Multi-Instance Secure Public-Key Encryption

Abstract. Mass surveillance targets many users at the same time with the goal of learning as much as possible. Intuitively, breaking many users' cryptography simultaneously should be at least as hard as that of only breaking a single one, but ideally security degradation is gradual: an adversary ought to work harder to break more. Bellare, Ristenpart and Tessaro (Crypto'12) introduced the notion of multi-instance security to capture the related concept for password hashing with salts. Auerbach, Giacon and Kiltz (Eurocrypt'20) motivated the study of public key encryption (PKE) in the multi-instance setting, yet their technical results are exclusively stated in terms of key encapsulation mechanisms (KEMs), leaving a considerable gap.
We investigate the multi-instance security of public key encryption. Our contributions are twofold. Firstly, we define and compare possible security notions for multi-instance PKE, where we include PKE schemes whose correctness is not perfect. Secondly, we observe that, in general, a hybrid encryption scheme of a multi-instance secure KEM and an arbitrary data encapsulation mechanism (DEM) is unlikely to inherit the KEM's multi-instance security. Yet, we show how with a suitable information-theoretic DEM, and a computationally secure key derivation function if need be, inheritance is possible. As far as we are aware, ours is the first inheritance result in the challenging multi-bit scenario.

Keywords: Multi-Instance Security · Hybrid Encryption · Property Inheritance · Mass Surveillance

1 Introduction

Security of cryptographic schemes is increasingly studied concretely. The question changes from whether a scheme is secure or not, to how secure it is. The change in emphasis also results in increased importance in more realistic security notions that model a world where an adversary might have many potential targets. If an adversary simply tries to learn something about one of its κ targets, then intuitively the more targets there are, the easier the adversary's job becomes. Indeed, using simple hybrid arguments results in a security degradation that is linear in κ. But what happens if the adversary is greedy and wants to learn more, maybe even targets everyone? On the one hand, one could argue that if breaking one instance is hard, then so is breaking many. Yet, on the other hand, one would hope that breaking multiple instances, say n, is strictly harder than breaking just a single one.

This second perspective made Bellare, Ristenpart and Tessaro [11], henceforth BRT, realize that new security notions are needed to reason about such greedy adversaries. They were motivated by how salts in password hashing protect against attackers re-using precomputation to retrieve multiple passwords.

For their study into probabilistic symmetric schemes, they identified left-or-right indistinguishability under xor as the strongest notion. Roughly speaking, there are κ keys in the system each associated with its own left-or-right challenge bit b_i and the goal of the adversary is to guess the xor of all those bits.

Recently, Auerbach, Giacon and Kiltz [4], henceforth AGK, argued the importance of BRT's concept to protect against mass surveillance. They introduced the (n, κ) scaling factor as the effort to break n out of κ instances relative to the effort needed to break a single instance. After recalling several well-known greedy attacks against public key schemes with dubious scaling factors, they set out to provide an encryption scheme with good, non-trivial scaling factor.

They discussed various versions of Hashed ElGamal that differed in whether users shared group parameters and/or generators, plus whether the underlying group was elliptic curve or finite field based. In the programmable random oracle model, they showed that the multi-instance security of Hashed ElGamal tightly relates to a novel multi-instance Gap Computational Diffie–Hellman (MI-GapCDH) assumption, whose validity was further supported by an analysis in the generic group model.

There was, or rather is, just one small problem: Hashed ElGamal is a key encapsulation mechanism (KEM), not a public key encryption (PKE) scheme. Indeed, although AGK use PKE as their motivation, their formalization is entirely centred around KEMs. Of course, Cramer and Shoup [17] already showed how a secure KEM can be combined with a secure data encapsulation mechanism (DEM) to create a secure PKE (for various notions of security). This so-called hybrid encryption paradigm is widely deployed in the real world, yet, can its composition theorem be easily lifted to the multi-instance setting?

For key unrecoverability, all seems fine, but for indistinguishability one quickly uncovers various challenges. Consider an adversary $\mathbb{A}$ that wants to recover n out of κ challenge bits b_i: it can attempt to recover roughly half of its b_i by somehow breaking the DEM, and recovering the remaining half by breaking the KEM. Intuitively, such a divide-and-conquer strategy essentially rules out inheriting full multi-instance security of both KEM and DEM simultaneously. Instead, perhaps we should aim to bound an adversary's multi-instance advantage against the hybrid encryption in terms of either breaking the full multi-instance security of the KEM or breaking only one of many instances of the DEM.

Special care would have to be taken to ensure that the corresponding multi-user DEM advantage is not overwhelming the multi-instance KEM advantage. After all, already when showing multi-user security of hybrid encryption, ensuring the DEM advantage does not overshadow the multi-user KEM advantage is challenging [21]. Furthermore, the study of multi-user KEMs highlights a second, more technical problem.

For multi-user security, there are essentially two different formalizations possible: one where each user comes with its own challenge bit and one where the users share a global challenge bit. Jager et al. [27] recently observed that only the latter lends itself to an easy adaptation of composition theorems using KEMs, as it allows a simple game-hop where all KEM-derived ephemeral keys are replaced

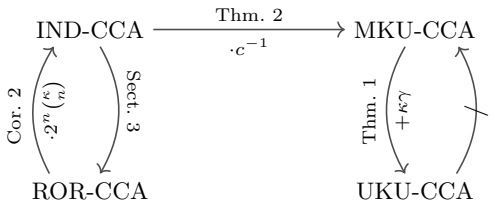

Fig. 1. An overview of multi-instance security notions for public-key encryption.

by randomly selected keys (decoupled from the KEM encapsulations). That proof technique fails when there are multiple challenge bits. Unfortunately, for multi-instance security, the only option available is a notion with multiple challenge bits. In such a setting, inheritance of security properties of the KEM to any construction based on the KEM is an open problem.

Our Contribution. As mentioned above, multi-instance security was introduced by BRT in the context of probabilistic symmetric primitives and later adapted to key encapsulation mechanisms by AGK, who provide an excellent motivation for the study of multi-instance security in a public key setting. We adapt those notions to multi-instance security for PKE schemes, but make a number of non-trivial changes in the process. Firstly, we observe that the mechanisms used by BRT and AGK to model multi-instance games differ, which seems to have gone unnoticed hitherto. BRT's mechanism is stronger as it allows for corruptions, yet AGK's mechanism is more expressive by making explicit how many instances an adversary should break. We use elements of both in our notions, incorporating both BRT's corruptions and AGK's explicit expression of the number of targeted instances. Secondly, we allow for correctness to be imperfect, which has ramifications for how to deal with decryption oracles (for chosen-ciphertext attacks) and corruptions. We delve into the differences between the various mechanisms in Sect. 3.3 and we use the revised mechanism to study a number of related notions, as summarized in Fig. 1.

In more detail, we start out by porting BRT's notion of key unrecoverability to the public-key setting. In fact, we consider two distinct versions of key unrecoverability: "UKU" where the adversary is tasked to recover the exact challenge private key(s) and "MKU" where it suffices to recover suitably equivalent private keys, where we leverage our imperfect correctness notion to define "suitably equivalent". As one would expect, this relaxed key unrecoverability notion implies the stronger, exact notion up to a small loss related to how we model imperfect correctness (Thm. 1).

For our main notion of multi-instance security, we follow BRT's identification of left-or-right xor-indistinguishability as the strongest notion and adapt it to the public key setting. As for the symmetric encryption setting studied by BRT, this indistinguishability notion implies the above key unrecoverability notions

(Thm. 2); however, the differences between perfect symmetric encryption and imperfect PKE affect the corresponding implications and their proofs.

Finally, we explore an alternative notion, namely real-or-random xor-indistinguishability. Trivially, left-or-right tightly implies real-or-random and in the multi-instance setting BRT showed that the usual factor-2 loss from the single instance implication between real-or-random to left-or right, becomes an exponential factor-2^κ loss. A similar loss is possible in our setting, however, we can also achieve a typically preferable bound of $\binom{\kappa}{n}2^n$ (Cor. 2).

With suitable notions for multi-instance PKE available, we focus on how to turn a suitably multi-instance secure KEM into a multi-instance secure PKE scheme using hybrid encryption. For key unrecoverability, inheritance is immediate, yet we would like to guarantee good multi-instance indistinguishability (the left-hand branch of Fig. 1). We summarize our findings in Fig. 2.

Our first observation is that we can expand the length of the ephemeral key to any desired length using a pseudorandom extendable output function (XOF). The resulting extendable KEM, or XEM, inherits the multi-instance security of the underlying KEM, provided the XOF is secure against multi-challenge adversaries (Thm. 5). To ensure that the XOF does not become the weakest link, its seed will need to be long enough, which in turn implies that the underlying KEM already needs to output a sufficiently long ephemeral key.

The XOF above of course plays the role of key derivation function, but it is more common that it is modelled as part of any key expansion done by the DEM. Moving it into the KEM allows us to use an information-theoretic DEM, read one-time pad (OTP), irrespective of the message length. The OTP's properties enable a simplified proof for the security of hybrid encryption (Thm. 6), where the PKE does indeed inherit the multi-instance security of the XEM, with two important caveats. Firstly, the OTP is only passively secure, so the PKE only achieves CPA not CCA security, and secondly, standard KEM indistinguishability only tightly provides real-or-random indistinguishability for the PKE (see the top line of Fig. 2).

Switching to the TagKEM framework [2], or in our case TagXEM, takes care of the first shortcoming and tightly achieves multi-instance ROR-CCA secure PKE, or IND-CCA non-tightly (Thm. 7). For the PKE to inherit multi-instance IND-CCA security tightly, we introduce a novel KEM indistinguishability notion that more closely matches PKE's left-or-right idea, namely real-or-permuted (ROP). Finally, we can show tight multi-instance inheritance for the most desirable PKE notion, based on a ROP-secure TagXEM (Thm. 8).

One small hiccough remains, as our KEM-to-XEM result unfortunately only works for classical KEM indistinguishability, not for ROP indistinguishability, nor does it look feasible to convert a KEM or XEM to a TagKEM or TagXEM, respectively, inheriting multi-instance security using standard reductions. Here, the random oracle, as used by AGK to prove their construction secure, comes to the rescue, although rather than looking at Hashed ElGamal under the MI-GapCDH assumption, we consider more general KEMs that are multi-instance one-way under plaintext checking attacks (unfortunately, also at this point we

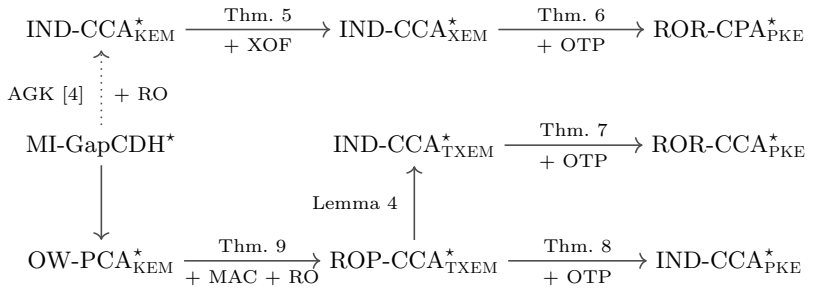

Fig. 2. An overview of our constructions achieving various flavours of multi-instance security. The left upwards arrow is dotted, as AGK did not consider corruptions.

need to restrict to perfect correctness), which we combine with Abe et al.'s TagKEM construction from a KEM and a MAC (message authentication code).

Recalling that the original random oracle [13] was in fact a XOF, we can bake the extendability into the random oracle, including the key needed for an information-theoretic secure MAC. Moreover, the power of the ROM allows proving the stronger ROP indistinguishability just as easily as classical KEM indistinguishability. All in all, with Thm. 9 we achieve a suitably multi-instance secure TagXEM based on a KEM that can be instantiated by Hashed ElGamal. In that case, the security relies on the MI-GapCDH assumption with corruptions.

For low granularity, which corresponds to a setting where every user generates its own group as part of its public key, AGK's technique can easily be extended to include corruptions and in the generic group model we arrive at the same bound for the hardness of MI-GapCDH with corruptions as AGK did (without corruptions). Unfortunately, for the more realistic high granularity setting, where users share the same (standardized) group, AGK's proof strategy does not easily allow incorporating corruptions. We provide details in Suppl. Mat. 3.

Thus, we can conclude that XOF-based Hashed ElGamal combined with a suitable information-theoretically secure MAC and the one-time-pad, provides good multi-instance security in the programmable random oracle model and generic group model, provided that users each select their own independent group. We briefly touch upon a concrete interpretation in Suppl. Mat. 4, where we also informally address AGK's scaling factor.

Related Work. Farshim and Tessaro [18] recently followed up BRT's line of work on the multi-instance security of password hashing by combining it with the related preprocessing setting. AGK [4] motivated their investigation into multi-instance security by the threat of mass surveillance. The latter had previously motivated Bellare et al. [10] to consider subversion, namely the ease with which a "big brother" might subvert an encryption algorithm by replacing it surreptitiously with a trapdoored one with otherwise identical behaviour.

The multi-instance setting is closely related to the multi-user setting, in which the adversary is tasked with breaking only one rather than n out of κ possible instances. Multi-user security was introduced by Bellare et al. [6] in the public-key setting, with the goal of deriving concrete security parameters in a more realistic setting. There have been many recent follow-up works, including how the hybrid paradigm generalizes to the setting without corruptions [21], and later with corruptions [29], as well as the construction of tightly-secure authenticated key exchange from multi-user KEMs [27].

One definitional subtlety of multi-user security is the number of challenge bits: either a single one, as originally conceived, or many, as typical for the multi-instance setting. The various definitions do not appear to imply each other tightly [24], which slightly hinders regarding the multi-user setting as a special case of the multi-instance setting (due to potential tightness losses).

2 Preliminaries

2.1 Notation

For a positive integer n, we write $[n]$ for the set $\{1, \ldots, n\}$. We use code-based experiments, where by convention all sets and lists are initialized empty. For a set $\mathtt{X}$, we use the shorthand $\mathtt{X} \xleftarrow{\cup} x$ for the operation $\mathtt{X} \leftarrow \mathtt{X} \cup \{x\}$. If $\mathtt{X}$ is a list, then $\mathtt{X} \xleftarrow{\frown} x$ denotes appending the element x to $\mathtt{X}$; to retrieve the ith element of the list, we write $\mathtt{X}[i]$ where by convention $\mathtt{X}[i] = \emptyset$ for out-of-bounds i.

We use $\Pr[Code : Event \mid Condition]$ to denote the conditional probability of $Event$ occurring when $Code$ is executed, conditioned on $Condition$. We omit $Code$ when it is clear from the context and $Condition$ when it is not needed. For Boolean values, we use $\{\mathsf{true}, \mathsf{false}\}$ and $\{0, 1\}$ interchangeably, where by convention 1 corresponds to true.

When proving relations between notions and security of constructions, we will often refer to simple fully black box (SFBB) reductions. A reduction is fully black box iff it works for all schemes and adversaries, and only accesses them in a black box manner [5, 34] (we leave the black box dependence implicit in our notation). Moreover, if the reduction only runs its adversary once and without rewinding, then the reduction is simple [30].

Finally, the respective games that the adversary and the reduction are playing often have matching (though not identical) oracles; for instance, both may have access to a decryption oracle or a key corruption oracle. We call a reduction type-preserving with respect to, say, a decryption oracle iff the reduction will make decryption queries iff its black-box adversary makes decryption queries. Type-preservation, without explicit mention of any oracles, is implicitly meant to imply for all meaningfully matching oracles (unless otherwise specified).

Type-preservation of reductions appears folklore and can easily be established by inspection. Intuitively, a type-preserving reduction can be used to show simultaneously that CCA security of some kind implies CCA security of another kind and that CPA security of the same kind implies CPA security of the other

kind. In Sect. 3.3 we will encounter several reductions that are only partially type-preserving.

2.2 PKE Syntax

A public-key encryption scheme PKE consists of four algorithms: the probabilistic key generation algorithm $\mathsf{PKE.Kg}$, which takes as input some system parameter pm (see also Remark 1) and outputs a public/private key pair $(\mathsf{pk}, \mathsf{sk})$; the deterministic key validation algorithm $\mathsf{PKE.Check}$, which takes as input the system parameters pm as well as a purported public/private key pair $(\mathsf{pk}, \mathsf{sk})$ and returns true or false (see Remark 2 below), the probabilistic encryption algorithm $\mathsf{PKE.Enc}$, which on input a public key pk and a message $m \in \mathcal{M}$ (see Remark 3), outputs a ciphertext c; and the deterministic decryption algorithm $\mathsf{PKE.Dec}$, which on input of a secret key sk and a ciphertext c, outputs either a message m, or a special symbol $\perp$ denoting failure.

Remark 1. The system parameters pm are implicitly input to $\mathsf{PKE.Enc}$ and $\mathsf{PKE.Dec}$ as well; for concreteness, they can for instance be the description of an elliptic curve group with generator for an ECDLP-based system or the dimensions and noise sampling algorithm for an LWE-based system. When one is interested in re-phrasing our results in an asymptotic setting, the parameters pm will be generated by a probabilistic, polynomial-time algorithm that only takes the security parameter as input.

Remark 2. For various modern cryptosystems, especially schemes targeting post-quantum security or tight multi-user security, the relationship between public and private keys is not one-to-one. For instance, a single public key can have various private keys [21] or a single private key can lead to various public keys [15]. However, it is usually possible to check easily whether a public key and private key belong together, which we model by the key validation algorithm $\mathsf{PKE.Check}$. We will define both correctness and key recoverability in terms of this key validation algorithm.

Remark 3. The message space $\mathcal{M}$ may depend on the parameters pm, but for simplicity we assume it independent of the public key pk. Often $\mathcal{M}$ consists of arbitrary length bitstrings, or at least all bitstrings up to some large length (e.g. 2^{64}) and messages of the same length are deemed equivalent as they are expected to yield ciphertexts of identical lengths. We will model these equivalences more abstractly by assuming that pm implicitly defines a number m of equivalence classes, together with an efficient method $[\![\cdot]\!] : \mathcal{M} \to [\mathsf{m}]$ to determine the class (e.g. length) of a message and an efficient algorithm to sample uniformly from a given equivalence class. We write $\sim$ for the equivalence, so for $m \in \mathcal{M}$, $m \sim m'$ iff $[\![m]\!] = [\![m']\!]$.

Correctness. Perfect correctness states that for all parameters pm, all key pairs $(\mathsf{pk}, \mathsf{sk})$ that can be output by $\mathsf{PKE.Kg}(\mathsf{pm})$, and all messages $m \in \mathcal{M}$, we always have that $\mathsf{PKE.Dec}_{\mathsf{sk}}(\mathsf{PKE.Enc}_{\mathsf{pk}}(m)) = m$. Yet modern schemes, especially

lattice-based ones, often allow a small decryption error, where occasionally decryption will fail or it will return a wrong message.

Various relaxations of correctness have appeared in the literature in order to argue about such schemes as it turns out that some classical results implicitly or subtly relied on perfect correctness. In order for our work to be meaningful for a large range of both classical and modern schemes, we introduce a stronger version of imperfect correctness based on the key validation algorithm.

Definition 1 ((γ,δ)-Correctness). *Let $\gamma, \delta \in [0,1]$. Then a public-key encryption scheme* PKE *is called (γ,δ)-correct iff*

1. $\Pr[(\mathsf{pk},\mathsf{sk}) \leftarrow\!\$\ \mathsf{PKE.Kg}(\mathsf{pm}) : \mathsf{PKE.Check}(\mathsf{pm},\mathsf{pk},\mathsf{sk}) = \mathsf{false}] \leq \gamma$;
2. *if* $\mathsf{PKE.Check}(\mathsf{pm},\mathsf{pk},\mathsf{sk}) = \mathsf{true}$ *then for all* $m \in \mathcal{M}$ *it holds that*

$$\Pr[\mathsf{PKE.Dec}_{\mathsf{sk}}(\mathsf{PKE.Enc}_{\mathsf{pk}}(m)) \neq m] \leq \delta\ .$$

Perfect correctness corresponds to $(0,0)$-correctness and any scheme is trivially both $(1,0)$-correct and $(0,1)$-correct. For good schemes γ and δ can simultaneously be chosen small, where typically increasing γ allows for decreasing δ. As we will see, both γ and δ will appear in various bounds, thus allowing larger γ to enable smaller δ (or vice versa) might give preferable bounds.

3 Multi-Instance Security of Public-Key Encryption

3.1 Two Flavours of Key Recovery

The minimal requirement for public-key encryption schemes is that, given a public key, it should be difficult to recover the private key. Although key unrecoverability is a very weak notion theoretically, its study has two main motivations: firstly, many multi-instance attacks target key recovery, and secondly, conceptually the notion is relatively simple, allowing both an instructive introduction of formalizing multi-instance security and an initial comparison between BRT's perfect symmetric encryption and our imperfect public key encryption.

At first sight, the generalization to the multi-instance setting appears immediate: an adversary tries to recover the respective private keys for a number of public keys. BRT introduced universal key unrecoverability (UKU) as a suitable notion for multi-instance security of symmetric encryption. We provide an analogue for public-key encryption, but there are some crucial changes in the game's mechanics (see also Sect. 3.3).

Let $0 < n \leq \kappa$ be integer parameters, then the universal key unrecoverability experiment $\mathsf{Exp}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A})$ for public-key encryption scheme PKE and adversary $\mathbb{A}$ is described in Fig. 3. It generates κ key pairs and provides the public keys to $\mathbb{A}$, who is then tasked with recovering exactly n of the corresponding private keys.

The adversary has access to both a decryption oracle $\mathcal{D}$ and a key corruption oracle $\mathcal{K}$, giving rise to chosen ciphertexts attacks with corruptions (CCA$\star$; the

Experiment $\mathsf{Exp}^{(n,\kappa)\text{-(u/m)ku-cca}\star}_{\mathrm{PKE}}(\mathbb{A})$

$(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \leftarrow\!\!\$\ \mathsf{PKE.Kg}$

$(\mathtt{I}, (\hat{\mathsf{sk}}_i)_{i\in\mathtt{I}}) \leftarrow\!\!\$\ \mathbb{A}^{\mathcal{D},\mathcal{K}}(\mathsf{pk}_1, \ldots, \mathsf{pk}_\kappa)$

if $|\mathtt{I}| \neq n \vee \mathtt{I} \cap \mathtt{K} \neq \emptyset$ **then return** 0

UKU : **return** $\bigwedge_{i\in\mathtt{I}} \mathsf{sk}_i = \hat{\mathsf{sk}}_i$

MKU : **return** $\bigwedge_{i\in\mathtt{I}} \mathsf{PKE.Check}(\mathsf{pk}_i, \hat{\mathsf{sk}}_i,)$

Oracle $\mathcal{D}(i, c)$

$m \leftarrow \mathsf{PKE.Dec}_{\mathsf{sk}_i}(c)$

return m

Oracle $\mathcal{K}(i)$

$\mathtt{K} \xleftarrow{\cup} i$

return sk_i

Fig. 3. The key recovery experiments $\mathsf{Exp}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A})$ and $\mathsf{Exp}^{(n,\kappa)\text{-mku-cca}\star}_{\mathrm{PKE}}(\mathbb{A})$; they only differ in their win condition.

$\star$ denotes corruptions). The decryption oracle $\mathcal{D}(i, c)$ takes as input an index i and a ciphertext c, and returns the output of the decryption algorithm $\mathsf{PKE.Dec}$ on input sk_i and c. The corruption oracle $\mathcal{K}(i)$ simply takes as input a key index i, and returns the corresponding private key sk_i. The game notes that the key pair with index i has been corrupted by adding it to the global set $\mathtt{K}$.

Eventually, $\mathbb{A}$ outputs a set of key indices $\mathtt{I}$ and a list $(\hat{\mathsf{sk}}_i)_{i\in\mathtt{I}}$ of guesses of the private keys corresponding to those indices. In order for $\mathtt{I}$ to be eligible, it needs to have cardinality n without containing any corrupted key pairs, that is, the sets of guessed keys $\mathtt{I}$ and corrupted keys $\mathtt{K}$ should be disjoint. If $\mathtt{I}$ is eligible and every guessed private key matches the corresponding sampled one, the adversary wins the game. In that case, the game halts with output 1; otherwise, it halts with output 0. The advantage is the probability that the game outputs 1.

Definition 2. *Let* PKE *be a public-key encryption scheme. Then the universal key unrecoverability advantage of an adversary* $\mathbb{A}$ *is*

$$\mathsf{Adv}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A}) = 1\right],$$

where the experiment is defined in Fig. 3.

Weaker notions emerge by dropping either or both of the two oracles. Without key corruption, standard CCA security results. Without decryption oracle, chosen plaintext security (CPA$\star$ resp. CPA) emerges. As usual, an encryption oracle is superfluous in the PKE setting.

For cryptosystems where a single public key may have many matching private keys (such as Cramer–Shoup [16]), universal key unrecoverability is rather weak. Hence, we consider a second, slightly stronger notion of key recovery, in which the recovered private keys are no longer required to be identical to those sampled in the game. Instead, it suffices that each passes the keypair checking algorithm $\mathsf{PKE.Check}$; here we leverage our correctness definition (Def. 1). We call the resulting notion *matching key unrecoverability* (MKU), whose game is included in Fig. 3. That MKU security indeed implies UKU security is captured by Thm. 1 below, where the error term $\kappa\gamma$ results from the unique correct keys

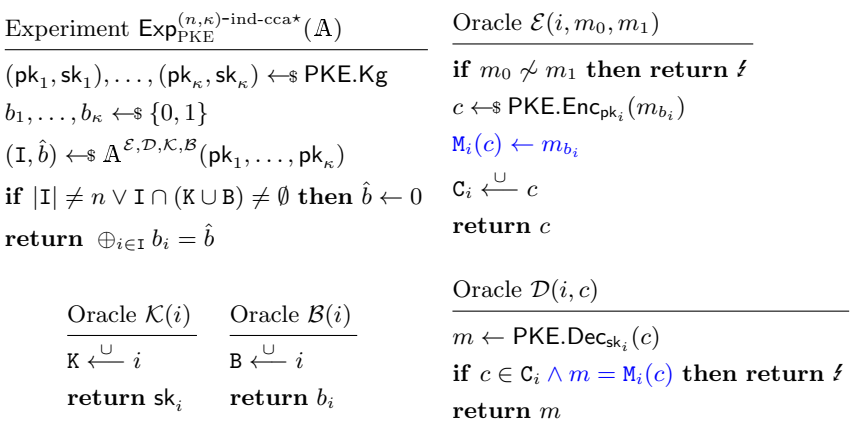

Fig. 4. Our main notion of multi-instance indistinguishability. In blue the slightly non-standard strengthening of the decryption oracle in case of imperfect correctness.

as output by the key generation not always passing the $\mathsf{PKE.Check}$ algorithm (see Suppl. Mat. 1.1 for the proof).

Theorem 1 (MKU $\longrightarrow$ UKU)**.** *Let $0 < n \leq \kappa$ be integer parameters and let* PKE *be a (γ,δ)-correct encryption scheme. Then, there is a type-preserving SFBB reduction $\mathbb{B}_{\mathrm{mku}}$, such that for every adversary $\mathbb{A}_{\mathrm{uku}}$,*

$$\mathsf{Adv}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A}_{\mathrm{uku}}) \leq \mathsf{Adv}^{(n,\kappa)\text{-mku-cca}\star}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{mku}}) + \kappa\gamma\,.$$

3.2 Left-or-Right XOR Indistinguishability

To capture a stronger notion of security than simply hardness of key recovery, BRT considered various generalizations of indistinguishability to the multi-instance setting. For perfect probabilistic symmetric encryption, they concluded that left-or-right xor-indistinguishability is the strongest notion. Here each key comes with its own challenge bit that determines the left-or-right nature of the corresponding challenge encryption oracle; the adversary is tasked to retrieve the xor of all the challenge bits. In Def. 3, we use our modified game mechanics to adapt left-or-right xor-indistinguishability for potentially non-perfect public-key encryption.

Definition 3. *Let* PKE *be a public-key encryption scheme. Then the xor-indistinguishability advantage of an adversary $\mathbb{A}$ is*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{A}) = 2 \cdot \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{A}) = 1\right] - 1\,,$$

where the experiment is defined in Fig. 4.

In the experiment $\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{A})$, the adversary gets access to κ independently drawn public keys and helper oracles $\mathcal{D}$ and $\mathcal{K}$ (as described in Sect. 3.1). Furthermore, $\mathbb{A}$ gets access to a challenge encryption oracle $\mathcal{E}$ and a separate bit corruption oracle $\mathcal{B}$.

On input two equivalent messages m_0 and m_1 and a public key index i, the challenge encryption oracle returns $\mathsf{PKE.Enc}_{\mathsf{pk}_i}(m_{b_i})$ where b_i is the challenge bit associated with the public key indexed by i. As usual for IND-CCA notions, challenge ciphertexts cannot be queried to the decryption oracle, which we catch on-the-fly [8]. Owing to the imperfect decryption, we allow a slight relaxation: if a challenge ciphertext decrypts incorrectly, we do not suppress the output and essentially allow the query. This relaxation strengthens the notion, but as challenge ciphertexts are honestly generated, the advantage gained by an adversary can be bound by the correctness parameters of the PKE using an identical-until-bad argument; however such a generic approach might not give bounds appropriate for the multi-instance setting.

Eventually, the adversary returns a set $\mathtt{I}$ of targets and a guess $\hat{b}$ of the xor of the corresponding challenge bits b_i. If $\mathtt{I}$ is a set of n uncorrupted indices, then intuitively an adversary's uncertainty about any of the n challenge bits will be affected in the final guess $\hat{b}$, so in that sense $\hat{b}$ neatly captures an adversary's need to break n instances in order to win. If $\mathtt{I}$ is not a set of n uncorrupted indices, the game resets $\mathbb{A}$'s guess $\hat{b}$ to 0, ensuring an adversary gains zero advantage from such a bad $\mathtt{I}$.

The Relationship with Key Recovery. BRT showed that in their perfect symmetric setting, multi-instance indistinguishability implies multi-instance universal key unrecoverability. While that may sound like a triviality, their proof [12, App. C] was not entirely straightforward and, to ensure that the advantages carried over neatly, the distinguishing reduction receiving recovered keys needed to amplify its success probability by repeated random challenge encryptions. Their bound ends up with an additive term that corresponds to the likelihood that decrypting using an incorrect key results in the opposite message from the decrypted one.

Our imperfect public key setting is slightly different. On the one hand, the reduction can check the recovered keys with the $\mathsf{PKE.Check}$ algorithm, yet on the other hand correct keys can still cause incorrect decryptions. As a result, our amplification based on multiple challenge encryptions differs from BRT's, as we move from unanimity to a plurality vote. Furthermore, our reduction can use fixed messages (to match how correctness is defined), which reduces a dependency (in the bound) on the size of the message space. We suspect that our amplification can be tightened further by a combination of exploiting randomness and more fine-tuned voting, coupled with more fine-grained bounding of probabilities.

As is, the complexity of the bound makes its behaviour somewhat opaque and for some parameter choices vacuous (when $c < 0$). The main idea is that $\mathbb{B}_{\mathrm{ind}}$ can increase q, the number of challenge encryptions per user, to counteract

the losses inferred by large n and/or large δ, with a small penalty to its running time. For $\delta = 2^{-64}$, $q = 1$ already suffices for $c > 1/2$ for $n < 2^{25}$. In case of perfect correctness for keys that check out, corresponding to $\delta = 0$, the bound is completely tight.

Theorem 2 (IND $\longrightarrow$ MKU**).** *Let* PKE *be a (γ, δ)-correct encryption scheme with $\delta < 1/2$. Then there is a type-preserving SFBB reduction $\mathbb{B}_{\mathrm{ind}}$ such that, for every $\mathbb{A}_{\mathrm{mku}}$,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{ind}}) \geq c \cdot \mathsf{Adv}^{(n,\kappa)\text{-mku-cca}\star}_{\mathrm{PKE}}(\mathbb{A}_{\mathrm{mku}})\,,$$

with $c = 2\left(1 - 2^q(\delta(1-\delta))^{\frac{q}{2}}\right)^n - 1$ where $q \in \mathbb{Z}_{>0}$ is an amplification parameter of the reduction; $\mathbb{B}_{\mathrm{ind}}$'s overhead consists of $q \cdot n$ calls to $\mathcal{E}$, n offline key checks, and $q \cdot n$ offline decryptions.

Proof. Let $\mathbb{B}_{\mathrm{ind}}$ run adversary $\mathbb{A}_{\mathrm{mku}}$ on the same κ public keys as it received itself. Whenever $\mathbb{A}_{\mathrm{mku}}$ makes a decryption or corruption query, $\mathbb{B}_{\mathrm{ind}}$ simply forwards the queries to its own oracle, relaying the response back to $\mathbb{A}_{\mathrm{mku}}$. Eventually, $\mathbb{A}_{\mathrm{mku}}$ terminates with output $(\mathtt{I}, (\hat{\mathsf{sk}}_i)_{i\in\mathtt{I}})$ and $\mathbb{B}_{\mathrm{ind}}$ first confirms whether $\mathbb{A}_{\mathrm{mku}}$ won, by checking, for all the returned private keys, whether $\mathsf{PKE.Check}(\mathsf{pk}_i, \hat{\mathsf{sk}}_i)$ holds. If any check fails, $\mathbb{B}_{\mathrm{ind}}$ halts with output 0.

Let m_0 and m_1 be two distinct yet equivalent messages. Then for all $i \in \mathtt{I}$, $\mathbb{B}_{\mathrm{ind}}$ creates a guess $\hat{b}_i$ by querying its challenge encryption oracle q times on those two messages, so q queries $\mathcal{E}(i, m_0, m_1)$ resulting in c_{ij}, for $j \in [q]$. It then decrypts those ciphertexts using the private key $\hat{\mathsf{sk}}_i$ it obtained from $\mathbb{A}_{\mathrm{mku}}$, resulting in purported messages $m_{ij} \leftarrow \mathsf{PKE.Dec}_{\hat{\mathsf{sk}}_i}(c_{ij})$. If, for a fixed i, there are strictly more than $q/2$ appearances of m_0 amongst the m_{ij}, it sets $\hat{b}_i$ to 0; if there are strictly more than $q/2$ appearances of m_1, then it sets $\hat{b}_i$ to 1. If neither message appears more than $q/2$ times, $\mathbb{B}_{\mathrm{ind}}$ halts with output 0. Once $\mathbb{B}_{\mathrm{ind}}$ has created a guess $\hat{b}_i$ for all $i \in \mathtt{I}$, it terminates on output $(\mathtt{I}, \bigoplus_{i\in\mathtt{I}} \hat{b}_i)$.

For $i \in \mathtt{I}$, let Check_i be the event that $\mathbb{A}_{\mathrm{mku}}$ outputs a key $\hat{\mathsf{sk}}_i$ that passes the test and let Good_i be the event that $\mathbb{B}_{\mathrm{ind}}$'s guess $\hat{b}_i$ actually equals b_i. Let $\mathsf{Check}_{\mathtt{I}}$ be the event that all Check_i hold (for $i \in \mathtt{I}$) and define $\mathsf{Good}_{\mathtt{I}}$ analogously.

As $\mathbb{B}_{\mathrm{ind}}$'s simulation of $\mathsf{Exp}^{(n,\kappa)\text{-mku-cca}\star}_{\mathrm{PKE}}$ is perfect, we know that

$$\mathsf{Adv}^{(n,\kappa)\text{-mku-cca}\star}(\mathbb{A}_{\mathrm{mku}}) = \Pr[\mathsf{Check}_{\mathtt{I}}]\,,$$

moreover,

$$\begin{aligned}\Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{ind}}) = 1\right] &\geq \Pr[\mathsf{Check}_{\mathtt{I}} \wedge \mathsf{Good}_{\mathtt{I}}] + \Pr[\neg\mathsf{Check}_{\mathtt{I}} \wedge b = 0] \\ &= \Pr[\mathsf{Good}_{\mathtt{I}} \mid \mathsf{Check}_{\mathtt{I}}]\Pr[\mathsf{Check}_{\mathtt{I}}] + \frac{1}{2}\left(1 - \Pr[\mathsf{Check}_{\mathtt{I}}]\right)\end{aligned}$$

which implies that

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{ind}}) \geq \left(2\Pr[\mathsf{Good}_{\mathtt{I}} \mid \mathsf{Check}_{\mathtt{I}}] - 1\right)\mathsf{Adv}^{(n,\kappa)\text{-mku-cca}\star}(\mathbb{A}_{\mathrm{mku}})\,.$$

To bound $\Pr[\mathsf{Good}_{\mathrm{I}} \mid \mathsf{Check}_{\mathrm{I}}]$ we exploit the correctness definition, specifically that its quantification (Def. 1) ensures that whenever Check_i holds, we have that $\Pr\Big[\mathsf{PKE.Dec}_{\hat{\mathsf{sk}}_i}(\mathsf{PKE.Enc}_{\mathsf{pk}_i}(m)) = m\Big] \geq 1 - \delta$, irrespective of m and where the probability is only over the randomness of $\mathsf{PKE.Enc}$.

If, for a given i, decryption is correct strictly more than $q/2$ times, then we are guaranteed that Good_i occurs. If we let $B\big(q,p\big)$ be the binomial distribution over q trials and with probability p, then

$$\Pr[\mathsf{Good}_i \mid \mathsf{Check}_i] \geq \Pr\Big[B\big(q,(1-\delta)\big) > \frac{q}{2}\Big]$$

and, as this bound only relies on the randomness of the challenge encryption oracle, guaranteed independent for differing i, we may conclude that

$$\Pr[\mathsf{Good}_{\mathrm{I}} \mid \mathsf{Check}_{\mathrm{I}}] \geq \Big(\Pr\Big[B\big(q,(1-\delta)\big) > \frac{q}{2}\Big]\Big)^n .$$

Finally, we note that

$$\Pr\Big[B\big(q,(1-\delta)\big) > \frac{q}{2}\Big] \geq 1 - 2^q \left(\delta(1-\delta)\right)^{\frac{q}{2}}$$

by a standard application of known bounds on binomial tails, requiring $\delta \leq 1/2$ (see details below). Plugging in all the various bounds recovers the theorem statement.

For the binomial tail bound, we use the Chernoff–Hoeffding bound [25], which states that, for a binomial distribution $B\big(q,p\big)$ over q trials and with probability p, and any k satisfying $p < \frac{k}{q} < 1$ the tail bound

$$\Pr\big[B\big(q,p\big) \geq k\big] \leq \exp\left[-qD\left(\frac{k}{q}\middle\| p\right)\right]$$

holds, where $D(a\|b)$ is the Kullback–Leibler divergence defined as $D(a\|b) = a\ln\left(\frac{a}{b}\right) + (1-a)\ln\left(\frac{1-a}{1-b}\right)$.

We further use the trick that $\Pr\big[B\big(q,(1-\delta)\big) > \frac{q}{2}\big] = 1 - \Pr\big[B\big(q,\delta\big) \leq \frac{q}{2}\big]$, so the relevant Kullback–Leibler divergence becomes

$$\begin{aligned} D\left(\frac{1}{2}\middle\| \delta\right) &= \frac{1}{2}\ln\left(\frac{\frac{1}{2}}{\delta}\right) + \left(1-\frac{1}{2}\right)\ln\left(\frac{\left(1-\frac{1}{2}\right)}{1-\delta}\right) \\ &= \frac{1}{2}\ln\left(\frac{1}{2\delta}\right) + \frac{1}{2}\ln\left(\frac{1}{2(1-\delta)}\right) \\ &= \ln\left[\left(\frac{1}{4\delta(1-\delta)}\right)^{\frac{1}{2}}\right], \end{aligned}$$

Experiment $\mathsf{Exp}^{(\leq\kappa,\kappa)\text{-ind-cca}^\star}_{\mathrm{PKE}}(\mathbb{A})$	Experiment $\mathsf{Exp}^{(\geq n,\kappa)\text{-ind-cca}^\star}_{\mathrm{PKE}}(\mathbb{A})$
4 : **if** $\lvert\mathtt{I}\rvert \neq \kappa$ **then** $\hat{b} \leftarrow 0$	4 : **if** $\lvert\mathtt{I}\rvert < n \vee \mathtt{I} \cap (\mathtt{K} \cup \mathtt{B}) \neq \emptyset$ **then** $\hat{b} \leftarrow 0$

Fig. 5. The main differences between our mechanism for multi-instance indistinguishability (Fig. 4) and prior art revolve around line 4: BRT's experiment $\mathsf{Exp}^{(\leq\kappa,\kappa)\text{-ind-cca}^\star}_{\mathrm{PKE}}(\mathbb{A})$ (left) and AGK's experiment $\mathsf{Exp}^{(\geq n,\kappa)\text{-ind-cca}^\star}_{\mathrm{PKE}}(\mathbb{A})$ (right). The differences are highlighted in blue.

which allows us to compute the bound

$$\begin{aligned}\Pr\Big[B\big(q,(1-\delta)\big) > \frac{q}{2}\Big] &\geq 1 - \exp\left[-qD\left(\frac{1}{2}\middle\|\,\delta\right)\right] \\ &= 1 - \exp\left[-q\ln\left[\left(\frac{1}{4\delta(1-\delta)}\right)^{\frac{1}{2}}\right]\right] \\ &= 1 - 2^q\left(\delta(1-\delta)\right)^{\frac{q}{2}}\,.\end{aligned}$$

□

Corollary 1 (IND $\longrightarrow$ UKU). *Let* PKE *be a* (γ,δ)*-correct encryption scheme with* $\delta < 1/2$*. Then there is a type-preserving SFBB reduction* $\mathbb{B}_{\mathrm{ind}}$ *such that, for every* $\mathbb{A}_{\mathrm{uku}}$*,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}^\star}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{ind}}) \geq c \cdot \mathsf{Adv}^{(n,\kappa)\text{-uku-cca}^\star}_{\mathrm{PKE}}(\mathbb{A}_{\mathrm{uku}}) - \kappa\gamma\,,$$

with c, q*, and* $\mathbb{B}_{\mathrm{ind}}$*'s overhead as above (Thm. 2).*

3.3 Alternative Mechanisms

As we mentioned before, our mechanism to capture multi-instance security differs slightly from those used by BRT and AGK, respectively, even when accounting for changes in primitive and correctness. At first sight, the differences might appear mostly cosmetic, though there are some subtleties involved.

The BRT Notion: Requiring $n = \kappa$, possibly corrupted, targets. BRT require an adversary to return the xor of all bits, but allow those bits or corresponding users to be corrupted. Fig. 5 reflects the small change needed in the code of our security experiment to match BRT's mechanism (ignoring a minor, inconsequential difference, as BRT have a single, merged corruption oracle that returns both key and bit). As motivation for including corruptions, BRT discuss the scenario that, say, half of the keys generated are hopelessly insecure: an adversary breaks the insecure half and corrupts the rest, thus being successful. Moreover, they mention that their choice implies security under a corruptionless notion with dynamically chosen $\mathtt{I}$.

Although the implication is of course true, and something can be said to target the strongest possible notion, corruptions have a habit of creating complications for reductions and provable security in general. Yet, we believe the inclusion of corruptions, or not, should reflect the threat model of the adversary and that choice should be orthogonal to the number of users being targeted. BRT, instead of having an explicit hardness parameter n, restrict an adversary to make at most q_c corruption queries to avoid trivial wins when $q_c = \kappa$. Yet, whether the resulting, intuitive hardness will or should then match $n = \kappa - q_c$, is unclear.

We address the equivalence between BRT's mechanism and our general mechanism (with corruptions) in Lemmas 1 and 2. Both lemmas have in common that the respective reductions may make up to $\kappa - n$ additional bit corruptions. In other words, the reductions are not type-preserving, making the equivalence somewhat sloppy. As an aside, using techniques similar to those to prove Thm. 2, the key corruption oracle could be used (at a loss) to simulate the bit corruption oracle instead (see Suppl. Mat. 1.2 and Suppl. Mat. 1.3 for the proofs).

Lemma 1 (main notion $\implies$ BRT). *Let $n \leq \kappa$ and $q_c \leq \kappa - n$. Then there is an SFBB reduction $\mathbb{B}$ such that, for every adversary $\mathbb{A}$ making at most q_c corruption oracle calls,*

$$\mathsf{Adv}_{\mathrm{PKE}}^{(\leq\kappa,\kappa)\text{-ind-cca}^\star}(\mathbb{A}) \leq \mathsf{Adv}_{\mathrm{PKE}}^{(n,\kappa)\text{-ind-cca}^\star}(\mathbb{B})\,,$$

where $\mathbb{B}$ makes at most $\kappa - n$ additional bit corruption oracle calls.

Lemma 2 (BRT $\implies$ main notion). *Let $n \leq \kappa$. Then there is an SFBB reduction $\mathbb{B}$ such that, for every adversary $\mathbb{A}$,*

$$\mathsf{Adv}_{\mathrm{PKE}}^{(n,\kappa)\text{-ind-cca}^\star}(\mathbb{A}) \leq \mathsf{Adv}_{\mathrm{PKE}}^{(\leq\kappa,\kappa)\text{-ind-cca}^\star}(\mathbb{B})\,,$$

where $\mathbb{B}$ makes at most $\kappa - n$ additional bit corruption oracle calls.

The AGK Notion: Allowing more than n targets without corruptions. When AGK studied KEMs in the multi-instance setting, they used a xor notion with the n as the *minimum* number of targets to attack (out of κ possible) as an explicit parameter; moreover, an adversary would not have access to any corruption oracles. Fig. 5 reflects the small change needed in the code of our security experiment to match AGK's mechanism with corruptions added (where we fixed a minor bug in their code; rather than setting $\hat{b} \leftarrow 0$ their experiment would immediately return 0 instead).

Absent corruptions, AGK indicated that for some pathological schemes, breaking more targets might paradoxically be easier than breaking fewer [3, App. C]. In those cases, the freedom to return a set $\mathtt{I}$ of cardinality greater than n would make life easier for an adversary, leading to a stronger notion.

In the presence of corruptions, requiring the adversary to target exactly n users as we do is without loss of generality. As an example, if an adversary can

figure out the xor of $n+1$ honest bits, it can bit-corrupt any single one of these $n+1$, and xor the resulting bit out of the initial guess to obtain a final one on n bits instead. We formalize this intuition below.

Lemma 3 (main notion $\implies$ AGK*). *There is an SFBB adversary $\mathbb{B}$ such that, for every $\mathbb{A}$,*

$$\mathsf{Adv}^{(\geq n,\kappa)\text{-ind-cca}^\star}_{\mathrm{PKE}}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-ind-cca}^\star}_{\mathrm{PKE}}(\mathbb{B})\,.$$

If $\mathbb{A}$ returns a list of n' targets, $\mathbb{B}$ makes $n'-n$ additional calls to its bit corruption oracle.

3.4 Real-or-Random XOR Indistinguishability

An alternative notion of indistinguishability, known as real-or-random indistinguishability (ROR), sees the adversary tasked with figuring out whether a challenge ciphertext contains the adversarially chosen message m or an unknown, randomly chosen message. The game $\mathsf{Exp}^{(n,\kappa)\text{-ror-cca}^\star}_{\mathrm{PKE}}$ is exactly as in Fig. 4, apart from the challenge encryption oracle $\mathcal{E}_{\mathrm{ROR}}(i,m)$, which sets $m_0 \leftarrow m$ and $m_1 \leftarrow\!\!\$\ [m]$ to then call (left-or-right) $\mathcal{E}(i,m_0,m_1)$.

By construction, left-or-right indistinguishability easily implies real-or-random indistinguishability. That statement is as true in the multi-instance setting as it is in the classical single-user setting. Conversely, in the single-user setting, it has long been established that the reduction from ROR to IND loses a factor 2 [7]. However, BRT showed that in the multi-instance setting, the factor 2 blows up exponentially to, in their case, 2^κ. Yet, BRT argue that this exponential loss is not as bad as it might seem, given that the multi-instance advantages are supposed to be exponentially smaller than their single-user counterparts. Thus, reductions incurring losses exponential in κ or n can still be valuable.

To adapt BRT's reduction to our setting, we require $n=\kappa$, implying that $\mathbb{A}$ cannot access its corruption oracles. Otherwise, corruptions would make the reduction noticeable once at least one b_i is set to 1, potentially influencing an adversary's behaviour in unpredictable ways (see Suppl. Mat. 1.4 for the proof).

Theorem 3. *There is an SFBB reduction $\mathbb{B}$ such that, for every adversary $\mathbb{A}$,*

$$\mathsf{Adv}^{(\kappa,\kappa)\text{-ind-cca}}_{\mathrm{PKE}}(\mathbb{A}) \leq 2^\kappa \cdot \mathsf{Adv}^{(\kappa,\kappa)\text{-ror-cca}}_{\mathrm{PKE}}(\mathbb{B})\,,$$

where $\mathbb{B}$ additionally draws κ bits uniformly at random.

Furthermore, a reduction playing an (n,n) game can exploit an adversary playing a (n,κ) game by guessing in advance the set $\mathtt{I}$ of targets that the adversary will return. A correct guess allows the reduction to simulate the remaining keys without being noticed (see Suppl. Mat. 1.5 for the proof).

Theorem 4. *There is an SFBB reduction $\mathbb{B}$ such that, for every adversary $\mathbb{A}$,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}^\star}_{\mathrm{PKE}}(\mathbb{A}) \leq \binom{\kappa}{n} \cdot \mathsf{Adv}^{(n,n)\text{-ind-cca}}_{\mathrm{PKE}}(\mathbb{B})\,.$$

$\mathbb{B}$*'s overhead consists of generating* $\kappa - n$ *fresh keypairs, sampling* $\kappa - n$ *bits, and choosing a subset of* $[\kappa]$ *of cardinality* n *uniformly at random.*

Composing Thm. 3 and 4, we obtain the following bound.

Corollary 2 (ROR $\implies$ IND)**.** *There is an SFBB reduction* $\mathbb{B}$ *such that, for any adversary* $\mathbb{B}$*,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{A}) \leq \binom{\kappa}{n} \cdot 2^n \cdot \mathsf{Adv}^{(n,n)\text{-ror-cca}}_{\mathrm{PKE}}(\mathbb{B})\,.$$

$\mathbb{B}$*'s overhead consists of generating* $\kappa - n$ *fresh keypairs, sampling* κ *bits, and choosing a subset of* $[\kappa]$ *of cardinality* n *uniformly at random.*

An alternative bound losing a factor 2^κ is possible by combining Thm. 3 with Lemma 2, however a simple analysis shows that whenever $n < \kappa/5$ the corollary above is preferable.

4 Inheriting Multi-Instance Security

4.1 TagKEM: Definition and Notion of Security

Our goal is to turn the AGK multi-instance secure KEM into a PKE. Yet, for the construction of hybrid encryption, the more general TagKEMs [2], where encapsulation is split into two algorithms ($\mathsf{TKEM.Key}$ and $\mathsf{TKEM.Enc}$) have proven more powerful. In Def. 4 we introduce a further generalization, called TagXEM, by allowing extendable output lengths for the ephemeral keys produced by the TagXEM.

Definition 4 (TagXEM). *A TagXEM is a tuple of algorithms* $\{\mathsf{TXEM.Kg}, \mathsf{TXEM.Key}, \mathsf{TXEM.Enc}, \mathsf{TXEM.Dec}, \mathsf{TXEM.Check}\}$*, where long-term key generation* $\mathsf{TXEM.Kg}$ *on input* pm *outputs a keypair* $(\mathsf{pk}, \mathsf{sk})$*; ephemeral key generation* $\mathsf{TXEM.Key}$*, on input* pk *and* $\ell \in \mathbb{Z}_{>0}$*, outputs an ephemeral key* $K \in \{0,1\}^\ell$ *and an internal state* σ*, subsequently encapsulation* $\mathsf{TXEM.Enc}$*, on input a state* σ *and a tag* $\tau \in \mathcal{T}$*, outputs an encapsulation* c*, or a special symbol* $\perp$ *denoting failure.* $\mathsf{TXEM.Dec}$ *takes input a private key* sk*, an encapsulation* c*, a tag* τ*, and a length* ℓ*, and outputs either a key* $K \in \{0,1\}^\ell$ *or* $\perp$ *to denote failure. Finally,* $\mathsf{TXEM.Check}$ *takes as input the system parameters* pm *as well as a purported public/private key pair* $(\mathsf{pk}, \mathsf{sk})$ *and returns* true *or* false*.*

If we restrict to a single value ℓ, the usual notion of TagKEMs appears; moreover if we restrict to a single value of τ, the $\mathsf{TXEM.Key}$ and $\mathsf{TXEM.Enc}$ algorithms can be merged into a single key encapsulation mechanism, leading to normal KEMs (or XEMs if the variable output length is still incorporated). Consequently, the correctness and security definitions for the more general TagXEMs, as discussed throughout this section, imply corresponding definitions for KEM, XEM, and TagKEM.

Experiment $\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\text{TXEM}}(\mathbb{A})$

$(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \leftarrow\!\$\ \mathsf{TXEM.Kg}$
$b_1, \ldots, b_\kappa \leftarrow\!\$\ \{0,1\}$
$(\mathtt{I}, \hat{b}) \leftarrow\!\$\ \mathbb{A}^{\mathcal{C},\mathcal{E},\mathcal{D},\mathcal{K},\mathcal{B}}(\mathsf{pk}_1, \ldots, \mathsf{pk}_\kappa)$
if $|\mathtt{I}| \neq n \vee \mathtt{I} \cap (\mathtt{K} \cup \mathtt{B}) \neq \emptyset$ **then** $\hat{b} \leftarrow 0$
return $\oplus_{i \in \mathtt{I}}\, b_i = \hat{b}$

Oracle $\mathcal{D}(i, \langle c, \tau \rangle, \ell)$

$K \leftarrow \mathsf{TXEM.Dec}_{\mathsf{sk}_i}(c, \tau, \ell)$
if $\mathtt{P}_i(c,\tau) \neq \emptyset$
  $K' \leftarrow \mathtt{P}_i(c,\tau), \ell' \leftarrow \min\{\ell, |K'|\}$
else
  $K' \leftarrow \varepsilon, \ell' \leftarrow 0$
if $\langle c, \tau \rangle \in \mathtt{C}_i \wedge K[\![\ell']\!] = K'[\![\ell']\!]$
  return $\lightning$
return K

Oracle $\mathcal{C}(i, \ell)$

$(K_0, \sigma) \leftarrow\!\$\ \mathsf{TXEM.Key}_{\mathsf{pk}_i}(\ell)$
$\mathtt{E}_i \xleftarrow{\frown} \langle \sigma, K_0 \rangle$
$K_1 \leftarrow\!\$\ \{0,1\}^\ell$
return K_{b_i}

Oracle $\mathcal{E}(i, j, \tau)$

if $\mathtt{E}_i[j] = \emptyset$ **then return** $\lightning$
$\langle \sigma, K \rangle \leftarrow \mathtt{E}_i[j], \mathtt{E}_i[j] \leftarrow \emptyset$
$c \leftarrow\!\$\ \mathsf{TXEM.Enc}(\sigma, \tau)$
$\mathtt{P}_i(c,\tau) \leftarrow K$
$\mathtt{C}_i \xleftarrow{\cup} \langle c, \tau \rangle$
return c

Oracle $\mathcal{K}(i)$

$\mathtt{K} \xleftarrow{\cup} i$
return sk_i

Oracle $\mathcal{B}(i)$

$\mathtt{B} \xleftarrow{\cup} i$
return b_i

Fig. 6. Multi-instance indistinguishability notion for TXEM. In blue the same strengthening as in Fig. 4 in the case of imperfect correctness, with a slightly more complex admin to accomodate tags and length extension. We take $K[\![\ell]\!]$ to mean the first ℓ bits of K and ε as the empty string.

For correctness, we allow the effective tag space $\mathcal{T}_\ell$ to depend on the length ℓ of the ephemeral key. Similarly to Def. 1, we define (γ, δ)-correctness for TagXEM. To ensure correctness for all τ, including those that depend on K, τ's quantifier sits inside the probability statement.

Definition 5 ((γ,δ)-Correctness TagXEM). *Let $\gamma, \delta \in [0,1]$. Then a tag extendable-output key encapsulation mechanism* TXEM *is called (γ,δ)-correct iff*

1. $\Pr[(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{TXEM.Kg}(\mathsf{pm}) : \mathsf{TXEM.Check}(\mathsf{pm}, \mathsf{pk}, \mathsf{sk}) = \mathsf{false}] \leq \gamma$;
2. *if* $\mathsf{TXEM.Check}(\mathsf{pm}, \mathsf{pk}, \mathsf{sk}) = \mathsf{true}$ *then for all $\ell \in \mathbb{Z}_{>0}$ it holds that*

$$\Pr\left[(K,\sigma) \leftarrow\!\$\ \mathsf{TXEM.Key}_{\mathsf{pk}}(\ell) \ :\ \exists \tau \in \mathcal{T}_\ell \ \mathbf{s.th.}\ \begin{array}{l} c \leftarrow \mathsf{TXEM.Enc}(\sigma,\tau) \\ \mathsf{TXEM.Dec}_{\mathsf{sk}}(c,\tau,\ell) \neq K \end{array}\right] \leq \delta\ .$$

For security, Abe et al.'s notion of TagKEM indistinguishability [2] transfers easily to the multi-instance setting. The relevant game is given in Fig. 6, where we also made the necessary changes to deal with the variable output length of TagXEMs, plus the strengthening of $\mathcal{D}$ in the case of imperfect correctness (cf. Sect. 3.2).

$\mathsf{TXEM.Key}_{\mathsf{pk}}(\ell)$

$(K^{\mathrm{kem}}, \sigma) \leftarrow\!\!\$\ \mathsf{TKEM.Key}_{\mathsf{pk}}$
$K^{\mathrm{xem}} \leftarrow F(K^{\mathrm{kem}}, \ell)$
$\sigma' \leftarrow \langle \sigma, \ell \rangle$
return $(K^{\mathrm{xem}}, \sigma')$

$\mathsf{TXEM.Enc}(\sigma', \tau)$

$\langle \sigma, \ell \rangle \leftarrow \sigma'$
if $\tau \notin \mathcal{T}_\ell$:
  return $\perp$
$c \leftarrow \mathsf{TKEM.Enc}(\sigma, \tau)$
return c

$\mathsf{TXEM.Dec}(c, \tau, \ell)$

if $\tau \notin \mathcal{T}_\ell$:
  return $\perp_{\mathrm{TAG}}$
$K^{\mathrm{kem}} \leftarrow \mathsf{TKEM.Dec}(c, \tau)$
if $K^{\mathrm{kem}} = \perp$:
  return $\perp_{\mathrm{KEM}}$
$K^{\mathrm{xem}} \leftarrow F(K^{\mathrm{kem}}, \ell)$
return K^{xem}

Fig. 7. A TagXEM TXEM from a TagKEM TKEM with keyspace $\{0,1\}^k$ and a XOF with seed space $\mathcal{X} = \{0,1\}^k$. The key generation algorithm $\mathsf{TXEM.Kg}$ is unchanged from $\mathsf{TKEM.Kg}$.

Definition 6. *Let* TXEM *be a TagXEM. Then the xor-indistinguishability advantage of an adversary* $\mathbb{A}$ *is*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}^\star}_{\mathrm{TXEM}}(\mathbb{A}) = 2 \cdot \Pr\Big[\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}^\star}_{\mathrm{TXEM}}(\mathbb{A}) = 1\Big] - 1\,,$$

where the experiment is defined in Fig. 6.

If we fix ℓ and set $\mathcal{T}_\ell$ to a single element, the notion captures multi-instance security for standard KEMs, which is near equivalent (see Sect. 3.3) the notion that AGK used. In other words, provided MI-gapCDH is hard, their construction achieves (n, κ)-IND-CCA security in the random oracle model, but only for fixed ℓ and trivial $\mathcal{T}_\ell$ [4, Thm. 2].

4.2 Extending the Output of a TagKEM

First, we show how combining a TagKEM with a fixed output length and a suitable pseudorandom extendable output function (XOF), yields a TagXEM that inherits the MI security of the underlying KEM. Recall that a XOF, for instance SHAKE128 and SHAKE256 as standardized by NIST [31], is a function $F : \mathcal{X} \times \mathbb{Z}_{>0} \to \{0,1\}^*$ for some finite domain $\mathcal{X}$ that on input input a seed $s \in \mathcal{X}$ and a desired output length ℓ, outputs a value $y \in \{0,1\}^\ell$. Moreover, if $\ell < \ell'$, then $F(s, \ell)$ is a prefix of $F(s, \ell')$ for all s. This prefix preservation is not a requirement of our constructions; rather we model the property to ensure SHAKE128 and SHAKE256 are suitable real-world instantiations.

As security notion for a XOF F we use its multi-challenge pseudorandomness, which is a standard distinguishing advantage $\mathsf{Adv}^{\mathrm{psrnd}}_F(\mathbb{A})$: an adversary needs to distinguish between either a real oracle that, on input a desired length ℓ, samples a seed $s \leftarrow\!\!\$\ \mathcal{X}$ uniformly at random and returns $F(s, \ell)$, or an ideal oracle that, on input said ℓ, simply returns a uniformly sampled string of length ℓ.

The construction of the TagXEM is given in Fig. 7 and the security claim follows in Thm. 5 (see Suppl. Mat. 1.6 for the proof). If the PsRND advantage

$\mathsf{PKE.Enc}_{\mathsf{pk}}(m)$

$(K, c_1) \leftarrow\!\!\$\ \mathsf{XEM.Enc}_{\mathsf{pk}}(|m|)$
$c_2 \leftarrow K \oplus m$
return $\langle c_1, c_2 \rangle$

$\mathsf{PKE.Dec}_{\mathsf{sk}}(\langle c_1, c_2 \rangle)$

$K \leftarrow \mathsf{XEM.Dec}_{\mathsf{sk}}(c_1, |c_2|)$
if $K = \perp$ **then return** $\perp$
$m \leftarrow K \oplus c_2$
return m

$\mathsf{PKE'.Enc}_{\mathsf{pk}}(m)$

$(K, \sigma) \leftarrow\!\!\$\ \mathsf{TXEM.Key}_{\mathsf{pk}}(|m|)$
$c_2 \leftarrow K \oplus m$
$c_1 \leftarrow \mathsf{TXEM.Enc}(\sigma, c_2)$
return $\langle c_1, c_2 \rangle$

$\mathsf{PKE'.Dec}_{\mathsf{sk}}(\langle c_1, c_2 \rangle)$

$K \leftarrow \mathsf{TXEM.Dec}_{\mathsf{sk}}(c_1, c_2, |c_2|)$
if $K = \perp$ **then return** $\perp$
$m \leftarrow K \oplus c_2$
return m

Fig. 8. Two hybrid encryption schemes: PKE (top row) is a conventional hybrid scheme combining a XEM with the OTP to yield a CPA-secure PKE, while PKE′ (bottom row) combines a TagXEM with the OTP to yield a CCA-secure PKE. The key generation and checking algorithms are equivalent to their XEM resp. TXEM counterparts.

of F is sufficiently small, then TXEM inherits the multi-instance security of TKEM; moreover, as the result holds for arbitrary $\mathcal{T}$ and $\mathcal{T}_\ell$, it holds for the trivial spaces, yielding a slightly simpler XEM from KEM result.

One concern is whether the PsRND advantage of F will be sufficiently small. Suppose k is the output length of the underlying TagKEM. A generic attacker would always be able to fix $\ell > k$ and evaluate F for, say, N seeds offline in the hope of colliding with any of the challenge evaluations. The distinguishing advantage of such an adversary is of order $(q_c + q_d)N/2^k$, indicating that the underlying TagKEM already needs to provide keys long enough to yield the desired multi-instance security.

Theorem 5. *Let* TKEM *be a* (γ, δ)*-correct TagKEM sampling keys from* $\{0,1\}^k$ *and with tagspace* $\mathcal{T}$*, let* $F : \{0,1\}^k \times \mathbb{Z}_{>0} \to \{0,1\}^*$ *be a XOF, and let* TXEM *be a TagXEM as given in Fig. 7 for arbitrary* $\mathcal{T}_\ell \subseteq \mathcal{T}$*. Then* TXEM *is* (γ, δ)*-correct, and there are SFBB reductions* $\mathbb{B}$ *and* $\mathbb{C}$ *such that, for every adversary* $\mathbb{A}$*,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{TXEM}}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{TKEM}}(\mathbb{B}) + 2 \cdot \mathsf{Adv}^{\mathrm{psrnd}}_{F}(\mathbb{C})\,.$$

If $\mathbb{A}$ *calls* $\mathcal{C}$ q_c *times and* $\mathcal{D}$ q_d *times, then* $\mathbb{B}$*'s overhead consists of at most* $q_c + q_d$ *evaluations of* F*, while* $\mathbb{C}$*'s overhead consists of doing* κ *executions of* $\mathsf{TKEM.Kg}$*, at most* q_c *executions of* $\mathsf{TKEM.Key}$ *and* $\mathsf{TKEM.Enc}$*, and at most* q_d *executions of* $\mathsf{TKEM.Dec}$*.*

4.3 A PKE Inheriting (Tag)XEM Security

As a multi-instance secure XEM provides us with ephemeral keys of any desired length, we can combine it with an information-theoretic DEM in order to

achieve PKE. Here we opt for the one-time-pad (OTP), as it is the simplest and best-known primitive providing perfect secrecy. The beauty of the OTP is that whether you switch out the ephemeral key for a uniform random one, or the message for a uniform random one, the resulting ciphertext distribution is the same. It allows the PKE to tightly inherit the MI-security of the XEM, albeit yielding only real-or-random security under chosen-plaintext attacks. The construction is provided in full in Fig. 8 (top row); the security claim is captured in Thm. 6 (see Suppl. Mat. 1.7 for the proof).

Theorem 6 (ROR-CPA PKE). *Let* XEM *be a* (γ, δ)*-correct XEM, and let* PKE *be a hybrid encryption scheme as given in Fig. 8. Then* PKE *is* (γ, δ)*-correct, and there is a type-preserving SFBB reduction* $\mathbb{B}$ *such that for every adversary* $\mathbb{A}$*,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ror-cpa}\star}_{\mathrm{PKE}}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-ind-cpa}\star}_{\mathrm{XEM}}(\mathbb{B})\,.$$

One might hope that adding information-theoretic MACs to the DEM would result in the inheritance of CCA security, but that is easier said than shown. For instance, the usual proof technique of a game hop where all decryption queries are disallowed does not work: after breaking only a single KEM private key, the reduction will be found out as not being faithful. Sadly, a single-instance break (of the reduction) suffices to show that that reduction cannot demonstrate multi-instance security.

Luckily, TagKEMs allow for a modified hybrid scheme for which the DEM no longer needs to satisfy CCA security for the resulting PKE to be guaranteed CCA-secure: in the single-instance setting, if the TagKEM is CCA-secure, then so is the PKE [2]. We upgrade the construction to use TagXEMs and the OTP in Fig. 8 (bottom row) and show its multi-instance inheritance in Thm. 7 (see Suppl. Mat. 1.8 for the proof).

Theorem 7 (ROR-CCA PKE). *Let* TXEM *be a* (γ, δ)*-correct TagXEM, and let* PKE' *be a hybrid encryption scheme as given in Fig. 8. Then* PKE' *is* (γ, δ)*-correct, and there is a type-preserving SFBB reduction* $\mathbb{B}$ *such that for every adversary* $\mathbb{A}$*,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ror-cca}\star}_{\mathrm{PKE}'}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{TXEM}}(\mathbb{B})\,.$$

While encouraging, the claim that the constructed PKE inherits the multi-instance security of the TagXEM is dampened by the exponential separation between the ROR security notion and IND, as argued in Sect. 3.4. Indeed, extrapolating to the latter notion by combining Thm. 7 with Cor. 2, we have only achieved the following bound.

Corollary 3. *Let* TXEM *be a* (γ, δ)*-correct TagXEM, and let* PKE' *be a hybrid encryption scheme as given in Fig. 8. Then* PKE' *is* (γ, δ)*-correct, and there is a type-preserving SFBB reduction* $\mathbb{B}$ *such that for every adversary* $\mathbb{A}$*,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}'}(\mathbb{A}) \leq \binom{\kappa}{n} \cdot 2^n \cdot \mathsf{Adv}^{(n,n)\text{-ind-cca}}_{\mathrm{TXEM}}(\mathbb{B})\,,$$

where $\mathbb{B}$*'s overhead is dominated by generating* $\kappa - n$ *fresh keypairs, sampling* κ *bits, and choosing a subset of* $[\kappa]$ *of cardinality* n *uniformly at random.*

Oracle $\mathcal{C}_{\mathrm{ROP}}(i, \ell, \Pi)$

$(K_0, \sigma) \leftarrow\!\!\$\ \mathsf{TXEM.Key}_{\mathsf{pk}_i}(\ell)$

$\mathtt{E}_i \xleftarrow{\frown} \sigma$

$K_1 \leftarrow \Pi(K_0)$

return K_{b_i}

Fig. 9. Fig. 6 is upgraded to $\mathsf{Exp}^{(n,\kappa)\text{-rop-cca}\star}_{\mathrm{TXEM}}$ by letting $\mathcal{C}_{\mathrm{ROP}}$ replace $\mathcal{C}$.

4.4 Real-Or-Permuted: A Strengthened Notion for KEM Security

If we want to achieve an IND-CCA PKE more tightly, we seem to need a different notion of security for our TagXEMs. What could such a notion look like?

Our solution is a novel, stronger KEM notion, which we will refer to as "real-or-permuted", or ROP for short. Fig. 9 provides the crucial new challenge oracle. The adversary has to guess whether a tentative K is the one encapsulated under c, or whether an adaptively chosen permutation has been applied to it. As permutations preserve the distribution of the sampling space, there are no choices of Π that make the game generically and trivially winnable.

Technically, we need to specify how the adversary provides Π such that it is guaranteed, or can be checked, to be a permutation. Hence, formally we define ROP with respect to a class of permutations $\mathcal{P}$, reminiscent of for instance key-dependent message [22] or related-key attack [9] definitions. We require that membership $\Pi \in \mathcal{P}$ is easy to check (e.g. ROP can simply index an element in $\mathcal{P}$) and that, by definition, $\mathcal{P}$ can be verified to indeed only contain permutations. For our main results, it suffices if $\mathcal{P}$ is the class of one-time pads, in the sense that Π specifies the key (or pad) of the one-time pad enciphering. Henceforth, we will assume that ROP is defined with respect to that class, unless explicitly stated otherwise.

The new notion ROP and IND relate to each other much the same way as IND and ROR for PKE. It is not hard to see that ROP tightly implies IND, whereas the other direction seems to incur the same loss as the ROR-to-IND implication for PKE (see Suppl. Mat. 2). For completeness, ROP lends itself equally well to XEMs and KEMs, or notions without corruptions or a decryption oracle. Finally, if any of the above primitives are constructed using an IND-secure PKE (e.g. using a Fujisaki–Okamoto style transform [19, 20, 26]), then achieving ROP is as easy as achieving IND: simply let K be the "left" message, and $\Pi(K)$ be the "right"!

4.5 PKE′ Tightly Inherits IND-CCA Security

Using ROP in place of IND, we are able to show directly that the PKE constructions of Fig. 8 are IND-CPA resp. IND-CCA secure, by (as before) giving a (Tag)XEM reduction that provides a perfect simulation for the PKE adversary.

$\underline{\mathsf{TXEM.Kg}}$

$(\mathsf{pk}', \mathsf{sk}') \leftarrow\!\!\$\ \mathsf{KEM.Kg}$
$\mathsf{pk} \leftarrow \mathsf{pk}'$
$\mathsf{sk} \leftarrow \langle \mathsf{pk}', \mathsf{sk}' \rangle$
return $(\mathsf{pk}, \mathsf{sk})$

$\underline{\mathsf{TXEM.Check}(\mathsf{pk}, \mathsf{sk})}$

$\langle \mathsf{pk}', \mathsf{sk}' \rangle \leftarrow \mathsf{sk}$
if $\mathsf{pk} \neq \mathsf{pk}'$ **then return** 0
return $\mathsf{KEM.Check}(\mathsf{pk}', \mathsf{sk}')$

$\underline{\mathsf{TXEM.Enc}(\sigma, \tau)}$

$\langle c, K^{\mathrm{mac}} \rangle \leftarrow \sigma$
$\mathsf{mac} \leftarrow \mathsf{MAC}_{K^{\mathrm{mac}}}(\tau)$
return $\langle c, \mathsf{mac} \rangle$

$\underline{\mathsf{TXEM.Key}_{\mathsf{pk}}(\ell)}$

$(K^{\mathrm{kem}}, c) \leftarrow\!\!\$\ \mathsf{KEM.Enc}_{\mathsf{pk}}$
$\ell' \leftarrow \ell + \ell_{\mathsf{mackey}}$
$K^{\mathrm{mac}} \| K^{\mathrm{xem}} \leftarrow F(\mathsf{pk}, c, K^{\mathrm{kem}}, \ell')$
$\sigma \leftarrow \langle c, K^{\mathrm{mac}} \rangle$
return K^{xem}

$\underline{\mathsf{TXEM.Dec}_{\mathsf{sk}}(\langle c, \mathsf{mac} \rangle, \tau, \ell)}$

$\langle \mathsf{pk}', \mathsf{sk}' \rangle \leftarrow \mathsf{sk}$
$K^{\mathrm{kem}} \leftarrow \mathsf{KEM.Dec}_{\mathsf{sk}'}(c)$
if $K^{\mathrm{kem}} = \perp$ **then return** $\perp$
$\ell' \leftarrow \ell + \ell_{\mathsf{mackey}}$
$K^{\mathrm{mac}} \| K^{\mathrm{xem}} \leftarrow F(\mathsf{pk}', c, K^{\mathrm{kem}}, \ell')$
if $\mathsf{MAC}_{K^{\mathrm{mac}}}(\tau) \neq \mathsf{mac}$ **then return** $\perp$
return K^{xem}

Fig. 10. A TagXEM from a KEM, a MAC, and an XOF F.

The crucial observation is that for any pair of messages $m_0, m_1 \in \{0,1\}^\ell$, there exist a permutation $\Pi_{m_0 \to m_1}$ on $\{0,1\}^\ell$ such that the message encapsulations are related as $K \oplus m_1 = \Pi_{m_0 \to m_1}(K) \oplus m_0$. Namely, the permutation that on input K, outputs $m_0 \oplus m_1 \oplus K$ (see Suppl. Mat. 1.9 for the proof).

Theorem 8 (IND-CCA PKE). *Let* TXEM *be a (γ, δ)-correct TagXEM, and let* PKE′ *be a hybrid encryption scheme as given in Fig. 8. Then* PKE′ *is (γ, δ)-correct, and there is a type-preserving SFBB reduction $\mathbb{B}$ such that for every adversary $\mathbb{A}$,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}'}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-rop-cca}\star}_{\mathrm{TXEM}}(\mathbb{B})\,.$$

We leave it to the reader to verify that as before, employing a ROP-CPA XEM in place of the TagXEM yields IND-CPA security for the PKE of Fig. 8 (top row), by adapting the proof of Thm. 6 to the above. We again stress that using an information-theoretically CCA-secure DEM together with a CCA XEM does not seem to yield a proof of CCA inheritance to the PKE (see Sect. 4.3).

4.6 TagXEM from a KEM, a MAC, and a Random Oracle

With Thm. 8, we achieved what we set out to do: demonstrating tight MI inheritance from a TagXEM to an IND-CCA PKE. However, AGK only showed how to construct an IND-CCA KEM, providing a reduction to the MI-GapCDH assumption in the programmable random oracle model. Without the crucial support of tags, our construction only achieves CPA security. Furthermore, Thm. 5

does *not* easily transfer to the ROP setting: it is not clear how to combine a ROP-CCA KEM with a XOF to yield a ROP-CCA XEM.

We complete the picture by providing a TagXEM construction from a KEM, a MAC, and a XOF. Our construction (Fig. 10) is inspired by Abe et al.'s TagKEM construction [2] and we show that with an information-theoretic MAC, if the KEM is perfectly correct, has unique encapsulations [23] and is multi-instance one-way secure under plaintext-checking attacks (OW-PCA), then the TagXEM is ROP-CCA secure in the programmable random oracle model (to model the XOF). Before stating our concrete security result (Thm. 9), we will define the relevant concepts and advantages below.

One-wayness for KEMs tasks an adversary to retrieve the ephemeral key that has been encapsulated, given the public key and the encapsulation. In the multi-instance setting, an adversary has access to many public keys and various encapsulations per public key and endeavours to find ephemeral keys for encapsulations for as many different public keys as possible (no reward for breaking multiple encapsulations under the same public key).

Plaintext-checking attacks (PCA) were introduced by Okamoto and Pointcheval [32, Definition 8] in a single-user public key encryption setting. Intuitively, PCA provides the adversary access to an oracle that, on input a pair (m, c) determines whether c encrypts m or not; more formally [1], the oracle checks whether c decrypts to m or not. In the context of KEMs, the PCA oracle takes a pair (K^{kem}, c) as input and determines whether c decapsulates to K^{kem} or not. The multi-user or multi-instance generalization is straightforward and the definition (in its modern decryption incarnation) inherently deals with imperfect correctness in the decryption.

Definition 7 considers one-wayness under plaintext checking attacks. For standard ElGamal KEM, where a (multiplicative) discrete-log group with generator g and of prime order q is given as part of the parameters, a public key consists of $h = g^x$ with $x \leftarrow\!\!\$\ \mathbb{Z}_q$ the private key, and an encapsulation outputs $(K^{\text{kem}}, c) = (h^r, g^r)$ for random $r \leftarrow\!\!\$\ \mathbb{Z}_q$, the one-wayness problem (in the single-user case) is equivalent to the computational Diffie–Hellman (CDH) problem. The plaintext checking oracle allows an adversary to learn, for group elements (k, c) of its choice, whether $k = c^x$ or not. The corresponding hardness assumption for OW-PCA is known as the Strong CDH assumption. An even stronger assumption is the GapCDH assumption, where an adversary instead can use an oracle that determines whether a quadruple of group elements is a Diffie–Hellman tuple or not.

Definition 7 (OW-PCA**).** *Let* KEM *be a key encapsulation mechanism. Then the one-way advantage under plaintext-checking attacks of an adversary* $\mathbb{A}$ *is*

$$\mathsf{Adv}^{(n,\kappa)\text{-ow-pca}\star}_{\text{KEM}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-ow-pca}\star}_{\text{KEM}}(\mathbb{A}) = 1\right],$$

where the experiment is defined in Fig. 11.

In addition to perfect correctness and OW-PCA security, the security reduction for our construction (Thm. 9) relies on two further properties of the

$\underline{\mathsf{Exp}^{(n,\kappa)\text{-ow-pca}\star}_{\mathrm{KEM}}(\mathbb{A})}$

$(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \leftarrow\!\!\$\ \mathsf{KEM.Kg}$

$(\mathtt{I}, (j_i, \hat{K}_i)_{i \in \mathtt{I}}) \leftarrow\!\!\$\ \mathbb{A}^{\mathcal{E}, \mathcal{P}, \mathcal{K}}(\mathsf{pk}_1, \ldots, \mathsf{pk}_\kappa)$

if $|\mathtt{I}| \neq n \vee \mathtt{I} \cap \mathtt{K} \neq \emptyset$ **then return** 0

return $\bigwedge_{i \in \mathtt{I}} \mathtt{P}_i[j_i] = \hat{K}_i$

$\underline{\mathcal{P}(i, c, K)}$

$K' \leftarrow \mathsf{KEM.Dec}_{\mathsf{sk}_i}(c)$

return $K = K'$

$\underline{\mathcal{E}(i)}$

$(K, c) \leftarrow\!\!\$\ \mathsf{KEM.Enc}_{\mathsf{pk}_i}$

$\mathtt{P}_i \xleftarrow{\frown} K$

return c

$\underline{\mathcal{K}(i)}$

$\mathtt{K} \xleftarrow{\cup} i$

return sk_i

Fig. 11. Multi-instance one-way security in the presence of plaintext checking attacks.

underlying KEM. Unique encapsulation captures that for a fixed public key and ephemeral key, the encapsulation corresponding to that ephemeral key is unique (without saying anything about how to compute it). Unique encapsulations have been used before, for instance by Heuer et al. [23] (see also Remark 4 below).

Definition 8 (Unique Encapsulation). *Let* KEM *be a perfectly correct KEM. Then it has unique encapsulations iff*

$$\Pr\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KEM.Kg} \\ (K_0^{kem}, c_0) \leftarrow\!\!\$\ \mathsf{KEM.Enc}_{\mathsf{pk}} \\ (K_1^{kem}, c_1) \leftarrow\!\!\$\ \mathsf{KEM.Enc}_{\mathsf{pk}}\end{array} : K_0^{kem} = K_1^{kem} \wedge c_0 \neq c_1\right] = 0\ .$$

The second additional property we require from the KEM is that collisions amongst encapsulations (under a single randomly drawn public key) are suitably rare. Def. 9 captures the relevant probability of a k-way encapsulation collision. If a KEM is perfectly correct with unique encapsulations, then colliding encapsulations are equivalent to colliding ephemeral keys; if, as is usually the case, these ephemeral keys are furthermore chosen uniformly at random from a finite set $\mathcal{X}$, we can upper bound $\epsilon_k(q)$ by $q^k/|\mathcal{X}|^{k-1}$ using a standard bound on k-way collisions (see e.g. [33, Appendix B]).

Definition 9 (Encapsulation Multi-Collisions). *Let* KEM *be a KEM, and let $q, k \in \mathbb{Z}_{>1}$ be parameters. Then the k-out-of-q encapsulation multi-collision probability is*

$$\epsilon_k(q) = \Pr\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KEM.Kg} \\ \forall_{i \in [q]} (K_i^{kem}, c_i) \leftarrow\!\!\$\ \mathsf{KEM.Enc}_{\mathsf{pk}}\end{array} : \exists_{\mathtt{J} \subseteq [q], |\mathtt{J}| = k} \forall_{i,j \in \mathtt{J}} c_i = c_j\right]\ .$$

For completeness, we also present definitions of a deterministic message authentication code, so we dispense with an explicit verification algorithm in Def. 10 (for concreteness, we restrict to bitstrings for both keys and tags, of length ℓ_{mackey} and ℓ_{mac} respectively), and an information-theoretic notion of forgeries (Def. 11) where we use the same parameter k as above (or rather

$k-1$ in Thm. 9), but this time to denote the number of valid message–tag pairs available to an adversary. The usual choice is $k=1$, e.g. when considering strongly universal$_2$ hash functions, but Wegman and Carter [36] already investigated $k>1$. Provided ℓ_{mackey} is large enough (at least $k\cdot\ell_{\mathsf{mac}}$), one can achieve $\hat{\epsilon}_k = 2^{-\ell_{\mathsf{mac}}}$, which is optimal.

Definition 10 (Message Authentication Code (MAC)). *A message authentication code* MAC *is a pair of algorithms* MAC.Kg *and* MAC.Mac*, where* MAC.Kg *randomly generates a* $K^{mac} \in \{0,1\}^{\ell_{\mathsf{mackey}}}$*,* MAC.Mac *takes a key* K^{mac} *and a message* $m \in \mathcal{M}$ *to output tag* $\mathsf{mac} \leftarrow \mathsf{MAC.Mac}_{K^{mac}}(m) \in \{0,1\}^{\ell_{\mathsf{mac}}}$*, The MAC scheme is correct iff, for all* $K^{mac} \in \{0,1\}^{\ell_{\mathsf{mackey}}}$ *and* $m \in \mathcal{M}$*, it holds that* mac *is correct iff* $\mathsf{MAC.Mac}_{K^{mac}}(m) = \mathsf{mac}$.

Definition 11 (Information-Theoretic MAC Forgeries). *Let* MAC *be given and let* $k \in \mathbb{Z}_{\geq 0}$ *be a parameter, then the forging advantage after observing* k *valid message–tag pairs is defined as*

$$\hat{\epsilon}_k = \max_{\substack{\forall i\in\{0\}\cup[k]\\(m_i,\mathsf{mac}_i)}} \Pr\Big[\mathsf{MAC.Mac}_{K^{mac}}(m_0)=\mathsf{mac}_0 \;\Big|\; \forall_{i\in[k]}\mathsf{MAC.Mac}_{K^{mac}}(m_i)=\mathsf{mac}_i\Big]\,.$$

With all elements in place, we can state the security of Fig. 10's TXEM, in Thm. 9 (see Suppl. Mat. 1.10 for the proof). The security bound depends on a tuning parameter k that feeds into both the collision probability of the underlying KEM and the forgery advantage of the MAC. The ability to tune the bound allows some flexibility when instantiating the three underlying primitives KEM, MAC, and XOF. For instance, the KEM can be instantiated using ElGamal, whose multi-instance security can easily be linked to the MI-GapCDH assumption (with corruptions). In that case, the ephemeral key is a group element and $|\mathcal{X}|$ corresponds to the group size. For fixed q_c, increasing k will result in a smaller upper bound on $\epsilon_k(q_c)$. To ensure that $\hat{\epsilon}_{k-1}$ does not dominate, it might be necessary to increase the key size ℓ_{mackey} of the information-theoretic MAC, otherwise instantiating the information-theoretic MAC and the XOF is relatively straightforward (with the usual ROM caveats for the latter).

Theorem 9. *Let* TXEM *be as in Fig. 10, let* KEM *be a perfectly correct KEM with unique encapsulations, and let* $k \in \mathbb{Z}_{>1}$*. Then there is an SFBB reduction* $\mathbb{B}$ *such that, for all* $\mathbb{A}$ *that makes* q_c *challenge and* q_d *decryption oracle queries,*

$$\mathsf{Adv}^{(n,\kappa)\text{-rop-cca}^\star}_{\mathrm{TXEM}}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-ow-pca}^\star}_{\mathrm{KEM}}(\mathbb{B}) + 2\big(q_d\hat{\epsilon}_{k-1} + \epsilon_k(q_c)\big)$$

in the programmable random oracle model, where $\hat{\epsilon}_{k-1}$ *is the forging advantage after observing* $k-1$ *valid message–tag pairs (Def. 11) and* $\epsilon_k(q_c)$ *is the* k*-out-of-*q_c *encapsulation multi-collision probability of KEM (Def. 9). If* $\mathbb{A}$ *makes* q_f *queries to the random oracle, then* $\mathbb{B}$ *makes at most* q_f *queries to its plaintext checking oracle.*

The proof borrows some ideas already used to prove AGK's Thm. 2. In fact, it is relatively straightforward to recast AGK's Thm. 2 as the multi-instance

version of a OW-PCA KEM plus a programmable random oracle yielding an IND-CCA KEM, although the presence of the error terms $\hat{\epsilon}_{k-1}$ and especially $\epsilon_k(q_c)$ render recovery of AGK's Thm. 2 as a special case of our Thm. 9 not immediate.

Our proof does rely on perfect correctness of the underlying KEM, thus excluding many popular post-quantum KEMs based on the hardness of LWE. Having said that, establishing the post-quantum security of TXEM would require a proof in the quantum random oracle model [14]. We leave the construction of a post-quantum TagXEM as an enticing open problem.

Combining Thm. 8 and 9 in Cor. 4, we can finally conclude how to construct a multi-instance secure PKE.

Corollary 4. *Let* PKE' *be as in Fig. 8, let the underlying TagXEM be as in Fig 10, let* KEM *be a perfectly correct KEM with unique encapsulations, and let* $k \in \mathbb{Z}_{>1}$*. Then, there is an SFBB reduction* $\mathbb{B}$ *such that, for all* $\mathbb{A}$ *that makes* q_c *challenge and* q_d *decryption oracle queries,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}'}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-ow-pca}\star}_{\mathrm{KEM}}(\mathbb{B}) + 2\big(q_d\hat{\epsilon}_{k-1} + \epsilon_k(q_c)\big)$$

in the programmable random oracle model, where $\hat{\epsilon}_{k-1}$ *is the forging advantage after observing* $k-1$ *valid message–tag pairs (Def. 11) and* $\epsilon_k(q_c)$ *is the* k*-out-of-*q_c *encapsulation multi-collision probability of KEM (Def. 9). If* $\mathbb{A}$ *makes* q_f *queries to the random oracle, then* $\mathbb{B}$ *makes at most* q_f *queries to its plaintext checking oracle.*

Remark 4. The resulting construction is remarkably similar to the PKE studied by Heuer et al. [23] in the context of selective opening attacks (and to a lesser extent its predecessor by Steinfeld et al. [35] and successor by Lai et al. [28]). They too use a random oracle to derive a MAC key and a one-time pad from an ephemeral KEM key. The only two differences are that Heuer et al. do not consider arbitrary length messages and that their random oracle outputs $K^{\mathrm{xem}} \| K^{\mathrm{mac}}$, i.e. the opposite order from what we do.

For fixed length messages, the order in which those two keys are output does not matter. However, when moving to arbitrary length-messages, the order of the XOF output does matter. Outputting $K^{\mathrm{xem}} \| K^{\mathrm{mac}}$ instead would allow a length extension attack enabling the adversary to recover the MAC key, at which point producing forgeries would be trivial.

In a way, the construction is quite brittle that these small details matter. Another example of brittleness is that our reduction for Theorem 9 requires $\perp$ produced from a KEM decryption error to be indistinguishable from a failed MAC verification. In implementations, a timing attack might well break this requirement.

Corollary 5. *Let* PKE' *be as in Fig. 8, let the underlying TagXEM be as in Fig 10, let* KEM *be a perfectly correct KEM with unique encapsulations based on the low granularity MI-GapCDH as explained in Suppl. Mat. 3, and let* $k \in \mathbb{Z}_{>1}$*. Then, there is an SFBB reduction* $\mathbb{B}$ *such that, for all* $\mathbb{A}$ *that makes* q_c *challenge*

and q_d decryption oracle queries and a total of q queries to the group-operation oracles,

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}'}(\mathbb{A}) \leq \left(\frac{4 \cdot e \cdot q^2}{n^2 \cdot p}\right)^n + 2\big(q_d \hat{\epsilon}_{k-1} + \epsilon_k(q_c)\big)$$

in the programmable random oracle model, where p is the minimal size of the generated generic groups, and at least n oracles receives at least $60 \log_2 p$ and $\sqrt{q_f}/2$ queries, $\hat{\epsilon}_{k-1}$ is the forging advantage after observing $k-1$ valid message–tag pairs (Def. 11) and $\epsilon_k(q_c)$ is the k-out-of-q_c encapsulation multi-collision probability of KEM (Def. 9). If $\mathbb{A}$ makes q_f queries to the random oracle, then $\mathbb{B}$ makes at most q_f queries to its $\mathcal{DDH}$ oracle. If $\mathbb{A}$ calls its challenge oracle q_c times, then $\mathbb{B}$ draws q_c uniformly random group elements.

References

1. Abdalla, M., Benhamouda, F., Pointcheval, D.: Public-key encryption indistinguishable under plaintext-checkable attacks. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 332–352. Springer, Heidelberg (Mar / Apr 2015). https://doi.org/10.1007/978-3-662-46447-2_15
2. Abe, M., Gennaro, R., Kurosawa, K.: Tag-KEM/DEM: A new framework for hybrid encryption. Journal of Cryptology **21**(1), 97–130 (Jan 2008). https://doi.org/10.1007/s00145-007-9010-x
3. Auerbach, B., Giacon, F., Kiltz, E.: Everybody's a target: Scalability in public-key encryption. Cryptology ePrint Archive, Report 2019/364 (2019), https://eprint.iacr.org/2019/364
4. Auerbach, B., Giacon, F., Kiltz, E.: Everybody's a target: Scalability in public-key encryption. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 475–506. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45727-3_16
5. Baecher, P., Brzuska, C., Fischlin, M.: Notions of black-box reductions, revisited. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 296–315. Springer, Heidelberg (Dec 2013). https://doi.org/10.1007/978-3-642-42033-7_16
6. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (May 2000). https://doi.org/10.1007/3-540-45539-6_18
7. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (Aug 1998). https://doi.org/10.1007/BFb0055718
8. Bellare, M., Hofheinz, D., Kiltz, E.: Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? Journal of Cryptology **28**(1), 29–48 (Jan 2015). https://doi.org/10.1007/s00145-013-9167-4
9. Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9_31

10. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_1
11. Bellare, M., Ristenpart, T., Tessaro, S.: Multi-instance security and its application to password-based cryptography. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 312–329. Springer, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_19
12. Bellare, M., Ristenpart, T., Tessaro, S.: Multi-instance security and its application to password-based cryptography. Cryptology ePrint Archive, Report 2012/196 (2012), https://eprint.iacr.org/2012/196
13. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993). https://doi.org/10.1145/168588.168596
14. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0_3
15. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehle, D.: Crystals - kyber: A cca-secure module-lattice-based kem. In: 2018 IEEE European Symposium on Security and Privacy (EuroS P). pp. 353–367 (2018). https://doi.org/10.1109/EuroSP.2018.00032
16. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (Aug 1998). https://doi.org/10.1007/BFb0055717
17. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing **33**(1), 167–226 (2003)
18. Farshim, P., Tessaro, S.: Password hashing and preprocessing. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part II. LNCS, vol. 12697, pp. 64–91. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77886-6_3
19. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_34
20. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of Cryptology **26**(1), 80–101 (Jan 2013). https://doi.org/10.1007/s00145-011-9114-1
21. Giacon, F., Kiltz, E., Poettering, B.: Hybrid encryption in a multi-user setting, revisited. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part I. LNCS, vol. 10769, pp. 159–189. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76578-5_6
22. Halevi, S., Krawczyk, H.: Security under key-dependent inputs. In: Ning, P., De Capitani di Vimercati, S., Syverson, P.F. (eds.) ACM CCS 2007. pp. 466–475. ACM Press (Oct 2007). https://doi.org/10.1145/1315245.1315303
23. Heuer, F., Jager, T., Kiltz, E., Schäge, S.: On the selective opening security of practical public-key encryption schemes. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 27–51. Springer, Heidelberg (Mar / Apr 2015). https://doi.org/10.1007/978-3-662-46447-2_2

24. Heum, H., Stam, M.: Tightness subtleties for multi-user pke notions. In: Paterson, M.B. (ed.) Cryptography and Coding. pp. 75–104. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-92641-0_5
25. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association **58**, 13–30 (1963)
26. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_12
27. Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 117–146. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_5
28. Lai, J., Yang, R., Huang, Z., Weng, J.: Simulation-based bi-selective opening security for public key encryption. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part II. LNCS, vol. 13091, pp. 456–482. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92075-3_16
29. Lee, Y., Lee, D.H., Park, J.H.: Tightly cca-secure encryption scheme in a multi-user setting with corruptions. Des. Codes Cryptogr. **88**(11), 2433–2452 (2020)
30. Lewko, A.B., Waters, B.: Why proving HIBE systems secure is difficult. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 58–76. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_4
31. NIST: SHA-3 standard: Permutation-based hash and extendable-output functions. Federal Information Processing Standards Publication 202, NIST (Aug 2015)
32. Okamoto, T., Pointcheval, D.: REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer, Heidelberg (Apr 2001). https://doi.org/10.1007/3-540-45353-9_13
33. Preneel, B.: Analysis and Design of Cryptographic Hash Functions. Ph.D. thesis, KU Leuven (Feb 1993)
34. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (Feb 2004). https://doi.org/10.1007/978-3-540-24638-1_1
35. Steinfeld, R., Baek, J., Zheng, Y.: On the necessity of strong assumptions for the security of a class of asymmetric encryption schemes. In: Batten, L.M., Seberry, J. (eds.) ACISP 02. LNCS, vol. 2384, pp. 241–256. Springer, Heidelberg (Jul 2002). https://doi.org/10.1007/3-540-45450-0_20
36. Wegman, M.N., Carter, L.: New hash functions and their use in authentication and set equality. Journal of Computer and System Sciences **22**, 265–279 (1981)

Suppl. Mat. 1 Deferred Proofs

Suppl. Mat. 1.1 Proof of Theorem 1

Theorem 1 (MKU → UKU). *Let $0<n\leq\kappa$ be integer parameters and let* PKE *be a (γ,δ)-correct encryption scheme. Then, there is a type-preserving SFBB reduction $\mathbb{B}_{\mathrm{mku}}$, such that for every adversary $\mathbb{A}_{\mathrm{uku}}$,*

$$\mathsf{Adv}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A}_{\mathrm{uku}}) \leq \mathsf{Adv}^{(n,\kappa)\text{-mku-cca}\star}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{mku}}) + \kappa\gamma\,.$$

Proof. $\mathbb{B}_{\mathrm{mku}}$ forwards all the experiment's messages until it receives from $\mathbb{A}_{\mathrm{uku}}$ the output $(\mathtt{I},\hat{\mathsf{sk}})$. Observe that these secret keys allow $\mathbb{A}_{\mathrm{uku}}$ to win if they were generated in the game setup, $(\mathsf{pk}_i,\hat{\mathsf{sk}}_i)\leftarrow\!\$\ \mathsf{PKE.Kg}$, while in order for $\mathbb{B}_{\mathrm{mku}}$ to win, the secret keys must pass the key validation algorithm, $\mathsf{PKE.Check}(\mathsf{pm},\mathsf{pk}_i,\hat{\mathsf{sk}}_i)=\mathsf{true}$. Briefly, $\mathbb{B}_{\mathrm{mku}}$ wins if, at least, $\mathbb{A}_{\mathrm{uku}}$ wins and the secret keys pass the validation algorithm. Formally,

$$\begin{aligned}
&\Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-mku-cca}\star}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{mku}})=1\right]\\
&\quad\geq \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A})=1\wedge\bigwedge_{i\in\mathtt{I}}\mathsf{PKE.Check}(\mathsf{pm},\mathsf{pk}_i,\hat{\mathsf{sk}}_i)=\mathsf{true}\right]\\
&\quad= \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A})=1\wedge\bigwedge_{i\in\mathtt{I}}\mathsf{PKE.Check}(\mathsf{pm},\mathsf{pk}_i,\mathsf{sk}_i)=\mathsf{true}\right]
\end{aligned}$$

where, since $\mathbb{A}_{\mathrm{uku}}$ wins, it holds that $\hat{\mathsf{sk}}_i=\mathsf{sk}_i$;

$$\begin{aligned}
&\geq \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A})=1\wedge\bigwedge_{i\in[\kappa]}\mathsf{PKE.Check}(\mathsf{pm},\mathsf{pk}_i,\mathsf{sk}_i)=\mathsf{true}\right]\\
&= \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A})=1\right]\\
&\quad-\Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A})=1\wedge\bigvee_{i\in[\kappa]}\mathsf{PKE.Check}(\mathsf{pm},\mathsf{pk}_i,\mathsf{sk}_i)=\mathsf{false}\right]\\
&\geq \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A})=1\right]-\Pr\left[\bigvee_{i\in[\kappa]}\mathsf{PKE.Check}(\mathsf{pm},\mathsf{pk}_i,\mathsf{sk}_i)=\mathsf{false}\right]\\
&= \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A})=1\right]-\sum_{i\in[\kappa]}\Pr[\mathsf{PKE.Check}(\mathsf{pm},\mathsf{pk}_i,\mathsf{sk}_i)=\mathsf{false}]\\
&= \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-uku-cca}\star}_{\mathrm{PKE}}(\mathbb{A})=1\right]-\kappa\gamma\,.
\end{aligned}$$

Applying Def. 2 then yields the stated bound. □

Suppl. Mat. 1.2 Proof of Lemma 1

Lemma 1. *Let $n\leq\kappa$ and $q_c\leq\kappa-n$. Then there is an SFBB reduction $\mathbb{B}$ such that, for every adversary $\mathbb{A}$ making at most q_c corruption oracle calls,*

$$\mathsf{Adv}^{(\leq\kappa,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{B})\,,$$

where $\mathbb{B}$ makes at most $\kappa-n$ additional bit corruption oracle calls.

Proof. As both games provide the same interface to $\mathbb{A}$, $\mathbb{B}$ can simply simulate the game for $\mathbb{A}$ by forwarding every oracle call, where $\mathbb{B}$ stores the input and output for later use. Upon halting, $\mathbb{A}$ returns a guess $\hat{b}_{\mathbb{A}}$ for the value of $\oplus_{i\in[\kappa]} b_i$.

As $\mathbb{A}$ makes at most $\kappa - n$ corruption queries, there are at least n uncorrupted keys left for $\mathbb{B}$, who may select an arbitrary uncorrupted subset $\mathtt{I}$ of cardinality n. Subsequently $\mathbb{B}$ bit-corrupts all remaining instances, possibly forgoing those that $\mathbb{A}$ already bit-corrupted. Eventually, $\mathbb{B}$ sets $\hat{b}_{\mathbb{B}} \leftarrow \hat{b}_{\mathbb{A}} \oplus \bigoplus_{i\in\mathtt{B}} b_i$, where $\mathtt{B}$ is the set of corrupted bits $\mathtt{B} = [\kappa]\backslash\mathtt{I}$, and halts with output $(\mathtt{I}, \hat{b}_{\mathbb{B}})$.

As the oracles behave exactly the same, the simulation is perfect and, by inspection, $\mathbb{B}$ wins iff $\mathbb{A}$ wins. □

Suppl. Mat. 1.3 Proof of Lemma 2

Lemma 2. *Let $n \le \kappa$. Then there is an SFBB reduction $\mathbb{B}$ such that, for every adversary $\mathbb{A}$,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{A}) \le \mathsf{Adv}^{(\le\kappa,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{B})\,,$$

where $\mathbb{B}$ makes at most $\kappa - n$ additional bit corruption oracle calls.

Proof. As both games provide the same interface to the adversary, $\mathbb{B}$ can simply simulate the game for $\mathbb{A}$ by forwarding every oracle call, where $\mathbb{B}$ stores the input and output for later use. Upon halting, $\mathbb{A}$ returns a list $\mathtt{I}$ and a guess $\hat{b}_{\mathbb{A}}$ for the value of $\oplus_{i\in\mathtt{I}} b_i$. Let $\mathtt{B} = [\kappa]\backslash\mathtt{I}$, then $\mathbb{B}$ bit-corrupts all of $\mathtt{B}$, so it can set $\hat{b}_{\mathbb{B}} \leftarrow \hat{b}_{\mathbb{A}} \oplus \bigoplus_{i\in\mathtt{B}} b_i$ and halt with output $\hat{b}_{\mathbb{B}}$. As the oracles behave exactly the same, the simulation is perfect and, by inspection, $\mathbb{B}$ wins iff $\mathbb{A}$ wins. □

Suppl. Mat. 1.4 Proof of Theorem 3

Theorem 3 (MKU → UKU)**.** *There is an SFBB reduction $\mathbb{B}$ such that, for every adversary $\mathbb{A}$,*

$$\mathsf{Adv}^{(\kappa,\kappa)\text{-ind-cca}}_{\mathrm{PKE}}(\mathbb{A}) \le 2^{\kappa} \cdot \mathsf{Adv}^{(\kappa,\kappa)\text{-ror-cca}}_{\mathrm{PKE}}(\mathbb{B})\,,$$

where $\mathbb{B}$ additionally draws κ bits uniformly at random.

Proof. First, $\mathbb{B}$ uniformly draws κ independent bits d_i. Let $d = \oplus_{i\in[\kappa]} d_i$ denote their xor. Then, whenever $\mathbb{A}$ calls $\mathcal{E}(i, m_0, m_1)$, $\mathbb{B}$ calls $\mathcal{E}_{\mathrm{ROR}}(i, m_{d_i})$. Calls to $\mathcal{D}$ are simply forwarded. Once $\mathbb{A}$ halts with output $\hat{d}$, $\mathbb{B}$ will guess "real" iff $\mathbb{A}$ guessed d correctly, that is, $\mathbb{B}$ sets $\hat{b} = \hat{d} \oplus d$ and halts with the output $\hat{b}$.

If $b_i = 0$ for all $i \in [\kappa]$, then the simulation is perfect, and $\mathbb{B}$ wins whenever $\mathbb{A}$ wins. If, on the other hand, $b_i = 1$ for some i, then the corresponding d_i is information-theoretically hidden from $\mathbb{A}$ and consequently, d itself will be perfectly hidden. Hence, $\hat{d} = d$ will occur with probability $1/2$, irrespective of $\mathbb{A}$'s behaviour. Using these observations, we obtain:

$$\begin{aligned}
\Pr\Big[\mathsf{Exp}^{(\kappa,\kappa)\text{-ror-cca}}_{\mathrm{PKE}}(\mathbb{B}) = 1\Big] &= \Pr\big[\forall_{i\in[\kappa]} b_i = 0\big] \cdot \Pr\Big[\hat{d} = d \;\Big|\; \forall_{i\in[\kappa]} b_i = 0\Big] \\
&\quad + \Pr\Big[\hat{d} = d \;\Big|\; \exists_{i\in[\kappa]} b_i = 1\Big] \cdot \big(1 - \Pr\big[\forall_{i\in[\kappa]} b_i = 0\big]\big) \\
&= \frac{1}{2^{\kappa}} \Pr\Big[\mathsf{Exp}^{(\kappa,\kappa)\text{-ind-cca}}_{\mathrm{PKE}}(\mathbb{A}) = 1\Big] + \frac{1}{2}\left(1 - \frac{1}{2^{\kappa}}\right) \\
&= \frac{1}{2^{\kappa}} \left(\Pr\Big[\mathsf{Exp}^{(\kappa,\kappa)\text{-ind-cca}}_{\mathrm{PKE}}(\mathbb{A}) = 1\Big] - \frac{1}{2}\right) + \frac{1}{2} \\
\implies \mathsf{Adv}^{(\kappa,\kappa)\text{-ror-cca}}_{\mathrm{PKE}}(\mathbb{B}) &= \frac{1}{2^{\kappa}} \left(2 \cdot \Pr\Big[\mathsf{Exp}^{(\kappa,\kappa)\text{-ind-cca}}_{\mathrm{PKE}}(\mathbb{A}) = 1\Big] - 1\right) \\
&= \frac{1}{2^{\kappa}} \mathsf{Adv}^{(\kappa,\kappa)\text{-ind-cca}}_{\mathrm{PKE}}(\mathbb{A})\,.
\end{aligned}$$

□

Suppl. Mat. 1.5 Proof of Theorem 4

Theorem 4. *There is an SFBB reduction* $\mathbb{B}$ *such that, for every adversary* $\mathbb{A}$*,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{A}) \leq \binom{\kappa}{n} \cdot \mathsf{Adv}^{(n,n)\text{-ind-cca}}_{\mathrm{PKE}}(\mathbb{B})\,.$$

$\mathbb{B}$*'s overhead consists of generating* $\kappa - n$ *fresh keypairs, sampling* $\kappa - n$ *bits, and a choosing a subset of* $[\kappa]$ *of cardinality* n *uniformly at random.*

Proof. After $\mathbb{B}$ receives the n public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, for each $i \in [\kappa]\backslash[n]$, it generates fresh keypairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ and draws a uniform challenge bit $\hat{b}_i$.

After reshuffling the index set $\{i\}_{i\in[\kappa]}$, $\mathbb{B}$ samples a subset $\mathtt{I}_{\mathbb{B}} \subseteq [\kappa]$ of cardinality n uniformly at random. Whenever $\mathbb{A}$ calls $\mathcal{E}(i, m_0, m_1)$, $\mathbb{B}$ checks if $i \in \mathtt{I}_{\mathbb{B}}$. If yes, $\mathbb{B}$ forwards the call to its own oracle and returns the result. Otherwise, it simulates the oracle by encrypting m_{b_i} under pk_i, storing the resulting challenge ciphertext in C_i and returning c. Similarly, whenever $\mathbb{A}$ calls $\mathcal{D}(i, c)$, if $i \in \mathtt{I}_{\mathbb{B}}$, $\mathbb{B}$ calls $\mathcal{D}(i, c)$ and returns the result. Otherwise, $\mathbb{B}$ simulates the decryption oracle by returning $\lightning$ if $c \in \mathsf{C}_i$, and decrypting using sk_i otherwise.

If $\mathbb{A}$ calls either of its corruption oracles on i, $\mathbb{B}$ returns the expected value whenever $i \notin \mathtt{I}_{\mathbb{B}}$. However, if $i \in \mathtt{I}_{\mathbb{B}}$, $\mathbb{B}$ will simply halt and output $(\mathtt{I}_{\mathbb{B}}, 0)$. Observe that this sudden abort still gives $\mathbb{B}$ a win probability of $1/2$.

Once $\mathbb{A}$ halts with output $(\mathtt{I}_{\mathbb{A}}, \hat{b})$, $\mathbb{B}$ checks if $\mathtt{I}_{\mathbb{B}} = \mathtt{I}_{\mathbb{A}}$. If true, then $\mathbb{B}$ outputs the guess $(\mathtt{I}_{\mathbb{B}}, \hat{b})$; otherwise, $\mathbb{B}$ outputs $(\mathtt{I}_{\mathbb{B}}, 0)$. Note that the simulation is perfectly faithful and that, in the case $\mathtt{I}_{\mathbb{B}} = \mathtt{I}_{\mathbb{A}}$, $\mathbb{B}$ wins iff $\mathbb{A}$ wins. Otherwise, $\mathbb{B}$ will again win with probability $1/2$. Denoting $b = \oplus_{i\in\mathtt{I}_{\mathbb{B}}} b_i$, this gives us

$$\begin{aligned}
\Pr\left[\mathsf{Exp}^{(n,n)\text{-ind-cca}}_{\mathrm{PKE}}(\mathbb{B}) = 1\right] &= \Pr[\mathtt{I}_{\mathbb{B}} = \mathtt{I}_{\mathbb{A}}] \cdot \Pr\left[\hat{b} = b \;\middle|\; \mathtt{I}_{\mathbb{B}} = \mathtt{I}_{\mathbb{A}}\right] \\
&\quad + \Pr[b = 0] \cdot (1 - \Pr[\mathtt{I}_{\mathbb{B}} = \mathtt{I}_{\mathbb{A}}]) \\
&= \Pr[\mathtt{I}_{\mathbb{B}} = \mathtt{I}_{\mathbb{A}}] \cdot \Pr\left[\hat{b} = b \;\middle|\; \mathtt{I}_{\mathbb{B}} = \mathtt{I}_{\mathbb{A}}\right] + \frac{1}{2} \cdot \left(1 - \Pr[\mathtt{I}_{\mathbb{B}} = \mathtt{I}_{\mathbb{A}}]\right) \\
&= \Pr[\mathtt{I}_{\mathbb{B}} = \mathtt{I}_{\mathbb{A}}] \cdot \left(\Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{A}) = 1\right] - \frac{1}{2}\right) + \frac{1}{2}\,.
\end{aligned}$$

Since the simulation is perfect, $\mathtt{I}_{\mathbb{B}}$ is hidden to $\mathbb{A}$. This means that whatever $\mathbb{A}$ chooses, the probability of uniformly samplying $\mathtt{I}_{\mathbb{B}}$ such that $\mathtt{I}_{\mathbb{B}} = \mathtt{I}_{\mathbb{A}}$ is $\binom{k}{n}^{-1}$. This gives us,

$$\begin{aligned}
&= \frac{1}{\binom{\kappa}{n}} \cdot \left(\Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{A}) = 1\right] - \frac{1}{2}\right) + \frac{1}{2} \\
\implies \mathsf{Adv}^{(n,n)\text{-ind-cca}}_{\mathrm{PKE}}(\mathbb{B}) &= \frac{1}{\binom{\kappa}{n}} \left(2 \cdot \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{A}) = 1\right] - 1\right) \\
&= \frac{1}{\binom{\kappa}{n}} \mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{PKE}}(\mathbb{A})\,.
\end{aligned}$$

□

Suppl. Mat. 1.6 Proof of Theorem 5

Theorem 5. *Let* TKEM *be a* (γ, δ)*-correct TagKEM sampling keys from* $\{0,1\}^k$ *and with tagspace* $\mathcal{T}$*, let* $F : \{0,1\}^k \times \mathbb{Z}_{>0} \to \{0,1\}^*$ *be a XOF, and let* TXEM *be a TagXEM as given in Fig. 7 for arbitrary* $\mathcal{T}_\ell \subseteq \mathcal{T}$*. Then* TXEM *is* (γ, δ)*-correct, and there are SFBB reductions* $\mathbb{B}$ *and* $\mathbb{C}$ *such that, for every adversary* $\mathbb{A}$*,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{TXEM}}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{TKEM}}(\mathbb{B}) + 2 \cdot \mathsf{Adv}^{\mathrm{psrnd}}_{F}(\mathbb{C})\,.$$

If $\mathbb{A}$ *calls* $\mathcal{C}$ q_c *times and* $\mathcal{D}$ q_d *times, then* $\mathbb{B}$*'s overhead consists of at most* $q_c + q_d$ *evaluations of* F*, while* $\mathbb{C}$*'s overhead consists of doing* κ *executions of* TKEM.Kg*, at most* q_c *executions of* TKEM.Key *and* TKEM.Enc*, and at most* q_d *executions of* TKEM.Dec*.*

Oracle $\mathcal{C}(i,\ell)$ (game G_0)	Oracle $\mathcal{C}(i,\ell)$ (game G_1)
$(K_0^{\text{kem}}, \sigma) \leftarrow\!\!\$\ \mathsf{TKEM.Key}_{\mathsf{pk}_i}$	$(K_0^{\text{kem}}, \sigma) \leftarrow\!\!\$\ \mathsf{TKEM.Key}_{\mathsf{pk}_i}$
$\mathtt{E}_i \xleftarrow{\frown} \sigma$	$\mathtt{E}_i \xleftarrow{\frown} \sigma$
$K_0^{\text{xem}} \leftarrow F(K_0^{\text{kem}}, \ell)$	$K_0^{\text{xem}} \leftarrow F(K_0^{\text{kem}}, \ell)$
$K_1^{\text{xem}} \leftarrow\!\!\$\ \{0,1\}^\ell$	$K_1^{\text{kem}} \leftarrow\!\!\$\ \{0,1\}^k$
return $K_{b_i}^{\text{xem}}$	$K_1^{\text{xem}} \leftarrow F(K_1^{\text{kem}}, \ell)$
	return $K_{b_i}^{\text{xem}}$

Fig. 12. Encryption oracle for the proof of Thm. 5. In blue the code added to G_1 compared to G_0.

Proof. Let $\mathbb{A}$ be an adversary for $\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\text{TXEM}}$. We will use $\mathrm{G}_0(\mathbb{A})$ as shorthand for that experiment, where in Fig. 12 we have expanded the challenge oracle $\mathcal{C}$ based on the actual construction of $\mathsf{TXEM.Key}$. When called on a desired key length ℓ and a tag τ, $\mathcal{C}$ returns $K_{b_i}^{\text{xem}}$ such that either $K_{b_i}^{\text{xem}}$ is computed as $F(K_0^{\text{kem}}, \ell)$, or it is uniformly drawn from the keyspace $\{0,1\}^\ell$.

Next, let G_1 be as G_0, except that K_1^{xem} is computed as $F(K_1^{\text{kem}}, \ell)$, for some key K_1^{kem} uniformly sampled from the underlying TagKEMs keyspace $\{0,1\}^k$. Thus the challenge now takes the form $F(K_{b_i}^{\text{kem}}, \ell)$.

We claim that there is a reduction $\mathbb{B}$ such that

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\text{TKEM}}(\mathbb{B}) = 2\Pr[\mathrm{G}_1(\mathbb{A}) = 1] - 1\,.$$

To see this, let $\mathbb{B}$ answer $\mathbb{A}$'s challenge queries by quering its own oracle and, upon receiving K^{kem}, compute and forward $K^{\text{xem}} = F(K^{\text{kem}}, \ell)$. Similarly, whenever $\mathbb{A}$ queries the decryption oracle on (c, ℓ), $\mathbb{B}$ queries its own decryption oracle on c, receives K^{kem} (resp. $\perp/\lightning$) and forwards the reply $F(K^{\text{kem}}, \ell)$ (resp. $\perp/\lightning$) to $\mathbb{A}$. The public keys and calls to the other oracles are simply forwarded, and, when $\mathbb{A}$ outputs $(\mathtt{I}, \hat{b})$, $\mathbb{B}$ halts with the same output.

The simulation of G_1 is perfect, and $\mathbb{B}$ wins iff $\mathbb{A}$ wins. Therefore,

$$\Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\text{TKEM}}(\mathbb{B}) = 1\right] = \Pr[\mathrm{G}_1(\mathbb{A}) = 1]\,,$$

implying the statement.

We next claim that there is a reduction $\mathbb{C}$ such that

$$\mathsf{Adv}^{\text{psrnd}}_{F}(\mathbb{C}) = \Pr[\mathrm{G}_0(\mathbb{A}) = 1] - \Pr[\mathrm{G}_1(\mathbb{A}) = 1]\,.$$

$\mathbb{C}$ simulates the game for $\mathbb{A}$ by generating key pairs and challenge bits as needed, forwarding the public keys to $\mathbb{A}$, and whenever $\mathbb{A}$ queries for the encapsulation, decapsulation and corruption oracle, using the TXEM algorithms to give faithful simulations. When $\mathbb{A}$ queries $\mathcal{C}(i,\ell)$, however, $\mathbb{C}$ first runs $\mathsf{TKEM.Enc}_{\mathsf{pk}_i}$ to get a $K^{\text{kem}} \in \{0,1\}^k$ and a state σ. It then sets K_0^{xem} to $F(K_0^{\text{kem}}, \ell)$, before it queries its own challenge oracle on length ℓ, and gives the result to K_1^{xem}. After adding (σ, ℓ) into $\mathtt{E}_i$, as required for the simulation, it returns $K_{b_i}^{\text{xem}}$ to $\mathbb{A}$. Note that if in the PsRND game b is set to 0, then $K_1^{\text{xem}} = F(K, \ell)$, meaning this is a faithful simulation of G_1. If $b = 1$, K_1^{xem} is drawn uniformly from $\{0,1\}^\ell$, making it a faithful simulation of G_0.

Once $\mathbb{A}$ halts with output $(\mathtt{I}, \hat{b})$, $\mathbb{C}$ checks whether $\mathtt{I}$ is eligible, i.e. of the correct cardinality and not containing any corrupted indices. If yes, $\mathbb{C}$ outputs 1 if $\oplus_{i\in\mathtt{I}} b_i = \hat{b}$ and 0 otherwise. If $\mathtt{I}$ is ineligible, it outputs the guess 0. Assuming $\mathbb{A}$'s output is eligible, $\mathbb{C}$ gets the following advantage.

$$\begin{aligned}
\Pr\left[\mathsf{Exp}^{\text{psrnd}}_{F}(\mathbb{C}) = 1\right] &= \Pr[\mathbb{A} \text{ wins} \wedge b = 1] + \Pr[\mathbb{A} \text{ loses} \wedge b = 0] \\
&= \Pr[b = 1]\Pr[\mathrm{G}_0(\mathbb{A}) = 1] + \Pr[b = 0]\Pr[\mathrm{G}_1(\mathbb{A}) = 0] \\
&= \frac{1}{2}\Pr[\mathrm{G}_0(\mathbb{A}) = 1] + \frac{1}{2}\left(1 - \Pr[\mathrm{G}_1(\mathbb{A}) = 1]\right) \\
&= \frac{1}{2}\left(\Pr[\mathrm{G}_0(\mathbb{A}) = 1] - \Pr[\mathrm{G}_1(\mathbb{A}) = 1] + 1\right)\,,
\end{aligned}$$

from which we obtain

$$\begin{aligned}\mathsf{Adv}^{\mathrm{psrnd}}_{F}(\mathbb{C}) &= 2\cdot\Pr\Big[\mathsf{Exp}^{\mathrm{psrnd}}_{F}(\mathbb{C})=1\Big]-1\\ &= 2\cdot\frac{1}{2}\left(\Pr[\mathrm{G}_0(\mathbb{A})=1]-\Pr[\mathrm{G}_1(\mathbb{A})=1]+1\right)-1\\ &= \Pr[\mathrm{G}_0(\mathbb{A})=1]-\Pr[\mathrm{G}_1(\mathbb{A})=1]\,.\end{aligned}$$

If $\mathbb{A}$'s output was ineligible, $\mathbb{C}$ gets advantage

$$2\Pr[b=0]-1=0=\Pr[\mathrm{G}_0(\mathbb{A})=1\mid \mathtt{I}\text{ ineligible}]-\Pr[\mathrm{G}_1(\mathbb{A})=1\mid \mathtt{I}\text{ ineligible}]\,,$$

so the relation holds also in this case. The advantage of $\mathbb{A}$ can be bounded as:

$$\begin{aligned}\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{TXEM}}(\mathbb{A}) &= 2\Pr[\mathrm{G}_0(\mathbb{A})=1]-1\\ &= 2\left(\Pr[\mathrm{G}_0(\mathbb{A})=1]+\Pr[\mathrm{G}_1(\mathbb{A})=1]-\Pr[\mathrm{G}_1(\mathbb{A})=1]\right)-1\\ &= 2\left(\Pr[\mathrm{G}_0(\mathbb{A})=1]-\Pr[\mathrm{G}_1(\mathbb{A})=1]\right)+2\Pr[\mathrm{G}_1(\mathbb{A})=1]-1\\ &\le 2\cdot\mathsf{Adv}^{\mathrm{psrnd}}_{F}(\mathbb{C})+\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{TKEM}}(\mathbb{B})\,,\end{aligned}$$

yielding the statement.

As for correctness, $\mathsf{TXEM.Kg}=\mathsf{TKEM.Kg}$, so $\gamma_{\mathrm{TXEM}}=\gamma_{\mathrm{TKEM}}$. If a decryption error happens in TKEM, it translates to F being called with the wrong seed, leading to a decryption error in TXEM except in the case of an output collision. Thus, $\delta_{\mathrm{TXEM}}\le\delta_{\mathrm{TKEM}}$, allowing us to conclude that TXEM is $(\gamma_{\mathrm{TKEM}},\delta_{\mathrm{TKEM}})$-correct. □

Suppl. Mat. 1.7 Proof of Theorem 6

Theorem 6. *Let* XEM *be a (γ,δ)-correct XEM, and let* PKE *be a hybrid encryption scheme as given in Fig. 8. Then* PKE *is (γ,δ)-correct, and there is a type-preserving SFBB reduction $\mathbb{B}$ such that for every adversary $\mathbb{A}$,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ror-cpa}\star}_{\mathrm{PKE}}(\mathbb{A})\le\mathsf{Adv}^{(n,\kappa)\text{-ind-cpa}\star}_{\mathrm{XEM}}(\mathbb{B})\,.$$

Proof. $\mathbb{B}$, playing $\mathsf{Exp}^{(n,\kappa)\text{-ind-cpa}\star}_{\mathrm{XEM}}$ is able to perfectly simulate $\mathsf{Exp}^{(n,\kappa)\text{-ror-cpa}\star}_{\mathrm{PKE}}$ for $\mathbb{A}$ as follows. If $\mathbb{A}$ calls its challenge encryption oracle, $\mathbb{B}$ first acquires an ephemeral OTP key of the correct length from its challenge encryption oracle, and an encapsulation c_1. It next uses K to encrypt the given message, producing the message encapsulation c_2, and returns $\langle c_1,c_2\rangle$. Corruption oracle calls are simply forwarded. Eventually, when $\mathbb{A}$ terminates on an output $(\mathtt{I},\hat{b})$, $\mathbb{B}$ terminates on that very same output.

We claim that $\mathbb{B}$ provides a perfect simulation of the $\mathsf{Exp}^{(n,\kappa)\text{-ror-cpa}\star}_{\mathrm{PKE}}$ game, where the implicit challenge bits in $\mathsf{Exp}^{(n,\kappa)\text{-ror-cpa}\star}_{\mathrm{PKE}}$ exactly match the unknown ones in $\mathsf{Exp}^{(n,\kappa)\text{-ind-cpa}\star}_{\mathrm{XEM}}$. After all,

$$\left\{K\oplus m\mid m\leftarrow\!\$\,\{0,1\}^{\ell}\right\}=\left\{K\oplus m\mid K\leftarrow\!\$\,\{0,1\}^{\ell}\right\}\,,$$

so a random ephemeral key in $\mathbb{B}$'s game neatly corresponds to a random message in $\mathbb{A}$'s game. As $\mathbb{A}$'s view after corrupting remains consistent with $\mathsf{Exp}^{(n,\kappa)\text{-ror-cpa}\star}_{\mathrm{PKE}}$, the simulation is perfect and $\mathbb{B}$ wins iff $\mathbb{A}$ wins.

Given the perfect correctness of the OTP, it is easy to see that the correctness parameters transfer directly from XEM to PKE: $\mathsf{PKE.Kg}=\mathsf{XEM.Kg}$, so $\gamma_{\mathrm{PKE}}=\gamma_{\mathrm{XEM}}$; meanwhile, an incorrect decryption in the XEM translates to an incorrect K being xored with c_2 (assuming $K\neq\perp$), leading to incorrect m; otherwise, decryption always succeeds. Therefore, $\delta_{\mathrm{PKE}}=\delta_{\mathrm{XEM}}$. □

Suppl. Mat. 1.8 Proof of Theorem 7

Theorem 5. *Let* TXEM *be a (γ,δ)-correct TagXEM, and let* PKE$'$ *be a hybrid encryption scheme as given in Fig. 8. Then* PKE$'$ *is (γ,δ)-correct, and there is a type-preserving SFBB reduction $\mathbb{B}$ such that for every adversary $\mathbb{A}$,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ror-cca}\star}_{\mathrm{PKE}'}(\mathbb{A})\le\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{TXEM}}(\mathbb{B})\,.$$

If $\mathbb{A}$ calls oracle $\mathcal{E}(i, m)$	If $\mathbb{A}$ calls oracle $\mathcal{D}(i, \langle c_1, c_2 \rangle)$
$q_i \leftarrow q_i + 1$	$K \leftarrow \mathcal{D}_{\mathbb{B}}(i, \langle c_1, c_2 \rangle, \lvert c_2 \rvert)$
$K \leftarrow \mathcal{C}_{\mathbb{B}}(i, \lvert m \rvert)$	**if** $K = \lightning$ **then return** $\lightning$
$c_2 \leftarrow K \oplus m$	**if** $K = \perp$ **then return** $\perp$
$c_1 \leftarrow \mathcal{E}_{\mathbb{B}}(i, q_i, c_2)$	$m \leftarrow K \oplus c_2$
return $\langle c_1, c_2 \rangle$	**return** m

Fig. 13. The reduction $\mathbb{B}$, playing $\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\text{TXEM}}$, while giving a perfect simulation of $\mathsf{Exp}^{(n,\kappa)\text{-ror-cca}\star}_{\text{PKE}}$ for $\mathbb{A}$. Not shown are the counters q_i being initialized to 0, and the corruption oracles being forwarded directly. At the end of the game, $\mathbb{B}$ halts with the same output as $\mathbb{A}$.

If $\mathbb{A}$ calls oracle $\mathcal{E}(i, m_0, m_1)$	If $\mathbb{A}$ calls oracle $\mathcal{D}(i, \langle c_1, c_2 \rangle)$
if $m_0 \not\sim m_1$ **then return** $\lightning$	$K \leftarrow \mathcal{D}_{\mathbb{B}}(i, \langle c_1, c_2 \rangle, \lvert c_2 \rvert)$
$q_i \leftarrow q_i + 1$	**if** $K = \lightning$ **then return** $\lightning$
$K \leftarrow \mathcal{C}_{\text{ROP}}(i, \lvert m_0 \rvert, \Pi_{m_0 \to m_1})$	**if** $K = \perp$ **then return** $\perp$
$c_2 \leftarrow K \oplus m_0$	$m \leftarrow K \oplus m$
$c_1 \leftarrow \mathcal{E}(i, q_i, c_2)$	**return** $\langle c_1, c_2 \rangle$
return $\langle c_1, c_2 \rangle$	

Fig. 14. The reduction $\mathbb{B}$, playing $\mathsf{Exp}^{(n,\kappa)\text{-rop-cca}\star}_{\text{TXEM}}$, while giving a perfect simulation of $\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\text{PKE}'}$ for $\mathbb{A}$. Counters q_i are initialized to 0, corruption queries are forwarded directly. $\mathbb{B}$ halts with the same output as $\mathbb{A}$.

Proof. This time the reduction $\mathbb{B}$, playing $\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\text{TXEM}}$, is able to perfectly simulate $\mathsf{Exp}^{(n,\kappa)\text{-ror-cca}\star}_{\text{PKE}}$ for $\mathbb{A}$, including decryption queries. The way $\mathbb{B}$ handles $\mathbb{A}$'s challenge encryption and decryption queries is specified in Fig. 13. Encryption queries are handled essentially as before, just taking into account the slightly different syntax of TagXEMs. Calls to the decryption oracle are forwarded, yielding an ephemeral key that is used to produce the message from c_2. Note that, since the message encapsulation is used as a tag when producing the challenge encryption, $\mathbb{B}$'s decryption oracle will return $\lightning$ iff $\langle c_1, c_2 \rangle$ was previously issued as a challenge. (Essentially, $\mathbb{B}$ can rely on its own experiment to keep track of admin relevant for the experiment it simulates for $\mathbb{A}$.)

Corruption oracle calls are simply forwarded. Eventually, when $\mathbb{A}$ terminates on an output $(\mathtt{I}, \hat{b})$, $\mathbb{B}$ terminates on that very same output.

From here on out, the proof tracks that of Thm. 6, where this time we claim that $\mathbb{B}$ provides a perfect simulation of the $\mathsf{Exp}^{(n,\kappa)\text{-ror-cca}\star}_{\text{PKE}}$ game, where the implicit challenge bits in $\mathsf{Exp}^{(n,\kappa)\text{-ror-cca}\star}_{\text{PKE}}$ exactly match the unknown ones in $\mathsf{Exp}^{(n,\kappa)\text{-ind-cca}\star}_{\text{TXEM}}$. Furthermore, by inspection, both the corruption oracle and crucially the decryption oracle are impeccable. As the simulation is perfect, $\mathbb{B}$ wins iff $\mathbb{A}$ wins.

As with PKE, the correctness parameters of PKE$'$ transfer directly from TXEM: $\mathsf{PKE.Kg} = \mathsf{XEM.Kg}$, so $\gamma_{\text{PKE}} = \gamma_{\text{XEM}}$. A decryption error in TXEM again translates to the wrong OTP key being xored with c_2, or an unexpected $\perp$. Def. 5 guarantees that with probability at least δ_{TXEM} there are no tags τ such that decrypting fails. Given that the decryption algorithm of PKE$'$ does not allow tampering of the tags without also affecting c_2, PKE$'$ could in fact well end up allowing for much smaller values of δ than that of TXEM. Nevertheless we conclude that $\delta_{\text{PKE}} \leq \delta_{\text{TXEM}}$, and that PKE$'$ is $(\gamma_{\text{TXEM}}, \delta_{\text{TXEM}})$-correct. □

Suppl. Mat. 1.9 Proof of Theorem 8

Theorem 8. *Let* TXEM *be a* (γ, δ)*-correct TagXEM, and let* PKE$'$ *be a hybrid encryption scheme as given in Fig. 8. Then* PKE$'$ *is* (γ, δ)*-correct, and there is a type-preserving SFBB reduction* $\mathbb{B}$ *such that for every adversary* $\mathbb{A}$*,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\text{PKE}'}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-rop-cca}\star}_{\text{TXEM}}(\mathbb{B})\,.$$

Proof. $\mathbb{B}$ is given in Fig. 14. The simulation is as in the proof of Theorem 7, except for the simulation of the challenge encryption oracle: here, $\mathbb{B}$ now forwards the permutation $\Pi_{m_0 \to m_1}$ that on input K outputs

$m_0 \oplus m_1 \oplus K$, receiving as a challenge either the honest key K or the permuted key $\Pi_{m_0 \to m_1}(K)$. To see that the simulation is perfect, consider the view of $\mathbb{A}$ in the case that $\mathbb{A}$ challenges instance i on messages m_0, m_1, then corrupts, receiving sk_i and decrypting the challenge. If the decryption is correct and $b_i = 0$, $\mathbb{A}$ recovers the honest key K_0, and therefore the message m_0. While, whenever $b_i = 1$, $\mathbb{A}$ still recovers K_0, but now the challenge was produced by encrypting m_0 using the permuted key. Decrypting will then yield

$$K_0 \oplus c_2 = K_0 \oplus K_1 \oplus m_0 = K_0 \oplus m_0 \oplus m_1 \oplus K_0 \oplus m_0 = m_1 \,,$$

just as expected. Note how this implies that the bits of each game correspond exactly to each other; the simulation is perfect, and $\mathbb{B}$ wins iff $\mathbb{A}$ wins.

The correctness parameters of PKE′ again transfer directly from TXEM (see Thm. 7). □

Suppl. Mat. 1.10 Proof of Theorem 9

Theorem 9. *Let* TXEM *be as in Fig. 10, let* KEM *be a perfectly correct KEM with unique encapsulations, and let $k \in \mathbb{Z}_{>1}$. Then there is an SFBB reduction $\mathbb{B}$ such that, for all $\mathbb{A}$ that makes q_d decryption oracle queries,*

$$\mathsf{Adv}^{(n,\kappa)\text{-rop-cca}\star}_{\text{TXEM}}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-ow-pca}\star}_{\text{KEM}}(\mathbb{B}) + 2\big(q_d \hat{\epsilon}_{k-1} + \epsilon_k(q_c)\big)$$

in the programmable random oracle model, where $\hat{\epsilon}_{k-1}$ is the forging advantage after observing $k-1$ valid message–tag pairs (Def. 11) and $\epsilon_k(q_c)$ is the k-out-of-q_c encapsulation multi-collision probability of KEM (Def. 9). If $\mathbb{A}$ makes q_f queries to the random oracle, then $\mathcal{F}$ makes at most q_f queries to its plaintext checking oracle.

Proof. For simplicity, we will in the following employ the convention $\mathsf{MAC}_{K^{\text{mac}}}(m) \coloneqq \mathsf{MAC.Mac}_{K^{\text{mac}}}(m)$.

The game $\text{G}_0(\mathbb{A})$, given in Fig. 15, is simply the ROP-CCA TagXEM experiment of Figs. 6 and 9 instantiated with the TagXEM construction of Fig. 10, with slightly simplified key management and including the XOF modelled as a lazily sampled random oracle $\mathcal{F}$. For the latter, we treat the bit string $\mathtt{F}(\mathsf{pk}, c, K^{\text{kem}})$ as a list and use $\xleftarrow{\frown}\$$ to sample (in this case a uniform random bit) and append to that string. The while loop thus ensures the correct prefix behaviour, while keeping the code concise. As output we take the first ℓ bits of $\mathtt{F}(\mathsf{pk}, c, K^{\text{kem}})$, denoted $\mathtt{F}(\mathsf{pk}, c, K^{\text{kem}})[\![\ell]\!]$.

In G_1 (Fig. 16), we make the $\mathcal{C}$ oracle independent of the ephemeral key returned by $\mathsf{KEM.Enc}$. To achieve this, the random oracle is altered so that now, whenever $\mathcal{F}$ is called together with consistent K^{kem} and c, the value of K^{kem} gets stored in a list $\mathtt{K}$. In the case that the correct key has been called to $\mathcal{F}$ before, we then simply retrieve it and call $\mathcal{F}$ as usual. Otherwise, we pre-sample K^{mac} and K^{xem}, and store them in the list $\mathtt{D}_i(c)$. Then, once $\mathcal{F}$ is called on that ciphertext with the corresponding K^{kem}, we program the random oracle using the value stored in $\mathtt{D}_i(c)$, ensuring consistency. Note how the programming does not alter the output distribution of $\mathcal{F}$, as each pre-sampled value is sampled uniformly at random, and is only ever programmed to a single element.

Additionally, we introduce the counters $ctr(\cdot,\cdot)$ to detect whether a k-wise multi-collision is created by the $\mathcal{C}$ oracle and, if so, the flag $\mathsf{bad}_{\mathsf{mc}}$ is set. This multi-collision detection does not change the behaviour of the game—it will assist us bounding another bad event in the next hop. We therefore have that G_0 and G_1 behave identically and

$$\Pr[\text{G}_0(\mathbb{A})] - \Pr[\text{G}_1(\mathbb{A})] = 0 \,. \tag{1}$$

In $\text{G}_2(\mathbb{A})$ (Fig. 17), we make the decryption oracle independent of the private keys by leveraging the lists introduced in $\text{G}_1(\mathbb{A})$: if $\mathbb{A}$ calls for the decryption of c under sk_i and $\mathtt{K}_i(c)$ has already been defined, we simply retrieve K^{kem} and call the random oracle as usual. Otherwise, we can conclude that $\mathcal{F}$ is yet to be called on the relevant values, and thus use $\mathtt{D}_i(c)$ to retrieve/define K^{mac} and check the validity of mac. If the latter is invalid, we return $\perp$; otherwise, the event $\mathsf{bad}_{\mathsf{mf}}$ occurs (and we may abort $\text{G}_2(\mathbb{A})$ with any output).

By design, G_1 and G_2 are identical-until-$\mathsf{bad}_{\mathsf{mf}}$, and $\mathsf{bad}_{\mathsf{mf}}$ only happens if the adversary is able to produce a valid MAC forgery, possibly after having seen a number of valid mac produced under the same key by the game's oracles. We use $\mathsf{bad}_{\mathsf{mc}}$ to bound the number of valid mac an adversary has seen: we claim that if $\mathsf{bad}_{\mathsf{mc}}$ has not been set, that number is at most $k-1$ (see the argument below), implying that the probability of

Game $\mathrm{G}_0(\mathbb{A})$

$(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_\kappa, \mathsf{sk}_\kappa) \leftarrow\!\!\$\ \mathsf{KEM.Kg}$
$b_1, \ldots, b_\kappa \leftarrow\!\!\$\ \{0,1\}$
$(\mathtt{I}, \hat{b}) \leftarrow\!\!\$\ \mathbb{A}^{\mathcal{C},\mathcal{E},\mathcal{D},\mathcal{K},\mathcal{B},\mathcal{F}}(\mathsf{pk}_1, \ldots, \mathsf{pk}_\kappa)$
if $|\mathtt{I}| \neq n \vee \mathtt{I} \cap (\mathtt{K} \cup \mathtt{B}) \neq \emptyset$ **then** $\hat{b} \leftarrow 0$
return $\oplus_{i \in \mathtt{I}}\, b_i = \hat{b}$

Oracle $\mathcal{D}(i, \langle \langle c, \mathsf{mac} \rangle, \tau \rangle, \ell)$

if $\langle \langle c, \mathsf{mac} \rangle, \tau \rangle \in \mathtt{C}_i$ **then return** $\lightning$
$K^{\mathrm{kem}} \leftarrow \mathsf{KEM.Dec}_{\mathsf{sk}_i}(c)$
if $K^{\mathrm{kem}} = \perp$ **then return** $\perp$
$\ell' \leftarrow \ell + \ell_{\mathsf{mackey}}$
$K^{\mathrm{mac}} \| K^{\mathrm{xem}} \leftarrow \mathcal{F}(\mathsf{pk}_i, c, K^{\mathrm{kem}}, \ell')$
if $\mathsf{MAC}_{K^{\mathrm{mac}}}(\tau) \neq \mathsf{mac}$ **then return** $\perp$
return K^{xem}

Oracle $\mathcal{C}(i, \ell, \Pi)$

$(K^{\mathrm{kem}}, c) \leftarrow\!\!\$\ \mathsf{KEM.Enc}_{\mathsf{pk}_i}$
$\ell' \leftarrow \ell + \ell_{\mathsf{mackey}}$
$K^{\mathrm{mac}} \| K_0^{\mathrm{xem}} \leftarrow \mathcal{F}(\mathsf{pk}_i, c, K^{\mathrm{kem}}, \ell')$
$\mathtt{E}_i \xleftarrow{\frown} \langle c, K^{\mathrm{mac}} \rangle$
$K_1^{\mathrm{xem}} \leftarrow \Pi(K_0^{\mathrm{xem}})$
return $K_{b_i}^{\mathrm{xem}}$

Oracle $\mathcal{E}(i, j, \tau)$

if $\mathtt{E}_i[j] = \emptyset$ **then return** $\lightning$
$\langle c, K^{\mathrm{mac}} \rangle \leftarrow \mathtt{E}_i[j], \mathtt{E}_i[j] \leftarrow \emptyset$
$\mathsf{mac} \leftarrow \mathsf{MAC}_{K^{\mathrm{mac}}}(\tau)$
$\mathtt{C}_i \xleftarrow{\cup} \langle \langle c, \mathsf{mac} \rangle, \tau \rangle$
return $\langle c, \mathsf{mac} \rangle$

Oracle $\mathcal{F}(\mathsf{pk}, c, K^{\mathrm{kem}}, \ell)$

while $\left|\mathtt{F}(\mathsf{pk}, c, K^{\mathrm{kem}})\right| < \ell$
  $\mathtt{F}(\mathsf{pk}, c, K^{\mathrm{kem}}) \xleftarrow{\frown}\!\!\$\ \{0,1\}$
return $\mathtt{F}(\mathsf{pk}, c, K^{\mathrm{kem}})[\![\ell]\!]$

Fig. 15. The game $\mathrm{G}_0(\mathbb{A})$ with $\mathcal{F}$ modelled as a lazily sampled random oracle. Not shown are the corruption oracles, which are as in Fig. 6, and will remain unchanged over the course of the game hops.

a forgery at that point is at most $\hat{\epsilon}_{k-1}$ (Def. 11) and by a simple union bound $\Pr[\mathsf{bad_{mf}} \mid \neg\mathsf{bad_{mc}}] \leq q_d \hat{\epsilon}_{k-1}$ where q_d is the number of decryption calls.

Back to flag $\mathsf{bad_{mc}}$: assume it has not yet been set, thus for all pk and c, the counters $ctr(\mathsf{pk}, c) \leq k-1$. As the KEM is perfectly correct and has unique encapsulations, the implication is that, for any input-triple, the number of "delayed" $\mathtt{F}(\mathsf{pk}, c, K^{\mathrm{kem}})$ calls during $\mathcal{C}$ is at most $k-1$ and each such $\mathcal{C}$ call can lead to at most one valid mac to be output by $\mathcal{E}$, proving our claim from above. Finally, we note that $\Pr[\mathsf{bad_{mc}}] \leq \epsilon_k(q_c)$ where q_c is the total number of $\mathcal{C}$ queries (here we use that $\epsilon_k(x) + \epsilon_k(y) \leq \epsilon_k(x+y)$).

The arguments above show that

$$\Pr[\mathrm{G}_1(\mathbb{A})] - \Pr[\mathrm{G}_2(\mathbb{A}) \mid \neg\mathsf{bad_{mf}}] \leq \Pr[\mathrm{G}_2(\mathbb{A}) \,:\, \mathsf{bad_{mf}}] \leq q_d \hat{\epsilon}_{k-1} + \epsilon_k(q_c)\,. \tag{2}$$

Now $\mathbb{B}$, given in Fig. 18, gives a faithful simulation of G_2 while playing $\mathsf{Adv}^{(n,\kappa)\text{-ow-pca}\star}(\mathbb{A})$, using its encryption oracle in place of $\mathsf{KEM.Enc}$ to produce the challenges in $\mathcal{C}$, and its plaintext checking oracle in place of $\mathsf{KEM.Dec}$ to check for consistent inputs to $\mathcal{F}$. At the end of its run, $\mathbb{A}$ returns a list of targets $\mathtt{I}$ and a guess $\hat{b}$. Assuming $\mathsf{bad_{mf}}$ was not triggered, and that none of the i in $\mathtt{I}$ were corrupted, $\mathbb{B}$ proceeds to check whether correct decapsulations were queried to the random oracle together with a challenge ciphertext produced under each targeted key, and collects one such decapsulation per key from $\mathtt{K}_i$. It finally halts with the collected keys, together with $\mathtt{I}$ and a set of indices j_i specifying the targeted challenge.

Let Q be the event that $\mathbb{A}$ returns a list $\mathtt{I}$ of uncompromised instances, for which $\mathcal{F}$ was queried with each key pk_i in $\mathtt{I}$ *at least once* with a K^{kem} satisfying $K^{\mathrm{kem}} = \mathsf{KEM.Dec}_{sk_i}(c)$. Note how $\neg Q$ then implies that either at least one challenge bit b_i is information-theoretically hidden from $\mathbb{A}$, and therefore also $\oplus_{i \in \mathtt{I}} b_i$,

Oracle $\mathcal{C}(i, \ell, \Pi)$

$(\cdot, c) \leftarrow\!\!\$\ \mathsf{KEM.Enc}_{\mathsf{pk}_i}$
$ctr(\mathsf{pk}_i, c) \leftarrow ctr(\mathsf{pk}_i, c) + 1$
if $ctr(\mathsf{pk}_i, c) \geq k$ **then** $\mathsf{bad}_{\mathsf{mc}} \leftarrow \mathsf{true}$
$\ell' \leftarrow \ell + \ell_{\mathsf{mackey}}$
if $\mathtt{K}_i(c) \neq \emptyset$
  $K^{\mathrm{kem}} \leftarrow \mathtt{K}_i(c)$
  $K^{\mathrm{mac}} \| K_0^{\mathrm{xem}} \leftarrow \mathcal{F}(\mathsf{pk}_i, c, K^{\mathrm{kem}}, \ell')$
else
  while $|\mathtt{D}_i(c)| < \ell'$
    $\mathtt{D}_i(c) \xleftarrow{\$} \{0,1\}$
  $K^{\mathrm{mac}} \| K_0^{\mathrm{xem}} \leftarrow \mathtt{D}_i(c)[\![\ell']\!]$
$\mathtt{E}_i \xleftarrow{} \langle c, K^{\mathrm{mac}} \rangle$
$K_1^{\mathrm{xem}} \leftarrow \Pi(K_0^{\mathrm{xem}})$
return $K_{b_i}^{\mathrm{xem}}$

Oracle $\mathcal{F}(\mathsf{pk}, c, K^{\mathrm{kem}}, \ell)$

if $\mathtt{F}(\mathsf{pk}, c, K^{\mathrm{kem}}) = \varepsilon$ **then**
  $\forall_{i \in [\kappa]} \mathsf{pk}_i = \mathsf{pk}$
    if $\mathsf{KEM.Dec}_{\mathsf{sk}_i}(c) = K^{\mathrm{kem}}$ **then**
      $\mathtt{K}_i(c) \leftarrow K^{\mathrm{kem}}$
      if $\mathtt{D}_i(c) \neq \varepsilon$ **then**
        $\mathtt{F}(\mathsf{pk}_i, c, K^{\mathrm{kem}}) \leftarrow \mathtt{D}_i(c)$
while $\left|\mathtt{F}(\mathsf{pk}, c, K^{\mathrm{kem}})\right| < \ell$
  $\mathtt{F}(\mathsf{pk}, c, K^{\mathrm{kem}}) \xleftarrow{\$} \{0,1\}$
return $\mathtt{F}(\mathsf{pk}, c, K^{\mathrm{kem}})[\![\ell]\!]$

Fig. 16. The oracles modified in game G_1, in which $\mathcal{C}$ is made independent of the sampled K^{kem}. The flag $\mathsf{bad}_{\mathsf{mc}}$ is initialized to false and the counters $ctr(\cdot,\cdot)$ to 0. Changes in blue.

or the output of $\mathbb{A}$ is invalid; in either case, $\mathbb{A}$ gets advantage 0. This gives us,

$$\begin{aligned}
\Pr[\mathrm{G}_2(\mathbb{A}) \mid \neg\mathsf{bad}_{\mathsf{mf}}] &= \Pr[\mathrm{G}_2(\mathbb{A}) \mid Q \wedge \neg\mathsf{bad}_{\mathsf{mf}}] \cdot \Pr[Q \mid \neg\mathsf{bad}_{\mathsf{mf}}] \\
&\quad + \Pr[\mathrm{G}_2(\mathbb{A}) \mid \neg Q \wedge \neg\mathsf{bad}_{\mathsf{mf}}] \cdot \Pr[\neg Q \mid \neg\mathsf{bad}_{\mathsf{mf}}] \\
&\leq \Pr[Q \mid \neg\mathsf{bad}_{\mathsf{mf}}] + \frac{1}{2}\left(1 - \Pr[Q \mid \neg\mathsf{bad}_{\mathsf{mf}}]\right) \qquad (3) \\
&= \frac{1}{2}\Pr[Q \mid \neg\mathsf{bad}_{\mathsf{mf}}] + \frac{1}{2}\,.
\end{aligned}$$

Then note that

$$\begin{aligned}
&\Pr[Q \mid \neg\mathsf{bad}_{\mathsf{mf}}] \leq \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-ow-pca}^\star}_{\mathrm{KEM}}(\mathbb{B})\right], \\
\implies &\Pr[\mathrm{G}_2(\mathbb{A}) \mid \neg\mathsf{bad}_{\mathsf{mf}}] \leq \frac{1}{2}\Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-ow-pca}^\star}_{\mathrm{KEM}}(\mathbb{B})\right] + \frac{1}{2}\,. \qquad (4)
\end{aligned}$$

Combining eqs. (1)–(4), we get

$$\begin{aligned}
\mathsf{Adv}^{(n,\kappa)\text{-rop-cca}^\star}_{\mathrm{TXEM}}(\mathbb{A}) &= 2\Pr[\mathrm{G}_0(\mathbb{A})] - 1 \\
&= 2\big(\Pr[\mathrm{G}_1(\mathbb{A})] - \Pr[\mathrm{G}_2(\mathbb{A}) \mid \neg\mathsf{bad}_{\mathsf{mf}}] + \Pr[\mathrm{G}_2(\mathbb{A}) \mid \neg\mathsf{bad}_{\mathsf{mf}}]\big) - 1 \\
&= 2\big(\Pr[\mathrm{G}_1(\mathbb{A})] - \Pr[\mathrm{G}_2(\mathbb{A}) \mid \neg\mathsf{bad}_{\mathsf{mf}}]\big) + 2\Pr[\mathrm{G}_2(\mathbb{A}) \mid \neg\mathsf{bad}_{\mathsf{mf}}] - 1 \\
&\leq 2\big(q_d\hat{\epsilon}_{k-1} + \epsilon_k(q_c)\big) + 2\left(\frac{1}{2}\Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-ow-pca}^\star}_{\mathrm{KEM}}(\mathbb{B})\right] + \frac{1}{2}\right) - 1 \\
&= 2\big(q_d\hat{\epsilon}_{k-1} + \epsilon_k(q_c)\big) + \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-ow-pca}^\star}_{\mathrm{KEM}}(\mathbb{B})\right] \\
&= 2\big(q_d\hat{\epsilon}_{k-1} + \epsilon_k(q_c)\big) + \mathsf{Adv}^{(n,\kappa)\text{-ow-pca}^\star}_{\mathrm{KEM}}(\mathbb{B})\,.
\end{aligned}$$

□

Suppl. Mat. 2 The Relationship between ROP and IND Security for KEMs

The following lemma shows that ROP is at least as strong as IND, in the sense that ROP tightly implies IND.

Oracle $\mathcal{D}(i, \langle\langle c, \mathsf{mac}\rangle, \tau\rangle, \ell)$

if $\langle\langle c, \mathsf{mac}\rangle, \tau\rangle \in \mathtt{C}_i$ **then return** $\lightning$
if $\mathtt{K}_i(c) = \emptyset$
  while $|\mathtt{D}_i(c)| < \ell_{\mathsf{mackey}}$
    $\mathtt{D}_i(c) \xleftarrow{\frown} \$ \{0,1\}$
  $K^{\mathrm{mac}} \leftarrow \mathtt{D}_i(c)[\![\ell_{\mathsf{mackey}}]\!]$
  if $\mathsf{MAC}_{K^{\mathrm{mac}}}(\tau) \neq \mathsf{mac}$ **then return** $\perp$
  else $\mathsf{bad}_{\mathsf{mf}} \leftarrow \mathsf{true}$ and abort
$K^{\mathrm{kem}} \leftarrow \mathtt{K}_i(c)$
$\ell' \leftarrow \ell + \ell_{\mathsf{mackey}}$
$K^{\mathrm{mac}} \| K^{\mathrm{xem}} \leftarrow \mathcal{F}(\mathsf{pk}_i, c, K^{\mathrm{kem}}, \ell')$
if $\mathsf{MAC}_{K^{\mathrm{mac}}}(\tau) \neq \mathsf{mac}$ **then return** $\perp$
return K^{xem}

Fig. 17. The modified $\mathcal{D}$ oracle in $\mathrm{G}_2(\mathbb{A})$ (changes in blue) is made independent of the private keys. The $\mathsf{bad}_{\mathsf{mf}}$ flag, initialized to false, is set when a valid forgery is made without $\mathcal{F}$ having been called on the relevant values.

Lemma 4 (ROP $\implies$ IND)**.** *There is a type-preserving SFBB reduction $\mathbb{B}$ such that, for any adversary $\mathbb{B}$,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{TXEM}}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-rop-cca}\star}_{\mathrm{TXEM}}(\mathbb{B})\,.$$

If $\mathbb{A}$ calls its challenge oracle q times, then $\mathbb{B}$ draws q keys uniformly at random.

Proof. Whenever $\mathbb{A}$ calls $\mathcal{C}(i, \ell)$, let $\mathbb{B}$ define the permutation Π that is simply the xor of the input with a freshly sampled key, and then call $\mathcal{C}_{\mathrm{ROP}}(i, \ell, \Pi)$. The resulting K_1 will look like a uniformly random key, with no relation to the encapsulated key K_0. □

In the other direction, a loss of $\binom{\kappa}{n} \cdot 2^n$ is inferred, as can be seen by adapting the proof techniques that led to Cor. 2.

Theorem 10 (IND $\implies$ ROP)**.** *There is an SFBB reduction $\mathbb{B}$ such that, for any adversary $\mathbb{B}$,*

$$\mathsf{Adv}^{(n,\kappa)\text{-rop-cca}\star}_{\mathrm{TXEM}}(\mathbb{A}) \leq \binom{\kappa}{n} \cdot 2^n \cdot \mathsf{Adv}^{(n,\kappa)\text{-ind-cca}\star}_{\mathrm{TXEM}}(\mathbb{B})\,.$$

$\mathbb{B}$'s overhead consists of generating $\kappa - n$ fresh keypairs, sampling κ bits, and a choosing a subset of $[\kappa]$ of cardinality n uniformly at random.

Suppl. Mat. 3 On the MI-GapCDH Problem with Corruptions

In this section we study the hardness of the MI-GapCDH problem, as introduced by AGK, when enhanced with corruptions. AGK considered three settings: the (realistic) high granularity setting, with one global (standardized) group and group generator; mid granularity, where the group remains global but generators are independently sampled per user; and low granularity, where the groups themselves are independently sampled among users. AGK went on to provide useful bounds linking the hardness of each to standard assumptions (in the algebraic and generic group models).

Restricting ourselves for now to low granularity MI-GapCDH in the generic group model, we can directly adapt AGK's bound [4, Thm. 6]. Denote by $\mathsf{Exp}^{(n,\kappa)\text{-gapcdh}\star}_{\text{low-gran}}$ the low granularity n-out-of-κ MI-GapCDH experiment with corruptions (Fig. 19); the advantage of an adversary $\mathbb{A}$ against the experiment is $\mathsf{Adv}^{(n,\kappa)\text{-gapcdh}\star}_{\text{low-gran}}(\mathbb{A}) := \Pr\left[\mathsf{Exp}^{(n,\kappa)\text{-gapcdh}\star}_{\text{low-gran}}(\mathbb{A}) = 1\right]$.

Compared to AGK, we make one more change to the experiment by defining valid lists $\mathtt{I}$, in addition to containing no compromised indices, to be of size exactly n rather than greater-or-equal n; this happens without loss of generality for computational problems (see also Sec. 3.3).

$\mathbb{B}(\mathsf{pk}_1, \ldots, \mathsf{pk}_\kappa)$

$b_1, \ldots, b_\kappa \leftarrow \{0,1\}$
$(\mathtt{I}, \hat{b}) \leftarrow \mathbb{A}^{\mathcal{C},\mathcal{E},\mathcal{D},\mathcal{K},\mathcal{B},\mathcal{F}}(\mathsf{pk}_1, \ldots, \mathsf{pk}_\kappa)$
$i_1, \ldots, i_n \leftarrow \mathtt{I}$
for $k \in [n]$:
  for c **s.th.** $\mathtt{K}_{i_k}(c) \neq \emptyset$:
    for j **s.th.** $c = \mathtt{P}_{i_k}[j]$:
      $j_{i_k} \leftarrow j, \hat{K}_{i_k} \leftarrow \mathtt{K}_{i_k}(c)$
return $(\mathtt{I}, (j_i, \hat{K}_i)_{i \in \mathtt{I}})$

If $\mathbb{A}$ calls oracle $\mathcal{C}(i, \Pi, \ell)$

$c \leftarrow \mathcal{E}_{\mathbb{B}}(i), \mathtt{P}_i \xleftarrow{\frown} c$
$\ell' \leftarrow \ell + \ell_{\mathsf{mackey}}$
if $\mathtt{K}_i(c) \neq \emptyset$:
  $K^{\mathrm{kem}} \leftarrow \mathtt{K}_i(c)$
  $K^{\mathrm{mac}} \| K_0^{\mathrm{xem}} \leftarrow \mathcal{F}(\mathsf{pk}_i, c, K^{\mathrm{kem}}, \ell')$
else :
  while $|\mathtt{D}_i(c)| < \ell'$:
    $\mathtt{D}_i(c) \xleftarrow{\frown}\$ \{0,1\}$
  $K^{\mathrm{mac}} \| K_0^{\mathrm{xem}} \leftarrow \mathtt{D}_i(c)[\![\ell']\!]$
$\mathtt{E}_i \xleftarrow{\frown} \langle c, K^{\mathrm{mac}} \rangle$
$K_1^{\mathrm{xem}} \leftarrow \Pi(K_0^{\mathrm{xem}})$
return $K_{b_i}^{\mathrm{xem}}$

Oracle $\mathcal{F}(\mathsf{pk}, c, K^{\mathrm{kem}}, \ell)$

if $\mathtt{F}(\mathsf{pk}, c, K^{\mathrm{kem}}) = \emptyset$:
  $\forall_{i \in [\kappa]} \mathsf{pk}_i = \mathsf{pk}$:
    if $\mathcal{P}_{\mathbb{B}}(i, c, K^{\mathrm{kem}})$:
      $\mathtt{K}_i(c) \leftarrow K^{\mathrm{kem}}$
      if $\mathtt{D}_i(c) \neq \varepsilon$:
        $\mathtt{F}(\mathsf{pk}_i, c, K^{\mathrm{kem}}) \leftarrow \mathtt{D}_i(c)$
while $\mathtt{F}(\mathsf{pk}, c, K^{\mathrm{kem}}) < \ell$:
  $\mathtt{F}(\mathsf{pk}, c, K^{\mathrm{kem}}) \xleftarrow{\frown}\$ \{0,1\}$
return $\mathtt{F}(\mathsf{pk}, c, K^{\mathrm{kem}})[\![\ell]\!]$

Fig. 18. The reduction $\mathbb{B}$, simulating G_2 while playing $\mathsf{Exp}^{(n,\kappa)\text{-ow-pca}\star}_{\mathrm{KEM}}$. Changes to the oracles are highlighted in blue; omitted oracles are simulated exactly as in G_2.

Theorem 11. *Let* GGen *be a group-generating algorithm generating generic groups of size at least* p*. Let* n, κ, q, q_D *and* q_i *for* $i \in [\kappa]$ *such that* $1 \leq n \leq \kappa$, $q = \sum_{i=1}^{n} q_i$, *and* q_i *are such that* $q_i \geq 60 \log_2 p$ *and* $4q_i^2 \geq q_D$*. Then, for any adversary* $\mathbb{A}$ *making at most* q_i *queries to the* i*-th group-operation oracle and* q_D *queries to the gap oracle,*

$$\mathsf{Adv}^{(n,\kappa)\text{-gapcdh}\star}_{\text{low-gran}}(\mathbb{A}) \leq \left(\frac{4 \cdot e \cdot q^2}{n^2 \cdot p}\right)^n .$$

Proof (sketch). The proof is identical to AGK's Theorem 6 up to the additional observation that, since instances are completely independent, knowing the solution to one cannot help in solving others. This does not alter any of the steps of their proof: as AGK point out, all the generated groups are independent and, to win, $\mathbb{A}$ must output a valid list $\mathtt{I}$ of indices and a corresponding list $\hat{\mathtt{Z}}$ of solutions; all we have done is alter what constitutes a valid list $\mathtt{I}$. □

AGK showed that low granularity schemes achieve optimal scaling, as apparent in the exponential nature of the above bound. Thus, Theorem 11 allows the multi-instance security of our constructions to lean on an explicit bound with optimal scaling; all that remains is to observe that for ElGamal KEM, its multi-instance OW-PCA security is easily achieved from the low granularity MI-GapCDH assumption. Writing X for g^x and Y for g^y, let KEM be such that $\mathsf{KEM.Kg}$ samples a group $(\mathbb{G}, g, p) \leftarrow\$ \mathsf{GGen}$, private exponent $x \leftarrow\$ \mathbb{Z}_p$ and sets $(\mathsf{pk}, \mathsf{sk}) \leftarrow (((\mathbb{G}, g, p), X), x)$, $\mathsf{KEM.Enc_{pk}}$ samples $y \leftarrow\$ \mathbb{Z}_p$ and sets $(K, c) \leftarrow (X^y, Y)$, and $\mathsf{KEM.Dec_{sk}}$, on input $c = Y$, returns $K \leftarrow Y^x$ (where we assume $\mathsf{KEM.Dec}$ also implicitly has access to the group description).

Experiment $\mathsf{Exp}^{(n,\kappa)\text{-gapcdh}\star}_{\text{low-gran}}(\mathbb{A})$

for $i \in [\kappa]$ **do**
  $(\mathbb{G}_i, g_i, p_i) \leftarrow\!\!\$\ \mathsf{GGen}$
  $\mathtt{G}[i] \leftarrow (\mathbb{G}_i, g_i, p_i)$
  $\mathtt{x}[i] \leftarrow\!\!\$\ \mathbb{Z}_{p_i}$; $\mathtt{X}[i] \leftarrow g_i^{\mathtt{x}[i]}$
  $\mathtt{y}[i] \leftarrow\!\!\$\ \mathbb{Z}_{p_i}$; $\mathtt{Y}[i] \leftarrow g_i^{\mathtt{y}[i]}$
  $\mathtt{Z}[i] \leftarrow g_i^{\mathtt{x}[i]\cdot\mathtt{y}[i]}$
$(\mathtt{I}, \hat{\mathtt{Z}}) \leftarrow\!\!\$\ \mathbb{A}^{\mathcal{DDH},\mathcal{K}}(\mathtt{G}, \mathtt{X}, \mathtt{Y})$
if $|\mathtt{I}| \neq n$ **then return** 0
if $\mathtt{I} \cap \mathtt{K} \neq \emptyset$ **then return** 0
return $\bigwedge_{i\in\mathtt{I}} \mathtt{Z}[i] = \hat{\mathtt{Z}}[i]$

Oracle $\mathcal{DDH}(i, X, Y, Z)$

parse $(X, Y) \to (g_i^x, g_i^y)$
return $g_i^{x\cdot y} = Z$

Oracle $\mathcal{K}(i)$

$\mathtt{K} \xleftarrow{\cup} i$
return $\mathtt{x}[i]$

Fig. 19. Multi-instance gap CDH with corruption for the low granularity case, i.e. each challenge is sampled for a completely independent group and generator. Highlighted in blue, the differences introduced concerning the uncorrupted version of Auerbach et al. [4].

Theorem 12. *Let* KEM *be as above. Then there is an SFBB reduction* $\mathbb{B}$ *such that, for all* $\mathbb{A}$*,*

$$\mathsf{Adv}^{(n,\kappa)\text{-ow-pca}\star}_{\text{KEM}}(\mathbb{A}) \leq \mathsf{Adv}^{(n,\kappa)\text{-gapcdh}\star}_{\text{low-gran}}(\mathbb{B})\,.$$

If $\mathbb{A}$ *calls its challenge oracle* q *times, then* $\mathbb{B}$ *draws* q *uniformly random group elements.*

Proof (sketch). The reduction matches public keys to the received $\mathtt{X}[i]$ and forwards them to $\mathbb{A}$. Given that the multi-instance OW-PCA game (Fig. 11) is multi-challenge, the reduction relies on random self-reducibility to provide multiple challenges per instance: for each challenge oracle call to instance i, $\mathbb{B}$ samples $w \leftarrow\!\!\$\ \mathbb{Z}_{p_i}$ and sets $c \leftarrow \mathtt{Y}[i] \cdot g_i^w$, saving w in list $\mathtt{W}_i$ for later. Calls to the oracle $\mathcal{P}$ are forwarded to oracle $\mathcal{DDH}$, and likewise for corruption calls.

Once $\mathbb{A}$ halts with output $(\mathtt{I}, (j_i, \hat{K}_i)_{i\in\mathtt{I}})$, for each $i \in \mathtt{I}$, the reduction retrieves $w \leftarrow \mathtt{W}_i[j_i]$ and sets $\hat{\mathtt{Z}}[i] \leftarrow \hat{K}_i/\mathtt{X}[i]^w$, finally halting with output $\hat{\mathtt{Z}}$. If $\mathbb{A}$ returned correct ephemeral keys, then for each i we have $\hat{K}_i = c^x = g_i^{xy} g_i^{xw}$, so that dividing by $\mathtt{X}[i]^w = g_i^{xw}$ produces $\hat{\mathtt{Z}}[i] = g_i^{xy}$, as desired.

As the group is cyclic and g_i^w are independently sampled, uniformly random group elements, the $\mathtt{Y}[i] \cdot g_i^w$ are independently uniform too; thus, the simulation is faithful, and $\mathbb{B}$ wins iff $\mathbb{A}$ wins. □

For high and mid granularity, an immediate adaptation of AGK's bounds to include corruptions does not seem possible. Take their Theorem 5, relating the (uncorrupted) high granularity n-out-of-κ MI-GapCDH to a (n, n) multi-instance gap discrete logarithm problem, for which they give a useful bound. The proof constructs a reduction $\mathbb{B}$ that obtains n discrete logarithm challenges $\mathtt{Z}$ and injects them into the κ CDH challenges, which are then given to the adversary. This injection is done in such a way that possession of any n-sized subset of solutions to the κ challenges allows for the reconstruction of the original n DL solutions.

This strategy breaks down in the presence of corruptions, as answering a corrupting query on any GapCDH instance would necessitate to compromise one of the DL challenges the reduction seeks to break. (It is possible that this could be patched by having the adversary guess beforehand the instances that will be corrupted, although this would likely lead to a looser bound.) We therefore leave the hardness of mid and high granularity MI-GapCDH as interesting open problems, with high granularity being of particular relevance, as it most closely matches typical deployment practice.

Suppl. Mat. 4 On the Scaling Factor

As argued in the introduction, we want a guarantee that breaking n PKE instances can not be done much more efficiently than breaking each independently. Ideally, then, the advantages should experience an

exponential dampening with increasing n, as achieved by the KEM of Suppl. Mat. 3; our main results can then be interpreted to say that for any given n, our constructions inherits this smallness of the underlying KEM.

Another way to capture this intuition is to say that any adversary should have to spend n times as much computing time to break n instances, than that needed to break 1 system (for some suitable notion of "breaking"). This intuition was formalized asymptotically by AGK as the Scaling Factor (SF). Informally:

$$\mathsf{SF}_{\mathrm{PKE}}^{(n,\kappa)} = \frac{\text{resources necessary to break } n \text{ out of } \kappa \text{ instances}}{\text{resources necessary to break 1 out of 1 instance}} .$$

Assume that $\mathsf{SF}_{\mathrm{KEM}}^{(n,\kappa)} \geq \eta$ for some $\eta \geq 1$; another way to interpret our results would then be to show that it follows that $\mathsf{SF}_{\mathrm{PKE}}^{(n,\kappa)} \geq \eta$ for each of our constructions. As our results are formulated in the concrete-security setting, we do not give a formal definition of the scaling factor and a proof of inheritance, although we sketch the argument below.

In order to conclude that a construction inherits the scaling factor of the underlying KEM, we would have to show two things. Firstly, that

$$\begin{aligned}&\text{resources necessary to break } n \text{ out of } \kappa \text{ PKE instances}\\ &\qquad\geq \text{resources necessary to break } n \text{ out of } \kappa \text{ KEM instances}\,;\end{aligned}$$

this follows (with a small loss) for our main construction from Corollary 4. Secondly, that

$$\begin{aligned}&\text{resources necessary to break 1 KEM instance}\\ &\qquad\geq \text{resources necessary to break 1 out of 1 PKE instance}\,.\end{aligned}$$

Then, it follows from

$$\begin{aligned}&\text{resources necessary to break } n \text{ out of } \kappa \text{ KEM instances}\\ &\qquad\geq \eta\,(\text{resources necessary to break 1 out of 1 KEM instance})\end{aligned}$$

that

$$\begin{aligned}&\text{resources necessary to break } n \text{ out of } \kappa \text{ PKE instances}\\ &\qquad\geq \eta\,(\text{resources necessary to break 1 out of 1 PKE instance})\ ,\end{aligned}$$

which, when dividing both sides by the single-instance resources (and omitting negligible terms) yields $\mathsf{SF}_{\mathrm{PKE}}^{(n,\kappa)} \geq \eta$.

However, the second point is somewhat counterintuitive as it seemingly requires showing a "reverse reduction", namely that a break against the KEM results in a break against the PKE. In other words, if the PKE inherits the multi-instance security of the KEM but itself is already harder to break than the KEM, the scaling factor might be reduced. Another subtlety is that the security notion used for the KEM might also cause unexpected complications.

Specifically for our constructions and proofs, the reverse reduction for Theorem 6 is straightforward. On the other hand, the reverse reduction for Theorem 9 already takes a bit more work: a reduction could simply forward the first part of a challenge ciphertext to the KEM adversary and use its decryption oracle to simulate the KEM's plaintext checking oracle.

In other words, we are confident that for our main construction, the scaling factor is essentially preserved. Nonetheless, as the scaling factor remains underdeveloped both in the concrete security setting and in the context of primitive-to-construction inheritance, we refrain from providing a full formal definition and analysis of the scaling factor and instead opted for a more classical interpretation of the advantages we achieve (Corr. 5).