Post-Quantum Multi-Recipient Public Key Encryption

Joël Alwen[1], Dominik Hartmann[2], Eike Kiltz[2], Marta Mularczyk[1], and Peter Schwabe[3]

[1] AWS Wickr
alwenjo@amazon.com, mulmarta@amazon.ch
[2] Ruhr-University Bochum
dominik.hartmann@rub.de, eike.kiltz@rub.de
[3] MPI-SP and Radboud University
peter@cryptojedi.org

Abstract. A *multi-message multi-recipient PKE* (mmPKE) encrypts a batch of messages, in one go, to a corresponding set of independently chosen receiver public keys. The resulting "multi-recipient ciphertext" can be then be reduced (by any 3rd party) to a shorter, receiver specific, "invidual ciphertext". Finally, to recover the i-th message in the batch from their indvidual ciphertext the i-th receiver only needs their own decryption key. A special case of mmPKE is multi-recipient PKE where all receivers are sent the same message. By treating (m)mPKE and their KEM counterparts as a stand-alone primitives we allow for more efficient constructions than trivially composing individual PKE/KEM instances. This is especially valuable in the post-quantum setting, where PKE/KEM ciphertexts and public keys tend to be far larger than their classic counterparts.

In this work we describe a collection of new results around batched KEMs and PKE. We provide both classic and post-quantum proofs for all results. Our results are geared towards practical constructions and applications (for example in the domain of PQ-secure group messaging).

Concretely, our results include a new non-adaptive to adaptive compiler for CPA-secure mKEMs resulting in public keys roughly half the size of the previous state-of-the-art [Hashimoto et.al., CCS'21]. We also prove their FO transform for mKEMs to be secure in the quantum random oracle model. We provide the first mKEM combiner as well as two mmPKE constructions. The first is an arbitrary message-length black-box construction from an mKEM (e.g. one produced by combining a PQ with a classic mKEM). The second is optimized for short messages and achieves hybrid PQ/classic security more directly. When encrypting n short messages (e.g. as in several recent mmPKE applications) at 256-bits of security the mmPKE ciphertext are $144n$ bytes shorter than the generic construction. Finally, we provide an optimized implementation of the (CCA secure) mKEM construction based on the NIST PQC winner Kyber and report benchmarks showing a significant speedup for batched encapsulation and up to 79% savings in ciphertext size compared to a naive solution.

1 Introduction

Public Key Encryption (PKE) and Key Encapsulation Mechanisms (KEM) are some of the most important and widely used cryptographic primitives. The advent of quantum computers has motivated a new generation of constructions with conjectured security against quantum adversaries. However, these come at a cost relative to their classic counterparts, most often in the form of greatly increased public key and ciphertext sizes.

Many Messages, Many Receivers. A useful collection of generalizations of PKE and KEM explicitly support the batching of operations. For example, a *multi-message multi-recipient PKE* (mmPKE) generalizes PKE to allow sending a *vector* of messages $\mathbf{m} = (m_1, \ldots, m_n)$ to a matching vector of recipients $\mathbf{R} := (R_1, \ldots, R_n)$ each with their own independently generated key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$. First introduced in [29], an mmPKE's encryption algorithm takes as input $\mathbf{m}$ and a public-keys vector $\{\mathsf{pk}_i\}$ to output a single "multi-recipient ciphertext" C. To reduce ciphertext size for a receiver R_i, C can later be converted into (presumably much shorter) receiver specific "individual ciphertexts" c_i destined for R_i. This so called *Extract* operation requires and reveals no secrets and so can be performed by any 3rd party; e.g. a server relaying traffic between parties. Finally, R_i can retrieve m_i on their own by feeding their individual ciphertext c_i and private key sk_i to the decryption algorithm. However roughly speaking, m_i should remain private even given the full multi-recipient ciphertext and all other decryption keys.

A special case of mmPKE is *multi-recipient PKE* (mPKE) which requires that all recipients are sent the same message: $\forall i, j\ m_i = m_j$. The KEM analogue of mmPKE/mPKE are *(multi-message) multi-recipient KEM* (mmKEM/mKEM). Rather than transmitting a predetermined vector of messages m, an mKEM instead transmits a random symmetric key k to the recipients.

Efficiency and Applications. In principle, constructing such generalizations of PKE/KEM is not difficult. For example, a trivial construction of an mmKEM is to simply use a separate instance of a standard KEM for each receiver to transmit key k_i to receiver R_i. To extended this to an mKEM choose a fresh key k and use an AEAD to encrypt k under each k_i. Similarly, a trivial construction of an mmPKE is to use a separate instance of PKE for each receiver. However, by treating these extensions of PKE/KEM as primitives in their own right we open the door to more efficient constructions. Already in the classic setting the mmPKEs of [29, 9, 8, 36] reduce computation time and ciphertext size by essentially half over the trivial construction. But it is in the post-quantum setting where such savings take on the greatest importance as we discuss below.

Indeed, switching to extractable batched versions of KEM/PKE has played a central role in reducing (at times, quite dramatically) the communication cost of some recent protocols. Notably, [28] proposed several PQ mKEMs to reduce the complexity of a (PQ version of) MLS - the upcoming Secure Group Messaging standard of the IETF. Compared to a trivial mKEM construction from Kyber their Kyber-based construction reduces ciphertext size by 90% (asymptotically in the number of recipients). Meanwhile, mPKE plays a central role in the recent

Secure Group Messaging protocol of [24]. Their Kyber-based mmPKE reduces ciphertext size by 50%, albeit at the cost of doubling public-key sizes. A generic mmPKE scheme is used in the recent Continuous Group Key Agreement protocol of [3]. However, no PQ mmPKE constructions are known to date (beyond trivial ones described above).

Security. Each of the above protocols is designed to defend against active adversaries. Thus, they all require strong non-malleability properties from their mKEM/mmPKE. While [24] requires full fledged CCA security, [3] gets away with a relaxed variant known as *replayable CCA* (RCCA) security [13]. Intuitively, CCA guarantees non-malleability of the challenge ciphertext, while RCCA "only" guarantees that its *semantics* are non-malleable. More formally, RCCA ensures that upon being given a challenge ciphertext that encrypts a message m the adversary cannot produce a new ciphertext decrypting to a message distinct from yet still meaningfully related to m (although, unlike for CCA, it may be possible to produce fresh encryptions of m).

Along with non-malleability, all the above applications need mKEM/mmPKE schemes to remain secure in the face of adaptive corruptions. In other words, being given a set of candidate receiver public keys, the adversary may request the secret keys of an arbitrary subset of their choosing. Security should then hold for the remaining uncorrupted key pairs. Adaptive security is often difficult to prove (despite the absence of any known attacks). For example, the PQ mKEM of [28] enjoys no such security proof. To date, the only batched primitives with provable adaptive security are the CCA secure mPKEs of [24] and mmPKE of [3]. Yet, both security proofs are in the classic setting only. There exist no batched primitives of any type with security proof for PQ adaptive security.

Another gap in the state-of-the-art concerns hybrid security for batched primitives. A "hybrid" security property holds if any one of multiple underlying assumptions holds. A special case of constructions enjoying hybrid security are combiners [25, 23]. A *combiner for primitive P* constructs an instance of P from two (or more) underlying instances of P. The construction is secure if any of the underlying primitives is secure. Hybrid security is a very useful tool for hedging against future breaks of underlying assumptions. This is particularly pertinent for PQ constructions as it allows us to use powerful but relatively new and untested assumptions while still falling back to lower risk (albeit classical) assumptions such as DDH. To date, no combiners exist for extractable batched primitives (PQ or otherwise). In particular, KEM combiners for CCA security [22, 33, 10, 12] generally require hashing or computing a MAC of the full ciphertext during decapsulation. Thus, it is not immediately clear how those schemes can be extended to an *extractable* batched primitive. Receivers, who only see their individual ciphertext, no longer have the same view; let alone know the entire multi-recipient ciphertext produced by the sender.

1.1 Our Contributions

Nominally, the goal of this work is to build hybrid PQ/classically (R)CCA secure mmPKEs. However, we have taken a very modular approach getting there,

resulting in a collection of constructions and theorems which are useful in the wider context of extractable batched PKE/KEMs, especially (but not exclusively) in the PQ setting. At the most abstract level, we first add adaptive security to existing CPA encryption schemes. Next, we transform the result into a CCA secure KEM using the FO transform followed by a KEM/DEM construction to obtain CCA secure encryption. For hybrid security, we show how to combine KEMs as well as an optimized direct construction of a hybrid CCA encryption scheme. The challenge in all this is to make this paradigm work for the extractable batch generalizations of PKE/KEM and to prove security against quantum (and classic) adversaries.

Adaptive CPA Security for mKEMs. In more detail, we begin with [28] which gives us PQ non-adaptively CPA-secure mPKE. Picking up from there, our first contribution is a new black-box compiler converting a non-adaptively CPA secure mPKE into an adaptively CPA secure one. Compared to the same type of compiler (implicit) in [24], our compiled ciphertext sizes have the same size but our public keys are about half as big as theirs.[4] We remark that [24] don't prove their compiler secure against quantum adversaries, although their classic proof should apply essentially unchanged in the quantum model.

To achieve this, our compiler places additional (mild) requirements on the original mKEM (which are already satisfied by all CPA secure mKEMs that we are aware of). First, for some arbitrary space of public keys $\mathcal{PK}$ the distributions of public keys output during key generation should look uniform over $\mathcal{PK}$. Second $\mathcal{PK}$ should be equipped with an (efficiently computable) group operation.

We prove classic and quantum security for the compiler. For the remainder of this section, unless stated explicitly otherwise, all security notions are implicitly adaptive.

From CPA to (R)CCA Security for mKEMs. Next, we consider adapting the FO transform [19, 20], which converts a CPA-secure PKE into a CCA-secure KEM, to the multi-recipient setting. The first multi-recipient FO transform was proposed in [28] where it is shown to be classically and quantum secure. However, the proofs from [28] do not work with adaptive corruptions and, according to [17], the proof of quantum security has a gap (carried over from the proof technique by Zhandry [42]). Partially fixing these issues, the work [24] proposes a new variant of the multi-recipient FO (the main difference is introducing explicit rejections) with adaptive corruptions but only prove it classically secure. In this work, we complete the above picture by proving quantum security for the FO transform from [24]. Our proof uses the framework of [17], thus avoiding the issue in the proof from [28]. We note that applying the framework of [17] to the multi-recipient setting is technically non-trivial. For example, in order to use it, we define a new notion of spreadness for mPKEs.

[4] We note that all applications of mKEM/mmPKE mentioned above require sending fresh public keys as part of every protocol packet [24]. In fact, [28, 3] both require multiple new keys per packet.

mKEM Combiner. We introduce the first mKEM combiner. It is extremely simple, essentially running the two input mKEM instances in parallel. If the first decapsulates to a key k_1 and the second to k_2 then the key for the combined scheme is simply $k := F(k_1, k_2)$.

We model F as a dual PRF and show two incomparable results for the construction. First, if at least one of the two schemes is RCCA-secure than so is the combined scheme (which suffices for the application in [3] for example).

Second, we define a notion of collision resistance (CR) for an mKEM and show that if one mKEM is CCA-secure and the other is CR than the combined scheme is also CCA-secure. In particular, we also show that an mKEM built using our FO scheme using a hash function H is itself CR in any model in which H is collision resistant. We also present a variant of standard Diffie-Hellman based mKEM which we can also show to be CR (against both classic and quantum adversaries). For this we need the underlying hash function to be collision resistant as well as the encryption function of the underlying DEM. Note that any deterministic DEM can be augmented to have this property by restricting the decryption algorithm to only output the plaintext if re-encrypting it produces the original ciphertext. We prove both classic and quantum versions of these results.

mmPKE. We present two new mmPKE constructions. The first is a black-box mKEM/DEM construction geared towards arbitrary length messages. We show two incomparable results for it. First, if both the mKEM and DEM are CCA-secure then so is the resulting mmPKE. Second, if both the mKEM and DEM are RCCA secure then so is the mmPKE.

Our second mmPKE construction is optimized to provide shorter multi-recipient and individual ciphertexts when encrypting short messages (as is the case in all three protocols [28, 24, 3]). The construction also uses an mKEM and DEM as building blocks. However, it also directly uses a Diffie-Hellman (DH) group to ensure classic security regardless of the mKEM. In particular, to obtain hybrid PQ/classic security for this mmPKE it is more efficient to use a PQ-only mKEM directly rather than a hybrid one. Specifically, we show 4 results for this construction.

1. If the mKEM and DEM are both RCCA secure, then so is the mmPKE.
2. If the mKEM and DEM are both CCA secure then so is the mmPKE.
3. If the DEM is RCCA secure, then the Double Strong Diffie-Hellman(DSDH)[5] assumption implies that the mmPKE is RCCA secure.
4. If the mKEM is collision resistant and the DEM is CCA secure then the (DSDH) assumption implies that the mmPKE is CCA secure.

Implementation. Finally, we present an implementation of a Kyber-based mKEM, which is based on the AVX2 implementation by the Kyber team. Our benchmarks show that, compared to a trivial Kyber-based mmKEM, computation time required for key generation and decapsulation increases, but encapsulation

[5] The DSDH is the standard assumption used to prove CCA security of DH based non-interactive key exchange. [18]

becomes significantly faster already for relatively small batches of, e.g., 10 recipients. More importantly, the ciphertext size is decreased by up to 79%. We place the implementation into the public domain (CC0). It is available from http://131.174.142.4/kyber-mkem.tar.bz2.

1.2 Related Work

The idea of improving efficiency of PKE schemes by encrypting to many receivers at once was first proposed by Kurosawa [29]. In particular, this work proposes the first constructions and security notions for mmPKE. The mmPKE primitive has been later considered in a line of works [9, 8, 36] which define stronger security notions for it (where "stronger" relates to the adversary's ability to corrupt receivers), propose different modularizations and construct various mmPKE schemes. For example, [36] defines mmKEMs to be used as a stepping stone in building mmPKEs. All known mmPKE constructions are only clasically secure.

A parallel sequence of works considers relaxations of mmPKE/mmKEM, namely, mPKE and mKEM. The mKEM primitive was introduced by Smart [37]. Afterwards, various mKEM constructions have been proposed based on hash proof systems [32] and pairings [40]. Finally, the first efficient post-quantum secure mKEMs have been constructed in [28]. In terms of mPKEs, the only construction with strong (stronger than IND-CPA) security was proposed recently by Hashimoto et al. [24]. Another line of works considers identity-based versions of (m)mPKE and (m)mKEMs [14, 35, 26, 30]. Finally, [5] build practical PQ secure Dual-PRFs.

2 Preliminaries

We recall necessary definitions for our results in this section. We start with quantum preliminaries and continue with primitives and their security notions. We present additional preliminaries in Appendix A.

2.1 Post-Quantum Preliminaries

We define the most important building blocks needed for our QROM proofs. For a more thorough introduction on quantum computation, see e.g. [34].

2.1.1 Oneway-to-Hiding (O2H) We recall the original version of the well-known Oneway-to-Hiding lemma [39], using the notation from [4]. Intuitively, the lemma allows us the bound the advantage of an adversary in detecting programming in the quantum random oracle model.

Lemma 1 (Original O2H [4], Theorem 3). *Let $\mathcal{R}$ be a set and $\mathcal{S} \subset \mathcal{R}$ be random. Let G, H be random functions with domain $\mathcal{R}$ satisfying $\forall r \notin \mathcal{S}$: $G(r) = H(r)$. Let z be a random classical value ($\mathcal{S}, G, H, z$ may have arbitrary joint distribution). Let $\mathcal{A}$ be a quantum oracle algorithm making q oracle queries,*

expecting input z. Let Ext *be the algorithm which on input z samples a uniform i from $\{1,\dots,q\}$, runs $\mathcal{A}$ right before its i-th query to G, measures its query input register and outputs the measurement m. Then*

$$\left|\Pr[\mathcal{A}^G(z) \Rightarrow 1] - \Pr[\mathcal{A}^H(z) \Rightarrow 1]\right| \leq 2q\sqrt{\Pr[m \in \mathcal{S} : m \leftarrow_{\$} \mathsf{Ext}^H(z)]}.$$

2.1.2 Extractable Quantum Random Oracle Simulation In general, it isn't possible to extract the values an adversary queried in a quantum random oracle without noticeable disrupting the adversaries quantum state. This problem was solved in the seminal work of Zhandry [42] using so-called *compressed oracles*. [17] simplified the formalism of compressed oracles and defined the clean abstraction of a quantum random oracle simulator which allows for extraction queries (under specific conditions) without noticeably changing the adversaries state. We recall their construction in Definition 2 together with some properties in Definition 1

Definition 1. *Let $n \in \mathbb{N}$, $\mathcal{X}, \mathcal{T}$ two set, $f : \mathcal{X} \times \{0,1\}^n \to \mathcal{T}$ a function and $R \subset \mathcal{X} \times \{0,1\}^n$ a relation. We define*

$$\Gamma(f) := \max_{\substack{x \in \mathcal{X} \\ t \in \mathcal{T}}} |\{y | f(x,y) = t\}|, \quad \Gamma'(f) := \max_{\substack{x,x' \in \mathcal{X} \\ y \in \{0,1\}^n}} |\{y | f(x,y) = f(x',y')\}|$$

and $\Gamma_R := \max_{x \in \mathcal{X}} |\{y | (x,y) \in R\}|$.

Definition 2 (Extractable Quantum Random Oracle Simulator). *Let $n \in \mathbb{N}$, $\mathcal{X}, \mathcal{T}$ two sets, $f : \mathcal{X} \times \{0,1\}^n \to \mathcal{T}$ a function and $R' \subset \mathcal{X} \times \mathcal{T}$ and $R \subset \mathcal{X} \times \{0,1\}^n$ relations with $(x,y) \in R \Leftrightarrow (x, f(x,y)) \in R'$.*

We define the stateful quantum simulator $\mathcal{S}_f$ *that has the (quantum accessible) interfaces $\mathcal{S}_f.RO : \mathcal{X} \to \{0,1\}^n$ and $\mathcal{S}_f.E : \mathcal{T} \to \mathcal{X} \cup \perp$ and the following properties:*

1. *If no query to $\mathcal{S}_f.E$ is made, $\mathcal{S}_f.RO$ is indistinguishable from a (quantum) random oracle.*
2. *Any two independent queries to $\mathcal{S}_f.RO$ (resp. $\mathcal{S}_f.E$) commute.*
3. *Any two subsequent queries to $\mathcal{S}_f.E$ and $\mathcal{S}_f.RO$ $8\sqrt{\frac{\Gamma(f)}{2^{n-1}}}$-almost-commute.*
4. *Any query to $\mathcal{S}_f.RO$ (resp. $\mathcal{S}_f.E$) is idempotent, i.e. returns the same result if no other query was made in between.*
5. *If $\hat{x} = \mathcal{S}_f.E(t)$ and $\hat{h} = \mathcal{S}_f.RO(\hat{x})$ are two subsequent classical queries, then $\Pr\left[\hat{x} \neq \perp \wedge f(\hat{x}, \hat{h}) \neq t\right] \leq \frac{2\Gamma(f)}{2^n}$.*
6. *If $h = \mathcal{S}_f.RO(x)$ and $\hat{x} = \mathcal{S}_f.E(f(x,h))$ are two subsequent classical queries, then $\Pr[\hat{x} = \perp] \leq \frac{1}{2^{n-1}}$.*
7. *Let $\mathcal{A}$ be an adversary making at most q queries to the $\mathcal{S}_f.RO$ oracle and no queries to the $\mathcal{S}_f.E$ oracle, which outputs $t \in \mathcal{T}$. Then $\Pr[(\hat{x},t) \in R' \mid t \leftarrow_{\$} \mathcal{A}^{\mathcal{S}_f.RO}; \hat{x} \leftarrow_{\$} \mathcal{S}_f.E(t)] \leq 128 \cdot q^2 \cdot \Gamma_R / 2^n$.*
8. *Let $\mathcal{A}$ be an adversary making at most q queries to $\mathcal{S}_f.RO$ and no queries to $\mathcal{S}_f.E$, that outputs $x,t \in \mathcal{X} \times \mathcal{T}$. Then $Pr[\hat{x} \neq x \wedge f(x,h) = t \mid t,x \leftarrow_{\$} \mathcal{A}^{\mathcal{S}_f.RO}; h \leftarrow_{\$} \mathcal{S}_f.RO(x); \hat{x} \leftarrow_{\$} \mathcal{S}_f.E(t)] \leq 40e^2 \cdot (q+2)^3 \Gamma'(f)/2^n$.*

Let us give some intuition on the properties of $\mathcal{S}$. Properties 1 and 2 ensure, that $\mathcal{S}$ behaves like a regular quantum random oracle, unless $\mathcal{S}.E$ is called and that independent query don't interfere with one another. Property 3 tells us that extraction only causes detectable change in the state of $\mathcal{S}$ with low probability (as long as f is sparse). Property 4 ensures that queries are consistent as long as the state of the oracle does not change between queries. Property 5 states that if extraction succeeds, then it returns a correct preimage with high probability and 6 state that extraction almost always works if an image was indeed generated via the oracle. Property 7 gives us a bound for finding a specific relation on input/output pairs of the simulator (i.e. quantum search is hard in the simulated random oracle). Lastly, Property 8 tells us that finding collisions in $\mathcal{S}$ is hard despite the extraction interface. Specifically, even with the extraction interface, the probability of finding a collision is still bounded by a cubic factor.

2.2 Multi-Message Multi-Recipient Encryption

A multi-message multi-recipient public-key encryption (mmPKE) scheme allows a sender to encrypt a *vector* of messages to a *vector* of public keys. Formally, an mmPKE scheme consists of the following algorithms. (Correctness can be found in Appendix A.)

Parameter Generation: The parameter generation algorithm $\mathsf{mmSetup}$ generates randomly generated public parameters pp.

Key Generation: The key generation algorithm $\mathsf{mmKGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$ takes the public parameters pp as input and generates a key pair $(\mathsf{pk}, \mathsf{sk})$.

Encryption: The encryption algorithm $\mathsf{mmEnc}(\mathsf{pp}, \vec{\mathsf{pk}}, \vec{m}) \to C$ takes the public parameters pp, a *vector* of public keys $\vec{\mathsf{pk}}$ and a *vector* of messages $\vec{m}$ as input and produces a multi-recipient multi-message ciphertext C. The i-th message in $\vec{m}$ is encrypted to the i-th public key in $\vec{\mathsf{pk}}$.

Extraction: The (deterministic) extraction algorithm $\mathsf{mmExt}(\mathsf{pp}, C, i) \to c/\perp$ takes the public parameters pp, a ciphertext C and an index i and outputs the individual ciphertext for the i-th receiver (or $\perp$).

Decryption: The decryption algorithm $\mathsf{mmDec}(\mathsf{pp}, \mathsf{sk}, c) \to m/\perp$ takes the public parameters pp, a secret key sk and an individual ciphertext c as input and returns either a decrypted message m or $\perp$ if decryption fails.

Security. We recall in Fig. 1 the $\mathsf{mmIND\text{-}CCA}$ and $\mathsf{mmIND\text{-}RCCA}$ security with adaptive corruptions from [3]. The notions build upon the analogous notions for regular encryption. For each notion, the security experiment starts with the challenger generating a number of mmPKE key pairs and sending the public keys to the adversary. At some point, the adversary can request a challenge: it sends to the challenger two challenge *vectors* of messages and a challenge *vector* of public keys, all vectors of the same length. The public keys in the challenge vector can be either chosen from those generated by the challenger or chosen by the adversary.

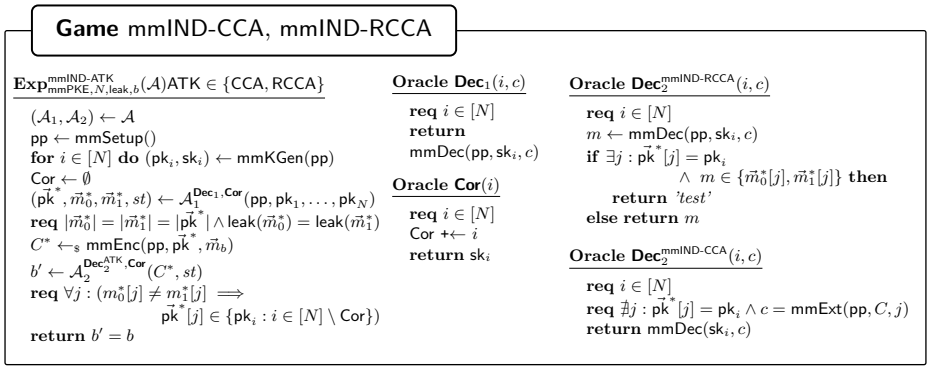

Fig. 1: The mmIND-RCCA and mmIND-CCA security game for mmPKE with an arbitrary leakage function $\mathsf{leak}(\vec{m})$.

In addition, throughout the experiment, the adversary can adaptively corrupt the key pairs generated by the challenger and access a decryption oracle. The oracle is defined in a way that does not allow the adversary to trivially decrypt the challenge. Note that this mechanism is slightly different for CCA and RCCA; this is the only difference between the notions.

To disallow trivial wins, we require that the challenge message vectors coincide on slots where the keys in the challenge public key vector are corrupt or chosen by the adversary.

Further, we parameterize the notion by a function $\mathsf{leak}(\vec{m})$ and require that the output of leak on the two challenge message vectors is the same. The function leak formalizes all metadata about encrypted vectors that need not be kept secret. For instance, these can be the lengths of individual messages or structure of the vector (e.g. whether two consecutive messages are the same).

Definition 3. *Let* mmPKE *be an mmPKE scheme, let N be an integer and let* $\mathrm{Exp}^{\mathsf{mmIND\text{-}ATK}}_{\mathsf{mmPKE},N,\mathsf{leak},b}(\mathcal{A})$ *be defined in Fig. 1. Further, let* leak *be a function with a domain containing all message vectors. For* $\mathsf{ATK} \in \{\mathsf{CCA},\mathsf{RCCA}\}$ *we denote the advantage of adversary $\mathcal{A}$ playing game* mmIND-ATK *with leakage* leak *by* $\mathrm{Adv}^{\mathsf{mmIND\text{-}ATK}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A})$ *and define it to be:*

$$\left|\Pr[\mathrm{Exp}^{\mathsf{mmIND\text{-}ATK}}_{\mathsf{mmPKE},N,\mathsf{leak},1}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{Exp}^{\mathsf{mmIND\text{-}ATK}}_{\mathsf{mmPKE},N,\mathsf{leak},0}(\mathcal{A}) \Rightarrow 1]\right|.$$

2.3 Multi-Recipient Key Encapsulation

A multi-recipient key-encapsulation mechanism (mKEM) allows to encapsulate a single key for multiple recipients. It consists of the following algorithms:

Parameter Generation: The algorithm $\mathsf{mSetup} \to \mathsf{pp}$ returns freshly sampled public parameters pp.

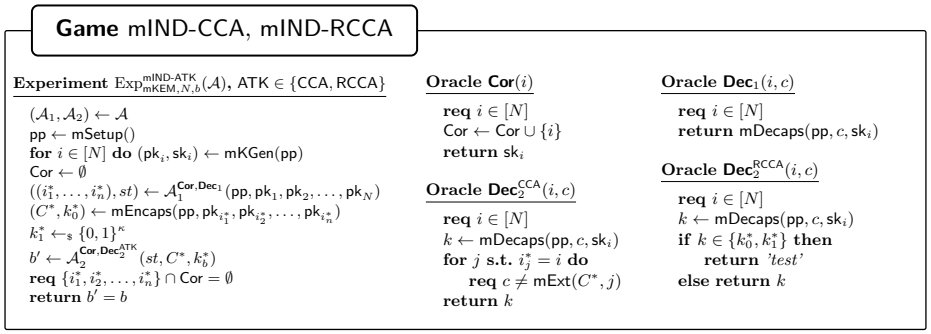

Fig. 2: mIND-CCA and mIND-RCCA security experiments for mKEM.

Key Generation: The key generation algorithm $\mathsf{mKGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$ takes as input a public parameter pp returns a fresh key pair $(\mathsf{pk}, \mathsf{sk})$.

Encapsulation: The (multi-recipient) encapsulation algorithm $\mathsf{mEncaps}(\mathsf{pp}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n) \to (C, k)$ takes in a sequence (of any length $n > 0$) of public keys and outputs a (multi-recipient) ciphertext C and encapsulated key k.

Extract: The deterministic extraction algorithm $\mathsf{mExt}(\mathsf{pp}, C, i) \to c/\perp$ takes as input a multi-recipient ciphertext C and position index i and returns a (individual) ciphertext c for the i-th recipient.

Decapsulation: The decapsulation algorithm $\mathsf{mDecaps}(\mathsf{pp}, c, \mathsf{sk}) \to k/\perp$ takes as input an individual ciphertext c and decapsulation secret key sk. If decapsulation succeeds it returns the encapsulated key k. (Otherwise, it returns an arbitrary bitstring or $\perp$.)

Correctness can be found in Appendix A.

Security. We define the strong security notion for mKEM with corruptions as (implicitly) defined in [24], called mIND-CCA and the slightly weaker notion of mIND-RCCA in Definition 4.

Definition 4. *Let* mKEM *be an mKEM scheme, let* N *be an integer and let* $\mathrm{Exp}^{\mathsf{mmIND\text{-}ATK}}_{\mathsf{mKEM},N,b}(\mathcal{A})$ *be defined in Fig. 2. For* $\mathsf{ATK} \in \{\mathsf{CCA}, \mathsf{RCCA}\}$*, we define the advantage of an adversary* $\mathcal{A}$ *against the* mIND-ATK *security of* mKEM *as*

$$\mathrm{Adv}^{\mathsf{mIND\text{-}ATK}}_{\mathsf{mKEM},N}(\mathcal{A}) = \left|\Pr[\mathrm{Exp}^{\mathsf{mIND\text{-}ATK}}_{\mathsf{mKEM},N,1}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{Exp}^{\mathsf{mIND\text{-}ATK}}_{\mathsf{mKEM},N,0}(\mathcal{A}) \Rightarrow 1]\right|.$$

2.4 Multi-Recipient Encryption

A multi-recipient public-key encryption (mPKE) scheme allows a sender to encrypt a *single* message to a *vector* of public keys. We will mainly use this primitive as a stepping stone towards constructing strongly secure mKEM (via a multi-recipient variant of the Fujisaki-Okamoto transform). Therefore, even though mPKE is a special case of mmPKE, we give syntax and security definitions

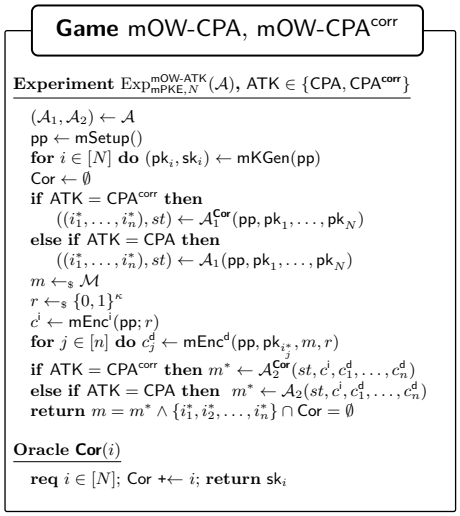

Fig. 3: One-way security of mPKE.

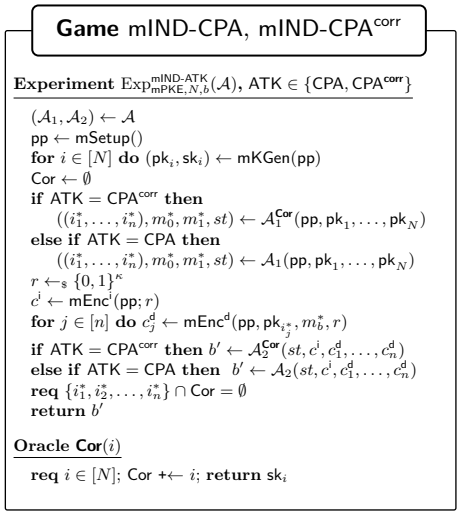

Fig. 4: IND security of mPKE.

for mPKE that are geared towards enabling the mKEM construction. In particular, we use the less general syntax from [28] and only consider security with passive attackers. Additionally, we need the notion of γ-*keyindependent spreadness*, which is a variant of classical γ-spreadness for public key encryption and will be used similarly in (a multi-recipient variant of) the Fujisaki-Okamoto transformation to turn a weakly secure mPKE into a strongly secure mKEM. An mPKE scheme consists of the following algorithms:

Parameter Generation: $\mathsf{mSetup}() \to \mathsf{pp}$ returns a fresh public parameter pp.
Key Generation: Key generation algorithm $\mathsf{mKGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$ takes as input a public parameter pp and samples and returns a fresh key pair.
Encryption: Encryption consists of two algorithms: first, $\mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp}) \to c^{\mathsf{i}}$ outputs a *recipient-independent* ciphertext component c^{i} needed by all recipients. Second, $\mathsf{mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}, m, r) \to c^{\mathsf{d}}$ takes as input a public key pk, a message m and the randomness r used by $\mathsf{mEnc}^{\mathsf{i}}$ and outputs a *recipient-dependent* ciphertext component c^{d} needed only by the recipient with pk.
Decryption: $\mathsf{mDec}(\mathsf{pp}, \mathsf{sk}, (c^{\mathsf{i}}, c^{\mathsf{d}})) \to m/\perp$ takes as input a secret key sk and a ciphertext consisting of a recipient-independent component c^{i} and a recipient-dependent component c^{d}. It outputs a message m or $\perp$ if decryption fails..

Correctness can be found in Appendix A.

Security. We define four security notions for mPKE, both one-way and indistinguishability with and without adaptive corruptions as in [24]. The notions are formally defined in Figures 3 and 4. Roughly, for $\mathsf{mOW\text{-}CPA}$ security, it should be hard for an adversary to find the encrypted random challenge message without

the secret key. For mIND-CPA, the adversary can choose two messages and has to decide, which of the two was encrypted. In the variants with adaptive corruptions, the adversary can additionally corrupt honest keys, but can only be challenged on honest keys to prevent trivial wins. mOW-CPA (resp., mOW-CPA$^{\mathsf{corr}}$) is trivially implied by mIND-CPA (resp., mIND-CPA$^{\mathsf{corr}}$).

Definition 5. *Let* mPKE *be an mPKE scheme and let N be an integer. For* $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CPA}^{corr}\}$*, we define the advantage of an adversary $\mathcal{A}$ against the* mOW-ATK *security of* mPKE *as*

$$\mathrm{Adv}^{\mathsf{mOW\text{-}ATK}}_{\mathsf{mPKE},N}(\mathcal{A}) = \Pr[\mathrm{Exp}^{\mathsf{mOW\text{-}ATK}}_{\mathsf{mPKE},N}(\mathcal{A}) \Rightarrow 1],$$

where $\mathrm{Exp}^{\mathsf{mOW\text{-}ATK}}_{\mathsf{mPKE},N}(\mathcal{A})$ *is defined in Fig. 3. Further, we define the advantage of $\mathcal{A}$ against the* mIND-ATK *security of* mPKE *as*

$$\mathrm{Adv}^{\mathsf{mIND\text{-}ATK}}_{\mathsf{mPKE},N}(\mathcal{A}) = \Pr[\mathrm{Exp}^{\mathsf{mIND\text{-}ATK}}_{\mathsf{mPKE},N,1}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{Exp}^{\mathsf{mIND\text{-}ATK}}_{\mathsf{mPKE},N,0}(\mathcal{A}) \Rightarrow 1],$$

where $\mathrm{Exp}^{\mathsf{mIND\text{-}ATK}}_{\mathsf{mPKE},N,b}(\mathcal{A})$ *is defined in Fig. 4.*

Remark 1. Looking ahead, the Fujisaki-Okamoto transform (see Section 4) only require oneway security. On the other hand, our compiler from security without corruptions to security with corruptions (see Section 3.1) only works for mIND-CPA security and is not applicable for mOW-CPA. Thus, so we define both notions to show the stronger result.

3 Adaptively CPA-Secure mPKE

The goal of this section is to obtain (single-message) multi-recipient PKE (mPKE) schemes that satisfy the mOW-CPA$^{\mathsf{corr}}$ security notion, i.e., they are one-way secure in the presence of passive attackers and corruptions. These schemes are meant to be turned into IND-CCA secure (single-key) multi-recipient KEMs (mKEMs) using the FO transform described in Section 4.

We observe that most (m)mPKE and (m)mKEM constructions proposed so far, including [29, 9, 8, 36, 28], have been analyzed in the setting without (adaptive) corruptions. This is rather surprising given that they are meant to be used in applications like secure messaging where such a setting is considered completely unrealistic. Even more surprisingly, there seems to be no way to adapt the proofs for the setting without corruptions to the setting with corruptions.[6]

For this reason, we construct in this section a black box compiler that takes as input an mPKE satisfying the standard security notion considered in the literature, mIND-CPA, and outputs an mPKE that satisfies mIND-CPA$^{\mathsf{corr}}$ security which is the same as mIND-CPA except the adversary can adaptively corrupt

[6] This is quite unintuitive – why would the adversary gain any meaningful power from the ability to corrupt key pairs that are independent of those used in the challenge? And indeed, we don't find any specific attacks. However, the known proof techniques fail for reasons related to the so called commitment problem.

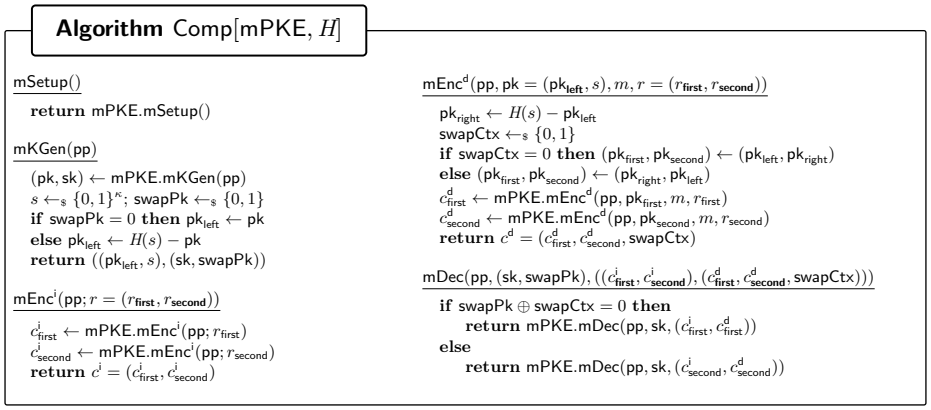

Fig. 5: The mIND-CPA$^{\text{corr}}$ secure mPKE scheme outputted by our compiler.

honest keys. We note that mIND-CPA$^{\text{corr}}$ implies mOW-CPA$^{\text{corr}}$, therefore, the output of the compiler achieves the goal of this section.

Finally, we conclude this section with a review of known mPKE constructions which can be used as input to our compiler.

3.1 The Compiler

The basic construction. The compiler is defined in Fig. 5. At a high level, the basic idea is to adapt the technique from [21] for obtaining adaptively secure broadcast encryption to the mPKE setting.

This means that the compiler essentially runs two parallel instances of the mIND-CPA secure scheme mPKE. Each recipient has two mPKE key pairs, called "left" and "right". He only knows one of the secret keys and he keeps it secret which one. To encrypt a message m to recipients 1 to n, the encryptor runs the mPKE encryption twice. Both times, the encrypted message is m. Further, for each recipient i, its left public key goes to one invocation of mPKE and its right public key to the other. For each i, the encryptor chooses at random whether the left key goes to the first or to the second invocation. The resulting ciphertext consists of the two mPKE ciphertexts as well as, for each i, the bit indicating the invocation which used the i-th left key.

The description in Fig. 5 formalizes the above intuition in the mPKE syntax with the encryption split into mEnc$^{\text{i}}$ and mEnc$^{\text{d}}$. In particular, values computed for the first invocation of the mPKE encryption are marked by the subscript first, while for the second invocation with the subscript second. The bit swapCtx sampled by mEnc$^{\text{d}}$ for the i-th recipient decides if the left mPKE public key of that recipient goes to the second invocation, i.e., the public keys are swapped, or to the first.

At this point, the above scheme may seem quite unintuitive. The reason is that we are not modifying mPKE in order to prevent any real attacks, but rather

to enable a security proof. This means that to get an intuition for the scheme, one has to first get an intuition for the proof. We note that this proof technique is not our contribution but the result of [21]; we invite all readers interested in why this works to look at the proof of our compiler's security in Appendix B.1.

We note that the recent work by Hashimoto et al. [24] also adapts the technique of [21] to the mPKE setting. However, they do not have the optimization we introduce next.

Key compression. To optimize the compiler, we introduce a technique called key compression. We observe that in the basic construction a recipient only needs one secret key, say for $\mathsf{pk}_{\mathsf{left}}$. This means that he can generate the right public key $\mathsf{pk}_{\mathsf{right}}$ without necessarily knowing the secret key, for example as the output of a hash function on a random seed s. Then instead of the full public key $(\mathsf{pk}_{\mathsf{left}}, \mathsf{pk}_{\mathsf{right}})$, he publishes $(\mathsf{pk}_{\mathsf{left}}, s)$. Since s can be much shorter than a public key, this would cut the public key size of the compiled scheme roughly in half.

Of course, the recipient cannot simply publish $(\mathsf{pk}_{\mathsf{left}}, s)$, because he should hide whether he knows the left or the right secret key. To fix this, we first assume that there is some group operation $+$ on the public key space of pk. Then, we make $H(s)$ the sum of $\mathsf{pk}_{\mathsf{left}}$ and $\mathsf{pk}_{\mathsf{right}}$. That is, given a public key $(\mathsf{pk}_{\mathsf{left}}, s)$, we can compute $\mathsf{pk}_{\mathsf{right}} = H(s) - \mathsf{pk}_{\mathsf{left}}$. Now if the recipient knows the right secret key only, he can publish the key $(H(s) - \mathsf{pk}_{\mathsf{right}}, s)$.

Remark 2. Key compression requires that one can define some group operation on the public key space of mPKE. This is typically very straightforward. For instance, the public key spaces of mPKE constructions based on Diffie-Hellman [36] and LWE [28] are by definition groups. Alternatively, if an mKEM scheme has a public key space with dense representation in bitstrings, then the group operation can be bit-wise XOR.

The technique also requires a hash function outputting elements of the public key space. For an overview of such hash functions in the (elliptic-curve) DH context see, e.g., [38]. For LWE-based constructions, the typical approach is to use rejection sampling on the output of a XOF; see, e.g., [2, Sec. 3&7]. However, if for some instantiation of mPKE such a hash function does not exist, the fallback is to use the compiler without key compression shown in [24].

Security. Security of the compiler, and in particular of the key compression, requires an additional property from mPKE. Roughly, the public key produced by key generation should look like a uniform random element of the public-key space. Intuitively, this is necessary to argue that the output of the hash looks like an honest public key, so by looking at a recipient's public key, one cannot tell if the recipient knows the left or the right secret key. This fact is necessary to use the proof technique from [21]. Formally, we define

Definition 6. *For a scheme* mPKE *with public-key space* $\mathcal{PK}$*, parameter-generation algorithm* mSetup *and key-generation algorithm* mKGen*, we define the advantage of an adversary* $\mathcal{A}$ *against random-key security of* mPKE*,* $\mathrm{Adv}^{\mathsf{mRND\text{-}PK}}_{\mathsf{mPKE}}(\mathcal{A})$*, as*

$$\Pr\left[\mathcal{A}(\mathsf{pp},\mathsf{pk}) \Rightarrow 1 \;\middle|\; \begin{array}{c}\mathsf{pp} \leftarrow_{\$} \mathsf{mSetup}() \\ (\mathsf{pk}, *) \leftarrow_{\$} \mathsf{mKGen}(\mathsf{pp})\end{array}\right] - \Pr\left[\mathcal{A}(\mathsf{pp},\mathsf{pk}) \Rightarrow 1 \;\middle|\; \begin{array}{c}\mathsf{pp} \leftarrow_{\$} \mathsf{mSetup}() \\ \mathsf{pk} \leftarrow_{\$} \mathcal{PK}\end{array}\right]$$

The following theorem formalizes security and correctness of the compiler. We prove it in Appendix B.1.

Theorem 1. *Let* $\mathsf{mPKE} = \mathsf{Comp}[\mathsf{mPKE}, H]$*, where* mPKE *is an mPKE with a group operation over the space of public keys and* H *is a hash function, be defined as in Fig. 5. For any number of recipients* N *and for any (classical or quantum) adversary* $\mathcal{A}$*, there exist (classical or quantum) adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *such that*

$$\mathrm{Adv}^{\mathsf{mIND\text{-}CPA}^{\mathsf{corr}}}_{\mathsf{mPKE},N}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{mIND\text{-}CPA}}_{\mathsf{mPKE},N}(\mathcal{B}_1) + 2N \cdot \mathrm{Adv}^{\mathsf{mRND\text{-}PK}}_{\mathsf{mPKE}}(\mathcal{B}_2),$$

where H *is modeled as a random oracle. Additionally, if* mPKE *is* δ*-correct, then* $\mathsf{Comp}[\mathsf{mPKE}, H]$ *is* δ*-correct as well.*

3.2 Examples of mIND-CPA Secure Schemes

mIND-CPA secure mPKE can be constructed from various classical and post-quantum assumptions. For lattice-based constructions, schemes based on the Lindner-Peiker framework [31] can be extended to the mPKE setting [28]. That work also provides an overview over various NIST competition candidates that fit this framewor. Additionally, [15] propose a CCA-secure mKEM from the LPN assumption, which implicitly also contains an mPKE scheme. For isogenies, there are variants from SIDH as well as CSIDH, which are both reminiscent of the well-known hashed ElGamal construction [27]. For a comparison in terms of ciphertext size and computation, we again refer the reader to [28]. Standard model constructions exist from prime-order groups[6, 9, 29, 36], which are variations on the ElGamal or Cramer-Shoup encryption schemes.

Note that the isogeny-based PKE called CSIDH-ECIES-KEM proposed in [41] is conceptually almost identical to the mmPKE scheme proven secure in [36][7]. However, there exists no *direct* proof of post-quantum *CCA* security for CSIDH-ECIES-KEM, so we opt to view it as a CPA-secure PKE and apply the same transformations as to the other schemes. Such a direct CCA security proof is an interesting open problem.

4 The FO Transform

The Fujisaki-Okamoto (FO) transform for regular encryption [19, 20] takes as input an OW-CPA secure PKE scheme and outputs an IND-CCA secure KEM. It works in the random oracle model (ROM). The work [28] adapts the FO to the multi-recipient setting, i.e., their transform takes as input an mOW-CPA mPKE and outputs an mIND-CCA secure mKEM. They prove the transform's security in both the ROM and the QROM. However, they do not consider adaptive corruptions; their proof fails in this setting, because their construction uses implicit rejections. Moreover, [17] claim that there is a non-trivial gap in the

[7] CSIDH-ECIES-KEM is defined as a regular KEM, but the same randomness reuse as in [36] yields an mmPKE.

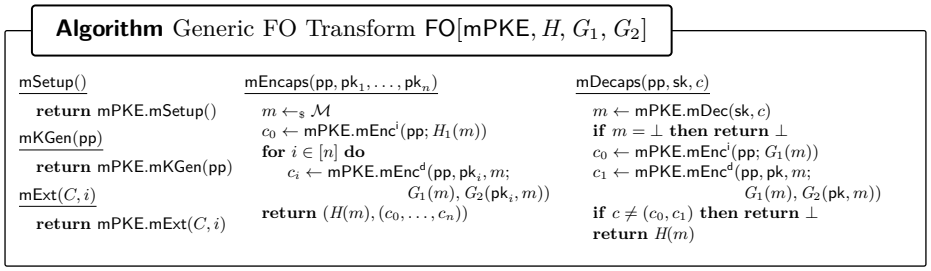

Fig. 6: The Fujisaki-Okamoto transform for mPKE with adaptive corruption.

proof. The work [24] partially fixes this problem by adding explicit rejections and proving security with adaptive corruptions in the ROM.

In this section, we complete the above picture by proving security of the multi-recipient FO transform in the QROM with adaptive corruptions. To fix the issue pointed out in [17], we use the online-extractable simulation technique [17].

New notion of spreadness. In order to apply the framework of [17], we introduce a new notion of spreadness. It is meant for the decomposable syntax of mPKEs from [28], i.e. one where ciphertexts consist of a public key independent part that all receiver need an a personalized, public key dependent part that is only needed by a single receiver. Regularly, γ-spreadness bounds the probability of the whole ciphertext taking a specific value. However due to this decomposition, we require only that it is unlikely for the public key *independent* part to take any specific value. We formalize this as *γ-keyindependent spreadness* in Definition 7

Definition 7 (γ-keyindependent spreadness). *Let $n \in \mathbb{N}$ and* mPKE *be an mPKE scheme. Let* mPKE *is* γ-keyindependent spread*, if*

$$\mathop{\mathbb{E}}_{\substack{\mathsf{pp}\in\mathsf{mSetup}\\ \mathsf{pk}\in\mathsf{mKGen}(\mathsf{pp})}} \left[\max_{\substack{m\in\mathcal{M}\\ ct\in\mathcal{C}^I}} \Pr_{r_0\in\mathcal{R}} \left[ct = \mathsf{mEnc}(\mathsf{pp}; r_0) \right] \right] \leq 2^{-\gamma},$$

where $\mathcal{C}^I$ denotes the set of all recipient-independent ciphertext parts.

Security. We recall the construction of the multi-recipient FO transform in Fig. 6. The following theorem formalizes its security with adaptive corruptions in the ROM and in the QROM.

Theorem 2. *Let* mPKE *be a δ-correct and γ-keyindependent spread mPKE with message space $\mathcal{M}$ and G_1, G_2 and H (classical or quantum) random oracles. Then for any (classical or quantum) adversary $\mathcal{A}$, there exists a (classical or quantum)*

adversary $\mathcal{B}'$ with approximately the same runtime as $\mathcal{A}$, s.t.

$$\mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{FO}[\mathsf{mPKE},G_1,G_2,H],N}(\mathcal{A}) \leq 4q \cdot \sqrt{\mathrm{Adv}^{\mathsf{mOW\text{-}CPA}^{\mathsf{corr}}}_{\mathsf{mPKE},N}(\mathcal{B}')} + 48q^2\sqrt{n\delta} + 23q\sqrt{q} \cdot 2^{-\frac{\gamma}{4}} + 28e\sqrt{q^3} \cdot 2^{-\frac{\gamma}{2}} + 28eq\frac{\sqrt{q^3}}{2^\kappa} + \frac{8q\sqrt{q}}{2^\kappa}$$

with κ as the output length of the random oracles and $q = 2q_D + q_H + q_D$, where q_D is the number of classical queries to the decapsulation oracle, and q_G and q_H is the number of (classical or quantum) queries to the random oracles G_1 and G_2 and to the random oracle H respectively made by $\mathcal{A}$.

Proof. The proof for the classical case can be found in [24]. For the quantum case, we adapt the security proof of the standard FO transform from [17] to the mPKE FO transform. The main difference is that [17] considers PKE and KEM, while we look at mPKE and mKEM.

The proof idea still remains similar: We switch the randomness used to encrypt the challenge message and the key to uniformly random and then argue that an adversary noticing this switch already breaks the $\mathsf{mOW\text{-}CPA}^{\mathsf{corr}}$ security of the underlying mPKE scheme. For the latter, we use the extractable quantum random oracle simulator to extract the queries an adversary made to the oracle and use them similarly to the classical setting to simulate the decapsulation oracle. Here, the main observation is that the recipient-independent part of a ciphertext is already a commitment to the encrypted message and we can extract the message.

For public parameters pp and keypair $(\mathsf{pk}, \mathsf{sk})$, let g_{pp} be the maximum probability of any user-independent ciphertext c_0 occurring and δ_{sk} the maximum probability of a decryption error for a given keypair. Then $\mathbb{E}[g_{\mathsf{pp}}] \leq 2^{-\gamma}$ and $\mathbb{E}[\delta_{\mathsf{sk}}] \leq \delta$ due to the definition of δ-correctness and γ-keyindependent spreadness, with the expectation take over the randomness of the parameter and key generation. Formally, we define the games G_0 to G_{11} in Fig. 7 and describe their relation in the following paragraphs.

Game G_0: This game is identical to the $\mathsf{mIND\text{-}CCA}$ game with $b = 0$, i.e. $\mathrm{Exp}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{FO}[\mathsf{mPKE},G_1,G_2,H],N,0}(\mathcal{A}) = \Pr[\mathsf{G}_0{}^{\mathcal{A}} = 1]$ Additionally, we interpret G_1 and G_2 as two interfaces of a single random oracle G with appropriate domain separation.

Game G_1: Now, we puncture the random oracles G and H on the challenge message m^*. Formally, we define two new *punctured* random oracles $\widehat{G}$ and $\widehat{H}$ as follows. For each $m \neq m^*$ and $i \in [N]$: $H(m) = \widehat{H}(m)$, $G(m) = \widehat{G}(m)$ and $G(\mathsf{pk}_i, m) = \widehat{G}(\mathsf{pk}_i, m)$. For each other input, the outputs of $\widehat{G}$ and $\widehat{H}$ are random and independent. The adversary still gets access to the unpunctured oracles G and H. However, we modify the decapsulation oracle as follows: Let $r^* = G_1(m^*)$, $r^*_i = G_2(\mathsf{pk}_i, m^*)$ for $i \in [N]$ and $c^* = (\mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp}; r^*), (\mathsf{mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}_{i^*_j}, m^*; r^*, r^*_{i^*_j}))_{j\in[n]})$. Analogously, let $\hat{r} = \widehat{G}_1(m^*)$, $\hat{r}_i = \widehat{G}_2(\mathsf{pk}_i, m^*)$ for $i \in [N]$ and $\hat{c} = (\mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp}; \hat{r}), (\mathsf{mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}_{i^*_j}, m^*; \hat{r}, \hat{r}_{i^*_j}))_{j\in[n]})$. The decapsulation oracle uses the punctured oracles on all queries except for parts of $\hat{c}$.

We argue that the view of the adversary doesn't change between the two games. Indeed, note that for all messages except for the challenge message m^*, nothing

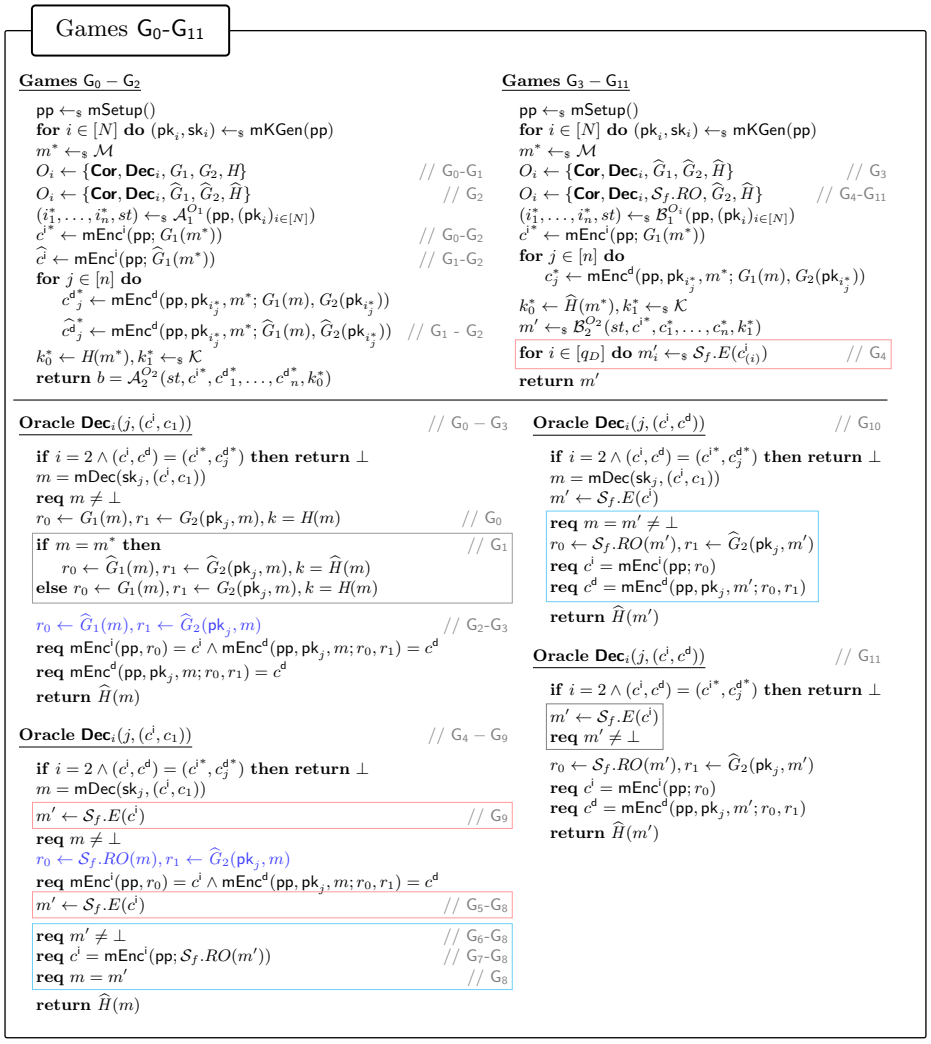

Fig. 7: Games G_0 to G_{11} for the proof of Theorem 2.

changes as the oracles coincide. For (all parts of) c^*, the decapsulation oracle won't answer by definition. For $\hat{c}$, the oracle will also output $\perp$ since the oracle uses the unpunctured oracle for $\hat{c}$, which in turn uses the punctured randomness, so the reencryption check will fail. Therefore, we have $\Pr\left[{\mathsf{G}_0}^{\mathcal{A}} = 1\right] = \Pr\left[{\mathsf{G}_1}^{\mathcal{A}} = 1\right]$.

Game G_2: Finally, in G_2 we switch the random oracles of the adversary to $\widehat{G}$ and $\widehat{H}$. Note that now c^* is an encryption of a random message with randomness

independent of the random oracles and k_0^* is a uniformly random key. Therefore, we can also switch the key to k_1^*.

To bound the difference between the two games, we use the original oneway-to-hiding(O2H) lemma (Lemma 1), which yields $\left|\Pr\left[\mathsf{G}_1{}^{\mathcal{A}}=1\right]-\Pr\left[\mathsf{G}_2{}^{\mathcal{A}}=1\right]\right| \leq 2(q_H+q_G+2q_D)\sqrt{\Pr\left[\mathsf{G}_3{}^{\mathcal{B}}=m^*\right]}$, where G_3 is identical to G_2 except that $\mathcal{B}$ simulates $\mathcal{A}$ and before a random query to one of the random oracles $\widehat{G}$, $\widehat{H}$ or the oracle queries made by a decryption query, $\mathcal{B}$ measures said query and outputs the result.[8] If $\mathcal{B}$ intends to measure a query but $\mathcal{A}$ makes less queries or aborts beforehand, $\mathcal{B}$ outputs $\perp$. Note that this differs from the approach of [17] in that we let $\mathcal{B}$ measure an arbitrary random oracle query including those in the decapsulation oracle. We don't need the addition distinction as done in their games since we always perform a reencryption check and therefore always perform the random oracle queries which $\mathcal{B}$ might need for extraction.

To get the final advantage of the adversary in $\mathrm{Exp}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{FO[mPKE},G_1,G_2,H],N,1}(\mathcal{A})$, we have to revert the changes made before, so overall, we get $\left|\mathrm{Exp}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{FO[mPKE},G_1,G_2,H],N,0}(\mathcal{A})-\mathrm{Exp}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{FO[mPKE},G_1,G_2,H],N,1}(\mathcal{A})\right| \leq 4(q_H+q_G+2q_D)\sqrt{\Pr\left[\mathsf{G}_3{}^{\mathcal{B}}=m^*\right]}$

Game G_4: In this game, we split $\widehat{G}$ again into $\widehat{G}_1$ and $\widehat{G}_2$ and replace $\widehat{G}_1$ by the extractable simulator $\mathcal{S}_f.RO$ for the function $f(m,r_0) := \mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp};r_0)$ as defined in Definition 2. Additionally, after $\mathcal{B}$ outputs its guess m', we query the extract interface $\mathcal{S}_f.E$ on all recipient-independent parts of encapsulations $c^{\mathsf{i}}{}_{(j)}$ that were queried to the decapsulation oracle. Since $\mathcal{S}$ simulates a random oracle perfectly when no extraction queries are made and all extraction queries are made *after* the execution of $\mathcal{B}$ is finished, this change is undetectable, so $\Pr\left[\mathsf{G}_4{}^{\mathcal{B}}=m^*\right]=\Pr\left[\mathsf{G}_3{}^{\mathcal{B}}=m^*\right]$.

Game G_5: Now, we move the extraction from the end of the execution to the decapsulation oracle. Whenever an extraction call and a regular oracle call are swapped, we apply property 3 to bound the distinguishing advantage of $\mathcal{B}$. Since $\mathcal{A}$ makes at most q_G queries to $\mathcal{S}_f.RO$ and there is at most one query in each of the q_D decapsulation queries, we get $\left|\Pr\left[\mathsf{G}_5{}^{\mathcal{B}}=m^*\right]-\Pr\left[\mathsf{G}_4{}^{\mathcal{B}}=m^*\right]\right| \leq 8(q_D+q_G)\sqrt{2\frac{\Gamma(f)}{2^\kappa}} = 8(q_D+q_G)\sqrt{2g_{\mathsf{pp}}}$.

Game G_6: In the next game, we abort if the extraction interface outputs $\perp$ in a decapsulation query. However, we perform this check *after* the queries to $\mathcal{S}_f.RO$ used for reencryption and the reencryption check. Since both the random oracle and extraction query are classical and subsequent, we can apply property 6 q_D times and get $\left|\Pr\left[\mathsf{G}_6{}^{\mathcal{B}}=m^*\right]-\Pr\left[\mathsf{G}_5{}^{\mathcal{B}}=m^*\right]\right| \leq 2q_D\cdot\frac{1}{2^n}$.

[8] Formally, we reprogram not only on m^* but also on all pairs (pk_i,m^*), but for simplicity, we choose the above notation as we can extract m^* from all these points.

Game G_7: In this game, we now require that the extracted message yields the same recipient-independent ciphertext as the decrypted message. Since we extract and immediately reencrypt and all queries in the decapsulation are classical, we can use property 5 q_D times, which yields $\left|\Pr\left[{\mathsf{G}_7}^{\mathcal{B}} = m^*\right] - \Pr\left[{\mathsf{G}_6}^{\mathcal{B}} = m^*\right]\right| \leq 2q_D \cdot \frac{\Gamma(f)}{2^\kappa} = 2q_D \cdot g_{\mathsf{pp}}$.

Game G_8: Next, we assert that the extracted message m' and the decrypted message m are identical *after* the second reencryption check using m'. There are two cases where this assertion can fail: 1) $m \neq m'$, but $\mathcal{S}_f.RO(m) = \mathcal{S}_f.RO(m')$, i.e. there is a collision in the (simulated) random oracle. We can bound this case using the collision finding probability proven in [16]. 2) $m \neq m'$ and $\mathcal{S}_f.RO(m) \neq \mathcal{S}_f.RO(m')$, but still $\mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp}; \mathcal{S}_f.RO(m)) = \mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp}; \mathcal{S}_f.RO(m'))$. We can bound this case using property 8, where we can bound $\Gamma'(f) \leq g_{\mathsf{pp}}$. Together with the collision bound for $q = (q_G + q_D + 1)$, we get $\left|\Pr\left[{\mathsf{G}_8}^{\mathcal{B}} = m^*\right] - \Pr\left[{\mathsf{G}_7}^{\mathcal{B}} = m^*\right]\right| \leq 40e^2(q_D + q_G + 1)^3 g_{\mathsf{pp}} + 42e^2 \frac{(q_D+q_G+1)^3}{2^\kappa}$, since the adversary makes at most q_G queries to $\mathcal{S}_f.RO$ and q_D decapsulation queries in addition to the one query to G_1 after $\mathcal{B}$ is finished.

Game G_9: We move the extraction queries in the decapsulation oracle to *before* the random oracle queries to compute the reencryption randomness. Due to the almost commutativity of the extractor (property 3), we can bound the difference between the two games by $\left|\Pr\left[{\mathsf{G}_9}^{\mathcal{B}} = m^*\right] - \Pr\left[{\mathsf{G}_8}^{\mathcal{B}} = m^*\right]\right| \leq 8q_D \cdot \sqrt{2g_{\mathsf{pp}}}$.

Game G_{10}: We re-order the 4 consecutive **req**'s. Re-ordering **req**'s can never change the observable behavior of the oracle. In detail: we anyway check that $m = m'$ and that they are not bot, so we can do this at the beginning as well. Now we know that $m = m'$ before the reencryption check, so we can replace all subsequent occurrences of m by m'. $\Pr\left[{\mathsf{G}_{10}}^{\mathcal{B}} = m^*\right] = \Pr\left[{\mathsf{G}_9}^{\mathcal{B}} = m^*\right]$.

Game G_{11}: Finally, in game G_{11}, we don't need the decrypted message m anymore, so we drop the decryption query (as well as the equality check $m = m'$). This introduces an error, if the adversary produces a ciphertext $c^{\mathsf{i}}, c^{\mathsf{d}}$, which is an honest decryption of m, but decrypts to another message $\hat{m}$. The reason is that our extractor produces m and the reencryption succeeds. That is, an adversary can distinguish the two games by finding a message causing a decryption error.

We define the relation $R := \{(m, (r_0 = \mathcal{S}_f.RO(m), r_1 = \widehat{G}_2(m, \mathsf{pk}_i)) \mid \exists i \in [n] : c^{\mathsf{i}} = \mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp}; r_0) \wedge c^{\mathsf{d}} = \mathsf{mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}_i, m, r_0, r_1) \wedge m \neq \mathsf{mDec}(\mathsf{sk}_i, (c^{\mathsf{i}}, c^{\mathsf{d}})\}$, i.e. all messages and their corresponding randomness for a given receiver which induce a decryption error when used for signing. Since both $\mathcal{S}_f.RO$ and $\widehat{G}_2$ are random oracles from the perspective of an adversary (with high probability), we can use the definition of δ_{sk_i} and the union-bound to see that $\frac{\Gamma_R}{2^\kappa} \leq n \cdot \max_{i \in [n]}(\delta_{\mathsf{sk}_i})$. Applying property 7 for R then gives us $\left|\Pr\left[{\mathsf{G}_{11}}^{\mathcal{B}} = m^*\right] - \Pr\left[{\mathsf{G}_{10}}^{\mathcal{B}} = m^*\right]\right| \leq 128(q_D + q_G)^2 n \cdot \max_{i \in [n]}(\delta_{\mathsf{sk}_i})$.

Finally, the secret keys aren't needed to compute the decapsulation queries in G_{11} and the randomness used in c^* is independent of $\widehat{G}_1/\mathcal{S}_f.RO$ and $\widehat{G}_2$. Therefore, we can use the $\mathsf{mOW\text{-}CPA}^{\mathsf{corr}}$ security of mPKE to bound $\mathcal{B}$'s advantage in G_{11}. Concretely, an adversary $\mathcal{B}'$ against the $\mathsf{mIND\text{-}CPA}^{\mathsf{corr}}$ security of mPKE chooses a random messages m^*, samples a random key K^* and forwards all keys, its challenge c^* and K^* to $\mathcal{B}$ according to the subset of keys chosen by $\mathcal{B}$. Corruption queries are forwarded to its own oracle. Finally, if $\mathcal{B}$ outputs m', the attacker $\mathcal{B}'$ forwards it. $\mathcal{B}'$ wins the $\mathsf{mOW\text{-}CPA}^{\mathsf{corr}}$ game if and only if $\mathcal{B}$ finds the message m^* for which the oracle was punctured, therefore $\Pr[\mathsf{G}_{11} \Rightarrow m^*] \leq \mathrm{Adv}^{\mathsf{mOW\text{-}CPA}^{\mathsf{corr}}}_{\mathsf{mPKE},N}(\mathcal{B}')$. Combining all probabilities and setting $\epsilon_1 = 8(q_D+q_G)\sqrt{2g_{\mathsf{pp}}}$, $\epsilon_2 = \frac{2q_D}{2^\kappa}$, $\epsilon_3 = 2q_D g_{\mathsf{pp}}$, $\epsilon_4 = 42e^2(q_D+q_G+1)^3\sqrt{g_{\mathsf{pp}}} + 42e^2\frac{(q_D+q_G+1)^3}{2^\kappa}$, $\epsilon_5 = 8q_D \cdot \sqrt{2g_{\mathsf{pp}}}$ and $\epsilon_6 = 128(q_D+q_G)n \cdot \max_{i\in[n]}(\delta_{\mathsf{sk}_i})$, we get $\mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{FO}[\mathsf{mPKE},G_1,G_2,H],N}(\mathcal{A}) \leq 4(q_H+q_G+2q_D)\sqrt{\mathrm{Adv}^{\mathsf{mOW\text{-}CPA}^{\mathsf{corr}}}_{\mathsf{mPKE},N}(\mathcal{B}') + \epsilon_1+\epsilon_2+\epsilon_3+\epsilon_4+\epsilon_5+\epsilon_6}$. With $q = (q_H+q_G+2q_D)$, using that the square root function is convex and taking the expected values, we get the bound in the theorem.

5 An mKEM Combiner

A KEM combiner is a construction that takes as input two KEMs and outputs a KEM that is secure as long as at least one of the input KEMs is secure. The goal of combining KEMs is to derive trust from multiple assumptions rather than relying on a single one.

KEM combiners are of particular importance in the post-quantum setting. Here, one typically does not want to rely solely on one of the relatively new assumptions believed to hold in the post-quantum world. Therefore, such an assumption is combined with a well studied classical assumption such as DL.

In this section, we consider mKEM combiners which are analogous to KEM combiners but in the multi-recipient setting.

Challenges in the multi-recipient setting. Simple and efficient KEM combiners are known in the single-recipient setting. For example, a combiner for IND-CCA secure KEMs [22] simply runs the two KEMs in parallel. Say the first KEM encapsulates a key k_1 in a ciphertext c_1, and the second KEM encapsulates k_2 in c_2. The key encapsulated by the combined scheme is $H(k_1, k_2, c_1, c_2)$, where H is a hash function modeled as a random oracle.

Including both ciphertexts in the hash is crucial for IND-CCA security of the combined scheme. This is demonstrated by a simple attack: Say we compute the key as $H(k_1, k_2)$ and the decapsulation of the insecure second KEM outputs $k_2 = 0$ on any ciphertext c_2. In this case, no matter how secure is the first KEM, an IND-CCA adversary can decapsulate the challenge ciphertext (c_1, c_2) by sending (c_1, c_2') for $c_2' \neq c_2$ to the decapsulation oracle.

This indicates that combiners for IND-CCA secure KEMs should mix the ciphertexts into the encapsulated key in some way. Indeed, this is the case for all

Algorithm Comb[mKEM1, mKEM2, PRF]

mSetup()
  return (mKEM1.mSetup(), mKEM2.mSetup())

mKGen((pp1, pp2))
  $(\mathsf{pk1}, \mathsf{sk1}) \leftarrow$ mKEM1.mKGen(pp1)
  $(\mathsf{pk2}, \mathsf{sk2}) \leftarrow$ mKEM2.mKGen(pp2)
  return $(\mathsf{pk} = (\mathsf{pk1}, \mathsf{pk2}), \mathsf{sk} = (\mathsf{sk1}, \mathsf{sk2}))$

mExt((pp1, pp2), $(C1, C2), i$)
  $c1 \leftarrow$ mKEM1.mExt(pp1, $C1, i$)
  $c2 \leftarrow$ mKEM2.mExt(pp2, $C2, i$)
  return $c = (c1, c2)$

mEncaps((pp1, pp2), $(\mathsf{pk1}_1, \mathsf{pk2}_1), \ldots, (\mathsf{pk1}_n, \mathsf{pk2}_n)$)
  $(C1, k1) \leftarrow$ mKEM1.mEncaps(pp1, $\mathsf{pk1}_1, \ldots, \mathsf{pk1}_n$)
  $(C2, k2) \leftarrow$ mKEM2.mEncaps(pp2, $\mathsf{pk2}_1, \ldots, \mathsf{pk2}_n$)
  return $(C = (C1, C2), k = \mathsf{PRF}(k1, k2))$

mDecaps((pp1, pp2), $(c1, c2)$, (sk1, sk2))
  $k1 \leftarrow$ mKEM1.mDecaps(pp1, $c1$, sk1)
  $k2 \leftarrow$ mKEM2.mDecaps(pp2, $c2$, sk2)
  if $k1 = \perp \vee k2 = \perp$ **then**
    return $\perp$
  else
    return $k = \mathsf{PRF}(k1, k2)$

Fig. 8: mKEM combiner that executes mKEM1 and mKEM2 in parallel and computes the key as $\mathsf{PRF}(k1, k2)$, where $k1$ and $k2$ come from mKEM1 and mKEM2, respectively.

constructions we are aware of. However, this is a showstopper for mKEMs, where each recipient should derive the same key using a different individual ciphertext.

The construction. Since the standard combiners are incompatible with mKEMs, we consider instead the simplest combiner possible which runs the two mKEMs in parallel and computes the encapsulated key as $\mathsf{PRF}(k_1, k_2)$. Here, PRF is a dual PRF [7], i.e. both $\mathsf{PRF}(k, x)$ and $\mathsf{PRF}'(k, x) = \mathsf{PRF}(x, k)$ have the PRF security property. For example, a random oracle is a dual PRF. Simple and post-quantum dual PRFs are also known in the standard model. We provide the pseudocode of the combiner in Fig. 8.

Since our combiner does not include ciphertexts in the encapsulated key, it is not true that its IND-CCA security is implied by IND-CCA security of at least one input mKEM. The reason is that it is vulnerable to the same attack as the one we described in the previous paragraph for single-recipient KEMs. Therefore, in this section we propose two different security statements for our combiner.

5.1 The First Statement: More Direct Guarantees

The first idea is to replace IND-CCA with the slightly weaker notion of replayable CCA, IND-RCCA. Roughly, the difference is that IND-RCCA does not consider it an attack if it is possible to, given a ciphertext, come up with a different ciphertext as long as it encapsulates the same key.

Intuitively, while IND-CCA requires that a scheme protects *the ciphertext string*, IND-RCCA requires that it protects *the ciphertext content*. Since in most use-cases all one cares about is the latter, IND-RCCA is often a more suited notion. Indeed, IND-RCCA is sufficient for secure communication [13] and continuous group key agreement, a component of group messaging [3].

We notice that the attack described at the beginning of this section breaks IND-CCA but not IND-RCCA. Indeed, we can prove that the combined scheme

is IND-RCCA secure if at least one of the input mKEMs is IND-RCCA secure. The proof can be found in Appendix B.2.

Theorem 3. *Let* $\mathsf{mKEM} = \mathsf{Comb}[\mathsf{mKEM1}, \mathsf{mKEM2}, \mathsf{PRF}]$*, where* $\mathsf{mKEM1}$ *and* $\mathsf{mKEM2}$ *are some mKEM schemes and* PRF *is a PRF, be defined as in Fig. 8. Let* $\mathrm{dual}(\mathsf{PRF})$ *denote the PRF obtained by swapping the input and key of* PRF*, i.e.,* $\mathrm{dual}(\mathsf{PRF})(k, x) = \mathsf{PRF}(x, k)$*. For any* $N \in \mathbb{N}$ *and for any (classical or quantum) adversary* $\mathcal{A}$*, there exist (classical or quantum) adversaries* $\mathcal{B}_1$*,* $\mathcal{B}_2$*,* $\mathcal{B}'_1$*,* $\mathcal{B}'_2$ *s.t.*

$$\mathrm{Adv}^{\mathsf{mIND\text{-}RCCA}}_{\mathsf{mKEM},N}(\mathcal{A}) \leq 2 \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}RCCA}}_{\mathsf{mKEM1},N}(\mathcal{B}_1) + \mathrm{Adv}^{\mathsf{PRF}}_{\mathsf{PRF}}(\mathcal{B}_2) \text{ and}$$
$$\mathrm{Adv}^{\mathsf{mIND\text{-}RCCA}}_{\mathsf{mKEM},N}(\mathcal{A}) \leq 2 \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}RCCA}}_{\mathsf{mKEM2},N}(\mathcal{B}'_1) + \mathrm{Adv}^{\mathsf{PRF}}_{\mathrm{dual}(\mathsf{PRF})}(\mathcal{B}'_2).$$

Additionally, if $\mathsf{mKEM1}$ *is* δ_1*-correct and* $\mathsf{mKEM2}$ *is* δ_2*-correct, then* mKEM *is* $(\delta_1 + \delta_2)$*-correct.*

5.2 The Second Statement: Stronger Assumptions

The second idea is to prove that the combined scheme is IND-CCA secure as long as one of the input mKEMs is IND-CCA secure and the other insecure mKEM satisfies a very weak property called collision resistance.

Roughly, collision resistance requires that, even given a secret key, it is hard to come up with two ciphertexts that decapsulate to the same (non $\perp$) key. It turns out that most known mKEMs already are collision resistant (against classical/quantum adversaries), assuming only collision resistance of their underlying components, such as hash functions (against classical/quantum adversaries). We give examples in the next subsection. We define collision resistance as follows.

Definition 8. *The advantage of an adversary* $\mathcal{A}$ *against* collision resistance *of* $\mathsf{mKEM} = (\mathsf{mSetup}, \mathsf{mKGen}, \mathsf{mEncaps}, \mathsf{mExt}, \mathsf{mDecaps})$ *is defined as*

$$\mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM}}(\mathcal{A}) = \Pr\left[c \neq c' \wedge \mathsf{mDecaps}(\mathsf{pp}, \mathsf{sk}, c) = \mathsf{mDecaps}(\mathsf{pp}, \mathsf{sk}, c') \neq \perp \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{mSetup}(), \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{mKGen}(\mathsf{pp}), \\ (c, c') \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}) \end{array} \right].$$

We can now prove the following theorem about our combiner. The proof can be found in Appendix B.2.

Theorem 4. *Let* $\mathsf{mKEM} = \mathsf{Comb}[\mathsf{mKEM1}, \mathsf{mKEM2}, \mathsf{PRF}]$*, where* $\mathsf{mKEM1}$ *and* $\mathsf{mKEM2}$ *are some mKEM schemes and* PRF *is a PRF, be defined as in Fig. 8. Let* $\mathrm{dual}(\mathsf{PRF})$ *denote the PRF obtained by swapping the input and key of* PRF*, i.e.,* $\mathrm{dual}(\mathsf{PRF})(k, x) = \mathsf{PRF}(x, k)$*. For any* $N \in \mathbb{N}$ *and for any (classical or quantum) adversary* $\mathcal{A}$*, there exist (classical or quantum) adversaries* $\mathcal{B}_1$ *to* $\mathcal{B}_3$ *and* $\mathcal{B}'_1$ *to* $\mathcal{B}'_3$ *such that*

$$\mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM},N}(\mathcal{A}) \leq 2 \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM1},N}(\mathcal{B}_1) + \mathrm{Adv}^{\mathsf{PRF}}_{\mathsf{PRF}}(\mathcal{B}_2) + 2N \cdot \mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM2}}(\mathcal{B}_3) \text{ and}$$
$$\mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM},N}(\mathcal{A}) \leq 2 \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM2},N}(\mathcal{B}'_1) + \mathrm{Adv}^{\mathsf{PRF}}_{\mathrm{dual}(\mathsf{PRF})}(\mathcal{B}'_2) + 2N \cdot \mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM1}}(\mathcal{B}'_3).$$

Additionally, if $\mathsf{mKEM1}$ *is* δ_1*-correct and* $\mathsf{mKEM2}$ *is* δ_2*-correct, then* mKEM *is* $(\delta_1 + \delta_2)$*-correct.*

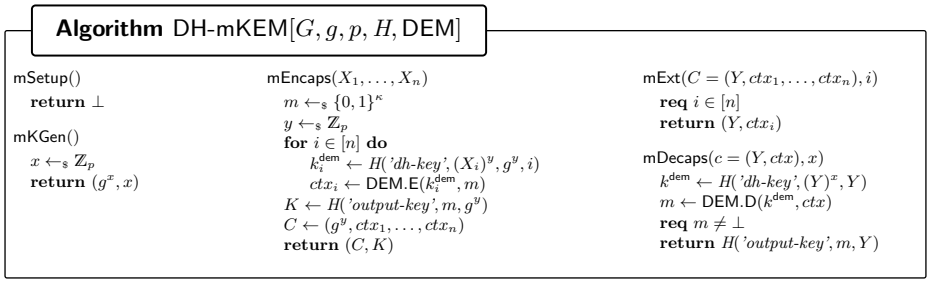

Fig. 9: The collision-resistant mKEM construction based on DH. Observe that to get the encapsulated key, we hash in the randomness $Y = g^y$.

5.3 Collision-Resistant mKEMs

From the FO transform. We show that any mKEM obtained using the FO transform is collision resistant assuming only collision resistance of the hash function used by the FO to derive the output key. This means that all post-quantum secure mKEMs from [28] are collision-resistant. The formal claim follows. The proof can be found in Appendix B.3.

Theorem 5. *Let* $\mathsf{mKEM} = \mathsf{FO}[\mathsf{mPKE}, H, G_1, G_2]$*, where* mPKE *is any mPKE scheme and* H*,* G_1 *and* G_2 *are any hash functions. For any (classical or quantum) adversary* $\mathcal{A}$*, there exists a (classical or quantum) adversary* $\mathcal{B}$ *such that*

$$\mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{CR}}_{H}(\mathcal{B}).$$

From Diffie-Hellman. Further, we consider the mKEM obtained by encrypting the same random key to all recipients using the Diffie-Hellman based mPKE [29, 36]. Roughly, the mPKE encrypts a message to all recipients using the ElGamal encryption and a DEM (a construction also known as the DHIES). The efficiency gain comes from the fact that the same randomness for the ElGamal ciphertext can be used for all recipients. To get mKEM, we encrypt a random message m and compute the key as a hash of (m, Y) (with an appropriate label for domain separation), where Y is the ElGamal randomness. Including Y is crucial for collision resistance.[9] The pseudocode of the construction is in Fig. 9.

Collision resistance of the DH-based mKEM relies on the collision resistance of H and the DEM scheme. The latter roughly means that the DEM's decryption function is collision resistance. Formally, we define it as follows.

Definition 9. *The advantage of an adversary* $\mathcal{A}$ *against* collision resistance *of* $\mathsf{DEM} = (\mathsf{E}, \mathsf{D})$ *is defined as*

$$\mathrm{Adv}^{\mathsf{CR}}_{\mathsf{DEM}}(\mathcal{A}) = \Pr\left[\begin{array}{c} c \neq c' \ \wedge \\ \mathsf{D}(k,c) = \mathsf{D}(k,c') \neq \perp \end{array} \middle| \ (k, c, c') \leftarrow \mathcal{A}() \right].$$

[9] Without this, the adversary can easily create a collision by encrypting the same message m with two different randomness values y and y'.

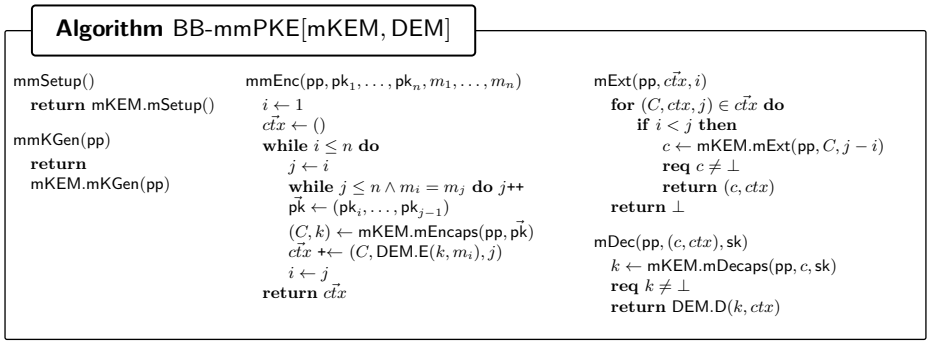

Fig. 10: The construction of mmPKE from mKEM and DEM.

A collision resistant DEM can be constructed from any deterministic DEM (note that security of our mKEM construction requires only one-time security of the DEM, so it can be deterministic). In particular, let DEM be deterministic. To get a collision-resistant DEM, we add an extra check at the end of DEM's decryption algorithm: we re-encrypt the message and output $\perp$ if the re-encryption does not match the original ciphertext.

In Appendix B.3 we show that the DH-based mKEM is collision resistance both in the classical and quantum settings.

6 Two mmPKE Constructions

Finally, we describe how to combine an mKEM and a DEM to built secure mmPKE. Our results hold for both classical and quantum adversaries. Additionally, we present a construction optimized for very short messages (i.e. messages as long as blocks of the DEM), which can be useful in applications such as Secure Group Messaging (SGM). Both constructions leak individual message length as well as whether two consecutive messages in the message vector are identical. Note that this leakage is sufficient for applications such as group messaging [3]. We compare the two constructions for short messages in Appendix B.7

Remark 3. For simplicity, the constructions in this section collect all consecutive occurrences of a message in the encrypted vector, and not all occurrences overall. Therefore they are more efficient, if the input vectors are sorted.

6.1 A Generic Construction of mmPKE from mKEM

The proof of the following theorem can be found in Appendix B.4.

Theorem 6. *Let* mKEM *and* DEM *be an mKEM and a DEM schemes, and let* mmPKE = BB-mmPKE[mKEM, DEM] *be defined as in Fig. 10. Further, let*

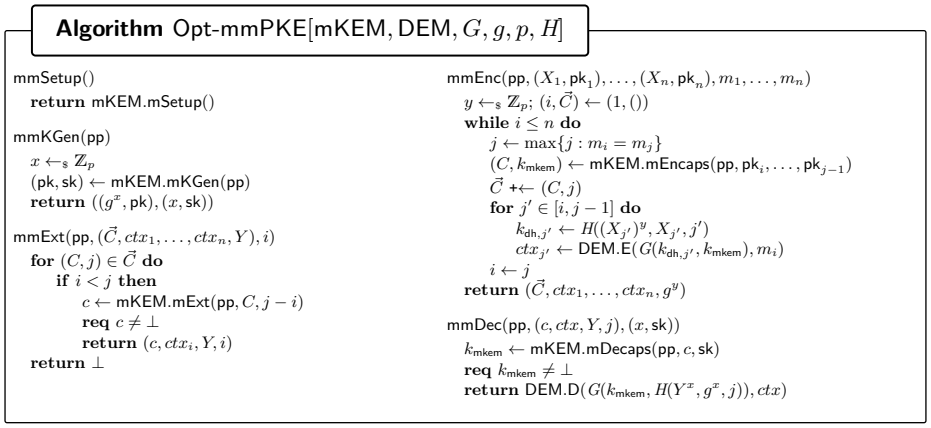

Fig. 11: The optimized construction of mmPKE. H and G are hash functions modeled as random oracles.

$\mathsf{leak}(\vec{m}) := (\{i : \vec{m}[i] = \vec{m}[i+1]\}, |\vec{m}[1]|, \ldots, |\vec{m}[n]|)$. *For any integer N and for any (classical or quantum) adversary $\mathcal{A}$, there exist (classical or quantum) adversaries $\mathcal{B}_1$ to $\mathcal{B}_4$ such that*

$$\mathrm{Adv}^{\mathsf{mmIND\text{-}CCA}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A}) \leq 2n \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM},N}(\mathcal{B}_1) + n \cdot \mathrm{Adv}^{\mathsf{OT\text{-}IND\text{-}CCA}}_{\mathsf{DEM}}(\mathcal{B}_2) \text{ and}$$
$$\mathrm{Adv}^{\mathsf{mmIND\text{-}RCCA}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A}) \leq 2n \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}RCCA}}_{\mathsf{mKEM},N}(\mathcal{B}_3) + n \cdot \mathrm{Adv}^{\mathsf{OT\text{-}IND\text{-}RCCA}}_{\mathsf{DEM}}(\mathcal{B}_4),$$

where n is (an upper bound on) the number of recipients of the challenge vector.[10] *Additionally, if* mKEM *is δ_1-correct and* DEM *is δ_2-correct, then* mmPKE *is δ-correct with $\delta(n) \leq n(\delta_1(n) + \delta_2(n))$.*

6.2 An Optimized Construction

For long messages, the mKEM/DEM approach as used in our generic construction is very efficient. However, if messages are as short as the blocksize of the DEM, then we can optimize the construction by directly encrypting all messages in the DEM instead of encrypting keys and then including separate encryptions for each message. Specifically, we instantiate the classically secure mKEM with an mmPKE based on the Hashed ElGamal encryption scheme (also called DHIES in e.g. [1]) from [36] together with a post-quantum secure mKEM. The resulting mmPKE is described in Fig. 11.

Remark 4 (Values in the hash). In our construction in Fig. 11 the DEM key k_j is computed as $H((X_j)^y, X_j, j)$. Including X_j enables a tighter reduction to DSDH.

[10] The bound is in fact slightly tighter — the factor by the mIND-CCA and mIND-RCCA advantages can be replaced by twice the number of different messages in an encrypted message vector (i.e., the number of mKEM instances used in a single encryption).

Including j is necessary to assume only one-time security of DEM (which means that it can be deterministic). Indeed, consider our scheme modified so that j is not hashed. Now if a public key X_j is a receiver of two messages $\vec{m}[i]$ and $\vec{m}[i']$ in one vector, then the DEM keys for $\vec{m}[i]$ and $\vec{m}[i']$ are the same, which requires multi-message security.

The following theorem formalizes security of Opt-mmPKE. Due to space constraints, we defer the proof to Appendices B.5 and B.6.

Theorem 7. *Let* mKEM, DEM *and* H *be an mKEM, a DEM and a hash function. Let* G *be a group of order* p*, generated by* g*. Further, let* $\mathsf{mmPKE} = \mathsf{Opt\text{-}mmPKE}[\mathsf{mKEM}, \mathsf{DEM}, G, g, p, H]$ *be defined as in Fig. 11. Moreover, let* $\mathsf{leak}(\vec{m}) := (\{i : \vec{m}[i] = \vec{m}[i+1]\}, |\vec{m}[1]|, \ldots, |\vec{m}[n]|)$*. For any integer* N *and for any (classical or quantum) adversary* $\mathcal{A}$*, there exist (classical or quantum) adversaries* $\mathcal{B}_1$ *to* $\mathcal{B}_8$ *such that*

$$\mathrm{Adv}^{\mathsf{mmIND\text{-}CCA}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A}) \leq 2n \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM},N}(\mathcal{B}_1) + n \cdot \mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{DEM}}(\mathcal{B}_2) + 2 \cdot \frac{q_h^2}{2^\kappa} \text{ and}$$
$$\mathrm{Adv}^{\mathsf{mmIND\text{-}RCCA}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A}) \leq 2n \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}RCCA}}_{\mathsf{mKEM},N}(\mathcal{B}_4) + n \cdot \mathrm{Adv}^{\mathsf{IND\text{-}RCCA}}_{\mathsf{DEM}}(\mathcal{B}_5) \text{ and}$$
$$\mathrm{Adv}^{\mathsf{mmIND\text{-}CCA}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A}) \leq 2n \cdot \mathsf{EG} + n \cdot \mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{DEM}}(\mathcal{B}_6) + 2n \cdot \mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM}}(\mathcal{B}_7) \text{ and}$$
$$\mathrm{Adv}^{\mathsf{mmIND\text{-}RCCA}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A}) \leq 2n \cdot \mathsf{EG} + n \cdot \mathrm{Adv}^{\mathsf{IND\text{-}RCCA}}_{\mathsf{DEM}}(\mathcal{B}_8),$$

where $\mathsf{EG} = \left(e^2 q_c \cdot \mathrm{Adv}^{\mathsf{DSDH}}_{G,g,p}(\mathcal{B}_3) + \frac{q_{d_1}+2q_h}{p}\right)$*, the hash functions* H *and* G *are modeled as random oracles,* e *is the Euler number,* n *is (an upper bound on) the number of recipients of the challenge vector, and* q_c*,* q_{d_1} *and* q_h *are (upper bounds on) the number of queries to the oracle* **Cor***, the oracle* $\mathbf{Dec}_1$ *and the random oracle, respectively. Additionally, if* mPKE *is* δ_1 *correct and* DEM *is* δ_2*-correct, then* mmPKE *is* δ*-correct with* $\delta(n) \leq n(\delta_1(n) + \delta_2(n))$*.*

7 Implementation and Benchmarks

In order to evaluate the computational performance of the constructions proposed in this paper, we implement the core underlying primitive, the mIND-CCA-secure mKEM based on the NIST PQC finalist Kyber. More specifically, we adapt the code optimized for 64-bit Intel platforms featuring the AVX2 vector instruction set by the Kyber submission team[11].

Benchmarks. We benchmark our Kyber-mKEM implementation on a single core of an Intel Core i7-4770K (Haswell) CPU with HT and TurboBoost turned off. All code is compiled with clang-11.0.1 and optimization flags `-mavx2 -mbmi2 -mpopcnt -march=native -mtune=native -O3 -fomit-frame-pointer`. We report median cycle counts for key generation and decapsulation of a single user over 1000 experiments. For encapsulation, we report median cycle counts over

[11] See https://github.com/pq-crystals/kyber/tree/master/avx2

1000 experiments for each size of the set of recipients; for these measurements we use the monolithic API (see Appendix C). We compare the mKEM cycle counts to a naive solution that encapsulates to each user individually using unmodified Kyber. For these benchmarks of Kyber we again report median cycle counts for a single key generation and decapsulation over 1000 experiments. For encapsulation we benchmark a single-recipient encapsulation operation and multiply this cycle count by the number of recipients. The results are collected in Tables 1, 3 and 4.

Note that our comparison is not exactly comparing apples to apples for two reasons: First, the optimized mKEM approach makes sure that all participants have the same shared key, while with the naive solution is an mmKEM, i.e., the encapsulating party has individual keys shared with each of the other participants. If the former is required, the naive approach would need to use the individual shared keys to encrypt and authenticate the joint group key. Second, when reporting the public-key size for the mKEM solution we omit the 32 bytes for the public seed needed to derive the matrix $\mathbf{A}$, while in unmodified Kyber these 32 bytes are part of the public key. One could decide to save those 32 bytes also in Kyber and handle the seed on application level like for the mKEM.

As expected, the cycle counts for key generation and decapsulation increase because of the additional effort required to achieve adaptive security. However, we see that already for rather small sets of recipients the cycle counts of encapsulation decrease significantly. This is because the most expensive operation of computing the first ciphertext component is amortized across recipients. Even more importantly, we see a massive decrease in ciphertext size reaching a factor of about 4.8 at security level 5 for 1000 recipients.

n	Kyber mKEM				Naive ($n\times$ Kyber)			
	cycles		bytes		cycles		bytes	
1	**gen**:	82772			**gen**:	69292		
	dec:	135544			**dec**:	69756		
1	**enc**:	104452	**pk**:	1568	**enc**:	88740	**pk**:	1568
			ct:	3137			**ct**:	1568
2	**enc**:	147716	**pk**:	3136	**enc**:	177480	**pk**:	3136
			ct:	3458			**ct**:	3136
10	**enc**:	512928	**pk**:	15680	**enc**:	887400	**pk**:	15680
			ct:	6026			**ct**:	15680
100	**enc**:	4929876	**pk**:	156800	**enc**:	8874000	**pk**:	156800
			ct:	34916			**ct**:	156800
1000	**enc**:	48633528	**pk**:	1568000	**enc**:	88740000	**pk**:	1568000
			ct:	323816			**ct**:	1568000

Table 1: Intel Haswell cycle counts and transmitted bytes for Kyber mKEM and naive $n\times$ application of Kyber at NIST security level 5 (Kyber1024)

References

[1] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. *CT-RSA 2001*, vol. 2020 of *LNCS*, 143–158. 2001.

[2] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange - A new hope. *USENIX Security 2016*, 327–343. 2016.

[3] J. Alwen, D. Hartmann, E. Kiltz, and M. Mularczyk. Server-aided continuous group key agreement. Cryptology ePrint Archive, Report 2021/1456, 2021.

[4] A. Ambainis, M. Hamburg, and D. Unruh. Quantum security proofs using semi-classical oracles. *CRYPTO 2019, Part II*, vol. 11693 of *LNCS*, 269–295. 2019.

[5] N. Aviram, B. Dowling, I. Komargodski, K. G. Paterson, E. Ronen, and E. Yogev. Practical (post-quantum) key combiners from one-wayness and applications to tls. Cryptology ePrint Archive, Report 2022/065, 2022.

[6] M. Barbosa and P. Farshim. Randomness reuse: Extensions and improvements. *11th IMA International Conference on Cryptography and Coding*, vol. 4887 of *LNCS*, 257–276. 2007.

[7] M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. *CRYPTO 2006*, vol. 4117 of *LNCS*, 602–619. 2006.

[8] M. Bellare, A. Boldyreva, K. Kurosawa, and J. Staddon. Multirecipient encryption schemes: How to save on bandwidth and computation without sacrificing security. *IEEE Transactions on Information Theory*, 53(11):3927–3943, 2007.

[9] M. Bellare, A. Boldyreva, and J. Staddon. Randomness re-use in multi-recipient encryption schemeas. *PKC 2003*, vol. 2567 of *LNCS*, 85–99. 2003.

[10] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, and D. Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, 206–226. 2019.

[11] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, and D. Stehlé. CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. Cryptology ePrint Archive, Report 2017/634, 2017.

[12] M. Campagna and A. Petcher. Security of hybrid key encapsulation. Cryptology ePrint Archive, Report 2020/1364, 2020.

[13] R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing chosen-ciphertext security. *CRYPTO 2003*, vol. 2729 of *LNCS*, 565–582. 2003.

[14] S. Chatterjee and P. Sarkar. Multi-receiver identity-based key encapsulation with shortened ciphertext. *INDOCRYPT 2006*, vol. 4329 of *LNCS*, 394–408. 2006.

[15] H. Cheng, X. Li, H. Qian, and D. Yan. CCA secure multi-recipient KEM from LPN. *ICICS 18*, vol. 11149 of *LNCS*, 513–529. 2018.

[16] K.-M. Chung, S. Fehr, Y.-H. Huang, and T.-N. Liao. On the compressed-oracle technique, and post-quantum security of proofs of sequential work. Cryptology ePrint Archive, Report 2020/1305, 2020.

[17] J. Don, S. Fehr, C. Majenz, and C. Schaffner. Online-extractability in the quantum random-oracle model. Cryptology ePrint Archive, Report 2021/280, 2021.

[18] E. S. V. Freire, D. Hofheinz, E. Kiltz, and K. G. Paterson. Non-interactive key exchange. *PKC 2013*, vol. 7778 of *LNCS*, 254–271. 2013.

[19] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *CRYPTO'99*, vol. 1666 of *LNCS*, 537–554. 1999.

[20] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, 2013.

[21] C. Gentry and B. Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). *EUROCRYPT 2009*, vol. 5479 of *LNCS*, 171–188. 2009.

[22] F. Giacon, F. Heuer, and B. Poettering. KEM combiners. *PKC 2018, Part I*, vol. 10769 of *LNCS*, 190–218. 2018.

[23] D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On robust combiners for oblivious transfer and other primitives. *EUROCRYPT 2005*, vol. 3494 of *LNCS*, 96–113. 2005.

[24] K. Hashimoto, S. Katsumata, E. Postlethwaite, T. Prest, and B. Westerbaan. A concrete treatment of efficient continuous group key agreement via multi-recipient pkes. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 1441–1462. 2021.

[25] A. Herzberg. On tolerant cryptographic constructions. *CT-RSA 2005*, vol. 3376 of *LNCS*, 172–190. 2005.

[26] H. Hiwatari, K. Tanaka, T. Asano, and K. Sakumoto. Multi-recipient public-key encryption from simulators in security proofs. *ACISP 09*, vol. 5594 of *LNCS*, 293–308. 2009.

[27] D. Jao and L. De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, 19–34. 2011.

[28] S. Katsumata, K. Kwiatkowski, F. Pintore, and T. Prest. Scalable ciphertext compression techniques for post-quantum KEMs and their applications. *ASIACRYPT 2020, Part I*, vol. 12491 of *LNCS*, 289–320. 2020.

[29] K. Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. *PKC 2002*, vol. 2274 of *LNCS*, 48–63. 2002.

[30] J. Li, X. Tang, Z. Wei, Y. Wang, W. Chen, and T. yu an. Identity-based multi-recipient public key encryption scheme and its application in iot. *Mobile Networks and Applications*, 26, 2021.

[31] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. *CT-RSA 2011*, vol. 6558 of *LNCS*, 319–339. 2011.

[32] T. Matsuda and G. Hanaoka. Key encapsulation mechanisms from extractable hash proof systems, revisited. *PKC 2013*, vol. 7778 of *LNCS*, 332–351. 2013.

[33] T. Matsuda and J. C. N. Schuldt. A new key encapsulation combiner. *2018 International Symposium on Information Theory and Its Applications (ISITA)*, 698–702. 2018.

[34] M. A. Nielsen and I. Chuang. Quantum computation and quantum information, 2002.

[35] J. H. Park, K. T. Kim, and D. H. Lee. Cryptanalysis and improvement of a multi-receiver identity-based key encapsulation at INDOCRYPT 06. *ASIACCS 08*, 373–380. 2008.

[36] A. Pinto, B. Poettering, and J. C. N. Schuldt. Multi-recipient encryption, revisited. *ASIACCS 14*, 229–238. 2014.

[37] N. P. Smart. Efficient key encapsulation to multiple parties. *SCN 04*, vol. 3352 of *LNCS*, 208–219. 2005.

[38] N. Sullivan and C. Wood. Hashing to elliptic curves (version 13). IETF Internet Draft, 2021.

[39] D. Unruh. Revocable quantum timed-release encryption. *EUROCRYPT 2014*, vol. 8441 of *LNCS*, 129–146. 2014.

[40] Z. Yang. On constructing practical multi-recipient key-encapsulation with short ciphertext and public key. 8(18), 2015.

[41] K. Yoneyama. Post-quantum variants of ISO/IEC standards: Compact chosen ciphertext secure key encapsulation mechanism from isogeny. *Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop*, SSR'19, 13–21. 2019.

[42] M. Zhandry. How to record quantum queries, and applications to quantum indifferentiability. *CRYPTO 2019, Part II*, vol. 11693 of *LNCS*, 239–268. 2019.

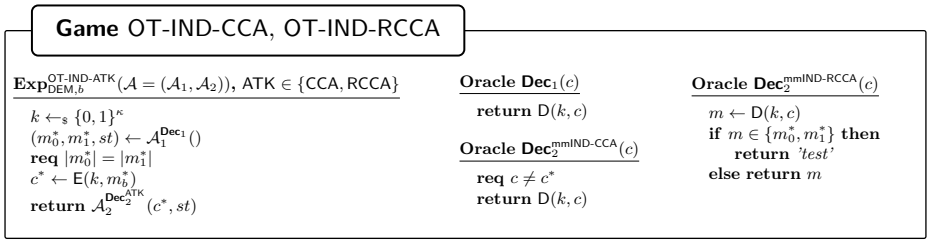

Fig. 12: The One-Time CCA and RCCA security games for DEM.

A Additional Preliminaries

A.1 Assumptions

We recall a variant of the computational Diffie-Hellman assumption from [18] used in our optimized construction in Section 6.

Definition 10 (Double Strong Diffie-Hellman Assumption). *Let $\mathcal{G} = (\mathbb{G}, p, g)$ be a cyclic group of prime order p with generator g. We define the advantage of an algorithm $\mathcal{A}$ in solving the* Double-Sided Strong Diffie-Hellman problem(DSDH) *with respect to $\mathcal{G}$ as*

$$\mathrm{Adv}^{\mathsf{DSDH}}_{\mathcal{G}}(\mathcal{A}) = \left[Z = g^{xy} \,\middle|\, \begin{matrix} x, y \leftarrow_{\$} \mathbb{Z}_p^2 \\ Z \leftarrow_{\$} \mathcal{A}^{\mathbf{O}_x(\cdot,\cdot),\mathbf{O}_y(\cdot,\cdot)}(\mathbb{G}, p, g, g^x, g^y), \end{matrix} \right]$$

where $\mathbf{O}_x, \mathbf{O}_y$ are oracles which on input U, V output 1, iff $U^x = V$ or $U^y = V$ respectively. The probability is taken over the random coins of the group generator, the choice of x and y and the adversaries random coins.

A.2 Data Encapsulation Mechanisms

A Data encapsulation mechanism (DEM) scheme consists of an encryption algorithm $\mathsf{E}(k, m) \to c$ and a decryption algorithm $\mathsf{D}(k, c) \to m$. We define the one-time (R)CCA security for DEMs. We note that one-time security can also be achieved by more efficient deterministic schemes.

Definition 11. *Let* DEM *be a DEM scheme. For* ATK $\in$ {CCA, RCCA}*, we define the advantage of an adversary $\mathcal{A}$ against the* ATK $\in$ {OT-IND-ATK} *security of* DEM *as*

$$\mathrm{Adv}^{\mathsf{ATK}}_{\mathsf{DEM}}(\mathcal{A}) = \left| \Pr[\mathrm{Exp}^{\text{OT-IND-ATK}}_{\mathsf{DEM},1}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{Exp}^{\text{OT-IND-ATK}}_{\mathsf{DEM},0}(\mathcal{A}) \Rightarrow 1] \right|,$$

where $\mathrm{Exp}^{\text{OT-IND-ATK}}_{\mathsf{DEM},b}(\mathcal{A})$ is defined in Fig. 12.

A.3 Correctness Notions

Here we gather the formal correctness notions for the various batched KEM/PKE notions in this work.

mmPKE Correctness. Let $\delta\colon \mathbb{N} \to [0,1]$. An mmPKE scheme mmPKE is *δ-correct*, if for all $n \in \mathbb{N}$ and message vectors $\vec{m}$,

$$\Pr\left[\begin{array}{c}\exists i \in [N]: \\ \mathsf{mmDec}(\mathsf{pp}, \mathsf{sk}_i, c_i) \neq \vec{m}[i]\end{array}\;\middle|\;\begin{array}{c}\mathsf{pp} \leftarrow_{\$} \mathsf{mmSetup} \\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow_{\$} \mathsf{mmKGen}(\mathsf{pp}) \forall i \in [N] \\ \vec{m} \leftarrow_{\$} \mathcal{M}^N, C \leftarrow_{\$} \mathsf{mmEnc}(\mathsf{pp}, \vec{\mathsf{pk}}, \vec{m}), \\ c_i \leftarrow \mathsf{mmExt}(\mathsf{pp}, C, i)\end{array}\right] \leq \delta(n)$$

mKEM Correctness. Let the real valued function $\delta\colon \mathbb{N} \to [0,1]$. An mKEM is *$\delta$-correct*, if every recipient can decapsulate the correct key with high probability. Formally, we require that for all $n \in \mathbb{N}$,

$$\Pr\left[\begin{array}{c}\exists i \in [n]: \\ \mathsf{mDecaps}(\mathsf{pp}, \mathsf{sk}_i, c_i) \neq k\end{array}\;\middle|\;\begin{array}{c}\mathsf{pp} \leftarrow_{\$} \mathsf{mSetup}() \\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow_{\$} \mathsf{mKGen}(\mathsf{pp}) \text{ for } i \in [n] \\ (C, k) \leftarrow_{\$} \mathsf{mEncaps}(\mathsf{pp}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n), \\ c_i \leftarrow \mathsf{mExt}(\mathsf{pp}, C, i) \text{ for } i \in [n]\end{array}\right] \leq \delta(n)$$

We will omit the dependence of δ on n if it is clear from context.

mPKE Correctness. An mPKE scheme mPKE is *δ-correct*, if for each message m,

$$\mathop{\mathbb{E}}_{\substack{\mathsf{pp}\in\mathsf{mSetup} \\ \mathsf{pk}\in\mathsf{mKGen}}}\left[\max_{m\in\mathcal{M}} \Pr\left[m \neq \mathsf{mDec}(\mathsf{sk}, (c^{\mathsf{i}}, c^{\mathsf{d}}))\;\middle|\;\begin{array}{c} r \leftarrow_{\$} \{0,1\}^{\kappa} \\ c^{\mathsf{i}} \leftarrow \mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp}, r), \\ c^{\mathsf{d}} \leftarrow \mathsf{mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}, m, r)\end{array}\right]\right] \leq \delta.$$

B Deferred Proofs

B.1 Security Proof for the mPKE Compiler

Theorem 1. *Let* $\mathsf{mPKE} = \mathsf{Comp}[\mathsf{mPKE}, H]$*, where* mPKE *is an mPKE with a group operation over the space of public keys and H is a hash function, be defined as in Fig. 5. For any number of recipients N and for any (classical or quantum) adversary $\mathcal{A}$, there exist (classical or quantum) adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\mathrm{Adv}^{\mathsf{mIND\text{-}CPA}^{\mathsf{corr}}}_{\mathsf{mPKE},N}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{mIND\text{-}CPA}}_{\mathsf{mPKE},N}(\mathcal{B}_1) + 2N \cdot \mathrm{Adv}^{\mathsf{mRND\text{-}PK}}_{\mathsf{mPKE}}(\mathcal{B}_2),$$

where H is modeled as a random oracle. Additionally, if mPKE *is δ-correct, then* $\mathsf{Comp}[\mathsf{mPKE}, H]$ *is δ-correct as well.*

Proof. The classical setting. We define a sequence of game hops transitioning from the $\mathsf{mIND\text{-}CPA}^{\mathsf{corr}}$ experiment with the bit $b = 0$ to the $\mathsf{mIND\text{-}CPA}^{\mathsf{corr}}$ experiment with $b = 1$.

Game G_0. This is the experiment with $b = 0$.

Game G_1. This game hop changes how the key N pairs are generated by the challenger. In particular, in G_1 both the left and the right key pairs are generated using $\mathsf{mPKE.mKGen}$, and H is programmed to the sum of public keys afterwards. Using the standard hybrid argument and the straightforward reduction, we get the following lemma

Lemma 2. *For any adversary $\mathcal{A}$, there exists a reduction $\mathcal{B}_2$ such that*

$$\Pr\left[G_1(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[G_0(\mathcal{A}) \Rightarrow 1\right] \leq N \cdot \mathrm{Adv}^{\mathsf{mRND\text{-}PK}}_{\mathsf{mPKE}}(\mathcal{B}_2).$$

Game G_2. This game hop switches m_0^* to m_1^* in the first invocation of $\mathsf{mPKE.mEnc}$ executed when computing the challenge ciphertext. More precisely, in G_2, the challenge ciphertext is generated as follows.

$(r_{\mathsf{first}}, r_{\mathsf{second}}) \leftarrow_{\$} \{0,1\}^{2\kappa}$
$(c^{\mathsf{i}}_{\mathsf{first}}, c^{\mathsf{i}}_{\mathsf{second}}) \leftarrow (\mathsf{mPKE.mEnc}^{\mathsf{i}}(\mathsf{pp}; r_{\mathsf{first}}), \mathsf{mPKE.mEnc}^{\mathsf{i}}(\mathsf{pp}; r_{\mathsf{second}}))$
for $j \in [n]$ **do**
  $(\mathsf{pk}_{\mathsf{left},i^*_j}, s_{i^*_j}) \leftarrow \mathsf{pk}_{i^*_j}$ // $\mathsf{pk}_{i^*_j}$ was generated at the beginning of the experiment
  $\mathsf{pk}_{\mathsf{right},i^*_j} \leftarrow H(s) - \mathsf{pk}_{\mathsf{left},i^*_j}$
  $\mathsf{swapCtx}_j \leftarrow_{\$} \{0,1\}$
  if $\mathsf{swapCtx} = 0$ **then** $(\mathsf{pk}_{\mathsf{first},i^*_j}, \mathsf{pk}_{\mathsf{second},i^*_j}) \leftarrow (\mathsf{pk}_{\mathsf{left},i^*_j}, \mathsf{pk}_{\mathsf{right},i^*_j})$
  else $(\mathsf{pk}_{\mathsf{first},i^*_j}, \mathsf{pk}_{\mathsf{second},i^*_j}) \leftarrow (\mathsf{pk}_{\mathsf{right},i^*_j}, \mathsf{pk}_{\mathsf{left},i^*_j})$
  $c^{\mathsf{d}}_{\mathsf{first},j} \leftarrow \mathsf{mPKE.mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{first},i^*_j}, \boxed{m_1^*}, r_{\mathsf{first}})$
  $c^{\mathsf{d}}_{\mathsf{second},j} \leftarrow \mathsf{mPKE.mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{second},i^*_j}, \boxed{m_0^*}, r_{\mathsf{second}})$ $c^{\mathsf{d}}_j \leftarrow (c^{\mathsf{d}}_{\mathsf{first},j}, c^{\mathsf{d}}_{\mathsf{second},j}, \mathsf{swapCtx}_j)$

We next show that games 1 and 2 are indistnguishable, assuming that mPKE is $\mathsf{mIND\text{-}CPA}$ secure (without corruptions), as formalized by the following lemma.

Lemma 3. *For any adversary $\mathcal{A}$, there exists a reduction $\mathcal{B}_1$ such that*

$$\Pr\left[G_2(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[G_1(\mathcal{A}) \Rightarrow 1\right] \leq \mathrm{Adv}^{\mathsf{mIND\text{-}CPA}}_{\mathsf{mPKE},N}(\mathcal{B}_1).$$

Proof: Let $\mathcal{A}$ be any adversary. The reduction $\mathcal{B}_1$ is described in detail in Fig. 13.

At a high level, $\mathcal{B}_1$ emulates for $\mathcal{A}$ the two instances of mPKE, one with the help of its challenger and one by running it itself. That is, each recipient $i \in [N]$ has a left and a right mPKE key. One of these keys is the i-th key received from $\mathcal{B}_1$'s challenger and the other is generated by $\mathcal{B}_1$. To decide which is which, $\mathcal{B}_1$ tosses a random coin bit_i: if $\mathsf{bit}_i = 0$, the left key is taken from the challenger, and $\mathsf{bit}_i = 1$, it is the right key. This way, $\mathcal{B}_1$ can handle corruptions: if the i-th recipient is corrupted, $\mathcal{B}_1$ outputs the mPKE key it generated and the bit $\mathsf{swapPk}_i = 1 - \mathsf{bit}_i$. This means that $\mathcal{B}_1$ pretends that the bit swapPk_i chosen by the i-th execution of the key generation at the beginning of the emulated $\mathsf{mIND\text{-}CPA}^{\mathsf{corr}}$ experiment was $\mathsf{swapPk}_i = \mathsf{bit}_i$.

Further, $\mathcal{B}_1$ computes the challenge ciphertext as follows. Recall that the encryption algorithm of the compiled scheme runs mPKE encryption twice. For the first invocation, $\mathcal{B}_1$ uses its challenger with m_0^* (as encrypted in G_1) and

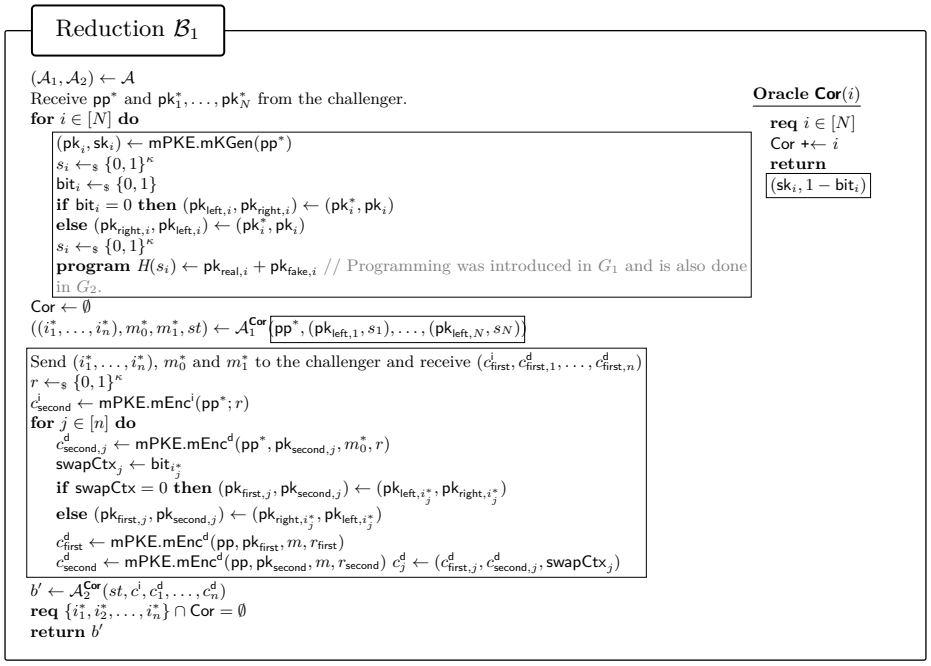

Fig. 13: The reduction $\mathcal{B}_1$ using a distinguisher $\mathcal{A}$ between G_1 and G_2 to break the mIND-CPA security of mPKE. The parts where $\mathcal{B}_1$'s experiment differs from mIND-CPA are marked by boxes.

m_1^* (as encrypted in G_2), while for the second it encrypts m_0^* itself (as in both games). For this to work, we need that for each recipient $j \in [n]$ of the challenge, the public key used in the first invocation comes from $\mathcal{B}_1$'s challenger. Therefore, $\mathcal{B}_1$ sets the bit $\mathsf{swapCtx}_j$ to $\mathsf{bit}i_j^*$, i.e., the bit it chose for the j-th recipient during key generation.

Observe that for each recipient $i \in [N]$, the only part of $\mathcal{A}$'s view that depends on bit_i are the bits swapPk_i outputted in case i is corrupted and $\mathsf{swapCtx}_j$ outputted in case i receives the challenge. Since these cases are exclusive, swapPk_i or $\mathsf{swapCtx}_j$ is distributed uniformly at random in $\mathcal{A}$'s experiment. ■

Game G_3. The game is the same as G_2, except the challenger encrypts m_1^*. We next show that games 2 and 3 are indistnguishable, assuming that mPKE is mIND-CPA secure (without corruptions), as formalized by the following lemma.

Lemma 4. *For any adversary $\mathcal{A}$, there exists a reduction $\mathcal{B}_1$ such that*

$$\Pr[G_3(\mathcal{A}) \Rightarrow 1] - \Pr[G_2(\mathcal{A}) \Rightarrow 1] \leq \mathrm{Adv}^{\mathsf{mIND\text{-}CPA}}_{\mathsf{mPKE},N}(\mathcal{B}_1).$$

Proof: The reduction $\mathcal{B}_1$ is analogous to the one used in the proof of the previous claim. The only difference is that the challenge is embedded in the second invocation of mPKE. ■

Game G_4. The game is the same as G_3, except the key pairs are generated as in the original mPKE scheme. This is the $\mathsf{mIND\text{-}CPA}$ experiment with $b = 1$. By the standard hybrid argument, we get the following lemma

Lemma 5. *For any adversary $\mathcal{A}$, there exists a reduction $\mathcal{B}_2$ such that*

$$\Pr[G_4(\mathcal{A}) \Rightarrow 1] - \Pr[G_3(\mathcal{A}) \Rightarrow 1] \leq N \cdot \mathrm{Adv}^{\mathsf{mRND\text{-}PK}}_{\mathsf{mPKE}}(\mathcal{B}_2).$$

Combining the four lemmas and applying the union bound concludes the proof.

The quantum setting. Note that although the random oracle is programmed, all programmed values are fixed a priori and are independent of the adversary's behavior. Therefore, to lift the previous proof to the quantum setting, we can simply simulate the necessary quantum random oracle by defining a programmed random oracle that answers with the programmed values, if such a point is queried, or with an internal random oracle if no programmed point is queried. Since everything is committed from the beginning, giving the quantum adversary quantum access to this oracle is indistinguishable from a regular quantum random oracle and the same reduction strategy works. □

B.2 Security Proofs for the Combiner

In this section, we prove the two security statements for the mKEM combiner. We start with the more more difficult proof of Theorem 4.

Theorem 4. *Let* $\mathsf{mKEM} = \mathsf{Comb}[\mathsf{mKEM1}, \mathsf{mKEM2}, \mathsf{PRF}]$*, where* $\mathsf{mKEM1}$ *and* $\mathsf{mKEM2}$ *are some mKEM schemes and* PRF *is a PRF, be defined as in Fig. 8. Let* $\mathrm{dual}(\mathsf{PRF})$ *denote the PRF obtained by swapping the input and key of* PRF*, i.e.,* $\mathrm{dual}(\mathsf{PRF})(k, x) = \mathsf{PRF}(x, k)$*. For any $N \in \mathbb{N}$ and for any (classical or quantum) adversary $\mathcal{A}$, there exist (classical or quantum) adversaries $\mathcal{B}_1$ to $\mathcal{B}_3$ and $\mathcal{B}'_1$ to $\mathcal{B}'_3$ such that*

$$\mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM},N}(\mathcal{A}) \leq 2 \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM1},N}(\mathcal{B}_1) + \mathrm{Adv}^{\mathsf{PRF}}_{\mathsf{PRF}}(\mathcal{B}_2) + 2N \cdot \mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM2}}(\mathcal{B}_3) \text{ and}$$
$$\mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM},N}(\mathcal{A}) \leq 2 \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM2},N}(\mathcal{B}'_1) + \mathrm{Adv}^{\mathsf{PRF}}_{\mathrm{dual}(\mathsf{PRF})}(\mathcal{B}'_2) + 2N \cdot \mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM1}}(\mathcal{B}'_3).$$

Additionally, if $\mathsf{mKEM1}$ *is δ_1-correct and* $\mathsf{mKEM2}$ *is δ_2-correct, then* mKEM *is $(\delta_1 + \delta_2)$-correct.*

Proof. The correctness bound follows directly from the fact that if one of the two mKEMs suffers a decapsulation error, then the combined scheme does so as well.

We prove only the first statement for the case that $\mathsf{mKEM1}$ is secure. The second proof is analogous. Our proof works for both the classical and the quantum setting (it is a straightline black-box reduction in the standard model).

We define a sequence of games transitioning from G_0, which is the real $\mathsf{mIND\text{-}CCA}$ experiment with the bit $b = 0$, to G_5, which is the ideal experiment with $b = 1$. The games are defined in Fig. 14.

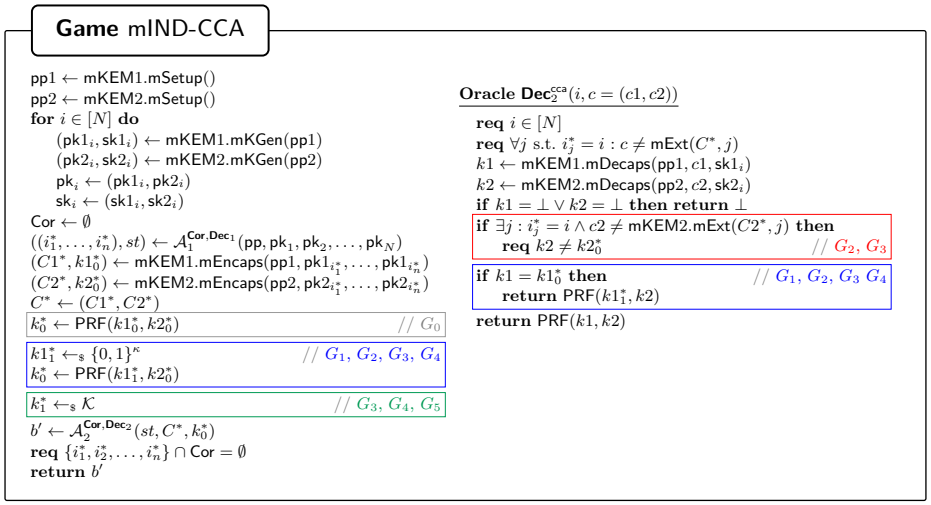

Fig. 14: The games for the proof of Theorem 4.

Game G_1. The game hop replaces the key $k1_0^*$ encapsulated by mKEM1 when the challenge is computed by a random and independent $k1_1^*$ in all inputs to PRF. Let $\mathcal{A}$ be any adversary. We next construct a reduction $\mathcal{B}_1$ such that

$$\Pr\left[G_1(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[G_0(\mathcal{A}) \Rightarrow 1\right] \leq \mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM1},N}(\mathcal{B}_1).$$

$\mathcal{B}_1$ simply runs $\mathcal{A}$ and evaluates mKEM2 itself. Moreover, it evaluates mKEM1 using its oracles. Let k^* be $\mathcal{B}_1$'s challenge key. $\mathcal{B}_1$ computes $\mathcal{A}$'s challenge key as $\mathsf{PRF}(k^*, k2_0^*)$, where $k2_0^*$ is generated by $\mathcal{B}_1$. If $\mathcal{A}$ sends $(c1, c2)$ to the decapsulation oracle such that $\mathcal{B}_1$ cannot query $c1$ to its oracle, then $\mathcal{B}_1$ outputs $\mathsf{PRF}(k^*, k2)$, where $k2$ is computed by decapsulating $c2$.

If $\mathcal{B}_1$ is in the mIND-CCA experiment with the bit $b = 0$, then it uses the real key $k^* = k1_0^*$ as the PRF input, so the experiment is the same as G_0. Now assume $b = 1$. Then $\mathcal{B}_1$ answers all calls to the decapsulation oracle that contaion $c2$ with $k^* = k1_1^*$, which results in an independent and random key for $\mathcal{A}$. Therefore, $\mathcal{B}_1$ simulates G_1.

Game G_2. The game hop adds an additional check enforcing that if the adversary inputs a ciphertext with a mKEM2 component $c2$ to the decapsulation oracle which is not an extraction of the mKEM2 component of the challenge, then mKEM2 decapsulates a different key than the mKEM2 challenge key $k2_0^*$.

We notice that G_1 and G_2 are identical unless there is a collision on one of the N instances of mKEM2 as defined in Definition 8. Therefore, for the straightforward reduction $\mathcal{B}_2$, we have

$$\Pr\left[G_2(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[G_1(\mathcal{A}) \Rightarrow 1\right] \leq N \cdot \mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM2}}(\mathcal{B}_2).$$

Game G_3. The game hop replaces the challenge key by k_1^*. For any adversary $\mathcal{A}$ there exists a reduction $\mathcal{B}_3$ such that

$$\Pr\left[G_3(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[G_2(\mathcal{A}) \Rightarrow 1\right] \leq \mathrm{Adv}^{\mathsf{PRF}}_{\mathsf{PRF}}(\mathcal{B}_2).$$

Observe that $k1_1^*$ is independent of the experiment apart from the PRF evaluations. Therefore, $\mathcal{B}_3$ can simply run $\mathcal{A}$, embed its challenge in $\mathcal{A}$'s challenge key and answer $\mathcal{A}$'s decapsulation queries as follows: If $k1 = k1_0^*$ (which $\mathcal{B}_3$ can check, because it evaluates mKEMs itself), then it uses the evaluation oracle for the PRF and computes the rest itself. Else, it computes everything itself. This works because the additional check introduced in G_2 prevents $\mathcal{A}$ from making $\mathcal{B}_3$ trigger a trivial win in the PRF game.

Games G_4 and G_5. The last two hops to G_4 and to G_5 revert the changes made in G_2 and G_1, respectively. The reductions $\mathcal{B}_3$ and $\mathcal{B}_4$ are analogous. □

that the mKEM combiner from Section 3.1 outputs an RCCA secure mKEM assuming (only) that one of the input mKEMs is RCCA secure.

Theorem 3. *Let* $\mathsf{mKEM} = \mathsf{Comb}[\mathsf{mKEM1}, \mathsf{mKEM2}, \mathsf{PRF}]$*, where* $\mathsf{mKEM1}$ *and* $\mathsf{mKEM2}$ *are some mKEM schemes and* PRF *is a PRF, be defined as in Fig. 8. Let* $\mathrm{dual}(\mathsf{PRF})$ *denote the PRF obtained by swapping the input and key of* PRF*, i.e.,* $\mathrm{dual}(\mathsf{PRF})(k, x) = \mathsf{PRF}(x, k)$*. For any* $N \in \mathbb{N}$ *and for any (classical or quantum) adversary* $\mathcal{A}$*, there exist (classical or quantum) adversaries* $\mathcal{B}_1$*,* $\mathcal{B}_2$*,* $\mathcal{B}_1'$*,* $\mathcal{B}_2'$ *s.t.*

$$\mathrm{Adv}^{\mathsf{mIND\text{-}RCCA}}_{\mathsf{mKEM},N}(\mathcal{A}) \leq 2 \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}RCCA}}_{\mathsf{mKEM1},N}(\mathcal{B}_1) + \mathrm{Adv}^{\mathsf{PRF}}_{\mathsf{PRF}}(\mathcal{B}_2) \text{ and}$$
$$\mathrm{Adv}^{\mathsf{mIND\text{-}RCCA}}_{\mathsf{mKEM},N}(\mathcal{A}) \leq 2 \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}RCCA}}_{\mathsf{mKEM2},N}(\mathcal{B}_1') + \mathrm{Adv}^{\mathsf{PRF}}_{\mathrm{dual}(\mathsf{PRF})}(\mathcal{B}_2').$$

Additionally, if $\mathsf{mKEM1}$ *is* δ_1*-correct and* $\mathsf{mKEM2}$ *is* δ_2*-correct, then* mKEM *is* $(\delta_1 + \delta_2)$*-correct.*

Proof. The proof is almost identical to the proof Theorem 4. We only observe that RCCA security does not rely on collision resistance, since all collisions result in the decyrption oracle returning *'test'*.

B.3 Proofs of mKEM Collision Resistance

Theorem 5. *Let* $\mathsf{mKEM} = \mathsf{FO}[\mathsf{mPKE}, H, G_1, G_2]$*, where* mPKE *is any mPKE scheme and* H*,* G_1 *and* G_2 *are any hash functions. For any (classical or quantum) adversary* $\mathcal{A}$*, there exists a (classical or quantum) adversary* $\mathcal{B}$ *such that*

$$\mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{CR}}_{H}(\mathcal{B}).$$

Proof. Let $\mathcal{A}$ be any adversary who outputs c and c' in the CR experiment with mKEM. Assume $\mathcal{A}$ wins, i.e., $c \neq c'$, $k = k' \neq \bot$, where k and k' be the key decapsulated from c and c'. We next use these facts to find a collision in H.

Let $(\mathsf{pk}, \mathsf{sk})$ and pp be the mKEM key pair and the public parameters chosen in the CR experiment. Since $k \neq \bot$, all mKEM decapsulation checks pass, i.e.,

1. $m = \mathsf{mPKE.mDec}(\mathsf{pp}, \mathsf{sk}, c) \neq \bot$,
2. $k = H(m)$,
3. $c = (c^{\mathsf{i}}, c^{\mathsf{d}})$ and $c^{\mathsf{i}} = \mathsf{mPKE.mEnc}^{\mathsf{i}}(\mathsf{pp}; G_1(m))$,
4. $c = (c^{\mathsf{i}}, c^{\mathsf{d}})$ and $c^{\mathsf{d}} = \mathsf{mPKE.mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}, m, G_1(m); G_2(m))$.

Since $k' \neq \bot$, analogous equations hold for $c' = (c^{\mathsf{i}\prime}, c^{\mathsf{d}\prime})$ and the decrypted message m'. Since $c \neq c'$, we have either $c^{\mathsf{i}} \neq c^{\mathsf{i}\prime}$ or $c^{\mathsf{d}} \neq c^{\mathsf{d}\prime}$. We claim that in both cases, this means that $f(m) \neq f(m')$ for some deterministic function f. Indeed, in the first case, $f = \mathsf{mPKE.mEnc}^{\mathsf{i}}$ (by point 3.) and in the second case, $f = \mathsf{mPKE.mEnc}^{\mathsf{d}}$ (by point 4.). Therefore, $m \neq m'$. Moreover, since $H(m) = k = k' = H(m')$ (by point 2.), we get a collision on H. □

Next, we show that DH-based mKEM is collision resistant both in the classical and in the quantum setting. We note that we make no statement about the scheme's (R)CCA security which doesn't hold in the quantum setting. In this setting, the scheme is only collision resistant. Note also that for quantum security, we need H and DEM to be collision resistant against quantum adversaries.

Theorem 8. *Let* $\mathsf{mKEM} = \mathsf{DH\text{-}mKEM}[G, g, p, H, \mathsf{DEM}]$ *be as defined in Fig. 9, where* G *is a group of prime order* p *generated by* g*,* H *is a hash function and* DEM *is a DEM scheme. For any (classical or quantum) adversary* $\mathcal{A}$*, there exist (classical or quantum) adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *such that*

$$\mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{CR}}_{H}(\mathcal{B}_1) + \mathrm{Adv}^{\mathsf{CR}}_{\mathsf{DEM}}(\mathcal{B}_2).$$

Proof. Let $\mathcal{A}$ be any adversary who outputs c and c' in the CR experiment with mKEM. Let k and k' be the decapsulated keys. Assume $\mathcal{A}$ wins, i.e., $c \neq c'$, $k = k'$ and $k, k' \neq \bot$.

Let (g^x, x) be the key pair generated in the CR experiment. We use the following notation: $c = (Y, ctx)$, $c' = (Y', ctx')$, $k^{\mathsf{dem}} = H(\text{'dh-key'}, (Y)^x, Y)$ and $k^{\mathsf{dem}\prime} = H(\text{'dh-key'}, (Y')^x, Y')$.

Since $k \neq \bot$, we have $m = \mathsf{D}(k^{\mathsf{dem}}, ctx) \neq \bot$ and $k = H(\text{'output-key'}, m, Y)$. Since $k' \neq \bot$, we have $m' = \mathsf{D}(k^{\mathsf{dem}\prime}, ctx') \neq \bot$ and $k' = H(\text{'output-key'}, m', Y')$.

If $(m, Y) \neq (m', Y')$, then an adversary $\mathcal{B}_1$ can find collisions in H. Else, $m = m'$ and $Y = Y'$. The latter implies that $(Y)^x = (Y')^x$ so $k^{\mathsf{dem}} = k^{\mathsf{dem}\prime}$. Therefore, $\mathcal{B}_2$ wins in the CR experiment with DEM using k^{dem}, ctx and ctx'. □

B.4 Security Proof for the Modular mmPKE Construction

Theorem 6. *Let* mKEM *and* DEM *be an mKEM and a DEM schemes, and let* $\mathsf{mmPKE} = \mathsf{BB\text{-}mmPKE}[\mathsf{mKEM}, \mathsf{DEM}]$ *be defined as in Fig. 10. Further, let* $\mathsf{leak}(\vec{m}) \coloneqq (\{i : \vec{m}[i] = \vec{m}[i+1]\}, |\vec{m}[1]|, \ldots, |\vec{m}[n]|)$*. For any integer* N *and for any (classical or quantum) adversary* $\mathcal{A}$*, there exist (classical or quantum) adversaries* $\mathcal{B}_1$ *to* $\mathcal{B}_4$ *such that*

$$\mathrm{Adv}^{\mathsf{mmIND\text{-}CCA}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A}) \leq 2n \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM},N}(\mathcal{B}_1) + n \cdot \mathrm{Adv}^{\mathsf{OT\text{-}IND\text{-}CCA}}_{\mathsf{DEM}}(\mathcal{B}_2) \text{ and}$$
$$\mathrm{Adv}^{\mathsf{mmIND\text{-}RCCA}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A}) \leq 2n \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}RCCA}}_{\mathsf{mKEM},N}(\mathcal{B}_3) + n \cdot \mathrm{Adv}^{\mathsf{OT\text{-}IND\text{-}RCCA}}_{\mathsf{DEM}}(\mathcal{B}_4),$$

where n is (an upper bound on) the number of recipients of the challenge vector.[12] *Additionally, if* mKEM *is δ_1-correct and* DEM *is δ_2-correct, then* mmPKE *is δ-correct with $\delta(n) \leq n(\delta_1(n) + \delta_2(n))$.*

Proof. The correctness statement follows directly from the fact that a decapsulation or decryption failure at one position directly causes a decryption failure of the whole mmPKE. Since both building blocks are repeated up to n times, a union-bound yields the claimed bound.

We next consider security. For simplicity, we only consider classical adversaries. Note however, that the reduction is straightline, black-box and in the standard model. Therefore adapting it to the post-quantum setting is straightforward, as long as the underlying mKEM and DEM are post-quantum secure.

Recall that in the mmIND-CCA (resp., mmIND-RCCA) game the challenge ciphertext has the form $((C_1, ctx_1, j_1), \ldots, (C_\ell, ctx_\ell, j_\ell))$. Observe that $\mathsf{leak}(\vec{m}_0^*) = \mathsf{leak}(\vec{m}_1^*)$ ensures that ℓ as well as $j_1, \ldots, j_\ell$ are the same when $\vec{m}_0^*$ is encrypted and when $\vec{m}_1^*$ is encrypted.

This means that we can define the following sequence of $3n$ game hops transitioning from the mmIND-CCA (resp., mmIND-RCCA) experiment with $b = 0$ to the experiment with $b = 1$. Each hop modifies one component (C_i, ctx_i, j_i). In the first hop, the key used to generate ctx_1 is switched from the key k_1 encapsulated in C_1 to an independent key. In the second hop, ctx_1 is switched from an encryption of $\vec{m}_0^*[1]$ to an encryption of $\vec{m}_1^*[1]$. In the third hop, the key used to generate ctx_1 is switched back to k_1. The next $3n - 3$ hops modify $(C_2, ctx_2, j_2), \ldots, (C_\ell, ctx_\ell, j_\ell)$ in the analogous way. Note that the last game is the mmIND-CCA (resp., mmIND-RCCA) experiment with $b = 1$.

Observe that the above sketch does not quite work, because if $\vec{m}_0^*[1] = \vec{m}_1^*[1]$, then the adversary is allowed to corrupt one of its receivers. This way it learns the real DEM key, which allows to distinguish between the first and the second game. To fix this, we modify all of the game hops so that they do not switch anything if $\vec{m}_0^*[j_i] = \vec{m}_1^*[j_i]$ (note that there is no need to switch from $\vec{m}_0^*[j_i]$ to $\vec{m}_1^*[j_i]$ in this case).

Indistinguishability of the first and the second, as well as the third and the fourth games follows from mIND-CCA (resp., mmIND-RCCA) security of mKEM. Indistinguishability of the second and the third game follows from OT-IND-CCA (resp., OT-IND-RCCA) security of DEM. (Each individual game hop is tight.) □

B.5 The First and Second Part of Theorem 7

Proof (of the first statement). We focus on classical adversaries and note that the proof can be adapted to the quantum setting in the straightforward way. We define a standard sequence of game hops transitioning from the mmIND-CCA experiment with $b = 0$ to the mmIND-CCA experiment with $b = 1$. Each hop modifies only how the challenge ciphertext is generated.

[12] The bound is in fact slightly tighter — the factor by the mIND-CCA and mIND-RCCA advantages can be replaced by twice the number of different messages in an encrypted message vector (i.e., the number of mKEM instances used in a single encryption).

Game G_0. This is the mmIND-CCA experiment with the Opt-mmPKE scheme and $b = 0$.

Game G_1. The difference from G_0 is that each key k_{mkem} used to compute the challenge encryption of $\vec{m}_0^*$ is random and independent.

Game G_2. The difference from G_1 is that each key inputted to the DEM is random and independent instead of being computed as $G(k_{\mathsf{mkem}}, k_{\mathsf{dh}})$.

Games G_3. The difference from G_2 is that the DEM ciphertexts $ctx_1, \ldots, ctx_n$ are computed by encrypting elements of $\vec{m}_1^*$ instead of elements of $\vec{m}_0^*$.

Game G_4. This game hop reverts the change from the first two hops, i.e., the difference from G_3 is that the keys inputted to DEM are $G(k_{\mathsf{mkem}}, k_{\mathsf{dh}})$ where k_{mkem} is outputted by mKEM.

The following lemmas bound the advantage of an adversary $\mathcal{A}$ in distinguishung each pair of consecutive games.

Lemma 6. *For any adversary $\mathcal{A}$, there exists a reduction $\mathcal{B}_1$ s.t.*

$$\Pr[G_1(\mathcal{A}) = 1] - \Pr[G_0(\mathcal{A}) = 1] \leq n \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM},N}(\mathcal{B}_1).$$

Proof: We notice that when the challenge is computed, mKEM is invoked at most n times (when all elements of $\vec{m}_0^*$ /$\vec{m}_1^*$ are different). The claim follows by the standard hybrid argument. ■

Lemma 7. *For any adversary $\mathcal{A}$,*

$$\Pr[G_2(\mathcal{A}) = 1] - \Pr[G_1(\mathcal{A}) = 1] \leq \frac{q_h^2}{2^\kappa}.$$

Proof: For a recipient index $j \in [n]$, let $k_j = G(k_{\mathsf{mkem},j}, k_{\mathsf{dh},j})$ be the key inputted to the j-th instance of DEM when the challenge is computed. Recall that $k_{\mathsf{dh},j} = H(*, *, j)$. This means that, unless there are collisions on H, $k_{\mathsf{dh},j}$ are distinct. Moreover, the values $k_{\mathsf{mkem},j}$ are only used in the challenge and are otherwise random and independent of the rest of the experiment (after the first hop). Since G is modeled as a random oracle, this means that each k_j is random and independent of the view of $\mathcal{A}$.

Finally, since H is a random oracle with output domain of size 2^κ, collisions on H happen with probability at most $q_h^2/2^\kappa$. ■

Lemma 8. *For any adversary $\mathcal{A}$, there exists a reduction $\mathcal{B}_2$ s.t.*

$$\Pr[G_3(\mathcal{A}) = 1] - \Pr[G_2(\mathcal{A}) = 1] \leq n \cdot \mathrm{Adv}^{\mathsf{OT\text{-}IND\text{-}CCA}}_{\mathsf{DEM}}(\mathcal{B}_2).$$

Proof: Since the DEM keys are random and independent, the claim follows by the standard hybrid argument. ■

Lemma 9. *For any adversary $\mathcal{A}$, there exists a reduction $\mathcal{B}_1$ s.t.*

$$\Pr[G_3(\mathcal{A}) = 1] - \Pr[G_2(\mathcal{A}) = 1] \leq n \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM},N}(\mathcal{B}_1) + \frac{q_h^2}{2^\kappa}.$$

Proof: The proof is analogous to the proofs of the first two lemmas. ■

□

B.6 Third and fourth part of Theorem 7

Proof (of the third statement). We focus on the classical case, since the theorem is trivial in the quantum case (where the DH assumption is false). Recall that the challenge ciphertext in the $\mathsf{mmIND\text{-}CCA}$ experiment contains $(C_1, j_1), \ldots, (C_\ell, j_\ell)$, $ctx_1, \ldots, ctx_n$ and Y. Since $\mathsf{leak}(\vec{m}_0^*) = \mathsf{leak}(\vec{m}_1^*)$, we know that ℓ as well as $j_1, \ldots, j_\ell$ are the same when $\vec{m}_0^*$ is encrypted and when $\vec{m}_1^*$ is encrypted.

The sequence of game hops. We define the following sequence of $7n$ game hops transitioning from the $\mathsf{mmIND\text{-}CCA}$ experiment with $b = 0$ to the experiment with $b = 1$.

Preventing Collisions In the first n game hops, we add an additional check to the decryption oracle, namely we additionally check if the mKEM part of the ciphertext encapsulates the same key as one of the first i challenge ciphertext parts if decryption is queried for one of the first i recipients. Specifically, in the i-th game, when queried on input $\mathsf{pp}, (c, ctx_j^*, Y^*, j)$ where ctx_j^* is the j-th challenge DEM ciphertext and Y^* is the DH challenge nonce for some $j \leq i$, it also outputs $\perp$, if $\mathsf{mKEM.mDecaps}(\mathsf{pp}, \mathsf{sk}, c) = \mathsf{mKEM.mDecaps}(\mathsf{pp}, \mathsf{sk}, c_j^*)$, where c_j^* is the j-th encapsulation in the challenge ciphertext. This check ensures that the adversary can't trivially decrypt by finding a collision in the (potentially insecure) mKEM. This is necessary in the next step where we replace the keys generated by the DH part with random. If the adversary could produce a forgery, it could use the decryption oracle to detect this switch to random keys. Each of these hops can be tightly reduced to the collision resistance of mKEM.

Note that if we only consider RCCA security, the first n game hops aren't needed. Indeed, the RCCA decryption oracle checks whether the decrypted message is equal to a challenge message and simply outputs Test in that case, so such a query doesn't imply breaking the scheme. So by simply removing the first and last n game hops and switching all definitions to RCCA security, the proof works as well and the third statement of the theorem follows.

Switching the keys The goal of the second n hops is to switch the n DH keys $k_{\mathsf{dh},i}$ used in the encryption of the challenge ciphertext to random keys *if* the message vectors differ at the respective position. Since the adversary can't corrupt these positions and the keys are unchanged at all other positions, we can make all these changes via standard hybrids.

In the next n hops, we replace the final DEM key $G(k_{\mathsf{dh},i}, k_i)$ by random. Since $k_{\mathsf{dh},i}$ is already random for all i and G is a random oracle, this is a purely conceptual change.

In the next n hops, the random values encrypted in $ctx_1, \ldots, ctx_n$ are one by one switched from $\vec{m}_0^*[1]$ to $\vec{m}_1^*[1]$. If the messages were the same, i.e. corruptions would be possible and we didn't change the keys to random, this doesn't change anything. For positions with different messages, we can use the security of DEM. In the final $3n$ hops revert the first $3n$ hops, i.e., the final DEM keys are again computed via G, the keys $k_{\mathsf{dh},1}, \ldots, k_{\mathsf{dh},n}$ are replaced by the real DH keys and we remove the collision check from the decryption oracle.

The reduction to DEM *security.* Observe that after the first $3n$ game hops, the DEM key $G(k_{\mathsf{dh},i}, k_{\mathsf{mkem}})$ used to generate ctx_i is random and independent of the rest of the experiment if the i-th components of $\vec{m}_0^*$ and $\vec{m}_1^*$ are different. Therefore, distinguishing between the games i and $i+1$ for $i \in [n, 2n]$ can be reduced to breaking OT-IND-CCA (resp., OT-IND-RCCA) security of DEM in the straightforward way.

The reduction to DSDH. Let G_0 and G_1 denote the first two games in the sequence, i.e. in G_0 all DH keys are real and in G_1, we swap the real DH key for a random one *if* $\vec{m}_0[0] \neq \vec{m}_1[0]$. We next show that for any adversary $\mathcal{A}$, there exists a reduction $\mathcal{B}$ such that

$$\Pr\left[\mathsf{G}_1(\mathcal{A}) = 1\right] - \Pr\left[\mathsf{G}_0(\mathcal{A}) = 1\right] \leq e^2 q_c \cdot \mathrm{Adv}^{\mathsf{DSDH}}_{G,g,p}(\mathcal{B}) + \frac{q_{d_1}}{p} + \frac{q_h}{p}. \tag{1}$$

An analogous reduction can be constructed for the game hops $2n+2$ to $4n$ and $5n+1$ to $6n$. Since we make this hop at most $2n$ times, the bound follows from the equation. We conclude by proving that Eq. (1) holds.

Observe that G_0 and G_1 are identical unless $\vec{m}_0^*[1] \neq \vec{m}_1^*[1]$ and $\mathcal{A}$ inputs to the RO the triple $(g^{xy}, g^x, 0)$, where g^x is contained in the public key of the first receiver of the challenge ciphertext and y is the discrete logarithm of the randomness carrier Y contained in the challenge ciphertext. We call this event E. We next construct a reduction $\mathcal{B}$ which solves DSDH if E and some other independent events occur. Here, we use that there are no collisions in the mKEM ciphertexts. Otherwise, the adversary could query the decryption oracle on the first position with a colliding mKEM ciphertext and detect the change that way.

$\mathcal{B}$ is given an DSDH instance (X, Y). It runs $\mathcal{A}$, emulating G_0 or G_1 as follows. At the beginning, $\mathcal{B}$ generates N mKEM public keys $\mathsf{pk}_{i,0}$ and corresponding DH public keys $\mathsf{pk}_{i,1}$. Some of the DH keys are "known", i.e., generated as in mmKGen, and other are "unknown", i.e., their DH-related component is set to X^{x_i} for a random x_i. In particular, each public key is unknown with probability $t = 1/q_c$. This choice maximizes the probability that $\mathcal{B}$ can answer all corrupt queries *and* the event E happens for an unknown key, which will enable it to extract a DSDH solution. A complete description of $\mathcal{B}$ can be found in Fig. 15.

In order to simulate the game consistently, $\mathcal{B}$ has to choose some keys in the decapsulation as well as hash oracle. These are saved in the list DL and HL respectively. It can check the list of the respective other oracle via its strong Diffie-Hellman oracles $\mathbf{O}_x$ and $\mathbf{O}_y$, since it embeds the challenge Y in some public keys and X in the challenge. Specifically, it can use the oracles to check if a hash query corresponds to a decapsulation query and vice versa without knowing the secret keys.

There are three cases in which $\mathcal{B}$ aborts, denoted Bad_1 to Bad_3. Bad_1 and Bad_2 happen, if $\mathcal{A}$ guesses the random group element X before the second phase. Since X is random and information-theoretically hidden form $\mathcal{A}$ before it receives its challenge, with a union-bound we get that the events happen with probability at most $\frac{q_H}{p}$ and $\frac{q_{D_1}}{p}$ respectively. Bad_3 occurs if either $\mathcal{A}$ queries a corruption query on an unknown key or uses a non-challenge key at position 1 of the challenge,

Algorithm Adversary $\mathcal{B}$ on DSDH

Adversary $\mathcal{B}^{\mathbf{O}_x,\mathbf{O}_y}(\mathbb{G},p,g,X,Y)$

$\mathsf{Corr} \leftarrow \emptyset, \mathsf{HL} \leftarrow \emptyset, \mathsf{DL} \leftarrow \emptyset$
$\mathsf{pp} \leftarrow_{\$} \mathsf{mSetup}()$
for $j \in [N]$ **do**
  $(\mathsf{pk}_{j,0}, \mathsf{sk}'_{j,0}) \leftarrow_{\$} \mathsf{mKGen}(\mathsf{pp})$
  Pick $b[j] \leftarrow \{0,1\}$ with $\Pr[b[j]=1] = \frac{1}{q_C}$
  $\alpha_j \leftarrow_{\$} \mathbb{Z}_p \setminus \{0\}$
  if $b[j]=1$ **then**
    $\mathsf{pk}_{j,1} \leftarrow Y^{\alpha_j}$ // Embed the challenge
  else
    $\mathsf{pk}_{j,1} \leftarrow g^{\alpha_j}$ // Allow corruption
  $\mathsf{pk}_j \leftarrow (\mathsf{pk}_{j,0}, \mathsf{pk}_{j,1})$
$\mathsf{Phase} \leftarrow 1$
$(\vec{m}_0^*, \vec{m}_1^*, \vec{\mathsf{pk}}^*, st) \leftarrow_{\$} \mathcal{A}^{\mathsf{H},\mathsf{G},\mathsf{D}_1,\mathsf{Cor}}(\mathsf{pk}_1,\ldots,\mathsf{pk}_N)$
req $|\vec{m}_0| = |\vec{m}_1| = |\vec{\mathsf{pk}}^*| = n$
Parse $\vec{\mathsf{pk}}^*$ as $\hat{\mathsf{pk}}_{i_1},\ldots,\hat{\mathsf{pk}}_{i_l},\mathsf{pk}^*_{l+1},\ldots,\mathsf{pk}^*_n$ for $l \in [n]$
s.t. $\forall j \in [l] : \vec{m}_0[i_j] \neq \vec{m}_1[i_j] \wedge \hat{\mathsf{pk}}_{i_j} \in \vec{\mathsf{pk}}$
if $b[i_1]=0$ **then**
  $\mathsf{Bad}_3 \leftarrow True$
  abort
$c_0^* \leftarrow X$
$(i, \vec{C}) \leftarrow (1, ())$
while $i \leq n$ **do**
  $j \leftarrow \max\{j : m_i = m_j\}$
  $(C, K_{\mathsf{mkem},j}) \leftarrow \mathsf{mKEM.mEncaps}(\mathsf{pp}, \mathsf{pk}_i, \ldots, \mathsf{pk}_{j-1})$
  $\vec{C} \mathrel{+}\!\leftarrow (C,j)$
  for $\ell \in [i, j-1]$ **do**
    $K_{\mathsf{dh},\ell} \leftarrow_{\$} \mathcal{K}$
    $c_j^* \leftarrow \mathsf{E}(G(K_{\mathsf{dh},\ell}, K_{\mathsf{mkem},j}), \vec{m}_b[j])$
    if $\exists k \in [N] : \mathsf{pk}_k = \vec{\mathsf{pk}}^*[j] \wedge k < i \wedge b[k]=1$ **then**
      $\mathsf{DL}[j, X, (c_j, j)] = K_{\mathsf{dh},\ell}$
    if $j > 1 \wedge b[j]=0$ **then**
      $\mathsf{HL}[X^{\alpha_j}, \mathsf{pk}^*[j], j] \leftarrow K_{\mathsf{dh},\ell}$
  $i \leftarrow j$
$\mathsf{Phase} \leftarrow 2$
$b' \leftarrow_{\$} \mathcal{A}^{\mathsf{H},\mathsf{G},\mathsf{D}_2,\mathsf{Cor}}(c_0^*,\ldots,c_n^*,st)$
return $\perp$

Oracle $\mathsf{Cor}(j)$

$\mathsf{Corr} \mathrel{+}\!\leftarrow j$
if $b[j]=1$ **then**
  $\mathsf{Bad}_3 \leftarrow True$
  abort
return α_j

Oracle $H(Z,W,j)$

if $\exists k \in [N] : W = \mathsf{pk}_k \wedge \mathsf{pk}_k = \mathsf{pk}^*[i] \wedge b[k]=1 \wedge \mathbf{O}_y(X^{\alpha_k}, Z) = 1$ **then**
  return $Z^{\frac{1}{\alpha_k}}$
if $\mathsf{Phase}=1 \wedge \mathbf{O}_x(W,Z)=1$ **then**
  $\mathsf{Bad}_2 \leftarrow 1$
  abort
if $\mathsf{Phase}=2 \wedge j \in [n] \wedge W = \mathsf{pk}^*[j] \wedge \mathbf{O}_x(W,Z)=1 \wedge j > i$ **then**
  return K_j
if $\exists j \in [N], c \in \mathbb{G}, t \in \mathcal{K} : W = \mathsf{pk}_j \wedge \mathsf{DL}[i,(c,(*,j))] = t \wedge \mathbf{O}_y(c^{\alpha_j}, Z) = 1$ **then**
  return t
if $\mathsf{HL}[Z,W,j] = \perp$ **then**
  $\mathsf{HL}[Z,W,j] \leftarrow_{\$} \mathcal{K}$
return $\mathsf{HL}[Z,W,j]$

Oracle $D_{\mathsf{Phase}}(i,(c_{\mathsf{mkem}},c_0,(c,j)))$

req $i \in [N]$
if $\mathsf{Phase}=1 \wedge c_0 = X$ **then**
  $\mathsf{Bad}_1 \leftarrow 1$
  abort
$K_{\mathsf{mkem}} \leftarrow \mathsf{mDecaps}(\mathsf{sk}_{i,0}, c_{\mathsf{mkem}})$
if $K_{\mathsf{mkem}} = K_{\mathsf{mkem},j}$ for j s.t. i in the j-th block **then**
  return $\perp$
if $\exists Z \in \mathbb{G}, t \in \mathcal{K} : \mathsf{HL}[Z, \mathsf{pk}_j, j] = t \wedge (b[j]=0 \implies Z = \mathsf{pk}_j^{\alpha_j} \wedge b[j]=1 \implies \mathbf{O}_y(c^{\alpha_j}, Z)=1)$ **then**
  $K_{\mathsf{dh}} \leftarrow t$
  return $\mathsf{D}(G(K_{\mathsf{dh}}, K_{\mathsf{mkem}}), c)$
if $\mathsf{DL}[i,(c_0(c,j))] = \perp$ **then**
  $\mathsf{DL}[i,(c_0,(c,j))] \leftarrow_{\$} \mathcal{K}$
$K_{\mathsf{dh}} \leftarrow \mathsf{DL}[i,(c_0,j)]$
return $\mathsf{D}(G(K_{\mathsf{dh}}, K_{\mathsf{mkem}}), c)$

Fig. 15: Description of adversary $\mathcal{B}$ for Theorem 7 for the game hop from G_1 to G_2.

i.e. $\mathcal{B}$ can't embed its challenge. Since all keys look like uniformly random group elements to $\mathcal{A}$, the b_i, i.e. the bits denoting whether a key is known or unknown, are hidden from $\mathcal{A}$ as well. Therefore, the probability that Bad_3 does *not* happen can be bounded by

$$\Pr[\mathsf{Bad}_3 = \mathtt{False}] = (1 - \frac{1}{q_C})^{q_C} \cdot \frac{1}{q_C} \leq \frac{1}{q_C e^2},$$

Scheme	$\lvert pk \rvert$	$\lvert c^{\mathsf{i}} \rvert$	$\lvert c^{\mathsf{d}} \rvert$	Sum Sender	Sum Receiver
Trivial (DH + Kyber)	864	0	1536	$1536 \cdot n$	1536
Generic (DH + Kyber) (Fig. 10)	864	1312	224	$k * 1312 + n * 224$	1536
Optimized (with Kyber) (Fig. 11)	864	1312	224	$32 + k * 1280 + n * 224$	1536

Table 2: Comparison of different mmPKE constructions. "Trivial" simply uses Kyber and HashedElGamal as KEMs with a regular KEM-combiner and does a complete KEM encapsulation for each message. "Generic" denotes the construction from Fig. 10 instantiated with Kyber [11] and HashedElGamal and "Optimizted" denotes the construction from Fig. 11 also with Kyber. Sizes are in bytes, n denotes the number of receivers and k the number of message blocks.

where we use that $\ln(1+x) \geq \frac{x}{x+1}$ for all $x \geq -1$ and rewrite $(1-\frac{1}{q_C})^{q_C} = e^{\ln((1-\frac{1}{q_C})^{q_C})} = e^{q_C \cdot \ln(1-\frac{1}{q_C})} \geq e^{-1/(1-\frac{1}{q_C})} \geq e^{-2}$ for $q_C > 1$.

Apart from the three bad cases, the simulation of the game for $\mathcal{A}$ is perfect. Additionally, if $\mathcal{A}$ makes a query as in the first case of the random oracle simulation, then $\mathcal{B}$ wins its DSDH game. As the two games for $\mathcal{A}$ are identical unless it makes such a query, the difference can be bounded as claimed.

Similarly to the correctness of the construction in Fig. 10, the bound follows from the fact that the DH-based construction is perfectly correct and a decapsulation error in mKEM or a decryption error in DEM already causes a decryption error in the overall construction. A union-bound yields the final bound. □

B.7 Comparison

We compare our two constructions with the trivial construction of parallel PKE composition in Table 2. For the Kyber part of all constructions, we consider the scheme using both our compiler from Section 3.1 and the FO transform from Section 4. Therefore, public key size increases by 1 hash value (32B) and ciphertext size approximately doubles from regular Kyber.

First, note that all constructions have identical public key size and receiver bandwidth. This is due to the fact that the used mmPKE constructions save bandwidth by reusing randomness, but a receiver still gets the same ciphertext in all cases.

For overall bandwidth, both our generic and optimized constructions are more efficient than the trivial construction as long as messages are actually repeated, since then the randomness reuse of the mKEM can be leveraged. For our optimized construction, we always save bandwidth compared to both other constructions, because we always only require one randomness carrier for the DDH-like mmPKE. The generic and trivial construction coincide, if all messages are distinct.

C API considerations

Our starting point for the API of the mKEM derived from Kyber is the NIST API for KEMs in the C programming language[13]:

```
typdef unsigned char u8;
int crypto_kem_keypair(u8 *pk, u8 *sk);
int crypto_kem_enc(u8 *ct, u8 *ss, const u8 *pk);
int crypto_kem_dec(u8 *ss, const u8 *ct, const u8 *sk);
```

The most obvious changes to this API required for supporting an mKEM, are to pass multiple public keys (i.e., an array of pointers) to the encapsulation routine and to output multiple ciphertexts from encapsulation. In order to support the optimizations proposed in this paper (or also the construction from [28]), we need to additionally split ciphertexts into two components: one that is equal for all recipients and one that differs for individual recipients. One final change to the API concerns the public seed used to generate the "system parameter" matrix $\mathbf{A}$. Kyber—like most other lattice-based NIST candidates—follows the approach of NewHope to make this seed part of the public key (see [2, Sec. 3]). However, in the optimized mKEM construction we need all recipients to use the same parameter $\mathbf{A}$. One solution would be to fix a seed (and thus a parameter $\mathbf{A}$) in the implementation, but we decided to give protocols the flexibility to use a different parameter for different groups of users and refresh the seed when possible. We thus pass a pointer to this seed as additional parameter. This results in the following API for the mKEM:

```
int crypto_mkem_keypair(u8 *pk, u8 *sk, const u8 *seed);
int crypto_mkem_enc(u8 *c1, u8 **c2s, u8 *ss, const u8 *seed,
                    size_t num_keys, u8 *const* pk);
int crypto_mkem_dec(u8 *ss, const u8 *c1, const u8 *c2, const u8 *sk);
```

In addition to this monolithic approach to encapsulation, our software also provides a split API for encapsulation consisting of one function that computes the common ciphertext component `c1` (which can be called before public keys are known) and one function that on input the public keys computes the array of individual second components.

D Additional Benchmark Results

[13] See https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/example-files/api-notes.pdf

n	Kyber mKEM				Naive ($n\times$ Kyber)			
	cycles		bytes		cycles		bytes	
1	**gen**:	43572			**gen**:	33224		
	dec:	67724			**dec**:	31060		
1	**enc**:	55364	**pk**:	800	**enc**:	43548	**pk**:	800
			ct:	1537			**ct**:	768
2	**enc**:	80588	**pk**:	1600	**enc**:	87096	**pk**:	1600
			ct:	1794			**ct**:	1536
10	**enc**:	282028	**pk**:	8000	**enc**:	435480	**pk**:	8000
			ct:	3850			**ct**:	7680
100	**enc**:	2837664	**pk**:	80000	**enc**:	4354800	**pk**:	80000
			ct:	26980			**ct**:	76800
1000	**enc**:	28099652	**pk**:	800000	**enc**:	43548000	**pk**:	800000
			ct:	258280			**ct**:	768000

Table 3: Intel Haswell cycle counts and transmitted bytes for Kyber mKEM and naive $n\times$ application of Kyber at NIST security level 1 (Kyber512)

n	Kyber mKEM				Naive ($n\times$ Kyber)			
	cycles		bytes		cycles		bytes	
1	**gen**:	63344			**gen**:	52244		
	dec:	96920			**dec**:	48188		
1	**enc**:	77204	**pk**:	1184	**enc**:	64288	**pk**:	1184
			ct:	2177			**ct**:	1088
2	**enc**:	111080	**pk**:	2368	**enc**:	128576	**pk**:	2368
			ct:	2434			**ct**:	2176
10	**enc**:	384240	**pk**:	11840	**enc**:	642880	**pk**:	11840
			ct:	4490			**ct**:	10880
100	**enc**:	3832316	**pk**:	118400	**enc**:	6428800	**pk**:	118400
			ct:	27620			**ct**:	108800
1000	**enc**:	37947108	**pk**:	1184000	**enc**:	64288000	**pk**:	1184000
			ct:	258920			**ct**:	1088000

Table 4: Intel Haswell cycle counts and transmitted bytes for Kyber mKEM and naive $n\times$ application of Kyber at NIST security level 3 (Kyber768)