Count Corruptions, Not Users: Improved Tightness for Signatures, Encryption and Authenticated Key Exchange

Mihir Bellare[1] Doreen Riepel[2] Stefano Tessaro[3] Yizhao Zhang[4]

August 8, 2024

Abstract

In the multi-user with corruptions (muc) setting there are $n \geq 1$ users, and the goal is to prove that, even in the face of an adversary that *adaptively* corrupts users to expose their keys, un-corrupted users retain security. This can be considered for many primitives including signatures and encryption. Proofs of muc security, while possible, generally suffer a factor n loss in tightness, which can be large. This paper gives new proofs where this factor is reduced to the number c of corruptions, which in practice is much smaller than n. We refer to this as corruption-parametrized muc (cp-muc) security. We give a general result showing it for a class of games that we call local. We apply this to get cp-muc security for signature schemes (including ones in standards and in TLS 1.3) and some forms of public-key and symmetric encryption. Then we give dedicated cp-muc security proofs for some important schemes whose underlying games are not local, including the Hashed ElGamal and Fujisaki-Okamoto KEMs and authenticated key exchange. Finally, we give negative results to show optimality of our bounds.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: mihir@eng.ucsd.edu. URL: http://cseweb.ucsd.edu/~mihir/. Supported in part by NSF grant CNS-2154272 and KACST.

[2] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: driepel@ucsd.edu. Supported in part by KACST.

[3] Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA, USA. Email: tessaro@cs.washington.edu. Supported in part by NSF grants CNS-2026774, CNS-2154174, a JP Morgan Faculty Award, a CISCO Faculty Award, and a gift from Microsoft.

[4] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: yiz191@ucsd.edu.

Contents

1 Introduction

In practice, keys can be exposed, through system infiltration by hackers or phishing attacks; a striking example is the exposure of a Microsoft signing key to the Storm-0558 threat actor in July 2023 [Whi23]. This motivates the multi-user-with-corruptions (muc) setting, where there are $n \geq 1$ users, each holding some secret key, and the goal is to prove that, even in the face of an adversary that *adaptively* corrupts users to expose their keys, un-corrupted users retain security. (The last means their signatures remain unforgeable, ciphertexts encrypted to them retain privacy or whatever else the underlying primitive decrees.)

The good news is that proving muc-security is possible. The bad is that, in general —and in particular for canonical and standardized schemes such as the Schnorr signature scheme [Sch91]— current reductions lose a factor of the *total* number n of users, which can be very large. This leaves implementations to either use larger security parameters (inefficient), ignore corruptions (dangerous) or turn to special schemes that, while offering tight reductions, are less efficient than the canonical schemes [BHJ+15, HJK+21, DGJL21]. And meanwhile, dauntingly and disappointingly, negative results [BJLS16, JSSOW17] appear to say that the factor n loss is unavoidable for the canonical schemes.

Counting corruptions. In practice, corruptions certainly happen, but we suggest that the *number* of successful corruptions will likely be small, much smaller than the total number of users. Why is this? Because key-owners, recognizing the loss they face if their keys are exposed, are taking significant steps to prevent it. Important Internet services are increasingly storing their TLS signing keys in HSMs (Hardware Security Modules), which makes them harder to expose. Breaches lead to systems being hardened to prevent further breaches. (Microsoft, for example, has taken steps to prevent a repetition of the exposure of their signing key to Storm-0558 [Mic23].) Threshold cryptography is used to make key-exposure more difficult, a mitigation particularly popular for the signing keys securing digital wallets in cryptocurrencies. Employees and users are regularly trained to not fall prey to phishing.

Contributions in brief. Based on the above, this paper brings to muc security a new dimension, namely to view the *number of corruptions* as an adversary resource parameter. Denoting it c, we then give techniques and results that prove muc-security with tightness loss c rather than n. We refer to this as *corruption-parametrized muc (cp-muc) security*. Since c is in practice much smaller than n, cp-muc security allows theoretically-sound instantiation with practical security parameters.

We show cp-muc security for many primitives including signatures, encryption and authenticated key exchange. Our main results are *universal*, applying to *all* schemes, including the canonical ones. Existing negative results are not contradicted because they implicitly assume $n - 1$ corruptions, and we give new negative results to show the optimality of our new bounds.

Our inspiration and starting point is a technique of Coron [Cor00] used to show security with improved bounds for the RSA-FDH signature scheme of [BR96]. We generalize and improve this via a modular approach. First, we define and study a primitive we call a Hamming-Weight Determined (HWD) sampler. Using HWD samplers, our general cp-muc security theorem then gives a reduction from multi-user (mu) to cp-muc security that (1) loses only a factor c and (2) holds for *any* security game satisfying a condition we call *locality*. Intuitively, this means that the game does not make use of global secrets across users, such as a global challenge bit. This directly yields cp-muc security for many schemes and goals, and avoids repeating similar proofs across them. For important games that are not local, we go on to give dedicated proofs of cp-muc security. In particular, we do so (for security games with a global challenge bit) for the Hashed ElGamal and Fujisaki-Okamoto KEMs .

BROADER CONTEXT. The idea of assuming a bound c on the number of corruptions, amongst some larger number n of users, arises, and is indeed the basis for security, in some other and prior contexts in cryptography. The most prominent example is certainly secret sharing [Sha79, Bla79], where c is the threshold. Thence it enters multi-party computation [GMW87] and threshold cryptography [DDFY94]. With cp-muc, we are bringing this classical perspective to a broader setting, and to basic primitives like signatures, encryption and authenticated key exchange.

1.1 Spotlight on signatures

Our techniques and results are general and yield improvements for many goals and schemes. The ideas are however best introduced via a concrete example. We'll use signatures, for a number of reasons. First, there are, in the wild, over 250 million TLS signing keys [Dra23]. Exposure is a real threat, making as-tight-as-possible muc-security of immediate practical interest. Second, muc-secure signatures with as-tight-as-possible security are an assumption and current bottleneck for tightly-secure authenticated key exchange [CCG+19, DG21, DJ21, HJK+21]. Our proofs of cp-muc security for signature schemes will help to fill these gaps.

THE SETTINGS. Recall that classical definitions of security consider a single-user (su) setting. For a signature scheme Sig, recall this means that the game (called UF, for unforgeability) generates a single pair (vk, sk) consisting of a verification key and associated signing key. The adversary gets an oracle SIGN to sign a message of its choice under sk, and wins if it produces a new message M and a valid signature of M under vk [GMR88]. In the multi-user (mu) setting [BBM00], there are $n \geq 1$ users. For signatures [MS04], the game generates n independent key pairs $(vk_1, sk_1), \ldots, (vk_n, sk_n)$. The signing oracle SIGN now takes a user index i in addition to a message and returns a signature of the message under sk_i, and to win the adversary would point to some user i and produce a new message M, and a valid signature for it, under vk_i. The multi-user with corruptions (muc) game augments the mu game with an oracle CORRUPT that takes a user index i and returns its secret key sk_i. The signing oracle SIGN is as before, and the winning forgery is required to be for an uncorrupted user.

By $\epsilon^{\text{uf-su}}_{\mathsf{Sig}}$, $\epsilon^{\text{uf-mu-}n}_{\mathsf{Sig}}$ and $\epsilon^{\text{uf-muc-}n}_{\mathsf{Sig}}$ we denote the adversary advantage (success probability) in the su, mu and muc games, respectively. Of course, this quantity depends on the running time, but we will consider reductions that preserve this time and omit it from the notation here. The body of the paper is more precise.

UNIVERSAL RESULTS. We are interested, first, in results that are *universal*, meaning hold for *all* signature schemes Sig. Indeed, universal results are valuable both theoretically (in terms of understanding the notions) and in practice (they apply to all schemes, including already implemented and standardized ones). An archetypal such result [BBM00, MS04] (shown by a hybrid, also called guess-then-simulate, argument) is that $\epsilon^{\text{uf-mu-}n}_{\mathsf{Sig}} \leq n \cdot \epsilon^{\text{uf-su}}_{\mathsf{Sig}}$. The proof turns out to be able to easily handle corruptions, yielding the only known universal result for muc security:

$$\underline{\text{P1}}: \ \forall\, \mathsf{Sig} \ : \ \ \epsilon^{\text{uf-muc-}n}_{\mathsf{Sig}} \leq n \cdot \epsilon^{\text{uf-su}}_{\mathsf{Sig}} \ . \tag{1}$$

We ask if alternative universal results are possible. Given that mu security has been extensively studied [Ber15, KMP16, GJKW07, HJ12, HS16, Lac18, PR20], we suggest to use it, rather than su, as the starting point. Let e be Euler's number. Our first universal result is then:

$$\underline{\text{N1}}: \ \forall\, \mathsf{Sig} \ : \ \ \epsilon^{\text{uf-muc-}(n,c)}_{\mathsf{Sig}} \leq e \cdot (c+1) \cdot \epsilon^{\text{uf-mu-}n}_{\mathsf{Sig}} \ . \tag{2}$$

The term on the left above is the cp-muc advantage where the adversary is restricted to c queries to oracle CORRUPT. Eq. (2) says that, in moving from the setting with no corruptions to the setting

with c corruptions, adversary advantage grows by at most a factor about c, *regardless of the number n of users* and *for all signature schemes* Sig. But we can do better. In Eq. (2), muc security for n users (and c corruptions) is provided assuming mu security for the same number n of users. It turns out, curiously, that we can (substantially) reduce the number of users, now denoted m, for which mu security is assumed. Our second universal result (Theorem 5.1) is:

$$\underline{\text{N2}}: \ \forall\, \mathsf{Sig} \ : \ \ \epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)} \leq e \cdot (c+1) \cdot \epsilon_{\mathsf{Sig}}^{\text{uf-mu-}m} \ \text{ for } m = \lfloor (n-1)/(c+1) \rfloor. \tag{3}$$

How and when the new N1, N2 results yield improvements over the prior P1 may not be obvious due to the starting points being different (mu for the former and su for the latter). The following will clarify this.

Specialized results. These are results that hold for *particular* (but not all) schemes. Let's call Sig *tightly-mu-secure* if (roughly) $\epsilon_{\mathsf{Sig}}^{\text{uf-mu-}n} = \epsilon_{\mathsf{Sig}}^{\text{uf-su}}$. In an important class of specialized results, prior work [Ber15,KMP16,GJKW07,HJ12,HS16,Lac18,PR20] shows that many schemes —including the Schnorr scheme [Ber15,KMP16]— have tight mu security. The proofs, however, exploit algebraic self-reducibility amongst keys and cannot tolerate corruptions, so that, even for these schemes, for muc security, Eq. (1) remains the best known bound. We offer a substantial improvement; Eq. (2) implies that

$$\underline{\text{N3}}: \text{ For all tightly-mu-secure } \mathsf{Sig}: \ \epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)} \leq e \cdot (c+1) \cdot \epsilon_{\mathsf{Sig}}^{\text{uf-su}} \ . \tag{4}$$

That is, the muc advantage degrades by at most a factor about c even relative to the (standard) su advantage, again regardless of n. Eq. (4) holds in particular for the Schnorr signature scheme.

While it would be desirable to show that su security implies cp-muc security with a loss c in general and for all schemes, such a statement does not seem possible without additional assumption about the scheme. Therefore, our results can be viewed as a middle ground, either (using N3) applying to tightly-mu-secure schemes or (using N2) reducing to mu security for a number of users m substantially smaller than n.

Numerical example. Let Sig be the Schnorr scheme over a size p elliptic curve group $\mathbb{G}$. Suppose we target 128-bit security for $n = 2^{30}$ users with $c = 2^{10}$ corruptions, and let t be the running-time of the adversary. Assuming discrete-logarithm computation over $\mathbb{G}$ is the best attack, we have $\epsilon_{\mathsf{Sig}}^{\text{uf-su}} \leq t^2/p$, which by the prior P1 result of (1) yields $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}n} \leq nt^2/p$. Now 128-bit security requires $nt^2/p \leq t/2^{128}$ or $p = 2^{128} \cdot nt$. This means we use a group $\mathbb{G}_1$ of size $p_1 = 2^{128} \cdot nt$. Meanwhile the tight mu security of Sig implies $\epsilon_{\mathsf{Sig}}^{\text{uf-mu-}n} \leq t^2/p$ which by our new N1 result of (2) (and ignoring the e factor) yields $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)} \leq ct^2/p$, and $ct^2/p \leq t/2^{128}$ now yields $p = 2^{128} \cdot ct$, so we use a group $\mathbb{G}_2$ of size $p_2 = 2^{128} \cdot ct$. Now note that $p_2/p_1 = c/n = 2^{-20}$ so we have dropped the group size by a factor of 2^{20} while retaining security. For example if $p_1 = 2^{256}$ then $p_2 = 2^{236}$. Exponentiation takes time cubic in the logarithm of the group size, so in this case we conclude that the cost of signing or verifying for Sig over $\mathbb{G}_1$ is $(236/256)^3 = 0.78$ times that in $\mathbb{G}_2$, meaning we have reduced the running time of the implemented scheme by 22%, a significant gain in practice, while retaining security.

Standards and key exchange. The best muc-security bound we have for the standardized RSA-SSAPSS [Nat23], EdDSA [BDL+12, Nat23, JL17] and ECDSA [Nat23] signature schemes is the generic one of Eq. (1) with its factor n loss. This lead tight proofs for the TLS 1.3 key exchange [DG21, DJ21] to simply assume tight muc security —meaning $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}n} = \epsilon_{\mathsf{Sig}}^{\text{uf-su}}$— for these schemes. Can we do better? Unfortunately, Result N3 (Eq. (4)) will not help since these schemes have evaded proofs of tight mu security. However, Result N2 (Eq. (3)) is helpful here. It says that assuming $\epsilon_{\mathsf{Sig}}^{\text{uf-mu-}m} = \epsilon_{\mathsf{Sig}}^{\text{uf-su}}$ for some small number m of users, we get $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)} \leq c \cdot \epsilon_{\mathsf{Sig}}^{\text{uf-su}}$. While

this is still a non-standard assumption, it is better than directly assuming $\epsilon^{\text{uf-muc-}n}_{\mathsf{Sig}} = \epsilon^{\text{uf-su}}_{\mathsf{Sig}}$. We only need to assume mu (rather than muc) security, and that too for a small number of users m.

TIGHT MUC SECURITY. The absence of results better than P1 (Eq. (1)) for standardized schemes lead researchers to develop new signature schemes that are tightly muc secure [BHJ+15, GJ18, HJK+21, DGJL21, PW22, HLG23]. (As above this means $\epsilon^{\text{uf-muc-}n}_{\mathsf{Sig}} = \epsilon^{\text{uf-su}}_{\mathsf{Sig}}$.) In terms of just the bound, this is better than cp-muc security. But canonical schemes (Schnorr and standardized ones) are not known to be in this class so there is no improvement for in-use signatures or TLS 1.3 key exchange. Moreover, the key sizes and computation time of the new schemes is larger than that of the canonical schemes. Hence, cp-muc security is a pragmatic alternative.

TECHNIQUES. We outline the simplest case of our proof technique, specialized to signatures and for the weaker of our two universal results, namely N1 (Eq. (2)). Given a cp-muc adversary $\mathcal{A}$ with advantage $\epsilon^{\text{uf-muc-}(n,c)}_{\mathsf{Sig}}$, we want to construct a mu adversary $\mathcal{B}$, with advantage $\epsilon^{\text{uf-mu-}n}_{\mathsf{Sig}}$ and about the same running time as $\mathcal{A}$, such that Eq. (2) holds. Letting p be a parameter in the range $0 \le p \le 1$, adversary $\mathcal{B}$ picks a vector $(c_1, \ldots, c_n)$ of independent, p-biased bits. (That is, each c_i is 1 with probability p and 0 with probability $1-p$.) We'll say user i is red if $c_i = 1$ and let $R \subseteq \{1, \ldots, n\}$ be the set of red users; correspondingly i is blue if $c_i = 0$ and B is the set of blue users. Now, $\mathcal{B}$ runs $\mathcal{A}$. In answering $\mathcal{A}$'s oracle queries, $\mathcal{B}$ simulates red users directly and forwards queries for blue users to its own mu game. In more detail, $\mathcal{B}$ generates a signing-verifying key pair (vk_i, sk_i) for each $i \in R$, allowing it to easily answer both SIGN and CORRUPT queries to i. It answers SIGN queries for $i \in B$ via its own SIGN oracle. The difficulty is a CORRUPT(i) query for $i \in B$; adversary $\mathcal{B}$ has no way to answer this, and aborts. If it does not abort, then $\mathcal{A}$ outputs a user j (called the forgery victim) and a forgery under vk_j. If $j \in B$ then $\mathcal{B}$ returns this to win its mu game, else it aborts. Let GD be the event that $i \in R$ for all CORRUPT(i) queries of $\mathcal{A}$ and also the forgery victim j is in B. The probability of GD can be computed as $f(p) = p^c(1-p)$ (Theorem 3.2) and a careful analysis (that we make precise via Lemma 2.2) shows that GD is independent of the success of $\mathcal{A}$, so that $\epsilon^{\text{uf-mu-}n}_{\mathsf{Sig}} \ge f(p) \cdot \epsilon^{\text{uf-muc-}(n,c)}_{\mathsf{Sig}}$. Calculus shows that $f(p)$ is maximized at $p = 1 - 1/(c+1)$ where $f(p) \ge 1/(e(c+1))$ (Theorem 3.2), yielding Eq. (2). For some intuition, note that with this choice the expected number of red users is $(cn-1)/(c+1)$, meaning a high number of red users are needed to successfully answer the small number c of CORRUPT queries.

This adapts Coron's [Cor00] proof of the security of the RSA-FDH signature scheme. However, his setting has only one user and no secret-key exposing corruptions; what for us are "users" and "corruptions" are for Coron messages queried to the random oracle and signing queries, respectively.

THE PATH AHEAD. We generalize and improve this with a modular approach. First, we introduce and study HWD samplers, as a way to find the best way to sample the vector $(c_1, \ldots, c_n)$. Second, we give a framework and general cp-muc security theorem that shows how to use any HWD sampler to promote mu security to cp-muc security for a large class of games satisfying a condition we call locality. Numerous applications, as well as improvements such as Eq. (3), are then obtained, some directly, some with more dedicated work.

1.2 HWD samplers and general cp-muc theorem

HWD SAMPLERS. We ask whether the above manner of choosing $(c_1, \ldots, c_n)$ is optimal. The answer is no; we can do better. To get there, we start with an abstraction, viewing the vector $(c_1, \ldots, c_n) \leftarrow\!\!\$\ \mathsf{D}(n, c)$ as being chosen by an algorithm D that we call a sampler. We identify, as a sufficient condition on D for the (above) simulation to work, that the probability of a vector $(c_1, \ldots, c_n)$ depends only on its Hamming weight, and call such samplers Hamming-

Weight Determined (HWD). Section 3 associates to any HWD sampler a success probability α and an error probability β. Continuing to use signatures as an example, Theorem 4.1 implies that $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)} \leq (1/\alpha) \cdot \epsilon_{\mathsf{Sig}}^{\text{uf-mu-}m}$ for an m that grows with β, so we want to make α large and β small. Now the way of sampling $(c_1, \ldots, c_n)$ discussed in Section 1.1 corresponds to a particular choice of D, that we call the biased-coin sampler and analyze in Theorem 3.2; it has good success probability but unfortunately high error probability. We then give a new sampler that we call the fixed-set-size sampler. Theorem 3.3 shows that it has not only optimal success probability but zero error probability, which is crucial to obtaining the improved Eq. (3). Figure 3 shows some numbers.

General framework and theorem. We now ask how far these techniques will generalize beyond signatures. We seek to show a result of the form "For all security games in a certain class, mu security can be promoted to cp-muc security with loss only a factor about c." To do this rigorously, we first (in Section 4) define something we call a *formal security specification* (FSS). Denoted Π, it is simply an algorithm that, given a scheme Sch, specifies the initializations and oracle-response computations (including for Corrupt queries) for the intended target notion of security for Sch. This leads naturally to an actual (code-based) game $\mathbf{G}_{\mathsf{Sch}}^{\Pi}$ defining su security (Figure 4), and thence to games $\mathbf{G}_{\mathsf{Sch}}^{\Pi\text{-mu-}n}$ and $\mathbf{G}_{\mathsf{Sch}}^{\Pi\text{-muc-}(n,c)}$ capturing mu and cp-muc security, respectively, where n is the number of users and c the number of allowed corruptions (Figure 5). For example, Figure 4 shows an FSS UF such that, when $\mathsf{Sch} = \mathsf{Sig}$ is a signature scheme, the corresponding games are exactly the standard su, mu and cp-muc games for signatures discussed above. The su, mu and muc versions of many other notions in the literature can in this way be recovered by simply giving a single FSS for the notion, as we will see. Let $\epsilon_{\mathsf{Sch}}^{\Pi\text{-su}}$, $\epsilon_{\mathsf{Sch}}^{\Pi\text{-mu-}n}$ and $\epsilon_{\mathsf{Sch}}^{\Pi\text{-muc-}(n,c)}$ denote the su, mu and muc adversary advantages, respectively, for FSS Π with scheme Sch.

In Section 4 we define a condition on an FSS that we call *locality*; roughly it means that in the mu and muc settings, different users share no common secret unknown to the adversary. This is what is needed for our technique to work. The main result is Theorem 4.1, showing how to use any HWD sampler D to promote mu to cp-muc for any local FSS Π for a scheme Sch, and giving bounds in terms of quantities related to D. The proof uses game playing including a crucial use of the Second Fundamental Lemma (Lemma 2.2). For applications however, it is easier to use Theorem 4.2, which sets D to the optimal fixed-set-size sampler to say that for any local FSS Π for a scheme Sch, we have

$$\epsilon_{\mathsf{Sch}}^{\Pi\text{-muc-}(n,c)} \leq e \cdot (c+1) \cdot \epsilon_{\mathsf{Sch}}^{\Pi\text{-mu-}m} \quad \text{for } m = \lfloor (n-1)/(c+1) \rfloor. \tag{5}$$

As an example, FSS UF for signature schemes is local, so this immediately yields Eqs. (2),(3). We now turn to further applications.

1.3 Applications

An overview of our applications is in Figure 1. The results are of the form $\epsilon_{\mathsf{TS}}^{\text{TG-}(n,c)} \leq c \cdot \epsilon_{\mathsf{SS}}^{\text{SG-}m}$ where $\mathrm{TG}, \mathrm{SG}, \mathsf{TS}, \mathsf{SS}$ are the target goal, starting goal, target scheme and starting scheme, respectively. That is, TG-security of TS for n users and c corruptions is shown with loss c assuming SG-security for SS for m users. What are "users" and "corruptions" depends on the goal, showcasing the breadth of our framework. The last column indicates the application type. Direct (D) means we show that an underlying FSS is local and apply Theorem 4.2. Indirect (I) means the FSS is *not* local but we can nevertheless show the result via a dedicated proof that uses the general theorem in an intermediate step.

Direct applications. **D1** is the result for signatures (Theorem 5.1). **D2** is for IND-CCA security of KEMs in the multi-challenge-bit (mb) setting [HS21] (Theorem 5.2). **D3** is for OW-PCVA (one-

	Target		Start		m	Type
	TG	TS	SG	SS		
D1	UF-muc	Sig	UF-mu	Sig	$\lfloor(n-1)/(c+1)\rfloor$	D
D2	CCA-MB-muc	KEM	CCA-MB-mu	KEM	$\lfloor(n-1)/(c+1)\rfloor$	D
D3	OW-PCVA-muc	PKE	OW-PCVA-mu	PKE	$\lfloor(n-1)/(c+1)\rfloor$	D
D4	AE-MB-muc	SE	AE-MB-mu	SE	$\lfloor(n-1)/(c+1)\rfloor$	D
SB1	CCA-SB-muc	HEG	St-CDH	$(\mathbb{G}, p, g)$	1	I
SB2	CCA-SB-muc	KEM	CPA-SB-mu	PKE	$\lfloor(n-1)/(c+1)\rfloor$	I
KE1	IND-WFS	CCGJJ	St-CDH	$(\mathbb{G}, p, g)$	$\lfloor(n-1)/(c+1)\rfloor$	I
KE2	IND-FFS	AKE	UF-mu	Sig	$\lfloor(n-1)/(c+1)\rfloor$	I
SO1	SIM-SO-CCA	DH-PKE	St-CDH	$(\mathbb{G}, p, g)$	1	I
SO2	SIM-SO-CCA	RSA-PKE	RSA	RSAGen	1	I

Figure 1: **Overview of our applications:** We show that $\epsilon_{\mathsf{TS}}^{\text{TG-}(n,c)} \leq c \cdot \epsilon_{\mathsf{SS}}^{\text{SG-}m}$ where TS, TG are the target scheme and goal, and SS, SG are the starting scheme and goal. **D1-D4** are direct applications of our general cp-muc security theorem; the rest are indirect. **SB1-SB2** relate to encryption in the single-bit setting. **KE1-KE2** are for (authenticated) key exchange. **SO1-SO2** are for selective opening security.

way security under plaintext checking and ciphertext validity attacks) security of PKE [HHK17] (Theorem 5.3), which will be used in our results for the FO transform. **D4** is for (nonce-based) authenticated encryption in the mb setting (Theorem 5.4), demonstrating the generality of our approach in that our results also apply in the symmetric setting. In each case we observe that the underlying FSSs, denoted UF, CCA-MB, OW-PCVA and AE-MB respectively, are local, yielding the results via Theorem 4.2. We stress that these rows are purely illustrative; there are far more such applications than we can exhaustively list.

Indirect applications. In the definition of mu security for encryption (KEM, PKE) from [BBM00], there is a single challenge bit across all users. Therefore, the underlying FSS CCA-SB is *not* local. However, this is recognized as the leading and stronger definition [GKP18] so rather than give up on it, we ask what one can show. Our answer is to show cp-muc security for particular, important schemes via dedicated proofs, making use of our general theorem for intermediate steps. We first prove such a result (**SB1**, Theorem 5.5) for the hashed ElGamal KEM HEG, assuming strong computational Diffie-Hellman (St-CDH) [ABR01]. Then, we use the modular FO transform [HHK17] to get a more general result (**SB2**, Theorem 5.8). In particular, we show how to turn a CPA-SB-mu PKE scheme into a CCA-SB-muc KEM. For this, we first go tightly from CPA-SB-mu to OW-PCVA-mu. Next, we exploit locality of the corresponding FSS OW-PCVA and use Theorem 4.2 to go to OW-PCVA-muc. Finally, we go tightly from the latter to CCA-SB-muc.

As a natural extension, we then turn to authenticated key exchange (AKE). Security games for AKE (e. g., [BR94]) also use a single challenge bit and are not local, but we can still give several cp-muc security results. The Diffie-Hellman based CCGJJ protocol [CCG+19] is shown in [CCG+19] to achieve weak forward secrecy (wfs) [Kra05], with a factor n loss from St-CDH. We show (**KE1**) that by considering our corruption-parameterized approach for AKE, we can reduce the loss to the number c of corruptions. Previous work on tightly-secure AKE [GJ18, LLGW20, HJK+21], including analyses of the TLS protocol [DJ21, DG21], all use muc-secure signatures as a building block. Our results on signatures allow AKE security based on UF-mu secure schemes (**KE2**) which yields tightness improvements when using the canonical and standardized signature schemes actually used in practice.

We then extend our indirect results to applications in a quite different setting, looking at selective opening security for PKE. In simulation-based selective opening (SIM-SO-CCA) security for PKE [BHK12, BDWY12], the adversary can reveal randomness underlying encrypted messages. Practical Diffie-Hellman and RSA-based PKE schemes are shown in [HJKS15] to be SIM-SO-CCA, with loss factor the total number n of ciphertexts encrypted. Modeling randomness reveal as a "corruption" and each encryption as a "user" in our framework allows us (with some extra steps) to reduce the loss to the number c of openings (**SO1-SO2**, Theorems C.1 and C.2). Finally, of pedagogic interest but no novelty, we show (Theorem A.2, Appendix A) how our framework can be used to recover Coron's result [Cor00].

1.4 Optimality results

We are interested in the optimality of our results, namely to show that the factor c loss is optimal for a number of schemes and games. Previous work studied optimality based on *non-interactive* complexity assumptions [BJLS16, HLG21], *bounded-round* assumptions [MPS20] or tailored to specific primitives or protocols [JSSOW17, CCG+19, GGJJ23]. However, a mu security game is mostly interactive and not bounded. Similar to our positive result, we aim for a general result capturing a variety of games.

Witness recovery games. More concretely, we consider the goal of proving security, via black-box reductions, for a *recovery* game $\mathbf{G}_{\mathsf{Rel}}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}$ parameterized by an efficiently verifiable relation Rel. After learning n inputs $x_1, \ldots, x_n$, the adversary is allowed to learn witnesses (with respect to Rel) for up to c of them, and finally wins if it recovers a witness for one of the remaining inputs. As already observed by [BJLS16], several mu security games $\mathbf{G}'$ allow to define a suitable relation Rel such that $\mathbf{G}_{\mathsf{Rel}}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}$ can be seen as a *restriction* of the set of allowable adversaries in $\mathbf{G}'$. Examples include relations for PKE and signatures, where the decryption resp. signing key serves as the witness.

Stateless games. Our first general result shows that any black-box reduction from an interactive *stateless* game $\mathbf{G}$ to $\mathbf{G}_{\mathsf{Rel}}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}$ must incur a loss $c+1$ when Rel is efficiently randomizable (Theorem 6.1). The same loss is hence also necessary if we instead reduce to a game $\mathbf{G}'$ for which $\mathbf{G}_{\mathsf{Rel}}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}$ can be seen as an adversarial restriction. Here, by stateless we mean that the game initially sets a state, but then oracle queries do not alter it. This abstraction serves a trade-off between simplicity and generality. Indeed, several games are stateless, such as CPA security for public-key encryption in both the multi-challenge-bit as well as in the single-challenge-bit setting.

However, since many games of interest are not stateless, we extend our result to relax the stateless condition on $\mathbf{G}$. This generalization allows the analysis of stateful games and, more concretely, we show optimality of the loss factor c for the IND-CCA game for PKE and the strong unforgeability game for randomized signatures. (Assuming ciphertexts and signatures have sufficient entropy.)

2 Preliminaries

Notation. If $\boldsymbol{w}$ is a vector then $|\boldsymbol{w}|$ is its length (the number of its coordinates) and $\boldsymbol{w}[i]$ is its i-th coordinate, where we start with $i = 1$. Strings are identified with vectors over $\{0,1\}^*$, so that $|Z|$ denotes the length of a string Z and $Z[i]$ denotes its i-th bit. By ε we denote the empty string or vector. By $x\|y$ we denote the concatenation of strings x, y. If x, y are equal-length strings then $x \oplus y$ denotes their bitwise xor. If S is a finite set, then $|S|$ denotes its size. For integers $a \le b$ we let $[a..b]$ be shorthand for $\{a, \ldots, b\}$. The notation $[\![B]\!]$ for a boolean statement B returns true if B is true and false otherwise. When we write e, we mean Euler's number.

If X is a finite set, we let $x \leftarrow\!\!\$\, X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. If A is an algorithm, we let $y \leftarrow\!\!\$\, A[\mathrm{O}_1, \ldots](x_1, \ldots)$ denote running A on inputs $x_1, \ldots$, with oracle access to $\mathrm{O}_1, \ldots$, and assigning the output to y. We let $\mathbf{Out}(A[\mathrm{O}_1, \ldots](x_1, \ldots))$ denote the set of all possible outputs of A on this run. Running time is worst case, which for an algorithm with access to oracles means across all possible replies from the oracles. We use $\perp$ (bot) as a special symbol to denote rejection, and it is assumed to not be in $\{0,1\}^*$.

We may consider an algorithm A that takes some input *in* and a current state, and returns an output *out* and updated state, written $(out, st) \leftarrow\!\!\$\, A(in, st)$. We may refer to such an algorithm as stateful, but note that A is, syntactically, just an algorithm like any other. There is no implicit state maintained by the algorithm; it is the responsibility of the game executing A to maintain the state variable *st*, which it will do explicitly.

GAMES. We use the code-based game-playing framework of BR [BR06]. By $\Pr[\mathrm{G}(\mathcal{A}) \Rightarrow y]$ we denote the probability that the execution of game G with adversary $\mathcal{A}$ results in the game output (what is returned by FIN) being y, and write just $\Pr[\mathrm{G}(\mathcal{A})]$ for $\Pr[\mathrm{G}(\mathcal{A}) \Rightarrow \mathsf{true}]$. Different games may have procedures (oracles) with the same names. If we need to disambiguate, we may write G.O to refer to oracle O of game G. In games, integer variables, set variables, boolean variables and string variables are assumed initialized, respectively, to 0, the empty set $\emptyset$, the boolean false and $\perp$. For the following, recall that games G, H are identical-until-bad if their code differs only in statements that follow the setting of flag bad to true [BR06].

Lemma 2.1 [First Fundamental Lemma of Game Playing [BR06]] *Let* G, H *be identical-until-*bad *games. Then for any adversary* $\mathcal{A}$ *we have*

$$|\Pr[\mathrm{G}(\mathcal{A})] - \Pr[\mathrm{H}(\mathcal{A})]| \leq \Pr[\mathrm{H}(\mathcal{A}) \text{ sets } \mathsf{bad}] = \Pr[\mathrm{G}(\mathcal{A}) \text{ sets } \mathsf{bad}] \ .$$

Lemma 2.2 [Second Fundamental Lemma of Game Playing [BNN07]] *Let* G, H *be identical-until-*bad *games and let* GD *be the event that* bad *is not set to* true *in the execution of these games with adversary* $\mathcal{A}$*. Then*

$$\Pr[\mathrm{G}(\mathcal{A}) \wedge \mathrm{GD}] = \Pr[\mathrm{H}(\mathcal{A}) \wedge \mathrm{GD}] \ .$$

SIGNATURE SCHEMES. A signature scheme Sig allows generation of a verifying key vk and corresponding signing key sk via $(vk, sk) \leftarrow\!\!\$\, \mathsf{Sig.Kg}$. A signature of a message M is produced as $\sigma \leftarrow\!\!\$\, \mathsf{Sig.Sign}(sk, M)$. Verification returns a boolean $d \leftarrow \mathsf{Sig.Vf}(vk, M, \sigma)$. Correctness asks that for all M we have $\mathsf{Sig.Vf}(vk, M, \mathsf{Sig.Sign}(sk, M)) = \mathsf{true}$ with probability 1, where the probability is over the choice of keys and the coins of the signing algorithm, if any. To measure security we define the uf advantage of an adversary $\mathcal{A}$ as $\mathbf{Adv}^{\mathsf{UF\text{-}su}}_{\mathsf{Sig}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathsf{UF}}_{\mathsf{Sig}}(\mathcal{A})]$ where the game is shown on the bottom right of Figure 4.

PUBLIC-KEY ENCRYPTION SCHEMES. A public-key encryption (PKE) scheme PKE allows generation of a public encryption key ek and corresponding secret decryption key dk via $(ek, dk) \leftarrow\!\!\$\, \mathsf{PKE.Kg}$. We denote the message space by $\mathsf{PKE.MS}$ and require that it is a length-closed set (i. e., for any $m \in \mathsf{PKE.MS}$ it is the case that $\{0,1\}^{|m|} \subseteq \mathsf{PKE.MS}$). Encryption of a message $M \in \mathsf{PKE.MS}$ is via $C \leftarrow\!\!\$\, \mathsf{PKE.Enc}(ek, M)$. Decryption $M \leftarrow \mathsf{PKE.Dec}(ek, dk, C)$ returns a value $M \in \{0,1\}^* \cup \{\perp\}$. Correctness asks that $\Pr[\mathsf{PKE.Dec}(dk, \mathsf{PKE.Enc}(ek, M)) = M] = 1$ for every $(ek, dk) \in \mathbf{Out}(\mathsf{PKE.Kg})$ and every $M \in \mathsf{PKE.MS}$. We will consider several security definitions; they will be given in Section 5 and Appendices B and C.

KEY-ENCAPSULATION MECHANISMS. A key-encapsulation mechanism (KEM) KEM allows generation of a public encapsulation key ek and corresponding secret decapsulation key dk via $(ek, dk) \leftarrow\!\!\$$

$\mathsf{KEM.Kg}$. There is no message. Instead, encapsulation $\mathsf{KEM.Encaps}(ek)$ returns a symmetric key $K \in \{0,1\}^{\mathsf{KEM.kl}}$ and a ciphertext C encapsulating it. Decapsulation $\mathsf{KEM.Decaps}(dk, C)$ returns a value $K \in \{0,1\}^{\mathsf{KEM.kl}} \cup \{\perp\}$. Correctness asks that $\Pr[\mathsf{KEM.Decaps}(dk, C) = K] = 1$ for every $(ek, dk) \in \mathbf{Out}(\mathsf{KEM.Kg})$, where the probability is over $(C, K) \leftarrow\!\!\$\ \mathsf{KEM.Encaps}(ek)$. Security definitions are in Section 5.

3 HWD samplers

We introduce and define HWD samplers and their success and error probabilities. We give and analyze a natural HWD sampler called the biased-coin sampler and then give an optimal HWD sampler called the fixed-set-size sampler.

HWD SAMPLERS. A *sampler* is an algorithm D that on input a positive integer n and a non-negative integer $c < n$ returns an n-bit string $s \leftarrow\!\!\$\ \mathsf{D}(n, c)$. In our application, n will be the number of users and c will be the number of corruptions. For $s \in \{0,1\}^n$, let $R(s) = \{\, i \in [1..n] : s[i] = 1 \,\}$ and $B(s) = \{\, i \in [1..n] : s[i] = 0 \,\}$. We think of sampling $s \leftarrow\!\!\$\ \mathsf{D}(n, c)$ as coloring the points $1, 2, \ldots, n$, with point $i \in [1..n]$ colored red if $s[i] = 1$ and colored blue if $s[i] = 0$, so that $R(s)$ and $B(s)$ are the sets of red and blue points, respectively.

We say that sampler D is *Hamming-weight determined* (HWD) if for all n, c and all $s_1, s_2 \in \{0,1\}^n$ we have: if $\mathbf{w}_{\mathrm{H}}(s_1) = \mathbf{w}_{\mathrm{H}}(s_2)$ then s_1 and s_2 have the same probability of arising as outputs of $\mathsf{D}(n, c)$. This will allow our applications. Note that this condition is true if and only if there is a function W such that $\Pr[s = s' : s' \leftarrow\!\!\$\ \mathsf{D}(n, c)] = \mathrm{W}(n, c, \mathbf{w}_{\mathrm{H}}(s))$ for all $s \in \{0,1\}^n$ and all n, c. We refer to W as D's *Hamming-weight probability function*.

As another way of understanding this, if $\pi : [1..n] \to [1..n]$ is a permutation, let $\pi(s) \in \{0,1\}^n$ denote the string whose j-th bit is $s[\pi(j)]$ for $j \in [1..n]$. That is, the bits of s are permuted according to π. Now say that D is *permutation invariant* if for all n, c, all $s \in \{0,1\}^n$ and all permutations $\pi : [1..n] \to [1..n]$ we have that s and $\pi(s)$ have the same probability of arising as outputs of $\mathsf{D}(n, c)$. Then it is easy to see that D is permutation invariant if and only if it is Hamming-weight determined.

For a set $C \subseteq [1..n]$ of size $c < n$, and $i \in [1..n] \setminus C$, we let

$$\mathbf{P}_{\mathsf{D},n,c}(C, i) = \Pr[\, i \in B(s) \text{ and } C \subseteq R(s) \ : \ s \leftarrow\!\!\$\ \mathsf{D}(n, c)] \,. \tag{6}$$

This is the probability that $s \leftarrow\!\!\$\ \mathsf{D}(n, c)$ colors the points in C red and colors i blue. (How points not in $C \cup \{i\}$ are colored does not matter.) Our applications need a lower bound on it. Towards this, we define the *sampler success probability*

$$\mathbf{P}^*_{\mathsf{D}}(n, c) = \textstyle\sum_{j=0}^{n-c-1} \mathrm{W}(n, c, c+j) \cdot \binom{n-c-1}{j} \,. \tag{7}$$

The following says $\mathbf{P}_{\mathsf{D},n,c}(C, i)$ is determined by the success probability and thus in particular independent of the particular choices of C, i.

Theorem 3.1 *Let D be a Hamming-weight determined sampler with Hamming-weight probability function W. Suppose $0 \le c < n$. Then for any size c set $C \subseteq [1..n]$ and any $i \in [1..n] \setminus C$ we have $\mathbf{P}_{\mathsf{D},n,c}(C, i) = \mathbf{P}^*_{\mathsf{D}}(n, c)$.*

Proof: Let $E = \{\, s \in \{0,1\}^n : i \in B(s) \text{ and } C \subseteq R(s) \,\}$. Recall that the characteristic function $\chi_E : \{0,1\}^n \to \{0,1\}$ of set E is defined by: $\chi_E(s) = 1$ if $s \in E$ and $\chi_E(s) = 0$ if $s \notin E$. Let $H_\ell = \{\, s \in \{0,1\}^n : \mathbf{w}_{\mathrm{H}}(s) = \ell \,\}$ and $E_\ell = E \cap H_\ell = \{\, s \in H_\ell : \chi_E(s) = 1 \,\}$. Writing $\mathrm{P}_{\mathsf{D}}(s)$ for the

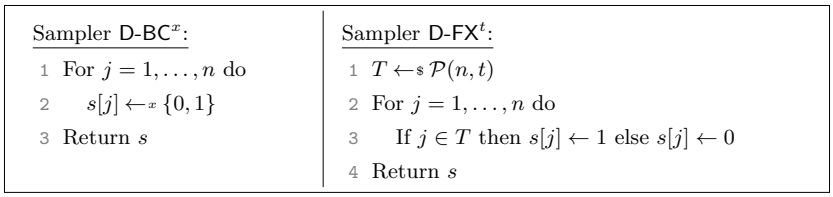

Figure 2: Our Hamming-Weight Determined Samplers. **Left:** Biased-coin sampler with probability parameter $x \in [0,1]$. **Right:** Fixed set size sampler with set-size parameter $t \in [0..n-1]$.

probability of s arising as an output of $\mathsf{D}(n,c)$, we have

$$\begin{aligned}\mathbf{P}_{\mathsf{D},n,c}(C,i) &= \sum_{s\in\{0,1\}^n} \chi_E(s)\cdot \mathrm{P}_{\mathsf{D}}(s) = \sum_{\ell=0}^{n}\sum_{s\in H_\ell} \chi_E(s)\cdot \mathrm{P}_{\mathsf{D}}(s)\\ &= \sum_{\ell=0}^{n}\sum_{s\in H_\ell} \chi_E(s)\cdot \mathrm{W}(n,c,\ell) = \sum_{\ell=0}^{n}\mathrm{W}(n,c,\ell)\sum_{s\in H_\ell}\chi_E(s)\\ &= \sum_{\ell=0}^{n}\mathrm{W}(n,c,\ell)\cdot|E_\ell|\ .\end{aligned}$$

The size of the set E_ℓ is 0 if $\ell < c$ or $\ell = n$, so we can confine the sum to $\ell = c,\ldots,n-1$. Now suppose $\ell \in \{c,\ldots,n-1\}$. Then $|E_\ell| = \binom{n-c-1}{\ell-c}$, because the c points in C must be red and i must be blue, leaving $n-c-1$ points from which to pick the remaining $\ell-c$ red points. Putting the above together, we have

$$\begin{aligned}\mathbf{P}_{\mathsf{D},n,c}(C,i) &= \sum_{\ell=c}^{n-1}\mathrm{W}(n,c,\ell)\cdot\tbinom{n-c-1}{\ell-c} = \sum_{j=0}^{n-c-1}\mathrm{W}(n,c,c+j)\cdot\tbinom{n-c-1}{j}\\ &= \mathbf{P}^*_{\mathsf{D}}(n,c)\ ,\end{aligned}$$

where the last equality is by Eq. (7). This completes the proof. ∎

Theorem 3.1 will be used in the proof of Theorem 4.1, our general cp-muc security result. Here it allows us to focus on the success probability as defined by Eq. (7). Now define the *error probability* of sampler D via

$$\mathbf{R}_{\mathsf{D}}(n,c,m) = \Pr[\ |B(s)| > m\ \ :\ s \leftarrow_\$ \mathsf{D}(n,c)]\ . \tag{8}$$

We want to upper bound this, which in our application will allow m to be the number of users for which mu security without corruptions is assumed.

Biased-coin sampler. We extract from the Coron technique [Cor00] a sampler $\mathsf{D}\text{-}\mathsf{BC}^x$ that we call the biased-coin sampler and show on the left in Figure 2. Here $x \in [0,1]$ is a probability and $b \leftarrow_x \{0,1\}$ returns a x-biased coin $b \in \{0,1\}$, meaning $\Pr[b=1] = x$ and $\Pr[b=0] = 1-x$. We let $\mathsf{D}\text{-}\mathsf{BC} = \mathsf{D}\text{-}\mathsf{BC}^x$ for $x = 1-1/(c+1)$ and refer to this as the optimal biased-coin sampler. Theorem 3.2 below justifies this name by showing that $\mathsf{D}\text{-}\mathsf{BC}$ maximizes the success probability across samplers in the biased-sampler class. It also gives a good lower bound on this success probability and bounds the sampler error probability.

Theorem 3.2 *Let $n, m \geq 1$ and $c \geq 0$ be integers such that $n/(c+1) < m$ and $c < n$. Let $\mathsf{D}\text{-}\mathsf{BC}^x$ be the biased-coin sampler associated to $x \in [0,1]$ as per Figure 2. Then $\mathsf{D}\text{-}\mathsf{BC}^x$ is Hamming-weight determined and $\mathbf{P}^*_{\mathsf{D}\text{-}\mathsf{BC}^x}(n,c) = x^c(1-x)$. Furthermore $\mathbf{P}^*_{\mathsf{D}\text{-}\mathsf{BC}}(n,c) \geq \mathbf{P}^*_{\mathsf{D}\text{-}\mathsf{BC}^x}(n,c)$ for all $x \in [0,1]$*

and

$$\mathbf{P}^*_{\mathsf{D\text{-}BC}}(n,c) = \left(1 - \frac{1}{c+1}\right)^c \frac{1}{c+1} \geq \frac{1}{e} \cdot \frac{1}{c+1} \ . \tag{9}$$

Letting $a = m - n/(c+1)$ *we also have* $\mathbf{R}_{\mathsf{D\text{-}BC}}(n,c,m) \leq e^{-a^2/2n}$ *if* $m < n$ *and* $\mathbf{R}_{\mathsf{D\text{-}BC}}(n,c,m) = 0$ *if* $m \geq n$.

Proof: Let $\mathrm{W}(n,c,\ell) = x^\ell(1-x)^{n-\ell}$. If $s \in \{0,1\}^n$ has Hamming weight ℓ then its probability of being output by $\mathsf{D\text{-}BC}^x$ is $\mathrm{W}(n,c,\ell)$, showing that the sampler $\mathsf{D\text{-}BC}^x$ is Hamming-weight determined with Hamming-weight probability function W. Define the function $f : [0,1] \to [0,1]$ by $f(x) = x^c(1-x)$. Now from Eq. (7) we have

$$\begin{aligned}
\mathbf{P}^*_{\mathsf{D\text{-}BC}^x}(n,c) &= \textstyle\sum_{j=0}^{n-c-1} x^{c+j}(1-x)^{n-c-j} \cdot \binom{n-c-1}{j} \\
&= x^c(1-x) \cdot \textstyle\sum_{j=0}^{n-c-1} x^j(1-x)^{n-c-1-j} \cdot \binom{n-c-1}{j} \qquad (10) \\
&= x^c(1-x) = f(x) \ . \qquad (11)
\end{aligned}$$

The sum in Eq. (10) has value 1 by the Binomial Theorem, justifying Eq. (11). Now we seek to maximize f. Let $p = 1-1/(c+1)$. The derivative of f is $f'(x) = cx^{c-1}(1-x) - x^c = x^{c-1}(c - x(c+1))$. This derivative is 0 for $x = c/(c+1) = 1 - 1/(c+1) = p$ and $x = 0$, and it is easy to see that f attains its maximum at p, so that

$$\mathbf{P}^*_{\mathsf{D\text{-}BC}}(n,c) = \mathbf{P}^*_{\mathsf{D\text{-}BC}^p}(n,c) = f(p) \geq f(x) = \mathbf{P}^*_{\mathsf{D\text{-}BC}^x}(n,c)$$

for all $x \in [0,1]$, as claimed in the Proposition statement. Now we evaluate $f(p)$ to get

$$\mathbf{P}^*_{\mathsf{D\text{-}BC}}(n,c) = f(p) = p^c(1-p) = \left(1 - \frac{1}{c+1}\right)^c \frac{1}{c+1} \ .$$

We now claim that

$$\left(1 - \frac{1}{c+1}\right)^c \geq \frac{1}{e} \ , \tag{12}$$

which will justify Eq. (9). Towards this we first note that for all ϵ in the range $0 \leq \epsilon < 1$ we have

$$\begin{aligned}
(1-\epsilon)\ln(1-\epsilon) &= -(1-\epsilon)(\epsilon + \epsilon^2/2 + \epsilon^3/3 + \cdots) \qquad (13) \\
&= -\epsilon + (\epsilon^2/2 + \epsilon^3/6 + \epsilon^4/12 + \cdots) \\
&\geq -\epsilon \ ,
\end{aligned}$$

where Eq. (13) used a Taylor series for $\ln(1-\epsilon)$. Now set $\epsilon = 1/(c+1)$. Then by the above we have $\ln(1 - 1/(c+1)) = \ln(1-\epsilon) \geq -\epsilon/(1-\epsilon) = -1/c$ and thus $1 - 1/(c+1) \geq e^{-1/c}$, from which we get Eq. (12).

Moving to the bounds on the error probability, the second claim is clear since the number of blue points cannot be more than n. To show the first claim, we first recall the following:

> *Chernoff bound.* Let $\mathsf{X}_1, \ldots, \mathsf{X}_n$ be independent random variables over $\{0,1\}$, each with expectation q, and let $\mathsf{X} = \mathsf{X}_1 + \cdots + \mathsf{X}_n$. Then $\Pr[\mathsf{X} - nq > a] \leq e^{-a^2/2n}$ for any $a > 0$.

To apply this let $\mathsf{X}_i(s) = 1 - s[i]$ for $i \in [1..n]$, with the probability over $s \leftarrow\!\!\$\ \mathsf{D\text{-}BC}$. Then $\mathsf{X}(s) = |B(s)|$ is the number of blue points in s. Each X_i has expectation $q = 1 - p$ so $nq = n(1-p) = n/(c+1)$ and we have assumed $m > n/(c+1)$ so $a = m - nq > 0$ and we have

$$\mathbf{R}_{\mathsf{D\text{-}BC}}(n,c,m) = \Pr[\ |B(s)| > m \ : \ s \leftarrow\!\!\$\ \mathsf{D\text{-}BC}(n,c)] = \Pr[\mathsf{X} > m]$$

n	c	D-BC		D-FX		$e(c+1)$
		$1/\mathbf{P}^*_{\mathsf{D\text{-}BC}}(n,c)$	m	$1/\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c)$	m	
100 M	10 K	27,184	150,665	27,183	9,999	27,185
100 M	100 K	271,829	143,293	271,694	999	271,830
100 M	1000 K	2,718,283	144,002	2,704,680	99	2,718,284
250 M	25 K	67,958	233,439	67,955	9,999	67,959
250 M	250 K	679,571	227,001	679,232	999	679,573
250 M	2500 K	6,795,705	228,633	6,761,699	99	6,795,707
500 M	50 K	135,915	327,085	135,909	9,999	135,916
500 M	500 K	1,359,142	321,695	1,358,463	999	1,359,143
500 M	5000 K	13,591,410	324,365	13,523,397	99	13,591,411

Figure 3: **Evaluation of samplers**, where n is stated in millions (M) and c in thousands (K). For D-BC we compute m such that $\mathbf{R}_{\mathsf{D\text{-}BC}}(n,c,m)/\mathbf{P}^*_{\mathsf{D\text{-}BC}}(n,c) \le 2^{-128}$ since this will be an additive term in our main theorem (cf. Theorem 4.1). For D-FX we set $m = \lfloor (n-1)/(c+1) \rfloor$. The last column is our estimate of $e(c+1)$ for the reciprocal of the success probabilities.

$$= \Pr[\mathsf{X} - nq > a] \le e^{-a^2/2n} ,$$

which completes the proof. ∎

FIXED-SET-SIZE SAMPLER. Can one do better? It turns out the success probability of D-BC is not optimal but still very good; its more important drawback is having non-zero error probability. Our fixed-set-size sampler fills both gaps. Its success probability is optimal, meaning maximal in the class of all HWD samplers, and it achieves this with zero error probability.

For intuition, returning to Eq. (7), let ℓ be such that $\binom{n-c-1}{\ell} \ge \binom{n-c-1}{j}$ for all $j \in [0..n-c-1]$. Then clearly $\mathbf{P}^*_{\mathsf{D}}(n,c)$ is maximized by setting $\mathrm{W}(n,c,c+j) = 1$ if $j = \ell$ and 0 otherwise. That is, all the probability is on strings of Hamming weight $t = c+\ell$. This leads to our fixed-set-size sampler $\mathsf{D\text{-}FX}^t$, which is parameterized by an integer $t \in [c..n-1]$ and shown on the right in Figure 2. Here $\mathcal{P}(n,t)$ denotes the set of all size t subsets of $[1..n]$. The sampler picks a random set $T \subseteq [1..n]$ of size t and returns as s its characteristic vector, meaning $s[j] = 1$ if $j \in T$ and $s[j] = 0$ otherwise for all $j \in [1..n]$. We let $\mathsf{D\text{-}FX} = \mathsf{D\text{-}FX}^t$ for $t = \lceil (cn-1)/(c+1) \rceil$ and refer to this as the optimal fixed-set-size sampler. Theorem 3.3 below justifies this name by showing that D-FX maximizes the success probability, first across all samplers in the fixed-set-size class, and second across *all HWD samplers*, and meanwhile has error probability is zero.

Theorem 3.3 *Let $n, m \ge 1$ and $c, t \ge 0$ be integers such that $n - t \le m$ and $c \le t < n$. Let $\mathsf{D\text{-}FX}^t$ be the fixed-set-size sampler associated to t as per Figure 2. Then $\mathsf{D\text{-}FX}^t$ is Hamming-weight determined and $\mathbf{P}^*_{\mathsf{D\text{-}FX}^t}(n,c) = \binom{n-c-1}{t-c} \cdot \binom{n}{t}^{-1}$. Furthermore $\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c) \ge \mathbf{P}^*_{\mathsf{D\text{-}FX}^t}(n,c)$ for all $t \in [c..n-1]$. Also $\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c) \ge \mathbf{P}^*_{\mathsf{D}}(n,c)$ for all HWD samplers D and*

$$\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c) \ge \frac{1}{e} \cdot \frac{1}{c+1} . \tag{14}$$

Finally, $\mathbf{R}_{\mathsf{D\text{-}FX}}(n,c,m) = 0$.

Proof: Let $\mathrm{W}(n,c,\ell) = 1/\binom{n}{\ell}$ if $\ell = t$ and 0 otherwise. If $s \in \{0,1\}^n$ has Hamming weight ℓ then its probability of being output by $\mathsf{D\text{-}FX}^t$ is $\mathrm{W}(n,c,\ell)$, showing that the sampler $\mathsf{D\text{-}FX}^t$ is Hamming-weight determined with Hamming-weight probability function W. Define the function

$f : [c..n-1] \to [0,1]$ by $f(t) = \binom{n-c-1}{t-c} \cdot \binom{n}{t}^{-1}$. Now from Eq. (7) we have

$$\begin{aligned}\mathbf{P}^*_{\mathsf{D\text{-}FX}^t}(n,c) &= \textstyle\sum_{j=0}^{n-c-1} \mathrm{W}(n,c,c+j) \cdot \binom{n-c-1}{j} \\ &= \mathrm{W}(n,c,t) \cdot \binom{n-c-1}{t-c} = \frac{\binom{n-c-1}{t-c}}{\binom{n}{t}} = f(t)\ .\end{aligned}$$

Now we seek to maximize f. Calculus does not seem much help in the face of this complex function. We instead look at ratios of consecutive terms, namely we seek t such that $f(t)/f(t+1) = 1$. We start by noting that

$$\frac{\binom{a}{b}}{\binom{a}{b+1}} = \frac{b+1}{a-b} \tag{15}$$

for any integers $0 \le b < a$. Now for $t \in [c..n-2]$, and using Eq. (15), we have

$$\frac{f(t)}{f(t+1)} = \frac{\binom{n-c-1}{t-c}}{\binom{n-c-1}{t+1-c}} \cdot \frac{\binom{n}{t+1}}{\binom{n}{t}} = \frac{t-c+1}{n-t-1} \cdot \frac{n-t}{t+1}\ . \tag{16}$$

This ratio is 1 when $(t-c+1)(n-t) = (n-t-1)(t+1)$ which solves to $t = (cn-1)/(c+1)$. Let us define $\bar{t}$ to be this value. The latter may however not be an integer. We observe that f increases to its maximum and then decreases, so the maximum for integer arguments occurs at $t^* = \lceil \bar{t} \rceil$, which was the choice of t used to define $\mathsf{D\text{-}FX} = \mathsf{D\text{-}FX}^{t^*}$. This shows the claim in the Theorem that $\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c) \ge \mathbf{P}^*_{\mathsf{D\text{-}FX}^t}(n,c)$ for all $t \in [c..n-1]$. Now the broader optimality claim, namely that $\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c) \ge \mathbf{P}^*_{\mathsf{D}}(n,c)$ for all HWD samplers D, follows from Eq. (7). Why? The expression in the equation is maximized by picking $t \in [c..n-1]$ such that $\binom{n-c-1}{t-c} \ge \binom{n-c-1}{j}$ for all $j \in [0..n-c-1]$ and setting $\mathrm{W}(n,c,c+j) = 1$ if $c+j = t$ and 0 otherwise. That is, $\mathbf{P}^*_{\mathsf{D}}(n,c) \le \mathbf{P}^*_{\mathsf{D\text{-}FX}^t}(n,c) \le \mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c)$. Eq. (14) now follows from Theorem 3.2. The error probability is clearly zero. This completes the proof. ▌

Numerical estimates. Figure 3 gives numbers for a few values of n (in millions) and c (in thousands). We see that $1/\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c) \le 1/\mathbf{P}^*_{\mathsf{D\text{-}BC}}(n,c)$, meaning D-FX is always better than D-BC, but the difference is small. (Intuitively this is because in the biased coin sampler, the expected Hamming weight of the sampled s is $w = cn/(c+1)$, which is very close to $t^* = \lceil w - 1/(c+1) \rceil$.) We see that the approximation $e(c+1)$ is very good, which is why we will use it in our applications. We see that $e(c+1)$, the factor our bounds will give up in applications, is much less than n, the factor that prior bounds gave up. In the least conservative estimation, we set $c = 1\%$ as this already gives improvements for concrete parameters. Depending on the application, an asymptotic improvement can be achieved if c can be bounded by $\sqrt{n}$. For D-BC, we have chosen m to put the error probability at 2^{-128}, and see that it is already significantly less than n, but the $m = \lfloor (n-1)/(c+1) \rfloor$ for D-FX is appreciably better (lower).

Conclusion. Moving forward, we will use the fixed-set-size sampler D-FX and Theorem 3.3. This may raise the question of why we have considered the biased-coin sampler at all. One reason is that the analysis and results of Theorem 3.2 are used and needed as comparison points to determine and eventually conclude that Theorem 3.3 does better. The other reason is historical, namely that the biased-coin sampler is the natural extension of Coron's technique and thus worthy of formulating and analyzing.

4 General framework and cp-muc security theorem

Our technique works across many games (security notions) and schemes. We want to avoid repeating similar proofs each time and also want to understand the scope and limits of the technique: under what conditions does it work, and when does it not work? This Section develops answers to these questions with a general framework.

Our goal is to prove statements of the form "For all security games satisfying a certain condition, mu security can be promoted to cp-muc security." For this to be mathematically sound requires a formal definition of a "security game." The only approach we know, code-based games [BR06], would require formalizing a programming language. We suggest here a simpler approach suited to our ends. We define an object called a formal security specification (FSS) that, formally, is just an algorithm that functionally specifies the initializations, oracle input-output behaviors (including what happens under corruptions) and final decision of the game underlying the target security notion. To an FSS Π we then associate three (standard, code-based) games capturing su, mu and muc security, respectively. We define locality of an FSS as the condition needed for the result, which is stated and proved in our General Theorem (Theorem 4.1).

SCHEMES. An FSS aims to define security of a scheme Sch, so we start with a simple and general abstraction of the latter. Namely, a *scheme* Sch is simply a tuple. Its entries may include algorithms as well as (descriptions of) associated sets or numbers. An example is a signature scheme, which specifies a key-generation algorithm $\mathsf{Sch.Kg}$, a signing algorithm $\mathsf{Sch.Sign}$ and a verification algorithm $\mathsf{Sch.Vf}$. As this indicates, we extract individual components of the scheme tuple with dot notation. Another example is an encryption scheme, specifying key-generation algorithm $\mathsf{Sch.Kg}$, encryption algorithm $\mathsf{Sch.Enc}$ and decryption algorithm $\mathsf{Sch.Dec}$. It may also specify the length $\mathsf{Sch.rl}$ of the randomness (coins) used by the encryption algorithm, and a message space $\mathsf{Sch.MS}$.

In the ROM, it is often the case that the range set of the RO needed depends on the scheme. (For example, for a KEM, it could be the set of strings of length the desired session key.) To accommodate this, we allow schemes to name the set $\mathsf{Sch.ROS}$ from which they ask their random oracle to be drawn.

FORMAL SECURITY SPECIFICATIONS AND THE SINGLE-USER GAME. A *formal security specification* (FSS) Π is an algorithm whose input is a string $\mathtt{name}$, drawn from a finite set $\mathsf{Names} \subseteq \{0,1\}^*$ associated to Π, that serves to name a sub-algorithm $\Pi(\mathtt{name}, \cdots)$. The names $\mathtt{gs}, \mathtt{init}, \mathtt{fin} \in \mathsf{Names}$, and in our context also $\mathtt{corr} \in \mathsf{Names}$, are reserved; they stand, respectively, for "Global Setup," "Initialize," "Finalize" and "Corrupt." Particular FSSs can define more names. Through (code-based) game $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$ shown in Figure 4, Π specifies a notion of security for a scheme Sch. In INIT, line 1 picks, from the random oracle space of the scheme, a function h that will serve as the random oracle. The global setup sub-algorithm $\Pi(\mathtt{gs})$ of the FSS is then run to produce a pair $(pp, os) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{gs})$ whose components are called the public parameters and oracle secrets, respectively. (An example os is a global challenge bit for an ind-style game.) As the notation indicates, sub-algorithms of Π have access to the scheme Sch and may thus call its algorithms, and they also have oracle access to h. At line 2, the initialize sub-algorithm $\Pi(\mathtt{init}, \cdot)$ is run. It takes as input pp, os and produces an output iout together with an initial state St. The adversary is given (pp, iout). The $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$ game will maintain the state St and update it as necessary. After that, the FSS maps to the actual game in a straightforward way. For any $\mathtt{name} \in \{\mathtt{gs}, \mathtt{init}, \mathtt{fin}, \mathtt{corr}\}$, sub-algorithm $\Pi(\mathtt{name}, \cdots)$ implements an actual oracle ORACLE$(\mathtt{name}, \cdots)$ in $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$. Given an argument arg, the latter runs $\Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{name}, \text{arg}, \text{St})$, where St is the current state, to get an output oout (that is returned to the adversary) and updated state St (that is maintained by game $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$). Game $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$ also provides a random oracle RO that gives the adversary access to h. FIN

Game $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$

INIT:
1 $\mathsf{h} \leftarrow\!\!\$\ \mathsf{Sch.ROS}$; $(pp, os) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{gs})$
2 $(\text{iout}, \text{St}) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{init}, (pp, os))$; Return (pp, iout)

ORACLE($\mathtt{name}$, arg): // $\mathtt{name} \notin \{\mathtt{gs}, \mathtt{init}, \mathtt{fin}, \mathtt{corr}\}$
3 $(\text{oout}, \text{St}) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{name}, \text{arg}, \text{St})$; Return oout

RO(x): // Random oracle
4 $y \leftarrow \mathsf{h}(x)$; Return y

FIN(farg):
5 $\text{dec} \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{fin}, \text{farg}, \text{St})$; Return dec

$\mathsf{UF}[\mathsf{Sig}, \mathsf{h}](\mathtt{gs})$:
1 Return $(\varepsilon, \varepsilon)$

$\mathsf{UF}[\mathsf{Sig}, \mathsf{h}](\mathtt{init}, (\varepsilon, \varepsilon))$:
2 $(vk, sk) \leftarrow\!\!\$\ \mathsf{Sig.Kg}[\mathsf{h}]$
3 St.vk $\leftarrow vk$; St.sk $\leftarrow sk$; St.S $\leftarrow \emptyset$
4 Return (vk, St)

$\mathsf{UF}[\mathsf{Sig}, \mathsf{h}](\mathtt{sign}, M, \text{St})$:
5 $\sigma \leftarrow\!\!\$\ \mathsf{Sig.Sign}[\mathsf{h}](\text{St.sk}, M)$
6 St.S $\leftarrow$ St.S $\cup \{M\}$; Return (σ, St)

$\mathsf{UF}[\mathsf{Sig}, \mathsf{h}](\mathtt{corr}, \varepsilon, \text{St})$:
7 Return $(\text{St.sk}, \text{St})$

$\mathsf{UF}[\mathsf{Sig}, \mathsf{h}](\mathtt{fin}, (M, \sigma), \text{St})$:
8 If $M \in \text{St.S}$ then return false
9 Return $\mathsf{Sig.Vf}[\mathsf{h}](\text{St.vk}, M, \sigma)$

Game $\mathbf{G}^{\mathsf{UF}}_{\mathsf{Sig}}$

INIT:
1 $\mathsf{h} \leftarrow\!\!\$\ \mathsf{Sig.ROS}$; S $\leftarrow \emptyset$
2 $(vk, sk) \leftarrow\!\!\$\ \mathsf{Sig.Kg}[\mathsf{h}]$
3 Return vk

ORACLE($\mathtt{sign}, M$):
4 $\sigma \leftarrow\!\!\$\ \mathsf{Sig.Sign}[\mathsf{h}](sk, M)$
5 S $\leftarrow$ S $\cup \{M\}$
6 Return σ

RO(x): // Random oracle
7 $y \leftarrow \mathsf{h}(x)$; Return y

FIN(M, σ):
8 If $M \in$ S then return false
9 Return $\mathsf{Sig.Vf}[\mathsf{h}](vk, M, \sigma)$

Figure 4: **Top:** Game associated to formal security specification Π and scheme Sch in the single-user setting. **Bottom Left:** Formal security specification UF. **Bottom Right:** The $\mathbf{G}^{\mathsf{UF}}_{\mathsf{Sig}}$ game, rewritten in standard form.

---

uses sub-algorithm $\Pi(\mathtt{fin}, \cdots)$ to return a game decision dec. Beyond the argument farg provided by the adversary, $\Pi(\mathtt{fin}, \cdots)$ gets the current state as input.

As an example, we show on the bottom left of Figure 4 the FSS UF corresponding to uf-security of a signature scheme Sig. To its right we show $\mathbf{G}^{\mathsf{UF}}_{\mathsf{Sig}}$ re-written in the usual way to see that it is the standard game. Of course, if one is interested in just one notion (such as uf-security) one can give $\mathbf{G}^{\mathsf{UF}}_{\mathsf{Sig}}$ directly and there is no need for FSSs, but as we have discussed, FSSs will allow us to give precise yet general results covering many games, and also allow us to classify games into different types.

The $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$ game captures the single-user setting and does not use sub-algorithm $\Pi(\mathtt{corr}, \cdots)$; it will be used below. The difference between pp and os is that the FSS gets both but the adversary gets only the former. This distinction will allow us to distill exactly what our general result needs to work and also let us to apply the latter broadly and in settings where "corrupt" does not have the usual interpretation.

An FSS Π has an associated type, which is either "search" or "decision." The advantage of an adversary $\mathcal{A}$ playing $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$ is defined as $\mathbf{Adv}^{\Pi}_{\mathsf{Sch}}(\mathcal{A}) = \Pr[\mathbf{G}^{\Pi}_{\mathsf{Sch}}(\mathcal{A})]$ if Π is a search FSS and $\mathbf{Adv}^{\Pi}_{\mathsf{Sch}}(\mathcal{A}) = 2\Pr[\mathbf{G}^{\Pi}_{\mathsf{Sch}}(\mathcal{A})] - 1$ if Π is a decision FSS. In the latter case we will make an extra technical assumption, namely that for all $\mathsf{Sch}, \mathsf{h}, \text{St}$ the decision $\text{dec} \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{fin}, \text{farg}, \text{St})$ is

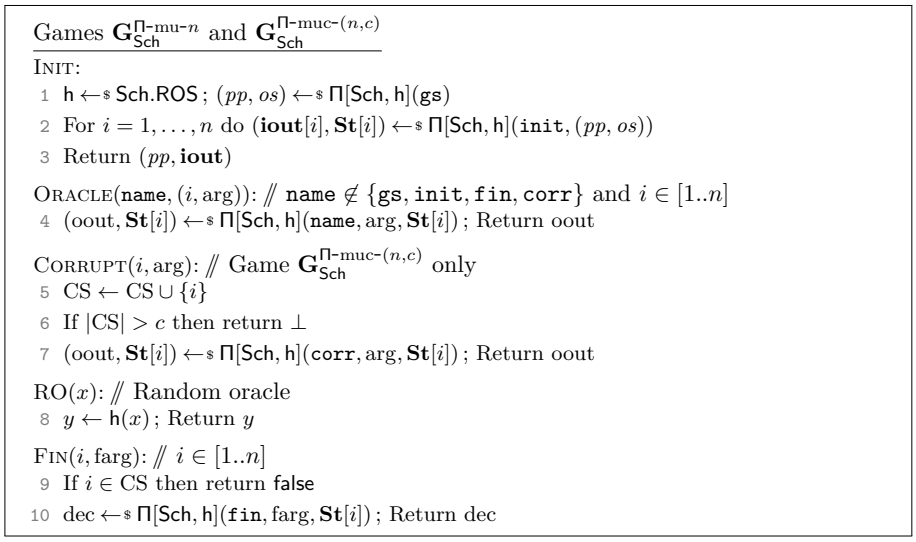

Figure 5: Game describing the execution of a formal security specification Π with scheme Sch in the multi-user setting, both without corruptions (mu) and with corruptions (muc). The difference is that oracle CORRUPT is included only in the second game.

---

uniformly distributed over $\{\mathsf{true}, \mathsf{false}\}$ when $\mathrm{farg} \leftarrow\!\!\$\ \{0,1\}$. This is typically true because os is a random challenge bit and $\Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{fin}, \mathrm{farg}, \mathrm{St})$ returns $[[\mathrm{farg} = os]]$, but it does not in general follow merely through the definition of the advantage as $2\Pr[\mathbf{G}^{\Pi}_{\mathsf{Sch}}(\mathcal{A})] - 1$. So we make it a separate requirement. This will be used in the proof of Theorem 4.1.

We say that an FSS Π is *local* if the oracle secret os is always ε. (A bit more formally, the probability that $os = \varepsilon$ when $(pp, os) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{gs})$ is 1 for all Sch, h.) Intuitively, different users in (upcoming) games $\mathbf{G}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}$ and $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$ will have no secret common to them all but unknown to the adversary. This captures the condition for our general result of Theorem 4.1.

MULTI-USER SECURITY FOR AN FSS. To FSS Π, scheme Sch, a number $n \geq 1$ of users and a number $c \geq 0$ of corruptions, we now associate a multi-user (without corruptions) game $\mathbf{G}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}$ and a multi-user with corruptions game $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$ as shown in Figure 5. We see here the reason to separate initialization into the two sub-algorithms $\Pi(\mathtt{gs})$ and $\Pi(\mathtt{init}, \cdots)$: The first is run once to produce parameters common to all users, while the second is run once per user and produces a state $\mathbf{St}[i]$ for each user $i \in [1..n]$. Oracle ORACLE($\mathtt{name}, \cdots$) now takes a user identity i in addition to arg and answers via the sub-algorithm $\Pi(\mathtt{name}, \cdots)$ using the state of user i. FIN also takes a user identity i and returns its decision using the state of that user. Oracle CORRUPT is present only in game $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$ and answers as per the associated sub-algorithm, again for a given user. Line 6 restricts the number of corruptions to c and line 9 excludes wins on corrupted users. Note that the models of prior works, which allow any number of corruptions, can be captured as the special case of our model in which we set $c = n - 1$. The advantage $\mathbf{Adv}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}(\mathcal{A})$ of an adversary $\mathcal{A}$ playing $\mathbf{G}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}$ is defined as $\Pr[\mathbf{G}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}(\mathcal{A})]$ if Π is a search FSS and $2\Pr[\mathbf{G}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}(\mathcal{A})] - 1$ if Π is a decision FSS, and correspondingly $\mathbf{Adv}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}(\mathcal{A})$ is defined as $\Pr[\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}(\mathcal{A})]$ or $2\Pr[\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}(\mathcal{A})] - 1$.

GENERAL CP-MUC THEOREMS. We give our main, general results that promote mu to corruption-parameterized muc security for *all* local game specifications. Later we will see many applications. Below we crucially leverage HWD samplers.

Theorem 4.1 *Let $n, m \geq 1$ and $c \geq 0$ be integers such that $m \leq n$ and $c < n$. Let D be a Hamming-weight determined sampler. Let $\alpha = \mathbf{P}^*_{\mathsf{D}}(n, c)$ and $\beta = \mathbf{R}_{\mathsf{D}}(n, c, m)$. Let Sch be a scheme, and Π a formal security specification for it. Assume Π is local. Let $\gamma = 1$ if Π is a search-type* FSS *and $\gamma = 2$ if Π is a decision-type* FSS*. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}$ such that*

$$\mathbf{Adv}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}(\mathcal{A}) \leq (1/\alpha) \cdot \mathbf{Adv}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}(\mathcal{B}) + \gamma \cdot \beta/\alpha \ . \tag{17}$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler D.

Before discussing the proof we state a corollary obtained by plugging the fixed-set-size sampler into the above; this is what we will most often use in applications.

Theorem 4.2 *Let n, c be integers such that $0 \leq c < n$, and let $m = \lfloor (n-1)/(c+1) \rfloor$. Let Sch be a scheme, and Π a formal security specification for it. Assume Π is local. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}$ such that*

$$\mathbf{Adv}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}(\mathcal{B}) \ . \tag{18}$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler D-FX.

Proof of Theorem 4.2: Let $\mathsf{D} = \mathsf{D}$-FX be the optimal fixed-set-size sampler and apply Theorem 4.1. We have $\alpha = \mathbf{P}^*_{\mathsf{D}\text{-}\mathsf{FX}}(n, c)$ and $\beta = \mathbf{R}_{\mathsf{D}\text{-}\mathsf{FX}}(n, c, m)$ in Eq. (17). Now Theorem 3.3 says that $1/\alpha \leq e(c+1)$, while $\beta = 0$, yielding Eq. (18). ▌

It remains to prove Theorem 4.1.

Proof of Theorem 4.1: We start with an overview. Adversary $\mathcal{B}$ (shown in detail in Figure 6) picks s by running the sampler D, thereby coloring each user $i \in [1..n]$ as either red ($s[i] = 1$) or blue ($s[i] = 0$). It now runs $\mathcal{A}$. In answering the game-$\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$ queries that $\mathcal{A}$ makes, $\mathcal{B}$ will simulate a red user i directly. This means it will pick and maintain this user's state $\mathbf{St}[i]$, allowing it to answer all queries to this user, including, most importantly, a $\textsc{Corrupt}(i)$ query. Meanwhile, it will aim to identify blue users with the users in its own underlying $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}$ game, using its oracles from that game to reply to queries of $\mathcal{A}$ for these users. It hopes that $\mathcal{A}$ runs to a win with a $\textsc{Fin}$ query to a blue user, in which case $\mathcal{B}$ will win its own game.

There are a number of bad events, ways in which this strategy can fail. The first is that $n - \mathbf{w}_{\mathrm{H}}(s) > m$, meaning the number of blue users is too large for them to be mapped to the users in game $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}$. Letting G_0 (Figure 7) be a game capturing $\mathcal{B}$'s advantage, a first game hop will bound the probability of this bad event via the First Fundamental Lemma of Game Playing (Lemma 2.1), and lead to the β/α term in the bound while putting us now in a game G_1 where the limitation on the number of blue users has vanished. The second source of failure is that either there is a $\textsc{Corrupt}$ query to a blue user ($\mathcal{B}$ has no way to answer this) or $\mathcal{A}$'s $\textsc{Fin}$ query is to a red user (in which case $\mathcal{B}$ won't win). Game G_2 (Figure 8) flags this through settings of flag bad. The difficulty is that the probability (that bad is set) is large, namely close to 1, and not small. So, while handling it again via the First Fundamental Lemma of Game Playing is possible, it would yield a large additive term in the bound, making the latter vacuous. Instead, letting GD be the probability that bad is not set and W the event that $\mathcal{B}$ wins, we seek to lower bound $\Pr[\mathrm{W} \wedge \mathrm{GD}]$ in G_2. A crucial use of the *Second* Fundamental Lemma of Game Playing (Lemma 2.2) equates this with the same probability in a game G_3 where events GD and W are (unlike in G_2) independent.

Adversary $\mathcal{B}$:

1 $s \leftarrow\!\$\ \mathsf{D}(n,c)$; $(pp, \mathbf{iout}') \leftarrow\!\$\ \mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}.\textsc{Init}$; $\text{farg}' \leftarrow\!\$\ \{0,1\}$
2 If $n - \mathbf{w}_{\mathrm{H}}(s) > m$ then $\text{dec} \leftarrow\!\$\ \mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}.\textsc{Fin}(1, \text{farg}')$

▷ Run $\mathcal{A}$, responding to its oracle queries as follows:

$\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}.\textsc{Init}$:
3 For $i = 1, \ldots, n$ do
4 If $s[i] = 1$ then
5 $\mathbf{St}[i].\text{pp} \leftarrow pp$; $(\mathbf{iout}[i], \mathbf{St}[i]) \leftarrow\!\$\ \Pi[\mathsf{Sch}, \mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}.\text{RO}](\mathtt{init}, \varepsilon, \mathbf{St}[i])$
6 Else $j \leftarrow j + 1$; $\mathbf{iout}[i] \leftarrow \mathbf{iout}'[j]$; $\pi(i) \leftarrow j$
7 Return $(pp, \mathbf{iout})$

$\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}.\textsc{Oracle}(\mathtt{name}, (i, \text{arg}))$:
8 If $s[i] = 1$ then $(\text{oout}, \mathbf{St}[i]) \leftarrow\!\$\ \Pi[\mathsf{Sch}, \mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}.\text{RO}](\mathtt{name}, \text{arg}, \mathbf{St}[i])$
9 Else $\text{oout} \leftarrow\!\$\ \mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}.\textsc{Oracle}(\mathtt{name}, (\pi(i), \text{arg}))$
10 Return oout

$\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}.\textsc{Corrupt}(i, \text{arg})$:
11 If $s[i] = 1$ then $(\text{oout}, \mathbf{St}[i]) \leftarrow\!\$\ \Pi[\mathsf{Sch}, \mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}.\text{RO}](\mathtt{corr}, \text{arg}, \mathbf{St}[i])$
12 Else $\text{oout} \leftarrow \perp$; $B \leftarrow B \cup \{i\}$
13 Return oout

$\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}.\text{RO}(x)$:
14 Return $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}.\text{RO}(x)$

$\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}.\textsc{Fin}(i, \text{farg})$:
15 If ($s[i] = 0$ and $B = \emptyset$) then $\text{dec} \leftarrow\!\$\ \mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}.\textsc{Fin}(\pi(i), \text{farg})$
16 Else $\text{dec} \leftarrow\!\$\ \mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}.\textsc{Fin}(\pi(i), \text{farg}')$

Figure 6: Adversary $\mathcal{B}$ for Theorem 4.1.

This leaves us with the product $\Pr[\text{W}] \cdot \Pr[\text{GD}]$ in G_3. To conclude, we will lower bound $\Pr[\text{W}]$ in terms of the advantage of $\mathcal{A}$. Then we will use Theorem 3.1 to lower bound $\Pr[\text{GD}]$ in terms of the sampler success probability $\mathbf{P}^{*}_{\mathsf{D}}(n,c)$ of Eq. (7).

A wrinkle is that the analysis sketched above is for the case that Π is a search-type FSS. The case of it being a decision-type FSS is more delicate. We will need to ensure that when bad events happen, $\mathcal{B}$ has advantage zero, which is done leveraging the assumption we have made that decision formal security specifications return a random decision in this case. While the adversary $\mathcal{B}$ and the games will be written to cover both cases (search and decision), the analyses are different enough that, below, we give them separately. (And indeed the bound in Eq. (17) is slightly different.)

Why do we need to assume that Π is local? The secret os in the $\mathbf{G}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}$ game is not available to $\mathcal{B}$, yet it has to simulate game $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$ for $\mathcal{A}$ in such a way that it is underlain by this same os. This happens correctly for blue users because their queries are forwarded, but $\mathcal{B}$ cannot simulate the oracle replies for red users to be consistent with os. If the FSS is local, however, $os = \varepsilon$ so the difficulty goes away.

Proceeding to the full proof, we assume $\mathcal{A}$ makes exactly c queries to its $\textsc{Corrupt}$ oracle and also respects the rules of its game, allowing us to omit the corresponding checks in our games. Adversary $\mathcal{B}$ is now shown in Figure 6. Recall that it is playing game $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}$, whose oracles it calls. At line 1 it picks s by running D. If there are too many blue users, it terminates at line 2 with a call to $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}.\textsc{Fin}$ using a random bit farg' (chosen at line 1) and (as default) user 1. Now $\mathcal{B}$ runs $\mathcal{A}$, responding to the latter's oracle queries as shown. In the simulation of $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}.\textsc{Init}$, if user i is red, then (line 5) its state is initialized honestly as per Π, and otherwise (line 6) i is identified with

Games $\boxed{\mathrm{G}_0}$, G_1

INIT:
1 $\mathsf{h} \leftarrow\!\!\$\ \mathsf{Sch.ROS}$; $(pp, \varepsilon) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{gs})$; $s \leftarrow\!\!\$\ \mathsf{D}(n, c)$; $\mathrm{farg}' \leftarrow\!\!\$\ \{0,1\}$
2 For $i = 1, \ldots, n$ do
3 $\quad (\mathbf{iout}[i], \mathbf{St}[i]) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{init}, (pp, \varepsilon))$
4 If $n - \mathbf{w}_{\mathrm{H}}(s) > m$ then
5 $\quad \mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\mathrm{dec} \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{fin}, \mathrm{farg}', \mathbf{St}[1])}$
6 Return $(pp, \mathbf{iout})$

ORACLE$(\mathtt{name}, (i, \mathrm{arg}))$: // $\mathtt{name} \notin \{\mathtt{init}, \mathtt{fin}, \mathtt{corr}\}$ and $i \in [1..n]$
7 $(\mathrm{oout}, \mathbf{St}[i]) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{name}, \mathrm{arg}, \mathbf{St}[i])$; Return oout

CORRUPT(i, arg): // $i \in [1..n]$
8 $\mathrm{CS} \leftarrow \mathrm{CS} \cup \{i\}$
9 If $s[i] = 1$ then $(\mathrm{oout}, \mathbf{St}[i]) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{corr}, \mathrm{arg}, \mathbf{St}[i])$
10 Else oout $\leftarrow \perp$; $B \leftarrow B \cup \{i\}$
11 Return oout

RO(x): // Random oracle
12 $y \leftarrow \mathsf{h}(x)$; Return y

FIN(i, farg): // $i \in [1..n]$
13 If dec $\neq \perp$ then return dec
14 If $i \in \mathrm{CS}$ then return false
15 If $(s[i] = 0$ and $B = \emptyset)$ then $\mathrm{dec} \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{fin}, \mathrm{farg}, \mathbf{St}[i])$
16 Else $\mathrm{dec} \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{fin}, \mathrm{farg}', \mathbf{St}[i])$
17 Return dec

Figure 7: Games $\mathrm{G}_0, \mathrm{G}_1$ for the proof of Theorem 4.1, where the former includes the boxed code and the latter does not.

---

user $\pi(i)$ of game $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}$. For queries $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$.ORACLE$(\mathtt{name}, (i, \mathrm{arg}))$, the reply is computed directly if i is red and forwarded to user $\pi(i)$ in game $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}$ otherwise. If i is red, a correct reply is possible, and given, to a $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$.CORRUPT$(i, \mathrm{arg})$ query, and otherwise the reply is $\perp$. Random oracle queries of $\mathcal{A}$ are simply forwarded to $\mathcal{B}$'s random oracle. (The latter is also used to reply to random oracle queries made by Π in any direct executions by $\mathcal{B}$ of the latter, as at lines 5, 8, 11.) When $\mathcal{A}$ calls $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$.FIN$(i, \mathrm{farg})$, adversary $\mathcal{B}$ calls its own $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}$.FIN for user $\pi(i)$ with the same argument farg, and otherwise fails through calling with a random argument farg$'$.

For the analysis, consider games $\mathrm{G}_0, \mathrm{G}_1$ of Figure 7, where the former includes the boxed code and the latter does not. These are to be executed with adversary $\mathcal{A}$. (Not $\mathcal{B}$.) In the following we make the case distinction between search-type and decision-type FSS.

CASE 1: SEARCH-TYPE FSS. Assume Π is a search-type FSS. The clump of equations below is justified after their statement:

$$\mathbf{Adv}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}(\mathcal{B}) = \Pr[\mathrm{G}_0(\mathcal{A})] \tag{19}$$

$$= \Pr[\mathrm{G}_1(\mathcal{A})] + \Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})] \tag{20}$$

$$\geq \Pr[\mathrm{G}_1(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \ . \tag{21}$$

We now justify the above. Game G_0 starts from game $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$ and adapts it to give $\mathcal{A}$ the environment that it sees when it is executed by $\mathcal{B}$. Thus, in addition to the choices made by $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$, it picks s (line 1) by running the sampler and fails to answer CORRUPT queries (line 10) for blue users. One subtlety is that $\mathcal{B}$ terminates at its line 2 (Figure 6) without running

Games $\boxed{\mathrm{G}_2}$, G_3

INIT:
1 $\mathsf{h} \leftarrow\!\!\$\ \mathsf{Sch.ROS}$; $(pp, \varepsilon) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{gs})$; $s \leftarrow\!\!\$\ \mathsf{D}(n, c)$; $\mathrm{farg}' \leftarrow\!\!\$\ \{0,1\}$
2 For $i = 1, \ldots, n$ do
3 $\quad (\mathbf{iout}[i], \mathbf{St}[i]) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{init}, (pp, \varepsilon))$
4 Return $(pp, \mathbf{iout})$

ORACLE($\mathtt{name}, (i, \mathrm{arg})$): // $\mathtt{name} \notin \{\mathtt{init}, \mathtt{fin}, \mathtt{corr}\}$ and $i \in [1..n]$
5 $(\mathrm{oout}, \mathbf{St}[i]) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{name}, \mathrm{arg}, \mathbf{St}[i])$; Return oout

CORRUPT(i, arg):
6 $\mathrm{CS} \leftarrow \mathrm{CS} \cup \{i\}$; $(\mathrm{oout}, \mathbf{St}[i]) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{corr}, \mathrm{arg}, \mathbf{St}[i])$
7 If $s[i] = 0$ then $B \leftarrow B \cup \{i\}$
8 If $s[i] = 0$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\mathrm{oout} \leftarrow \bot}$
9 Return oout

RO(x): // Random oracle
10 $y \leftarrow \mathsf{h}(x)$; Return y

FIN(i, farg): // $i \in [1..n]$
11 If $i \in \mathrm{CS}$ then return false
12 $\mathrm{dec} \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{fin}, \mathrm{farg}, \mathbf{St}[i])$
13 If ($s[i] = 1$ or $B \neq \emptyset$) then
14 $\quad \mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\mathrm{dec} \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{fin}, \mathrm{farg}', \mathbf{St}[i])}$
15 Return dec

Figure 8: Games $\mathrm{G}_2, \mathrm{G}_3$ for the proof of Theorem 4.1, where the former includes the boxed code and the latter does not.

$\mathcal{A}$ if there are too many blue users, while G_0 computes the decision in the boxed code at line 5 but its execution with $\mathcal{B}$ would continue; however, the outcome is the same due to line 13 (Figure 7) and the assumption that adversaries always call their FIN oracle. Recalling that Π is a search-type FSS, this justifies Eq. (19). (Note that we use here the assumption that Π is local, meaning $os = \varepsilon$, without which this equation may not be true.) Eq. (20) is trivial. We have written games $\mathrm{G}_0, \mathrm{G}_1$ to be identical-until-bad, so Eq. (21) follows from the First Fundamental Lemma of Game Playing (Lemma 2.1). Now $n - \mathbf{w}_{\mathrm{H}}(s)$ at line 4 is the number of blue points in s so by Eq. (8) we have

$$\Pr[\mathrm{G}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \mathbf{R}_{\mathsf{D}}(n, c, m) = \beta. \tag{22}$$

The benefit of moving from G_0 to G_1 is that the restriction on the number of blue users has vanished. We turn to lower bounding $\Pr[\mathrm{G}_1(\mathcal{A})]$. For this consider games $\mathrm{G}_2, \mathrm{G}_3$ of Figure 8 (the former includes the boxed code and the latter does not), again to be executed with $\mathcal{A}$. In either game, let GD be the event that bad is not set to true in the execution of $\mathcal{A}$ with the game. We claim that

$$\begin{aligned}
\Pr[\mathrm{G}_1(\mathcal{A})] &= \Pr[\mathrm{G}_2(\mathcal{A})] && (23)\\
&\geq \Pr[\mathrm{G}_2(\mathcal{A}) \wedge \mathrm{GD}] && (24)\\
&= \Pr[\mathrm{G}_3(\mathcal{A}) \wedge \mathrm{GD}] && (25)\\
&= \Pr[\mathrm{G}_3(\mathcal{A})] \cdot \Pr[\mathrm{GD}]\ . && (26)
\end{aligned}$$

Let us justify this. With the intent of moving towards game $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$, game G_2 initializes all users honestly. However, the boxed code at line 8 ensures that it fails on CORRUPT queries to blue users like G_1, and the boxed code on line 14 ensures the decision returned is that same as in G_1, justifying Eq. (23). Eq. (24) is trivial. Games $\mathrm{G}_2, \mathrm{G}_3$ have been defined to be identical-until-bad, so

that we may now (crucially) apply the Second Fundamental Lemma of Game Playing (Lemma 2.2) to get Eq. (25). The benefit of this move is that in G_3, due to the absence of the boxed code, the setting of bad does not influence what the game returns, so the events "$\mathrm{G}_3(\mathcal{A})$" and GD are independent, justifying Eq. (26).

Next we observe that

$$\Pr[\mathrm{G}_3(\mathcal{A})] = \mathbf{Adv}^{\mathsf{\Pi\text{-}muc\text{-}}(n,c)}_{\mathsf{Sch}}(\mathcal{A}) \tag{27}$$

$$\Pr[\mathrm{GD}] = \mathbf{P}^{*}_{\mathsf{D}}(n,c) = \alpha\ . \tag{28}$$

With the boxed code gone, G_3 is the same as $\mathbf{G}^{\mathsf{\Pi\text{-}muc\text{-}}(n,c)}_{\mathsf{Sch}}$, justifying Eq. (27). If bad is set at line 8 it is also set at line 13, so for the purpose of computing $\Pr[\mathrm{GD}]$ we can consider only the latter. Now we see that the probability that bad is not set at line 13 in the $\textsc{Fin}(i, \mathrm{farg})$ call is exactly $\mathbf{P}_{\mathsf{D},n,c}(\mathrm{CS}, i)$ as defined via Eq. (6). But we have assumed $\mathcal{A}$ makes exactly c queries to $\textsc{Corrupt}$, so $|\mathrm{CS}| = c$. Now Proposition 3.1 implies that $\mathbf{P}_{\mathsf{D},n,c}(\mathrm{CS}, i) = \mathbf{P}^{*}_{\mathsf{D}}(n,c)$. (In particular, it does not depend on the choices of CS, i, which here are random variables.) This justifies Eq. (28).

Putting together the above we have

$$\mathbf{Adv}^{\mathsf{\Pi\text{-}mu\text{-}}m}_{\mathsf{Sch}}(\mathcal{B}) \geq \alpha \cdot \mathbf{Adv}^{\mathsf{\Pi\text{-}muc\text{-}}(n,c)}_{\mathsf{Sch}}(\mathcal{A}) - \beta\ ,$$

yielding Eq. (17), and completing the proof, in the case that Π is a search-type FSS.

Case 2: Decision-type FSS. We now proceed to the case that Π is a decision-type FSS. The adversary $\mathcal{B}$ and games are already written to handle this. We indicate how to adapt the analysis. We start with the key claim, namely that

$$\Pr[\,\mathrm{G}_2(\mathcal{A}) \mid \neg\mathrm{GD}\,] = \frac{1}{2} \tag{29}$$

where $\neg$GD, the complement of GD, is the event that bad is set to true in the game. The reason for this is that whenever $\neg$GD happens, the output of the game is determined at line 14 with farg' the random bit from line 1, and our definition of decision formal security specifications then says that the game output $\mathrm{dec} \in \{\mathsf{true}, \mathsf{false}\}$ is also random. Note Eq. (29) implies that

$$\begin{aligned} &2\Pr[\mathrm{G}_2(\mathcal{A}) \wedge \neg\mathrm{GD}] - \Pr[\neg\mathrm{GD}] \\ &= 2\Pr[\,\mathrm{G}_2(\mathcal{A}) \mid \neg\mathrm{GD}\,] \cdot \Pr[\neg\mathrm{GD}] - \Pr[\neg\mathrm{GD}] = 0\ , \end{aligned} \tag{30}$$

which we will use below. Now, turning to the full analysis, we start with

$$\begin{aligned} \mathbf{Adv}^{\mathsf{\Pi\text{-}mu\text{-}}m}_{\mathsf{Sch}}(\mathcal{B}) &= 2\Pr[\mathrm{G}_0(\mathcal{A})] - 1 \\ &= 2\Pr[\mathrm{G}_1(\mathcal{A})] + 2\Pr[\mathrm{G}_0(\mathcal{A})] - 2\Pr[\mathrm{G}_1(\mathcal{A})] - 1 \\ &\geq (2\Pr[\mathrm{G}_1(\mathcal{A})] - 1) - 2\Pr[\mathrm{G}_1(\mathcal{A}) \text{ sets } \mathsf{bad}]\ . \end{aligned} \tag{31}$$

The justifications of the above equations are as in the search case, and we have already bounded $\Pr[\mathrm{G}_1(\mathcal{A}) \text{ sets } \mathsf{bad}]$ in Eq. (22). Proceeding,

$$\begin{aligned} &2\Pr[\mathrm{G}_1(\mathcal{A})] - 1 = 2\Pr[\mathrm{G}_2(\mathcal{A})] - 1 \\ &= 2\Pr[\mathrm{G}_2(\mathcal{A}) \wedge \mathrm{GD}] + 2\Pr[\mathrm{G}_2(\mathcal{A}) \wedge \neg\mathrm{GD}] - (\Pr[\mathrm{GD}] + \Pr[\neg\mathrm{GD}]) \\ &= 2\Pr[\mathrm{G}_2(\mathcal{A}) \wedge \mathrm{GD}] - \Pr[\mathrm{GD}] + 0 \\ &= 2\Pr[\mathrm{G}_3(\mathcal{A}) \wedge \mathrm{GD}] - \Pr[\mathrm{GD}] \\ &= 2\Pr[\mathrm{G}_3(\mathcal{A})] \cdot \Pr[\mathrm{GD}] - \Pr[\mathrm{GD}] \end{aligned} \tag{32}$$

$$= (2\Pr[\mathrm{G}_3(\mathcal{A})] - 1) \cdot \Pr[\mathrm{GD}] \ .$$

Eq. (32) is by Eq. (30). Other equations are either standard probability theory or justified as above in the search case. We also use throughout that (by Lemma 2.1) $\Pr[\mathrm{GD}]$ is the same in G_2 and G_3, and hence do not indicate the game in the notation. Now we observe that

$$2\Pr[\mathrm{G}_3(\mathcal{A})] - 1 = \mathbf{Adv}^{\mathsf{\Pi\text{-}muc\text{-}}(n,c)}_{\mathsf{Sch}}(\mathcal{A}) \ . \tag{33}$$

Then using Eq. (28) and putting all this together we have

$$\mathbf{Adv}^{\mathsf{\Pi\text{-}mu\text{-}}m}_{\mathsf{Sch}}(\mathcal{B}) \geq \alpha \cdot \mathbf{Adv}^{\mathsf{\Pi\text{-}muc\text{-}}(n,c)}_{\mathsf{Sch}}(\mathcal{A}) - 2\beta \ ,$$

yielding Eq. (17), and completing the proof, in the case that Π is a decision-type FSS. ▮

5 Applications

We can promote mu to cp-muc for many target goals and schemes simply by noting that the FSS is local and applying Theorem 4.2. We call these direct applications, and below we illustrate with a few examples, but there are many more. However, the FSSs for some important goals, most notably ind-style games with a single challenge bit across all users, are not local. Nonetheless, we can give dedicated proofs for specific schemes. Since these results use Theorem 4.2 in an intermediate step, we call them indirect applications.

5.1 Direct applications

Digital Signatures. We formally state our theorem for signature schemes underlying the informal claim of Eq. (2) in our use of signatures as an example in the Introduction. The FSS for signatures is UF (Figure 4), which is local. The corresponding mu(c) games are obtained via UF as described in Figure 5. Theorem 4.2 now yields:

Theorem 5.1 *Let n, c be integers such that $0 \leq c < n$, and let $m = \lfloor (n-1)/(c+1) \rfloor$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\mathsf{UF\text{-}muc\text{-}}(n,c)}_{\mathsf{Sig}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\mathsf{UF\text{-}mu\text{-}}m}_{\mathsf{Sig}}$ such that*

$$\mathbf{Adv}^{\mathsf{UF\text{-}muc\text{-}}(n,c)}_{\mathsf{Sig}}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}^{\mathsf{UF\text{-}mu\text{-}}m}_{\mathsf{Sig}}(\mathcal{B}) \ .$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler $\mathsf{D\text{-}FX}$.

As explained in Section 1.1: (1) for tightly mu secure schemes, this implies

$$\mathbf{Adv}^{\mathsf{UF\text{-}muc\text{-}}(n,c)}_{\mathsf{Sig}}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}^{\mathsf{UF}}_{\mathsf{Sig}}(\mathcal{B}) \ ,$$

a significant improvement over the factor n from the hybrid argument bound, and (2) this yields tightness improvements for standardized schemes that are used in practical applications such as AKE, which we will discuss in Section 5.2. Further, we note that the analogous result is true for strongly-unforgeability of signatures.

Public-Key Encryption. Mu (and thus muc) security for public-key encryption and KEMs can be defined with a single challenge bit across all users [BBM00], which we call the single-bit (sb) setting, or with a per-user challenge bit, which we call the multi-bit (mb) setting. The settings are asymptotically equivalent, but we are interested in tightness. Heum and Stam [HS21] show that

CCA-MB[KEM, h](gs):

1 Return $(\varepsilon, \varepsilon)$

CCA-MB[KEM, h](init, ε):

2 $(ek, dk) \leftarrow\!\!\$\ \mathsf{KEM.Kg}[\mathsf{h}]$

3 St.ek $\leftarrow ek$; St.dk $\leftarrow dk$

4 $b \leftarrow\!\!\$\ \{0,1\}$; St.b $\leftarrow b$

5 Return (ek, St)

CCA-MB[KEM, h](enc, ε, St):

6 $(C, K_0) \leftarrow\!\!\$\ \mathsf{KEM.Encaps}[\mathsf{h}](\mathrm{St.ek})$

7 $K_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{KEM.kl}}$

8 St.S $\leftarrow$ St.S $\cup \{C\}$

9 Return $(C, K_{\mathrm{St.b}}, \mathrm{St})$

CCA-MB[KEM, h](dec, C, St):

10 If $C \in$ St.S then return $\perp$

11 $K \leftarrow \mathsf{KEM.Decaps}[\mathsf{h}](\mathrm{St.dk}, C)$

12 Return (K, St)

CCA-MB[KEM, h](corr, ε, St):

13 Return $(\mathrm{St.dk}, \mathrm{St})$

CCA-MB[KEM, h](fin, b', St):

14 Return $([\![\mathrm{St.b} = b']\!], \mathrm{St})$

Game $\mathbf{G}^{\text{CCA-MB-muc-}(n,c)}_{\mathsf{KEM}}$

INIT:

1 $\mathsf{h} \leftarrow\!\!\$\ \mathsf{Sch.ROS}$

2 For $i = 1, \ldots, n$ do

3 $\quad (ek_i, dk_i) \leftarrow\!\!\$\ \mathsf{KEM.Kg}[\mathsf{h}]$

4 $\quad b_i \leftarrow\!\!\$\ \{0,1\}$

5 Return $(ek_1, \ldots, ek_n)$

ORACLE(enc, i): // $i \in [1..n]$

6 $(C, K_0) \leftarrow\!\!\$\ \mathsf{KEM.Encaps}[\mathsf{h}](ek_i)$

7 $K_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{KEM.kl}}$

8 $\mathrm{S}_i \leftarrow \mathrm{S}_i \cup \{C\}$; Return (C, K_{b_i})

ORACLE(dec, (i, C)): // $i \in [1..n]$, $C \notin \mathrm{S}_i$

9 $K \leftarrow \mathsf{KEM.Decaps}[\mathsf{h}](dk_i, C)$

10 Return K

CORRUPT(i): // $i \in [1..n]$

11 If $\mathrm{S}_i \neq \emptyset$ then return $\perp$

12 CS $\leftarrow$ CS $\cup \{i\}$; Return dk_i

RO(x):

13 $h \leftarrow \mathsf{h}(x)$; Return h

FIN(i, b'): // $i \in [1..n]$

14 If $i \in$ CS then return false

15 Return $[\![b_i = b']\!]$

Figure 9: **Left:** Formal security specification CCA-MB[KEM, h] capturing security with multiple challenge bits. **Right:** Game $\mathbf{G}^{\text{CCA-MB-muc-}(n,c)}_{\mathsf{KEM}}$ describing the execution of CCA-MB[KEM, h] in the muc setting.

without corruptions, SB security is the stronger definition, but which is stronger in the presence of adaptive corruptions is open.

Our framework is able to capture both settings, in the sense that we can give FSSs for both, but interestingly, only the multi-bit one is local, so our general result only applies in the MB setting. We discuss this here; we give results for the SB setting in Section 5.2.

Our FSS CCA-MB for KEMs capturing indistinguishability under chosen-ciphertext attacks in the MB setting and the corresponding muc game are shown in Figure 9. It picks one challenge bit for each user which is used for encryption queries. During the game, the adversary can adaptively learn user's decryption keys via a corruption, thus also learning the user's challenge bit. In the end, the adversary has to guess the challenge bit of an uncorrupted user. Note that CCA-MB does not need oracle secret, i. e., it is local, and we can apply Theorem 4.2 to obtain the following result.

Theorem 5.2 *Let n, c be integers such that $0 \leq c < n$, and let $m = \lfloor (n-1)/(c+1) \rfloor$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\text{CCA-MB-muc-}(n,c)}_{\mathsf{KEM}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\text{CCA-MB-mu-}m}_{\mathsf{KEM}}$ such that*

$$\mathbf{Adv}^{\text{CCA-MB-muc-}(n,c)}_{\mathsf{KEM}}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}^{\text{CCA-MB-mu-}m}_{\mathsf{KEM}}(\mathcal{B}) \ .$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler D-FX.

As a second illustrative example, we consider one-way security of PKE under plaintext checking and ciphertext validity attacks [Den03, HHK17], which serves as a useful intermediate notion for the

OW-PCVA[PKE, h](init, ε):
1 $(ek, dk) \leftarrow\$ \mathsf{PKE.Kg}[\mathsf{h}]$
2 $\mathrm{St.ek} \leftarrow ek$; $\mathrm{St.dk} \leftarrow dk$
3 Return (ek, St)

OW-PCVA[PKE, h](enc, St):
4 $M \leftarrow\$ \mathsf{PKE.MS}$
5 $C \leftarrow\$ \mathsf{PKE.Enc}[\mathsf{h}](\mathrm{St.ek}, M)$
6 $\mathrm{St.S} \leftarrow \mathrm{St.S} \cup \{C\}$; Return (C, St)

OW-PCVA[PKE, h](cvo, C, St):
7 If $C \in \mathrm{St.S}$ then return $\bot$
8 $M \leftarrow \mathsf{PKE.Dec}[\mathsf{h}](\mathrm{St.dk}, C)$
9 Return $([\![M \in \mathsf{PKE.MS}]\!], \mathrm{St})$

OW-PCVA[PKE, h](pco, (M, C), St):
10 If $M \notin \mathsf{PKE.MS}$ then return $\bot$
11 $M' \leftarrow \mathsf{PKE.Dec}[\mathsf{h}](\mathrm{St.dk}, C)$
12 Return $([\![M' = M]\!], \mathrm{St})$

OW-PCVA[PKE, h](corr, ε, St):
13 Return $(\mathrm{St.dk}, \mathrm{St})$

OW-PCVA[PKE, h](fin, (M, C), St):
14 If $C \notin \mathrm{St.S}$ then return $(\mathsf{false}, \epsilon)$
15 $M' \leftarrow \mathsf{PKE.Dec}[\mathsf{h}](\mathrm{St.dk}, C)$
16 Return $([\![M' = M]\!], \epsilon)$

$\mathbf{G}^{\text{OW-PCVA-muc-}(n,c)}_{\mathsf{PKE}}$

Init:
1 $\mathsf{h} \leftarrow\$ \mathsf{Sch.ROS}$
2 For $i = 1, \ldots, n$ do
3 $\quad (ek_i, dk_i) \leftarrow\$ \mathsf{PKE.Kg}[\mathsf{h}]$
4 Return $(ek_1, \ldots, ek_n)$

Oracle(enc, i): // $i \in [1..n]$
5 $M \leftarrow\$ \mathsf{PKE.MS}$
6 $C \leftarrow\$ \mathsf{PKE.Enc}[\mathsf{h}](ek, M)$
7 $\mathrm{S}_i \leftarrow \mathrm{S}_i \cup \{C\}$; Return C

Oracle(cvo, (i, C)): // $i \in [1..n]$, $C \notin \mathrm{S}_i$
8 $M \leftarrow\$ \mathsf{PKE.Dec}[\mathsf{h}](dk_i, C)$
9 Return $[\![M \in \mathsf{PKE.MS}]\!]$

Oracle(pco, (i, M, C)): // $i \in [1..n]$, $M \in \mathsf{PKE.MS}$
10 Return $[\![\mathsf{PKE.Dec}[\mathsf{h}](dk_i, C) = M]\!]$

Corrupt(i): // $i \in [1..n]$
11 $\mathrm{CS} \leftarrow \mathrm{CS} \cup \{i\}$; Return dk_i

RO(x):
12 $h \leftarrow \mathsf{h}(x)$; Return h

Fin(i, M, C): // $i \in [1..n]$
13 If $i \in \mathrm{CS}$ or $C \notin \mathrm{S}_i$ then return false
14 $M' \leftarrow \mathsf{PKE.Dec}[\mathsf{h}](dk_i, C)$
15 Return $[\![M' = M]\!]$

Figure 10: **Left:** Formal security specification OW-PCVA[PKE, h], where the global setup is trivial. **Right:** Game $\mathbf{G}^{\text{OW-PCVA-muc-}(n,c)}_{\mathsf{PKE}}$ describing the execution of OW-PCVA[PKE, h] in the muc setting.

Fujisaki-Okamoto (FO) transform [FO99, FO13], which we will have a closer look at in Section 5.2. The FSS OW-PCVA is given in Figure 10. It is local, and thus we get:

Theorem 5.3 *Let n, c be integers such that $0 \leq c < n$, and let $m = \lfloor (n-1)/(c+1) \rfloor$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\text{OW-PCVA-muc-}(n,c)}_{\mathsf{PKE}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\text{OW-PCVA-mu-}m}_{\mathsf{PKE}}$ such that*

$$\mathbf{Adv}^{\text{OW-PCVA-muc-}n}_{\mathsf{PKE}}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}^{\text{OW-PCVA-mu-}m}_{\mathsf{PKE}}(\mathcal{B}) \ .$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler D-FX.

Authenticated Encryption. A (nonce-based) symmetric encryption (SE) scheme SE specifies deterministic algorithms SE.Enc and SE.Dec, a key length SE.kl and nonce space $\mathcal{N}$. The encryption algorithm takes as input a key $k \in \{0,1\}^{\mathsf{SE.kl}}$, a nonce $N \in \mathcal{N}$ and a message $M \in \{0,1\}^*$, and returns a ciphertext $C \in \{0,1\}^{\mathsf{SE.cl}(|M|)}$, where SE.cl is the ciphertext length function. The decryption algorithm takes as input k, N, C and returns either a message $M \in \{0,1\}^*$ or $\bot$ indicating failure. Correctness asks that for every $k \in \{0,1\}^{\mathsf{SE.kl}}$, $N \in \mathcal{N}$, $M \in \{0,1\}^*$, if $C \leftarrow \mathsf{SE.Enc}(k, N, M)$, then $\mathsf{SE.Dec}(k, N, C)$ returns M.

Multi-user security with corruptions for authenticated encryption was studied in [JSSOW17], in both the single and the multi-bit setting. As for PKE schemes, only the multi-bit setting can be

$\underline{\mathsf{AE\text{-}MB}[\mathsf{SE},\mathsf{h}](\mathtt{init},\varepsilon)\text{:}}$

$k \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.kl}}$; $\mathrm{St.k} \leftarrow k$

$b \leftarrow\!\!\$\ \{0,1\}$; $\mathrm{St.b} \leftarrow b$

Return $(\varepsilon, \mathrm{St})$

$\underline{\mathsf{AE\text{-}MB}[\mathsf{SE},\mathsf{h}](\mathtt{enc},(M,N),\mathrm{St})\text{:}}$

$C_0 \leftarrow\!\!\$\ \mathsf{SE.Enc}[\mathsf{h}](\mathrm{St.k}, M, N)$

$C_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.cl}(|M|)}$

$\mathrm{St.S}_i \leftarrow \mathrm{St.S}_i \cup \{(C_{\mathrm{St.b}}, N)\}$

Return $(C_{\mathrm{St.b}}, \mathrm{St})$

$\underline{\mathsf{AE\text{-}MB}[\mathsf{SE},\mathsf{h}](\mathtt{dec},(C,N),\mathrm{St})\text{:}}$

If $(C,N) \in \mathrm{St.S}$ then return $\perp$

$M_0 \leftarrow \mathsf{SE.Dec}[\mathsf{h}](\mathrm{St.k}, C, N)$; $M_1 \leftarrow \perp$

Return $(M_{\mathrm{St.b}}, \mathrm{St})$

$\underline{\mathsf{AE\text{-}MB}[\mathsf{SE},\mathsf{h}](\mathtt{corr},\varepsilon,\mathrm{St})\text{:}}$

Return $(\mathrm{St.k}, \mathrm{St})$

$\underline{\mathsf{AE\text{-}MB}[\mathsf{SE},\mathsf{h}](\mathtt{fin},b',\mathrm{St})\text{:}}$

Return $([\![\mathrm{St.b} = b']\!], \epsilon)$

$\underline{\text{Game } \mathbf{G}^{\mathsf{AE\text{-}MB\text{-}muc\text{-}(n,c)}}_{\mathsf{SE}}}$

INIT:

$\mathsf{h} \leftarrow\!\!\$\ \mathsf{Sch.ROS}$

For $i = 1, \ldots, n$ do

  $k_i \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.kl}}$; $b_i \leftarrow\!\!\$\ \{0,1\}$

ORACLE$(\mathtt{enc}, (i, M, N))$: // $i \in [1..n]$

$C_0 \leftarrow\!\!\$\ \mathsf{SE.Enc}[\mathsf{h}](k_i, M, N)$

$C_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.cl}(|M|)}$

$\mathrm{S}_i \leftarrow \mathrm{S}_i \cup \{C_{b_i}\}$; Return C_{b_i}

ORACLE$(\mathtt{dec}, (i, C, N))$: // $i \in [1..n]$, $(C,N) \notin \mathrm{S}_i$

$M_0 \leftarrow\!\!\$\ \mathsf{SE.Dec}[\mathsf{h}](dk_i, C, N)$; $M_1 \leftarrow \perp$

Return M_{b_i}

CORRUPT(i): // $i \in [1..n]$

$\mathrm{CS} \leftarrow \mathrm{CS} \cup \{i\}$; Return k_i

RO(x):

$h \leftarrow \mathsf{h}(x)$; Return h

FIN(i, b'): // $i \in [1..n]$

If $i \in \mathrm{CS}$ then return false

Return $[\![b_i = b']\!]$

Figure 11: **Left:** Formal security specification $\mathsf{AE\text{-}MB}[\mathsf{SE},\mathsf{h}]$ for multi-bit security of authenticated encryption. **Right:** Game $\mathbf{G}^{\mathsf{AE\text{-}MB\text{-}muc\text{-}(n,c)}}_{\mathsf{SE}}$ describing the execution of $\mathsf{AE\text{-}MB}[\mathsf{SE},\mathsf{h}]$ in the muc setting.

---

expressed via a local FSS which we provide in Figure 11. We then apply our general theorem to get the following.

Theorem 5.4 *Let n, c be integers such that $0 \le c < n$, and let $m = \lfloor (n-1)/(c+1) \rfloor$. Let SE be an SE scheme and let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\mathsf{AE\text{-}MB\text{-}muc\text{-}(n,c)}}_{\mathsf{SE}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\mathsf{AE\text{-}MB\text{-}mu\text{-}}m}_{\mathsf{SE}}$ such that*

$$\mathbf{Adv}^{\mathsf{AE\text{-}MB\text{-}muc\text{-}(n,c)}}_{\mathsf{SE}}(\mathcal{A}) \le e(c+1) \cdot \mathbf{Adv}^{\mathsf{AE\text{-}MB\text{-}mu\text{-}}m}_{\mathsf{SE}}(\mathcal{B}) \ .$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler $\mathsf{D\text{-}FX}$.

5.2 Indirect applications

Our general result does not apply to FSS which are not local, i. e., the game depends on some global oracle secret like a single challenge bit which is used across multiple users. This motivates us to study settings where the target game (and possibly the starting game) are described by non-local FSS, but where we can still achieve muc-security with a tightness loss of c via some transformation.

Here, we are also interested in settings where corruptions are not necessarily referring to signing or decryption keys. As an example, we show how to apply our framework to recover Coron's result on RSA-FDH signatures in Appendix A. We now provide more details for our new results.

SB SECURITY FOR KEMS. Bellare, Boldyreva and Micali [BBM00] introduced SB-mu security and also showed that su security implies mu security with a loss linear in the number of users. This initiated further study on tightly-secure PKE and KEM schemes in the mu setting [HJ12, LJYP14, GHKW16, GHK17, HLLG19]. Recently, we have also seen advances on tightly-secure schemes in the

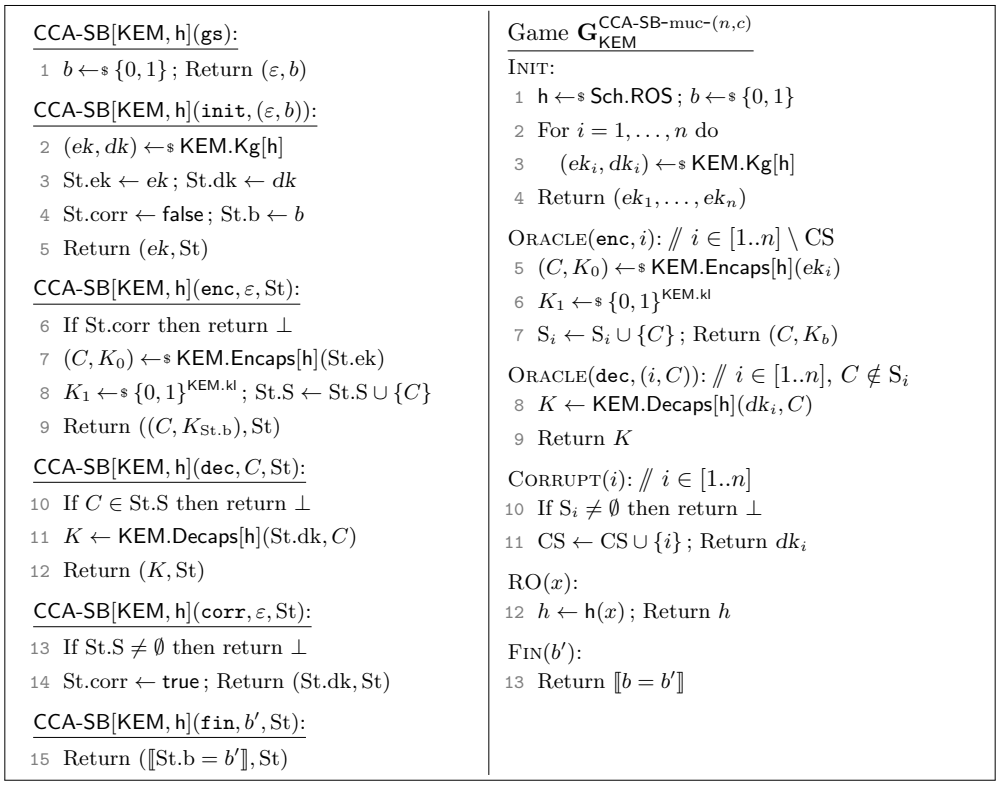

Figure 12: **Left:** Formal security specification CCA-SB[KEM, h] capturing security with a single challenge bit. **Right:** Game $\mathbf{G}^{\text{CCA-SB-muc-}(n,c)}_{\text{KEM}}$ describing the execution of CCA-SB[KEM, h] in the muc setting.

muc setting [LLP20, HLG23, HLWG23]. As for signature schemes, we can however observe that the latter schemes are less efficient in terms of size (of public keys and ciphertexts) and computation complexity than canonical schemes.

In Figure 12, we define the FSS CCA-SB for KEM schemes capturing indistinguishability under chosen-ciphertext attacks in the SB setting and its muc game. (The one for PKE is similar and provided in Figure 27 in Appendix B.) The game setup chooses a global challenge bit (stored as oracle secret *os*); thus the game is not local. This also requires to restrict queries to the challenge and corruption oracle, otherwise the adversary can trivially learn the challenge bit.

The main motivation to study security in the presence of adaptive corruptions for KEMs is that they are of high importance in practice. Here, the SB setting is particularly interesting for composition using the KEM/DEM paradigm [CS03]. While it has already been studied in the mu setting without corruptions [Zav12, GKP18], the result can be trivially extended to the muc setting using a CCA-SB-muc secure KEM. (This composition result was recently studied for simulation-based security definitions [Jae23].) For these reasons, we now aim to establish CCA-SB-muc security with improved tightness for practical encryption schemes.

HASHED ELGAMAL. On the left-hand side of Figure 13, we describe the KEM underlying the hashed ElGamal encryption scheme, which we denote by HEG. It is known that HEG can be proven tightly mu secure under the strong computational Diffie-Hellman (St-CDH) assumption [ABR01] which we define in terms of game $\mathbf{G}^{\text{St-CDH}}_{(\mathbb{G},p,g)}$ on the right-hand side of Figure 13. We denote the advantage of an adversary $\mathcal{A}$ in the game by $\mathbf{Adv}^{\text{St-CDH}}_{(\mathbb{G},p,g)}(\mathcal{A})$. For muc security, we only know the

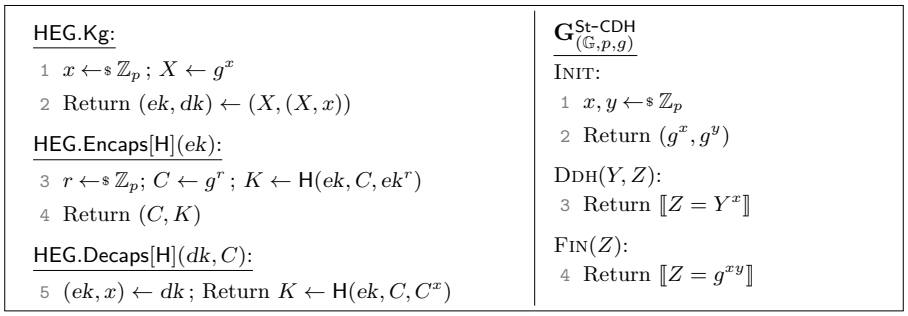

Figure 13: **Left:** Hashed ElGamal KEM HEG for a group $(\mathbb{G}, p, g)$, where HEG.ROS is the set of all $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\mathsf{HEG.kl}}$. **Right:** Game for the strong computational Diffie-Hellman assumption in group $(\mathbb{G}, p, g)$.

---

generic result with a security loss in the number of users. Thus, we ask whether we can do better using our technique and we provide an affirmative answer.

Theorem 5.5 *Let* HEG *be the KEM defined in Figure 13. Let n, c be integers such that $0 \le c < n$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\mathsf{CCA\text{-}SB\text{-}muc\text{-}}(n,c)}_{\mathsf{HEG}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}$ such that*

$$\mathbf{Adv}^{\mathsf{CCA\text{-}SB\text{-}muc\text{-}}(n,c)}_{\mathsf{HEG}}(\mathcal{A}) \le e(c+1) \cdot \mathbf{Adv}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}(\mathcal{B}) \ .$$

Let q_{ro} be the number of random oracle queries $\mathcal{A}$ makes. Then $\mathcal{B}$ makes at most q_{ro} queries to oracle Ddh*. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler* D-FX*, re-randomization of group elements and maintaining lists of encryption, decryption and random oracle queries.*

Proof: We modularize the proof as illustrated on the left of Figure 14, using the multi-user strong *asymmetric* CDH assumption with corruptions as introduced in [KPRR23] for which we can apply Theorem 4.2. We further use the fact that without corruptions, this assumption is tightly equivalent to the standard St-CDH assumption (cf. Figure 13). Finally, we show that its muc variant implies muc security of HEG.

Multi-User Strong Asymmetric CDH. In Figure 15 we define the asymmetric multi-user version of CDH with corruptions as a formal security specification k-St-ACDH$[(\mathbb{G}, p, g)]$ for an integer $k \ge 1$. The game requires to compute the Diffie-Hellman share of two group elements which can be chosen from two (distinct) sets of group elements. Asymmetric means that exponents of group elements of only one of the sets can be revealed. We can view these group elements as the users in the game. The other set of k elements is generated during game setup and part of the public parameters pp. As in $\mathbf{G}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}$, the game provides Oracle(ddh, $\cdot$) which can be queried for each user.

Note that without corruptions (i. e., game $\mathbf{G}^{k\text{-St-ACDH-mu-}n}_{(\mathbb{G},p,g)}$), the game is essentially a multi-user version of St-CDH and known to be tightly equivalent to single-user St-CDH which is the same as 1-St-ACDH.[1] The following lemma combines this fact with Theorem 4.2.

[1] More precisely, showing that St-CDH implies n-user St-CDH loses a factor of 2, cf. Lemma 9 in the full version of [KPRR23]. However, the asymmetry in k-St-ACDH-mu avoids this factor since the final output is restricted to a smaller set of solutions.

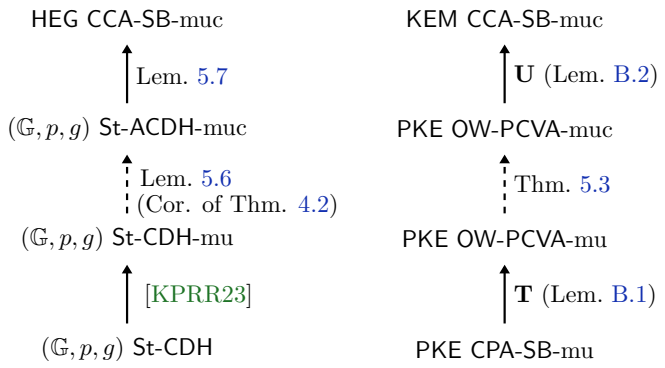

Figure 14: **Achieving CCA-SB-muc security for KEM:** We prove security for the hashed ElGamal KEM (top left) based on the St-CDH assumption (bottom left) via the intermediate St-ACDH-mu(c) assumption. We also lift the modular analysis of the FO transform [HHK17] to the muc setting showing CCA-SB-muc security for a KEM (top right) based on a CPA-SB-mu secure PKE (bottom right) via the intermediate notion of OW-PCVA-mu(c) security. Solid arrows denote tight transformations (in the ROM), dashed arrows apply our general cp-muc theorem with a loss of c.

---

k-St-ACDH$[(\mathbb{G}, p, g)]$(gs):
1 For $j = 1, \ldots, k$ do
2 $\quad y_j \leftarrow\!\!\$\ \mathbb{Z}_p$
3 $\mathbf{Y} \leftarrow [g^{y_1}, \ldots, g^{y_k}]$
4 Return $(pp, os) \leftarrow (\mathbf{Y}, \varepsilon)$

k-St-ACDH$[(\mathbb{G}, p, g)]$(init, $(\mathbf{Y}, \varepsilon)$):
5 $x \leftarrow\!\!\$\ \mathbb{Z}_p$; $X \leftarrow g^x$
6 St.x $\leftarrow x$; St.$\mathbf{Y} \leftarrow \mathbf{Y}$
7 Return (X, St)

k-St-ACDH$[(\mathbb{G}, p, g)]$(ddh, (Y, Z), St):
8 If $Y^{\mathrm{St.x}} = Z$ then return (true, St)
9 Return (false, St)

k-St-ACDH$[(\mathbb{G}, p, g)]$(corr, ε, St):
10 Return $((\mathrm{St.x}), \mathrm{St})$

k-St-ACDH$[(\mathbb{G}, p, g)]$(fin, (j^*, Z), St):
11 $Y_{j^*} \leftarrow \mathrm{St.}\mathbf{Y}[j^*]$
12 Return $([\![Y_{j^*}^{\mathrm{St.x}} = Z]\!], \epsilon)$

Game $\mathbf{G}^{k\text{-St-ACDH-muc-}(n,c)}_{(\mathbb{G},p,g)}$

Init:
1 For $i = 1, \ldots, n$ do
2 $\quad x_i \leftarrow\!\!\$\ \mathbb{Z}_p$; $X_i \leftarrow g^{x_i}$
3 For $j = 1, \ldots, k$ do
4 $\quad y_j \leftarrow\!\!\$\ \mathbb{Z}_p$; $Y_j \leftarrow g^{y_j}$
5 Return $(X_1, \ldots, X_n, Y_1, \ldots, Y_k)$

Oracle(ddh, (i, Y, Z)): // $i \in [1..n]$
6 Return $(Y^{x_i} = Z)$

Corrupt(i): // $i \in [1..n]$
7 CS $\leftarrow$ CS $\cup \{i\}$
8 Return x_i

Fin(i, j^*, Z): // $i \in [1..n]$
9 If $i \in$ CS then return false
10 Return $[\![Y_{j^*}^{x_i} = Z]\!]$

Figure 15: **Left:** Formal security specification k-St-ACDH$[(\mathbb{G}, p, g)]$ for the strong asymmetric CDH problem in $(\mathbb{G}, p, g)$, where $k \geq 1$ is an integer. **Right:** Game $\mathbf{G}^{k\text{-St-ACDH-muc-}(n,c)}_{(\mathbb{G},p,g)}$ describing the execution of k-St-ACDH$[(\mathbb{G}, p, g)]$ in the muc setting.

Lemma 5.6 *Let $n, k \geq 1$ and $c \geq 0$ such that $c < n$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{k\text{-St-ACDH-muc-}(n,c)}_{(\mathbb{G},p,g)}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}$ such that*

$$\mathbf{Adv}^{k\text{-St-ACDH-muc-}n}_{(\mathbb{G},p,g)}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}(\mathcal{B}) \ .$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler D-FX *and re-randomization of group elements.*

It remains to show that St-ACDH-muc tightly implies the security of HEG. We capture this in the following lemma.

Lemma 5.7 *Let* HEG *be the KEM defined in Figure 13. Let n, c be integers such that $0 \leq c < n$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\mathsf{CCA\text{-}SB\text{-}muc\text{-}}(n,c)}_{\mathsf{HEG}}$ that issues at most k queries to* Oracle(`enc`, $\cdot$). *Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{k\text{-St-ACDH-muc-}(n,c)}_{(\mathbb{G},p,g)}$ such that*

$$\mathbf{Adv}^{\mathsf{CCA\text{-}SB\text{-}muc\text{-}}(n,c)}_{\mathsf{HEG}}(\mathcal{A}) \leq \mathbf{Adv}^{k\text{-St-ACDH-muc-}(n,c)}_{(\mathbb{G},p,g)}(\mathcal{B}) \ .$$

Adversary $\mathcal{B}$ makes the same number of corruption queries as $\mathcal{A}$ makes. Let q_{ro} be the number of random oracle queries $\mathcal{A}$ makes. Then $\mathcal{B}$ makes at most q_{ro} queries to Oracle(`ddh`, $\cdot$). *The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for maintaining lists of decryption and random oracle queries.*

Proof: We construct an adversary $\mathcal{B}$ as described in Figure 16. It runs adversary $\mathcal{A}$ and simulates game $\mathbf{G}^{\mathsf{CCA\text{-}SB\text{-}muc\text{-}}(n,c)}_{\mathsf{HEG}}$ as follows.

To initialize the game, $\mathcal{B}$ queries its own Init oracle and receives a list of $n + k$ group elements, where it outputs the first n group elements as public keys. It uses a counter j for the remaining k group elements. On the j-th query to Oracle(`enc`, i), $\mathcal{B}$ will use the j-th group element as ciphertext C. If a user i was corrupted, $\mathcal{B}$ does not respond to encryption queries as specified by the game. Since $\mathcal{B}$ does not know the exponent to C, it cannot compute the key K using RO. Instead, it chooses K uniformly at random from $\{0,1\}^{\mathsf{KEM.kl}}$ and stores (C, K) in a set S_i. Note that as long as RO is not queried on the respective input, this simulation is perfect.

For queries to Oracle(`dec`, (i, C)), $\mathcal{B}$ keeps an additional set DS which stores tuples of the form (ek_i, C, K). If a query is repeated, the same key is given as output. If a new ciphertext is queried, $\mathcal{B}$ chooses a random key K and stores it in DS. In order to be consistent with random oracle queries, we will also add entries there, which we will describe below. Corruption queries, if allowed, are simply forwarded.

It remains to describe the simulation of RO. Queries are stored in a list HS and repeating queries are answered consistently. If ek belongs to one of the users in the game, $\mathcal{B}$ queries Oracle(`ddh`, $\cdot$) on (i, C, Z) provided by $\mathcal{A}$. If (ek_i, C, Z) is a valid DDH tuple, where C was previously output by a query to `enc`, then (since corruption and encryption queries are not allowed for the same user) $\mathcal{B}$ stops the execution of $\mathcal{A}$ and queries Fin. If C was previously queried to Oracle(`dec`, $\cdot$), $\mathcal{B}$ patches the random oracle using list DS. Otherwise, it adds a new entry. Note that until now we were looking at the case that the query forms a valid DDH tuple. If this is not the case, we only add an entry to HS and output a fresh K.

Adversary $\mathcal{B}$

▷ Run $\mathcal{A}$, responding to its oracle queries as follows:

$\mathbf{G}^{\text{CCA-SB-muc-}(n,c)}_{\mathsf{HEG}}$.Init:

1 $(X_1, \ldots, X_n, Y_1, \ldots, Y_n) \leftarrow \mathbf{G}^{k\text{-St-ACDH-muc-}(n,c)}_{(\mathbb{G},p,g)}$.Init

2 $j \leftarrow 0$; Return $(ek_1, \ldots, ek_n) \leftarrow (X_1, \ldots, X_n)$

$\mathbf{G}^{\text{CCA-SB-muc-}(n,c)}_{\mathsf{HEG}}$.Oracle($\mathtt{enc}, i$):

3 If $i \in \mathrm{CS}$ then return $\bot$

4 $j \leftarrow j + 1$; $C \leftarrow Y_j$; $K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{KEM.kl}}$

5 $\mathrm{S}_i \leftarrow \mathrm{S}_i \cup \{(C, K)\}$; Return (C, K)

$\mathbf{G}^{\text{CCA-SB-muc-}(n,c)}_{\mathsf{HEG}}$.Oracle($\mathtt{dec}, (i, C)$):

6 If $(C, \cdot) \in \mathrm{S}_i$ then return $\bot$

7 If $\exists K$ s. t. $(ek_i, C, K) \in \mathrm{DS}$ then return K

8 $K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{KEM.kl}}$; $\mathrm{DS} \leftarrow \mathrm{DS} \cup \{(ek_i, C, K)\}$; Return K

$\mathbf{G}^{\text{CCA-SB-muc-}(n,c)}_{\mathsf{HEG}}$.Corrupt($i$):

9 If $\mathrm{S}_i \neq \emptyset$ then return $\bot$

10 $\mathrm{CS} \leftarrow \mathrm{CS} \cup \{i\}$; Return $dk_i \leftarrow \mathbf{G}^{k\text{-St-ACDH-muc-}(n,c)}_{(\mathbb{G},p,g)}$.Corrupt($i$)

$\mathbf{G}^{\text{CCA-SB-muc-}(n,c)}_{\mathsf{HEG}}$.RO($ek, C, Z$):

11 If $\exists K$ s. t. $(ek, C, Z, K) \in \mathrm{HS}$ then return K

12 $K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{KEM.kl}}$

13 If $\exists i$ s. t. $ek = ek_i$ and $\mathbf{G}^{k\text{-St-ACDH-muc-}(n,c)}_{(\mathbb{G},p,g)}$.Oracle($\mathtt{ddh}, (i, C, Z)$) = true:

14 If $\exists j^*$ s. t. $C = Y_{j^*}$ then stop and return $\mathbf{G}^{k\text{-St-ACDH-muc-}(n,c)}_{(\mathbb{G},p,g)}$.Fin($i, j^*, Z$)

15 If $\exists K'$ s. t. $(ek, C, K') \in \mathrm{DS}$ then $K \leftarrow K'$

16 Else: $\mathrm{DS} \leftarrow \mathrm{DS} \cup \{(ek, C, K)\}$

17 $\mathrm{HS} \leftarrow \mathrm{HS} \cup \{(ek, M, C, K)\}$; Return K

$\mathbf{G}^{\text{CCA-SB-muc-}(n,c)}_{\mathsf{HEG}}$.Fin($i, b$):

18 $\mathbf{G}^{k\text{-St-ACDH-muc-}(n,c)}_{(\mathbb{G},p,g)}$.Fin($1, \bot, \bot$)

Figure 16: Adversary $\mathcal{B}$ for Theorem 5.7.

In order to win, $\mathcal{A}$ must query the random oracle on a valid pair of (ek_i, C, Z), otherwise K is uniformly random in any case. Further, if $\mathcal{A}$ issues such a query, $\mathcal{B}$ can win game $\mathbf{G}^{k\text{-St-ACDH-muc-}(n,c)}_{(\mathbb{G},p,g)}$, which concludes the proof. ▮

Theorem 5.5 now follows from Lemmas 5.6 and 5.7. ▮

Fujisaki-Okamoto transform. The Fujisaki-Okamoto (FO) transform [FO99,FO13] is a generic approach that turns any cpa secure PKE scheme into one that is cca secure in the random oracle model. (Here, we define CPA-SB in terms of CCA-SB, by restricting attention to adversaries that do not query the decryption oracle.) Especially since the announcement of NIST's post-quantum competition [Nat], the FO transform has seen adoption in the design of many schemes for practical use, including Kyber [BDK+18] which has been selected for standardization. The FO has been studied in the mu setting [DHK+21], showing that CPA-SB-mu security of the PKE scheme tightly implies CCA-SB-mu security of the transformed KEM.

Due to its practical relevance, we want to extend this analysis to the muc setting, targeting a tightness loss in the number of corruptions. In order to do so, we make use of the modular approach of the FO transform as considered in [HHK17]. The two transformations $\mathbf{T}$ and $\mathbf{U}$ are described in Figure 17 and combining them gives us a KEM scheme $\mathsf{KEM} = \mathbf{U}[\mathbf{T}[\mathsf{PKE}], s]$. Our result is Theorem 5.8. Recall that PKE is γ-spread [FO99] if $\max_{C \in \{0,1\}^*} \Pr[\mathsf{PKE.Enc}(ek, M) = C] \leq 2^{-\gamma}$

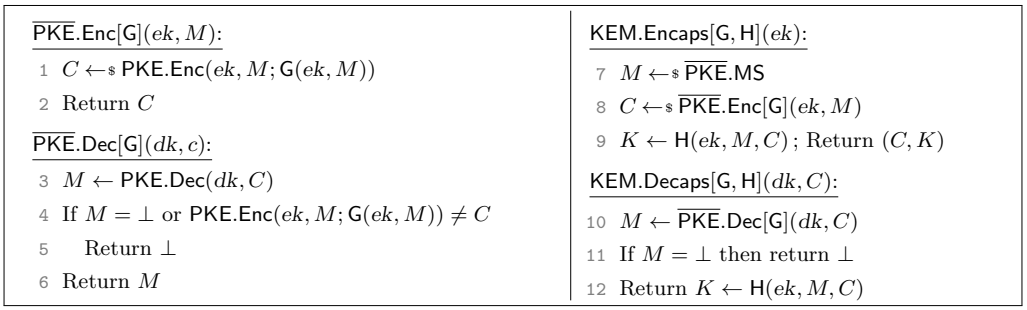

Figure 17: **Left:** PKE scheme $\overline{\mathsf{PKE}} = \mathbf{T}[\mathsf{PKE}]$ obtained from the T-Transform, where $\overline{\mathsf{PKE}}.\mathsf{ROS}$ is the set of all $\mathsf{G} : \{0,1\}^* \to \{0,1\}^{\mathsf{PKE.rl}}$ and $\overline{\mathsf{PKE}}.\mathsf{Kg} := \mathsf{PKE}.\mathsf{Kg}$. **Right:** KEM scheme $\mathsf{KEM} = \mathbf{U}[\overline{\mathsf{PKE}}, s]$ obtained from the U-Transform, where $\mathsf{KEM}.\mathsf{ROS}$ is the set of all (G, H) such that $\mathsf{G} \in \overline{\mathsf{PKE}}.\mathsf{ROS}$ and $\mathsf{H} : \{0,1\}^* \to \{0,1\}^s$. Further, $\mathsf{KEM}.\mathsf{Kg} := \overline{\mathsf{PKE}}.\mathsf{Kg}$.

---

for every $(ek, dk) \in \mathbf{Out}(\mathsf{PKE.Kg})$ and every $M \in \mathsf{PKE.MS}$, where the probability is taken over the coins of $\mathsf{PKE.Enc}$.

Theorem 5.8 *Let* PKE *be* γ*-spread and* $\mathsf{KEM} = \mathbf{U}[\mathbf{T}[\mathsf{PKE}], s]$ *the KEM scheme obtained from applying the modular FO transform as described in Figure 17. Let* n, c *be integers such that* $0 \leq c < n$*, and let* $m = \lfloor (n-1)/(c+1) \rfloor$*. Let* $\mathcal{A}$ *be an adversary for game* $\mathbf{G}^{\mathsf{CCA\text{-}SB\text{-}muc\text{-}}(n,c)}_{\mathsf{KEM}}$ *that issues at most* q_e *queries to* $\textsc{Oracle}(\mathtt{enc}, \cdot)$*,* q_d *queries to* $\textsc{Oracle}(\mathtt{dec}, \cdot)$ *and* q_{ro} *queries to both random oracles. Then we construct an adversary* $\mathcal{B}$ *for game* $\mathbf{G}^{\mathsf{CPA\text{-}SB\text{-}mu\text{-}}m}_{\mathsf{PKE}}$ *such that*

$$\mathbf{Adv}^{\mathsf{CCA\text{-}SB\text{-}muc\text{-}}(n,c)}_{\mathsf{KEM}}(\mathcal{A}) \leq 2e(c+1) \cdot \mathbf{Adv}^{\mathsf{CPA\text{-}SB\text{-}mu\text{-}}m}_{\mathsf{PKE}}(\mathcal{B}) + q_d \cdot 2^{-\gamma} + \frac{2nq_eq_{ro}+1}{|\mathsf{PKE.MS}|} \ .$$

Adversary $\mathcal{B}$ *makes the same number of encryption queries as* $\mathcal{A}$ *makes. The running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$ *plus the time for maintaining lists of encryption, decryption and random oracle queries and the time for an execution of the sampler* $\mathsf{D\text{-}FX}$*.*

We provide the full proof in Appendix B. Starting with a CPA-SB-mu secure scheme PKE, step (1) applies transform $\mathbf{T}$ to obtain a OW-PCVA-mu secure scheme $\overline{\mathsf{PKE}}$. In step (2), we apply Theorem 5.3 for OW-PCVA. Finally, in step (3), we apply the second part of the transform, namely, we take the OW-PCVA-muc secure scheme and use transform $\mathbf{U}$ to obtain the CCA-SB-muc scheme KEM. We want to note that Jaeger [Jae23] analyzes transform $\mathbf{U}$ in a simulation-based setting with corruptions, focusing on step (3) only.

Corruption-parameterized security for AKE. Security models for authenticated key exchange (AKE) protocols (e. g., [BR94]) allow for corruptions of long-term secret keys which makes the construction of tightly-secure AKE [BHJ+15, GJ18, LLGW20, JKRS21, HJK+21, PWZ23] non-trivial. All these works require some kind of muc security for their building blocks. As for signatures and encryption, we propose to study AKE with the number of corruptions being an additional adversary resource. This allows us to give improved tightness results for many AKE protocols.

In [CCG+19] Cohn-Gordon et al. analyze very efficient Diffie-Hellman based AKE protocols. We denote their main protocol by CCGJJ which is proven secure based on St-CDH with a loss linear in the number of users. They also show that this loss is optimal for their and similar DH-based protocols. By considering cp security, however, we can improve the previous bound to the number of corruptions. For this we make use of the analysis in [KPRR23] which proves tight security of the protocol based on the variant of St-CDH which we used for HEG. The improved bound for CCGJJ follows from combining Theorem 5.6 with [KPRR23, Thm. 3].

A common approach to generically construct AKE protocols is to use signatures for authentication, as for example in TLS. This means that long-term keys are now signing keys of a signature scheme. When aiming for tight security proofs, we thus require UF-muc security. We want to use our direct results for signatures here. Combining it with the results in [GJ18,DJ21,HJK+21,DG21], we get AKE with forward secrecy and with a security loss linear in the number of corruptions, requiring a UF-mu secure signature scheme. The other assumptions made remain untouched and their bound is already tight.

Improved bounds for selective opening security. In selective opening security, the adversary is given multiple PKE ciphertexts and is allowed to reveal not only the encrypted messages, but also the encryption randomness. We then want non-revealed ciphertexts to remain secure. In this setting, cp security concerns the number of opened ciphertexts. The security notion we are interested in is simulation-based selective opening security (in the following denoted by $\mathsf{SIM\text{-}SO\text{-}CCA}$) which is considered the strongest notion of security [BHK12, BDWY12].

While our framework is designed for game-based rather than simulation-based security, we show how to leverage our framework to improve the bounds of the schemes considered in [HJKS15]. In particular, we show that their Diffie-Hellman based PKE scheme can be proven secure assuming $\mathsf{St\text{-}CDH}$ with a loss linear in the number of opened ciphertexts rather than the total number of ciphertexts. The scheme is conceptually simpler than the scheme of [PZ22] (which has full tightness) and allows to save one group element in the ciphertext. Further, we show that the improved bound also applies to the RSA-based variant of the scheme. We provide formal definitions and theorem statements in Appendix C.

6 Optimality results

We want to show that the loss of c is indeed optimal for a large class of schemes and games. For this, we use the framework of Bader, Jager, Li and Schäge [BJLS16]. We adapt their definitions to our framework of formal security specifications.

Re-randomizable relations. A relation Rel specifies a generation algorithm $\mathsf{Rel.Gen}$ that via $(x, \omega) \leftarrow\!\!\$\ \mathsf{Rel.Gen}$ outputs an instance x and witness ω. The (possibly randomized) verification algorithm $d \leftarrow\!\!\$\ \mathsf{Rel.Vf}(x, \omega')$ returns a boolean decision d as to whether ω' is a valid witness for x. Correctness asks that $\Pr[\mathsf{Rel.Vf}(\mathsf{Rel.Gen})] = 1$, where the probability is over the coins of $\mathsf{Rel.Gen}$ (and possibly $\mathsf{Rel.Vf}$). Let $\mathsf{Rel.WS}(x) = \{\, \omega : \mathsf{Rel.Vf}(x, \omega) = \mathsf{true} \,\}$ be the witness set of x. We say an algorithm $\mathsf{Rel.ReRand}$ is a *re-randomizer* for Rel if on input an instance x and a witness ω such that $\mathsf{Rel.Vf}(\omega, x) = \mathsf{true}$, it outputs a witness ω' which is uniformly distributed over $\mathsf{Rel.WS}(x)$ with probability 1.

We define a witness recovery game for a scheme Rel via FFS REC. Its formal description and muc game $\mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}}$ are given in Figure 18. The adversary in this game gets n statements and may ask for witnesses of c of them. Its task is then to compute a witness for an "uncorrupted" statement.

Simple adversaries and (black-box) reductions. Our optimality results will involve a special class of "simple" adversaries issuing all corruptions at once. More precisely, a (n, c)-*simple adversary* for relation Rel is a randomized and stateful algorithm sA, which induces an adversary $\mathcal{A} = \mathcal{A}[\mathsf{sA}]$ for $\mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}}$ as described on the left-hand side of Figure 19.

We also consider a class of natural black-box reductions that transform an (n, c)-simple adversary sA for Rel into an adversary for a game $\mathbf{G}$. Hence, in the following, $\mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}}$ will play the role of the security game of the considered scheme, whereas $\mathbf{G}$ will be associated with

$\underline{\mathsf{REC}[\mathsf{Rel},\mathsf{h}](\texttt{init},(\varepsilon,\varepsilon))}$:
1 $(x,\omega) \leftarrow\!\!\$\ \mathsf{Rel.Gen}$
2 $(\mathrm{St.x},\mathrm{St.wit}) \leftarrow (x,\omega)$
3 Return (x,St)

$\underline{\mathsf{REC}[\mathsf{Rel},\mathsf{h}](\texttt{corr},\varepsilon,\mathrm{St})}$:
4 Return $(\mathrm{St.wit},\mathrm{St})$

$\underline{\mathsf{REC}[\mathsf{Rel},\mathsf{h}](\texttt{fin},\omega^*,\mathrm{St})}$:
5 Return $(\mathsf{Rel.Vf}(\omega^*,\mathrm{St.x}),\varepsilon)$

$\underline{\text{Game } \mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}}}$

INIT:
1 $\mathsf{h} \leftarrow\!\!\$\ \mathsf{Rel.ROS}$
2 For $i = 1,\ldots,n$ do
3 $\quad (x_i,\omega_i) \leftarrow\!\!\$\ \mathsf{Rel.Gen}$
4 Return $(x_1,\ldots,x_n)$

CORRUPT(i): // $i \in [1..n]$
5 $\mathrm{CS} \leftarrow \mathrm{CS} \cup \{i\}$; Return ω_i

RO(x):
6 $h \leftarrow \mathsf{h}(x)$; Return h

FIN(i,ω^*): // $i \in [1..n]$
7 If $i \in \mathrm{CS}$ then return false
8 Return $\mathsf{Rel.Vf}(\omega^*,x_i)$

Figure 18: **Left:** Formal security specification $\mathsf{REC}[\mathsf{Rel},\mathsf{h}]$ defining a witness recovery game for Rel. The global setup is trivial. **Right:** Game $\mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}}$ describing the execution of $\mathsf{REC}[\mathsf{Rel},\mathsf{h}]$ in the multi-user setting with corruptions.

Adversary $\mathcal{A}[\mathsf{sA}]$:
1 $(x_1,\ldots,x_n) \leftarrow\!\!\$\ \mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}}.\text{INIT}$
2 $st \leftarrow \varepsilon$
3 $(\mathrm{CS},st) \leftarrow\!\!\$\ \mathsf{sA}(x_1,\ldots,x_n,st)$
4 If $|\mathrm{CS}| > c$ then abort
5 For all $i \in \mathrm{CS}$ do
6 $\quad \omega_i \leftarrow \mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}}.\text{CORRUPT}(i)$
7 $(i^*,\omega^*) \leftarrow\!\!\$\ \mathsf{sA}((\omega_i)_{i\in\mathrm{CS}},st)$
8 $\mathsf{win} \leftarrow \mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}}.\text{FIN}(i^*,\omega^*)$

Adversary $\mathcal{R}[\mathsf{sR},\mathsf{sA}]$:
1 $st_{\mathsf{sR}} \leftarrow \varepsilon$
2 For $k = 1,\ldots,r$ do
3 $\quad st_{\mathsf{sA}} \leftarrow \varepsilon$
4 $\quad (x_1,\ldots,x_n,\rho,st_{\mathsf{sR}}) \leftarrow\!\!\$\ \mathsf{sR}(st_{\mathsf{sR}})$
5 $\quad (\mathrm{CS},st_{\mathsf{sA}}) \leftarrow \mathsf{sA}(x_1,\ldots,x_n,st_{\mathsf{sA}})$
6 $\quad ((\omega_i)_{i\in\mathrm{CS}},st_{\mathsf{sR}}) \leftarrow\!\!\$\ \mathsf{sR}(\mathrm{CS},st_{\mathsf{sR}})$
7 $\quad (i^*,\omega^*) \leftarrow \mathsf{sA}((\omega_i)_{i\in\mathrm{CS}},st_{\mathsf{sA}})$
8 $\quad st_{\mathsf{sR}} \leftarrow\!\!\$\ \mathsf{sR}(i^*,\omega^*,st_{\mathsf{sR}})$
9 $\mathsf{win} \leftarrow\!\!\$\ \mathbf{G}.\text{FIN}(st_{\mathsf{sR}})$

Figure 19: **Left:** Adversary $\mathcal{A}[\mathsf{sA}]$ induced by an (n,c)-simple adversary sA for Rel. **Right:** Adversary $\mathcal{R}[\mathsf{sR},\mathsf{sA}]$ defined by an (n,c,r)-simple reduction sR.

---

some underlying assumption. More formally, an (n,c,r)-*simple* $(\mathbf{G} \to \mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}})$-*reduction* sR is a stateful and randomized algorithm which, for every (n,c)-simple adversary sA, induces the adversary $\mathcal{R} = \mathcal{R}[\mathsf{sR},\mathsf{sA}]$ described on the right of Figure 19. Crucially, we require $\mathcal{R}$ to be a valid adversary for game $\mathbf{G}$. In particular, sR can make calls to the oracles provided by $\mathbf{G}$, except for calling its FIN procedure. Here, we do not allow the reduction to choose the random tape of the adversary, but note that it is easy to incorporate this into our formalization (and to extend Theorem 6.1 to take this into account) via standard techniques.[2]

STATELESS GAMES. We prove our first tightness result, showing that any (n,c,r)-*simple* $(\mathbf{G} \to \mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}})$-*reduction* sR must incur an advantage loss of roughly a factor $c+1$ for any re-randomizable relation Rel whenever the game $\mathbf{G}$ is *stateless*. A *stateless* game is one for which the

[2]More specifically, sR in Figure 19 outputs the random coins to be used as input to sA. In the proof, we then make the meta-adversary deterministic, and use an efficient and secure pseudorandom function, with a hard coded key, to efficiently simulate the random choices based on its input.

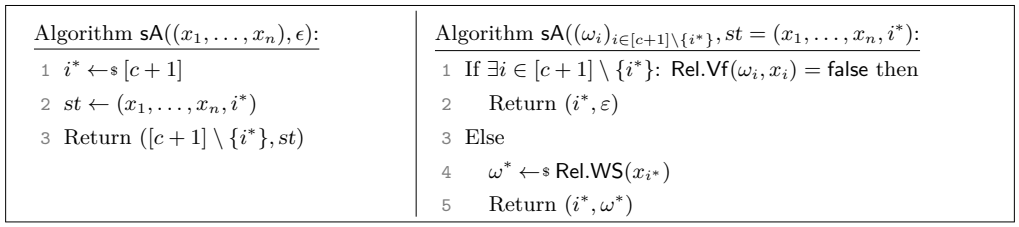

Figure 20: Hypothetical adversary in the proof of Theorem 6.1. The first and the second stage are given on the left-hand and right-hand sides, respectively.

INIT procedure sets some global variables, and queries to all game oracles never update these global variables, and the query output only depends on the global variables and the query input.

Our result will not require the game itself to be efficiently implementable, and often a game can have both an inefficient stateless version, as well as an equivalent efficient stateful version. For example, the efficient realization of the game defining PRF security would proceed by lazy sampling a random function, whereas an equivalent stateless version would initially sample a whole random function table. This is an example of game for which our result applies. Other games, such as those modeling the unforgeability of signatures under chosen message attacks, are inherently stateful, as the validity of the final forgery depends on previously issued signing queries.

Here, we will consider search games $\mathbf{G}$ (rather than distinguishing games) as the starting point for our reductions, and thus we associate an advantage metric $\mathbf{Adv}_{\mathbf{G}}(\mathcal{A}) = \Pr[\mathbf{G}(\mathcal{A})]$. However, it is straightforward to extend our result to distinguishing games.

TIGHTNESS RESULT. The proof of the following theorem generalizes that of Bader et al. [BJLS16] by extending it to stateless games and to a setting with c out of n corruptions.

Theorem 6.1 *Let n, c be integers such that $0 \le c < n$. Let $\mathbf{G}$ be a stateless game and let* Rel *be a relation with re-randomizer* ReRand. *There exists an (n,c)-simple adversary* sA *such that $\mathbf{Adv}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}}(\mathcal{A}[\mathsf{sA}]) = 1$, and such that for any (n,c,r)-simple $(\mathbf{G} \to \mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}})$-reduction* sR*, there exists an adversary $\mathcal{B}$ for $\mathbf{G}$ such that*

$$\mathbf{Adv}_{\mathbf{G}}(\mathcal{B}) \ge \mathbf{Adv}_{\mathbf{G}}(\mathcal{R}) - \frac{r}{c+1}$$

for $\mathcal{R} = \mathcal{R}[\mathsf{sR},\mathsf{sA}]$. The number of queries $\mathcal{B}$ makes to the oracles in $\mathbf{G}$ is at most c times the number of queries sR *makes. Further, the running time of $\mathcal{B}$ is at most $r\cdot(c+1)$ times the running time of* sR *plus running the relation's verification algorithm $rc\cdot(c+1)$ times plus running the re-randomizer once.*

Proof: We start by giving an inefficient hypothetical simple adversary sA which operates as described in Figure 20. The adversary initially selects the index i^* of one of the first $c+1$ users it does not corrupt, and then learns all remaining c witnesses. (Overall, thefore, it corrupts exactly c users.) Then, the adversary checks it received indeed valid witnesses for all $j \in [c+1]\setminus\{i^*\}$, and if so, it outputs a random witness for x_{i^*}. (The latter step is of course in general inefficient.)

The rest of the proof will culminate in a so-called "meta-adversary" $\mathcal{B}$ which will *efficiently* simulate the adversary $\mathcal{R} = \mathcal{R}[\mathsf{sR},\mathsf{sA}]$, up to some small error probability. We will get to $\mathcal{B}$ by providing a sequence of three adversaries $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2$ for game $\mathbf{G}$, where $\mathcal{B}_0 = \mathcal{R}$ and $\mathcal{B}_2$ is the meta-adversary $\mathcal{B}$. We describe them in Figure 21. In these descriptions, and the analysis, we also assume that sR is a deterministic algorithm. (It is easy to drop this assumption by having the initial state of sR by a sufficiently long random tape, instead of ε.)

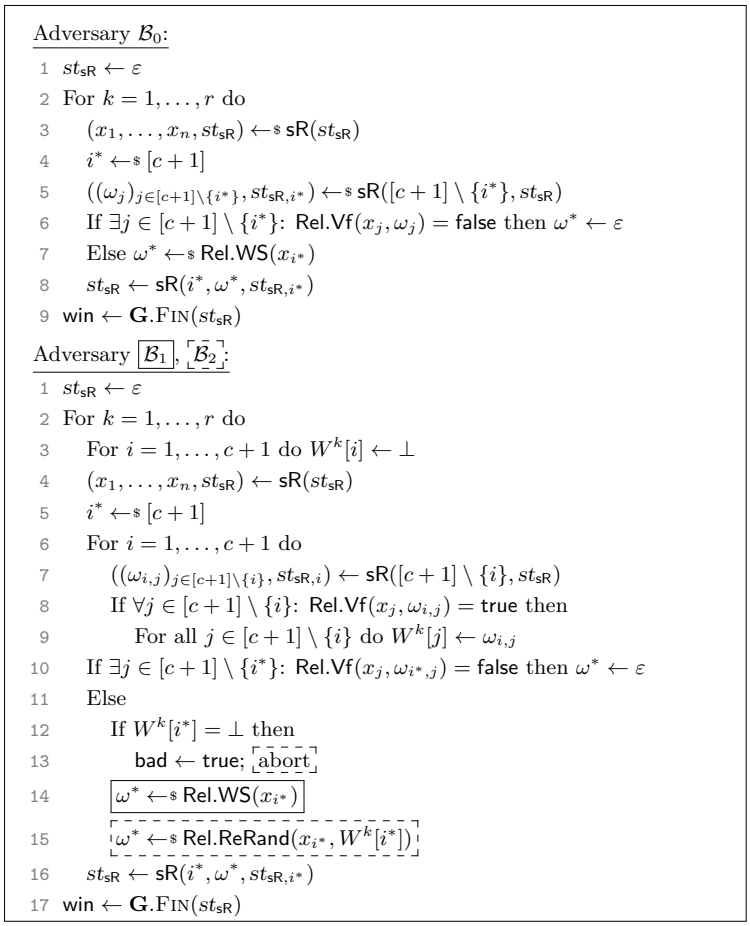

Figure 21: The sequence of adversaries $\mathcal{B}_0$, $\mathcal{B}_1$ and $\mathcal{B}_2$. Boxed statements are only executed by either of $\mathcal{B}_1$ or $\mathcal{B}_2$.

By inspection, it is clear that $\mathcal{B}_0 = \mathcal{R}$. The main difference between $\mathcal{B}_0$ and $\mathcal{B}_1$ is that in the latter, after i^* is sampled in an iteration, the reduction is run, on the *same* state st_{sR}, for every possible choice of the corruption sets $[c+1] \setminus \{i\}$, and not only on $[c+1] \setminus \{i^*\}$. Further, the adversary also keeps track of the witnesses output by sR when run on each i, as long as they are all indeed correct. This is done by keeping an array W^k in the k-th iteration, so that $W^k[i]$ is always set to $\perp$ or to a valid witness for x_i. Finally, $\mathcal{B}_1$ sets a flag bad whenever all witnesses output by the reduction for the case $i = i^*$ are indeed correct, yet $W^k[i^*]$ has not been set.

Note that sR can make calls to $\mathbf{G}$'s procedures, but because the game is stateless, these calls do not affect the overall state of $\mathbf{G}$, and the behavior of $\mathcal{B}_1$ is only affected by the state st_{sR,i^*} output for the case $i = i^*$. Therefore, these additional executions to sR do not modify the behavior with respect to $\mathcal{B}_0$. Thus,

$$\mathbf{Adv}_{\mathbf{G}}(\mathcal{R}) = \Pr\left[\mathbf{G}(\mathcal{B}_0)\right] = \Pr\left[\mathbf{G}(\mathcal{B}_1)\right]. \tag{34}$$

Moving to the final adversary $\mathcal{B}_2$, we make two modifications. One is that if the flag bad is set to true, the adversary actually aborts. Further, instead of sampling ω^* (inefficiently) from the witness set $\mathsf{Rel.WS}(x_{i^*})$, we sample from $\mathsf{Rel.ReRand}(x_{i^*}, W^k[i^*])$. The latter is identical, because the instruction is only executed if $\mathcal{B}_2$ has not previously aborted, and thus $W^k[i^*]$ is set to a valid

witness of x_{i^*}. Therefore, $\mathbf{G}(\mathcal{B}_1)$ and $\mathbf{G}(\mathcal{B}_2)$ are identical until bad is set to true, and thus

$$\Pr\left[\mathbf{G}(\mathcal{B}_1)\right] - \Pr\left[\mathbf{G}(\mathcal{B}_2)\right] \leq \Pr\left[\mathbf{G}(\mathcal{B}_2) \text{ sets } \mathsf{bad}\right] . \tag{35}$$

Further, as we set $\mathcal{B} = \mathcal{B}_2$, we have $\Pr\left[\mathbf{G}(\mathcal{B}_2)\right] = \mathbf{Adv}_{\mathbf{G}}(\mathcal{B})$. By (34) and (35) we have

$$\begin{aligned} \mathbf{Adv}_{\mathbf{G}}(\mathcal{R}) &= \Pr\left[\mathbf{G}(\mathcal{B}_1)\right] - \Pr\left[\mathbf{G}(\mathcal{B}_2)\right] + \Pr\left[\mathbf{G}(\mathcal{B}_2)\right] \\ &\leq \Pr\left[\mathbf{G}(\mathcal{B}_2) \text{ sets } \mathsf{bad}\right] + \mathbf{Adv}_{\mathbf{G}}(\mathcal{B}) . \end{aligned} \tag{36}$$

To upper bound $\Pr\left[\mathbf{G}(\mathcal{B}_2) \text{ sets } \mathsf{bad}\right]$, in a particular iteration k, for each $i \in [c+1]$, define the event $\textsc{BadRun}^k(i)$ as the event that there exists $j \in [c+1] \setminus \{i\}$ such that $\omega_{i,j}$ is invalid. We also let, for $\ell \in [c+1]$,

$$\textsc{BadPick}^k(\ell) = \neg\textsc{BadRun}^k(\ell) \wedge \bigwedge_{i \in [c+1]\setminus\{\ell\}} \textsc{BadRun}^k(i)$$

Note that the events $\textsc{BadRun}^k(i)$ are all determined by the value st_{sR} when entering the For loop on Line 6. We observe that bad is set to true in a particular iteration $k \in \{1, \ldots, r\}$ if and only if $\textsc{BadPick}^k(i^*)$ happens. However, note that $\textsc{BadPick}^k(\ell)$ can only happen for at most a single ℓ. Thus, by the union bound,

$$\Pr\left[\mathbf{G}(\mathcal{B}_2) \text{ sets } \mathsf{bad}\right] \leq \sum_{k=1}^{r} \Pr\left[\textsc{BadPick}^k(i^*)\right] \leq \frac{r}{c+1} .$$

This concludes the proof. ∎

Interpretation of the above result. To explain optimality, let us define tightness more formally. Let sA be an (n,c)-simple adversary for $\mathbf{G}$ running in time $t_{\mathcal{A}} = t_{\mathsf{sA}}$ with advantage $\varepsilon_{\mathcal{A}} = \mathbf{Adv}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}}(\mathcal{A}[\mathsf{sA}])$. Following [BJLS16], we say that an (n,c,r)-simple $(\mathbf{G} \to \mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}})$-reduction sR running in time t_{sR} with advantage $\varepsilon_{\mathcal{R}} = \mathbf{Adv}_{\mathbf{G}}(\mathcal{R}[\mathsf{sR},\mathsf{sA}])$ *loses* a factor ℓ if $t_{\mathcal{R}}/\varepsilon_{\mathcal{R}} \geq \ell \cdot t_{\mathcal{A}}/\varepsilon_{\mathcal{A}}$, where $t_{\mathcal{R}} = t_{\mathsf{sR}} + r \cdot t_{\mathsf{sA}}$.

Now we want to look closer at Theorem 6.1. Applying the bound gives us

$$\frac{t_{\mathcal{R}}}{\varepsilon_{\mathcal{R}}} = \frac{t_{\mathsf{sR}} + r \cdot t_{\mathsf{sA}}}{\varepsilon_{\mathcal{R}}} \geq \frac{r \cdot t_{\mathsf{sA}}}{\varepsilon_{\mathcal{B}} + r/(c+1)} = r\left(\varepsilon_{\mathcal{B}} + \frac{r}{c+1}\right)^{-1} \cdot \frac{t_{\mathsf{sA}}}{1} = \frac{r(c+1)}{\varepsilon_{\mathcal{B}}(c+1) + r} \cdot \frac{t_{\mathcal{A}}}{\varepsilon_{\mathcal{A}}} .$$

In the last step we use that the reduction must work for any (n,c)-simple adversary and, in particular, for the adversary sA with advantage 1 that we construct in the proof. We conclude that if $\varepsilon_{\mathcal{B}}$ is small, then $\ell \approx c+1$ and $\mathcal{R}$ must lose a factor $c+1$.

Example 1: CPA-secure KEM. The above theorem naturally covers non-interactive complexity assumptions as considered by [BJLS16] which are, by definition, stateless. As a more interesting example, the result also applies when using $\mathbf{G}^{\mathsf{CPA\text{-}SB\text{-}mu\text{-}}n}_{\mathsf{KEM}}$ for KEM schemes as a starting game $\mathbf{G}$ and a suitable relation, $\mathsf{Rel}[\mathsf{KEM}]$, which we provide in Figure 22. That is, if there exists an efficient re-randomizer $\mathsf{Rel}[\mathsf{KEM}].\mathsf{ReRand}$, then any (n,c,r)-simple $(\mathbf{G}^{\mathsf{CPA\text{-}SB\text{-}mu\text{-}}n}_{\mathsf{KEM}} \to \mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}[\mathsf{KEM}]})$-reduction must lose a factor $c+1$.

All schemes for which the above relation is unique, i. e., there exists exactly one decryption key such that the relation holds, have a trivial efficient re-randomizer. An example is the ElGamal KEM.

In order to conclude that the optimality result also holds when the target game is the corresponding muc game (rather than the relation game), we describe in Figure 22 how to transform *any* simple adversary sA for $\mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}[\mathsf{KEM}]}$ into an equivalent adversary $\mathcal{A}'[\mathsf{sA}]$ for game $\mathbf{G}^{\mathsf{CPA\text{-}SB\text{-}muc\text{-}}(n,c)}_{\mathsf{KEM}}$. Following [BJLS16], the existence of a black-box reduction from $\mathbf{G}^{\mathsf{CPA\text{-}SB\text{-}mu\text{-}}n}_{\mathsf{KEM}}$

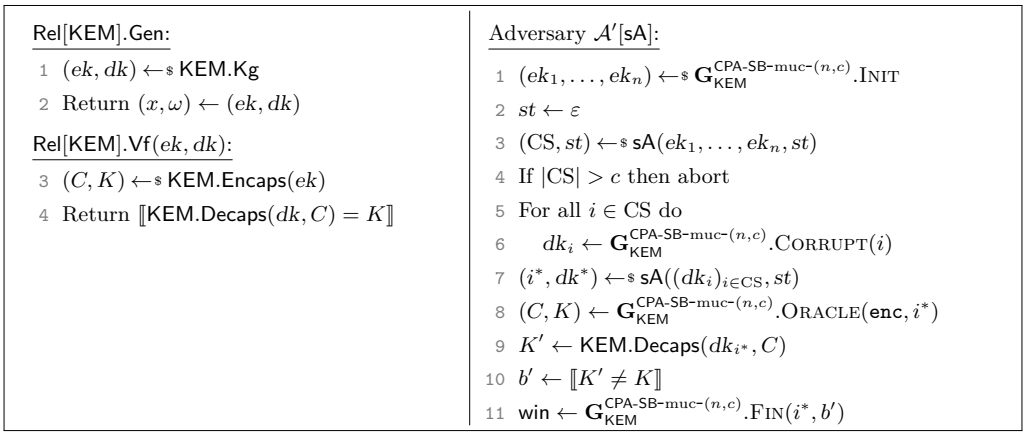

Figure 22: **Left:** Relation Rel[KEM] for a KEM scheme KEM. **Right:** The adversary $\mathcal{A}'[\mathsf{sA}]$ induced by an (n, c)-simple adversary sA for relation Rel[KEM].

---

to $\mathbf{G}_{\mathsf{KEM}}^{\text{CPA-SB-muc-}(n,c)}$, rewinding the given adversaries r times, implies the existence of an (n, c, r)-simple $(\mathbf{G}_{\mathsf{KEM}}^{\text{CPA-SB-mu-}n} \to \mathbf{G}_{\mathsf{Rel[KEM]}}^{\text{REC-muc-}(n,c)})$-reduction sR with the same loss. Roughly, this is because we can build sR which, when given access to sA, behaves exactly the same as the reduction using $\mathcal{A}'[\mathsf{sA}]$. As in [BJLS16], we do not make this argument fully formal, as it requires formalizing a more general notion of black-box reduction.

Generalizing beyond stateless games. We proceed to generalizing Theorem 6.1 to an arbitrary, not necessarily stateless, game $\mathbf{G}$ via the notion of a *branching adversary* for $\mathbf{G}$. Such an adversary is defined by a pair of algorithms $\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}$ which can (implicitly) call procedures from $\mathbf{G}$, except for Fin, and such that the adversaries

$$\mathcal{B} = \mathcal{B}[\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}] \ , \quad \mathcal{B}_\perp = \mathcal{B}[\mathsf{B}_{\mathsf{main}}, \perp]$$

defined according to the transformation on the left-hand side of Figure 23 are indeed valid adversaries for $\mathbf{G}$, where $\perp$ is a function that does nothing. In particular, $\mathsf{B}_{\mathsf{main}}$ is stateful, and proceeds in stages. At the end of each stage, it outputs its state st, a side state st', along with a Boolean value halt. If halt is false, the execution continues, until halt is true. In that case, the state of $\mathsf{B}_{\mathsf{main}}$ is interpreted as the input to Fin. At each step, $\mathsf{B}_{\mathsf{side}}$ is also executed on input state st', along with a step counter. The output of $\mathsf{B}_{\mathsf{side}}$ is simply ignored. For such a branching adversary, we define

$$\mathbf{Adv}_{\mathbf{G}}^{\text{branch}}(\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}) = \Pr\left[\mathbf{G}(\mathcal{B})\right] - \Pr\left[\mathbf{G}(\mathcal{B}_\perp)\right] ,$$

where $\mathcal{B}$ and $\mathcal{B}_\perp$. In other words, we are interested in how much the actions of $\mathsf{B}_{\mathsf{side}}$ alter the execution of the adversary as defined by $\mathsf{B}_{\mathsf{main}}$.

Intuitively, Theorem 6.1 remains meaningful for games where $\mathbf{Adv}_{\mathbf{G}}^{\text{branch}}(\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}})$ is generally small for a suitable branching adversary. Indeed, we can easily generalize Theorem 6.1 to *any* game $\mathbf{G}$, by relaxing the statement that $\Pr\left[\mathbf{G}(\mathcal{B}_0)\right] = \Pr\left[\mathbf{G}(\mathcal{B}_1)\right]$ in the proof to

$$\Pr\left[\mathbf{G}(\mathcal{B}_0)\right] \leq \Pr\left[\mathbf{G}(\mathcal{B}_1)\right] + \mathbf{Adv}_{\mathbf{G}}^{\text{branch}}(\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}})$$

for a suitable branching adversary $\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}$ defined as on the right-hand size of Figure 23. Then, the quantitative statement of the theorem now becomes

$$\mathbf{Adv}_{\mathbf{G}}(\mathcal{B}) \geq \mathbf{Adv}_{\mathbf{G}}(\mathcal{R}) - \mathbf{Adv}_{\mathbf{G}}^{\text{branch}}(\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}) - \frac{r}{c+1} \ .$$

Adversary $\mathcal{B}[\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}]$:

$st \leftarrow \varepsilon$; $\mathsf{it} \leftarrow 0$; $\mathsf{halt} \leftarrow \mathsf{false}$
While $\neg\mathsf{halt}$ do
  $\mathsf{it} \leftarrow \mathsf{it} + 1$
  $(st, st', \mathsf{halt}) \leftarrow\$ \mathsf{B}_{\mathsf{main}}(st, \mathsf{it})$
  $v \leftarrow\$ \mathsf{B}_{\mathsf{side}}(st', \mathsf{it})$
$\mathsf{win} \leftarrow \mathbf{G}.\textsc{Fin}(st)$

Algorithm $\mathsf{B}_{\mathsf{main}}(st_{\mathsf{sR}}, \mathsf{it})$:

$(x_1, \ldots, x_n, st_{\mathsf{sR}}) \leftarrow\$ \mathsf{sR}(st_{\mathsf{sR}})$
$i^* \leftarrow\$ [c+1]$
$st' \leftarrow (st_{\mathsf{sR}}, i^*)$
$\mathrm{CS} \leftarrow [c+1] \setminus \{i^*\}$
$((\omega_j)_{j\in[c+1]\setminus\{i^*\}}, st_{\mathsf{sR}}) \leftarrow\$ \mathsf{sR}(\mathrm{CS}, st_{\mathsf{sR}})$
If $\exists j \in \mathrm{CS}$: $\mathsf{Rel.Vf}(x_j, \omega_j) = \mathsf{false}$ then
  $\omega^* \leftarrow \varepsilon$
Else
  $\omega^* \leftarrow\$ \mathsf{Rel.WS}(x_{i^*})$
$st_{\mathsf{sR}} \leftarrow \mathsf{sR}(i^*, \omega^*, st_{\mathsf{sR}})$
If $\mathsf{it} < r$ then $\mathsf{halt} \leftarrow \mathsf{false}$ else $\mathsf{halt} \leftarrow \mathsf{true}$
Return $(st_{\mathsf{sR}}, st', \mathsf{halt})$

Algorithm $\mathsf{B}_{\mathsf{side}}(st' = (st_{\mathsf{sR}}, i^*), \mathsf{it})$:

$\mathrm{CS} \leftarrow [c+1] \setminus \{i^*\}$
For all $i \in \mathrm{CS}$ do
  $\mathrm{CS}_i \leftarrow [c+1] \setminus \{i\}$
  $((\omega_{i,j})_{j\in[c+1]\setminus\{i\}}, st_{\mathsf{sR},i}) \leftarrow \mathsf{sR}(\mathrm{CS}_i, st_{\mathsf{sR}})$
Return ϵ

Figure 23: **Left:** The adversary $\mathcal{B}[\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}]$ for $\mathbf{G}$ defined by the branching adversary $(\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}})$. **Right:** The branching adversary used to generalize Theorem 6.1 to an arbitrary game $\mathbf{G}$.

It is clear that if $\mathbf{G}$ is stateless, then $\mathbf{Adv}^{\mathrm{branch}}_{\mathbf{G}}(\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}) = 0$ for every $\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}$, since the executions of $\mathsf{B}_{\mathsf{side}}$ do not affect the state of $\mathbf{G}$.

Example 2: Randomized Signatures. Towards a more interesting example, consider the obvious extension SUF of UF (defined in Figure 4) that describes strong unforgeability of a randomized signature scheme Sig. Also assume that there exists a value γ such that for all choices of h, all messages M, and all signing keys sk, the produced signature $\sigma \leftarrow\$ \mathsf{Sig.Sign}[\mathsf{h}](sk, M)$ takes each possible value with probability at most $2^{-\gamma}$ (or, in other words, its *min-entropy* is at least γ). Then, for all n, with $\mathbf{G} = \mathbf{G}^{\mathsf{SUF\text{-}mu\text{-}}n}_{\mathsf{Sig}}$, and all $\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}$,

$$\mathbf{Adv}^{\mathrm{branch}}_{\mathbf{G}}(\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}) \leq q_s/2^{\gamma} \ , \tag{37}$$

where q_s is a bound on the overall number of signatures obtained by all executions of $\mathsf{B}_{\mathsf{side}}$. To prove (37), it is easy to verify that the advantage is upper bounded by the probability that one of the executions of $\mathsf{B}_{\mathsf{side}}$ makes a signing query which corresponds to the same user, message, and signature used for the final forgery output by $\mathsf{B}_{\mathsf{main}}$. This probability is upper bounded by $q_s/2^{\gamma}$, using the min-entropy of signatures and union bound over the total number of signing queries.

For signatures we specify a relation $\mathsf{Rel}[\mathsf{Sig}]$ as shown on the left-hand side of Figure 24. That is, if there exists an efficient re-randomizer $\mathsf{Rel}[\mathsf{Sig}].\mathsf{ReRand}$, then any (n, c, r)-simple $(\mathbf{G}^{\mathsf{SUF\text{-}mu\text{-}}n}_{\mathsf{Sig}} \rightarrow \mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}[\mathsf{Sig}]})$-reduction must lose a factor $c+1$. For completeness, and similar to $\mathsf{Rel}[\mathsf{KEM}]$ considered in Section 6, we show how to transform *any* simple adversary sA for $\mathbf{G}^{\mathsf{REC\text{-}muc\text{-}}(n,c)}_{\mathsf{Rel}[\mathsf{Sig}]}$ into an equivalent adversary $\mathcal{A}'[\mathsf{sA}]$ for game $\mathbf{G}^{\mathsf{SUF\text{-}muc\text{-}}(n,c)}_{\mathsf{Sig}}$. Adversary $\mathcal{A}'[\mathsf{sA}]$ is given on the right-hand side of Figure 24, which gives us the desired optimality result.

As a concrete example consider the Boneh-Boyen signature scheme [BB08] in bilinear groups for which the relation $\mathsf{Rel}[\mathsf{Sig}]$ is unique and thus efficiently re-randomizable. Further, signatures have min-entropy $\log(p)$, where p is the size of the group.

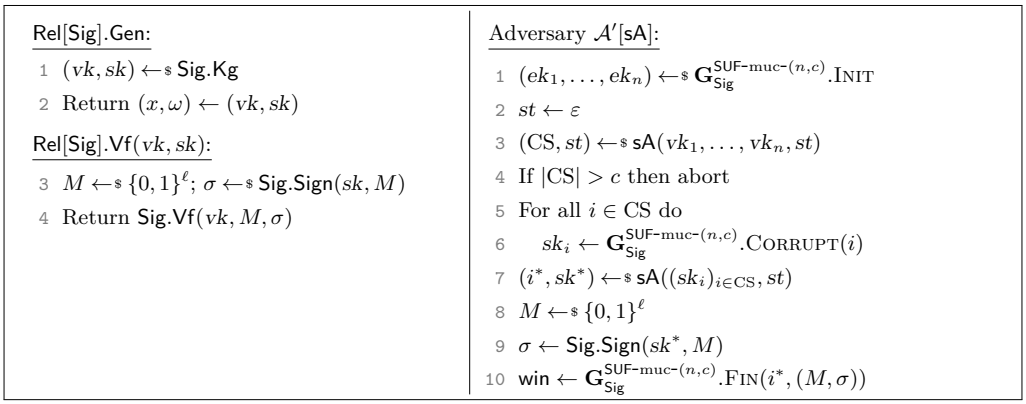

Figure 24: **Left:** Relation Rel[Sig] for a signature scheme Sig, where we assume w.l.o.g. the message space is finite, i. e., $\{0,1\}^\ell$ for some ℓ. **Right:** The adversary $\mathcal{A}'[\mathsf{sA}]$ induced by an (n, c)-simple adversary sA for relation Rel[Sig].

Example 3: CCA-secure KEM. We can make a similar argument for KEM schemes. The CCA-mu game is not stateless since it disallows to query challenge ciphertexts to the decryption oracle. Thus, we need to bound the probability that either any such query happens. The idea is to also use min-entropy (or γ-spreadness), similar to that defined for PKE in Section 5.2. Then, for all n, with $\mathbf{G} \in \{\mathbf{G}_{\mathsf{PKE}}^{\text{CCA-SB-mu-}n}, \mathbf{G}_{\mathsf{PKE}}^{\text{CCA-MB-mu-}n}\}$, and all $\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}$,

$$\mathbf{Adv}_{\mathbf{G}}^{\text{branch}}(\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}}) \leq (q_e q_d)/2^\gamma \ , \tag{38}$$

where q_e and q_d are the bound on the overall number of encryption resp. decryption queries made in $\mathsf{B}_{\mathsf{main}}$ and all executions of $\mathsf{B}_{\mathsf{side}}$. To prove (38), observe that the advantage is upper bounded by the probability that any execution of $\mathsf{B}_{\mathsf{main}}$ or $\mathsf{B}_{\mathsf{side}}$ makes a decryption query which corresponds to a previously received challenge ciphertext. Recall that ciphertexts have min-entropy γ and those received in $\mathsf{B}_{\mathsf{side}}$ do not influence st_{sR}, meaning that a subsequent execution of $\mathsf{B}_{\mathsf{main}}$ or $\mathsf{B}_{\mathsf{side}}$ with st_{sR} will make an invalid query can be upper bounded by γ-spreadness. The same argument applies to $\mathsf{B}_{\mathsf{side}}$. The state given to $\mathsf{B}_{\mathsf{side}}$ in the k-th iteration does not contain information about the ciphertexts received in the k-th execution of $\mathsf{B}_{\mathsf{main}}$. Union bound over all decryption queries gives us the bound. Note that it does not matter here whether there is a single or multiple challenge bits.

Note that the relation Rel[KEM] and adversary $\mathcal{A}'[\mathsf{sA}]$ from Figure 22 apply to CCA security as well.

References

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer, Heidelberg, April 2001. 7, 27

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008. 39

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EURO-*

CRYPT 2000, volume 1807 of *LNCS*, pages 259–274. Springer, Heidelberg, May 2000. 3, 7, 23, 26

[BDK+18] Joppe Bos, Leo Ducas, Eike Kiltz, T Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367, 2018. 31

[BDL+12] Daniel J Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of cryptographic engineering*, 2(2):77–89, 2012. 4

[BDWY12] Mihir Bellare, Rafael Dowsley, Brent Waters, and Scott Yilek. Standard security does not imply security against selective-opening. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 645–662. Springer, Heidelberg, April 2012. 8, 33

[Ber15] Daniel J. Bernstein. Multi-user Schnorr security, revisited. Cryptology ePrint Archive, Report 2015/996, 2015. https://eprint.iacr.org/2015/996. 3, 4

[BHJ+15] Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. Tightly-secure authenticated key exchange. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 629–658. Springer, Heidelberg, March 2015. 2, 5, 32

[BHK12] Florian Böhl, Dennis Hofheinz, and Daniel Kraschewski. On definitions of selective opening security. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 522–539. Springer, Heidelberg, May 2012. 8, 33

[BJLS16] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 273–304. Springer, Heidelberg, May 2016. 2, 8, 33, 35, 37, 38

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313–317, 1979. 3

[BNN07] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 411–422. Springer, Heidelberg, July 2007. 9

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1994. 7, 32

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, Heidelberg, May 1996. 2

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. 9, 15

[CCG+19] Katriel Cohn-Gordon, Cas Cremers, Kristian Gjøsteen, Håkon Jacobsen, and Tibor Jager. Highly efficient key exchange protocols with optimal tightness. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 767–797. Springer, Heidelberg, August 2019. 3, 7, 8, 32

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Heidelberg, August 2000. 2, 5, 8, 11, 46

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003. 27

[DDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *26th ACM STOC*, pages 522–533. ACM Press, May 1994. 3

[Den03] Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *9th IMA International Conference on Cryptography and Coding*, volume 2898 of *LNCS*, pages 133–151. Springer, Heidelberg, December 2003. 24

[DG21] Hannah Davis and Felix Günther. Tighter proofs for the SIGMA and TLS 1.3 key exchange protocols. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS 21, Part II*, volume 12727 of *LNCS*, pages 448–479. Springer, Heidelberg, June 2021. 3, 4, 7, 33

[DGJL21] Denis Diemert, Kai Gellert, Tibor Jager, and Lin Lyu. More efficient digital signatures with tight multi-user security. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 1–31. Springer, Heidelberg, May 2021. 2, 5

[DHK+21] Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, and Gregor Seiler. Faster lattice-based KEMs via a generic fujisaki-okamoto transform using prefix hashing. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2722–2737. ACM Press, November 2021. 31, 49

[DJ21] Denis Diemert and Tibor Jager. On the tight security of TLS 1.3: Theoretically sound cryptographic parameters for real-world deployments. *Journal of Cryptology*, 34(3):30, July 2021. 3, 4, 7, 33

[Dra23] SSL Dragon. Top 12 revealing ssl stats you should know, May 2023. https://www.ssldragon.com/blog/ssl-stats/. 3

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999. 25, 31

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013. 25, 31

[GGJJ23] Kai Gellert, Kristian Gjøsteen, Håkon Jacobsen, and Tibor Jager. On optimal tightness for key exchange with full forward secrecy via key confirmation. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 297–329. Springer, Heidelberg, August 2023. 8

[GHK17] Romain Gay, Dennis Hofheinz, and Lisa Kohl. Kurosawa-desmedt meets tight security. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 133–160. Springer, Heidelberg, August 2017. 26

[GHKW16] Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Tightly CCA-secure encryption without pairings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 1–27. Springer, Heidelberg, May 2016. 26

[GJ18] Kristian Gjøsteen and Tibor Jager. Practical and tightly-secure digital signatures and authenticated key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 95–125. Springer, Heidelberg, August 2018. 5, 7, 32, 33

[GJKW07] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the Diffie-Hellman problems. *Journal of Cryptology*, 20(4):493–514, October 2007. 3, 4

[GKP18] Federico Giacon, Eike Kiltz, and Bertram Poettering. Hybrid encryption in a multi-user setting, revisited. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 159–189. Springer, Heidelberg, March 2018. 7, 27

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. 3

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. 3

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Heidelberg, November 2017. 7, 24, 29, 31

[HJ12] Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 590–607. Springer, Heidelberg, August 2012. 3, 4, 26

[HJK+21] Shuai Han, Tibor Jager, Eike Kiltz, Shengli Liu, Jiaxin Pan, Doreen Riepel, and Sven Schäge. Authenticated key exchange and signatures with tight security in the standard model. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 670–700, Virtual Event, August 2021. Springer, Heidelberg. 2, 3, 5, 7, 32, 33

[HJKS15] Felix Heuer, Tibor Jager, Eike Kiltz, and Sven Schäge. On the selective opening security of practical public-key encryption schemes. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 27–51. Springer, Heidelberg, March / April 2015. 8, 33, 51

[HLG21] Shuai Han, Shengli Liu, and Dawu Gu. Key encapsulation mechanism with tight enhanced security in the multi-user setting: Impossibility result and optimal tightness. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 483–513. Springer, Heidelberg, December 2021. 8

[HLG23] Shuai Han, Shengli Liu, and Dawu Gu. Almost tight multi-user security under adaptive corruptions & leakages in the standard model. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 132–162. Springer, Heidelberg, April 2023. 5, 27

[HLLG19] Shuai Han, Shengli Liu, Lin Lyu, and Dawu Gu. Tight leakage-resilient CCA-security from quasi-adaptive hash proof system. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 417–447. Springer, Heidelberg, August 2019. 26

[HLWG23] Shuai Han, Shengli Liu, Zhedong Wang, and Dawu Gu. Almost tight multi-user security under adaptive corruptions from LWE in the standard model. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 682–715. Springer, Heidelberg, August 2023. 27

[HS16] Goichiro Hanaoka and Jacob C. N. Schuldt. On signatures with tight security in the multi-user setting. In *2016 International Symposium on Information Theory and Its Applications (ISITA)*, pages 91–95, 2016. 3, 4

[HS21] Hans Heum and Martijn Stam. Tightness subtleties for multi-user PKE notions. In Maura B. Paterson, editor, *18th IMA International Conference on Cryptography and Coding*, volume 13129 of *LNCS*, pages 75–104. Springer, Heidelberg, December 2021. 6, 23

[Jae23] Joseph Jaeger. Let attackers program ideal models: Modularity and composability for adaptive compromise. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 101–131. Springer, Heidelberg, April 2023. 27, 32

[JK22] Joseph Jaeger and Akshaya Kumar. Memory-tight multi-challenge security of public-key encryption. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part III*, volume 13793 of *LNCS*, pages 454–484. Springer, Heidelberg, December 2022. 49, 50

[JKRS21] Tibor Jager, Eike Kiltz, Doreen Riepel, and Sven Schäge. Tightly-secure authenticated key exchange, revisited. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 117–146. Springer, Heidelberg, October 2021. 32

[JL17] Simon Josefsson and Ilari Liusvaara. Edwards-curve digital signature algorithm (EdDSA). RFC 8032, Jan. 2017. https://datatracker.ietf.org/doc/html/rfc8032. 4

[JSSOW17] Tibor Jager, Martijn Stam, Ryan Stanley-Oakes, and Bogdan Warinschi. Multi-key authenticated encryption with corruptions: Reductions are lossy. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 409–441. Springer, Heidelberg, November 2017. 2, 8, 25

[KMP16] Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Heidelberg, August 2016. 3, 4

[KPRR23] Eike Kiltz, Jiaxin Pan, Doreen Riepel, and Magnus Ringerud. Multi-user CDH problems and the concrete security of NAXOS and HMQV. In Mike Rosulek, editor, *CT-RSA 2023*, volume 13871 of *LNCS*, pages 645–671. Springer, Heidelberg, April 2023. 28, 29, 32

[Kra05] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer, Heidelberg, August 2005. 7

[Lac18] Marie-Sarah Lacharité. Security of BLS and BGLS signatures in a multi-user setting. *Cryptography and Communications*, 10(1):41–58, 2018. 3, 4

[LJYP14] Benoît Libert, Marc Joye, Moti Yung, and Thomas Peters. Concise multi-challenge CCA-secure encryption and signatures with almost tight security. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 1–21. Springer, Heidelberg, December 2014. 26

[LLGW20] Xiangyu Liu, Shengli Liu, Dawu Gu, and Jian Weng. Two-pass authenticated key exchange with explicit authentication and tight security. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 785–814. Springer, Heidelberg, December 2020. 7, 32

[LLP20] Youngkyung Lee, Dong Hoon Lee, and Jong Hwan Park. Tightly CCA-secure encryption scheme in a multi-user setting with corruptions. *DCC*, 88(11):2433–2452, 2020. 27

[Mic23] Microsoft. Results of major technical investigations for storm-0558 key acquisition. Microsoft Blog, September 2023. https://msrc.microsoft.com/blog/2023/09/results-of-major-technical-investigations-for-storm-0558-key-acquisition/. 2

[MPS20] Andrew Morgan, Rafael Pass, and Elaine Shi. On the adaptive security of MACs and PRFs. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 724–753. Springer, Heidelberg, December 2020. 8

[MS04] Alfred Menezes and Nigel Smart. Security of signature schemes in a multi-user setting. *Designs, Codes and Cryptography*, 33(3):261–274, 2004. 3

[Nat] National Institute for Standards and Technology (NIST). Post-quantum cryptography standardization. https://csrc.nist.gov/projects/post-quantum-cryptography. 31

[Nat23] National Institute of Standards and Technology. Digital Signature Standard (DSS). FIPS PUB 186-5, Feb. 2023. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf. 4

[PR20] Jiaxin Pan and Magnus Ringerud. Signatures with tight multi-user security from search assumptions. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 485–504. Springer, Heidelberg, September 2020. 3, 4

[PW22] Jiaxin Pan and Benedikt Wagner. Lattice-based signatures with tight adaptive corruptions and more. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 347–378. Springer, Heidelberg, March 2022. 5

[PWZ23] Jiaxin Pan, Benedikt Wagner, and Runzhi Zeng. Lattice-based authenticated key exchange with tight security. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 616–647. Springer, Heidelberg, August 2023. 32

[PZ22] Jiaxin Pan and Runzhi Zeng. Compact and tightly selective-opening secure public-key encryption schemes. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part III*, volume 13793 of *LNCS*, pages 363–393. Springer, Heidelberg, December 2022. 33

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991. 2

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. 3

[Whi23] Zack Whittacker. Microsoft lost its keys, and the government got hacked. TechCrunch, July 2023. https://techcrunch.com/2023/07/17/microsoft-lost-keys-government-hacked/. 2

[Zav12] G.M. Zaverucha. Hybrid encryption in the multi-user setting. Cryptology ePrint Archive, Report 2012/159, 2012. https://eprint.iacr.org/2012/159. 27

A Recovering Coron's result on RSA-FDH

We want to use our framework to recover the original result of Coron for RSA-FDH signatures [Cor00]. For this, recall that an RSA instance generator RSAGen is a randomized algorithm that outputs (N, e, d), where N is the product of two distinct primes and e and d are integers such that $ed \equiv 1 \bmod \phi(N)$, where $\phi(\cdot)$ is Euler's phi function.

We describe the signature scheme $\mathsf{RSA\text{-}FDH}$ in Figure 25. We also define an FFS RSA in Figure 26. Note that the induced mu game $\mathbf{G}^{\mathsf{RSA\text{-}mu\text{-}}n}_{\mathsf{RSAGen}}$ is known to be tightly equivalent to the single-user game $\mathbf{G}^{\mathsf{RSA}}_{\mathsf{RSAGen}}$, which is implicitly used in Coron's proof [Cor00]. The following lemma combines this fact with Theorem 4.2, by observing that RSA is local.

Lemma A.1 *Let n, c be integers such that $0 \le c < n$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\mathsf{RSA\text{-}muc\text{-}}(n,c)}_{\mathsf{RSAGen}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\mathsf{RSA}}_{\mathsf{RSAGen}}$ such that*

$$\mathbf{Adv}^{\mathsf{RSA\text{-}muc\text{-}}(n,c)}_{\mathsf{RSAGen}}(\mathcal{A}) \le e(c+1) \cdot \mathbf{Adv}^{\mathsf{RSA}}_{\mathsf{RSAGen}}(\mathcal{B}) \ .$$

The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler $\mathsf{D\text{-}FX}$ and re-randomization of elements.

RSA-FDH.Kg:
1 $(N, e, d) \leftarrow\!\!\$$ RSAGen
2 $(vk, sk) \leftarrow ((N, e), (N, d))$
3 Return (vk, sk)

RSA-FDH.Sign[H](sk, M):
4 $y \leftarrow \mathsf{H}(M)$
5 Return $\sigma \leftarrow y^d \bmod N$

RSA-FDH.Vf[H](vk, M, σ):
6 $y \leftarrow \sigma^e \bmod N$
7 Return $[\![y = \mathsf{H}(M)]\!]$

Figure 25: Signature scheme RSA-FDH for an RSA instance generator RSAGen, where RSA-FDH.ROS is the set of all $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_N^*$.

RSA[RSAGen](gs):
1 $(N, e, d) \leftarrow\!\!\$$ RSAGen
2 Return $(pp, os) \leftarrow ((N, e), \varepsilon)$

RSA[RSAGen](init, $((N, e), \varepsilon)$):
3 St.N $\leftarrow N$; St.e $\leftarrow e$
4 $x \leftarrow\!\!\$ \mathbb{Z}_N^*$; St.x $\leftarrow x$; St.y $\leftarrow x^e \bmod N$
5 Return (y, St)

RSA[RSAGen](corr, ε, St):
6 Return (St.x, St)

RSA[RSAGen](fin, x, St):
7 Return $([\![\text{St.x} = x]\!], \epsilon)$

Game $\mathbf{G}^{\text{RSA-muc-}(n,c)}_{\text{RSAGen}}$

Init:
1 $(N, e, d) \leftarrow\!\!\$$ RSAGen
2 For $i = 1, \ldots, n$ do
3 $\quad x_i \leftarrow\!\!\$ \mathbb{Z}_N^*$, $y_i \leftarrow x_i^e \bmod N$
4 Return $(N, e, y_1, \ldots, y_n)$

Corrupt(i): // $i \in [1..n]$
5 CS $\leftarrow$ CS $\cup \{i\}$
6 Return x_i

Fin(i, x): // $i \in [1..n]$
7 If $i \in$ CS then return false
8 Return $[\![x_i = x]\!]$

Figure 26: **Left:** Formal security specification RSA for an RSA instance generator RSAGen. **Right:** Game $\mathbf{G}^{\text{RSA-muc-}(n,c)}_{\text{RSAGen}}$ describing the execution of RSA[RSAGen] in the muc setting.

We are now ready to state our result for RSA-FDH.

Theorem A.2 *Let* RSA-FDH *be the signature scheme defined in Figure 25. Let q_{ro}, q_s be integers such that $0 \le q_s < q_{ro}$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\text{UF}}_{\text{RSA-FDH}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\text{RSA}}_{\text{RSAGen}}$ such that*

$$\mathbf{Adv}^{\text{UF}}_{\text{RSA-FDH}}(\mathcal{A}) \le e(q_s + 1) \cdot \mathbf{Adv}^{\text{RSA}}_{\text{RSAGen}}(\mathcal{B}) \ ,$$

where q_s is the number of signing queries and q_{ro} be the number of random oracle queries $\mathcal{A}$ makes. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler D-FX *and re-randomization of elements.*

We prove the theorem via the following lemma, showing that RSA-muc tightly implies the security of RSA-FDH. Theorem A.2 then follows by combining Lemmas A.1 and A.3.

Lemma A.3 *Let* RSA-FDH *be the signature scheme defined in Figure 25. Let q_{ro}, q_s be integers such that $0 \le q_s < q_{ro}$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\text{UF}}_{\text{RSA-FDH}}$ that issues at most q_{ro} queries to random oracle* RO *and q_s queries to the signing oracle. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\text{RSA-muc-}(q_{ro},q_s)}_{\text{RSAGen}}$ such that*

$$\mathbf{Adv}^{\text{UF}}_{\text{RSA-FDH}}(\mathcal{A}) \le \mathbf{Adv}^{\text{RSA-muc-}(q_{ro},q_s)}_{\text{RSAGen}}(\mathcal{B}) \ .$$

The running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

Proof of Lemma A.3: We construct adversary $\mathcal{B}$ for game $\mathbf{G}^{\text{RSA-muc-}(q_{ro},q_s)}_{\text{RSAGen}}$ as follows: When $\mathcal{A}$ calls $\mathbf{G}^{\text{UF}}_{\text{RSA-FDH}}$.Init, $\mathcal{B}$ calls its own Init oracle to receive (N, e) and $y_1, \ldots, y_{q_{ro}}$. It sets $vk \leftarrow (N, e)$

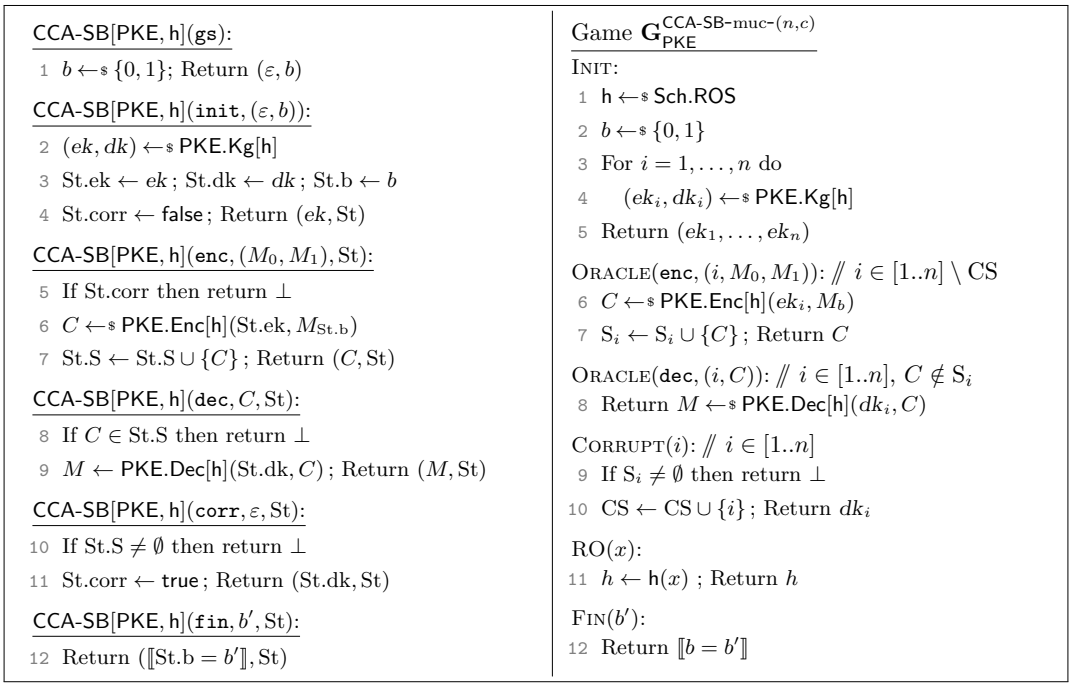

Figure 27: **Left:** FSS CCA-SB[PKE, h] capturing security with a single challenge bit. **Right:** Game $\mathbf{G}^{\text{CCA-SB-muc-}(n,c)}_{\text{PKE}}$ describing the execution of CCA-SB[PKE, h] in the muc setting. The FFS for CPA-security CPA-SB[PKE, h] is obtained by omitting the dec sub-algorithm.

---

and returns it to $\mathcal{A}$. We assume w.l.o.g. that $\mathcal{A}$ makes a query to RO for each message M it queries to the signing oracle and the message M^* for its final forgery. On the i-th query to RO, $\mathcal{B}$ sets the output to y_i. When $\mathcal{A}$ queries the signing oracle for message M such that $\text{RO}(M) = y_i$, $\mathcal{B}$ queries $\text{Corrupt}(i)$ to obtain $x_i = y_i^d \bmod N$ which is a valid signature for M since $y_i = x_i^e \bmod N$. When $\mathcal{A}$ outputs its forgery (M^*, σ^*), $\mathcal{B}$ looks up the index i^* such that $\text{RO}(M^*) = y_{i^*}$ and calls Fin for (i^*, σ^*). Note that if $\mathcal{A}$ is successful, i. e., σ^* is a valid forgery and M^* has not been queried to the signing oracle, then $\mathcal{B}$ wins game $\mathbf{G}^{\text{RSA-muc-}(q_{ro}, q_s)}_{\text{RSAGen}}$ and the statement follows. ∎

B Proof of FO-Transform

Since we start from a PKE scheme, we provide the FSS for single-bit security of PKE in Figure 27.

Proof of Theorem 5.8: We prove the theorem in three steps. This allows us to prove security of the resulting KEM scheme in the sb setting with corruptions. As an intermediate notion we will use one-way security for PKE as defined via the local FFS from Figure 10 in Section 5.1. For a schematic overview we refer to the right side of Figure 14 in Section 5.2.

T-Transform. We start with the PKE scheme $\overline{\text{PKE}}$ constructed using the T-Transform $\mathbf{T}[\text{PKE}, \text{G}]$ as shown in Figure 17, where we model G as a random oracle mapping to randomness space PKE.RS. The following lemma shows OW-PCVA-mu security of $\overline{\text{PKE}}$.

Lemma B.1 *Let* PKE *be γ-spread and* $\overline{\text{PKE}} = \mathbf{T}[\text{PKE}, \text{G}]$ *as described in Figure 17. Let $m \geq 1$ be an integer. Let $\mathcal{A}$ be an adversary for game* $\mathbf{G}^{\text{OW-PCVA-mu-}m}_{\overline{\text{PKE}}}$ *that issues q_e queries to* Oracle(enc, $\cdot$),

q_p queries to $\textsc{Oracle}(\mathtt{pco}, \cdot)$*, q_v queries to* $\textsc{Oracle}(\mathtt{cvo}, \cdot)$ *and q_{ro} queries to random oracle* RO*. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\mathsf{CPA\text{-}SB\text{-}mu}\text{-}m}_{\overline{\mathsf{PKE}}}$ such that*

$$\mathbf{Adv}^{\mathsf{OW\text{-}PCVA\text{-}mu}\text{-}m}_{\overline{\mathsf{PKE}}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\mathsf{CPA\text{-}SB\text{-}mu}\text{-}m}_{\overline{\mathsf{PKE}}}(\mathcal{B}) + q_v \cdot 2^{-\gamma} + \frac{nq_e(q_{ro} + q_p) + 1}{|\overline{\mathsf{PKE}}.\mathsf{MS}|} \ .$$

Adversary $\mathcal{B}$ makes the same number of encryption queries as $\mathcal{A}$ makes. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for maintaining lists of encryption and random oracle queries.

Note that this lemma does not consider corruptions and is basically the same as the one proved by Jaeger and Kumar in [JK22]. However, they apply small changes to the scheme in order to additionally achieve memory-tightness. Further, Duman et al. [DHK+21] prove multi-user security (without corruptions) of the full FO transform. Since our proof is very similar to those, we only give a sketch here.

Proof: Starting with the OW-PCVA-mu game, we first change the way how $\textsc{Oracle}(\mathtt{pco}, \cdot)$ and $\textsc{Oracle}(\mathtt{cvo}, \cdot)$ work. Instead of checking whether C decrypts to M, $\textsc{Oracle}(\mathtt{pco}, \cdot)$ only checks whether M encrypts to C. Assuming perfect correctness, this does not make a difference. $\textsc{Oracle}(\mathtt{cvo}, \cdot)$ performs its check using the random oracle queries and rejects if there does not exist a query that explains C. This change can be bounded by the γ-spreadness, i. e., $q_v \cdot 2^{-\gamma}$.

The next modification is that the game sets a flag bad when the random oracle is queried on any message output by the encryption oracle. In order to bound the probability of bad being set, we construct a reduction $\mathcal{B}$ for game $\mathbf{G}^{\mathsf{CPA\text{-}SB\text{-}mu}\text{-}m}_{\overline{\mathsf{PKE}}}$. For each encryption query $\mathcal{B}$ picks two random messages (M_0, M_1) and forwards them to its own encryption oracle to receive an encryption of M_b. If the random oracle is queried on one of these messages, $\mathcal{B}$ outputs the corresponding bit. With probability $1/|\mathsf{PKE.MS}|$, $\mathcal{A}$ queries a message M_{1-b} to either the random oracle or $\textsc{Oracle}(\mathtt{pco}, \cdot)$. Union bound over the number of encryption queries and users gives us $nq_e(q_{ro} + q_p)/|\mathsf{PKE.MS}|$. Otherwise, if $\mathcal{A}$ queries M_b, $\mathcal{B}$ wins.

We construct another reduction $\mathcal{B}'$ to bound the probability that the final game returns true. Note that the game aborts when the random oracle is queried on any M that is used for encryption queries. $\mathcal{B}'$ is similar to $\mathcal{B}$. It chooses two random messages and forwards them to its encryption oracle. When $\mathcal{A}$ queries finalize with a message M, $\mathcal{B}'$ checks whether this message equals one of the messages and outputs the respective bit. Note that for this the message space needs to be sufficiently large. Folding the two adversaries into a single adversary and collecting the probabilities gives us the stated bound. ▌

From mu to muc. Luckily, the OW-PCVA FSS is local and we can apply Theorem 5.3 from Section 5.1 to go from multi-user security without corruptions to multi-user security with corruptions.

U-Transform. Finally, we prove security of the KEM scheme KEM constructed using the U-Transform $\mathbf{U}[\mathsf{PKE}, \mathsf{H}]$ as shown in Figure 17, where we now model H as a random oracle mapping to $\{0,1\}^{\mathsf{KEM.kl}}$.

Lemma B.2 *Let* $\mathsf{KEM} = \mathbf{U}[\mathsf{PKE}, \mathsf{H}]$ *be as defined in Figure 17. Let n, c be integers such that $0 \leq c < n$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\mathsf{CCA\text{-}SB\text{-}muc}\text{-}(n,c)}_{\mathsf{KEM}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\mathsf{OW\text{-}PCVA\text{-}muc}\text{-}(n,c)}_{\mathsf{PKE}}$ such that*

$$\mathbf{Adv}^{\mathsf{CCA\text{-}SB\text{-}muc}\text{-}(n,c)}_{\mathsf{KEM}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{OW\text{-}PCVA\text{-}muc}\text{-}(n,c)}_{\mathsf{PKE}}(\mathcal{B}) \ .$$

Adversary $\mathcal{B}$ makes the same number of corruption and encryption queries as $\mathcal{A}$ makes. Let q_d and q_{ro} be the number of decryption and random oracle queries $\mathcal{A}$ makes. Then $\mathcal{B}$ makes at most q_d queries to $\textsc{Oracle}(\texttt{cvo}, \cdot)$ *and at most q_{ro} queries to* $\textsc{Oracle}(\texttt{pco}, \cdot)$*. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for maintaining lists of encryption, decryption and random oracle queries.*

Jaeger and Kumar in [JK22] prove this statement without corruptions. Since we already moved to a setting with corruptions, this means that both games now allow corruptions such that the reduction we will construct can simply forward those queries.

Proof: The proof is very similar to that of Theorem 5.7, where here $\textsc{Oracle}(\texttt{pco}, \cdot)$ serves the purpose of the $\textsc{Oracle}(\texttt{ddh}, \cdot)$ oracle. The adversary $\mathcal{B}$ which we will construct is similar to that in Figure 16, thus we will only describe it in words. It runs adversary $\mathcal{A}$ and simulates game $\mathbf{G}_{\mathsf{KEM}}^{\mathsf{CCA\text{-}SB\text{-}muc\text{-}}(n,c)}$ as follows.

To initialize the game, $\mathcal{B}$ queries its own $\textsc{Init}$ oracle and receives a list of n encryption keys. When $\mathcal{A}$ queries $\textsc{Oracle}(\texttt{enc}, i)$, $\mathcal{B}$ first checks whether i was corrupted in which case this query is not allowed. Otherwise it proceeds and queries its own oracle $\textsc{Oracle}(\texttt{enc}, i)$ to receive a ciphertext C. Since its task is to compute the underlying message for one of the ciphertexts it receives, it cannot compute the key K honestly. Instead, it chooses K uniformly at random from $\{0,1\}^{\mathsf{KEM.kl}}$ and stores (C, K) in a set S_i. Note that as long as RO is not queried on the respective input, this simulation is perfect.

For queries to $\textsc{Oracle}(\texttt{dec}, (i, C))$, $\mathcal{B}$ keeps an additional set DS which stores tuples of the form (ek_i, C, K). If a query is repeated, the same key is given as output. If a new ciphertext is queried and it is not a valid ciphertext which can be checked using $\mathcal{B}$'s oracle $\textsc{Oracle}(\texttt{cvo}, \cdot)$, $\mathcal{B}$ outputs $\perp$. Otherwise it chooses a random key K and stores it in DS. In order to be consistent with random oracle queries, we will also add entries there, which we will describe below. Corruption queries, if allowed, are simply forwarded.

It remains to describe the simulation of RO. Queries are stored in a list HS and repeating queries are answered consistently. If ek belongs to one of the users in the game, $\mathcal{B}$ queries $\textsc{Oracle}(\texttt{pco}, \cdot)$ on (M, C) provided by $\mathcal{A}$. If C is a valid ciphertext for M and C was previously output by a query to $\textsc{Oracle}(\texttt{enc}, \cdot)$, then (since corruption and encryption queries are not allowed for the same user) $\mathcal{B}$ stops the execution of $\mathcal{A}$ and queries $\textsc{Fin}$. If C was previously queried to $\textsc{Oracle}(\texttt{dec}, \cdot)$, $\mathcal{B}$ patches the random oracle using list DS. Otherwise, it adds a new entry. Note that until now we were looking at the case that C is a valid encryption of M. If this is not the case, we only add an entry to HS and output K.

In order to win, $\mathcal{A}$ must query the random oracle on a valid pair of (M, C), otherwise K is uniformly random in any case. Further, if $\mathcal{A}$ issues such a query, $\mathcal{B}$ can win game $\mathbf{G}_{\mathsf{PKE}}^{\mathsf{OW\text{-}PCVA\text{-}muc\text{-}}(n,c)}$, which concludes the proof. ∎

The statement in Theorem 5.8 now follows from Lemmas B.1 and B.2 and Theorem 5.3. ∎

C Definitions and proofs for selective opening security

In this section we want to provide formal definitions and proofs for the application of our framework to selective opening security. We will use the standard notion of simulation-based security

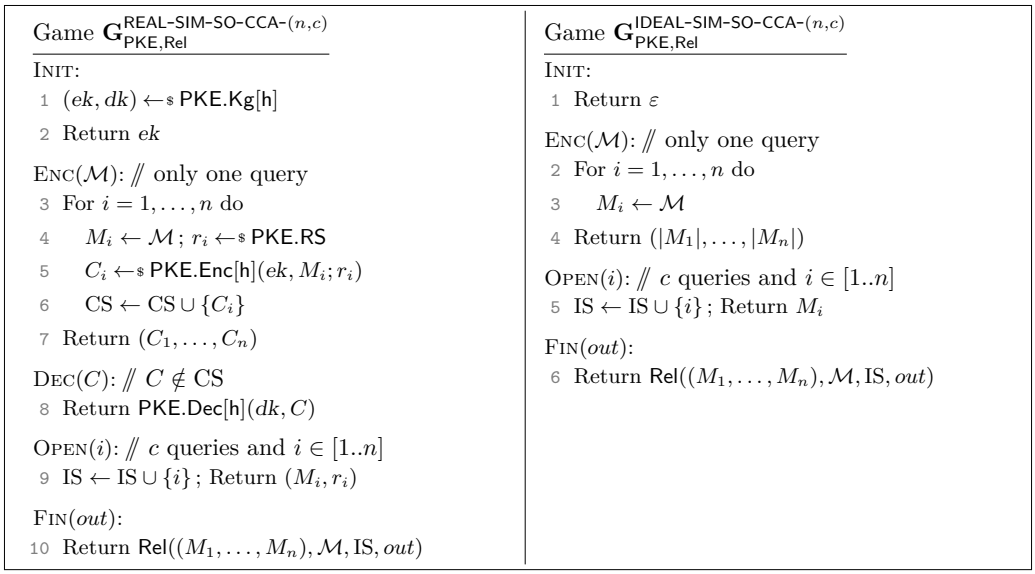

Figure 28: Games $\mathbf{G}^{\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA\text{-}}(n,c)}_{\mathsf{PKE},\mathsf{Rel}}$ and $\mathbf{G}^{\mathsf{IDEAL\text{-}SIM\text{-}SO\text{-}CCA\text{-}}(n,c)}_{\mathsf{PKE},\mathsf{Rel}}$ for selective opening security.

under selective opening attacks. We then show that we can improve the tightness of the practical encryption schemes based on Diffie-Hellman and RSA considered in [HJKS15] in the sense that the security loss is linear in the number of openings rather than linear in the number of ciphertexts.

Simulation-based security for selective openings. We recall the definition of SIM-SO-CCA security for PKE schemes from [HJKS15]. We define two games, a *real* game $\mathbf{G}^{\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA\text{-}}(n,c)}_{\mathsf{PKE},\mathsf{Rel}}$ which runs an adversary $\mathcal{A}$ and an *ideal* game $\mathbf{G}^{\mathsf{IDEAL\text{-}SIM\text{-}SO\text{-}CCA\text{-}}(n,c)}_{\mathsf{PKE},\mathsf{Rel}}$ which runs a simulator $\mathcal{S}$. The games are presented in Figure 28, where $\mathcal{M}$ is a distribution over the message space $\mathsf{PKE.MS}$ specified by $\mathcal{A}$ from which n messages will be drawn and Rel is a relation. We define the advantage function as

$$\mathbf{Adv}^{\mathsf{SIM\text{-}SO\text{-}CCA\text{-}}(n,c)}_{\mathsf{PKE},\mathsf{Rel}}(\mathcal{A}) := \left|\Pr[\mathbf{G}^{\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA\text{-}}(n,c)}_{\mathsf{PKE},\mathsf{Rel}}(\mathcal{A})] - \max_{\mathcal{S}} \Pr[\mathbf{G}^{\mathsf{IDEAL\text{-}SIM\text{-}SO\text{-}CCA\text{-}}(n,c)}_{\mathsf{PKE},\mathsf{Rel}}(\mathcal{S})]\right| .$$

Diffie-Hellman and RSA-based PKE. We present schemes DH-PKE and RSA-PKE in Figure 29. They are instantiations of the KEM-based scheme from [HJKS15]. In contrast to [HJKS15], we prove their security with a tightness loss in the number of Open queries instead of the number of ciphertexts n. This is captured in the following theorems.

Theorem C.1 *Let* DH-PKE *be the PKE scheme as defined in Figure 29. Let n, c be integers such that $0 \le c < n$ and* Rel *a relation. For any adversary $\mathcal{A}$ in game $\mathbf{G}^{\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA\text{-}}(n,c)}_{\mathsf{DH\text{-}PKE},\mathsf{Rel}}$, there exists an adversary $\mathcal{B}$ against $\mathbf{G}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}$ such that*

$$\mathbf{Adv}^{\mathsf{SIM\text{-}SO\text{-}CCA\text{-}}(n,c)}_{\mathsf{DH\text{-}PKE},\mathsf{Rel}}(\mathcal{A}) \le e(c+1) \cdot \mathbf{Adv}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}(\mathcal{B}) + \frac{nq_d}{p} + \frac{q_{ro}}{2^{\mathsf{kl}}} + \frac{q_d}{2^{\mathsf{tl}}} ,$$

where $\mathsf{kl} = \mathsf{DH\text{-}PKE.kl}$ *and* $\mathsf{tl} = \mathsf{DH\text{-}PKE.tl}$. *Let q_{ro} be the number of random oracle queries $\mathcal{A}$ makes. Then $\mathcal{B}$ makes at most q_{ro} queries to oracle* Ddh. *The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler* D-FX *and re-randomization of group elements.*

Theorem C.2 *Let* RSA-PKE *be the PKE scheme as defined in Figure 29. Let n, c be integers such that $0 \le c < n$ and* Rel *a relation. For any adversary $\mathcal{A}$ in game $\mathbf{G}^{\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA\text{-}}(n,c)}_{\mathsf{RSA\text{-}PKE},\mathsf{Rel}}$, there*

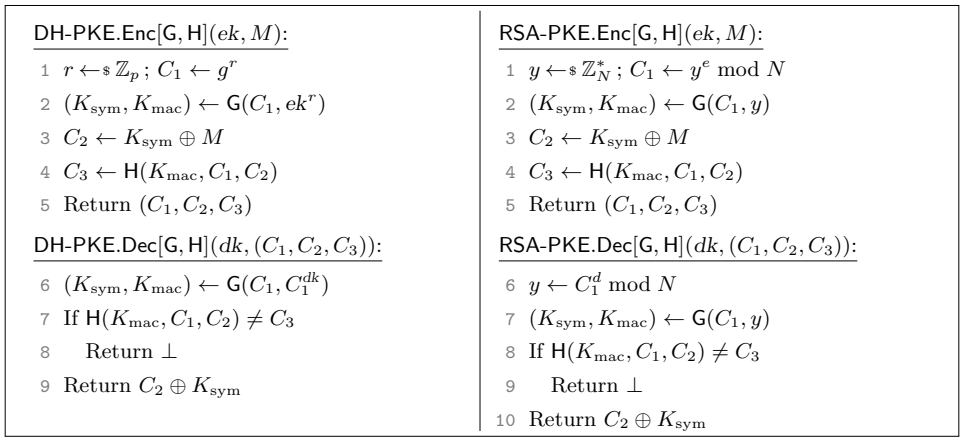

Figure 29: **Left:** PKE scheme DH-PKE for a group $(\mathbb{G}, p, g)$, where DH-PKE.Kg = HEG.Kg. **Right:** PKE scheme RSA-PKE for RSAGen, where RSA-PKE.Kg = RSA-FDH.Kg. For both $\mathsf{PKE} \in \{\mathsf{DH\text{-}PKE}, \mathsf{RSA\text{-}PKE}\}$, we have PKE.ROS contains all $\mathsf{G} : \{0,1\}^* \to \mathsf{PKE.MS} \times \{0,1\}^{\mathsf{PKE.kl}}$ and $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\mathsf{PKE.tl}}$ for integers kl, tl.

---

exists an adversary $\mathcal{B}$ *against* $\mathbf{G}^{\mathsf{RSA}}_{\mathsf{RSAGen}}$ *such that*

$$\mathbf{Adv}^{\mathsf{SIM\text{-}SO\text{-}CCA}\text{-}(n,c)}_{\mathsf{RSA\text{-}PKE,Rel}}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}^{\mathsf{RSA}}_{\mathsf{RSAGen}}(\mathcal{B}) + \frac{nq_d}{|\mathbb{Z}^*_N|} + \frac{q_{ro}}{2^{\mathsf{kl}}} + \frac{q_d}{2^{\mathsf{tl}}} \ ,$$

where $\mathsf{kl} = \mathsf{RSA\text{-}PKE.kl}$ *and* $\mathsf{tl} = \mathsf{RSA\text{-}PKE.tl}$. *The running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$ *plus the time for an execution of the sampler* D-FX *and re-randomization of elements.*

We start with the proof of Theorem C.1 and give a proof sketch of Theorem C.2 noting the differences.

Proof of Theorem C.1: We prove the theorem via the sequence of games G_0 to G_5 in Figure 30, where ROG and ROH use lazy sampling to simulate random oracles for G and H, respectively. G_0 is the same as $\mathbf{G}^{\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA}\text{-}(n,c)}_{\mathsf{DH\text{-}PKE,Rel}}$, except that we draw r_i and compute $C_{1,i}$ and $(K_{\mathrm{sym},i}, K_{\mathrm{mac},i})$ already during Init. We also introduce a boolean flag queried-enc to record whether the encryption oracle has been queried. We will use it in a later game hop. Since these are only conceptual changes, we have

$$\Pr[\mathrm{G}_0(\mathcal{A})] = \Pr[\mathbf{G}^{\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA}\text{-}(n,c)}_{\mathsf{DH\text{-}PKE,Rel}}(\mathcal{A})] \ .$$

In G_1, we change the way decryption queries are handled. Instead of calling ROG directly, we will use lists GS and DS, where the latter stores entries which have not been queried to ROG by $\mathcal{A}$. More specifically, whenever $\mathcal{A}$ queries Dec, then the game checks whether there exist $K_{\mathrm{sym}}, K_{\mathrm{mac}}$ in either GS or DS and if so, uses them for decryption. If this is not the case, fresh $K_{\mathrm{sym}}, K_{\mathrm{mac}}$ are chosen and added to DS together with C_1. Then, these are used to perform the decryption. We also adapt the behavior of ROG accordingly. Namely, we check list DS and if Z is the correct value, ROG returns $(K_{\mathrm{sym}}, K_{\mathrm{mac}})$ which were chosen during decryption. If a new query to ROG happens and $Z = C_1^{dk}$, the game adds this entry to DS as well in order to simulate future decryption queries consistently. Note that G_1 behaves exactly as G_0, thus $\Pr[\mathrm{G}_1(\mathcal{A})] = \Pr[\mathrm{G}_0(\mathcal{A})]$.

In G_2, we change decryption again. G_2 sets bad_1 if $\mathcal{A}$ queries the decryption oracle on a value $C_{1,i}$ before the encryption oracle is even queried. We have

$$|\Pr[\mathrm{G}_2(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})]| \leq \Pr[\mathrm{G}_2(\mathcal{A}) \text{ sets } \mathsf{bad}_1] \ ,$$

$\underline{\mathrm{G}_0\text{-}\mathrm{G}_6}$

INIT:

1 $dk \leftarrow\!\!\$\ \mathbb{Z}_p$; $ek \leftarrow g^{dk}$

2 For $i = 1, \ldots, n$ do

3 $\quad r_i \leftarrow\!\!\$\ \mathbb{Z}_p$; $C_{1,i} \leftarrow g^{r_i}$

4 $\quad (K_{\mathrm{sym},i}, K_{\mathrm{mac},i}) \leftarrow \mathrm{ROG}(C_{1,i}, ek^{r_i})$ // G_0-G_5

5 $\quad (C_{2,i}, K_{\mathrm{mac},i}) \leftarrow\!\!\$\ \{0,1\}^{|\mathsf{DH\text{-}PKE.MS}|} \times \{0,1\}^{\mathsf{DH\text{-}PKE.kl}}$ // G_6

6 $\quad C_{3,i} \leftarrow \mathrm{ROH}(K_{\mathrm{mac},i}, C_{2,i}, C_{3,i})$ // G_6

7 Return ek

ENC($\mathcal{M}$): // only one query

8 $\mathsf{queried\text{-}enc} \leftarrow \mathsf{true}$

9 For $i = 1, \ldots, n$ do

10 $\quad M_i \leftarrow \mathcal{M}$

11 $\quad C_{2,i} \leftarrow K_{\mathrm{sym},i} \oplus M_i$; $C_{3,i} \leftarrow \mathrm{ROH}(K_{\mathrm{mac},i}, C_{2,i}, C_{3,i})$ // G_0-G_5

12 $\quad C_i \leftarrow (C_{1,i}, C_{2,i}, C_{3,i})$; $\mathrm{CS} \leftarrow \mathrm{CS} \cup \{C_i\}$

13 Return $(C_1, \ldots, C_n)$

DEC(C_1, C_2, C_3): // $(C_1, C_2, C_3) \notin \mathrm{CS}$

14 If $\exists i \in [n]$ s. t. $C_1 = C_{1,i}$: // G_2-G_6

15 $\quad$ If $\mathsf{queried\text{-}enc} = \mathsf{false}$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$; Abort // G_2-G_6

16 $\quad$ If $C_2 \neq C_{2,i} \wedge (K_{\mathrm{mac},i}, C_1, C_2, C_3) \notin \mathrm{HS}$ then return $\perp$ // G_3-G_6

17 $\quad$ If $C_2 \neq C_{2,i}$ then return $\perp$ // G_5-G_6

18 $(K_{\mathrm{sym}}, K_{\mathrm{mac}}) \leftarrow \mathrm{ROG}(C_1, C_1^{dk})$ // G_0

19 If $\exists K, K'$ s. t. $(C_1, K, K') \in \mathrm{DS}$ // G_1-G_6

20 $\quad (K_{\mathrm{sym}}, K_{\mathrm{mac}}) \leftarrow (K, K')$ // G_1-G_6

21 Else // G_1-G_6

22 $\quad (K_{\mathrm{sym}}, K_{\mathrm{mac}}) \leftarrow\!\!\$\ \{0,1\}^{|\mathsf{DH\text{-}PKE.MS}|} \times \{0,1\}^{\mathsf{DH\text{-}PKE.kl}}$ // G_1-G_6

23 $\quad \mathrm{DS} \leftarrow \mathrm{DS} \cup \{(C_1, K_{\mathrm{sym}}, K_{\mathrm{mac}})\}$ // G_1-G_6

24 If $\mathrm{ROH}(K_{\mathrm{mac}}, C_1, C_2) \neq C_3$ then return $\perp$

25 Else return $C_2 \oplus K_{\mathrm{sym}}$

OPEN(i): // only after ENC and $i \in [1..n]$

26 $\mathrm{IS} \leftarrow \mathrm{IS} \cup \{i\}$

27 $\mathrm{GS} \leftarrow \mathrm{GS} \cup \{(C_{1,i}, ek^{r_i}, C_{2,i} \oplus M_i, K_{\mathrm{mac},i})\}$ // G_6

28 Return (M_i, r_i)

ROG(C_1, Z):

29 If $\exists i \in [n] \setminus \mathrm{IS}$ s. t. $C_1 = C_{1,i}$ and $Z = C_1^{dk}$ then $\mathsf{bad}_2 \leftarrow \mathsf{false}$; Abort // G_4-G_6

30 If $\exists K_{\mathrm{sym}}, K_{\mathrm{mac}}$ s. t. $(C_1, Z, K_{\mathrm{sym}}, K_{\mathrm{mac}}) \in \mathrm{GS}$ then return $(K_{\mathrm{sym}}, K_{\mathrm{mac}})$

31 If $\exists K_{\mathrm{sym}}, K_{\mathrm{mac}}$ s. t. $(C_1, K_{\mathrm{sym}}, K_{\mathrm{mac}}) \in \mathrm{DS}$ and $Z = C_1^{dk}$: // G_1-G_6

32 $\quad$ Return $(K_{\mathrm{sym}}, K_{\mathrm{mac}})$ // G_1-G_6

33 $(K_{\mathrm{sym}}, K_{\mathrm{mac}}) \leftarrow\!\!\$\ \{0,1\}^{|\mathsf{DH\text{-}PKE.MS}|} \times \{0,1\}^{\mathsf{DH\text{-}PKE.kl}}$

34 If $Z = C_1^{dk}$ then $\mathrm{DS} \leftarrow \mathrm{DS} \cup \{(C_1, K_{\mathrm{sym}}, K_{\mathrm{mac}})\}$ // G_1-G_6

35 $\mathrm{GS} \leftarrow \mathrm{GS} \cup \{(C_1, Z, K_{\mathrm{sym}}, K_{\mathrm{mac}})\}$; Return $(K_{\mathrm{sym}}, K_{\mathrm{mac}})$

ROH($K_{\mathrm{mac}}, C_1, C_2$):

36 If $\exists C_3$ s. t. $(K_{\mathrm{mac}}, C_1, C_2, C_3) \in \mathrm{HS}$ then return C_3

37 $C_3 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{DH\text{-}PKE.tl}}$; $\mathrm{HS} \leftarrow \mathrm{HS} \cup \{(K_{\mathrm{mac}}, C_1, C_2, C_3)\}$; Return C_3

FIN(out):

38 Return $\mathsf{Rel}((M_1, \ldots, M_n), \mathcal{M}, \mathrm{IS}, out)$

Figure 30: Games G_0 to G_6 for the proof of Theorem C.1.

since the games are identical until bad_1 is set to true and we can use Theorem 2.1. As long as $\mathcal{A}$ has not queried the encryption oracle, the values $C_{1,i}$ are hidden from $\mathcal{A}$ so that we can bound $\Pr[\mathrm{G}_2(\mathcal{A}) \text{ sets } \mathsf{bad}_1] \leq (nq_d)/p$.

In G_3, the decryption oracle rejects all ciphertexts, where C_1 equals that of a challenge $C_{1,i}$ and C_2 is new, but there exists no entry $(K_{\text{mac}}, C_1, C_2, C_3)$ in the list HS for random oracle ROH. We have

$$|\Pr[G_3(\mathcal{A})] - \Pr[G_2(\mathcal{A})]| \leq \frac{q_d}{2^{\mathsf{tl}}} .$$

For this note that if there has not been a query $(K_{\text{mac}}, C_1, C_2)$ to ROH at all, C_3 can only be valid with probability $2^{-\mathsf{tl}}$. If $(K_{\text{mac}}, C_1, C_2)$ has been queried, but C_3 stored in HS is different to C_3 queried to DEC, then the output would have been $\perp$ anyway.

In G_4, we add a flag bad_2 if ROG is queried on a challenge ciphertext $C_{1,i}$ and Z such that i has not been queried to OPEN and $Z = C_{1,i}^{dk}$. We have

$$|\Pr[G_4(\mathcal{A})] - \Pr[G_3(\mathcal{A})]| \leq \Pr[G_4(\mathcal{A}) \text{ sets } \mathsf{bad}_2] .$$

We will bound the event directly by giving an adversary for $\mathbf{G}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}$ that uses the subset sampler. An alternative approach would be to use another variant of the asymmetric St-CDH assumption which additionally provides ORACLE($\mathsf{ddh}, \cdot$) for elements from the global parameters, which would require to extend our main theorem. The analysis below is however very close to that of Theorem 4.1.

We now describe how adversary $\mathcal{B}$ for $\mathbf{G}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}$ works. When $\mathcal{A}$ queries INIT, $\mathcal{B}$ also queries its own INIT oracle to obtain a challenge (X, Y). It will assign $ek \leftarrow X$ and runs the sampler to obtain an n-bit string s. If $s[i] = 0$, then it sets $C_{1,i} \leftarrow Y \cdot g^{\tilde{r}_i}$ for $\tilde{r}_i \leftarrow\!\!\$\, \mathbb{Z}_p$. Otherwise, it chooses $r_i \leftarrow\!\!\$\, \mathbb{Z}_p$ and sets $C_{1,i} \leftarrow g^{r_i}$. For all i such that $s[i] = 1$, instead of querying ROG directly, $\mathcal{B}$ chooses $(K_{\text{sym},i}, K_{\text{mac},i})$ at random. It returns ek to $\mathcal{A}$. Given the preparation in INIT, $\mathcal{B}$ can now answer queries to ENC, DEC, ROH as described in G_4 of Figure 30. When $\mathcal{A}$ queries OPEN(i), $\mathcal{B}$ checks whether $s[i] = 1$ and in this case simply returns M_i and r_i. If $s[i] = 0$, $\mathcal{B}$ does not know r_i and aborts. Finally, we describe how queries (C_1, Z) to ROG are handled. First, if C_1 corresponds to a challenge ciphertext $C_{1,i}$ such that i has not been opened, $\mathcal{B}$ asks DDH(C_1, Z). If the output is true and $s[i] = 0$, then $\mathcal{B}$ calls FIN(Z') for $Z' \leftarrow Z \cdot X^{-\tilde{r}_i}$. If $s[i] = 1$, $\mathcal{B}$ aborts. If C_1 is new, $\mathcal{B}$ also uses DDH to check whether the query needs to be added to set DS in order to consistently answer decryption queries.

We need to bound the probability that $\mathcal{B}$ is successful and does not abort. We do so using the argument of Theorem 4.1 and applying the bound for the subset sampler from Theorem 3.3. Thus, we can bound the probability for bad_2 by

$$\Pr[G_4(\mathcal{A}) \text{ sets } \mathsf{bad}_2] \leq e(c+1) \cdot \mathbf{Adv}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}(\mathcal{B}) .$$

In G_5, the decryption oracle rejects all ciphertexts where $C_{1,i}$ was part of a challenge, that is we do not have to check list HS anymore. We claim that

$$|\Pr[G_5(\mathcal{A})] - \Pr[G_4(\mathcal{A})]| \leq \frac{q_{ro}}{2^{\mathsf{kl}}} , \tag{39}$$

which follows from observing that $K_{\text{mac},i}$ is determined by the output of ROG but the game aborts when ROG is queried on the correct input for $C_{1,i}$, thus $K_{\text{mac},i}$ is hidden from $\mathcal{A}$. In order to produce a valid C_2, C_3, it must query the random oracle on $K_{\text{mac},i}$. Since $C_{1,i}$ is included in the hash, a critical query happens with probability $2^{-\mathsf{kl}}$. Union bound over all queries gives us Eq. (39).

Finally, in G_6, we compute the whole ciphertext at the beginning of the game without querying ROG explicitly and before the message is known. For this, the game simply chooses $C_{2,i}$ and $K_{\text{mac},i}$ uniformly at random. Note that $K_{\text{sym},i}$ will then only be defined when ENC is queried. Once OPEN is queried, the game will add the correct value to list GS. Due to the game aborting

when ROG is queried, this maintains consistency and the adversary's view does not change. We have $\Pr[\mathrm{G}_6(\mathcal{A})] = \Pr[\mathrm{G}_5(\mathcal{A})]$.

We now construct a simulator $\mathcal{S}$ that runs in the ideal game and simulates G_6 for $\mathcal{A}$. We show that

$$\Pr[\mathbf{G}^{\mathsf{IDEAL\text{-}SIM\text{-}SO\text{-}CCA\text{-}}(n,c)}_{\mathsf{PKE},\mathsf{Rel}}(\mathcal{S})] = \Pr[\mathrm{G}_6(\mathcal{A})] \ .$$

First, $\mathcal{S}$ runs $\mathsf{DH\text{-}PKE.Kg}$ and gives ek to $\mathcal{A}$. When $\mathcal{A}$ calls ENC, $\mathcal{S}$ calls its own ENC oracle to obtain the lengths of messages. It then computes the ciphertexts obliviously without knowing M_i as described in G_6. If $\mathcal{A}$ queries OPEN, $\mathcal{S}$ queries OPEN as well to receive M_i. Since it has chosen the randomness r_i, it can now simply output (M_i, r_i) to $\mathcal{A}$. All other oracles can be simulated as described in G_6. When $\mathcal{A}$ calls FIN, $\mathcal{S}$ calls FIN in the $\mathbf{G}^{\mathsf{IDEAL\text{-}SIM\text{-}SO\text{-}CCA\text{-}}(n,c)}_{\mathsf{PKE},\mathsf{Rel}}$ game with the same output. The theorem then follows from collecting the bounds. ▮

We now briefly sketch the differences for the proof of scheme $\mathsf{RSA\text{-}PKE}$.

Proof of Theorem C.2: The proof is very similar to that of Theorem C.1 and we essentially do the same sequence of games. Since the randomness for C_1 is chosen from $\mathbb{Z}^*_N$, we bound the probability that G sets bad_1 by $(nq_d)/|\mathbb{Z}^*_N|$. Further, we bound the event that G sets bad_2 by constructing an adversary for game $\mathbf{G}^{\mathsf{RSA}}_{\mathsf{RSAGen}}$. Here, we we can make use of game $\mathbf{G}^{\mathsf{RSA\text{-}muc\text{-}}(n,c)}_{\mathsf{RSAGen}}$ which we also use for $\mathsf{RSA\text{-}FDH}$ in Appendix A. This simplifies the proof and does not require to use the sampler directly. The difference to the previous proof lies in the fact that we can check whether Z belongs to C_1 by checking whether $Z^e = C_1$. The remaining steps are then similar to the proof of Theorem C.1. ▮