}}(\mathsf{pk}, (\mathsf{ct}_1^*, \mathsf{ct}_2^*), K_b)$
return $1_{b'=b}$

Oracle $\mathcal{O}^{\mathsf{Dec}}(\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2))$

$\mathsf{flag} \leftarrow$ **false**
if $\mathsf{ct} = \mathsf{ct}^*$ **then return** $\perp$
if $\exists((1, \sigma_1, \sigma_2), g_1) \in \mathsf{Ext}_g(\mathcal{L}_\mathcal{A})$
  s.t. $\mathsf{enc}_1(\mathsf{pk}_1, \sigma_1; g_1) = \mathsf{ct}_1$
  and $\exists((2, \sigma_1, \sigma_2), g_2) \in \mathsf{Ext}_g(\mathcal{L}_\mathcal{A})$
  s.t. $\mathsf{enc}_2(\mathsf{pk}_2, \sigma_2; g_2) = \mathsf{ct}_2$: // Γ^3-Γ^5
  return $h(\sigma_1, \sigma_2)$ // Γ^3-Γ^5
return $\perp$ // Γ^4-Γ^5
$K' \leftarrow \mathsf{decaps}(\mathsf{pp}, \mathsf{sk}, \mathsf{ct})$ // Γ^0-Γ^3
return K' // Γ^0-Γ^3

$H(m)$

if $\exists h$ s.t. $(m, h) \in \mathcal{L}_H$:
  return h
if $m = (\sigma_1^* \oplus \sigma_2^*)$: // Γ^5
  $\mathsf{chal}^2 =$ **true** // Γ^5
  abort // Γ^5
$h \leftarrow\$\, \{0,1\}^n$
$\mathcal{L}_H \leftarrow \mathcal{L}_H \cup \{(m, h)\}$
if $\sigma_1^* \in \mathsf{Ext}_h(1, \sigma_2^*, \mathcal{L}_H)$: // Γ^5
  $\mathsf{chal}_h^1 \leftarrow$ **true** // Γ^5
if $\sigma_2^* \in \mathsf{Ext}_h(2, \sigma_1^*, \mathcal{L}_H)$: // Γ^5
  $\mathsf{chal}_h^2 \leftarrow$ **true** // Γ^5
if chal_h^1 **and** chal_h^2 : // Γ^5
  abort // Γ^5
return h

$G(m)$

if $\exists g'$ s.t. $(m, g') \in \mathcal{L}_G$: $g \leftarrow g'$
else : $g \leftarrow\$\, \{0,1\}^n$
$\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m, g)\}$
for $((i, \sigma_1, \sigma_2), g) \in \mathsf{Ext}_g(\mathcal{L}_G)$: // Γ^1-Γ^5
  if $\mathsf{dec}_i(\mathsf{sk}_i, \mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g)) \neq \sigma_i$: // Γ^1-Γ^5
    abort // Γ^1-Γ^5
  if $\sigma_1 = \sigma_1^*$ **and** $\sigma_2 \neq \sigma_2^*$: // Γ^2-Γ^5
    if $\mathsf{enc}_1(\mathsf{pk}_1, \sigma_1; g) = \mathsf{ct}_1^*$: // Γ^2-Γ^5
      abort // Γ^2-Γ^5
  if $\sigma_2 = \sigma_2^*$ **and** $\sigma_1 \neq \sigma_1^*$: // Γ^2-Γ^5
    if $\mathsf{enc}_2(\mathsf{pk}_2, \sigma_2; g) = \mathsf{ct}_2^*$: Γ^2-Γ^5
      abort // Γ^2-Γ^5
if m queried by $\mathcal{A}$
  $\mathcal{L}_\mathcal{A} \leftarrow \mathcal{L}_\mathcal{A} \cup \{(m, g)\}$
if $((1, \sigma_1^*, \sigma_2^*), g) \in \mathsf{Ext}_g(\mathcal{L}_\mathcal{A})$
  or $((2, \sigma_1^*, \sigma_2^*), g) \in \mathsf{Ext}_g(\mathcal{L}_\mathcal{A})$: // Γ^5
  abort // Γ^5
return g

Fig. 25: OW-CPA adversaries for the proof of Theorem 10.

$\mathcal{B}_k(\mathcal{A})$

$\mathsf{pp} \leftarrow\$\, \mathsf{setup}(1^\lambda)$; $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$\, \mathsf{gen}(\mathsf{pp})$
$(\sigma_1^*, \sigma_2^*) \leftarrow\$\, \mathcal{M}_1 \times \mathcal{M}_2$
$\mathsf{coins}_1 \leftarrow g(1, \sigma_1^*, \sigma_2^*); \mathsf{coins}_2 \leftarrow g(2, \sigma_1^*, \sigma_2^*)$
$\mathsf{ct}_1^* \leftarrow \mathsf{enc}_1(\mathsf{pk}_1, \sigma_1^*; \mathsf{coins}_1)$
$\mathsf{ct}_2^* \leftarrow \mathsf{enc}_2(\mathsf{pk}_2, \sigma_2^*; \mathsf{coins}_2)$
if $\exists i \in [2]$ s.t. $\mathsf{dec}_i(\mathsf{sk}_i, \mathsf{ct}_i^*) \neq \sigma_i^*$: **abort**
$b \leftarrow\$\, \{0,1\}$
$K_0 \leftarrow h(\sigma_1^*, \sigma_2^*)$
$K_1 \leftarrow\$\, \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Dec}}}(\mathsf{pk}, (\mathsf{ct}_1^*, \mathsf{ct}_2^*), K_b)$
return $1_{b'=b}$

Oracle $\mathcal{O}^{\mathsf{Dec}}(\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2))$

if $\mathsf{ct} = \mathsf{ct}^*$ **then return** $\perp$
if $\exists((1, \sigma_1, \sigma_2), g_1) \in \mathsf{Ext}_g(\mathcal{L}_G)$ s.t. $\mathsf{enc}_1(\mathsf{pk}_1, \sigma_1; g_1) = \mathsf{ct}_1$
  and $\exists((2, \sigma_1, \sigma_2), g_2) \in \mathsf{Ext}_g(\mathcal{L}_G)$
    s.t. $\mathsf{enc}_2(\mathsf{pk}_2, \sigma_2; g_2) = \mathsf{ct}_2$:
  if k-th query : **abort**
  return $h(\sigma_1, \sigma_2)$
if k-th query :
  $(\sigma_1', \sigma_2') \leftarrow (\mathsf{dec}_1(\mathsf{sk}_1, \mathsf{ct}_1), \mathsf{dec}_2(\mathsf{sk}_2, \mathsf{ct}_2))$
  for i s.t. $\nexists((i, \sigma_1, \sigma_2), g_i) \in \mathsf{Ext}_g(\mathcal{L}_G)$
      s.t. $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g_i) = \mathsf{ct}_i$
    $g_i' \leftarrow g(i, \sigma_1', \sigma_2')$
    if $\mathsf{enc}_i(\mathsf{pk}_i, \sigma_1'; g_i') = \mathsf{ct}_i$: **return** 1
  abort
return $\perp$

$G(m)$

if $\exists g$ s.t. $(m, g) \in \mathcal{L}_G$:
  return g
$g \leftarrow\$\, \{0,1\}^n$
$\mathcal{L}_G \leftarrow \mathcal{L}_G \cup \{(m, g)\}$
for $((i, \sigma_1, \sigma_2), g) \in \mathsf{Ext}_g(\mathcal{L}_G)$:
  if $\mathsf{dec}_i(\mathsf{sk}_i, \mathsf{enc}_i(\mathsf{pk}_i, \sigma_i; g)) \neq \sigma_i$:
    abort
  if $\sigma_1 = \sigma_1^*$ **and** $\sigma_2 \neq \sigma_2^*$:
    if $\mathsf{enc}_1(\mathsf{pk}_1, \sigma_1; g) = \mathsf{ct}_1^*$:
      abort
  if $\sigma_2 = \sigma_2^*$ **and** $\sigma_1 \neq \sigma_1^*$:
    if $\mathsf{enc}_2(\mathsf{pk}_2, \sigma_2; g) = \mathsf{ct}_2^*$:
      abort

Fig. 26: Adversary $\mathcal{B}_k$ for the proof of Lemma 6.

Proposition 3. *The two functions $g(i, \sigma_1, \sigma_2)$ presented in Table 2 are ERF.*

Proof.

- $G(\sigma_1 \oplus \sigma_2) \oplus G(i, \sigma_i)$: We first define the extractor Ext_g. We can define $\mathcal{L}_{G_i}$ as the list of query/responses (m, g) s.t. m is of the form (i, σ_i) and $\mathcal{L}_G$ is the list of remaining query responses. The extractor outputs:
  1. $\{((1, \sigma_1, \sigma_1 \oplus \sigma), g \oplus g_1) : (\sigma, g) \in \mathcal{L}_G, ((1, \sigma_1), g_1) \in \mathcal{L}_{G_1}\}$. That is for each $1, \sigma_1$ that has been queried, it recovers σ_2 from the queries in $\mathcal{L}_G$ and outputs the corresponding value of $g(1, \sigma_1, \sigma_2)$.
  2. $\{((2, \sigma_2 \oplus \sigma, \sigma_2), g \oplus g_g) : (\sigma, g) \in \mathcal{L}_G, ((2, \sigma_2), g_2) \in \mathcal{L}_{G_2}\}$.

  This is straightforward to see that this function fulfils the properties of an extractor. In particular, if q_G queries are made to G, we have $q_E \leq q_G^2$ and $q_E^1 \leq q_G$. Finally, we show that if $g(i, \sigma_1, \sigma_2)$ is not in the output of the

$\mathcal{B}_1^{\mathcal{A},G}(\mathsf{pk}_1, \mathsf{ct}_1^*)$

$\mathsf{pp}_2 \leftarrow\$\, \mathsf{setup}_2(1^\lambda)$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\$\, \mathsf{gen}_2(\mathsf{pp}_2)$
$\sigma_2^* \leftarrow\$\, \mathcal{M}_2$
$\mathsf{ct}_2^* \leftarrow \mathsf{enc}_2(\mathsf{pk}, \sigma_2^*)$
$K \leftarrow\$\, \mathcal{K}$
run $\mathcal{A}^{\mathcal{O}_2^{\mathsf{Dec}}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), K)$
$\mathcal{L}_G^* \leftarrow \{\sigma_1 : ((1, \sigma_1, \sigma_2^*), g) \in \mathsf{Ext}_g(\mathcal{L}_G)\}$
$\sigma_1 \leftarrow\$\, \mathcal{L}_G^* \cup \mathsf{Ext}_h(1, \sigma_2^*, \mathcal{L}_H)$
return σ_1

Fig. 27: OW-CPA adversary for the proof of Theorem 10.

extractor, then it is indistinguishable from a value sampled uniformly at random. Let $g(i, \sigma_1, \sigma_2) = Y + X$ with $Y = G(\sigma_1 \oplus \sigma_2)$ and $X = G(i, \sigma_i)$. Clearly, by the property of RO, if $\sigma_1 \oplus \sigma_2$ or (i, σ_i) was not queried to G, we have Y, resp. X uniformly distributed. Then, $g(i, \sigma_1, \sigma_2)$ is uniformly distributed as well. Finally, if both are queried, the extractor will be able to compute $g(i, \sigma_1, \sigma_2)$.

- $G_1(i, \sigma_1) \oplus G_2(i, \sigma_2)$: We define the extractor as follows. For a given i, let $\mathcal{L}_{G_{ij}}$ be the list of query/answer for queries of the type (i, σ_j) to G_j. For any i (here $i \in [2]$), the extractor consider all pairs of tuples $((i, \sigma_1), g_1), ((i, \sigma_2), g_2) \in \mathcal{L}_{G_{i1}} \times \mathcal{L}_{G_{i2}}$ and for each of them outputs $((i, \sigma_1, \sigma_2), g_1 \oplus g_2)$. Clearly, such an extractor fulfils the necessary properties. Now, we show that $g(i, \sigma_1, \sigma_2)$ is distributed uniformly at random unless the extractor outputs a corresponding tuple. For a given i, let $X_j = G_j(i, \sigma_j)$ and $Z = X_1 + X_2$. Then, following a similar argument as in the previous point holds that Z looks uniform unless (i, σ_1) and (i, σ_2) have been queried to G_1 and G_2, respectively. If that happens, the extractor recovers $g(i, \sigma_1, \sigma_2)$. Finally, as in the previous function, we have $q_E \leq q_G^2$ and $q_E^1 \leq q_G$.

□