Post-Quantum Multi-Recipient Public Key Encryption

Joël Alwen
alwenjo@amazon.com
Amazon.com, Incorporated
Seattle, WA, USA

Dominik Hartmann
dominik.hartmann@rub.de
Ruhr-Universität Bochum
Bochum, Germany

Eike Kiltz
eike.kiltz@rub.de
Ruhr-Universität Bochum
Bochum, Germany

Marta Mularczyk
mulmarta@amazon.ch
Amazon.com, Incorporated
Seattle, WA, USA

Peter Schwabe
peter@cryptojedi.org
Max Planck Institute for Security and Privacy & Radboud University
Bochum, Germany

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 (mPKE) 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 mKEMs and (m)mPKEs. 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 presence of adaptive corruptions in the quantum random oracle model. Further, we provide the first mKEM combiner. Finally, we give 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 (which is suited for several recent mmPKE applications) and achieves hybrid PQ/classic security more directly. When encrypting n short messages 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 encapsulation and up to 79% savings in ciphertext size compared to a naive solution.

CCS CONCEPTS

• **Security and privacy** → **Public key encryption**; **Key management**.

KEYWORDS

multi recipient; public key encryption, key encapsulation mechanism, post-quantum security

ACM Reference Format:
Joël Alwen, Dominik Hartmann, Eike Kiltz, Marta Mularczyk, and Peter Schwabe. 2023. Post-Quantum Multi-Recipient Public Key Encryption. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3623185

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 [26], 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, "multi-recipient ciphertext" 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 secret key sk_i to the decryption algorithm. However, roughly speaking, m_i should stay secret given the full multi-recipient ciphertext and all other decryption keys. A special

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11.
https://doi.org/10.1145/3576915.3623185

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 [8, 9, 26, 32] 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 primitives (that is, (m)mKEM or (m)mPKE) has played a central role in reducing (at times, quite dramatically) the communication cost of some recent protocols. Notably, [25] 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 [22]. Their Kyber-based mPKE 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 defends against active adversaries. Thus, they all require strong non-malleability properties from their mKEM/mmPKE. While [22] requires full fledged CCA security, [3] gets away with a relaxed variant known as *replayable CCA* (RCCA) security [12]. 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 [25] enjoys no such security proof. To date, the only extractable primitives with provable adaptive security are the CCA secure mPKEs of [22] and mmPKE of [3]. Yet, both security proofs are in the classic setting only. There exist no extractable primitives of any type with security proof for PQ adaptive security.

Another gap in the state-of-the-art concerns hybrid security for extractable primitives. A "hybrid" security property holds if any one of multiple underlying assumptions holds. A special case of constructions enjoying hybrid security are combiners [21, 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 primitives (PQ or otherwise). In particular, KEM combiners for CCA security [10, 11, 20, 29] 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* 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 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 [25] 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 [22], our compiled ciphertext sizes have the same size but our public keys are about half as big as theirs.[1] We remark that [22] 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

[1] We note that all applications of mKEM/mmPKE mentioned above require sending fresh public keys as part of every protocol packet [22]. In fact, [3, 25] both require multiple new keys per packet.

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

From CPA to (R)CCA Security for mKEMs. Next, we consider adapting the FO transform [17, 18], 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 [25] where it is shown to be classically and quantum secure. However, the proofs from [25] do not work with adaptive corruptions and, according to [15], the proof of quantum security has a gap (carried over from the proof technique by Zhandry [37]). Partially fixing these issues, the work [22] 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 [22]. Our proof uses the framework of [15], thus avoiding the issue in the proof from [25]. We note that applying the framework of [15] 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.

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 then the combined scheme is also CCA-secure. It turns out that many natural mKEM contructions are CR. In particular, we show that any mKEM constructed using the FO of [22] with a hash function H is CR assuming only collision resistance of H. We also present a variant of standard Diffie-Hellman based mKEM which we can also show to be CR (it is easy to get a variant that is CCA-secure but not CR by removing one value from a hash input). For this we need the underlying hash function as well as the encryption function of the underlying DEM to be collision resistant. 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 classic and quantum versions of these results (note that in the quantum setting the DH-based mKEM is not CPA secure but it is CR).

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 [3, 22, 25]). 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 mKEM and DEM are both RCCA secure, then so is mmPKE.
(2) If mKEM and DEM are both CCA secure then so is mmPKE.
(3) If the DEM is RCCA secure, then the Double Strong Diffie-Hellman[2] assumption implies that mmPKE is RCCA secure.
(4) If mKEM is collision resistant and DEM is CCA secure then the DSDH assumption implies that mmPKE is CCA secure.

Implementation. Finally, we present an implementation of an mKEM based on Kyber, 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 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 [26]. 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 [8, 9, 32] 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, [32] 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 [33]. Afterwards, various mKEM constructions have been proposed based on hash proof systems [28] and pairings [36]. Finally, the first efficient post-quantum secure mKEMs have been constructed in [25]. In terms of mPKEs, the only construction with strong (stronger than IND-CPA) security was proposed recently by Hashimoto et al. [22]. Another line of works considers identity-based versions of (m)mPKE and (m)mKEMs [13, 24, 27, 31]. Finally, [6] build practical PQ secure Dual-PRFs.

2 PRELIMINARIES

In this section we recall the well-known Oneway-to-Hiding (O2H) lemma [35] as well as the Extractable Quantum Random Oracle Simulator from [15] used in post-quantum proofs (see [30] for a thorough introduction). Then we recall multi-message multi-recipient encryption. We give additional preliminaries in the full version [4].

2.1 Oneway-to-Hiding (O2H)

We recall the original version of the well-known Oneway-to-Hiding lemma [35], using the notation from [5]. Intuitively, the lemma

[2]The DSDH is the standard assumption used to prove CCA security of DH based non-interactive key exchange. [16]

allows us the bound the advantage of an adversary in detecting programming in the quantum random oracle model.

Lemma 2.1 (Original O2H [5], 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 $\{1, \ldots, q\}$, runs $\mathcal{A}$ right before its i-th query to G, measures its query input register and outputs the measurement m. Then*

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

2.2 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 [37] using so-called *compressed oracles*. [15] simplified the above formalism and defined the clean abstraction of a quantum random oracle simulator which allows for extraction queries (under specific conditions) without noticeably changing the adversary's state. We recall their construction in Definition 2.3 together with some properties in Definition 2.2

Definition 2.2. 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

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

Definition 2.3 (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 subsequent independent queries to $\mathcal{S}_f.RO$ (resp. $\mathcal{S}_f.E$) commute.
(3) Any two subsequent independent 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 states 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.

2.3 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 the full version [4].)

Setup: $\mathsf{mSetup}() \to \mathsf{pp}$ returns a fresh public parameter pp.
Key Generation: $\mathsf{mmKGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$ generates a key pair.
Encryption: $\mathsf{mmEnc}(\mathsf{pp}, \vec{\mathsf{pk}}, \vec{m}) \to C$ 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 ciphertext C. The i-th message in $\vec{m}$ is encrypted to the i-th public key in $\vec{\mathsf{pk}}$.
Extraction: The (deterministic) algorithm $\mathsf{mmExt}(\mathsf{pp}, C, i) \to c/\perp$ takes the public parameters pp, a ciphertext C and an index i. It outputs the individual ciphertext for the i-th receiver (or $\perp$).
Decryption: $\mathsf{mmDec}(\mathsf{pp}, \mathsf{sk}, c) \to m/\perp$ takes the public parameters pp, a secret key sk and an individual ciphertext c as input and returns either a decrypted message m or $\perp$.

We recall in Fig. 1 the mmIND-CCA and mmIND-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 challenge public-key vector can contain keys generated by the challenger and ones chosen by the adversary.

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. This mechanism is slightly different for CCA and RCCA, which is the only difference

Game mmIND-CCA, mmIND-RCCA

$\mathbf{Exp}^{\text{mmIND-ATK}}_{\text{mmPKE},N,\text{leak},b}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))\text{ATK} \in \{\text{CCA}, \text{RCCA}\}$

pp ← mmSetup()
for $i \in [N]$ **do** $(\text{pk}_i, \text{sk}_i) \leftarrow \text{mmKGen}(\text{pp})$
Cor ← ∅
$(\vec{\text{pk}}^*, \vec{m}_0^*, \vec{m}_1^*, st) \leftarrow \mathcal{A}_1^{\mathbf{Dec}_1, \mathbf{Cor}}(\text{pp}, (\text{pk}_i)_{i \in [N]})$
req $|\vec{m}_0^*| = |\vec{m}_1^*| = |\vec{\text{pk}}^*| \wedge \text{leak}(\vec{m}_0^*) = \text{leak}(\vec{m}_1^*)$
$C^* \leftarrow_\$ \text{mmEnc}(\text{pp}, \vec{\text{pk}}^*, \vec{m}_b)$
$b' \leftarrow \mathcal{A}_2^{\mathbf{Dec}_2^{\text{ATK}}, \mathbf{Cor}}(C^*, st)$
req $\forall j : (m_0^*[j] \neq m_1^*[j] \implies \vec{\text{pk}}^*[j] \in \{\text{pk}_i : i \in [N] \setminus \text{Cor}\})$
return $b' = b$

$\mathbf{Oracle\ Dec}_2^{\text{mmIND-RCCA}}(i, c)$

req $i \in [N]$
$m \leftarrow \text{mmDec}(\text{pp}, \text{sk}_i, c)$
if $\exists j : \vec{\text{pk}}^*[j] = \text{pk}_i \wedge m \in \{\vec{m}_0^*[j], \vec{m}_1^*[j]\}$ **then return** *'test'*
else return m

$\mathbf{Oracle\ Dec}_2^{\text{mmIND-CCA}}(i, c)$

req $i \in [N]$
req $\nexists j : \vec{\text{pk}}^*[j] = \text{pk}_i \wedge c = \text{mmExt}(\text{pp}, C, j)$
return $\text{mmDec}(\text{sk}_i, c)$

$\mathbf{Oracle\ Dec}_1(i \in [N], c)$

return $\text{mmDec}(\text{pp}, \text{sk}_i, c)$

$\mathbf{Oracle\ Cor}(i \in [N])$

$\text{Cor} \mathrel{+}\!\leftarrow i$; **return** sk_i

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

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 $\text{leak}(\vec{m})$ 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 2.4. Let mmPKE be an mmPKE scheme, let N be an integer and let $\text{Exp}^{\text{mmIND-ATK}}_{\text{mmPKE},N,\text{leak},b}(\mathcal{A})$ be defined in Fig. 1. Further, let leak be a function with a domain containing all message vectors. For $\text{ATK} \in \{\text{CCA}, \text{RCCA}\}$ we define the advantage of adversary $\mathcal{A}$ playing game mmIND-ATK with leakage leak as

$$\text{Adv}^{\text{mmIND-ATK}}_{\text{mmPKE},N,\text{leak}}(\mathcal{A}) := \Big|\Pr[\text{Exp}^{\text{mmIND-ATK}}_{\text{mmPKE},N,\text{leak},1}(\mathcal{A}) \Rightarrow 1] - \Pr[\text{Exp}^{\text{mmIND-ATK}}_{\text{mmPKE},N,\text{leak},0}(\mathcal{A}) \Rightarrow 1]\Big|.$$

2.4 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. See the full version [4] for correctness.

Setup: $\text{mSetup}() \to \text{pp}$ returns a fresh public parameter pp.

Game mIND-CCA, mIND-RCCA

Experiment $\text{Exp}^{\text{mIND-ATK}}_{\text{mKEM},N,b}((\mathcal{A}_1, \mathcal{A}_2))$**,** $\text{ATK} \in \{\text{CCA}, \text{RCCA}\}$

pp ← mSetup(); Cor ← ∅
for $i \in [N]$ **do** $(\text{pk}_i, \text{sk}_i) \leftarrow \text{mKGen}(\text{pp})$
$((i_1^*, \ldots, i_n^*), st) \leftarrow \mathcal{A}_1^{\mathbf{Cor}, \mathbf{Dec}_1}(\text{pp}, \text{pk}_1, \text{pk}_2, \ldots, \text{pk}_N)$
$(C^*, k_0^*) \leftarrow \text{mEnc}(\text{pp}, \text{pk}_{i_1^*}, \text{pk}_{i_2^*}, \ldots, \text{pk}_{i_n^*})$
$k_1^* \leftarrow_\$ \{0,1\}^\kappa$; $b' \leftarrow \mathcal{A}_2^{\mathbf{Cor}, \mathbf{Dec}_2^{\text{ATK}}}(st, C^*, k_b^*)$
req $\{i_1^*, i_2^*, \ldots, i_n^*\} \cap \text{Cor} = \emptyset$
return $b' = b$

$\mathbf{Oracle\ Cor}(i)$

req $i \in [N]$
$\text{Cor} \leftarrow \text{Cor} \cup \{i\}$
return sk_i

$\mathbf{Oracle\ Dec}_2^{\text{CCA}}(i, c)$

req $i \in [N]$
$k \leftarrow \text{mDec}(\text{pp}, c, \text{sk}_i)$
for j **s.t.** $i_j^* = i$ **do**
  req $c \neq \text{mExt}(C^*, j)$
return k

$\mathbf{Oracle\ Dec}_1(i, c)$

req $i \in [N]$
return $\text{mDec}(\text{pp}, c, \text{sk}_i)$

$\mathbf{Oracle\ Dec}_2^{\text{RCCA}}(i, c)$

req $i \in [N]$
$k \leftarrow \text{mDec}(\text{pp}, c, \text{sk}_i)$
if $k \in \{k_0^*, k_1^*\}$ **then**
  return *'test'*
else return k

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

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

Encapsulation: $\text{mEnc}(\text{pp}, \text{pk}_1, \ldots, \text{pk}_n) \to (C, k)$ 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 algorithm $\text{mExt}(\text{pp}, C, i) \to c/\perp$ takes as input a multi-recipient ciphertext C and position index i and returns a (individual) ciphertext c for the i-th recipient.

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

Security. We define the strong security notion for mKEM with corruptions as (implicitly) defined in [22], called mIND-CCA, and the weaker notion of mIND-RCCA.

Definition 2.5. Let mKEM be an mKEM scheme, let N be an integer and let $\text{Exp}^{\text{mmIND-ATK}}_{\text{mKEM},N,b}(\mathcal{A})$ be defined in Fig. 2. For $\text{ATK} \in \{\text{CCA}, \text{RCCA}\}$, we define the advantage $\text{Adv}^{\text{mIND-ATK}}_{\text{mKEM},N}(\mathcal{A})$ of an adversary $\mathcal{A}$ against the mIND-ATK security of mKEM as

$$\Big|\Pr[\text{Exp}^{\text{mIND-ATK}}_{\text{mKEM},N,1}(\mathcal{A}) \Rightarrow 1] - \Pr[\text{Exp}^{\text{mIND-ATK}}_{\text{mKEM},N,0}(\mathcal{A}) \Rightarrow 1]\Big|.$$

2.5 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 for mPKE that are geared towards enabling the mKEM construction. In particular, we use the less general syntax from [25] 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. Correctness can be found in the full version [4].

Setup: $\mathsf{mSetup}() \to \mathsf{pp}$ returns a fresh public parameter pp.

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

Encryption: Encryption consists of two algorithms: $\mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp}) \to c^{\mathsf{i}}$ outputs a *recipient-independent* ciphertext component c^{i} needed by all recipients. Second, $\mathsf{mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}, m, r) \to c^{\mathsf{d}}$ on input a public key pk, a message m and the randomness r used by $\mathsf{mEnc}^{\mathsf{i}}$, 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..

Security. We define four security notions for mPKE, both one-way and indistinguishability with and without adaptive corruptions as in [22]. The notions are formally defined in Figure 3. Roughly, for mOW-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., $\mathsf{mOW\text{-}CPA}^{\mathsf{corr}}$) is trivially implied by mIND-CPA (resp., $\mathsf{mIND\text{-}CPA}^{\mathsf{corr}}$).

Definition 2.6. Let mPKE be an mPKE scheme, N an integer, $\mathcal{A}$ an adversary and $\mathrm{Exp}^{\mathsf{mOW\text{-}ATK}}_{\mathsf{mPKE},N}(\mathcal{A})$ and $\mathrm{Exp}^{\mathsf{mIND\text{-}ATK}}_{\mathsf{mPKE},N,b}(\mathcal{A})$ be defined in Fig. 3. For $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CPA}^{\mathsf{corr}}\}$, we define the advantage of $\mathcal{A}$ against the mOW-ATK security of mPKE as

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

and the advantage of $\mathcal{A}$ against mIND-ATK security of mPKE as

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

Looking ahead, the Fujisaki-Okamoto transform (see Section 4) only require one-way security. On the other hand, our compiler from security without corruptions to security with corruptions (see Section 3.1) requires mIND-CPA security. 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 $\mathsf{mOW\text{-}CPA}^{\mathsf{corr}}$ 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

Game mOW-CPA, $\mathsf{mOW\text{-}CPA}^{\mathsf{corr}}$

Experiment $\mathrm{Exp}^{\mathsf{mOW\text{-}ATK}}_{\mathsf{mPKE},N}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CPA}^{\mathsf{corr}}\}$

$\mathsf{pp} \leftarrow \mathsf{mSetup}(); \mathsf{Cor} \leftarrow \emptyset$
for $i \in [N]$ **do** $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{mKGen}(\mathsf{pp})$
if $\mathsf{ATK} = \mathsf{CPA}^{\mathsf{corr}}$ **then** $\mathbf{O} \leftarrow \mathbf{Cor}$
else $\mathbf{O} \leftarrow \emptyset$
$((i_1^*, \ldots, i_n^*), st) \leftarrow \mathcal{A}_1^{\mathbf{O}}(\mathsf{pp}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)$
$m \leftarrow_{\$} \mathcal{M}$; $r \leftarrow_{\$} \{0,1\}^{\kappa}$; $c^{\mathsf{i}} \leftarrow \mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp}; r)$
for $j \in [n]$ **do** $c_j^{\mathsf{d}} \leftarrow \mathsf{mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}_{i_j^*}, m, r)$
$m^* \leftarrow \mathcal{A}_2^{\mathbf{O}}(st, c^{\mathsf{i}}, c_1^{\mathsf{d}}, \ldots, c_n^{\mathsf{d}})$
return $m = m^* \wedge \{i_1^*, i_2^*, \ldots, i_n^*\} \cap \mathsf{Cor} = \emptyset$

Oracle Cor(i)
req $i \in [N]$; $\mathsf{Cor} \mathrel{+}\!\leftarrow i$; **return** sk_i

Game mIND-CPA, $\mathsf{mIND\text{-}CPA}^{\mathsf{corr}}$

Experiment $\mathrm{Exp}^{\mathsf{mIND\text{-}ATK}}_{\mathsf{mPKE},N,b}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CPA}^{\mathsf{corr}}\}$

$\mathsf{pp} \leftarrow \mathsf{mSetup}(); \mathsf{Cor} \leftarrow \emptyset$
for $i \in [N]$ **do** $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{mKGen}(\mathsf{pp})$
if $\mathsf{ATK} = \mathsf{CPA}^{\mathsf{corr}}$ **then** $\mathbf{O} \leftarrow \mathbf{Cor}$
else $\mathbf{O} \leftarrow \emptyset$
$((i_1^*, \ldots, i_n^*), m_0^*, m_1^*, st) \leftarrow \mathcal{A}_1^{\mathbf{O}}(\mathsf{pp}, (\mathsf{pk}_i)_{i \in [N]})$
$r \leftarrow_{\$} \{0,1\}^{\kappa}$; $c^{\mathsf{i}} \leftarrow \mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp}; r)$
for $j \in [n]$ **do** $c_j^{\mathsf{d}} \leftarrow \mathsf{mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}_{i_j^*}, m_b^*, r)$
$b' \leftarrow \mathcal{A}_2^{\mathbf{O}}(st, c^{\mathsf{i}}, c_1^{\mathsf{d}}, \ldots, c_n^{\mathsf{d}})$
req $\{i_1^*, i_2^*, \ldots, i_n^*\} \cap \mathsf{Cor} = \emptyset$
return b'

Oracle Cor(i)
req $i \in [N]$; $\mathsf{Cor} \mathrel{+}\!\leftarrow i$; **return** sk_i

Figure 3: OW(top) and IND (bottom) mPKE security.

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 [8, 9, 25, 26, 32], 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. More surprisingly, there seems to be no way to adapt the proofs for the setting without corruptions to the setting with corruptions.[3]

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 $\mathsf{mIND\text{-}CPA}^{\mathsf{corr}}$ security which is the same as mIND-CPA except the adversary can adaptively corrupt honest keys. We note that $\mathsf{mIND\text{-}CPA}^{\mathsf{corr}}$ implies $\mathsf{mOW\text{-}CPA}^{\mathsf{corr}}$, therefore, the output of the compiler achieves the goal of this section.

[3] 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.

Algorithm Comp[mPKE, H]

mSetup()
  return mPKE.mSetup()

mKGen(pp)
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{mPKE.mKGen}(\mathsf{pp})$
  $s \leftarrow_\$ \{0,1\}^\kappa$; $\mathsf{swapPk} \leftarrow_\$ \{0,1\}$
  if $\mathsf{swapPk} = 0$ **then** $\mathsf{pk}_{\mathsf{left}} \leftarrow \mathsf{pk}$
  else $\mathsf{pk}_{\mathsf{left}} \leftarrow H(s) - \mathsf{pk}$
  return $((\mathsf{pk}_{\mathsf{left}}, s), (\mathsf{sk}, \mathsf{swapPk}))$

$\mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp}; r = (r_1, r_2))$
  $c_1^{\mathsf{i}} \leftarrow \mathsf{mPKE.mEnc}^{\mathsf{i}}(\mathsf{pp}; r_1)$
  $c_2^{\mathsf{i}} \leftarrow \mathsf{mPKE.mEnc}^{\mathsf{i}}(\mathsf{pp}; r_2)$
  return $c^{\mathsf{i}} = (c_1^{\mathsf{i}}, c_2^{\mathsf{i}})$

$\mathsf{mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk} = (\mathsf{pk}_{\mathbf{left}}, s), m, r = (r_1, r_2))$
  $\mathsf{pk}_{\mathsf{right}} \leftarrow H(s) - \mathsf{pk}_{\mathsf{left}}$
  $\mathsf{swapCtx} \leftarrow_\$ \{0,1\}$
  if $\mathsf{swapCtx} = 0$ **then** $(\mathsf{pk}_1, \mathsf{pk}_2) \leftarrow (\mathsf{pk}_{\mathsf{left}}, \mathsf{pk}_{\mathsf{right}})$
  else $(\mathsf{pk}_1, \mathsf{pk}_2) \leftarrow (\mathsf{pk}_{\mathsf{right}}, \mathsf{pk}_{\mathsf{left}})$
  $c_1^{\mathsf{d}} \leftarrow \mathsf{mPKE.mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}_1, m, r_1)$
  $c_2^{\mathsf{d}} \leftarrow \mathsf{mPKE.mEnc}^{\mathsf{d}}(\mathsf{pp}, \mathsf{pk}_2, m, r_2)$
  return $c^{\mathsf{d}} = (c_1^{\mathsf{d}}, c_2^{\mathsf{d}}, \mathsf{swapCtx})$

$\mathsf{mDec}(\mathsf{pp}, (\mathsf{sk}, \mathsf{swapPk}), ((c_{\mathbf{1}}^{\mathsf{i}}, c_{\mathbf{2}}^{\mathsf{i}}), (c_{\mathbf{1}}^{\mathsf{d}}, c_{\mathbf{2}}^{\mathsf{d}}, \mathsf{swapCtx})))$
  if $\mathsf{swapPk} \oplus \mathsf{swapCtx} = 0$ **then**
    return $\mathsf{mPKE.mDec}(\mathsf{pp}, \mathsf{sk}, (c_1^{\mathsf{i}}, c_1^{\mathsf{d}}))$
  else return $\mathsf{mPKE.mDec}(\mathsf{pp}, \mathsf{sk}, (c_2^{\mathsf{i}}, c_2^{\mathsf{d}}))$

Figure 4: The mIND-CPA$^{\mathsf{corr}}$ secure mPKE scheme outputted by our compiler. The hash function H outputs elements of the underlying mPKE key space.

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 compiler is defined in Fig. 4. At a high level, the idea is to adapt the technique from [19] 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. 4 formalizes the above intuition in the mPKE syntax with the encryption split into $\mathsf{mEnc}^{\mathsf{i}}$ and $\mathsf{mEnc}^{\mathsf{d}}$. Values computed for the first and second invocation of mPKE are marked by the subscripts first and second, respectively. The bit swapCtx sampled by $\mathsf{mEnc}^{\mathsf{d}}$ 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 [19]; we invite all readers interested in why this works to look at the proof of our compiler's security in the full version [4].

We note that the recent work by Hashimoto et al. [22] also adapts the technique of [19] 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 $(H(s) - \mathsf{pk}_{\mathsf{right}}, s)$.

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 [32] and LWE [25] 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., [34]. 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]. But, if for some instantiation of mPKE such a hash function does not exist, the fallback is to use the compiler without key compression from [22].

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 [19]. Formally, we define:

Definition 3.1. For a scheme mPKE with public-key space $\mathcal{PK}$, setup algorithm mSetup and key-generation algorithm mKGen, we define the advantage of an adversary $\mathcal{A}$ against random-key security of mPKE, $\mathsf{Adv}^{\mathsf{mRND\text{-}PK}}_{\mathsf{mPKE}}(\mathcal{A})$, as

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

Algorithm Generic FO Transform FO[mPKE, H, G_1, G_2]

mSetup()
  return mPKE.mSetup()

mKGen(pp)
  return mPKE.mKGen(pp)

mExt(C, i)
  return mPKE.mExt(C, i)

mEnc(pp, $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$)
  $m \leftarrow_\$ \mathcal{M}$; $c_0 \leftarrow$ mPKE.mEnc$^{\mathsf{i}}$(pp; $G_1(m)$)
  for $i \in [n]$ **do**
    $c_i \leftarrow$ mPKE.mEnc$^{\mathsf{d}}$(pp, pk_i, m; $G_1(m)$, $G_2(\mathsf{pk}_i, m)$)
  return $(H(m), (c_0, \ldots, c_n))$

mDec(pp, sk, c)
  $m \leftarrow$ mPKE.mDec(sk, c)
  if $m = \perp$ **then return** $\perp$
  $c_0 \leftarrow$ mPKE.mEnc$^{\mathsf{i}}$(pp; $G_1(m)$)
  $c_1 \leftarrow$ mPKE.mEnc$^{\mathsf{d}}$(pp, pk, m; $G_1(m)$, $G_2(\mathsf{pk}, m)$)
  if $c \neq (c_0, c_1)$ **then return** $\perp$
  return $H(m)$

Figure 5: FO transform for mPKE with adaptive corruption.

The following theorem formalizes security and correctness of the compiler. We prove it in the full version [4].

Theorem 3.2. *Let* mPKE = Comp[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. 4. 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*

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

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

4 THE FO TRANSFORM

The Fujisaki-Okamoto (FO) transform for regular encryption [17, 18] 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 [25] 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, [15] claim that there is a non-trivial gap in the proof. The work [22] 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 [15], we use the online-extractable simulation technique [15].

New notion of spreadness. In order to apply the framework of [15], we introduce a new notion of spreadness. It is meant for the decomposable syntax of mPKEs from [25], 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 4.1.

Definition 4.1 (γ-keyindependent spreadness). Let mPKE be an mPKE scheme and $n \in \mathbb{N}$. Let mPKE is *γ-keyindependent spread*, if

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

where C^I denotes the set of recipient-independent ciphertext parts.

Security. We recall the construction of the multi-recipient FO transform in Fig. 5. The following theorem formalizes its security with adaptive corruptions in the ROM and in the QROM.

Theorem 4.2. *Let* mPKE *be a δ-correct, 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.*

$$\mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{FO}[\mathsf{mPKE},G_1,G_2,H],N}(\mathcal{A}) \leq 4q \cdot \sqrt{\mathrm{Adv}^{\mathsf{mOW\text{-}CPA}^{\mathrm{corr}}}_{\mathsf{mPKE},N}(\mathcal{B}')} + \frac{q_D}{|\mathcal{M}|} + 48q^2\sqrt{n\delta} + 16q\sqrt{q} \cdot 2^{-\frac{\gamma}{4}} + 28e\sqrt{q^3} \cdot 2^{-\frac{\gamma}{2}} + 28eq\frac{\sqrt{q^3}}{2^{\kappa/2}}$$

with κ as the output length of the random oracles and $q = 2q_D + q_H + q_G$, 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 [22]. For the quantum case, we adapt the security proof of the standard FO transform from [15] to the mPKE FO transform. The main difference is that [15] considers PKE/KEM, while we look at mPKE/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 mOW-CPA$^{\mathrm{corr}}$ 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 (pk, 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. 6 and describe their relation in the following paragraphs.

Games G_0-G_{11}

Games $G_0 - G_2$

$pp \leftarrow_\$ \mathsf{mSetup}(); m^* \leftarrow_\$ \mathcal{M}$
for $i \in [N]$ **do** $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow_\$ \mathsf{mKGen}(pp)$
$O_i \leftarrow \{\mathbf{Cor}, \mathbf{Dec}_i, G_1, G_2, H\}$ // G_0-G_1
$O_i \leftarrow \{\mathbf{Cor}, \mathbf{Dec}_i, \widehat{G}_1, \widehat{G}_2, \widehat{H}\}$ // G_2
$(i_1^*, \ldots, i_n^*, st) \leftarrow_\$ \mathcal{A}_1^{O_1}(pp, (\mathsf{pk}_i)_{i\in[N]})$
$c^{i*} \leftarrow \mathsf{mEnc}^i(pp; G_1(m^*))$ // G_0-G_2
$\widehat{c}^{i} \leftarrow \mathsf{mEnc}^i(pp; \widehat{G}_1(m^*))$ // G_1-G_2
for $j \in [n]$ **do**
  $c^{d*}_j \leftarrow \mathsf{mEnc}^d(pp, \mathsf{pk}_{i_j^*}, m^*; G_1(m^*), G_2(\mathsf{pk}_{i_j^*}, m^*))$
  $\widehat{c}^{d*}_j \leftarrow \mathsf{mEnc}^d(pp, \mathsf{pk}_{i_j^*}, m^*; \widehat{G}_1(m^*), \widehat{G}_2(\mathsf{pk}_{i_j^*}, m^*))$ // G_1 - G_2
$k_0^* \leftarrow H(m^*), k_1^* \leftarrow_\$ \mathcal{K}$
return $b = \mathcal{A}_2^{O_2}(st, c^{i*}, c^{d*}_1, \ldots, c^{d*}_n, k_0^*)$

Games $G_3 - G_{11}$

$pp \leftarrow_\$ \mathsf{mSetup}(); m^* \leftarrow_\$ \mathcal{M}$
for $i \in [N]$ **do** $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow_\$ \mathsf{mKGen}(pp)$
$O_i \leftarrow \{\mathbf{Cor}, \mathbf{Dec}_i, \widehat{G}_1, \widehat{G}_2, \widehat{H}\}$ // G_3
$O_i \leftarrow \{\mathbf{Cor}, \mathbf{Dec}_i, \mathcal{S}_f.RO, \widehat{G}_2, \widehat{H}\}$ // G_4-G_{11}
$(i_1^*, \ldots, i_n^*, st) \leftarrow_\$ \mathcal{B}_1^{O_i}(pp, (\mathsf{pk}_i)_{i\in[N]})$
$c^{i*} \leftarrow \mathsf{mEnc}^i(pp; G_1(m^*))$
for $j \in [n]$ **do** $c^*_j \leftarrow \mathsf{mEnc}^d(pp, \mathsf{pk}_{i_j^*}, m^*; G_1(m), G_2(\mathsf{pk}_{i_j^*}, m^*))$
$k_0^* \leftarrow \widehat{H}(m^*), k_1^* \leftarrow_\$ \mathcal{K}$
$m' \leftarrow_\$ \mathcal{B}_2^{O_2}(st, c^{i*}, c_1^*, \ldots, c_n^*, k_1^*)$
for $i \in [q_D]$ **do** $m'_i \leftarrow_\$ \mathcal{S}_f.E(c^i_{(i)})$ // G_4-G_5
  req $m'_i = \bot \vee m'_i = m_i$ // G_5
return m'

Oracle $\mathbf{Dec}_i(j, (c^i, c_1))$ // $G_0 - G_3$
if $i = 2 \wedge (c^i, c^d) = (c^{i*}, c^{d*}_j)$ **then return** $\bot$
$m = \mathsf{mDec}(\mathsf{sk}_j, (c^i, c_1))$; **req** $m \neq \bot$
$r_0 \leftarrow G_1(m), r_1 \leftarrow G_2(\mathsf{pk}_j, m), k = H(m)$ // G_0
if $m = m^*$ **then** $r_0 \leftarrow G_1(m), r_1 \leftarrow G_2(\mathsf{pk}_j, m), k = H(m)$ // G_1
else $r_0 \leftarrow \widehat{G}_1(m), r_1 \leftarrow \widehat{G}_2(\mathsf{pk}_j, m), k = \widehat{H}(m)$
$r_0 \leftarrow \widehat{G}_1(m), r_1 \leftarrow \widehat{G}_2(\mathsf{pk}_j, m), k \leftarrow \widehat{H}(m)$ // G_2-G_3
req $\mathsf{mEnc}^i(pp, r_0) = c^i \wedge \mathsf{mEnc}^d(pp, \mathsf{pk}_j, m; r_0, r_1) = c^d \wedge \mathsf{mEnc}^d(pp, \mathsf{pk}_j, m; r_0, r_1) = c^d$
return k

Oracle $\mathbf{Dec}_i(j, (c^i, c_1))$ // k-th decaps // $G_4 - G_9$
if $i = 2 \wedge (c^i, c^d) = (c^{i*}, c^{d*}_j)$ **then return** $\bot$
$m_k = \mathsf{mDec}(\mathsf{sk}_j, (c^i, c_1))$
$m'_k \leftarrow \mathcal{S}_f.E(c^i)$ // G_9
req $m_k \neq \bot$
$r_0 \leftarrow \mathcal{S}_f.RO(m_k), r_1 \leftarrow \widehat{G}_2(\mathsf{pk}_j, m_k)$
req $\mathsf{mEnc}^i(pp, r_0) = c^i \wedge \mathsf{mEnc}^d(pp, \mathsf{pk}_j, m_k; r_0, r_1) = c^d$
$m'_k \leftarrow \mathcal{S}_f.E(c^i)$ // G_6-G_8
req $m'_k = \bot \vee m'_k = m_k$ // G_6-G_9

req $m' \neq \bot$ // G_7-G_9
req $c^i = \mathsf{mEnc}^i(pp; \mathcal{S}_f.RO(m'_k))$ // G_8-G_9
return $\widehat{H}(m)$

Oracle $\mathbf{Dec}_i(j, (c^i, c^d))$ // G_{10}
if $i = 2 \wedge (c^i, c^d) = (c^{i*}, c^{d*}_j)$ **then return** $\bot$
$m = \mathsf{mDec}(\mathsf{sk}_j, (c^i, c_1))$
$m' \leftarrow \mathcal{S}_f.E(c^i)$
req $m = m' \neq \bot$
$r_0 \leftarrow \mathcal{S}_f.RO(m')$
$r_1 \leftarrow \widehat{G}_2(\mathsf{pk}_j, m')$
req $c^i = \mathsf{mEnc}^i(pp; r_0)$
req $c^d = \mathsf{mEnc}^d(pp, \mathsf{pk}_j, m'; r_0, r_1)$
return $\widehat{H}(m')$

Oracle $\mathbf{Dec}_i(j, (c^i, c^d))$ // G_{11}
if $i = 2 \wedge (c^i, c^d) = (c^{i*}, c^{d*}_j)$ **then return** $\bot$
$m' \leftarrow \mathcal{S}_f.E(c^i)$; **req** $m' \neq \bot$
$r_0 \leftarrow \mathcal{S}_f.RO(m')$
$r_1 \leftarrow \widehat{G}_2(\mathsf{pk}_j, m')$
req $c^i = \mathsf{mEnc}^i(pp; r_0)$
req$c^d = \mathsf{mEnc}^d(pp, \mathsf{pk}_j, m'; r_0, r_1)$
return $\widehat{H}(m')$

Figure 6: Games G_0 to G_{11} for the proof of Theorem 4.2.

Game G_0: This game is identical to the mIND-CCA game with $b = 0$, i.e. $\mathrm{Exp}^{\text{mIND-CCA}}_{\mathsf{FO}[\mathsf{mPKE}, G_1, G_2, H], N, 0}(\mathcal{A}) = \Pr[\mathrm{G}_0{}^{\mathcal{A}} = 1]$ 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 m^*. Formally, we define *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 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 $c^* = (\mathsf{mEnc}^i(pp; r^*), (\mathsf{mEnc}^d(pp, \mathsf{pk}_{i_j^*}, m^*; r^*, r^*_{i_j^*}))_{j\in[n]})$. Let $\hat{r} = \widehat{G}_1(m^*), \hat{r}_i = \widehat{G}_2(\mathsf{pk}_i, m^*)$ for $i \in [N]$ and $\hat{c} = (\mathsf{mEnc}^i(pp; \hat{r}), (\mathsf{mEnc}^d(pp, \mathsf{pk}_{i_j^*}, m^*; \hat{r}, \hat{r}_{i_j^*}))_{j\in[n]})$. Decapsulation oracle now uses $\widehat{G}$ and $\widehat{H}$ on all queries except for parts of $\hat{c}$.

We argue that the view of the adversary doesn't change between the two games unless it queries c^{i*}, c^{d*}_j to the decapsulation oracle for some $j \in [n]$ in its first query phase, i.e. before producing the vector $(i_1^*, \ldots, i_n^*)$. In that case, since G and $\widehat{G}$ differ on m^*, the decapsulation fails, which the adversary can notice. However, in order to make such a query, $\mathcal{A}_1$ would need to guess the message

m^*, which can only occur with probability $1/|\mathcal{M}|$ as the message is only chosen afterwards by the game. This is the only way to distinguish the games. Indeed, note that for all messages except for the challenge message m^*, nothing changes as the oracles coincide. For (all parts of) c^*, the decapsulation oracle won't answer by definition in the second phase. For $\hat{c}$, the oracle will also output $\perp$ in both games since the oracle uses the unpunctured oracle for $\hat{c}$, which in turn uses the punctured randomness, so the reencryption check will fail. Therefore, since $\mathcal{A}$ makes at most q_D decapsulation queries, we have $\left|\Pr\left[{G_0}^{\mathcal{A}}=1\right]-\Pr\left[{G_1}^{\mathcal{A}}=1\right]\right| \leq q_D/|\mathcal{M}|$.

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 accessible by the adversary and k_0^* is a uniformly random key (it is still the output of then random oracle H). Therefore, we can also switch the key to k_1^*.

To bound the difference between the games, we use the original oneway-to-hiding lemma (Lemma 2.1), which yields $\left|\Pr\left[{G_1}^{\mathcal{A}}=1\right]-\Pr\left[{G_2}^{\mathcal{A}}=1\right]\right| \leq 2(q_H+q_G+2q_D)\sqrt{\Pr\left[{G_3}^{\mathcal{B}}=m^*\right]}$, 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.[4] 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 [15] in that we let $\mathcal{B}$ measure an arbitrary random oracle query including those in the decapsulation oracle. We don't need the additional 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 in $\mathrm{Exp}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{FO}[\mathsf{mPKE},G_1,G_2,H],N,1}(\mathcal{A})$, we have to revert the changes made before, resulting in the upper bound $4(q_H+q_G+2q_D)\sqrt{\Pr\left[{G_3}^{\mathcal{B}}=m^*\right]}$ on the difference between $\mathrm{Exp}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{FO}[\mathsf{mPKE},G_1,G_2,H],N,0}(\mathcal{A})$ and $\mathrm{Exp}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{FO}[\mathsf{mPKE},G_1,G_2,H],N,1}(\mathcal{A})$.

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 2.3. 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 $\Pr\left[{G_4}^{\mathcal{B}}=m^*\right]=\Pr\left[{G_3}^{\mathcal{B}}=m^*\right]$.

Game G_5**:** Next, we require that the extraction queries at the end of the game coincide with the decapsulated messages if they succeed. Specifically, we add the requirement that $m_i'=\perp \vee m_i=m_i'$. 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 [14]. 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 of $\mathcal{S}_f.RO$, where we can bound $\Gamma'(f)\leq g_{\mathsf{pp}}$. With the collision bound for $q=(q_G+q_D+1)$, we get $\left|\Pr\left[{G_4}^{\mathcal{B}}=m^*\right]-\Pr\left[{G_5}^{\mathcal{B}}=m^*\right]\right| \leq 40e^2(q_D+q_G+1)^3 g_{\mathsf{pp}} + 42e^2\frac{(q_D+q_G+1)^3}{2^\kappa}$, 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_6**:** Now, we move the extraction (and the condition on the m_i's) 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 of $\mathcal{S}_f.RO$ to bound the distinguishing advantage of $\mathcal{B}$. Property 3 requires independence of the queries. However, this is easily satisfied as we only swap extractions with queries made *after* the decapsulation where the extraction is performed. Therefore, all queries to $\mathcal{S}_f.RO$ that are relevant to the ciphertext from which we extract have to be made before that decapsulation query and we do not swap with any of these. 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 $\left|\Pr\left[{G_6}^{\mathcal{B}}=m^*\right]-\Pr\left[{G_5}^{\mathcal{B}}=m^*\right]\right| \leq 8(q_D+q_G)\sqrt{2\frac{\Gamma(f)}{2^\kappa}} = 8(q_D+q_G)\sqrt{2g_{\mathsf{pp}}}$.

Game G_7**:** 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 RO and extraction query are classical and subsequent, we can apply property 6 of $\mathcal{S}_f.RO$ q_D times and get $\left|\Pr\left[{G_7}^{\mathcal{B}}=m^*\right]-\Pr\left[{G_6}^{\mathcal{B}}=m^*\right]\right| \leq 2q_D\cdot\frac{1}{2^\kappa}$.

Game G_8**:** In this game, we now require that the extracted message yields the same recipient-independent ciphertext as the decrypted message. Since we already require that the extracted messages is not $\perp$ and equal to the decrypted message, this holds trivially, so $\left|\Pr\left[{G_8}^{\mathcal{B}}=m^*\right]=\Pr\left[{G_7}^{\mathcal{B}}=m^*\right]\right|$

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 $\left|\Pr\left[{G_9}^{\mathcal{B}}=m^*\right]-\Pr\left[{G_8}^{\mathcal{B}}=m^*\right]\right| \leq 8q_D\cdot\sqrt{2g_{\mathsf{pp}}}$.

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'. $\Pr\left[{G_{10}}^{\mathcal{B}}=m^*\right]=\Pr\left[{G_9}^{\mathcal{B}}=m^*\right]$.

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, i.e., an adversary can distinguish the two games by finding a message causing a decryption error.

Let R be the relation $\{(m,(r_0=\mathcal{S}_f.RO(m),r_1=\widehat{G}_2(m,\mathsf{pk}_i)) \mid \exists i\in[n]: c^{\mathsf{i}}=\mathsf{mEnc}^{\mathsf{i}}(\mathsf{pp};r_0)\wedge c^{\mathsf{d}}=\mathsf{mEnc}^{\mathsf{d}}(\mathsf{pp},\mathsf{pk}_i,m,r_0,r_1)\wedge m\neq \mathsf{mDec}(\mathsf{sk}_i,(c^{\mathsf{i}},c^{\mathsf{d}})\}$, i.e. all messages and their randomness for a given receiver which induce a decryption error when used for encryption. 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 $\frac{\Gamma_R}{2^\kappa}\leq$

[4] 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.

$n \cdot \max_{i\in[n]}(\delta_{\mathsf{sk}_i})$. Using property 7 of $\mathcal{S}_f.RO$ for R we get the bound $\left|\Pr\left[\mathrm{G}_{11}{}^{\mathcal{B}} = m^*\right] - \Pr\left[\mathrm{G}_{10}{}^{\mathcal{B}} = m^*\right]\right| \leq 128(q_D+q_G)^2 n \cdot \max_{i\in[n]}(\delta_{\mathsf{sk}_i})$.

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 mOW-CPA$^{\mathsf{corr}}$ security of mPKE to bound $\mathcal{B}$'s advantage in G_{11}. Concretely, an adversary $\mathcal{B}'$ against the mIND-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 mOW-CPA$^{\mathsf{corr}}$ game if and only if $\mathcal{B}$ finds the message m^* for which the oracle was punctured, therefore $\Pr[\mathrm{G}_{11} \Rightarrow m^*] \leq \mathrm{Adv}^{\text{mOW-CPA}^{\mathsf{corr}}}_{\mathsf{mPKE},N}(\mathcal{B}')$. Combining all probabilities and setting $\epsilon_1 = 8(q_D + q_G)\sqrt{2g_{\mathsf{pp}}}$, $\epsilon_2 = \frac{2q_D}{2^\kappa}$, $\epsilon_3 = 42e^2(q_D + q_G + 1)^3\sqrt{g_{\mathsf{pp}}} + 42e^2\frac{(q_D+q_G+1)^3}{2^\kappa}$, $\epsilon_4 = 8q_D \cdot \sqrt{2g_{\mathsf{pp}}}$ and $\epsilon_5 = 128(q_D + q_G)^2 n \cdot \max_{i\in[n]}(\delta_{\mathsf{sk}_i})$, we get $\mathrm{Adv}^{\text{mIND-CCA}}_{\mathsf{FO}[\mathsf{mPKE},G_1,G_2,H],N}(\mathcal{A}) \leq 4(q_H+q_G+2q_D)\sqrt{\mathrm{Adv}^{\text{mOW-CPA}^{\mathsf{corr}}}_{\mathsf{mPKE},N}(\mathcal{B}') + \epsilon_1 + \ldots + \epsilon_5} + \frac{q_D}{|M|}$. With $q = (q_H + q_G + 2q_D)$, using that square root is convex and taking the expected values, we get the bound in the theorem. □

5 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 [20] 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 the entire content of 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 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.

Algorithm Comb[mKEM1, mKEM2, PRF]

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

mKGen((pp1, pp2))
  (pk1, sk1) ← mKEM1.mKGen(pp1)
  (pk2, sk2) ← mKEM2.mKGen(pp2)
  return ((pk1, pk2), (sk1, sk2))

mExt((pp1, pp2), (C1, C2), i)
  return (mKEM1.mExt(pp1, C1, i), mKEM2.mExt(pp2, C2, i))

mEnc((pp1, pp2), (pk1$_1$, pk2$_1$), ..., (pk1$_n$, pk2$_n$))
  $(C1, k1) \leftarrow$ mKEM1.mEnc(pp1, (pk1$_i$)$_{i\in[n]}$)
  $(C2, k2) \leftarrow$ mKEM2.mEnc(pp2, (pk2$_i$)$_{i\in[n]}$)
  return $(C = (C1, C2), k = \mathsf{PRF}(k1, k2))$

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

Figure 7: 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.**

Note that even if there is a "header" h included in all individual ciphertexts (e.g. the c_0 in [25] or the commitment tag T in [22]), an mKEM combiner which computes the encpsulated key as $H(k_1, k_2, h_1, h_2)$ does *not* work. The reason is that the simple attack above still works – the adversary wins by computing an individual ciphertext $((h_1, c_1), (h_2, c_2))$ (where c_1, c_2 are the individual parts) from the challenge multi-recipient ciphertext C and sending $((h_1, c_1), (h_2, c_2'))$ to the decryption oracle.

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. 7.

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 [12] 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 prove in the full version [4]. that the combined scheme is IND-RCCA secure if at least one of the input mKEMs is IND-RCCA secure.

Theorem 5.1. *Define* $\mathsf{mKEM} = \mathsf{Comb}[\mathsf{mKEM1}, \mathsf{mKEM2}, \mathsf{PRF}]$, *where* mKEM1 *and* mKEM2 *are some mKEM schemes and* PRF *is a PRF, be defined as in Fig. 7. Let* dual(PRF) *denote the PRF obtained by swapping the input and key of* PRF, *i.e.,* $\mathrm{swp}(\mathsf{PRF})(k,x) = \mathsf{PRF}(x,k)$. *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.*

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

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

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

5.2 Second Statement: Stronger Assumptions

The second idea is to prove that the combined scheme is IND-CCA secure if one of the input mKEMs is IND-CCA secure and the other insecure mKEM has a 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.

Definition 5.2. The advantage $\mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM}}(\mathcal{A})$ of an adversary $\mathcal{A}$ against *collision resistance* of $\mathsf{mKEM} = (\mathsf{mSetup}, \mathsf{mKGen}, \mathsf{mEnc}, \mathsf{mExt}, \mathsf{mDec})$ is defined as

$$\Pr\left[\begin{array}{c} c \neq c' \wedge \mathsf{mDec}(\mathsf{pp},\mathsf{sk},c) = \\ \mathsf{mDec}(\mathsf{pp},\mathsf{sk},c') \neq \perp \end{array} \middle| \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{mSetup}(), \\ (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{mKGen}(\mathsf{pp}), \\ (c,c') \leftarrow \mathcal{A}(\mathsf{pp},\mathsf{pk},\mathsf{sk}) \end{array}\right].$$

In the full version [4] we prove the following.

Theorem 5.3. *Define* $\mathsf{mKEM} = \mathsf{Comb}[\mathsf{mKEM1}, \mathsf{mKEM2}, \mathsf{PRF}]$, *where* mKEM1 *and* mKEM2 *are mKEM schemes and* PRF *is a PRF, be defined as in Fig. 7. Let* dual(PRF) *denote the PRF obtained by swapping the input and key of* PRF, *i.e.,* $\mathrm{swp}(\mathsf{PRF})(k,x) = \mathsf{PRF}(x,k)$. *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* $\mathcal{B}_3'$ *s.t.*

$$\begin{aligned}\mathrm{Adv}^{\text{mIND-CCA}}_{\mathsf{mKEM},N}(\mathcal{A}) &\le 2 \cdot \mathrm{Adv}^{\text{mIND-CCA}}_{\mathsf{mKEM1},N}(\mathcal{B}_1) + \mathrm{Adv}^{\mathsf{PRF}}_{\mathsf{PRF}}(\mathcal{B}_2) \\ &\quad + 2N \cdot \mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM2}}(\mathcal{B}_3) \text{ and} \\ \mathrm{Adv}^{\text{mIND-CCA}}_{\mathsf{mKEM},N}(\mathcal{A}) &\le 2 \cdot \mathrm{Adv}^{\text{mIND-CCA}}_{\mathsf{mKEM2},N}(\mathcal{B}_1') + \mathrm{Adv}^{\mathsf{PRF}}_{\mathrm{swp}(\mathsf{PRF})}(\mathcal{B}_2') \\ &\quad + 2N \cdot \mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM1}}(\mathcal{B}_3').\end{aligned}$$

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

Algorithm DH-mKEM$[G, g, p, H, \mathsf{DEM}]$

$\underline{\mathsf{mSetup}()}$
return $\perp$

$\underline{\mathsf{mKGen}()}$
$x \leftarrow_\$ \mathbb{Z}_p$
return (g^x, x)

$\underline{\mathsf{mExt}((Y, c_1, \ldots, c_n), i)}$
req $i \in [n]$
return (Y, c_i)

$\underline{\mathsf{mEnc}(X_1, \ldots, X_n)}$
$m \leftarrow_\$ \{0,1\}^\kappa$; $y \leftarrow_\$ \mathbb{Z}_p$
for $i \in [n]$ **do**
  $k_i^{\mathrm{dem}} \leftarrow H(\text{'dh-key'}, (X_i)^y, g^y, i)$
  $ctx_i \leftarrow \mathsf{DEM.E}(k_i^{\mathrm{dem}}, m)$
$K \leftarrow H(\text{'output-key'}, m, g^y)$
return $(C = (g^y, ctx_1, \ldots, ctx_n), K)$

$\underline{\mathsf{mDec}(c = (Y, ctx), x)}$
$k^{\mathrm{dem}} \leftarrow H(\text{'dh-key'}, (Y)^x, Y)$
$m \leftarrow \mathsf{DEM.D}(k^{\mathrm{dem}}, ctx)$
req $m \neq \perp$
return $H(\text{'output-key'}, m, Y)$

Figure 8: Collision-resistant mKEM based on DH.

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 [25] are collision-resistant. The formal claim follows with the proof in the full version [4].

Theorem 5.4. *Let* $\mathsf{mKEM} = \mathsf{FO}[\mathsf{mPKE}, H, G_1, G_2]$ *for an mPKE scheme* mPKE *and hash functions* H, G_1 *and* G_2. *For any (classical or quantum) adversary* $\mathcal{A}$, *there exists a (classical or quantum) adversary* $\mathcal{B}$ *s.t* $\mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM}}(\mathcal{A}) \le \mathrm{Adv}^{\mathsf{CR}}_{H}(\mathcal{B})$.

From Diffie-Hellman. Further, we consider the mKEM obtained by encrypting the same random key to all recipients using the Diffie-Hellman based mPKE [26, 32]. 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.[5] The pseudocode of the construction is in Fig. 8.

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 resistant. Formally,

Definition 5.5. The advantage of an adversary $\mathcal{A}$ against *collision resistance* of $\mathsf{DEM} = (\mathsf{E}, \mathsf{D})$ is defined as

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

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

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

Algorithm BB-mmPKE[mKEM, DEM]

mmSetup()
  return mKEM.mSetup()

mmKGen(pp)
  return mKEM.mKGen(pp)

mDec(pp, (c, ctx), sk)
  $k \leftarrow$ mKEM.mDec(pp, c, sk)
  req $k \neq \perp$
  return DEM.D(k, ctx)

mExt(pp, $\vec{ctx}, i$)
  for $(C, ctx, j) \in \vec{ctx}$ **do**
    if $i < j$ **then**
      $c \leftarrow$ mKEM.mExt(pp, $C, j - i$)
      return (c, ctx)
  return $\perp$

mmEnc(pp, $\mathsf{pk}_1, \ldots, \mathsf{pk}_n, m_1, \ldots, m_n$)
  $i \leftarrow 1$; $\vec{ctx} \leftarrow ()$
  while $i \leq n$ **do**
    $j \leftarrow i$
    while $j \leq n \wedge m_i = m_j$ **do** j++
    $(C, k) \leftarrow$ mKEM.mEnc(pp, $(\mathsf{pk}_i, \ldots, \mathsf{pk}_{j-1})$)
    $\vec{ctx} \mathrel{+}\!\leftarrow (C,$ DEM.E$(k, m_i), j)$
    $i \leftarrow j$
  return $\vec{ctx}$

Figure 9: The construction of mmPKE **from** mKEM **and** DEM**.**

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 the full version [4] 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 the full version [4].

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 Generic Construction from mKEM

The proof of the following theorem is in the full version [4].

Theorem 6.1. *Let* mKEM *and* DEM *be an mKEM and a DEM schemes, let* mmPKE = BB-mmPKE[mKEM, DEM] *be defined as in Fig. 9, and 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 $\mathcal{A}$,*

Algorithm Opt-mmPKE[mKEM, DEM, G, g, p, H]

mmSetup()
  return mKEM.mSetup()

mmKGen(pp)
  $x \leftarrow_\$ \mathbb{Z}_p$
  (pk, sk) $\leftarrow$ mKEM.mKGen(pp)
  return $((g^x, \mathsf{pk}), (x, \mathsf{sk}))$

mmExt(pp, $(\vec{C}, ctx_1, \ldots, ctx_n, Y), i$)
  for $(C, j) \in \vec{C}$ **do**
    if $i < j$ **then**
      $c \leftarrow$ mKEM.mExt(pp, $C, j - i$)
      req $c \neq \perp$; **return** (c, ctx_i, Y, i)
  return $\perp$

mmEnc(pp, $(X_1, \mathsf{pk}_1), \ldots, (X_n, \mathsf{pk}_n), m_1, \ldots, m_n$)
  $y \leftarrow_\$ \mathbb{Z}_p$; $(i, \vec{C}) \leftarrow (1, ())$
  while $i \leq n$ **do**
    $j \leftarrow \max\{j : m_i = m_j\}$
    $(C, k_{\mathsf{mkem}}) \leftarrow$ mKEM.mEnc(pp, $\mathsf{pk}_i, \ldots, \mathsf{pk}_{j-1}$)
    $\vec{C} \mathrel{+}\!\leftarrow (C, j)$
    for $j' \in [i, j-1]$ **do**
      $k_{\mathsf{dh},j'} \leftarrow H((X_{j'})^y, X_{j'}, j')$
      $ctx_{j'} \leftarrow$ DEM.E$(G(k_{\mathsf{dh},j'}, k_{\mathsf{mkem}}), m_i)$
    $i \leftarrow j$
  return $(\vec{C}, ctx_1, \ldots, ctx_n, g^y)$

mmDec(pp, $(c, ctx, Y, j), (x, \mathsf{sk})$)
  $k_{\mathsf{mkem}} \leftarrow$ mKEM.mDec(pp, c, sk)
  req $k_{\mathsf{mkem}} \neq \perp$; **return** DEM.D$(G(k_{\mathsf{mkem}}, H(Y^x, g^x, j)), ctx)$

Figure 10: The optimized construction of mmPKE**.** H **and** G **are hash functions modeled as random oracles.**

there exist (classical or quantum) adversaries $\mathcal{B}_1$ - $\mathcal{B}_4$ s.t.

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

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

where n is (an upper bound on) the number of recipients of the challenge vector. 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 [32] together with a post-quantum secure mKEM. The resulting mmPKE is described in Fig. 10.

In our construction in Fig. 10 the DEM key k_j is computed as $H((X_j)^y, X_j, j)$. Including X_j enables a tighter reduction to DSDH. 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.

In the full version [4] we prove the following theorem.

Theorem 6.2. *Let* mKEM, DEM *and H be an mKEM, a DEM and a hash function. Let* $\mathbb{G}$ *be a group of order p, generated by g. Further, let* mmPKE = Opt-mmPKE[mKEM, DEM, $\mathbb{G}, g, p, H$] *be defined as in Fig. 10. 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, for any (classical or quantum) adversary* $\mathcal{A}$*, there is (classical or quantum) adversaries* $\mathcal{B}_1$ *to* $\mathcal{B}_5$ *s.t.*

$$\mathrm{Adv}^{\mathsf{mmIND\text{-}CCA}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A}) \le 2n \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}CCA}}_{\mathsf{mKEM},N}(\mathcal{B}_1) + \mathsf{D} + 2 \cdot \frac{q_h^2}{2^\kappa},$$

$$\mathrm{Adv}^{\mathsf{mmIND\text{-}RCCA}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A}) \le 2n \cdot \mathrm{Adv}^{\mathsf{mIND\text{-}RCCA}}_{\mathsf{mKEM},N}(\mathcal{B}_2) + \mathsf{D},$$

$$\mathrm{Adv}^{\mathsf{mmIND\text{-}CCA}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A}) \le 2n \cdot \mathsf{EG} + \mathsf{D} + 2n \cdot \mathrm{Adv}^{\mathsf{CR}}_{\mathsf{mKEM}}(\mathcal{B}_3) \text{ and}$$

$$\mathrm{Adv}^{\mathsf{mmIND\text{-}RCCA}}_{\mathsf{mmPKE},N,\mathsf{leak}}(\mathcal{A}) \le 2n \cdot \mathsf{EG} + \mathsf{D},$$

where $\mathsf{EG} = \left(e^2 q_c \cdot \mathrm{Adv}^{\mathsf{DSDH}}_{G,g,p}(\mathcal{B}_4) + \frac{q_{d_1}+2q_h}{p}\right)$*, the hash functions H and G are modeled as random oracles,* $\mathsf{D} = n \cdot \mathrm{Adv}^{\mathsf{IND\text{-}RCCA}}_{\mathsf{DEM}}(\mathcal{B}_5)$*, 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* $\delta(n) \le n(\delta_1(n) + \delta_2(n))$.

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[6].

Benchmarks. We benchmark our Kyber-based mKEM implementation on a single core of 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 -maes -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 1000 experiments for each size of the set of recipients. We compare the mKEM cycle counts to a naive solution that encapsulates to each user individually with 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. Results in terms of sizes (at NIST security level 5) are presented in Table 1; results in terms of clock cycles of an optimized implementation in Table 2. Results for lower security levels and (for completeness) of the reference implementation are listed in the full version [4].

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

n		Kyber mKEM	$n\times$ Kyber
1	**pk**:	1568	1568
	ct:	3137	1568
2	**pk**:	3136	3136
	ct:	3458	3136
10	**pk**:	15680	15680
	ct:	6026	15680
100	**pk**:	156800	156800
	ct:	34916	156800
1000	**pk**:	1568000	1568000
	ct:	323816	1568000

Table 1: Sizes in bytes for Kyber mKEM **and naive** $n\times$ **application of Kyber at NIST security level 5 (Kyber1024)**

n		Kyber mKEM	$n\times$ Kyber
	gen:	76988	63040
	dec:	135896	66176
1	**enc**:	104260	81560
2	**enc**:	147316	163120
10	**enc**:	506716	815600
100	**enc**:	4931444	8156000
1000	**enc**:	48756832	81560000

Table 2: Intel Haswell cycle counts for AVX2-optimized implementations of Kyber1024 mKEM **and naive** $n\times$ **application of Kyber1024**

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.

ACKNOWLEDGMENTS

Dominik Hartmann and Eike Kiltz were supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CASA - 390781972, and by the European Union (ERC AdG REWORC - 101054911). Peter

Schwabe was supported by European Research Council through Starting Grant No. 805031 (EPOQUE).

REFERENCES

[1] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. 2001. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In *CT-RSA 2001 (LNCS, Vol. 2020)*, David Naccache (Ed.). Springer, Heidelberg, 143–158. https://doi.org/10.1007/3-540-45353-9_12

[2] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. 2016. Post-quantum Key Exchange - A New Hope. In *USENIX Security 2016*, Thorsten Holz and Stefan Savage (Eds.). USENIX Association, 327–343.

[3] Joël Alwen, Dominik Hartmann, Eike Kiltz, and Marta Mularczyk. 2022. Server-Aided Continuous Group Key Agreement. In *ACM CCS 2022*, Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (Eds.). ACM Press, 69–82. https://doi.org/10.1145/3548606.3560632

[4] Joël Alwen, Dominik Hartmann, Eike Kiltz, Marta Mularczyk, and Peter Schwabe. 2022. Post-Quantum Multi-Recipient Public Key Encryption. Cryptology ePrint Archive, Paper 2022/1046. https://eprint.iacr.org/2022/1046 https://eprint.iacr.org/2022/1046.

[5] Andris Ambainis, Mike Hamburg, and Dominique Unruh. 2019. Quantum Security Proofs Using Semi-classical Oracles. In *CRYPTO 2019, Part II (LNCS, Vol. 11693)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Heidelberg, 269–295. https://doi.org/10.1007/978-3-030-26951-7_10

[6] Nimrod Aviram, Benjamin Dowling, Ilan Komargodski, Kenneth G. Paterson, Eyal Ronen, and Eylon Yogev. 2022. Practical (Post-Quantum) Key Combiners from One-Wayness and Applications to TLS. Cryptology ePrint Archive, Report 2022/065. https://eprint.iacr.org/2022/065.

[7] Mihir Bellare. 2006. New Proofs for NMAC and HMAC: Security without Collision-Resistance. In *CRYPTO 2006 (LNCS, Vol. 4117)*, Cynthia Dwork (Ed.). Springer, Heidelberg, 602–619. https://doi.org/10.1007/11818175_36

[8] Mihir Bellare, Alexandra Boldyreva, Kaoru Kurosawa, and Jessica Staddon. 2007. Multirecipient Encryption Schemes: How to Save on Bandwidth and Computation Without Sacrificing Security. *IEEE Transactions on Information Theory* 53, 11 (2007), 3927–3943. https://doi.org/10.1109/TIT.2007.907471

[9] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. 2003. Randomness Re-use in Multi-recipient Encryption Schemeas. In *PKC 2003 (LNCS, Vol. 2567)*, Yvo Desmedt (Ed.). Springer, Heidelberg, 85–99. https://doi.org/10.1007/3-540-36288-6_7

[10] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. 2019. Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange. In *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, Jintai Ding and Rainer Steinwandt (Eds.). Springer, Heidelberg, 206–226. https://doi.org/10.1007/978-3-030-25510-7_12

[11] Matthew Campagna and Adam Petcher. 2020. Security of Hybrid Key Encapsulation. Cryptology ePrint Archive, Report 2020/1364. https://eprint.iacr.org/2020/1364.

[12] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. 2003. Relaxing Chosen-Ciphertext Security. In *CRYPTO 2003 (LNCS, Vol. 2729)*, Dan Boneh (Ed.). Springer, Heidelberg, 565–582. https://doi.org/10.1007/978-3-540-45146-4_33

[13] Sanjit Chatterjee and Palash Sarkar. 2006. Multi-receiver Identity-Based Key Encapsulation with Shortened Ciphertext. In *INDOCRYPT 2006 (LNCS, Vol. 4329)*, Rana Barua and Tanja Lange (Eds.). Springer, Heidelberg, 394–408.

[14] Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. 2021. On the Compressed-Oracle Technique, and Post-Quantum Security of Proofs of Sequential Work. In *EUROCRYPT 2021, Part II (LNCS, Vol. 12697)*, Anne Canteaut and François-Xavier Standaert (Eds.). Springer, Heidelberg, 598–629. https://doi.org/10.1007/978-3-030-77886-6_21

[15] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. 2022. Online-Extractability in the Quantum Random-Oracle Model. In *EUROCRYPT 2022, Part III (LNCS, Vol. 13277)*, Orr Dunkelman and Stefan Dziembowski (Eds.). Springer, Heidelberg, 677–706. https://doi.org/10.1007/978-3-031-07082-2_24

[16] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. 2013. Non-Interactive Key Exchange. In *PKC 2013 (LNCS, Vol. 7778)*, Kaoru Kurosawa and Goichiro Hanaoka (Eds.). Springer, Heidelberg, 254–271. https://doi.org/10.1007/978-3-642-36362-7_17

[17] Eiichiro Fujisaki and Tatsuaki Okamoto. 1999. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *CRYPTO'99 (LNCS, Vol. 1666)*, Michael J. Wiener (Ed.). Springer, Heidelberg, 537–554. https://doi.org/10.1007/3-540-48405-1_34

[18] Eiichiro Fujisaki and Tatsuaki Okamoto. 2013. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *Journal of Cryptology* 26, 1 (Jan. 2013), 80–101. https://doi.org/10.1007/s00145-011-9114-1

[19] Craig Gentry and Brent Waters. 2009. Adaptive Security in Broadcast Encryption Systems (with Short Ciphertexts). In *EUROCRYPT 2009 (LNCS, Vol. 5479)*, Antoine Joux (Ed.). Springer, Heidelberg, 171–188. https://doi.org/10.1007/978-3-642-01001-9_10

[20] Federico Giacon, Felix Heuer, and Bertram Poettering. 2018. KEM Combiners. In *PKC 2018, Part I (LNCS, Vol. 10769)*, Michel Abdalla and Ricardo Dahab (Eds.). Springer, Heidelberg, 190–218. https://doi.org/10.1007/978-3-319-76578-5_7

[21] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. 2005. On Robust Combiners for Oblivious Transfer and Other Primitives. In *EUROCRYPT 2005 (LNCS, Vol. 3494)*, Ronald Cramer (Ed.). Springer, Heidelberg, 96–113. https://doi.org/10.1007/11426639_6

[22] Keitaro Hashimoto, Shuichi Katsumata, Eamonn Postlethwaite, Thomas Prest, and Bas Westerbaan. 2021. A Concrete Treatment of Efficient Continuous Group Key Agreement via Multi-Recipient PKEs. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 1441–1462.

[23] Amir Herzberg. 2005. On Tolerant Cryptographic Constructions. In *CT-RSA 2005 (LNCS, Vol. 3376)*, Alfred Menezes (Ed.). Springer, Heidelberg, 172–190. https://doi.org/10.1007/978-3-540-30574-3_13

[24] Harunaga Hiwatari, Keisuke Tanaka, Tomoyuki Asano, and Koichi Sakumoto. 2009. Multi-recipient Public-Key Encryption from Simulators in Security Proofs. In *ACISP 09 (LNCS, Vol. 5594)*, Colin Boyd and Juan Manuel González Nieto (Eds.). Springer, Heidelberg, 293–308.

[25] Shuichi Katsumata, Kris Kwiatkowski, Federico Pintore, and Thomas Prest. 2020. Scalable Ciphertext Compression Techniques for Post-quantum KEMs and Their Applications. In *ASIACRYPT 2020, Part I (LNCS, Vol. 12491)*, Shiho Moriai and Huaxiong Wang (Eds.). Springer, Heidelberg, 289–320. https://doi.org/10.1007/978-3-030-64837-4_10

[26] Kaoru Kurosawa. 2002. Multi-recipient Public-Key Encryption with Shortened Ciphertext. In *PKC 2002 (LNCS, Vol. 2274)*, David Naccache and Pascal Paillier (Eds.). Springer, Heidelberg, 48–63. https://doi.org/10.1007/3-540-45664-3_4

[27] Jing Li, Xiangyan Tang, Zhijun Wei, Yu Wang, Wenbin Chen, and Tan yu an. 2021. Identity-based Multi-Recipient Public Key Encryption Scheme and Its Application in IoT. *Mobile Networks and Applications* 26 (08 2021). https://doi.org/10.1007/s11036-019-01490-6

[28] Takahiro Matsuda and Goichiro Hanaoka. 2013. Key Encapsulation Mechanisms from Extractable Hash Proof Systems, Revisited. In *PKC 2013 (LNCS, Vol. 7778)*, Kaoru Kurosawa and Goichiro Hanaoka (Eds.). Springer, Heidelberg, 332–351. https://doi.org/10.1007/978-3-642-36362-7_21

[29] Takahiro Matsuda and Jacob C. N. Schuldt. 2018. A New Key Encapsulation Combiner. In *2018 International Symposium on Information Theory and Its Applications (ISITA)*. 698–702. https://doi.org/10.23919/ISITA.2018.8664317

[30] Michael A Nielsen and Isaac Chuang. 2002. Quantum computation and quantum information.

[31] Jong Hwan Park, Ki Tak Kim, and Dong Hoon Lee. 2008. Cryptanalysis and improvement of a multi-receiver identity-based key encapsulation at INDOCRYPT 06. In *ASIACCS 08*, Masayuki Abe and Virgil Gligor (Eds.). ACM Press, 373–380.

[32] Alexandre Pinto, Bertram Poettering, and Jacob C. N. Schuldt. 2014. Multi-recipient encryption, revisited. In *ASIACCS 14*, Shiho Moriai, Trent Jaeger, and Kouichi Sakurai (Eds.). ACM Press, 229–238.

[33] Nigel P. Smart. 2005. Efficient Key Encapsulation to Multiple Parties. In *SCN 04 (LNCS, Vol. 3352)*, Carlo Blundo and Stelvio Cimato (Eds.). Springer, Heidelberg, 208–219. https://doi.org/10.1007/978-3-540-30598-9_15

[34] Nick Sullivan and Christopher Wood. 2021. Hashing to Elliptic Curves (version 13). IETF Internet Draft. https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-hash-to-curve.

[35] Dominique Unruh. 2014. Revocable Quantum Timed-Release Encryption. In *EUROCRYPT 2014 (LNCS, Vol. 8441)*, Phong Q. Nguyen and Elisabeth Oswald (Eds.). Springer, Heidelberg, 129–146. https://doi.org/10.1007/978-3-642-55220-5_8

[36] Zheng Yang. 2015. On Constructing Practical Multi-Recipient Key-Encapsulation with Short Ciphertext and Public Key. 8, 18 (2015). https://doi.org/10.1002/sec.1334

[37] Mark Zhandry. 2019. How to Record Quantum Queries, and Applications to Quantum Indifferentiability. In *CRYPTO 2019, Part II (LNCS, Vol. 11693)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Heidelberg, 239–268. https://doi.org/10.1007/978-3-030-26951-7_9