On IND-qCCA security in the ROM and its applications

CPA security is sufficient for TLS 1.3

Loïs Huguenin-Dumittan, Serge Vaudenay

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

Abstract. Bounded IND-CCA security (IND-qCCA) is a notion similar to the traditional IND-CCA security, except the adversary is restricted to a constant number q of decryption/decapsulation queries. We show in this work that IND-qCCA is easily obtained from any passively secure PKE in the (Q)ROM. That is, simply adding a confirmation hash or computing the key as the hash of the plaintext and ciphertext holds an IND-qCCA KEM. In particular, there is no need for derandomization or re-encryption as in the Fujisaki-Okamoto (FO) transform [15]. This makes the decapsulation process of such IND-qCCA KEM much more efficient than its FO-derived counterpart. In addition, IND-qCCA KEMs could be used in the recently proposed KEMTLS protocol [29] that requires IND-1CCA ephemeral key-exchange mechanisms, or in TLS 1.3. Then, using similar proof techniques, we show that CPA-secure KEMs are sufficient for the TLS 1.3 handshake to be secure, solving an open problem in the ROM. In turn, this implies that the PRF-ODH assumption used to prove the security of TLS 1.3 is not necessary and can be replaced by the CDH assumption in the ROM. We also highlight and briefly discuss several use cases of IND-1CCA KEMs in protocols and ratcheting primitives.

1 Introduction

As the NIST standardization process for post-quantum (PQ) public-key cryptography progresses, studying how these new PQ schemes could be integrated into existing protocols has become a hot topic. In particular, the newly adopted TLS 1.3 in its standard form is already "PQ-obsolete" in the sense that only traditional Diffie-Hellman (DH) key-exchange is supported. Indeed, as most PQ schemes come into the form of *Key-Encapsulation Mechanisms* (KEMs) and not Key-Exchange (KEX) such as DH, TLS 1.3 needs modifications to be quantum-resistant.

Several implementations of PQ TLS 1.3 have already been experimented, the most well-known one surely being the OQS-OpenSSL project [1]. This library implements a TLS handshake that supports KEMs and *hybrid cryptography* (i.e. the final shared secret is a combination of a DH secret and a KEM secret/key). The changes compared to the standard version of the TLS 1.3 handshake are minimal. That is, the client (resp. server) DH share is replaced by a public-key (resp. a ciphertext encapsulated under the public-key), and the shared secret is the key encapsulated in the ciphertext. Several works have analysed the performance and implementation challenges of OQS-OpenSSL (e.g. [9,25]).

More recently, based on the observation that (PQ) KEM public-keys/ciphertexts are usually more compact than (PQ) public-keys/signatures, Schwabe et al. [29] proposed KEMTLS as a variant of the TLS 1.3 handshake. The main difference between both protocols is that KEMTLS uses a KEM for (implicit) server authentication instead of a signature. This reduces the overall bandwidth of the handshake and the computation time on the server-side. Thus, two KEMs are used in KEMTLS: one for establishing an ephemeral shared secret and the other one to authenticate the server. While the latter needs to be IND-CCA secure as it uses long-term keys, the authors showed that IND-1CCA security is sufficient for the former KEM for the whole handshake to be secure. That is, the KEM needs to be secure against an adversary that can make a *unique* decapsulation query. Similarly, in the security proof of TLS 1.3 handshake by Dowling et al. [12], DH key-exchange can be replaced by an IND-1CCA KEM and the proof would still go through.

However, in KEMTLS or PQ implementations of TLS 1.3 (e.g. [1]), the ephemeral KEMs are implemented with IND-CCA KEMs, which are usually obtained by applying the Fujisaki-Okamoto (FO) transform or a variant (e.g. [15,19]) on an OW/IND-CPA public-key encryption scheme (PKE). The FO construction re-encrypts the decrypted plaintext during decapsulation, making it an expensive operation. This motivates the present work, which studies whether IND-1CCA KEM can be obtained from CPA-secure PKEs through a more efficient transform than FO (in the ROM). We reply by the affirmative by showing that IND-1CCA KEMs with much faster decapsulation than FO-derived IND-CCA KEM can be obtained from any CPA-secure PKE. Using similar tools, we also study the security of the PQ TLS 1.3 handshake when the KEM used for key exchange is only CPA-secure.

Our contributions

We show how to build an efficient IND-qCCA KEM (i.e. the adversary can only make q decapsulation queries) from any OW-CPA PKE in the ROM. The bound has a loose factor of 2^q, making it insecure or impractical for large q. However, such construction is sufficient to build an efficient IND-1CCA KEM from any OW-CPA public-key encryption scheme. The transform simply sends a confirmation hash along the ciphertext encrypting the seed. In addition, we prove the security of this construction in the QROM as well.

Such a transform might be useful in several applications such as the KEMTLS protocol [29] mentioned above, PQ variants of TLS 1.3 or ratcheting, as discussed in Section 5.

Similarly, we show that deriving the key as $K := H(m, \mathsf{ct})$, where m is the seed encrypted in the ciphertext ct, holds an IND-qCCA KEM in the ROM. The bound is worse compared to the first transform, having a $\approx q_H^{2q}$ factor, where q_H

is the number of queries an adversary can make to the random oracle H. The intuition is that any decapsulation query that returns $H(m, \mathsf{ct})$ with $\mathsf{ct} \neq \mathsf{ct}^*$ does not help much the adversary to recover the real key $H(m^*, \mathsf{ct}^*)$ due to the independence of RO values. However, each query to the decapsulation oracle still leaks a little information (such as equality between decrypted values), leading to the $\approx q_H^{2q}$ factor.

Compared to the FO transform and its variants, our CPA-to-qCCA transforms offer several advantages. The main one is a significant speed boost in decapsulation, as there is no need for re-encryption. Depending on the cost of encryption of the underlying scheme, the difference can be large. For instance, removing the re-encryption check in the optimized version of the isogeny-based scheme SIKE [21] cuts by more than 50% the decapsulation time (32235377 vs 73282449 cycles for SIKEp434_compressed on Ubuntu 21.04 with 2.8GHz Intel Core i7-1165G7). Another interesting feature of our transform is that we do not need to de-randomize the encryption (i.e. computing the random coins for encapsulation as the hash of the message/seed), removing the need for an additional random oracle.

We then consider the PQ TLS 1.3 handshake as it is implemented in OQS-OpenSSL [1]. Based on the observation that the key-schedule computes the keys as key-derivation functions (KDFs) applied on the shared secret and (the hash of) the transcript so far (including the ciphertext), we prove that if the KEM is OW-CPA secure, then the handshake is secure in the MultiStage model of Dowling et al. [12]. The proof is inspired by the proof of security of our second transform. Note that this result holds in the ROM (the KDFs/hash function are assumed to be ROs) and the security bound is very much "non-tight". Still, this shows that CPA-secure KEMs are sufficient for the TLS 1.3 handshake to be secure, solving an open problem raised by several authors (e.g. [12,25]). Then, since one can consider DH as a KEM, this implies that TLS 1.3 is secure as long as the *computational Diffie-Hellman* (CDH) problem is hard, showing that the PRF-ODH assumption used in the original proof [12] is not necessary (in the ROM). We note that this last result can also be derived from the fact that DH as used in TLS 1.3 is a IND-1CCA KEM in the ROM, assuming that CDH is hard. We prove this in the full version of the paper [20].

Finally, in Section 5, we discuss possible use cases of IND-qCCA in the context of communication protocols and ratcheting primitives. In particular, we note that IND-1CCA security is sufficient in many recent applications as the trend is to move to forward secure schemes, which discard key pairs after one use.

Remark on IND-CPA vs IND-1CCA

We note that plain IND-CPA PQ schemes are often not IND-1CCA. In particular, it is stated in Section 4.3 of the KEMTLS paper [30]:

"We leave as an open question to what extent non-FO-protected post-quantum KEMs may be secure against a single decapsulation query, but at this point IND-CCA is the safe choice."

The answer to this question obviously depends on how the "non-FO protected" IND-CPA PKE is used as a KEM. However, if it used in the trivial way (i.e. $m \leftarrow\$ \mathcal{M}$, $K := H(m), \mathsf{ct} := \mathsf{enc}(\mathsf{pk}, m)$), the resulting KEM can usually be broken with 1 query for most of the PQ schemes. The adversary receives $K^*, \mathsf{ct}^* := \mathsf{enc}(\mathsf{pk}, m^*)$, queries $\mathsf{ct}^* + \delta$ and gets back $H(m^*)$ with high probability, if δ is "small". Then, it can just compare whether $H(m^*) = K^*$ or not and break IND-1CCA security. The reaction attacks (e.g. [13]) requiring thousands of queries mentioned in the same paper [30] are key-recovery attacks, not distinguishing attacks. The simple distinguishing adversary given above actually gives a good intuition of why adding a confirmation hash $H'(m, \mathsf{ct})$ along the ciphertext as in our first transform holds a IND-qCCA KEM. In order to submit a valid decapsulation query, the adversary must compute $H'(m, \mathsf{ct})$ with $\mathsf{ct} \neq \mathsf{ct}^*$. Hence, the adversary itself needs to query $H'(m, \mathsf{ct})$ beforehand, thus it knows m and the decapsulation query is (nearly) useless.

Related work

The notion of bounded IND-CCA (i.e. IND-qCCA) has been studied in several works. Cramer et al. [8] defined IND-qCCA and showed that one can build an IND-qCCA PKE from any CPA-secure PKE in a black-box manner in the standard model, using one-time signatures. While this construction is valid in the standard model and ours in the ROM only, their reduction is inefficient compared to FO transforms, which we aim to improve. Following their work, Peirera et al. [26] built a more efficient IND-qCCA PKE based on the CDH assumption, and Yamakawa et al. [32] proposed other constructions based on the factoring and bilinear CDH assumptions. As far as we know, we are the first to note that a IND-qCCA KEM can be obtained from any CPA-secure PKE through a very simple and efficient transform in the ROM.

Starting from the original Fujisaki-Okamoto transform [14,15], many works have been dedicated to building variants of FO with tighter security bounds in the QROM (e.g. [19,4,23,28]). While these are CPA-to-CCA transforms, ours guarantee qCCA security only but at a lesser computational cost.

Dowling et al. [12] proved the security of the standard TLS 1.3 handshake in the MultiStage security model. We extend their result by showing that TLS 1.3 security still holds if the DH KEX is replaced by a CPA-secure KEM (in the ROM). In turn, this also implies that the CDH assumption is sufficient for proving the security of the original TLS 1.3, which was based on the PRF-ODH assumption so far. In two more recent works, Diemert et al. [11] and Davis et al. [10] aimed at proving a tighter security bound for TLS 1.3. Their proofs are valid in the ROM and are based on the Strong Diffie-Hellman (SDH) assumption. Our result on TLS 1.3 is complementary to theirs in the sense that we prove that TLS security holds under a weaker assumption but with a looser security bound.

Brendel et al. [5] studied the PRF-ODH assumption. In particular, they showed that PRF-ODH is hard if the SDH assumption holds in the ROM. The PRF-ODH notion considered in their work is generic as the adversary can query two types of "decapsulation" oracles multiple times. On the other hand, if we restrict ourselves to the notion where the adversary can make a unique query (which is sufficient for TLS 1.3 security), we show in the full version of the paper [20] that CDH hardness is sufficient for PRF-ODH (with one query) to hold.

Finally, following the KEMTLS paper [29], several recent works used the notion of IND-1CCA KEM to build secure protocols (e.g. [18,31,6]), showing the growing importance of such a notion.

2 Preliminaries

2.1 Notation

For $\mathcal{A}$ a randomized algorithm, we write $b \leftarrow\$ \mathcal{A}$ to indicate b is set to 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 $\{1, \ldots, n\}$. For $\mathcal{A}$ an algorithm, we write $\mathcal{A} \Rightarrow b$ to denote the event $\mathcal{A}$ *outputs* b. Finally, in a game, we write **abort** to mean that the algorithm is stopped.

2.2 Public-Key Encryption scheme

A Public-Key Encryption (PKE) scheme is defined as follows.

Definition 1 (Public-Key Encryption). *A Public-Key Encryption scheme over a domain* $\mathcal{M}$ *is composed of three algorithms* $\mathsf{gen}, \mathsf{enc}, \mathsf{dec}$*:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{gen}(1^\lambda)$*: The key generation algorithm takes the security parameter as input and outputs the public key* pk *and the secret key* sk*.*
- $\mathsf{ct} \leftarrow\$ \mathsf{enc}(\mathsf{pk}, \mathsf{pt})$*: The encryption algorithm takes as inputs the public key* pk *and a plaintext* $\mathsf{pt} \in \mathcal{M}$ *and it outputs a ciphertext* ct*.*
- $\mathsf{pt}' \leftarrow \mathsf{dec}(\mathsf{sk}, \mathsf{ct})$*: The decryption procedure takes as inputs 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 gen *and* enc *are probabilistic algorithms that can be made deterministic by adding random coins as inputs. The decryption procedure is deterministic.*

Correctness. We say a PKE scheme is δ correct if for any ppt adversary $\mathcal{A}$ playing the game CORR defined in Fig. 1, we have

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

where λ is the security parameter, we omit it from now on for the sake of simplicity.

$\mathrm{CORR}_{\mathsf{PKE}}(\mathcal{A})$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{gen}(1^\lambda)$
$\mathsf{pt} \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk})$
$\mathsf{ct} \leftarrow\$ \mathsf{enc}(\mathsf{pk}, \mathsf{pt})$
return $1_{\mathsf{dec}(\mathsf{sk},\mathsf{ct}) \neq \mathsf{pt}}$

Fig. 1: Correctness game.

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

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

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

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

ATK	CPA	PCA
$\mathcal{O}^{\mathsf{ATK}}$	$\perp$	$\mathcal{O}^{\mathsf{PCO}}$

Fig. 2: One-Wayness games.

Plaintext Checking. We recall the notions of One-Wayness under Chosen Plaintext Attacks (OW-CPA) and Plaintext-Checking Attacks (OW-PCA).

Definition 2 (One-Wayness and Plaintext Checking). *Let $\mathcal{M}$ be a finite message space,* PKE *a PKE scheme over $\mathcal{M}$ and we consider the games defined on the left in Fig. 2 with the different oracles as defined in the table on the right of Fig. 2. Then,* PKE *is* OW-ATK*, for* ATK $\in \{\mathrm{CPA}, \mathrm{PCA}\}$*, if for any ppt adversary $\mathcal{A}$ we have*

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

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

2.3 Key Encapsulation Mechanism (KEM)

A Key Encapsulation Mechanism is defined as follows.

Definition 3 (Key Encapsulation Mechanism). *A KEM over $\mathcal{K}$ is a tuple of three algorithms* $\mathsf{gen}, \mathsf{encaps}, \mathsf{decaps}$*:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{gen}(1^\lambda)$*: The key generation algorithm takes as inputs the security parameter and it outputs the public key* pk *and the secret key* sk*.*
- $\mathsf{ct}, K \leftarrow\$ \mathsf{encaps}(\mathsf{pk})$*: The encapsulation algorithm takes as inputs the public key* pk *and it outputs a ciphertext* $\mathsf{ct} \in \mathcal{C}$ *and a key $K \in \mathcal{K}$.*
- $K' \leftarrow \mathsf{decaps}(\mathsf{sk}, \mathsf{ct})$*: The decapsulation procedure takes as inputs 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 $K \in \mathcal{K}$.*

$\underline{\text{IND-(q)CCA}_{\mathsf{KEM}}(\mathcal{A})}$	**Oracle** $\mathcal{O}^{\mathsf{Dec}}(\mathsf{ct})$
$(\mathsf{pk},\mathsf{sk}) \leftarrow\!\$\ \mathsf{gen}(1^\lambda)$	**if** $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
$b \leftarrow\!\$\ \{0,1\}$	**if** more than q queries : **return** $\perp$ // If IND-qCCA
$\mathsf{ct}^*, K_0 \leftarrow\!\$\ \mathsf{encaps}(\mathsf{pk})$	$K' \leftarrow \mathsf{decaps}(\mathsf{sk},\mathsf{ct})$
$K_1 \leftarrow\!\$\ \mathcal{K}$	**return** K'
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Dec}}}(\mathsf{pk},\mathsf{ct}^*,K_b)$	
return $1_{b'=b}$	

Fig. 3: Indistinguishability games.

The gen *and* encaps *are probabilistic algorithms. The randomness can be made explicit by adding random coins as inputs. The decapsulation function is deterministic.*

Indistinguishability security. KEM indistinguishability is defined as follows.

Definition 4 (KEM Indistinguishability). *We consider the games defined in Fig. 3. Let $\mathcal{K}$ be a finite key space. A KEM scheme over $\mathcal{K}$ $\mathsf{KEM} = (\mathsf{gen},\mathsf{encaps},\mathsf{decaps})$ is* IND-CCA *(resp.* IND-qCCA*) if for any ppt adversary $\mathcal{A}$ (resp. any ppt $\mathcal{A}$ limited to q decapsulation queries) we have*

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

where $\Pr\left[\text{IND}-\text{(q)CCA}_{\mathsf{KEM}}(\mathcal{A}) \Rightarrow 1\right]$ *is the probability that $\mathcal{A}$ wins the* IND-*(q)CCA$_{\mathsf{KEM}}(\mathcal{A})$ game defined in Fig. 3.*

We can also define OW-CPA for KEMs, which is similar to the equivalent notion for PKE.

Definition 5 (KEM OW-CPA). *A KEM scheme* $\mathsf{KEM} = (\mathsf{gen},\mathsf{encaps},\mathsf{decaps})$ *is* OW-CPA *if for any ppt adversary $\mathcal{A}$ we have*

$$\begin{aligned}\mathsf{Adv}_{\mathsf{KEM}}^{\text{ow-cpa}}(\mathcal{A}) &= \Pr\left[\mathcal{A}(\mathsf{pk},\mathsf{ct}^*) \Rightarrow K : (\mathsf{pk},\mathsf{sk}) \leftarrow\!\$\ \mathsf{gen}(1^\lambda); (K,\mathsf{ct}^*) \leftarrow\!\$\ \mathsf{encaps}(\mathsf{pk})\right] \\ &= \mathsf{negl}(\lambda)\ ,\end{aligned}$$

where the probability is taken over the randomness of the public-key generation, encapsulation and the adversary $\mathcal{A}$.

3 OW-CPA to IND-qCCA transforms

We first prove the following simple lemma.

Lemma 1. *Let* PKE *be a PKE. Then, for any ppt OW-PCA adversary $\mathcal{A}$ making at most q queries to the PCO oracle, there exists a OW-CPA adversary $\mathcal{B}$ s.t.*

$$\mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pca}}(\mathcal{A}) \leq 2^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-cpa}}(\mathcal{B})\ .$$

$\mathsf{gen}()$	$\mathsf{encaps}(\mathsf{pk})$	$\mathsf{decaps}(\mathsf{sk}, \mathsf{ct})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{gen}^{\mathsf{p}}()$	$\sigma \leftarrow\$ \mathcal{M}$	$(\mathsf{ct}_0', \mathsf{tag}') \leftarrow \mathsf{ct}$
return $(\mathsf{pk}, \mathsf{sk})$	$\mathsf{ct}_0 \leftarrow\$ \mathsf{enc}^{\mathsf{p}}(\mathsf{pk}, \sigma)$	$\sigma' \leftarrow \mathsf{dec}^{\mathsf{p}}(\mathsf{sk}, \mathsf{ct}_0')$
	$\mathsf{tag} \leftarrow H'(\sigma, \mathsf{ct}_0)$	**if** $H'(\sigma', \mathsf{ct}_0') \neq \mathsf{tag}'$:
	$K \leftarrow H(\sigma)$	**return** $\perp$
	return $K, (\mathsf{ct}_0, \mathsf{tag})$	**return** $H(\sigma')$

Fig. 4: $\mathsf{T_{CH}}$ transform.

Proof. We can simply see that the PCO oracle returns 1 bit of information, thus PKE loses at most q bits of security when a PCO oracle is available. More formally, given $\mathcal{A}$, one can build $\mathcal{B}$ as follows. It passes its input to $\mathcal{A}$ and simulates the PCO oracle by sampling a response at random in $\{0,1\}$. Then, it returns the response of $\mathcal{A}$. Its probability of success is $\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}}(\mathcal{B}) \geq \frac{1}{2^q}\mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathcal{A})$, as the probability the q responses are correct is $\frac{1}{2^q}$. □

We consider the transform $\mathsf{T_{CH}}$ given in Fig. 4. This construction takes a PKE $\mathsf{PKE} = (\mathsf{gen}^{\mathsf{p}}, \mathsf{enc}^{\mathsf{p}}, \mathsf{dec}^{\mathsf{p}})$ and outputs a KEM $(\mathsf{gen}, \mathsf{encaps}, \mathsf{decaps})$. Note that $\mathsf{T_{CH}}$ is basically the REACT transform [24] without the asymmetric part (to get a KEM instead of a PKE).

We now show that the resulting KEM is IND-qCCA assuming the underlying PKE is OW-PCA.

Theorem 1. *We consider two random oracles $H, H' : \{0,1\}^* \mapsto \{0,1\}^n$. Let* KEM *be the KEM resulting from applying the* $\mathsf{T_{CH}}$ *transform to a δ-correct* PKE. *Then, for any IND-qCCA adversary $\mathcal{A}$ that makes at most q_H (resp. $q_{H'}$) queries to H (resp. H'), there exists a OW-PCA adversary $\mathcal{B}$ s.t.*

$$\mathsf{Adv}^{\mathrm{ind-qcca}}_{\mathsf{KEM}}(\mathcal{A}) \leq \frac{(q + q_{H'} + 1)^2}{2^n} + \delta + \frac{q}{2^n} + (q_H + q_{H'}) \cdot \mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathcal{B}) \ ,$$

where $\mathcal{B}$ makes at most q queries to its plaintext-checking oracle. In addition, if PKE *is a deterministic encryption scheme, the bound becomes*

$$\mathsf{Adv}^{\mathrm{ind-qcca}}_{\mathsf{KEM}}(\mathcal{A}) \leq \frac{(q + q_{H'} + 1)^2}{2^n} + \delta + \frac{q}{2^n} + \mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathcal{B}) \ .$$

Proof. We proceed by game hopping, the sequence of games is presented in Fig. 5. Let $\mathcal{L}_H$ (resp. $\mathcal{L}_{H'}$) be the list of queries (x, h) made to the RO H (resp. H') s.t. $H(x) = h$ (resp. $H'(x) = h$). In addition, let the challenge ciphertext be $\mathsf{ct}^* = (\mathsf{ct}_0^*, h^*)$, and σ^* be s.t. $\mathsf{enc}^{\mathsf{p}}(\mathsf{pk}, \sigma^*) = \mathsf{ct}_0^*$. We start with game Γ^0 which is the IND-qCCA game, except we abort if the adversary finds a collision on H' (i.e. $H'(x) = H'(x')$ for $x \neq x'$ and $(x,h), (x',h) \in \mathcal{L}_{H'}$). This happens with prob. at most $\frac{(q+q_{H'}+1)^2}{2^n}$ and we have

$$\left|\Pr\left[\mathrm{IND-qCCA}_{\mathsf{KEM}}(\mathcal{A}) \Rightarrow 1\right] - \Pr[\Gamma^0(\mathcal{A}) \Rightarrow 1]\right| \leq \frac{(q + q_{H'} + 1)^2}{2^n} \ .$$

$\Gamma^{0-3}(\mathcal{A})$

```
(pk, sk) ←$ gen
b ←$ {0, 1}
σ* ←$ {0, 1}^n
ct*_0 ←$ enc^p(pk, σ*)
K_0 ← H(σ*); h* ← H'(σ*, ct*_0)
K_1 ←$ K
ct* ← (ct*_0, h*)
b' ← A^{O^Dec}(pk, ct*, K_b)      // Γ^0-Γ^1
b' ← A^{O^Dec2}(pk, ct*, K_b)     // Γ^2-Γ^3
if query : abort                  // Γ^3
return 1_{b'=b}
```

$H(\sigma)$

```
if ∃h s.t. (σ, h) ∈ L_H :
  return h
if σ = σ* : query ← true          // Γ^3
h ←$ {0, 1}^n
L_H ← L_H ∪ {(σ, h)}
return h
```

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

```
if ct = ct* : return ⊥
if more than q queries :
  return ⊥
(ct_0, h) ← ct
if ct_0 = ct*_0 or h = h* :       // Γ^1-Γ^3
  return ⊥                        // Γ^1-Γ^3
σ' ← dec^p(sk, ct'_0)
if H'(σ', ct_0) ≠ h :
  return ⊥
return H(σ')
```

$H'(\sigma, \mathsf{ct})$

```
if ∃h s.t. ((σ, ct), h) ∈ L_{H'} :
  return h
if σ = σ* : query ← true          // Γ^3
h ←$ {0, 1}^n
L_{H'} ← L_{H'} ∪ {((σ, ct), h)}
if ∃x, x', h s.t. x ≠ x'
        ∧ (x, h) ∈ L_{H'}
        ∧ (x', h) ∈ L_{H'} :
  abort
return h
```

Oracle $\mathcal{O}^{\mathsf{Dec2}}(\mathsf{ct})$

```
if ct = ct* : return ⊥
if more than q queries :
  return ⊥
(ct_0, h) ← ct
if ct_0 = ct*_0 or h = h* :
  return ⊥
if ∃σ s.t. ((σ, ct_0), h) ∈ L_{H'} :
  if O^PCO(σ, ct_0) :
    return H(σ)
return ⊥
```

Fig. 5: Sequence of games for the proof of Thm 1. $\mathcal{O}^{\mathsf{PCO}}$ is defined as in the OW-PCA game (see Fig. 2).

$\underline{\Gamma^1:}$ The decapsulation oracle is modified s.t. it returns $\perp$ whenever ct_0^* or h^* is queried (note that both cannot be submitted at the same time). This game is the same as Γ^0 except if the oracle in Γ^0 does not return $\perp$ on such queries. Let bad be this event. We split this into two cases:

- $\mathcal{O}^{\mathsf{Dec}}(\mathsf{ct}_0^*, h \neq h^*) \neq \perp$. This happens only if
$$H'(\mathsf{dec}(\mathsf{sk}, \mathsf{ct}_0^*), \mathsf{ct}_0^*) = h \neq h^* = H'(\sigma^*, \mathsf{ct}_0^*) \ .$$
In turn, this implies that $\mathsf{dec}(\mathsf{sk}, \mathsf{ct}_0^*) \neq \sigma^*$ and thus it is a correctness error. Such an error happens at most with probability δ.
- $\mathcal{O}^{\mathsf{Dec}}(\mathsf{ct}_0 \neq \mathsf{ct}_0^*, h^*) \neq \perp$. It means that $h^* = H'(\sigma^*, \mathsf{ct}_0^*) = H'(\sigma', \mathsf{ct}_0)$, with $\sigma' \leftarrow \mathsf{dec}^{\mathsf{p}}(\mathsf{sk}, \mathsf{ct}_0)$, which is not possible since $\mathsf{ct}_0 \neq \mathsf{ct}_0^*$ and we assume no collision occurs.

Therefore, overall $\Pr[\mathsf{bad}] \leq \delta$ and
$$|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]| \leq \Pr[\mathsf{bad}] \leq \delta \ .$$

$\underline{\Gamma^2:}$ We modify the decapsulation oracle into another oracle $\mathcal{O}^{\mathsf{Dec}^2}$ as follows. On a decapsulation query (ct_0, h) (with $\sigma' \leftarrow \mathsf{dec}(\mathsf{sk}, \mathsf{ct}_0)$):

1. If there is no $((*, \mathsf{ct}_0), h)$ in $\mathcal{L}_{H'}$: return $\perp$. This differs from the previous game only if $h = H'(\sigma', \mathsf{ct}_0)$ but (σ', ct_0) was never queried to H'. As the RO values are uniformly distributed, this happens at most with probability $\frac{1}{2^n}$.
2. If $((\sigma, \mathsf{ct}_0), h) \in \mathcal{L}_{H'}$ for some σ: If $\mathcal{O}^{\mathsf{PCO}}(\sigma, \mathsf{ct}_0) := 1_{\mathsf{dec}(\mathsf{sk},\mathsf{ct}_0)=\sigma} = 1$, return $H(\sigma)$. Otherwise, return $\perp$. This perfectly simulates the previous oracle as $\mathcal{O}^{\mathsf{PCO}}(\sigma, \mathsf{ct}_0) = 1$ iff $\sigma = \sigma'$ and we know $h = H(\sigma = \sigma', \mathsf{ct}_0)$.
   Note that there is at most one σ s.t. $((\sigma, \mathsf{ct}_0), h) \in \mathcal{L}_{H'}$ as we assume no collision occurs. In particular, it means that $\mathcal{O}^{\mathsf{PCO}}$ is called at most once every decapsulation query.

Therefore, by a union bound we get

$$|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]| \leq \frac{q}{2^n} .$$

$\underline{\Gamma^3:}$ Finally, we abort whenever $\mathcal{A}$ queries σ^* to H or $(\sigma^*, \cdot)$ to H'. Let this event be query. Note that $\mathcal{A}$ could also learn the value of $H(\sigma^*)$ through a query to $\mathcal{O}^{\mathsf{Dec}2}$. However, the latter oracle would return $H(\sigma^*)$ only if $\mathcal{A}$ queried $H'(\sigma^*, \cdot)$ before (thus triggering query).

Then, we can build a OW-PCA adversary $\mathcal{B}$ (shown in Fig. 6) that perfectly simulates $\mathcal{A}$'s view as long as query does not happen. More precisely, $\mathcal{B}$ can simulate the decapsulation oracle using its PCO oracle. Then, on input $(\mathsf{pk}, \mathsf{ct}_0^*)$, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pk}, (\mathsf{ct}_0^*, h^*), K^*)$, where h^* and K^* are picked at random. Unless query occurs, $\mathcal{A}$ cannot distinguish between these random h^*, K^* and the real ones. Finally, if query occurs, $\mathcal{B}$ can recover σ^* with probability $\frac{1}{q_H + q_{H'}}$ by sampling a random σ from $S = \{\sigma : (\sigma, *) \in \mathcal{L}_H^{\mathcal{A}} \vee ((\sigma, *), *) \in \mathcal{L}_{H'}\}$, where $\mathcal{L}_H^{\mathcal{A}}$ is the set of queries to H made by $\mathcal{A}$. Thus,

$$|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^3 \Rightarrow 1]| \leq \Pr[\mathsf{query}] \leq (q_H + q_{H'}) \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pca}}(\mathcal{B})$$

where $\mathcal{B}$ makes q query to the PCO oracle. Note that if PKE is deterministic, $\mathcal{B}$ can check whether $\mathsf{enc}(\mathsf{pk}, \sigma) = \mathsf{ct}_0^*$ for all $\sigma \in S$ to find σ^*. This fails only if there exists $\sigma' \neq \sigma^*$ s.t. $\mathsf{enc}(\mathsf{pk}, \sigma') = \mathsf{ct}_0^*$. In turn this implies that there exists $\sigma \in S \cup \{\sigma^*\}$ that would break the correctness, but such an event is already covered by the previous δ factor. In this case, we obtain

$$|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^3 \Rightarrow 1]| \leq \Pr[\mathsf{query}] \leq \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pca}}(\mathcal{B}) .$$

Finally, since $\mathcal{A}$ cannot query σ^* to H anymore, it cannot distinguish between a random key and $H(\sigma^*)$. Hence, $\Pr[\Gamma^3 \Rightarrow 1] = \frac{1}{2}$. Collecting the probabilities holds the result. □

$\mathcal{B}^{\mathcal{O}^{\mathsf{PCO}}}(\mathsf{pk}, \mathsf{ct}^*)$	**Oracle** $\mathcal{O}^{\mathsf{Dec2}}(\mathsf{ct})$
init $\mathcal{L}_H, \mathcal{L}_{H'} \leftarrow \emptyset$	**if** more than q queries :
$h^* \leftarrow\$ \{0,1\}^n$	**return** $\perp$
$K^* \leftarrow\$ \{0,1\}^n$	$(\mathsf{ct}_0, h) \leftarrow \mathsf{ct}$
simulate H, H' for $\mathcal{A}$ with lazy sampling:	**if** $\mathsf{ct}_0 = \mathsf{ct}_0^*$ or $h = h^*$:
run $\mathcal{A}^{H,H',\mathcal{O}^{\mathsf{Dec2}}}(\mathsf{pk}, (\mathsf{ct}^*, h^*), K^*)$	**return** $\perp$
$\sigma' \leftarrow\$ \{\sigma : \sigma \in \mathcal{L}_H^{\mathcal{A}} \vee (\sigma, *) \in \mathcal{L}_{H'}\}$	**if** $\exists \sigma$ s.t. $((\sigma, \mathsf{ct}), h) \in \mathcal{L}_{H'}$:
return σ'	**if** $\mathcal{O}^{\mathsf{PCO}}(\sigma, \mathsf{ct}_0)$: **return** $H(\sigma)$
	return $\perp$

Fig. 6: $\mathcal{B}$ adversary for the proof of Thm 1.

Corollary 1. *We consider two random oracles $H, H' : \{0,1\}^* \mapsto \{0,1\}^n$. Let* KEM *be the KEM resulting from applying the* $\mathsf{T_{CH}}$ *transform to a δ-correct* PKE. *Then, for any IND-qCCA adversary $\mathcal{A}$ that makes at most q_H (resp. $q_{H'}$) queries to H (resp. H'), there exists a OW-CPA adversary $\mathcal{B}$ s.t.*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathcal{A}) \leq \frac{(q + q_{H'})^2}{2^n} + \delta + \frac{q}{2^n} + (q_H + q_{H'} + q)2^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-cpa}}(\mathcal{B}) \ .$$

If PKE *is deterministic, we get*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathcal{A}) \leq \frac{(q + q_{H'})^2}{2^n} + \delta + \frac{q}{2^n} + 2^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-cpa}}(\mathcal{B}) \ .$$

In particular, in the case of IND-1CCA (i.e. $q = 1$), if the underlying PKE is OW-CPA the KEM obtained from the $\mathsf{T_{CH}}$ transform is IND-1CCA with a security loss of ≈ 1 bit compared to the OW-CPA advantage (if we omit the other negligible terms). Finally, we note that as q is a constant that does not depend on the security parameter (e.g. n) of the PKE, if the OW-CPA advantage of the PKE is negligible, so is the KEM IND-qCCA one. However, in practice, we would need to take n very large to guarantee security for more than a few queries.

3.1 Security in the QROM.

We also show that the $\mathsf{T_{CH}}$ transform is secure in the Quantum Random Oracle Model (QROM) by proving that Thm 1 holds in the QROM.

Theorem 2. *We consider two quantum random oracles $H, H' : \{0,1\}^* \mapsto \{0,1\}^n$. Let* KEM *be the KEM resulting from applying the* $\mathsf{T_{CH}}$ *transform to a* PKE. *Then, for any IND-qCCA adversary $\mathcal{A}$ that makes at most q_H (resp. $q_{H'}$) quantum queries to H (resp. H'), there exists a OW-PCA adversary $\mathcal{B}$ s.t.*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathcal{A}) \leq \delta + 2(2q_{H'} + q_H + q) \cdot \sqrt{(2(2q_{H'} + q))^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pca}}(\mathcal{B})}$$

where $\mathcal{B}$ makes at most q queries to its plaintext-checking oracle.

$\underline{\mathsf{gen}()}$	$\underline{\mathsf{encaps}(\mathsf{pk})}$	$\underline{\mathsf{decaps}(\mathsf{sk},\mathsf{ct})}$
$(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{gen}^{\mathsf{p}}()$	$\sigma \leftarrow\$ \mathcal{M}$	$\sigma' \leftarrow \mathsf{dec}^{\mathsf{p}}(\mathsf{sk},\mathsf{ct})$
return $(\mathsf{pk},\mathsf{sk})$	$\mathsf{ct} \leftarrow\$ \mathsf{enc}^{\mathsf{p}}(\mathsf{pk},\sigma)$	**if** $\sigma' = \bot$: **return** $\bot$
	$K \leftarrow H(\sigma,\mathsf{ct})$	**return** $H(\sigma',\mathsf{ct})$
	return K,ct	

Fig. 7: $\mathsf{T_H}$ transform.

$\underline{\mathcal{O}^{\mathsf{i}}(\mathcal{L}_H,\mathsf{ct})}$

sort $\mathcal{L}_H$ according to query order :
$\mathcal{L}_H = ((\sigma_i,\mathsf{ct}_i),K_i)_{i\in\{1,\ldots,|\mathcal{L}_H|\}}$
$\sigma' \leftarrow \mathsf{dec}^{\mathsf{p}}(\mathsf{sk},\mathsf{ct})$
if $\sigma' = \bot$: **return** $\bot_d$
for $i \in \{1,\ldots,|\mathcal{L}_H|\}$:
  if $\mathsf{ct}_i = \mathsf{ct}$ **and** $\sigma' = \sigma_i$:
    return i
return $\bot$

Fig. 8: $\mathcal{O}^{\mathsf{i}}$ oracle for the proof of Thm 3.

Proof. Due to space constraint, we defer the proof to the full version of the paper [20]. □

Corollary 2. *We consider two quantum random oracles $H, H' : \{0,1\}^* \mapsto \{0,1\}^n$. Let* KEM *be the KEM resulting from applying the* $\mathsf{T_{CH}}$ *transform to a δ-correct* PKE*. Then, for any IND-qCCA adversary $\mathcal{A}$ that makes at most q_H (resp. $q_{H'}$) queries to H (resp. H'), there exists a OW-CPA adversary $\mathcal{B}$ s.t.*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathcal{A}) \leq \delta + 2(2q_{H'} + q_H + q)\cdot 2^q \sqrt{((2q_{H'}+q))^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pca}}(\mathcal{B})}\ .$$

3.2 Hashing the plaintext and ciphertext

One can also wonder what is the leakage of the decapsulation oracle in the ROM, when the key is simply the hash of the seed and the plaintext. That is, we consider the simple PKE to KEM transform given in Fig. 7, which we call $\mathsf{T_H}$. Note that this is the same transform as the one called $\mathsf{U}^{\bot}$ in [19]. We now show that if q is small (logarithmic in the security parameter), then $\mathsf{T_H}$ holds a secure IND-qCCA scheme in the ROM, given that the underlying PKE is OW-CPA.

Theorem 3. *We consider a random oracle $H : \{0,1\}^* \mapsto \{0,1\}^n$. Let* KEM *be the KEM resulting from applying the* $\mathsf{T_H}$ *transform to a PKE* PKE *(which never queries H). Then, for any IND-qCCA adversary $\mathcal{A}$ that makes at most q_H queries to H, there exists a OW-CPA adversary $\mathcal{B}$ s.t.*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathcal{A}) \leq q_H \cdot ((q_H+1)(q_H+2))^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-cpa}}(\mathcal{B})\ .$$

```
Oracle O^Dec(ct)
if ct = ct* : return ⊥
if more than q queries :
  return ⊥
σ' ← dec^p(sk, ct)
if σ' = ⊥ : return ⊥
return H(σ', ct)

H(σ, ct)
if ∃h s.t. ((σ, ct), h) ∈ L_H :
  return h
h ←$ {0,1}^n
L_H ← L_H ∪ {((σ, ct), h)}
return h
```

```
Oracle O^Dec'(ct)
if ct = ct* : return ⊥
if more than q queries : return ⊥
if ∃K s.t. (ct, K) ∈ L_K :
  return K
i ← O^i(L_H, ct)
if i = ⊥_d : return ⊥
if i ≠ ⊥ :
  ((σ_i, ct_i), K_i) ← L_H[i]
  return K_i        // return i-th valued returned by H'
K ←$ {0,1}
L_K ← L_K ∪ {(ct, K)}
return K

H'(σ, ct)
if ∃h s.t. ((σ, ct), h) ∈ L_H :
  return h
if ∃K s.t. (ct, K) ∈ L_K :
  if O^PCO(σ, ct) :
    L_H ← L_H ∪ {((σ, ct), K)}
    return K
h ←$ {0,1}^n
L_H ← L_H ∪ {((σ, ct), h)}
return h
```

Fig. 9: Original and modified oracles for the proof of Thm 3.

If PKE *is deterministic, we get*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathcal{A}) \leq \delta + ((q_H+1)(q_H+2))^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-cpa}}(\mathcal{B}) \ .$$

Proof. We start by defining an oracle $\mathcal{O}^{\mathsf{i}}(\mathcal{L}_H, \mathsf{ct})$ (see Fig. 8). This oracle returns the index i s.t. $((\sigma_i, \mathsf{ct}_i), K_i) \in \mathcal{L}_H$ (we first sort $\mathcal{L}_H$ according to some fixed order) and $\mathsf{ct}_i = \mathsf{ct}$ and $\mathsf{dec}^{\mathsf{p}}(\mathsf{sk}, \mathsf{ct}_i) = \sigma_i$. If such a i does not exist and $\mathsf{dec}^{\mathsf{p}}(\mathsf{sk}, \mathsf{ct}_i) = \perp$ it returns $\perp_d$, otherwise it returns $\perp$.

Now we show how to simulate the IND-qCCA decapsulation oracle in the ROM, using $\mathcal{O}^i$ and $\mathcal{O}^{\mathsf{PCO}}$ only. The original (resp. modified) oracles $\mathcal{O}^{\mathsf{Dec}}$ and H (resp. $\mathcal{O}^{\mathsf{Dec}'}$ and H') are on the left (resp. right) in Fig. 9. We now prove that any IND-qCCA adversary cannot distinguish between the real and modified oracles.

First, we show that the outputs of the ROs H and H' on any query (σ, ct) have the same distribution, given the adversary's view. We break this into four subcases:

- (σ, ct) was queried before to H (resp. H'): In this case, both H and H' return the value h returned on the previous similar query. Thus, we assume from now on that every RO query made by the adversary is unique.
- ct was never queried to the decapsulation oracle before: In this case, both H and H' return a random value h and store the query/response in $\mathcal{L}_H$.
- ct was queried to the decapsulation oracle before: In both cases (original and modified oracles) one can see that if the decryption of ct either fails or $\sigma' = \mathsf{dec}^\mathsf{p}(\mathsf{sk}, \mathsf{ct})$ is different from σ, then the output of the decapsulation oracle is independent of $H(\sigma, \mathsf{ct})$ (and $H'(\sigma, \mathsf{ct})$). In both cases, the ROs sample a fresh value (H' will do so because $\mathcal{O}^{\mathsf{PCO}}(\sigma, \mathsf{ct})$ will output 0 in this case, as $\sigma \neq \sigma'$ or the ciphertext is not valid). Now, if ct decrypts to σ, the original decapsulation oracle outputs $H(\sigma, \mathsf{ct})$. In the modified game, the decapsulation oracle outputs a random K. Indeed, as we assume (σ, ct) was never queried to H, $\mathcal{O}^\mathsf{i}(\mathcal{L}_H, \mathsf{ct})$ outputs $\perp$. Then, the modified RO will output the same K, as $\mathcal{O}^{\mathsf{PCO}}(\sigma, \mathsf{ct})$ will verify. In both cases, the ROs output the same value as the decapsulation oracle.

We now show that the decapsulation oracles $\mathcal{O}^{\mathsf{Dec}}$ and $\mathcal{O}^{\mathsf{Dec}'}$ are indistinguishable. Let ct be the queried ciphertext and $\sigma = \mathsf{dec}^\mathsf{p}(\mathsf{sk}, \mathsf{ct})$.

- $\mathsf{ct} = \mathsf{ct}^*$: both oracles return $\perp$.
- $\sigma = \perp$: Both oracles return $\perp$, as $\mathcal{O}^\mathsf{i}(\mathcal{L}_H, \mathsf{ct})$ returns $\perp_d$.
- $H(\sigma, \mathsf{ct})$ (resp. $H'(\sigma, \mathsf{ct})$) was never queried. Both oracles return a random value if ct was never queried, or a consistent value if it was. It is straightforward to see this is the case in the original oracle. In the modified oracle, as $H'(\sigma, \mathsf{ct})$ was never queried, we have $\mathcal{O}^\mathsf{i}(\mathcal{L}_H, \mathsf{ct})$ that returns $\perp$. Thus, the decapsulation oracle returns a random K if ct was not queried or a consistent K if it was.
- $H(\sigma, \mathsf{ct})$ (resp. $H'(\sigma, \mathsf{ct})$) was queried and it output K. Both oracles return K. In the modified decapsulation oracle, $\mathcal{O}^\mathsf{i}(\mathcal{L}_H, \mathsf{ct})$ will output a valid i s.t. $H'(\sigma_i, \mathsf{ct}) = h_i$ and h_i is returned. Thus, the answer is consistent with the RO.

Now we can prove the theorem by game hopping as before. We define Γ^0 as the original IND-qCCA game.

$\underline{\Gamma^1 :}$ We modify the original IND-qCCA game into another game Γ^1 where the random/decapsulation oracles are the modified ones (i.e. H' and $\mathcal{O}^{\mathsf{Dec}'}$) described above. As shown, both games are indistinguishable and thus

$$|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]| = 0 \ .$$

$\underline{\Gamma^2 :}$ We replace the challenge key by a random one, as in the previous proof. Then, similarly, the real key is indistinguishable from a random one unless $H(\sigma^*, \mathsf{ct}^*)$ is queried. We define this event as query and

$$|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]| \leq \Pr[\mathsf{query}] \ .$$

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

init $\mathcal{L}_H, \mathcal{L}_K \leftarrow \emptyset$
init $\mathcal{L}_q \leftarrow []$
$K^* \leftarrow\$ \mathcal{K}$
run $\mathcal{A}^{H'', \mathcal{O}^{\mathsf{Dec}''}}(\mathsf{pk}, \mathsf{ct}^*, K^*)$
sample random query (σ', ct') made to H''
return σ'

$H''(\sigma, \mathsf{ct})$

$i_q \leftarrow$ query number
if $\exists h$ s.t. $((\sigma, \mathsf{ct}), h) \in \mathcal{L}_H$:
  return h
if $\exists K$ s.t. $(\mathsf{ct}, K) \in \mathcal{L}_K$:
  if $\mathcal{L}_q[\mathsf{ct}] = i_q$:
    $\mathcal{L}_H \leftarrow \mathcal{L}_H \cup \{((\sigma, \mathsf{ct}), K)\}$
    return K
$h \leftarrow\$ \{0,1\}^n$
$\mathcal{L}_H \leftarrow \mathcal{L}_H \cup \{((\sigma, \mathsf{ct}), h)\}$
return h

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

if $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
if more than q queries : **return** $\perp$
if $\exists K$ s.t. $(\mathsf{ct}, K) \in \mathcal{L}_K$:
  return K
$i \leftarrow\$ \{1, \ldots, q_H, \perp, \perp_d\}$
if $i = \perp_d$: **return** $\perp$
if $i \neq \perp$:
  $(\mathsf{ct}_i, K_i) \leftarrow \mathcal{L}_H[i]$
  return K_i // return i-th valued returned by H''
$K \leftarrow\$ \{0,1\}$
$\mathcal{L}_K \leftarrow \mathcal{L}_K \cup \{(\mathsf{ct}, K)\}$
$\mathcal{L}_q[\mathsf{ct}] \leftarrow\$ \{0, \ldots, q_H\}$
return K

Fig. 10: $\mathcal{B}$ adversary for the proof of Thm 3.

We can upper bound this probability by the advantage of a OW-CPA adversary $\mathcal{B}$ against PKE. That is, given a IND-qCCA adversary playing game Γ^2, we build an adversary $\mathcal{B}$ as shown in Fig. 10. One can see that if $\mathcal{B}$ was simulating $\mathcal{A}$ with the H' and $\mathcal{O}^{\mathsf{Dec}'}$ oracles (instead of its own oracles H'' and $\mathcal{O}^{\mathsf{Dec}''}$), the simulation would be perfect as long as query did not occur. Then, whenever query would happen, $\mathcal{B}$ would recover σ^* with prob. $\frac{1}{q_H}$. Now $\mathcal{B}$ does not simulate the modified oracles perfectly but instead makes some guessing in its own oracles H'' and $\mathcal{O}^{\mathsf{Dec}''}$:

- $\mathcal{O}^{\mathsf{Dec}''}$: In line 5, i is picked at random instead of being the returned value of the $\mathcal{O}^{\mathsf{i}}$ oracle. On each query the simulation is perfect with prob. $1/(q_H+2)$ and overall with probability $\frac{1}{(q_H+2)^q}$, as there are at most q queries to this oracle. In line 11, we associate a random index to each ct s.t. $(\mathsf{ct}, *) \in \mathcal{L}_K$.
- H'': In line 5, when $(\mathsf{ct}, *) \in \mathcal{L}_K$, instead of querying the plaintext-checking oracle we check whether the corresponding sampled index $\mathcal{L}_q[\mathsf{ct}]$ is equal to the query number. If it is, we reply with K s.t. $(\mathsf{ct}, K) \in \mathcal{L}_K$ otherwise we proceed as before (i.e. as in H'). Let's assume w.l.o.g that each query to H'' is unique. For each ct s.t. $(\mathsf{ct}, *) \in \mathcal{L}_K$, there can be at most one query (σ, ct) s.t. $\mathcal{O}^{\mathsf{PCO}}(\sigma, \mathsf{ct})$ returns 1 (it is when σ is the decryption of ct). Here, $\mathcal{B}$ guesses beforehand which query it is (or if no such query will be made) and gets the correct answer with prob. $\frac{1}{q_H+1}$. Note that $\mathcal{B}$ needs to make one guess per query to $\mathcal{O}^{\mathsf{Dec}''}$ (not per query to H''). Overall, the probability H'' simulates correctly H' is $\frac{1}{(q_H+1)^q}$.

From this we can deduce that $\mathcal{B}$ correctly simulates Γ^2 with probability $\frac{1}{((q_H+1)(q_H+2))^q}$ and wins the OW-CPA game with prob. at least $\frac{1}{q_H} \cdot \Pr[\mathsf{query}]$. Hence,

$$|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]| \leq \Pr[\mathsf{query}] \leq q_H \cdot ((q_H+1)(q_H+2))^q \cdot \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}}(\mathcal{B}) \ .$$

Note that when PKE is deterministic, in order to recover σ^*, $\mathcal{B}$ can check which σ' queried is s.t. $\mathsf{enc}(\mathsf{pk}^*, \sigma') = \mathsf{ct}^*$ instead of guessing. This works as long as the challenge seed σ^* and queried seeds are correct. If that is not the case, one can build an adversary that wins the correctness game defined in Fig. 1. Note that this adversary knows which will be the correct seed as it is given sk and the PKE is deterministic. As the correctness advantage is upper bounded by δ, we obtain that for deterministic PKEs the last inequality becomes

$$|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]| \leq \Pr[\mathsf{query}] \leq \delta + ((q_H+1)(q_H+2))^q \cdot \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}}(\mathcal{B}) \ .$$

Finally, in game Γ^2, the challenge key is always random and thus $\Pr[\Gamma^2 \Rightarrow 1] = \frac{1}{2}$. Collecting the probabilities holds the result. □

4 CPA-security is sufficient for TLS 1.3 in the ROM

We show in this section that a CPA-secure KEM is sufficient for the handshake in TLS 1.3 to be secure in the ROM. The security bound is very loose, but this still solves an interesting open problem. TLS 1.3 only supports DH key-exchange but it can be trivially modified to support KEMs as done in several PQ variants of TLS (e.g.[1,7]). That is, the client runs $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\, \mathsf{gen}$ and sends pk as its share (instead of g^x). Then, the server runs $K, \mathsf{ct} \leftarrow\!\$\, \mathsf{encaps}$ and sends ct as its secret share (instead of g^y). Finally, the client runs $K \leftarrow \mathsf{decaps}(\mathsf{sk}, \mathsf{ct})$ and the shared secret is set to K. By abuse of language, we refer to this modified protocol as TLS 1.3 in what follows. An overview of this modified handshake is given in Fig. 14.

4.1 IND-1CCA-MAC

In order to show that a CPA-secure KEM is sufficient for TLS 1.3 to be secure, we first introduce an intermediary notion of security for KEMs, called IND-1CCA-MAC. This security definition has no application and will serve only as a useful intermediary building block for the proof.

Definition 6 (IND-1CCA-MAC). *We consider the games defined in Fig. 11. Let $\mathcal{K}$ be the key space, G, H_1, H_2, H_3, H_4 and H_D be key-derivation functions with images in $\{0,1\}^n$, H_T be a hash function with images in $\{0,1\}^n$, and MAC a MAC scheme. A KEM scheme $\mathsf{KEM} = (\mathsf{gen}, \mathsf{encaps}, \mathsf{decaps})$ is* IND-1CCA-MAC *if for any ppt adversary $\mathcal{A}$ we have*

$$\mathsf{Adv}^{\text{ind-1cca-mac}}_{KEM}(\mathcal{A}) := \left| \Pr\left[\mathrm{IND-1CCA-MAC}_{\mathsf{KEM}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda)$$

IND-1CCA-MAC$_{\mathsf{KEM}}(\mathcal{A})$

$b \leftarrow\$ \{0,1\}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{gen}()$
$\mathsf{ct}^*, K^* \leftarrow\$ \mathsf{encaps}(\mathsf{pk})$
$n^* \leftarrow\$ \{0,1\}^n$
$\mathsf{HS}^* \leftarrow G(K^*)$
$\mathsf{CHTS}_0 \leftarrow H_1(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$
$\mathsf{SHTS}_0 \leftarrow H_2(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$
$\mathsf{dHS}_0 \leftarrow H_3(\mathsf{HS}^*)$
$(\mathsf{CHTS}_1, \mathsf{SHTS}_1, \mathsf{dHS}_1) \leftarrow\$ \{0,1\}^{3n}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Dec}}, \mathcal{O}^{\mathsf{Dec}}_{\mathsf{MAC}}}(\mathsf{pk}, \mathsf{ct}^*, n^*, (\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b))$
return $1_{b'=b}$

Oracle $\mathcal{O}^{\mathsf{Dec}}((\mathsf{ct}, n))$

if more than 1 query : **return** $\perp$
if $(\mathsf{ct}, n) = (\mathsf{ct}^*, n^*)$: **return** $\perp$
$K' \leftarrow \mathsf{decaps}(\mathsf{sk}, \mathsf{ct})$
if $K' = \perp$: **return** $\perp$
$\mathsf{HS}' \leftarrow G(K')$
$\mathsf{CHTS} \leftarrow H_1(\mathsf{HS}', H_T(\mathsf{ct}, n))$
$\mathsf{SHTS} \leftarrow H_2(\mathsf{HS}', H_T(\mathsf{ct}, n))$
$\mathsf{tk_c} \leftarrow H_D(\mathsf{CHTS}); \mathsf{tk_s} \leftarrow H_D(\mathsf{SHTS})$
return $(\mathsf{tk_c}, \mathsf{tk_s})$

Oracle $\mathcal{O}^{\mathsf{Dec}}_{\mathsf{MAC}}(\mathsf{ct}, n, \mathsf{tag}, \mathsf{txt})$

if more than 1 query : **return** $\perp$
if $(\mathsf{ct}, n) = (\mathsf{ct}^*, n^*)$: **return** $\perp$
$K' \leftarrow \mathsf{decaps}(\mathsf{sk}, \mathsf{ct})$
$\mathsf{HS}' \leftarrow G(K'); \mathsf{SHTS} \leftarrow H_2(\mathsf{HS}', H_T(\mathsf{ct}, n))$
$\mathsf{fk}_S \leftarrow H_4(\mathsf{SHTS})$
if $\mathsf{MAC.Vrf}(\mathsf{fk}_S, \mathsf{txt}, \mathsf{tag}) = \mathbf{true}$:
  return HS'
return $\perp$

Fig. 11: IND-1CCA-MAC game.

where $\Pr\left[\mathrm{IND-1CCA-MAC}^b_{\mathsf{KEM}}(\mathcal{A}) \Rightarrow 1\right]$ *is the probability that* $\mathcal{A}$ *wins the IND-1CCA-MAC*$^b_{\mathsf{KEM}}(\mathcal{A})$ *game defined in Fig. 11.*

In this game, the adversary receives a challenge ciphertext encapsulating a key K, a nonce n^*, and either three secrets $(\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b)$ derived from K through a key schedule, or three random secrets. Jumping ahead, these three values are computed (nearly) in the same as way as their identically named counterparts in the modified TLS 1.3 protocol. The adversary has also access to two oracles that it can query *at most once*. The first is simply a decapsulation oracle that applies a key schedule (similar to TLS's) on the decapsulated key and returns two secrets $\mathsf{tk_c}$ and $\mathsf{tk_s}$. The second oracle takes a ciphertext (which must be different than the challenge ciphertext), a tag, and some data. Then, the ciphertext is decrypted to recover a secret HS' that is passed through a key schedule to get a MAC key fk_S. Finally, the oracle checks whether tag is a valid MAC on the data with the key fk_S. If this is the case it returns HS', otherwise it returns an error $\perp$. Informally, this last oracle outputs the root secret HS if the adversary can forge a valid tag corresponding to the tuple (ct, n). In the TLS proof, this will be used to argue that if a participant can send a valid tag, it should know the root secret HS.

4.2 OW-CPA implies IND-1CCA-MAC

First, we briefly define the notion of MAC unforgeability we will need.

Definition 7 (MAC EUF-0T). *Let* $\mathsf{MAC} = (\mathsf{MAC.Vrf}, \mathsf{MAC.Tag})$ *be a message authentication code scheme (MAC). We say* MAC *is EUF-0T if for any ppt adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathrm{euf-0t}}_{\mathsf{MAC}}(\mathcal{A}) := \Pr[\mathsf{MAC.Vrf}(K, M, T) = 1 : (M, T) \leftarrow\$ \mathcal{A}; K \leftarrow\$ \mathcal{K}]$$

is negligible in the security parameter, where the probability is taken over the sampling of the key and the randomness of the adversary.

We now prove that any OW-CPA KEM is also IND-1CCA-MAC secure in the ROM if the MAC used is EUF-0T secure. More precisely, the KDFs G, H_1, H_2, H_3, H_4 and H_D, and the hash function H_T in the IND-1CCA-MAC games are assumed to be ROs.

Theorem 4. *Let* $\mathsf{KEM} = (\mathsf{gen}, \mathsf{encaps}, \mathsf{decaps})$ *be a KEM. Let the KDFs and the hash function in the IND-1CCA-MAC game be modelled as random oracles. Then, for any ppt adversary* $\mathcal{A}$ *making at most* $q_G, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4} q_{H_D}, q_{H_T}$ *queries to* $G, H_1, H_2, H_3, H_4, H_D, H_T$ *respectively, there exists a OW-CPA adversary* $\mathcal{B}$ *s.t.*

$$\mathsf{Adv}^{\text{ind-1cca-mac}}_{\mathsf{KEM}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{euf-0t}}_{\mathsf{MAC}}(\mathcal{B}) + \frac{3q_{H_1} + 4q_{H_2} + q_{H_3} + q_{H_4} + q_{H_D} + 1}{2^n} + \frac{(q_{H_T} + 4)^2}{2^n} + q_G(q_{H_1} + 2)^2(q_{H_2} + 2)^3 \cdot \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{KEM}}(\mathcal{C}) \ ,$$

where $\mathcal{B}$ *has approximately the same running time as* $\mathcal{A}$*.*

Proof. The first step of the proof is very similar to the proof of Theorem 3. Indeed, one can see that the decapsulation oracle outputs secrets that are computed as (a function of) $H_i(\mathsf{HS}, H_T(\mathsf{ct}, n))$, where H_i and H_T are ROs. Note that the only difference is that H_T is applied on (ct, n). However, as H_T is a RO, this difference will not matter much in the proof. Hence, as in Theorem 3, one can program the ROs s.t. the decapsulation oracle $\mathcal{O}^{\mathsf{Dec}}$ can be simulated without the secret key. In a second step, we show that the adversary can also simulate the $\mathcal{O}^{\mathsf{Dec}}_{\mathsf{MAC}}$ oracle with good probability. More precisely, let HS be the secret corresponding to the submitted ciphertext ct. Then, either $H_2(\mathsf{HS}, H_T(\mathsf{ct}, n))$ has been queried by the adversary or it is very unlikely that $\mathcal{A}$ knows the MAC key fk_S. In the first case we can recover HS from the list of queries, and in the second we can return $\perp$ as most likely the MAC verification will fail.

We proceed with a sequence of games, which are given in detail in Fig. 12.

$\underline{\Gamma^0:}$ This is the original IND-1CCA-MAC game. From now on, we assume w.l.o.g. that each query to ROs are unique (i.e. they never repeat).

$\Gamma^{0\text{-}6}_{\mathsf{KEM}}(\mathcal{A})$

$b \leftarrow\$ \{0,1\}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{gen}()$
$\mathsf{ct}^*, K^* \leftarrow\$ \mathsf{encaps}(\mathsf{pk})$
$n^* \leftarrow\$ \{0,1\}^n$
$\mathsf{HS}^* \leftarrow G(K^*)$
$\mathsf{CHTS}_0 \leftarrow H_1(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$
$\mathsf{SHTS}_0 \leftarrow H_2(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$
$\mathsf{dHS}_0 \leftarrow H_3(\mathsf{HS}^*)$
$(\mathsf{CHTS}_1, \mathsf{SHTS}_1, \mathsf{dHS}_1) \leftarrow\$ \{0,1\}^{3n}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Dec}}, \mathcal{O}^{\mathsf{Dec}}_{\mathsf{MAC}}, H_1, H_2}(\mathsf{pk}, \mathsf{ct}^*, n^*, (\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b))$ // Γ^0-Γ^3
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Dec}'}, \mathcal{O}^{\mathsf{Dec}}_{\mathsf{MAC}}, H_1', H_2'}(\mathsf{pk}, \mathsf{ct}^*, n^*, (\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b))$ // Γ^4-
if collision on H_T : **abort** // Γ^1-
if $\mathcal{A}$ queries $H_i(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*)),\ i \in [2]$ or $H_3(\mathsf{HS}^*)$:
  abort // Γ^6
  if $\mathcal{A}$ did not query $G(K^*)$: **abort** // Γ^5
return $1_{b'=b}$

Oracle $\mathcal{O}^{\mathsf{Dec}}_{\mathsf{MAC}}(\mathsf{ct}, n, \mathsf{tag}, \mathsf{txt})$

if more than 1 query : **return** $\bot$
if $(\mathsf{ct}, n) = (\mathsf{ct}^*, n^*)$: **return** $\bot$
$K' \leftarrow \mathsf{decaps}(\mathsf{sk}, \mathsf{ct})$
$\mathsf{HS}' \leftarrow G(K'); \mathsf{SHTS} \leftarrow H_2(\mathsf{HS}', H_T(\mathsf{ct}, n))$
$\mathsf{fk}_S \leftarrow H_4(\mathsf{SHTS})$
if $\mathsf{SHTS} = \mathsf{SHTS}_b$: // Γ^2-
  abort // Γ^2-
if $\mathsf{MAC.Vrf}(\mathsf{fk}_S, \mathsf{txt}, \mathsf{tag}) = \mathbf{true}$:
  if $\mathcal{A}$ did not query $H_4(\mathsf{SHTS})$: // Γ^2-
    abort // Γ^2-
  if $\mathcal{A}$ did not query $H_2(\mathsf{HS}', H_T(\mathsf{ct}, n))$: // Γ^3-
    abort // Γ^3-
  return HS'
return $\bot$

$H_j(\mathsf{HS}, y),\ j \in \{1,2\}$

if $\nexists(\mathsf{ct}, n)$ s.t. $((\mathsf{ct}, n), y) \in \mathcal{L}_{H_T}$: // Γ^1-
  $h \leftarrow\$ \{0,1\}^n$; **return** h // Γ^1-
usual lazy sampling

Fig. 12: Games for the proof of Thm 4. The adversary has access to all the other ROs G, H_3, H_4 and H_D, even if it is not explicited in the games. H_1', H_2' and $\mathcal{O}^{\mathsf{Dec}'}$ are defined in Fig. 13.

Γ^1 : We modify the previous game as follows. First, we abort if a collision on H_T occurs in the game. As there are at most $q_{H_T} + 4$ queries to H_T in the game, a collision occurs with prob. less than $\frac{(q_{H_T}+4)^2}{2^n}$. Then, on adversary's queries $H_j(\mathsf{HS}, y),\ j \in \{1,2\}$, if $H_T(\mathsf{ct}, n) = y$ was never queried by $\mathcal{A}$ for some (ct, n), we mark y as *unpaired* and return a random value. The only way it differs from the previous game, is if a query $H_j(\mathsf{HS}, y)$ for an *unpaired* y is performed by the game (i.e. not by the adversary), either before or after y was marked as *unpaired*. Now, the $\mathcal{A}$ does not get any information about values $H_T(\mathsf{ct}, n)$ from the game (or oracles), except a few values $H_j(\mathsf{HS}, H_T(\mathsf{ct}, n))$ (or values that depends on these), for some HS. Note that these values completely "hide" the result of the H_T query, as H_j is a RO. Hence, the best strategy for $\mathcal{A}$ to query $H_j(\mathsf{HS}, y)$ s.t. y is *unpaired* but is queried by the game at some point, is to try random values for y. As the game makes at most 2 queries to H_1 (one in the challenge part and one in the decapsulation oracle) and 3 queries to H_2 (one in the challenge part and one in each oracle), the probability that a random unpaired y is s.t. y was the result of a H_T query by the game at some point, is at most $\frac{2}{2^n}$ for a H_1 call, and $\frac{3}{2^n}$ for a H_2 call. Overall, we have

$$|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]| \leq \frac{2q_{H_1} + 3q_{H_2}}{2^n} .$$

We note that this step ensures that on a query $H_j(\mathsf{HS}, y)$, one can recover a unique tuple (ct, n) s.t. $H_T(\mathsf{ct}, n) = y$, or a random value is returned.

$\underline{\Gamma^2:}$ We modify the original game s.t. we abort whenever the MAC verification succeeds on the query $\mathcal{O}^{\mathsf{Dec}}_{\mathsf{MAC}}(\mathsf{ct}, n, \mathsf{tag}, \mathsf{txt})$ but $\mathsf{fk}_S := H_4(\mathsf{SHTS})$ was never queried, where $\mathsf{SHTS} := H_2(G(K), H_T(\mathsf{ct}, n))$ and $K := \mathsf{decaps}(\mathsf{sk}, \mathsf{ct})$. If that is the case, it means the MAC key $\mathsf{fk}_S := H_4(\mathsf{SHTS})$ is indistinguishable from a random value for $\mathcal{A}$, but it managed to forge a valid tag. Thus, one can build an adversary $\mathcal{B}$ that breaks MAC unforgeability. More formally, $\mathcal{B}$ samples a pair of keys $(\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathsf{gen}$, generates a valid input for $\mathcal{A}$ and simulates the decryption oracle with the secret key. Then, when $\mathcal{A}$ submits $(\mathsf{ct}, n, \mathsf{tag}, \mathsf{txt})$ to $\mathcal{O}^{\mathsf{Dec}}_{\mathsf{MAC}}$, $\mathcal{B}$ outputs $(\mathsf{txt}, \mathsf{tag})$ as a forgery. We also abort if the value SHTS computed in the oracle is s.t. $\mathsf{SHTS} = \mathsf{SHTS}_b$. As there are no collision on H_T and $(\mathsf{ct}, n) \neq (\mathsf{ct}^*, n^*)$, this happens with probability at most $\frac{1}{2^n}$. Then, we have

$$|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathrm{euf-0t}}_{\mathsf{MAC}}(\mathcal{B}) + \frac{1}{2^n} \ .$$

$\underline{\Gamma^3:}$ We abort whenever the MAC verification succeeds on the query $\mathcal{O}^{\mathsf{Dec}}_{\mathsf{MAC}}(\mathsf{ct}, n, \mathsf{tag}, \mathsf{txt})$ but $H_2(G(K), H_T(\mathsf{ct}, n))$ was never queried, where $K := \mathsf{decaps}(\mathsf{sk}, \mathsf{ct})$. By the previous game, it means that the adversary queried $\mathsf{SHTS} := H_2(G(K), H_T(\mathsf{ct}, n))$ to H_4 without having queried $H_2(G(K), H_T(\mathsf{ct}, n))$ beforehand. If we analyse what information $\mathcal{A}$ has about $\mathsf{SHTS} \neq \mathsf{SHTS}_b$ if it did not query $H_2(G(K), H_T(\mathsf{ct}, n))$, we see that the only potential "leakage" is from a decapsulation query that returns $\mathsf{tk_s} := H_D(\mathsf{SHTS})$, where H_D is a RO perfectly hiding SHTS.

Thus, the best strategy for $\mathcal{A}$ to find SHTS without querying H_2 is to query random values $x \in \{0,1\}^n$ to H_D or H_4 until it finds x s.t. $H_D(x) = \mathsf{tk_s}$ or $H_4(x) = \mathsf{fk}_S$. This happens with probability at most $\frac{q_{H_D}+q_{H_4}}{2^n}$. Hence, we have

$$|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^3 \Rightarrow 1]| \leq \frac{q_{H_D} + q_{H_4}}{2^n} \ .$$

$\underline{\Gamma^4:}$ We program both ROs H_1 and H_2 s.t. we can perfectly simulate the decapsulation oracle with an oracle $\mathcal{O}^{\mathsf{i}}_G$. This follows exactly the idea of the proof of Theorem 3. First, we introduce an oracle $\mathcal{O}^{\mathsf{i}}_G$ in Fig. 13 that takes a list of RO queries, a nonce n and a ciphertext ct, checks whether $(G(K), H_T(\mathsf{ct}, n))$ (where K is the key encapsulated in ct) was ever queried and if that is the case, the index of the corresponding query. This is exactly the same as the oracle $\mathcal{O}^{\mathsf{i}}$ in the proof of Thm 3, except we query the decapsulated K to the RO G and there is the additional nonce. Then, we can program the ROs $H_j,\ j \in \{1,2\}$ and simulate the (1-time) decapsulation oracle as shown in Fig. 13.

The simulation works nearly as in the proof of Thm 3. Let ct be the unique decapsulation query, $K := \mathsf{decaps}(\mathsf{sk}, \mathsf{ct})$ and $\mathsf{HS} := G(K)$. For $j \in [2]$, the

Oracle $\mathcal{O}^{\mathsf{Dec}'}(\mathsf{ct}, n)$

```
if (ct, n) = (ct*, n*) : return ⊥
if more than 1 query : return ⊥
q_1 ←$ {0, ..., q_{H_1}}
q_2 ←$ {0, ..., q_{H_2}}
i ← O^i(L_{H_1}, ct, n)
if i = ⊥_d : return ⊥
if i ≠ ⊥ :
  // get i-th valued returned by H_1
  ((HS_i, ct_i, n_i), h_i) ← L_{H_1}[i]
  CHTS ← h_i
else :
  CHTS ←$ {0, 1}
L_K^1 ← (ct, n, CHTS)
i ← O^i(L_{H_2}, ct, n)
if i ≠ ⊥ :
  // get i-th valued returned by H_2
  ((HS_i, ct_i, n_i), h_i) ← L_{H_2}[i]
  SHTS ← h_i
else :
  SHTS ←$ {0, 1}
L_K^2 ← (ct, n, SHTS)
return (H_D(CHTS), H_D(SHTS))
```

$H'_j(\mathsf{HS}, y),\ j \in \{1, 2\}$

```
if ∄(ct, n) s.t. ((ct, n), y) ∈ L_{H_T} :
  h ←$ {0, 1}^n; return h
set (ct, n) s.t. ((ct, n), y) ∈ L_{H_T}
if L_K^j = (ct, n, h) for some h :
  if HS = G(decaps(sk, ct)) :
    L_{H_j} ← L_{H_j} ∪ {((HS, ct, n), h)}
    return h
h ←$ {0, 1}^n
L_{H_j} ← L_{H_j} ∪ {((HS, ct, n), h)}
return h
```

$\mathcal{O}^{\mathsf{i}}_G(\mathcal{L}, n, \mathsf{ct})$

```
sort L according to query order :
  L = ((HS_i, ct_i, n_i), h_i)_{i∈{1,...,|L_H|}}
K' ← decaps(sk, ct)
if K' = ⊥ : return ⊥_d
HS' ← G(K')
for i ∈ {1, ..., |L|} :
  if (ct_i, n_i) = (ct, n) and HS' = HS_i :
    return i
return ⊥
```

Fig. 13: Simulation of decapsulation and random oracles with sub-oracle $\mathcal{O}^{\mathsf{i}}_G$ for the proof of Thm 4. Note that as we assume that each query to H_j is unique, H'_j does not check whether a query was previously made.

simulated decapsulation oracle checks whether $(G(K), H_T(\mathsf{ct}, n))$ was already queried to H_j using $\mathcal{O}^{\mathsf{i}}_G$, if that is the case it recovers the corresponding value, otherwise it means $H_j(\mathsf{HS}, H_T(\mathsf{ct}, n))$ was never queried by the adversary *nor* the challenger, as $(\mathsf{ct}, n) \neq (\mathsf{ct}^*, n^*)$. Thus it samples the hash value at random, queries it to H_D and returns it to the adversary.

The simulation of H_j is such that it is consistent with the values returned by the simulated decapsulation oracle. First, if $H_j(\mathsf{HS}, y)$ is queried s.t. y is *unpaired*, we can simply return a random value, this is consistent with the game. Then, if y is not *unpaired*, one can recover the unique (as there are no collision) tuple (ct, n) s.t. $y = H_T(\mathsf{ct}, n)$. We consider from now on only queries with y s.t. $H_T(\mathsf{ct}, n) = y$ for some (ct, n). On a query $H_j(\mathsf{HS}, H_T(\mathsf{ct}, n))$, if (ct, n) was already queried to the decapsulation oracle, then $h := H_j(\mathsf{HS}, \mathsf{ct}, n)$ was set by $\mathcal{O}^{\mathsf{Dec}'}$ iff $\mathsf{HS} = G(K)$, where $K := \mathsf{decaps}(\mathsf{sk}, \mathsf{ct})$. Hence, we return the same K if $G(\mathsf{decaps}(\mathsf{sk}, \mathsf{ct})) = \mathsf{HS}$. Otherwise we sample a random value and return it. Note that this is the only place where the secret key sk is used anymore (except implicitly in the $\mathcal{O}^{\mathsf{i}}_G$ oracle). The simulation is perfect and therefore we have

$$|\Pr[\Gamma^3 \Rightarrow 1] - \Pr[\Gamma^4 \Rightarrow 1]| = 0 .$$

$\underline{\Gamma^5 :}$ In game Γ^5, we abort whenever the adversary did not query $G(K^*)$ (which is equal to HS^*) but it queried $H_1(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$, $H_2(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$ or $H_3(\mathsf{HS}^*)$. Note that the (modified) decryption oracle never queries $H_1(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$, $H_2(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$ or $H_3(\mathsf{HS}^*)$. In addition, the challenge values given to $\mathcal{A}$ are either perfectly random or completely hide HS^*. Thus, the probability that $\mathcal{A}$ queries HS^* to H_1, H_2 or H_3 is upper bounded by $\frac{q_{H_1}+q_{H_2}+q_{H_3}}{2^n}$ and hence we have

$$|\Pr[\Gamma^4 \Rightarrow 1] - \Pr[\Gamma^5 \Rightarrow 1]| \leq \frac{q_{H_1} + q_{H_2} + q_{H_3}}{2^n} .$$

$\underline{\Gamma^6 :}$ Finally, in game Γ^6 we abort whenever $H_1(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$, $H_2(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$ or $H_3(\mathsf{HS}^*)$ is queried by the adversary. Let query be this event. By the previous game, it means that K^* was queried to G before query happens. Finally, as in the previous proofs, we can upper bound $\Pr[\mathsf{query}]$ by the advantage of a OW-CPA adversary times a constant. The challenge keys $(\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b)$ are sampled at random in the reduction, as long query does not happen both the real and random cases are perfectly indistinguishable. The only challenge for such a OW-CPA adversary $\mathcal{C}$ is to simulate the oracles without having access to the secret key. It proceeds as follows (see the full version of the paper [20] for a complete and detailed presentation of $\mathcal{C}$).

- $\mathcal{O}^{\mathsf{Dec}''}_{\mathsf{MAC}}$: This oracle returns something else than $\perp$ iff $(\mathsf{HS}, H_T(\mathsf{ct}, n))$ was queried to H_2, where $\mathsf{HS} := G(K)$ and $K := \mathsf{decaps}(\mathsf{sk}, \mathsf{ct})$. Hence, one can simply pick a random value $r \leftarrow\$ \{0, \ldots, q_{H_2}\}$ and guess whether $\mathcal{O}^{\mathsf{Dec}}_{\mathsf{MAC}}(\mathsf{ct})$ fails (if $r = 0$) or succeeds and HS is in the r-th query made to H_2. In the latter case, one can recover HS in the r-th query and return it. Overall the simulation works with probability $\frac{1}{q_{H_2}+1}$.
- $\mathcal{O}^{\mathsf{Dec}''}$: In this oracle, the secret-key is used only in the $\mathcal{O}^{\mathsf{i}}_G$ sub-oracle. A reply of $\mathcal{O}^{\mathsf{i}}_G$ is in the set $\{\perp, \perp_d, 1, \ldots, q_{H_j}\}$ for $j \in [2]$. Thus, one can guess the correct reply by sampling a random value in that set, which gives a success probability of $\frac{1}{(q_{H_j}+2)}$. Overall, there are at most 2 calls to $\mathcal{O}^{\mathsf{i}}_G$ (one for $j = 1$ and $j = 2$) and therefore the probability that the simulation is successful is $\frac{1}{(q_{H_1}+2)(q_{H_2}+2)}$.
- H''_j, $j \in [2]$: The only time the secret key is used is when there is a query $(\mathsf{HS}, H_T(\mathsf{ct}, n))$ s.t. (ct, n) was already queried to $\mathcal{O}^{\mathsf{Dec}'}$ (i.e. $\mathcal{L}^j_K = (\mathsf{ct}, n, h)$ for some h). In this case h is returned iff $G(\mathsf{decaps}(\mathsf{sk}, \mathsf{ct})) = \mathsf{HS}$ (let's call this Condition (1)). Recalling that queries to H_j never repeat by assumption, there will be at most one query $H''_j(\mathsf{HS}, H_T(\mathsf{ct}, n))$ s.t. (ct, n) was queried to the decapsulation oracle and Condition (1) is fulfilled. Hence, one can simulate H_j by sampling an index $q_j \in \{0, \ldots, q_{H_j}\}$ and returning h (if it exists) in the q_j-th query or never in case $q_j = 0$. This successfully simulates H_j with prob. $\frac{1}{(q_{H_j}+1)}$. Overall, the probability that both H_1 and H_2 are simulated correctly is $\frac{1}{(q_{H_1}+1)(q_{H_2}+1)}$.

The other ROs can be simulated perfectly by $\mathcal{C}$ using lazy sampling. Collecting the probabilities holds that $\mathcal{C}$ simulates perfectly $\mathcal{A}$'s view in game $\mathit{\Gamma}^6$ (as long as query does not occur) with probability

$$p = \frac{1}{(q_{H_2}+1)^2(q_{H_1}+2)(q_{H_2}+2)(q_{H_1}+1)} \ .$$

Then if query happens, K^* will be in the list of queries made by $\mathcal{A}$ to G. The adversary can guess which one it is and succeeds with probability $\frac{1}{q_G}$. Hence, we have

$$|\Pr[\mathit{\Gamma}^5 \Rightarrow 1] - \Pr[\mathit{\Gamma}^6 \Rightarrow 1]| \leq \Pr[\mathsf{query}] \leq q_G(q_{H_1}+2)^2(q_{H_2}+2)^3 \cdot \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{KEM}}(\mathcal{C}) \ .$$

Finally, in game $\mathit{\Gamma}^6$, as $H_1(\mathsf{HS}^*, \mathsf{ct}^*, n^*)$, $H_2(\mathsf{HS}^*, \mathsf{ct}^*, n^*)$ or $H_3(\mathsf{HS}^*)$ cannot be queried anymore, the challenge keys are perfectly indistinguishable from random for the adversary. Hence,

$$\Pr[\mathit{\Gamma}^6 \Rightarrow 1] = \frac{1}{2} \ .$$

Collecting the probabilities holds the result. □

4.3 Security of TLS 1.3 with IND-1CCA-MAC KEM

We can now use the (slightly modified) notion IND-1CCA-MAC KEM to prove the security of the TLS 1.3 handshake in the multi-stage security model of Günther [17]. We provide a brief reminder of the notion of MultiStage security in the full version of the paper [20]. The security of (the original) TLS 1.3 handshake was proven by Dowling et al. [12] and we refer the reader to their work for a complete analysis of the handshake. We will simply show that IND-1CCA-MAC KEMs, thus OW-CPA KEMs (if the MAC is secure), can be used in place of the original snPRF-ODH assumption for DH key-exchange.

First, we show the relevant part of the (full 1-RTT) handshake of TLS 1.3 in Fig. 14. One can see that the key schedule is nearly identical to the ones used in the IND-1CCA-MAC game. Note that several simplifications have been made and several steps irrelevant to our proofs are missing. In particular, we do not see the derivation of the finals keys, which all depend on the secret dHS. As we will show, the intermediary secrets $(\mathsf{CHTS}, \mathsf{SHTS}, \mathsf{dHS})$ are secure (i.e. indistinguishable from random for a Multi-Stage adversary), thus all subsequent keys will be secure as well, assuming the KDFs are secure. Finally, we write $\mathsf{HKDF.Exp}_i(\mathsf{HS}, T_2)$ for $\mathsf{HKDF.Exp}(\mathsf{HS}, \mathsf{label}_i, T_2)$, where label_i is some string. As we consider the KDFs $\mathsf{HKDF.Ext}$ and $\mathsf{HKDF.Exp}$ to be ROs, this denotes the fact that the label implements oracle separation.

The security of the modified 1-RTT TLS 1.3 handshake is stated in the following theorem.

Theorem 5. *Let* $\mathsf{HKDF.Ext}$*,* $\mathsf{HKDF.TK}$ *and* $\mathsf{HKDF.Exp}_j$*,* $j \in \{0, 4, 5, 6\}$ *(the KDFs in TLS 1.3) be random oracles. Let* Hash *(the hash function used to compute the hashed transcripts* T_i*) be a RO, and* Sig *the signature scheme used for*

TLS 1.3 with KEM Handhsake

Client		**Server**
$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{gen}$		
$\mathsf{CH} : n_c \leftarrow\!\$\ \{0,1\}^{256}$		
$+\ \mathsf{pk}$		
	$\mathsf{dES} \leftarrow \mathsf{HKDF}(\text{constant})$	
	$\xrightarrow{\mathsf{CH}}$	
		$K, \mathsf{ct} \leftarrow\!\$\ \mathsf{encaps}(\mathsf{pk})$
		$\mathsf{SH} : n_s \leftarrow\!\$\ \{0,1\}^{256}$
		$+\ \mathsf{ct}$
	$\xleftarrow{\mathsf{SH}}$	
$K \leftarrow \mathsf{decaps}(\mathsf{sk}, \mathsf{ct})$		
	$\mathsf{HS} \leftarrow \mathsf{HKDF.Ext}(\mathsf{dES}, K)$	
	$\mathsf{CHTS} \leftarrow \mathsf{HKDF.Exp}_4(\mathsf{HS}, T_2)$	
	$\mathsf{SHTS} \leftarrow \mathsf{HKDF.Exp}_5(\mathsf{HS}, T_2)$	
	$\mathsf{dHS} \leftarrow \mathsf{HKDF.Exp}_0(\mathsf{HS}, T_0)$	
....	(Stage 1) accept $\mathsf{tk_c} \leftarrow \mathsf{HKDF.TK}(\mathsf{CHTS})$	....
....	(Stage 2) accept $\mathsf{tk_s} \leftarrow \mathsf{HKDF.TK}(\mathsf{SHTS})$	....
	...	
	$\mathsf{fk}_S \leftarrow \mathsf{HKDF.Exp}_6(\mathsf{SHTS})$	
		$\{\mathsf{SF}\} : \mathsf{MAC}(\mathsf{fk}_S, T_7)$
	$\xleftarrow{\{\mathsf{SF}\}}$	
if $\mathsf{MAC}(\mathsf{fk}_S, T_7) \neq \mathsf{SF}$: **abort**		
	...	

Fig. 14: TLS 1.3 handshake with KEM. $\{\ldots\}$ indicates an encrypted message with $\mathsf{tk_S}$, T_i is the hash of the transcript up to message i. For simplicity, the CH (resp. SH) message captures both the *ClientHello* and *ClientKeyShare* (resp. *ServerHello* and *ServerKeyShare*). Only the relevant steps for the proof are shown. Keys in the remaining stages (3-6, not shown) are all derived from dHS.

server authentication (not shown in Fig. 14). For any Multi-Stage ppt adversary $\mathcal{A}$ there exist ppt adversaries $\{\mathcal{B}_i\}_{i\in[6]}$ s.t.

$$\begin{aligned}\mathsf{Adv}^{\text{multi-stage}}_{\mathsf{TLS1.3-1RTT}}(\mathcal{A}) \leq 6t_s\Big(&\mathsf{Adv}^{\text{coll}}_{H}(\mathcal{B}_1) + t_u\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig}}(\mathcal{B}_2)\\ &+ t_s\Big(\mathsf{Adv}^{\text{ind-1cca-mac}}_{\mathsf{KEM}}(\mathcal{B}_3) + 2\cdot\mathsf{Adv}^{\text{prf}}_{\mathsf{HKDF.Exp}}(\mathcal{B}_4)\\ &+ \mathsf{Adv}^{\text{prf}}_{\mathsf{HKDF.Ext}}(\mathcal{B}_5) + \mathsf{Adv}^{\text{prf}}_{\mathsf{HKDF.Exp}}(\mathcal{B}_6)\Big)\Big)\ .\end{aligned}$$

where t_s (resp. t_u) is the maximal number of sessions (resp. users). Note that for the sake of the comparison with the original bound, we keep several PRF advantages and the collision advantage in the bound, even though they could be replaced by negligible terms, as the KDFs and Hash *are ROs.*

Proof sketch. Due to space constraint, we defer the proof to the full version of the paper [20]. However, as hinted above, the idea of the proof is simply to replace the snPRF-ODH step of the original proof by using our IND-1CCA-MAC. Note that while the snPRF-ODH assumption is used to replace the root secret HS by a random one, we will be able to replace the values ($\mathsf{CHTS}, \mathsf{SHTS}, \mathsf{dHS}$) by random ones in one step, due to the structure of the IND-1CCA-MAC definition. From a high-level point of view, the proof goes through because CHTS and SHTS are computed similarly as in the $\mathsf{T_H}$ transform (i.e. the secrets are the hashed seed and ciphertext) and thus resist to 1 adversarial decapsulation query. Then, dHS is used only once a MAC has been verified, which implies that an adversary relaying a correct tag should already know the root key HS. □

Similarly, one can prove the security of the modified TLS 1.3 PSK-(EC)DHE 0-RTT handshake. Note that in our case the key-exchange will be done with KEMs, but we keep the "-(EC)DHE" in the name for consistency with the original protocol. We state this in the following informal theorem.

Theorem 6. *The modified TLS 1.3 handshake in the pre-shared key (optional) 0-RTT mode with key-exchange (i.e. TLS 1.3 PSK-(EC)-DHE 0-RTT) is secure in the MultiStage model if the underlying KEM is OW-CPA (and signature, MAC, etc. are secure), in the sense of Dowling et al. [12].*

Proof. The only step in the original proof involving the KEMs can be dealt with a similar reduction from IND-1CCA-MAC as in the proof of Theorem 5. □

Corollary 3. *The original TLS 1.3 handshake is MultiStage secure in the ROM if the CDH problem is hard (and the signature, MAC, etc. are secure). Stronger assumptions used in previous proofs (e.g. PRF-ODH [12]) are not necessary.*

Proof. This simply follows from the fact that DH can be described as a KEM $(\mathsf{sk}, \mathsf{pk}) := (x, g^x)$, $(K, \mathsf{ct}) := (\mathsf{pk}^y, g^y)$ and $\mathsf{decaps}(\mathsf{sk}, \mathsf{ct}) := \mathsf{ct}^x$. Integrating this KEM in our modified TLS 1.3 handshake exactly holds the standard TLS 1.3 handshake. Finally, this KEM is OW-CPA as long as the CDH problem is hard, thus by Theorems 4 and 5, the handshake is secure. One can also directly show that DH as used in TLS 1.3 is a IND-1CCA KEM. We provide such a proof in the full version [20]. □

Remarks. Note that due to non-tightness of the bound in Theorem 4, the overall bound for TLS security is very much non-tight. This is clearly not sufficient to guarantee security in practice, and we leave as an interesting open question the improvement of the bounds. In addition, we leave security in the QROM as future work. As we extensively use the programming property of ROs, new QROM techniques such as the *compressed oracles* by Zhandry [33] might be of use in such a proof.

5 Impact

The transforms introduced in Section 3 produce IND-qCCA KEMs without any derandomization and re-encryption steps. Thus, using IND-1CCA ephemeral KEMs obtained through these transforms could speed up the decapsulation process in several protocols.

KEMTLS. As discussed in the introduction, improving the KEMTLS protocol [29] was the main motivation of this work. In particular, a more efficient decapsulation in the ephemeral KEM would decrease overall latency and computation on the client-side. In particular, this could be of interest for less powerful clients like IoT devices, which would not need to perform re-encryption. Overall, the efficiency gain in practice would obviously depend on the ephemeral KEM used, as encryption is expensive in some schemes while it is not in others. For instance, using KEMTLS with a modified version of SIKE (i.e. obtained through our transform instead of the FO one) would reduce probably significantly the handshake latency and computation cost on the client-side.

The same remarks apply to the very recent variants of KEMTLS with predistributed keys proposed by Günther et al. [18] and Schwabe et al. [31].

Note also that following a similar proof as the one in Section 4, we conjecture that one should be able to prove that CPA-security of the ephemeral KEM should suffice for KEMTLS to be secure in the ROM (but at the expense of a non-tight security bound, as in the TLS case).

TLS 1.3. TLS 1.3 only supports ephemeral DH as a key-exchange. In turn, in the original security proof [12], the snPRF-ODH assumption is used for the key-exchange security. The snPRF-ODH assumption can be seen as a variant of the hashed Diffie-Hellman assumption with a 1-time "decapsulation" oracle. More precisely, an adversary is given (g, g^u, g^v) and either $y_0 := \mathsf{PRF}(g^{uv}, \mathsf{ad}^*)$ or a random y_1, where ad^* is some auxiliary data chosen by the adversary. Then, the adversary must distinguish between y_0 and y_1 with the help of *one* query to an oracle $\mathcal{O}((x, \mathsf{ad}) \neq (g^u, \mathsf{ad}^*)) := \mathsf{PRF}(x^v, \mathsf{ad})$.

One can notice that snPRF-ODH security is very close to IND-1CCA security transposed to DH key-exchange. Actually, one can show that IND-1CCA KEM is sufficient for the PQ TLS 1.3 handshake to hold. Indeed, instead of using our IND-1CCA-MAC assumption in the proof, one can use the decapsulation oracle of the IND-1CCA adversary to recover the key if needed. One can check the transition between games $B.1$ and $B.2$ in the proof of KEMTLS security [29] for more details.

Therefore, using IND-1CCA KEMs in the PQ TLS 1.3 handshake seems a sound idea, as in this case the security bound will offer better guarantees than with a OW-CPA KEM. In addition, the handshake would be faster using IND-1CCA KEMs generated by our transforms instead of the slower IND-CCA KEMs derived with FO.

Finally, by Corollary 3, we now know that the snPRF-ODH assumption is not necessary in the ROM for TLS 1.3 to be secure (even though the security bound is very much non-tight), but CDH is sufficient. Alternatively, as shown in the full version [20], DH as used in TLS 1.3 is actually an IND-1CCA KEM ($\approx$ snPRF-ODH) in the ROM if CDH holds. This gives a tighter security bound compared to Corollary 3.

Ratcheting. IND-1CCA security is also a property used (often implicitly) in several works on ratcheting. For instance, Jost et al. [22] build a *healable and key-updating public-key encryption* scheme based on a *one time* IND-CCA2 PKE (with authenticated data). The latter primitive can easily be made out of an IND-1CCA KEM using KEM/DEM techniques. Another paper by Poettering et al. [27] introduces a construction of *unidirectional ratcheted key exchange* (URKE) that is based (implicitly) on IND-1CCA KEMs, as noticed by Balli et al. [2].

In another recent paper, Brendel et al. [6] propose an alternative to the Signal handshake based on KEMs and designated verifier signature schemes. They first define a core protocol that uses two KEMs in the same vein as KEMTLS: one with long-term keys for implicit authentication of one of the parties and another one with ephemeral keys for guaranteeing forward security. Again, the latter one requires only IND-1CCA security for the handshake to be secure. Similarly, in the full Signal-like handshake built upon the core protocol (called SPQR), three KEMs are used and one requires only IND-1CCA security.

Concerns over key-reuse. The main security risk of using an IND-1CCA KEM instead of its IND-CCA counterpart is the vulnerability to key-reuse/misuse attacks. Indeed, if a system/protocol is misimplemented s.t. the IND-1CCA KEM is used with a "static" public key instead of an ephemeral one, an adversary might be able to recover the secret key after several decryption queries. In KEMTLS, this risk is mitigated by the use of an IND-CCA KEM in addition to the ephemeral one (which can be IND-1CCA). In particular, the final shared key is derived from shares of both KEMs. Thus, even if the public-key meant to be ephemeral is reused, the final shared key should remain "secure" (but forward security would be lost).

In other systems (e.g. TLS 1.3), the risk of key recovery after a few reuses could be mitigated by using hybrid cryptography. For instance, a very efficient

IND-CCA KEM could be combined with an IND-1CCA one. That would improve the overall security and resistance against key-reuse attacks at a small cost (see e.g. Giacon et al. [16] or Bindel et al. [3] for KEM combiners). Finally, we stress again that if ephemeral keys were misimplemented as static ones in these systems, the forward security property would be lost.

Conclusion. Ratcheting and several recent protocols (e.g. TLS 1.3) are aiming at forward security, which often implies generating a new pair of public/secret keys for each message exchanged. Informally, in many settings this means that an adversary requesting a decryption will be able to do so only once for a given key pair. Thus, IND-1CCA security of the underlying encryption/encapsulation primitive might be sufficient to guarantee the security of such systems.

Acknowledgments. We thank Daniel Collins for pointing out possible use-cases of IND-1CCA KEMs in ratcheting and the anonymous reviewers for their helpful comments. Loïs Huguenin-Dumittan is supported by a grant (project Nº 192364) of the Swiss National Science Foundation (SNSF).

References

1. OQS OpenSSL, https://github.com/open-quantum-safe/openssl, June 2021
2. Balli, F., Rösler, P., Vaudenay, S.: Determining the core primitive for optimally secure ratcheting. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 621–650. Springer (2020)
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. Brendel, J., Fischlin, M., Günther, F., Janson, C.: PRF-ODH: Relations, instantiations, and impossibility results. Cryptology ePrint Archive, Report 2017/517 (2017), https://ia.cr/2017/517
6. Brendel, J., Fiedler, R., Günther, F., Janson, C., Stebila, D.: Post-quantum asynchronous deniable key exchange and the signal handshake. Cryptology ePrint Archive, Report 2021/769 (2021), https://eprint.iacr.org/2021/769
7. Celi, S., Faz-Hernández, A., Sullivan, N., Tamvada, G., Valenta, L., Wiggers, T., Westerbaan, B., , Wood, C.A.: Implementing and measuring KEMTLS. Cryptology ePrint Archive, Report 2021/1019 (2021), https://ia.cr/2021/1019
8. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded CCA2-secure encryption. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 502–518. Springer (2007)
9. Crockett, E., Paquin, C., Stebila, D.: Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH. IACR Cryptol. ePrint Arch. **2019**, 858 (2019)

10. Davis, H., Günther, F.: Tighter proofs for the SIGMA and TLS 1.3 key exchange protocols. In: International Conference on Applied Cryptography and Network Security. pp. 448–479. Springer (2021)
11. Diemert, D., Jager, T.: On the tight security of TLS 1.3: Theoretically sound cryptographic parameters for real-world deployments. Journal of Cryptology **34**(3), 1–57 (2021)
12. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the TLS 1.3 handshake protocol. Journal of Cryptology (2020)
13. Fluhrer, S.: Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085 (2016), https://eprint.iacr.org/2016/085
14. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Annual International Cryptology Conference. pp. 537–554. Springer (1999)
15. 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
16. Giacon, F., Heuer, F., Poettering, B.: KEM Combiners. Cryptology ePrint Archive, Report 2018/024 (2018), https://eprint.iacr.org/2018/024
17. Günther, F.: Modeling advanced security aspects of key exchange and secure channel protocols (2018)
18. Günther, F., Towa, P.: KEMTLS with delayed forward identity protection in (almost) a single round trip. Cryptology ePrint Archive, Report 2021/725 (2021), https://eprint.iacr.org/2021/725
19. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Theory of Cryptography Conference. pp. 341–371. Springer (2017)
20. Huguenin-Dumittan, L., Vaudenay, S.: A note on IND-qCCA security in the ROM and its applications: CPA security is sufficient for TLS 1.3. Cryptology ePrint Archive, Report 2021/844 (2021), https://ia.cr/2021/844
21. Jao, D., Azarderakhsh, R., Campagna, M., Costello, C., Feo, L.D., Hess, B., Jalali, A., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Renes, J., Soukharev, V., Urbanik, D., Pereira, G., Karabina, K., Hutchinson, A.: SIKE (2020), https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions
22. Jost, D., Maurer, U., Mularczyk, M.: Efficient ratcheting: Almost-optimal guarantees for secure messaging. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 159–188. Springer (2019)
23. 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)
24. 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)
25. Paquin, C., Stebila, D., Tamvada, G.: Benchmarking post-quantum cryptography in TLS. Cryptology ePrint Archive, Report 2019/1447 (2019), https://eprint.iacr.org/2019/1447
26. Pereira, M., Dowsley, R., Hanaoka, G., Nascimento, A.C.: Public key encryption schemes with bounded CCA security and optimal ciphertext length based on the CDH assumption. In: International Conference on Information Security. pp. 299–306. Springer (2010)

27. Poettering, B., Rösler, P.: Towards bidirectional ratcheted key exchange. In: Annual International Cryptology Conference. pp. 3–32. Springer (2018)
28. 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)
29. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 1461–1480 (2020)
30. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. Cryptology ePrint Archive, Report 2020/534 (2020), https://eprint.iacr.org/2020/534
31. Schwabe, P., Stebila, D., Wiggers, T.: More efficient post-quantum KEMTLS with pre-distributed public keys. Cryptology ePrint Archive, Report 2021/779 (2021), https://eprint.iacr.org/2021/779
32. Yamakawa, T., Yamada, S., Matsuda, T., Hanaoka, G., Kunihiro, N.: Reducing public key sizes in bounded CCA-secure KEMs with optimal ciphertext length. In: Information Security, pp. 100–109. Springer (2015)
33. Zhandry, M.: How to record quantum queries, and applications to quantum indifferentiability. In: Annual International Cryptology Conference. pp. 239–268. Springer (2019)