Memory-Tight Multi-challenge Security of Public-Key Encryption

Joseph Jaeger(✉) and Akshaya Kumar

School of Cybersecurity and Privacy Georgia Institute of Technology, Atlanta, GA, USA
{josephjaeger,akshayakumar}@gatech.edu

Abstract. We give the first examples of public-key encryption schemes which can be proven to achieve multi-challenge, multi-user CCA security via reductions that are tight in time, advantage, *and* memory. Our constructions are obtained by applying the KEM-DEM paradigm to variants of Hashed ElGamal and the Fujisaki-Okamoto transformation that are augmented by adding uniformly random strings to their ciphertexts.

The reductions carefully combine recent proof techniques introduced by Bhattacharyya'20 and Ghoshal-Ghosal-Jaeger-Tessaro'22. Our proofs for the augmented ECIES version of Hashed-ElGamal make use of a new computational Diffie-Hellman assumption wherein the adversary is given access to a pairing to a random group, which we believe may be of independent interest.

Keywords: Public-key cryptography · Provably security · Memory-tightness

1 Introduction

Secure deployment of cryptography requires concrete analysis of schemes to understand how the success probabilities of attackers grow with the amount of resources they employ to attack a system. The use of reduction-based cryptography enables such analysis by using an attacker with running time t and success probability ϵ to construct a related adversary with running time t' and success probability ϵ' against a computational problem whose security is better understood. A gold standard for concrete security reductions are tight reductions for which $t' \approx t$ and $\epsilon' \approx \epsilon$. We refer to such a reduction as *TA-tight* (time-advantage-tight) to distinguish it from other notions of tightness.

Auerbach, Cash, Fersch, and Kiltz [3] argued that the memory usage of an attacker can be crucial in determining its likelihood of success. This kicked of a line of works [7,9–12,17–19,24,28,30,31] on *memory-aware* cryptography which accounts for the memory usage of attackers in security analyses. Auerbach, et al. focused in particular on incorporating memory considerations into the study of reductions. We refer to a reduction as TAM-tight if it is TA-tight and additionally $s \approx s'$ where these variables, respectively, denote the amount of memory used by the original adversary and the reduction adversary.

In this work, we construct the first public-key encryption schemes with TAM-tight proofs of multi-challenge (and multi-user) chosen-ciphertext attack (CCA) security. Our schemes are based on variants of the Hashed ElGamal and Fujisaki-Okamoto transformation key encapsulation mechanisms. These variants augment ciphertexts with random strings that are included in hash function calls.

Multi-challenge setting. As mentioned, our focus in this work is on multi-challenge and multi-user security. This is simply motivated by the fact that encryption schemes get deployed across many different users each of whom will encrypt many messages, so it is important to understand how the security of a scheme degrades as the number of encryptions increase. In particular, the goal of tight proofs is to show that security does not meaningfully degrade. Multiple papers [16,22,25] have looked at this in the non-memory-aware setting, providing schemes with TA-tight proofs of security. However, extending any of these proofs to the memory-aware setting is quite difficult.

Prior works on memory-tight CCA secure encryption have identified a primary difficulty in the multi-challenge setting which lies in how the decryption oracle handles challenge ciphertexts. Simply decrypting a challenge ciphertext would lead to trivial attacks against any scheme, so instead the decryption oracle has to recognize these ciphertexts and respond to them in a special manner.[1] This makes writing memory-tight security proofs difficult because the reduction adversary must emulate this differing behavior on decryption queries for challenge or non-challenge ciphertexts, but it is unclear how to go about identifying which are challenge ciphertexts other than remembering and checking against all ciphertexts that were previously returned to encryption queries. In the single-challenge setting is a non-issue, because storing the single challenge ciphertext requires minimal memory.

Memory-tightness of Hashed ElGamal. In recent years, several papers have discussed the challenge of providing memory-tight security proofs for Hashed ElGamal. Auerbach, et al. [3] gave it at as an example of a proof they considered the memory complexity of, but were unable to improve. Follow-up work by Bhattacharyya [7] and Ghoshal and Tessaro [19] analyzed this further, giving what might seem at first to be contradictory results. Bhattacharyya gave a memory-tight proof for Hashed ElGamal in the single-challenge setting while Ghoshal and Tessaro proved a lower-bound establishing that a memory-tight proof for Hashed ElGamal was not possible.

Resolving this contradiction requires more precisely understanding each result. The lower bound applies specifically for reductions to Strong Computational Diffie-Hellman (CDH) security [2] which are "black-box" in several ways, including that they do not depend on the particular group used. Ghoshal and Tessaro note that Bhattacharyya's result (for single-challenge security) avoids the lower bound by not being black-box in this manner; it depends on the group having an

[1] An alternate definitional style would disallow the adversary from querying challenge ciphertexts to its decryption oracle, but prior the works argue this is an inappropriate restriction in the memory-bounded setting [17,18].

Scheme/Transform	Assumption	Result
Hashed ElGamal (aECIES)	Pair CDH	\$CCA KEM
Cramer-Shoup ElGamal (aCS)	Strong CDH	\$CCA KEM
Twin ElGamal (aTWIN)	CDH	\$CCA KEM
T	\$CPA PKE	OW-PCA PKE
aV	OW-PCA PKE	OW-PCVA PKE
$\mathsf{aU}^{\perp}$	OW-PCVA PKE	CCA KEM
KEM/DEM (KD)	\$CCA KEM/SKE	\$CCA PKE

Fig. 1. TAM-tight reductions we provide. Transformations T, aV, and $\mathsf{aU}^{\perp}$ are FO transforms discussed in Sect. 5. Results are multi-user, multi-challenge security.

efficient pairing. However, for efficiency it is preferable to implement schemes using elliptical curves for which efficient pairing are believed not to exist.

Our result for Hashed ElGamal is black-box in the sense of Ghoshal and Tessaro. We avoid the lower-bound without requiring an efficient pairing by introducing and using an assumption (Pair CDH) which is stronger than Strong CDH, but is reasonable to assume holds in typical groups based on elliptic curves.[2] We discuss this assumption in more detail momentarily. Indeed, Ghoshal and Tessaro say in their paper [19, Sec. 3.1, p.42], "it appears much harder to extend our result to different types of oracles than [the Strong CDH oracle], as our proof is tailored at this oracle." Our new security notion gives an example of such an oracle to which their result cannot be extended.

1.1 Our Results

We summarize our results in Fig. 1. Omitted proofs and results are provided in the full version of this paper [23].

Hashed ElGamal. Our first results consider the security of Hashed ElGamal. Following Bhattacharyya, we actually consider two variants which we refer to as the ECIES [1] and the Cramer-Shoup [8] variants. The negative results of Ghoshal and Tessaro apply only to the ECIES variant. In both, the decryption key is a value $x \in \mathbb{Z}_p^*$ and the encapsulation key is $X = \mathbf{g}^x$. Here $\mathbf{g}$ is a generator of a group of prime order p. For encapsulation, one samples a fresh $y \leftarrow\!\!\$\ \mathbb{Z}_p^*$ and returns $\mathbf{g}^y$ as the ciphertext. For ECIES the derived key is $H(X^y)$, while for Cramer-Shoup it is $H(\mathbf{g}^y, X^y)$. Our main results concern "augmented" versions of both of these schemes where the ciphertexts are instead $(a, \mathbf{g}^y)$ where a is a uniformly random bitstring used as an additional input to the hash function.

[2] Technically, the lower bound also does not apply because we are considering an augmented scheme which differs from the one analyzed by Ghoshal and Tessaro. However, the augmentation is not important for this comparison, because in the single-challenge setting where the lower-bound was proven, Pair CDH TAM-tightly implies security of the non-augmented scheme as well.

To understand these results, let us discuss the high level idea of proving security for ECIES. A standard, single-challenge proof would work from the Strong CDH assumption in the random oracle model. In Strong CDH an adversary is given $X = \mathbf{g}^x$, $Y = \mathbf{g}^y$, and an oracle O which on input B, C tells whether $B^x = C$. Its goal is to return X^y. The only way to distinguish $H(X^y)$ from random is to query H on input X^y. So a reduction adversary will give X as the encryption key, Y as the challenge ciphertext, simulate random oracle and decapsulation queries, and checks if any of the random oracle queries are X^y in which case it returns that. The oracle O is used for checking whether random oracle queries are X^y (for a random oracle query Z, one queries $\mathrm{O}(Y, Z)$ to check) and for maintaining consistency between random oracle and decapsulation queries for non-challenge ciphertexts. A decapsulation query for Y and a random oracle query for Y, Z should return the same result if $Y^x = Z$. The reduction can maintain this consistency by remembering all of the queries made to both oracles and then using O to check for this consistency. This is neither time- nor memory-tight.

Bhattacharyya was able to make this TAM-tight by introducing a new technique for this consistency aspect. They simulate the random oracle $H(C)$ by $h(e(\mathbf{g}, C))$ where h is a random function and e is a pairing. Then the output of a non-challenge decapsulation query B can be simulated as $h(e(X, B))$. In our proof we use a similar technique, but replace the requirement for a pairing-friendly group by using a new variant of CDH we will discuss momentarily.

The first step in making the proof work in the multi-challenge setting is to use Diffie-Hellman rerandomization techniques so we can have multiple Diffie-Hellman challenges. We let the u-th user's public key be X^{x_u} and the i-th ciphertext by Y^{y_i}. For memory-tightness, we pick x_u and y_i using a (pseudo-)random function.[3] Then if the adversary makes a random oracle query $H(C)$ where $C = X^{x_u \cdot y \cdot y_u}$, we have $C^{1/(x_u \cdot y_u)} = X^y$. A challenge here is to know which u and i to use for such a random oracle query. A reduction could check each choice of u, i, but this would lose time-tightness. At the same time, for a decapsulation query B we must be able to identify if B was a prior challenge query. Storing all prior challenge queries loses memory-tightness.

Both of these issues are solved by our addition of an auxiliary string a to each ciphertext and hash query. The idea here is based on memory-tightness techniques of GGJT [17], in that we are going to hide the pertinent information we need in a. Rather than sampling a at random, if the i-th challenge query is made to user u then our reduction adversary picks a to be the "encryption" of (u, i). Now on future random oracle and decapsulation queries we can recover u, i by "decrypting" a. This allows us to properly simulate the view of the adversary.

Pair CDH security. As we have been mentioning, we avoid the need for groups with pairing in our result for ECIES by making use of a new computational assumption. This assumption, we refer to as Pair CDH security, extends CDH security by giving the adversary access to an oracle which, on input A and B

[3] In the body, this is separated out as a proof that single-challenge CDH tightly implies multi-challenge CDH.

(with discrete logarithms a and b) computes a and b then returns a random function applied to $a \cdot b$. This acts, in essence, as a pairing from the group under consideration to a randomly chosen group. Our use of this in our security proof for ECIES takes advantage of the fact that (i) the pairing is only needed for the proof, not in the construction itself and (ii) the proof does not require the ability to efficiently perform group operations with the output of the pairing. We think this notion may be of further interest if other proofs can be found where better tightness can be achieved using a pairing only in the reduction.

To justify our new assumption we analyze how it compares to existing assumptions. Pair CDH security is implied by CDH security if the group under consideration has an efficient pairing. This holds because we can emulate the random pairing by first applying the efficient pairing and then applying a random function (which may be pseudorandomly instantiated for efficiency). In turn, Pair CDH implies the Gap CDH assumption because a pairing can be used to check whether given group elements form a Diffie-Hellman triple.

These results do not justify the use of Pair CDH security for typical groups based on elliptic curves which do not have pairings. For this, we turn to non-standard models (i.e. algebraic or generic group models [13,26,29]). In these models, we are able to show that CDH and Pair CDH are equivalent because learning anything from the oracle requires the ability to find non-trivial collisions in the pairing. The ability to find such collisions can in turn be used to solve the discrete logarithm problem.

Fujisaki-Okamoto Transformation. The other KEMs we consider are those derived from the Fujisaki-Okamoto Transformation which starts with a CPA secure public key encryption scheme and applies several random oracle based transformations to construct a CCA secure KEM. Hofheinz, Hövelmanns, and Kiltz [21] gave a nice modular approach for proving the security of several variants of this transformation. Bhattacharyya showed how to make these proofs memory-tight in the single-challenge setting (in some cases requiring one additional intermediate transformation). We extend these to the multi-challenge setting. For the final step of the transform, we need to consider an augmented transform of the existing scheme in which random strings are added to each ciphertext and incorporated into the hash queries. As before, our reduction samples these string as the encryption of the pertinent information it would need to identify challenge ciphertexts and respond to them appropriately.

For these results, we require that the starting CPA scheme have good multi-challenge security. This is a significantly weaker starter point than multi-challenge CCA security because it avoids the issue of having to be able to identify challenge ciphertexts for the decryption oracle.

Lifting to public-key encryption. The approaches described above are for key encapsulation mechanisms. This raises the question of whether these tight reductions can be applied to public-key encryption via the KEM-DEM paradigm. It uses a KEM to generate a new symmetric key for a data encapsulation mechanism to encrypt the actual message with. This was previously looked at by GGJT [17], who gave a TAM-tight proof of security. However, because of their particular motivations, the proof assumed the KEM was constructed from a public-key encryption scheme. We show that (with some modifications) the proof works with generic KEMs as well.

2 Preliminaries

2.1 Notation

We recall basic notion and security definitions we will use in our paper.

Pseudocode. For our proofs, we use the code based framework of [6]. If $\mathcal{A}$ is an algorithm, then $x \leftarrow \mathcal{A}^{\mathrm{O}}(x_1, x_2, ...; r)$ denotes running $\mathcal{A}$ on inputs $x_1, x_2, ...$ with coins r and having access to the set of oracles O to produce output x. We use the notation $\leftarrow\$$ instead of $\leftarrow$ when not explicitly specifying the coins r. If S is a set, $|S|$ denotes its size and $x \leftarrow\$ S$ denotes sampling x uniformly from S. We use the symbol $\perp$ to indicate rejection. When not specified, tables are initialized empty and integers are initialized to 0.

Security notions are defined with games such as the one in Fig. 3. The probability that the game G outputs $\mathtt{true}$ is $\Pr[\mathsf{G}]$. We sometimes use a sequence of "hybrid" games in one figure for our proofs. We use comments of the form $//\mathsf{G}_{[i,j)}$ to indicate that a line of code is included in games G_k for $i \leqslant k < j$. To identify changes made to the k^{th} hybrid, one looks for lines of code commented as $//\mathsf{G}_{[i,k)}$ for code that is no longer included in the k^{th} hybrid and $//\mathsf{G}_{[k,j)}$ for code that is new to the k^{th} hybrid.

Complexity Measures. Following ACFK [3], we measure the local complexities of algorithms and do not include the complexity of oracles that they interact with. We focus on the worst case runtime $\mathbf{Time}(\mathcal{A})$ and memory used for local computation $\mathbf{Mem}(\mathcal{A})$ of any algorithm $\mathcal{A}$.

Functions and ideal models. We define $\mathsf{Fcs}(D, R)$ (resp. $\mathsf{Inj}(D, R)$) to be the set of all functions (resp. injections) mapping from D to R. For $f \in \mathsf{Inj}(D, R)$, we define f^{-1} to be its inverse (with $f^{-1}(y) = \perp$ if y has no preimage). If D_t and R_t are sets for each $t \in T$, then we define $\mathsf{Fcs}(T, D, R)$ (resp. $\mathsf{Inj}(T, D, R)$) to be the set of functions f so that $f(t, \cdot) \in \mathsf{Fcs}(D_t, R_t)$ (resp. $f(t, \cdot) \in \mathsf{Inj}(D_t, R_t)$). We let $f_t(\cdot) = f(t, \cdot)$.

For $f \in \mathsf{Inj}(D, R)$ we let $f^{\pm}$ denote the function defined by $f^{\pm}(+, x) = f(x)$ and $f^{\pm}(-, x) = f^{-1}(x)$. We often write $f(x)$ or $f^{-1}(x)$ in place of $f^{\pm}(+, x)$ or $f^{\pm}(-, x)$. We let $\mathsf{Inj}^{\pm}(D, R) = \{f^{\pm} : f \in \mathsf{Inj}(D, R)\}$ and extend this to define $\mathsf{Inj}^{\pm}(T, D, R)$ analogously.

Ideal models (e.g. the random oracle or ideal cipher model) are captured by having a scheme S specify a set of functions $\mathsf{S.IM}$. Then, at the beginning of a security game for S, a random $\mathcal{H} \in \mathsf{S.IM}$ is sampled. The adversary and some algorithms of the scheme S are then given oracle access to $\mathcal{H}$. The standard model is captured by $\mathsf{S.IM}$ being a singleton set containing the identity function.

If F and G are sets of functions, then we define $(\mathsf{F}, \mathsf{G}) = \mathsf{F} \times \mathsf{G} = \{ f \times g : f \in \mathsf{F}, g \in \mathsf{G} \}$. Here, $f \times g$ is the function defined by $f \times g(0, x) = f(x)$

and $f \times g(1, x) = g(x)$. In the code of an algorithm expecting oracle access to $f \times g \in \mathsf{F} \times \mathsf{G}$, we write $f(x)$ or $g(x)$ with the natural meaning. We extend this notation to more than two sets of functions as well.

Switching lemma. Our proofs make use of the indistinguishability of random functions and injections, as captured by the following standard result.

Lemma 1 (Switching Lemma). *Fix T, D, R and $N = \min_{t \in T} |R_t|$. For any adversary $\mathcal{A}$ making at most q queries, it holds that $|\Pr[\mathcal{A}^f \Rightarrow 1] - \Pr[\mathcal{A}^g \Rightarrow 1]| \leqslant 0.5 \cdot q^2/N$, where the probability is taken over the randomness of $\mathcal{A}$, sampling $f \leftarrow_{\$} \mathsf{Fcs}(T, D, R)$, and sampling $g \leftarrow_{\$} \mathsf{Inj}(T, D, R)$.*

2.2 Memory-Tightness Background

$\mathcal{F}$-oracle adversaries. We adopt GGJT's [17] oracle adversary formulation for our proofs in the memory-aware setting, i.e., we allow reductions to access uniformly random functions or invertible random injections. Our reductions are of the form shown below for some set of functions $\mathcal{F}$ and algorithm $\mathcal{B}$. We call such an adversary $\mathcal{A}$ an $\mathcal{F}$-oracle adversary.

```
Adversary $\mathcal{A}^{\mathrm{O}}(\mathrm{in})$
$f \leftarrow_{\$} \mathcal{F}$
out $\leftarrow_{\$} \mathcal{B}^{\mathrm{O},f}(\mathrm{in})$
Return out
```

The complexity of adversary $\mathcal{A}$ would include the (large) complexity of sampling, storing, and computing f. However, as proposed in [17], we present theorems in terms of the *reduced complexity* of an oracle aided adversary which is defined as $\mathbf{Time}^*(\mathcal{A}) = \mathbf{Time}(\mathcal{B})$ and $\mathbf{Mem}^*(\mathcal{A}) = \mathbf{Mem}(\mathcal{B})$.

We refer readers to Lemma 2 of [17] which bounds how much an adversary may be aided by a random object by replacing it with a pseudorandom version of the object. Pseudorandom injections can typically be instantiated by appropriately chosen encryption schemes.

There is a small issue when pseudorandomly instantiating a random function if the game $\mathcal{A}$ plays is inefficient. This is the case for some of our reduction adversaries playing CDH variants wherein they have access to some inefficient oracle based on the group. Then the pseudorandomness reduction adversary from [17] will be inefficient because it simulates the game that $\mathcal{A}$ is playing. However, we can simply use pseudorandom schemes believed to be secure even against adversaries with access to the inefficient oracle. This seems reasonable as we can choose a pseudorandom scheme which seems unrelated to the group.

Message encoding techniques. The message encoding technique proposed by GGJT in [17] programs randomness that a reduction provides to an adversary in a special way that stores retrievable state information. This is achieved by generating randomness as the output of random injections. The reduction may then invert randomness generated thusly to retrieve state information. For

PKE Syntax	KEM Syntax	SKE Syntax
$(ek, dk) \leftarrow\$ \mathsf{PKE.K}$	$(ek, dk) \leftarrow\$ \mathsf{KEM.K}$	$K \leftarrow\$ \mathsf{SKE.K}$
$c \leftarrow\$ \mathsf{PKE.E}^{\mathcal{H}}(ek, m)$	$(c, K) \leftarrow\$ \mathsf{KEM.E}^{\mathcal{H}}(ek)$	$c \leftarrow\$ \mathsf{SKE.E}^{\mathcal{H}}(K, m)$
$m \leftarrow \mathsf{PKE.D}^{\mathcal{H}}(dk, c)$	$K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk, c)$	$m \leftarrow \mathsf{SKE.D}^{\mathcal{H}}(K, c)$

Fig. 2. Syntax of a public key encryption scheme PKE, key encapsulation mechanism KEM, and symmetric key encryption scheme SKE. The ideal model oracle is $\mathcal{H}$.

example, consider a key encapsulation mechanism that outputs ciphertexts of the form (a, c) where a is uniformly random. Then a reduction can simulate challenge ciphertexts by setting $a = f(i)$ where f is a random injection and i is some pertinent information the reduction would want to know if the adversary later makes oracle queries for the same ciphertext. Then the reduction can recover this information during future queries as $i \leftarrow f^{-1}(a)$.

Map-then-random-function. We describe the main proof technique of Bhattacharyya [7], namely "map-then-rf".[4] This technique allows the reduction to use the composition of an injection and a random function to replace a random function. This relies on the simple fact that if $h \in \mathsf{Inj}(D, S)$, then sampling f according to $f \leftarrow\$ \mathsf{Fcs}(D, R)$ or $g \leftarrow\$ \mathsf{Fcs}(S, R); f \leftarrow g \circ h$ are equivalent, meaning, if g is a random function, and h is any injection, then $f \leftarrow g \circ h$ is a random function. This allows a reduction to compute the output $f(x)$ given $h(x)$, even if it does not know x.

2.3 Public Key Encryption

Syntax. A public key encryption scheme, PKE, specifies three algorithms - the key generation algorithm (PKE.K) that returns a pair of keys (ek, dk) where ek is the encryption key and dk is the corresponding decryption key, the encryption algorithm (PKE.E) that takes the encryption key ek and a message m and returns ciphertext c, and the decryption algorithm PKE.D that takes the decryption key dk and a ciphertext c and returns message m (or the special symbol $\perp$ to indicate rejection). The syntax of these algorithms is given in Fig. 2.

Perfect correctness requires $\mathsf{PKE.D}^{\mathcal{H}}(dk, c) = m$ for all $(ek, dk) \in [\mathsf{PKE.K}]$, all m, all $\mathcal{H} \in \mathsf{PKE.IM}$, and all $c \in [\mathsf{PKE.E}^{\mathcal{H}}(ek, m)]$. The weaker notion of δ-correctness requires that for all (not necessarily efficient) $\mathcal{D}$,

$$\Pr[\mathsf{PKE.D}^{\mathcal{H}}(dk, \mathsf{PKE.E}^{\mathcal{H}}(ek, m)) \neq m \ : \ m \leftarrow\$ \mathcal{D}^{\mathcal{H}}(ek, dk)] \leqslant \delta(q)$$

where q upper bounds the number of $\mathcal{H}$ queries $\mathcal{D}$ makes. The probability is over $(ek, dk) \leftarrow\$ \mathsf{PKE.K}$, $\mathcal{H} \leftarrow\$ \mathsf{PKE.IM}$, and the coins of $\mathcal{D}$ and PKE.E. When not stated otherwise, schemes are assumed to be perfectly correct.

[4] Bhattacharyya actually uses "map-then-prf", as they were not using the oracle adversary formulation.

```
Game $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{PKE},b}(\mathcal{A})$
$\mathcal{H} \leftarrow\!\!\$\ \mathsf{PKE.IM}$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\!\!\$\ \mathsf{PKE.K}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{New},\textsc{Enc}_b,\textsc{Dec},\mathcal{H}}$
Return $b' = 1$

$\textsc{New}(u)$
Return $ek_u$

$\textsc{Enc}_b(u, m)$
$c_1 \leftarrow\!\!\$\ \mathsf{PKE.E}^{\mathcal{H}}(ek_u, m)$
$c_0 \leftarrow\!\!\$\ \mathsf{PKE}.\mathcal{C}(ek_u, |m|)$
$M[u, c_b] \leftarrow m$
Return $c_b$

$\textsc{Dec}(u, c)$
If $M[u, c] \neq \bot$
    Return $M[u, c]$
$m \leftarrow \mathsf{PKE.D}^{\mathcal{H}}(dk, c)$
Return $m$
```

Fig. 3. Game defining mu-\$cca security of PKE.

We define the encryption keyspace as $\mathsf{PKE.Ek} = \{ek : (ek, dk) \in [\mathsf{PKE.K}]\}$ and assume that for each $ek \in \mathsf{PKE.Ek}$ and allowed message length n, there exists a set $\mathsf{PKE}.\mathcal{C}(ek, n)$ such that $\mathsf{PKE.E}^{\mathcal{H}}(ek, m) \in \mathsf{PKE}.\mathcal{C}(ek, |m|)$ always holds. We assume this set is disjoint for distinct message lengths and let $\mathsf{PKE}.\mathcal{C}^{-1}(ek, c)$ return n such that $c \in \mathsf{PKE}.\mathcal{C}(ek, n)$. We let $\mathsf{PKE}.\mathcal{R}$ denote the set from which PKE.E draws its randomness. Sometimes we assume that all messages to be encrypted are drawn from a set $\mathsf{PKE}.\mathcal{M}$ of equal length messages and then let $\mathsf{PKE}.\mathcal{C}$ simply denote the set of all possible ciphertexts.

Indistinguishable from Random Security. We consider indistinguishable from random, chosen ciphertext attack (\$CCA) security as captured by Fig. 3. The definition multi-user and multi-challenge (allowing multiple challenges per user). It requires ciphertexts output by the encryption scheme be indistinguishable from random, even when given access to a decryption oracle. In this game, the adversary obtains the encryption key ek_u for user u by querying $\textsc{New}(u)$. It makes an encryption query $\textsc{Enc}(u, m)$ to receive a challenge encryption of m by u and a decryption query $\textsc{Dec}(u, c)$ to have u decrypt c. The adversary needs to distinguish between the real world ($b = 1$) in which a query to $\textsc{Enc}(u, m)$ returns a real encryption of m and the ideal world ($b = 0$) in which the same query returns a uniformly random element of $\mathsf{PKE}.\mathcal{C}(ek_u, |m|)$.

Table entry $M[u, c]$ stores the message encrypted in user u's challenge ciphertext c. If the adversary queries $\textsc{Dec}$ with a challenge ciphertext it returns $M[u, c]$ rather than performing the decryption. Prior works on memory-aware cryptography [17,19] considered other ways a decryption oracle might respond to challenge ciphertexts and argued that this is the "correct" convention. The advantage of an adversary $\mathcal{A}$ is0 $\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{PKE}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{PKE},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{PKE},0}(\mathcal{A})]$. In this and future definition we let $\mathcal{U}$ denote the set of allowed user identifiers u.

The general framework of capturing multi-user security by allowing the attacker to access separate instances of oracles for each user with shared secret bit across them is originally due to Bellare, Boldyreva, and Micali [5] who provided a definition for IND-CPA secure public-key encryption.

One-Wayness Security. Following the one-wayness security definitions in [21], we define variants of one-wayness security of PKE schemes in the multi-user, multi-challenge setting in Fig. 4. We define three variants - One-Wayness under

```
Game $\mathsf{G}^{\mathsf{mu\text{-}ow\text{-}}w}_{\mathsf{PKE}}(\mathcal{A})$
$\mathcal{H} \leftarrow\!\!\$\ \mathsf{PKE.IM}$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\!\!\$\ \mathsf{PKE.K}$
$\mathrm{O} \leftarrow \bot$ $//w = \mathsf{cpa}$
$\mathrm{O} \leftarrow \mathrm{PCO}$ $//w = \mathsf{pca}$
$\mathrm{O} \leftarrow (\mathrm{PCO}, \mathrm{CVO})$ $//w = \mathsf{pcva}$
$(m', u, i) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{New},\textsc{Chal},\mathrm{O},\mathcal{H}}$
Return $\mathrm{PCO}(u, m', \textsc{Chal}(u, i))$

$\textsc{Chal}(u, i)$
If $C[u, i] \neq \bot$
    Return $C[u, i]$
$m \leftarrow\!\!\$\ \mathsf{PKE}.\mathcal{M}$
$c \leftarrow\!\!\$\ \mathsf{PKE.E}^{\mathcal{H}}(ek_u, m)$
$C[u, i] \leftarrow c$
Return $c$

$\textsc{New}(u)$
Return $ek_u$

$\mathrm{PCO}(u, m, c)$
$m' \leftarrow \mathsf{PKE.D}^{\mathcal{H}}(dk_u, c)$
Return $m = m'$

$\mathrm{CVO}(u, c)$
$m' \leftarrow \mathsf{PKE.D}^{\mathcal{H}}(dk_u, c)$
Return $(m' \in \mathsf{PKE}.\mathcal{M})$
```

Fig. 4. Game defining $\mathsf{mu\text{-}ow\text{-}}w$ security of PKE for $w \in \{\mathsf{cpa}, \mathsf{pca}, \mathsf{pcva}\}$.

Chosen Plaintext Attacks (OW-CPA), One-Wayness under Plaintext Checking Attacks (OW-PCA) and One-Wayness under Plaintext and Validity Checking Attacks (OW-PCVA). The difference between each variant $w \in \{\mathsf{cpa}, \mathsf{pca}, \mathsf{pcva}\}$ is in the auxilliary oracle(s) O that the adversary is given access to.

In each variant, the adversary is tasked with finding the decryption of a challenge ciphertext which encrypt a message randomly sampled from $\mathsf{PKE}.\mathcal{M}$. In the game $\mathsf{G}^{\mathsf{mu\text{-}ow\text{-}cpa}}_{\mathsf{PKE}}$, the adversary does not have access to any auxilliary oracle as indicated by $\mathrm{O} \leftarrow \bot$. In the game $\mathsf{G}^{\mathsf{mu\text{-}ow\text{-}pca}}_{\mathsf{PKE}}$, the adversary has access to the Plaintext Checking Oracle PCO which takes as input a valid message-ciphertext pair, and returns true if the message is a valid decryption of the ciphertext and false otherwise. The adversary has access to both oracles, PCO and CVO, in $\mathsf{G}^{\mathsf{mu\text{-}ow\text{-}va}}_{\mathsf{PKE}}$ where CVO takes as input a ciphertext, and returns true if the ciphertext decrypts to a valid message. For each variant, we define $\mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}}w}_{\mathsf{PKE}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{mu\text{-}ow\text{-}}w}_{\mathsf{PKE}}]$. Note that an adversary may re-query $\textsc{Chal}(u, i)$ to get back the same ciphertext. This makes it hard to prove one-wayness, but easier write proofs starting from one-wayness. We sometime assume challenge identifiers, i, are drawn from a fixed set $\mathcal{I}$.

2.4 Key Encapsulation Mechanisms

Syntax. A key encapsulation mechanism, KEM, consists of three algorithms - the key generation algorithm ($\mathsf{KEM.K}$) that returns a pair of keys (ek, dk) where ek is the encapsulation key and dk is the corresponding decapsulation key, the encapsulation algorithm ($\mathsf{KEM.E}$) that takes the encapsulation key ek and returns a ciphertext-key pair (c, K) where $K \in \mathsf{KEM}.\mathcal{K}$ and the decapsulation algorithm $\mathsf{KEM.D}$ that takes the decapsulation key dk and a ciphertext c and returns a key K (or $\bot$ to indicate rejection). The syntax of these algorithms is shown in Fig. 2. Perfect correctness requires that $\mathsf{KEM.D}^{\mathcal{H}}(dk, c) = K$ for all $(ek, dk) \in [\mathsf{KEM.K}]$, all $\mathcal{H} \in \mathsf{KEM.IM}$, and all $(c, K) \in [\mathsf{KEM.E}^{\mathcal{H}}(ek)]$.

We define encryption keyspace $\mathsf{KEM.Ek} = \{ek : (ek, dk) \in [\mathsf{KEM.K}]\}$. For $ek \in \mathsf{KEM.Ek}$ we let $\mathsf{KEM}.\mathcal{C}(ek)$ denote the ciphertext set $\{c : (c, K) \in [\mathsf{KEM.E}(ek)]\}$ and define $|\mathsf{KEM}.\mathcal{C}| = \min_{ek \in \mathsf{KEM.Ek}} |\mathsf{KEM}.\mathcal{C}(ek)|$. We let $\mathsf{KEM}.\mathcal{R}$ denote the set

```
Game $\mathsf{G}^{\text{mu-\$cca}}_{\mathsf{KEM},b}(\mathcal{A})$
$\mathcal{H} \leftarrow\$ \mathsf{KEM.IM}$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\$ \mathsf{KEM.K}$
$b' \leftarrow\$ \mathcal{A}^{\text{New},\text{Encap}_b,\text{Decap},\mathcal{H}}$
Return $b' = 1$

New($u$)
Return $ek_u$

Encap$_b(u)$
$(c_1, K_1) \leftarrow\$ \mathsf{KEM.E}^{\mathcal{H}}(ek_u)$
$c_0 \leftarrow\$ \mathsf{KEM}.\mathcal{C}(ek_u)$
$K_0 \leftarrow\$ \mathsf{KEM}.\mathcal{K}$
$T[u, c_b] \leftarrow K_b$
Return $(c_b, K_b)$

Decap($u, c$)
If $T[u, c] \neq \perp$:
    Return $T[u, c]$
$K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c)$
Return $K$
```

Fig. 5. Game defining mu-\$cca security of KEM.

from which KEM.K draws it randomness. We say that KEM is *ϵ-uniform* if for all $ek \in \mathsf{KEM.Ek}$, $\mathcal{H} \in \mathsf{KEM.IM}$, and (not necessarily efficient) $\mathcal{D}$ it holds that

$$\Pr[\mathcal{D}(c) = 1 : c \leftarrow\$ \mathsf{KEM}.\mathcal{C}(ek)] - \Pr[\mathcal{D}(c) = 1 : (c, \cdot) \leftarrow\$ \mathsf{KEM.E}^{\mathcal{H}}(ek)] \leqslant \epsilon.$$

Indistinguishable from Random Security. Our notion of \$CCA security for KEMs is presented in Fig. 5, which requires that keys and ciphertexts output by the scheme be indistinguishable from random. The adversary is given a user instantiation oracle New, encapsulation oracle Encap, and a decapsulation oracle Decap. Its goal is to distinguish between the real world ($b = 1$) where Encap returns true outputs from KEM.E and the ideal world ($b = 0$) where it returns a pair (c, K) chosen uniformly at random from $\mathsf{KEM}.\mathcal{C}(ek) \times \mathsf{KEM}.\mathcal{K}$.

The table T stores the keys corresponding to challenge ciphertexts output by the encapsulation oracle. The decapsulation oracle uses T to respond to challenge queries. The advantage of an adversary $\mathcal{A}$ is defined as $\mathsf{Adv}^{\text{mu-\$cca}}_{\mathsf{KEM}}(\mathcal{A}) = \Pr[\mathsf{G}^{\text{mu-\$cca}}_{\mathsf{KEM},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\text{mu-\$cca}}_{\mathsf{KEM},0}(\mathcal{A})]$. We also define CCA security (via $\mathsf{Adv}^{\text{mu-cca}}_{\mathsf{KEM}}$ and $\mathsf{G}^{\text{mu-cca}}_{\mathsf{KEM},b}$) analogously to \$CCA security, except in the Encap oracle c_0 is set to equal c_1 rather than being sampled at random.

2.5 Symmetric Key Encryption

Syntax. A symmetric key encryption scheme, SKE, consists of three algorithms - the key generation algorithm (SKE.K) that returns a key K, the encryption algorithm (SKE.E) that takes the key K and a message m and returns ciphertext c, and the decryption algorithm SKE.D that takes the key K and a ciphertext c and returns message m (or $\perp$ to indicate rejection). The syntax of these algorithms is given in Fig. 2. Perfect correctness requires that $\mathsf{SKE.D}^{\mathcal{H}}(K, c) = m$ for $K \in [\mathsf{SKE.K}]$, all m, all $\mathcal{H} \in \mathsf{SKE.IM}$, and all $c \in [\mathsf{SKE.E}^{\mathcal{H}}(K, m)]$. We define the ciphertext, message, and expansion lengths of SKE by $\mathsf{SKE.cl}(|m|) = |\mathsf{SKE.E}^{\mathcal{H}}(K, m)|$ (requiring this to hold for all $\mathcal{H}, K, m$), $\mathsf{SKE.ml}(\mathsf{SKE.cl}(l)) = l$, and $\mathsf{SKE.xl} = \min_l \mathsf{SKE.cl}(l) - l$ respectively.

```
Game $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE},b}(\mathcal{A})$
    $\mathcal{H} \leftarrow\$ \mathsf{SKE.IM}$
    $K_{(\cdot)} \leftarrow\$ \mathsf{SKE.K}$
    $b' \leftarrow\$ \mathcal{A}^{\textsc{Enc}_b,\textsc{Dec},\mathcal{H}}$
    Return $b' = 1$

$\textsc{Enc}_b(u, m)$
    $c_1 \leftarrow\$ \mathsf{SKE.E}^{\mathcal{H}}(K_u, m)$
    $c_0 \leftarrow\$ \{0,1\}^{\mathsf{SKE.cl}(|m|)}$
    $M[u, c_b] \leftarrow m$
    Return $c_b$

$\textsc{Dec}(u, c)$
    If $M[u, c] \neq \perp$:
        Return $M[u, c]$
    $m \leftarrow \mathsf{SKE.D}^{\mathcal{H}}(K_u, c)$
    Return $m$
```

Fig. 6. Game defining mu-\$cca security of SKE.

Indistinguishable from Random CCA Security. Our notion of \$CCA security for SKE schemes is captured by Fig. 6, which requires that ciphertexts output by the encryption scheme be indistinguishable from ciphertexts chosen at random. In this game, the adversary is given access to an encryption oracle Enc and a decryption oracle Dec. The adversary needs to distinguish between the real world ($b = 1$), where Enc returns an encryption of m under K_u and the ideal world ($b = 0$) where the output of Enc is sampled uniformly at random. The advantage of an adversary $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE},0}(\mathcal{A})]$. We will only need "one-time" security in which the adversary only makes one encryption query per user.

3 Diffie-Hellman Definitions

In this section, we introduce the Computational Diffie-Hellman (CDH) assumptions we need for our later proofs. The first is a multi-user, multi-challenge variant of Strong CDH (which we need for one of our coming KEM proofs). We verify this is TAM-tightly implied by single-challenge variants. The second is a new definition we introduce, Pair CDH, which gives the adversary oracle access to a pairing from the group under consideration to a random group. We provide several results to understand the plausibility of Pair CDH security. We show that it always implies Gap CDH security and is {AM,TM}-tightly equivalent to CDH in algebraic/generic group models [13,26,29] or if the group has a pairing.

3.1 Group Syntax

A prime order group $\mathbb{G}$ is a tuple $(\mathbf{g}, p, \circ)$ where g is a group generator of prime order p under the group operation $\circ$. In our definitions we will treat the group as a priori fixed. We typically omit writing the group operation $\circ$ explicitly and instead write group operations using multiplicative notation. We let $\langle \mathbf{g} \rangle = \{\mathbf{g}^a : a \in \mathbb{N}\}$. The discrete log(arithm) of an element $X \in \langle \mathbf{g} \rangle$ is the value $\mathrm{dlog}(X) \in \mathbb{Z}_p$ such that $\mathbf{g}^{\mathrm{dlog}(X)} = X$. We let $1_{\mathbb{G}} = \mathbf{g}^0$ denote the identity element. A pairing from $\mathbb{G} = (\mathbf{g}, p, \circ)$ to $\mathbb{G}_2 = (\mathbf{g}_2, p_2, \circ_2)$ is a map $e : \langle \mathbf{g} \rangle \times \langle \mathbf{g} \rangle \to \langle \mathbf{g}_2 \rangle$ satisfying $e(\mathbf{g}^x, \mathbf{g}^y) = \mathbf{g}_2^{xy}$. We let $\mathbf{Time}(\mathbb{G})$ and $\mathbf{Mem}(\mathbb{G})$ denote the time and memory complexity of computing exponentiations or multiplications in $\langle \mathbf{g} \rangle$.

```
Game $\mathsf{G}^{\mathsf{x}}_{\mathbb{G}}(\mathcal{A})$
    $(\mathbf{g}, p, \circ) \leftarrow \mathbb{G}$
    $x_{(\cdot)} \leftarrow\$ \mathbb{Z}_p^*;\ y_{(\cdot,\cdot)} \leftarrow\$ \mathbb{Z}_p^*$
    $\mathrm{O} \leftarrow \bot$ $//\mathsf{G}^{\mathsf{cdh}}$
    $\mathrm{O} \leftarrow \textsc{Strong}$ $//\mathsf{G}^{\mathsf{scdh}}$
    $\mathrm{O} \leftarrow \textsc{Gap}$ $//\mathsf{G}^{\mathsf{gcdh}}$
    $\mathrm{O} \leftarrow \textsc{Pair}$ $//\mathsf{G}^{\mathsf{pcdh}}$
    $f \leftarrow\$ \mathsf{Inj}(\mathbb{Z}_p, \mathbb{Z}_p)$ $//\mathsf{G}^{\mathsf{pcdh}}$
    $(u, i, Z) \leftarrow\$ \mathcal{A}^{\textsc{New},\textsc{Chal},\mathrm{O}}$
    Return $(Z = \mathbf{g}^{x_u y_{u,i}})$

$\textsc{New}(u)$
    $X_u \leftarrow \mathbf{g}^{x_u}$; Return $X_u$

$\textsc{Chal}(u, i)$
    $Y_{u,i} \leftarrow \mathbf{g}^{y_{u,i}}$; Return $Y_{u,i}$

$\textsc{Gap}(A, B, C)$
    $a \leftarrow \mathrm{dlog}(A)$; $b \leftarrow \mathrm{dlog}(B)$
    Return $(C = \mathbf{g}^{ab})$

$\textsc{Strong}(u, B, C)$
    Return $\textsc{Gap}(\mathbf{g}^{x_u}, B, C)$

$\textsc{Pair}(A, B)$
    $a \leftarrow \mathrm{dlog}(A)$; $b \leftarrow \mathrm{dlog}(B)$
    Return $f(ab)$
```

Fig. 7. Security games capturing several variants of the computational Diffie-Hellman problem, namely, CDH, Gap CDH, Strong CDH, and Pair CDH. The last of these is a new notion we introduce which gives the attacker access to a pairing from $\mathbb{G}$ to a random group.

3.2 Computational Diffie-Hellman Variants

In this paper we will make use of several variants of the Computational Diffie-Hellman assumption. These security notions are defined by the game shown in Fig. 7. In each, the adversary is given access to a $\mathbf{g}^x$ and $\mathbf{g}^y$ with the goal of producing $\mathbf{g}^{xy}$. For our later security proofs, it was useful to write "multi-user" and "multi-challenge" version of these games. Thus rather than giving the adversary a single $\mathbf{g}^x$, we give it access to an oracle New which on input a string u (which we think of as identifying a user) returns a fresh $\mathbf{g}^{x_u}$. Similarly, the adversary is given access to an oracle Chal which on inputs string u and i (which we think of as identifying a challenge) returns a fresh $\mathbf{g}^{y_{u,i}}$. For the memory-tightness of future proofs, it is important that the attacker can repeat queries, obtaining the same result as before. The goal of the attacker is to return $\mathbf{g}^{x_u y_{u,i}}$ for any choice of u and i.

The different variants of CDH are captured by the games differing in what (if any) auxiliary oracle O the adversary is given. The standard notion of CDH security is captured by the game $\mathsf{G}^{\mathsf{cdh}}$ in which the adversary is not given any auxiliary oracle, as expressed by the code $\mathrm{O} \leftarrow \bot$. Gap CDH security [27] is captured by $\mathsf{G}^{\mathsf{gcdh}}$ in which the adversary's oracle Gap takes as input a tuple (A, B, C) and outputs a boolean indicating whether this is a valid Diffie-Hellman tuple (i.e. $C = \mathbf{g}^{\mathrm{dlog}(A)\,\mathrm{dlog}(B)}$). The Strong CDH game $\mathsf{G}^{\mathsf{scdh}}$ [2] is a weakened version of Gap CDH in which the oracle only allows tuples of the form $(\mathbf{g}^{x_u}, B, C)$.

The final variant is a new security notion we introduce called Pair CDH. In this game $\mathsf{G}^{\mathsf{pcdh}}$, the adversary is given access to the oracle Pair which on input (A, B) returns $f(\mathbf{g}^{ab})$ where a, b are the discrete logs of A, B and f is a random injection. This oracle can be thought of being a pairing to a random group $\mathbb{G}_2 = (\mathbf{g}_2, p, \circ_2)$ where $\mathbf{g}_2 = \textsc{Pair}(\mathbf{g}, \mathbf{g})$ and $h \circ_2 h' = f(f^{-1}(h) \circ f^{-1}(h'))$. Note that $\mathcal{A}$ is not able to efficiently compute the operation $\circ_2$.

```
Adversary $\mathcal{B}_{\mathsf{x}}^{\textsc{New},\textsc{Chal},\mathrm{O}}$
    $(\mathbf{g}, p, \circ) \leftarrow \mathbb{G}$
    $g \leftarrow\!\$\ \mathsf{Fcs}(\mathcal{U}, \mathbb{Z}_p^*)$; $h \leftarrow\!\$\ \mathsf{Fcs}(\mathcal{U} \times \mathcal{I}, \mathbb{Z}_p^*)$
    $X \leftarrow \textsc{New}(1)$; $Y \leftarrow \textsc{Chal}(1,1)$
    If $\mathsf{x} = \mathsf{scdh}$ then $\textsc{SimO} \leftarrow \textsc{SimStrong}$
    Else $\textsc{SimO} \leftarrow \mathrm{O}$
    $(u, i, Z) \leftarrow\!\$\ \mathcal{A}^{\textsc{SimNew},\textsc{SimChal},\textsc{SimO}}$
    $x'_u \leftarrow g(u)$; $y'_{u,i} \leftarrow h(u,i)$
    Return $(1, 1, Z^{1/(x'_u y'_{u,i})})$

$\textsc{SimNew}(u)$
    $x'_u \leftarrow g(u)$
    Return $X^{x'_u}$

$\textsc{SimChal}(u, i)$
    $y'_{u,i} \leftarrow h(u,i)$
    Return $Y^{y'_{u,i}}$

$\textsc{SimStrong}(u, B, C)$
    $x'_u \leftarrow g(u)$
    Return $\mathrm{O}(1, B, C^{1/x'_u})$
```

Fig. 8. Adversary used for Lemma 2.

For $\mathsf{x} \in \{\mathsf{cdh}, \mathsf{scdh}, \mathsf{gcdh}, \mathsf{pcdh}\}$ we define $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{x}}(\mathcal{A}) = \Pr[\mathsf{G}_{\mathbb{G}}^{\mathsf{x}}(\mathcal{A})]$. We sometimes need to restrict user identifiers, u, to be from some fixed set $\mathcal{U}$ and challenge identifiers, i, to be a from a fixed set $\mathcal{I}$.

Multi-challenge security. Standard proofs use Diffie-Hellman rerandomization techniques to show that single-challenge security TA-tightly implies multi-challenge security for most variants of Diffie-Hellman-based security notions. The following lemma extends this to TAM-tightness for the notions considered in this paper. The proof is an extension of standard Diffie-Hellman rerandomization techniques that picks the values used for rerandomization as the output of a random function, rather than picking them randomly and storing them.

Lemma 2 (Single-challenge ⇒ multi-challenge). *Let $\mathbb{G}$ be a group and $\mathsf{x} \in \{\mathsf{cdh}, \mathsf{scdh}, \mathsf{gcdh}, \mathsf{pcdh}\}$. Let $\mathcal{A}$ be an adversary for $\mathsf{G}_{\mathbb{G}}^{\mathsf{x}}$ with $(q_{\textsc{New}}, q_{\textsc{Chal}}, q_{\mathrm{O}}) = \mathbf{Query}(\mathcal{A})$. Then we can construct a $(\mathsf{Fcs}(\mathcal{U}, \mathbb{Z}_p^*), \mathsf{Fcs}(\mathcal{U} \times \mathcal{I}, \mathbb{Z}_p^*))$-oracle adversary $\mathcal{B}_{\mathsf{x}}$ (given in the proof) such that*

$$\begin{aligned}
&\mathsf{Adv}_{\mathbb{G}}^{\mathsf{x}}(\mathcal{A}) = \mathsf{Adv}_{\mathbb{G}}^{\mathsf{x}}(\mathcal{B}_{\mathsf{x}}) \\
&\mathbf{Query}(\mathcal{B}_{\mathsf{x}}) = (1, 1, q_{\mathrm{O}}) \\
&\mathbf{Time}^*(\mathcal{B}_{\mathsf{x}}) = O(\mathbf{Time}(\mathcal{A}) + (q_{\textsc{New}} + q_{\textsc{Chal}} + q_{\mathrm{O}} + 1)\mathbf{Time}(\mathbb{G})) \\
&\mathbf{Mem}^*(\mathcal{B}_{\mathsf{x}}) = O(\mathbf{Mem}(\mathcal{A}) + 2\mathbf{Mem}(\mathbb{G})).
\end{aligned}$$

Proof of Lemma 2). Consider the adversary $\mathcal{B}_{\mathsf{x}}$ shown in Fig. 8. It makes a single query $\textsc{New}(1)$ to obtain a group element X and a single query $\textsc{Chal}(1,1)$ to obtain a group element Y. Then it runs $\mathcal{A}$. Let $x = \mathrm{dlog}(X)$ and $y = \mathrm{dlog}(Y)$.

It responds to $\textsc{SimNew}(u)$ queries with $X^{x'_u}$ for x'_u the output of its random function. Letting $x_u = x x'_u$, note that $X^{x'_u} = \mathbf{g}^{x_u}$ and x_u is uniformly random because x'_u is. So this oracle has the correct distribution. It responds to $\textsc{SimChal}$ queries with $Y^{y'_{u,i}}$ for $y'_{u,i}$ output by its random function. Letting $y_{u,i} = y y'_{u,i}$, note that $Y^{y'_{u,i}} = \mathbf{g}^{y_{u,i}}$ and that $y_{u,i}$ is uniformly random because $y'_{u,i}$ is.

For CDH, Gap CDH, or Pair CDH security $\mathcal{B}_{\mathsf{x}}$ gives $\mathcal{A}$ direct access to O. For Strong CDH security ($\mathsf{x} = \mathsf{scdh}$), $\mathcal{B}_{\mathsf{x}}$ simulates the oracle by replacing a query (u, B, C) with a query $(1, B, C^{1/x'_u})$ which has the same behavior.

When $\mathcal{A}$ finally halts and outputs (u, i, Z) the adversary $\mathcal{B}_{\mathsf{x}}$ halts and outputs $(1, 1, Z^{1/(x'_u y'_{u,i})})$. We claim that $\mathcal{B}_{\mathsf{x}}$ wins whenever $\mathcal{A}$ would. To see this, note that if $\mathcal{A}$ wins then $Z = \mathbf{g}^{x_u y_{u,i}} = \mathbf{g}^{(xx'_u)(yy'_{u,i})}$ and so $Z^{1/(x'_u y'_{u,i})} = \mathbf{g}^{xy}$. □

3.3 Studying Pair CDH

Pair CDH is a new computational assumption that we've introduced for this work. In this section we provide a few results to give a sense of its difficulty. Namely, we note that Pair CDH security implies Gap CDH security and that it is equivalent to CDH security in certain settings (if $\mathbb{G}$ has an efficient pairing to some $\mathbb{G}_2$, in the algebraic model, and in the generic group model). Given Lemma 2, we focus on the case that the adversary makes only single query each to New and Chal. For brevity, we sketch the relationships here.

Pair CDH $\Rightarrow$ Gap CDH. To see that Pair CDH security implies Gap CDH security we need only note that $\text{Gap}(A, B, C) = \mathsf{true}$ if and only if $\text{Pair}(A, B) = \text{Pair}(\mathbf{g}, C)$. Hence a Pair CDH adversary can efficiently simulate the view of a Gap CDH adversary.

CDH + pairing $\Rightarrow$ Pair CDH. We claim that CDH security implies Pair CDH security if $\mathbb{G}$ has an efficient pairing e to some group $\mathbb{G}_2$ with generator $\mathbf{g}_2$. We can achieve TAM-tightness in this implication by using a $\mathsf{Inj}(\langle \mathbf{g}_2 \rangle, \mathbb{Z}_p)$-oracle adversary. Letting f' denote the random injection, our CDH adversary can simulate Pair by responding to queries for (A, B) with $f'(e(A, B))$. If $a = \text{dlog}(A)$ and $b = \text{dlog}(B)$, then $f'(e(A, B)) = f'(e(\mathbf{g}, \mathbf{g})^{ab})$. Note that $F(\cdot) = e(\mathbf{g}, \mathbf{g})^{(\cdot)}$ is an injection and so $f(\cdot) = f'(e(\mathbf{g}, \mathbf{g})^{(\cdot)})$ is a random injection. Hence this perfectly emulates Pair.

CDH + AGM/GGM $\Rightarrow$ Pair CDH. We claimed that CDH security implies Pair CDH security in the algebraic group model. More precisely, we {AM,TM}-tightly show that CDH security implies Pair CDH security using a $\mathsf{Fcs}(\mathbb{Z}_p^6, \mathbb{Z}_p)$-oracle adversary. We show a way to imperfectly simulate Pair for algebraic adversaries such that distinguishing this from the real oracle requires the ability to solve the discrete log problem (given $\mathbf{g}^c$ for a random c, find c). Noting that CDH security implies discrete log security gives our claim.

Let X and Y denote the challenge group elements and let $x = \text{dlog}(X)$ and $y = \text{dlog}(Y)$. An algebraic adversary, when making an oracle query (A, B) to Pair is required to additionally provide "explanations" (a_1, a_2, a_3) and (b_1, b_2, b_3) such that $A = g^{a_1} X^{a_2} Y^{a_3}$ and $B = g^{b_1} X^{b_2} Y^{b_3}$. Then the true Pair would respond with $f((a_1 + a_2 x + a_3 y) \cdot (b_1 + b_2 x + b_3 y))$. Our CDH adversary will think of this input to f as a degree-two polynomial $P_{A,B}(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}_p[\mathbf{x}, \mathbf{y}]$ whose coefficients it can compute given the explanations for A and B. Letting $(c_1, c_2, \ldots, c_6)$ denote these coefficients and $f' \in \mathsf{Fcs}(\mathbb{Z}_p^6, \mathbb{Z}_p)$, we simulate the output of Pair as $f'(c_1, c_2, \ldots, c_6)$. Distinguishing this from the true oracle requires finding (A, B) and (A', B') such that $P_{A,B} \neq P_{A',B'}$ (as polynomials), but $P_{A,B}(x, y) = P_{A',B'}(x, y)$. Using analysis techniques from [4], we can use the ability to find such "colliding" polynomials to solve the discrete log problem. We provide details of this analysis in the full version [23].

To achieve TM-tightness, the discrete log reduction picks two of the PAIR oracle queries at random and assumes that they give colliding polynomials. To achieve AM-tightness, we can check every pair of queries for collisions using the memory-tight rewinding technique of Auerbach, et al [3]. Namely, each time we reach a new PAIR oracle query while running the Pair CDH adversary, we pause and run an extra copy of that adversary from the start using the same coins. While running this extra copy, each time it makes a PAIR oracle query we check if this gives a colliding polynomial with the query we paused at in the first adversary. Ignoring memory tightness, $\mathcal{A}$ could simply remember all of the PAIR oracle queries and check them at the end of execution, but then it is not clear how to achieve better time efficiency than checking each pair of queries.

When working in a generic group model [26,29] we can use the same line of reasoning and then information theoretically bound the probability that an adversary finds colliding polynomials by $O(q^2/p)$ where q is the number of queries the Pair CDH adversary makes.

4 Hashed ElGamal KEMs

In this section we present the first example of KEMs with TAM-tight proofs in the multi-challenge setting. The KEMs we consider are variants of the ECIES and Cramer-Shoup Hashed ElGamal KEMs. These variants augment the existing schemes by adding random strings to the ciphertexts and random oracle queries. Our reductions make use of these strings to store pertinent information that will be needed to answer later oracle queries.

4.1 Augmented ECIES

AUGMENTED VERSION. We start with the ECIES [1] variant of Hashed ElGamal. Our augmented version of ECIES includes a random string a in the ciphertext. The augmented ECIES key encapsulation mechanism $\mathsf{aECIES}[\mathbb{G}, \mathcal{K}, l]$ is parameterized by a group $\mathbb{G} = (\mathbf{g}, p, \circ)$, key space $\mathcal{K}$, and length of the random string, l. The parameters $\mathbb{G}, \mathcal{K}$, and l are fixed for an instantiation of ECIES, so we use aECIES and $\mathsf{aECIES}[\mathbb{G}, \mathcal{K}, l]$ interchangeably. We define the scheme as follows with $\mathsf{aECIES}.\mathcal{K} = \mathcal{K}$ and $\mathsf{aECIES}.\mathsf{IM} = \mathsf{Fcs}(\{0,1\}^l \times \mathbb{G}, \mathcal{K})$.

```
$\underline{\mathsf{aECIES.K}}$
$x \leftarrow_\$ \mathbb{Z}_p^*$
$ek \leftarrow \mathbf{g}^x$
$dk \leftarrow x$
Return $(ek, dk)$
```

```
$\underline{\mathsf{aECIES.E}^{\mathcal{H}}(ek)}$
$a \leftarrow_\$ \{0,1\}^l$
$y \leftarrow_\$ \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(a, Z)$
Return $((a, Y), K)$
```

```
$\underline{\mathsf{aECIES.D}^{\mathcal{H}}(dk, (a, Y))}$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(a, Z)$
Return $K$
```

Overview of existing techniques and associated challenges. Bhattacharyya [7] studied ECIES in the memory-aware setting. They pointed out the technique of simulating random oracles with PRFs introduced in [3] cannot be used for this family of KEMs, as in general, decapsulation queries cannot be simulated by the reduction. For example, if a PRF F is used to compute hashes as $\mathsf{F}(k, Z)$ instead of the random oracle H, for a decapsulation query Y the reduction would need to return $\mathsf{F}(k, Y^{dk})$ which it cannot compute.[5]

Bhattacharyya used the map-then-prf technique as a workaround for groups with pairings. In this technique, the input Z to the random oracle is first operated on by a bilinear map $e(g, Z)$, and then by the PRF F. Hence, the query $\mathrm{H}(Z)$ is simulated as $\mathsf{F}(k, e(\mathbf{g}, Z))$ and a decapsulation query for Y can be simulated as $\mathsf{F}(k, e(ek, Y))$ for all non-challenge ciphertexts. The reduction remembers the challenge ciphertext and returns appropriately when it is queried to Decap.

This does not scale to the multi-user, multi-challenge setting since it requires that the reduction remembers all the challenge ciphertexts, incurring a memory overhead. Our solution for augmented ECIES combines Ghoshal et al.'s message encoding technique [17] with the map-then-rf technique. We encode the identifying information of challenge ciphertexts in a using a random injection so that this information can be recovered when an appropriate oracle query is made. To avoid the need for an efficiently computable pairing we make use of our new Pair CDH assumption. Our result is captured in Theorem 1.

Theorem 1 (Pair CDH $\Rightarrow$ \$CCA). *Let* $\mathsf{aECIES} = \mathsf{aECIES}[\mathbb{G}, \mathcal{K}, l]$ *where* $\mathbb{G} = (\mathbf{g}, p, \circ)$ *is a prime order group. Define* $D_1 = \{0,1\}^l \times \mathbb{G}$ *and* $D_2 = \mathcal{U} \times [q_{\text{Encap}}]$. *Let* $\mathcal{A}$ *be an adversary with* $\mathbf{Query}(\mathcal{A}) = (q_{\text{New}}, q_{\text{Encap}}, q_{\text{Decap}}, q_{\text{H}})$ *and assume* $2^l > |\mathcal{U}| \cdot q_{\text{Encap}}$. *Then Fig. 12 gives a* $(\mathsf{Fcs}(D_1, \mathcal{K}), \mathsf{Inj}(D_2, \{0,1\}^l))$*-oracle adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aECIES}}(\mathcal{A}) \leqslant 2\mathsf{Adv}^{\mathsf{pcdh}}_{\mathbb{G}}(\mathcal{B}) + \frac{q^2_{\text{Encap}}}{2^l} + \frac{2q_{\text{Encap}}(2q_{\text{Decap}} + |\mathcal{U}| \cdot q_{\text{H}})}{2^l(p-1)}$$

$$\mathbf{Query}(\mathcal{B}) = ((q_{\text{New}} + q_{\text{Encap}}), (q_{\text{Encap}} + q_{\text{Decap}} + q_{\text{H}}), (q_{\text{Encap}} + q_{\text{Decap}} + 2q_{\text{H}}))$$

$$\mathbf{Time}^*(\mathcal{B}) = O(\mathbf{Time}(\mathcal{A})) \text{ and } \mathbf{Mem}^*(\mathcal{B}) = O(\mathbf{Mem}(\mathcal{A})).$$

Choosing the auxiliary string length. When instantiating this scheme one must choose the parameter l which determines the length of a. Larger l incurs a communication cost, while too small of a l can harm the concrete security results. We can expect the $q^2_{\text{Encap}}/2^l$ term to dominate the information theoretic part of the bound. With cautious choice of $l = 256$, the size of the ciphertext is not too significantly increased, but even with $q_{\text{Encap}} = 2^{64}$ encapsulations (an intentional overestimate of what seems likely) we get $q^2_{\text{Encap}}/2^l = 2^{-128}$.

Several of coming theorems use of an auxiliary string a of length l. Similar reasoning applies and $l = 256$ seems like a sufficient choice for all of them.

[5] We discuss the use of PRFs to match prior work, but in the oracle adversary framework, we use random function oracles instead.

Intuition. For each Encap query, our Pair CDH adversary programs the random string a as the output of a random injection applied to user identity u and counter i and simulates the random oracle $\mathcal{H}(a, Z)$ as $\mathcal{H}(a, \text{Pair}(\mathbf{g}, Z))$. This allows us to simulate decapsulations because $\text{Pair}(\mathbf{g}^x, \mathbf{g}^y) = \text{Pair}(\mathbf{g}, \mathbf{g}^{xy})$.

Our adversary simulates j-th challenge ciphertexts $Y_{u.i}$ as $\text{Chal}(u, i)$. To determine if a decapsulation query (u, a, Y) is for a challenge ciphertext, the reduction first inverts a to obtain (v, j). If $v = u$, it re-queries $\text{Chal}(u, j)$ obtain the corresponding ciphertext $Y_{u,j}$. If $Y = Y_{u,j}$, the reduction assumes this was a challenge ciphertext. Finally, when the adversary $\mathcal{A}$ queries the oracle H with (a, Z) such that $a^{-1} = (u, j)$ and $\text{Pair}(\mathbf{g}, Z) = \text{Pair}(\text{New}(u), \text{Chal}(u, j))$, the reduction outputs Z and wins the Pair CDH game.

Proof (of Theorem 1). We use a sequence of hybrids H_0^1 through H_1^1, H_0^2 through H_2^2, and H_0^3 through H_3^3 presented in Figs. 9, 10, and 11 where we establish the following claims that upper bound the advantage of adversary $\mathcal{A}$.

1. $\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aECIES}}(\mathcal{A}) = 2\Pr[\mathsf{H}_0^1] - 1$
2. $\Pr[\mathsf{H}_0^1] \leqslant \Pr[\mathsf{H}_1^1] + \frac{q_{\text{Encap}}^2}{2 \cdot 2^l}$
3. $\Pr[\mathsf{H}_1^1] = \Pr[\mathsf{H}_0^2] = \Pr[\mathsf{H}_1^2]$
4. $\Pr[\mathsf{H}_1^2] = \Pr[\mathsf{H}_2^2] = \Pr[\mathsf{H}_0^3]$
5. $\Pr[\mathsf{H}_0^3] = \Pr[\mathsf{H}_1^3]$
6. $\Pr[\mathsf{H}_1^3] \leqslant \Pr[\mathsf{H}_2^3] + \frac{q_{\text{Encap}}(2q_{\text{Decap}} + |\mathcal{U}| \cdot q_{\text{H}})}{2^l(p-1)}$
7. $\Pr[\mathsf{H}_2^3] \leqslant \Pr[\mathsf{H}_3^3] + \Pr[\mathsf{bad}]$
8. $\Pr[\mathsf{H}_3^3] \leqslant \frac{1}{2}$
9. $\Pr[\mathsf{bad}] \leqslant \mathsf{Adv}^{\mathsf{pcdh}}_{\mathbb{G}}(\mathcal{B})$

Transition to H_0^1. We claim that the view of $\mathcal{A}$ in H_0^1 is identical to its view in $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aECIES},b}$ (Fig. 5) if b is chosen uniformly. In the latter, Encap_1 returns a ciphertext-key pair (c_1, K_1) such that c is the encapsulation of K_1, and Encap_0 returns a ciphertext-key pair (c_0, K_0) where c_0 and K_0 are uniformly random elements of the ciphertext space and key space respectively. The same holds for the Encap oracle in H_0^1. The table T in $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aECIES},b}$ is indexed by (u, c) and stores the key that was returned by the Encap_b oracle for ciphertext c. The table T in H_0^1 behaves analogously. The Decap oracle in $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aECIES},b}$ returns key K_b that was output by the Encap_b oracle when queried on a challenge ciphertext, and returns honest decapsulations otherwise. The same is true for H_0^1. Note that H_0^1's final output is whether $b' = b$, so standard conditional probability calculations give that $\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aECIES}}(\mathcal{A}) = 2\Pr[\mathsf{H}_0^1] - 1$.

Transition H_0^1 to H_1^1. In H_1^1, we make the following changes.

1. In the Encap oracle we switch from
   (a) sampling the values $a_{(.,.)}$ uniformly to assigning them as the output of a random injection g evaluated on (u, i).
   (b) sampling the values $y_{(.,.)}$ uniformly to assigning them the output of a random function h evaluated on (u, i). Switching to the random function h does not change the view of the adversary because the ordered pair (u, i) never repeats.
2. In the Decap oracle we switch the If condition from checking whether $T[u, a, Y] \neq \bot$ to evaluating the boolean $u = v \wedge j \in I[u] \wedge Y = \mathbf{g}^{h(u,i)}$. These conditons are equivalent since $T[u, a, Y] \neq \bot$ iff $a = g(u, j)$ and $Y = \mathbf{g}^{h(u,j)}$ for some $j \in I[u]$.

```
Hybrids $\mathsf{H}^1_\kappa$ for $0 \leqslant \kappa \leqslant 1$
$b \leftarrow\$ \{0,1\}$
$i \leftarrow 0$
$x_{(\cdot)} \leftarrow\$ \mathbb{Z}_p^*$; $X_{(\cdot)} \leftarrow \mathbf{g}^{x_{(\cdot)}}$
$\mathcal{H} \leftarrow\$ \mathsf{aECIES.IM}$
$g \leftarrow\$ \mathsf{Inj}(D_2, \{0,1\}^l)$ //$\mathsf{H}^1_{[1,\infty)}$
$h \leftarrow\$ \mathsf{Fcs}(D_2, \mathbb{Z}_p^*)$ //$\mathsf{H}^1_{[1,\infty)}$
$b' \leftarrow\$ \mathcal{A}^{\text{New},\text{Encap}_b,\text{Decap},\text{H}}$
Return $b' = b$

$\text{Encap}_b(u)$
$i \leftarrow i + 1$
$I[u] \leftarrow I[u] \cup \{i\}$
$a_{u,i} \leftarrow \{0,1\}^l$ //$\mathsf{H}^1_{[0,1)}$
$y_{u,i} \leftarrow\$ \mathbb{Z}_p^*$ //$\mathsf{H}^1_{[0,1)}$
$a_{u,i} \leftarrow g(u,i)$ //$\mathsf{H}^1_{[1,\infty)}$
$y_{u,i} \leftarrow h(u,i)$ //$\mathsf{H}^1_{[1,\infty)}$
$Y_{u,i} \leftarrow \mathbf{g}^{y_{u,i}}$
$Z_{u,i} \leftarrow X_u^{y_{u,i}}$
$K^1_{u,i} \leftarrow \mathcal{H}(a_{u,i}, Z_{u,i})$
$K^0_{u,i} \leftarrow\$ \mathcal{K}$
$T[u, a_{u,i}, Y_{u,i}] \leftarrow K^b_{u,i}$
Return $((a_{u,i}, Y_{u,i}), K^b_{u,i})$

$\text{New}(u)$
Return $X_u$

$\text{H}(a, Z)$
Return $\mathcal{H}(a, Z)$

$\text{Decap}(u, a, Y)$
$(v, j) \leftarrow g^{-1}(a)$ //$\mathsf{H}^1_{[1,\infty)}$
$y_{u,i} \leftarrow h(u, j)$ //$\mathsf{H}^1_{[1,\infty)}$
If $T[u, a, Y] \neq \perp$ //$\mathsf{H}^1_{[0,1)}$
If $u = v \wedge j \in I[u] \wedge Y = \mathbf{g}^{y_{u,j}}$ //$\mathsf{H}^1_{[1,\infty)}$
    Return $T[u, a, Y]$
$Z \leftarrow Y^{x_u}$
$K \leftarrow \mathcal{H}(a, Z)$
Return $K$
```

Fig. 9. First set of hybrids H^1_0 through H^1_1 used for proof of Theorem 1.

The only change in the adversary's view comes from 1(a). The switching lemma (Lemma 1) gives us $\Pr[\mathsf{H}^1_0] \leqslant \Pr[\mathsf{H}^1_1] + \frac{q^2_{\text{Encap}}}{2 \cdot 2^l}$.

Transition H^1_1 to H^2_0. The transition to hybrid H^2_0 is shown in Fig. 10. We have highlighted the ways in which H^2_0 differs from H^1_2. Our changes are the following.

1. In the Encap oracle, the table $\tilde{T}$ has been added to record the responses to the $\mathcal{H}$ queries made within Encap.
2. In the H oracle, an If block is added to check if the input tuple (a, Z) was previously queried to $\mathcal{H}$ from within the Encap oracle. In essence, the If block returns $\mathcal{H}(a, Z)$ when queried on a challenge ciphertext. The H oracle would behave the same way without the If block, as in H^1_1.

Hence, $\Pr[\mathsf{H}^1_1] = \Pr[\mathsf{H}^2_0]$.

Transition H^2_0 to H^2_1. In game H^2_1, we introduce the pairing oracle Pair. It is only used by oracles within the game; the adversary does not have direct access to it. Note that $\text{Pair}(\mathbf{g}, Z) = \text{Pair}(\mathbf{g}^{x_u}, Y)$ iff $Z = Y^{x_u}$.

In the H oracle, we switch the condition from checking $\tilde{T}[a, Z] \neq \perp$ to evaluating the boolean $j \in I[u] \wedge \text{Pair}(\mathbf{g}, Z) = \text{Pair}(X_u, Y)$ where $(u, j) = g^{-1}(a), Y = \mathbf{g}^{h(u,j)}$. These two conditions are equivalent. Note that $\tilde{T}[a, Z] \neq \perp$ means it was filled in an encapsulation query. Suppose this was the j-th query and was to u.

```
Hybrids $\mathsf{H}^2_\kappa$ for $0 \leqslant \kappa \leqslant 2$
$b \leftarrow\!\$\ \{0,1\}$
$i \leftarrow 0$
$x_{(\cdot)} \leftarrow\!\$\ \mathbb{Z}_p^*$; $X_{(\cdot)} \leftarrow \mathbf{g}^{x(\cdot)}$
$\mathcal{H} \leftarrow\!\$\ \mathsf{aECIES.IM}$ //$\mathsf{H}^2_{[0,2)}$
$\tilde{\mathcal{H}} \leftarrow\!\$\ \mathsf{Fcs}(D_1, \mathcal{K})$ //$\mathsf{H}^2_{[2,\infty)}$
$q \leftarrow\!\$\ \mathsf{Inj}(\mathbb{Z}_p, \mathbb{Z}_p)$ //$\mathsf{H}^2_{[1,\infty)}$
$g \leftarrow\!\$\ \mathsf{Inj}(D_2, \{0,1\}^l)$; $h \leftarrow\!\$\ \mathsf{Fcs}(D_2, \mathbb{Z}_p^*)$
$b' \leftarrow\!\$\ \mathcal{A}^{\text{NEW},\text{ENCAP}_b,\text{DECAP},\text{H}}$
Return $b' = b$

$\text{ENCAP}_b(u)$
$i \leftarrow i + 1$
$I[u] \leftarrow I[u] \cup \{i\}$
$a_{u,i} \leftarrow g(u,i)$
$y_{u,i} \leftarrow h(u,i)$
$Y_{u,i} \leftarrow \mathbf{g}^{y_{u,i}}$
$Z_{u,i} \leftarrow X_u^{y_{u,i}}$
$K^1_{u,i} \leftarrow \mathcal{H}(a_{u,i}, Z_{u,i})$ //$\mathsf{H}^2_{[0,2)}$
$K^1_{u,i} \leftarrow \tilde{\mathcal{H}}(a_{u,i}, \text{PAIR}(\mathbf{g}, Z_{u,i}))$ //$\mathsf{H}^2_{[2,\infty)}$
$K^0_{u,i} \leftarrow\!\$\ \mathcal{K}$
$\tilde{T}[a_{u,i}, Z_{u,i}] \leftarrow K^1_{u,i}$
$T[u, a_{u,i}, Y_{u,i}] \leftarrow K^b_{u,i}$
Return $((a_{u,i}, Y_{u,i}), K^b_{u,i})$
```

```
$\text{NEW}(u)$
Return $X_u$

$\text{H}(a, Z)$
$(u,j) \leftarrow g^{-1}(a)$ //$\mathsf{H}^2_{[1,\infty)}$
$y_{u,j} \leftarrow h(u,j)$; $Y_{u,j} \leftarrow \mathbf{g}^{y_{u,j}}$ //$\mathsf{H}^2_{[1,\infty)}$
If $\tilde{T}[a, Z] \neq \perp$ //$\mathsf{H}^2_{[0,1)}$
If $j \in I[u] \wedge \text{PAIR}(\mathbf{g}, Z) = \text{PAIR}(X_u, Y_{u,j})$ //$\mathsf{H}^2_{[1,\infty)}$
    Return $\tilde{T}[a, Z]$
Return $\mathcal{H}(a, Z)$ //$\mathsf{H}^2_{[0,2)}$
Return $\tilde{\mathcal{H}}(a, \text{PAIR}(\mathbf{g}, Z))$ //$\mathsf{H}^2_{[2,\infty)}$

$\text{PAIR}(X, Y)$ //$\mathsf{H}^2_{[1,\infty)}$, internal
$x \leftarrow \text{dlog}(X)$; $y \leftarrow \text{dlog}(Y)$
Return $q(xy)$

$\text{DECAP}(u, a, Y)$
$(v,j) \leftarrow g^{-1}(a)$
$y_{u,j} \leftarrow h(u,j)$
If $u = v \wedge j \in I[u] \wedge Y = \mathbf{g}^{y_{u,j}}$
    Return $T[u, a, Y]$
$Z \leftarrow Y^{x_u}$
$K \leftarrow \mathcal{H}(a, Z)$ //$\mathsf{H}^2_{[0,2)}$
$K \leftarrow \tilde{\mathcal{H}}(a, \text{PAIR}(\mathbf{g}, Z))$ //$\mathsf{H}^2_{[2,\infty)}$
Return $K$
```

Fig. 10. Second set of hybrids H^2_0 through H^2_2 used for proof of Theorem 1. Grey highlighting is used to show the difference between H^1_1 and H^2_0. Note that PAIR is used internally by other oracles and is not directly accessible to $\mathcal{A}$.

Then $a = g(u,j)$ must hold, j would have been added to $I[u]$, and $Z = X_u^{h(u,j)}$ (so $\text{PAIR}(\mathbf{g}, Z) = \text{PAIR}(X_u, Y)$). Thus, $\Pr[\mathsf{H}^2_0] = \Pr[\mathsf{H}^2_1]$.

TRANSITION H^2_1 TO H^2_2(map-then-rf). In H^2_2, the random function $\mathcal{H}$ is replaced by a random function $\tilde{\mathcal{H}}$ from $\mathsf{Fcs}(D_1, \mathcal{K})$ where $D_1 = \{0,1\}^l \times \mathbb{Z}_p$. We replaced the function $\mathcal{H}$ as $\mathcal{H}(a, Z) = \tilde{\mathcal{H}}(a, \text{PAIR}(\mathbf{g}, Z))$. Then $\mathcal{H}$ is a random function if $\tilde{\mathcal{H}}$ is, $\text{PAIR}(g, .)$ is an injection. Hence, $\Pr[\mathsf{H}^2_1] = \Pr[\mathsf{H}^2_2]$.

TRANSITION H^2_2 TO H^3_0. Game H^3_0 is shown in Fig. 11. We have highlighted the ways in which H^3_0 differs from H^2_2. In H^3_0, the key $K^0_{(.,.)}$ is assigned the output of a random function $\mathcal{E}$ from $\mathsf{Fcs}(D_1, \mathcal{K})$, instead of sampling it at random. This does not change the adversary's view as a never repeats. Hence, $\Pr[\mathsf{H}^2_2] = \Pr[\mathsf{H}^3_0]$.

TRANSITION H^3_0 TO H^3_1. In H^3_1, the table T is no longer used. The change is in the DECAP oracle where the If condition evaluates whether the bit b is 0. Note that the DECAP oracle always returns K^b. When $b = 0$, it returns $\mathcal{E}(a, \text{PAIR}(\mathbf{g}, Z))$

```
Hybrids $\mathsf{H}^3_\kappa$ for $0 \leqslant \kappa \leqslant 3$
$b \leftarrow_{\$} \{0,1\}$
$i \leftarrow 0$
$x_{(\cdot)} \leftarrow_{\$} \mathbb{Z}_p^*$; $X_{(\cdot)} \leftarrow \mathbf{g}^{x(\cdot)}$
$\tilde{\mathcal{H}} \leftarrow_{\$} \mathsf{Fcs}(D_1, \mathcal{K})$
$\mathcal{E} \leftarrow_{\$} \mathsf{Fcs}(D_1, \mathcal{K})$
$q \leftarrow_{\$} \mathsf{Inj}(\mathbb{Z}_p, \mathbb{Z}_p)$
$g \leftarrow_{\$} \mathsf{Inj}(D_2, \{0,1\}^l)$;
$h \leftarrow_{\$} \mathsf{Fcs}(D_2, \mathbb{Z}_p^*)$
$\mathsf{bad} \leftarrow \mathsf{false}$ $//\mathsf{H}^3_{[3,\infty)}$
$b' \leftarrow_{\$} \mathcal{A}^{\text{NEW},\text{ENCAP}_b,\text{DECAP},\text{H}}$
Return $b' = b$

$\text{ENCAP}_b(u)$
$i \leftarrow i + 1$
$I[u] \leftarrow I[u] \cup \{i\}$ $//\mathsf{H}^3_{[0,2)}$
$a_{u,i} \leftarrow g(u,i)$
$y_{u,i} \leftarrow h(u,i)$
$Y_{u,i} \leftarrow \mathbf{g}^{y_{u,i}}$
$Z_{u,i} \leftarrow X_u^{y_{u,i}}$
$K^1_{u,i} \leftarrow \tilde{\mathcal{H}}(a_{u,i}, \text{PAIR}(\mathbf{g}, Z_{u,i}))$
$K^0_{u,i} \leftarrow \mathcal{E}(a_{u,i}, \text{PAIR}(\mathbf{g}, Z_{u,i}))$
$\tilde{T}[a_{u,i}, Z_{u,i}] \leftarrow K^1_{u,i}$ $//\mathsf{H}^3_{[0,3)}$
$T[u, a_{u,i}, Y_{u,i}] \leftarrow K^b_{u,i}$ $//\mathsf{H}^3_{[0,1)}$
Return $((a_{u,i}Y_{u,i}), K^b_{u,i})$

$\text{NEW}(u)$
Return $X_u$

$\text{H}(a, Z)$
$(u,j) \leftarrow g^{-1}(a)$
$y_{u,j} \leftarrow h(u,j)$; $Y_{u,j} \leftarrow \mathbf{g}^{y_{u,j}}$
If $j \in I[u] \wedge \text{PAIR}(\mathbf{g}, Z) = \text{PAIR}(X_u, Y_{u,j})$ $//\mathsf{H}^3_{[0,2)}$
If $\text{PAIR}(\mathbf{g}, Z) = \text{PAIR}(X_u, Y_{u,j})$ $//\mathsf{H}^3_{[2,\infty)}$
        Return $\tilde{T}[a, Z]$ $//\mathsf{H}^3_{[0,3)}$
        $\mathsf{bad} \leftarrow \mathsf{true}$ $//\mathsf{H}^3_{[3,\infty)}$
        $\mathsf{ABORT}$ $//\mathsf{H}^3_{[3,\infty)}$
Return $\tilde{\mathcal{H}}(a, \text{PAIR}(\mathbf{g}, Z))$

$\text{PAIR}(X, Y)$ //Internal
$x \leftarrow \text{dlog}(X)$; $y \leftarrow \text{dlog}(Y)$
Return $q(xy)$

$\text{DECAP}(u, a, Y)$
$(v,j) \leftarrow g^{-1}(a)$
$y_{u,j} \leftarrow h(u,j)$
$Z \leftarrow Y^{x_u}$
If $u = v \wedge j \in I[u] \wedge Y = \mathbf{g}^{y_{u,j}}$ $//\mathsf{H}^3_{[0,2)}$
If $u = v \wedge j \in I[u] \wedge Y = \mathbf{g}^{y_{u,j}} \wedge b = 0$ $//\mathsf{H}^3_{[1,2)}$
If $u = v \wedge Y = \mathbf{g}^{y_{u,j}} \wedge b = 0$ $//\mathsf{H}^3_{[2,\infty)}$
    Return $T[u, a, Y]$ $//\mathsf{H}^3_{[0,1)}$
    Return $\mathcal{E}(a, \text{PAIR}(\mathbf{g}, Z))$ $//\mathsf{H}^3_{[1,\infty)}$
$K \leftarrow \tilde{\mathcal{H}}(a, \text{PAIR}(\mathbf{g}, Z))$
Return $K$
```

Fig. 11. Third set of hybrids H^3_0 through H^3_3 used for proof of Theorem 1.

which is what was used to compute K^0 in ENCAP. The If condition evaluates to false under two cases

1. (a,Y) is a challenge ciphertext and $b = 1$
2. (a,Y) is not a challenge ciphertext.

In both these cases, the DECAP oracle returns $\tilde{\mathcal{H}}(a, \text{PAIR}(\mathbf{g}, Z))$ which is the same as K^1. Therefore, this modification does not change the view of the adversary and $\Pr[\mathsf{H}^3_0] = \Pr[\mathsf{H}^3_1]$.

TRANSITION H^3_1 TO H^3_2. In H^3_2 we change the If statements in the H and DECAP oracles. In both places, we remove the check $j \in I[u]$. H^3_1 and H^3_2 differ only under the following events:

1. The adversary makes a DECAP query for (u, a, Y) such that $(u, j) = g^{-1}(a)$ and $Y = \mathbf{g}^{h(u,j)}$ but $j \notin I[u]$.

2. The adversary makes a H query for (a, Z) such that $\textsc{Pair}(\mathbf{g}, Z) = \textsc{Pair}(X_u, Y)$ and $Y = \mathbf{g}^{h(u,j)}$ (where $(u, j) = g^{-1}(a)$) but $j \notin I[u]$.

To cause either of these events, the adversary must "guess" a point a in the image of g other than the at most $q_{\textsc{Encap}}$ such points for which it was given the corresponding ciphertexts by $\textsc{Encap}$. We analyze the probability of this event in H_1^3 where the behaviors of $\textsc{Decap}$ and H can only depend on the values of $h(u, \cdot)$ and $g(u, \cdot)$ for inputs in $I[u]$. Hence, $\mathcal{A}$ only learns about $h(u, \cdot)$ and $g(u, \cdot)$ through its queries to $\textsc{Encap}$. At some fixed point in time, let n denote the total number of $\textsc{Encap}$ queries that $\mathcal{A}$ has made so far and n_u denote the number of $\textsc{Encap}$ queries it has made to user u.

First consider a query $\textsc{Decap}(u, a, Y)$. For this to differ between the games it must hold that a is in the image of $g(u, \cdot)$. There are $q_{\textsc{Encap}}$ total such values, of which the adversary has already seen n_u from $\textsc{Encap}$. This is out of the 2^l values in the codomain, of which the adversary has already seen n from $\textsc{Encap}$. Thus there is a $(q_{\textsc{Encap}} - n_u)/(2^l - n) \leqslant (q_{\textsc{Encap}} - n_u + n)/2^l \leqslant 2q_{\textsc{Encap}}/2^l$ chance that the adversary picks such an a. The adversary must additionally have guessed the correct $\mathbf{g}^{h(u,j)}$, which it has an at most $1/(p-1)$ chance of having done.

Now consider a query $\mathrm{H}(a, Z)$. For this to differ between the games it must hold that a is in the image of $g(\cdot, \cdot)$. There are $|\mathcal{U}| \cdot q_{\textsc{Encap}}$ total such values, of which the adversary has already seen n from $\textsc{Encap}$. This is out of the 2^l values in the codomain, of which the adversary has already seen n from $\textsc{Encap}$. Thus there is a $(|\mathcal{U}| \cdot q_{\textsc{Encap}} - n)/(2^l - n) \leqslant |\mathcal{U}| \cdot q_{\textsc{Encap}}/2^l$ chance that the adversary picks such an a. The adversary must additionally have guessed the correct $X_u^{h(u,j)}$, which it has an at most $1/(p-1)$ chance of having done.

Applying a union bound over all $\textsc{Decap}$ and H queries gives the claimed bound $\Pr[\mathsf{H}_1^3] \leqslant \Pr[\mathsf{H}_2^3] + (2q_{\textsc{Encap}}q_{\textsc{Decap}} + |\mathcal{U}| \cdot q_{\textsc{Encap}}q_{\mathrm{H}})/(2^l \cdot (p-1))$.

$\textsc{Transition}$ H_2^3 $\textsc{to}$ H_3^3. In H_3^3, the table $\tilde{T}$ is removed. The change is in the H oracle, wherein if the adversary queries the H oracle with a challenge ciphertext, the H oracle aborts. Using the fundamental lemma of game playing, this probability is bounded as $\Pr[\mathsf{H}_2^3] \leqslant \Pr[\mathsf{H}_3^3] + \Pr[\mathsf{bad}]$.

In H_3^3, the adversary is unable to compute $\mathrm{H}(a, Z)$ for challenge ciphertexts using the H oracle or the $\textsc{Decap}$ oracle. The $\textsc{Decap}$ oracle returns K_b on all challenge ciphertexts (which is the same value that was returned by the $\textsc{Encap}$ oracle) and the H oracle aborts on challenge ciphertexts. Therefore, the adversary's view in H_3^3 is independent of the bit b. Hence, $\Pr[\mathsf{H}_3^3] \leqslant 1/2$.

To bound $\Pr[\mathsf{bad}]$, we construct an adversary $\mathcal{B}$ given in Fig. 12 against the Pair CDH security of $\mathbb{G}$. We claim that $\mathcal{B}$ perfectly simulates H_3^3 for $\mathcal{A}$. Note that the challenge ciphertext is computed using $\mathcal{B}$'s challenge oracle, and the If condition in the H has been replaced with a call to $\mathcal{B}$'s $\textsc{Pair}$ oracle. Whenever the flag bad is set, $\mathcal{B}$ outputs the corresponding (u, j, Z) and wins the Pair CDH game. Therefore, $\Pr[\mathsf{bad}] \leqslant \mathsf{Adv}_{\mathbb{G}}^{\mathsf{pcdh}}(\mathcal{B})$. □

```
Adversary $\mathcal{B}_{\mathsf{pcdh}}^{\textsc{New},\textsc{Chal},\textsc{Pair}}$
$b \leftarrow_\$ \{0,1\}$
$i \leftarrow 0$
$\tilde{\mathcal{H}} \leftarrow_\$ \mathsf{Fcs}(D_1, \mathcal{K})$
$\mathcal{E} \leftarrow_\$ \mathsf{Fcs}(D_1, \mathcal{K})$
$g \leftarrow_\$ \mathsf{Inj}(D_2, \{0,1\}^l)$
$\mathsf{bad} \leftarrow \mathsf{false}$
$b' \leftarrow_\$ \mathcal{A}^{\mathrm{O}}$
Return $\perp$

$\textsc{SimH}(a, Z)$
$(u, j) \leftarrow g^{-1}(a)$
$X_u \leftarrow \textsc{New}(u)$
$Y_{u,j} \leftarrow \textsc{Chal}(u, j)$
If $\textsc{Pair}(X_u, Y_{u,j}) \leftarrow \textsc{Pair}(\mathbf{g}, Z)$
    $\mathsf{bad} \leftarrow \mathsf{true}$
    $\mathsf{OUTPUT}(u, j, Z)$
Return $\tilde{\mathcal{H}}(a, \textsc{Pair}(\mathbf{g}, Z))$

$\textsc{SimEncap}_b(u)$
$i \leftarrow i + 1$
$X_u \leftarrow \textsc{New}(u)$
$a_{u,i} \leftarrow g(u, i)$
$Y_{u,i} \leftarrow \textsc{Chal}(u, i)$
$w \leftarrow \textsc{Pair}(X_u, Y_{u,i})$
$K^1_{u,i} \leftarrow \tilde{\mathcal{H}}(a, w)$
$K^0_{u,i} \leftarrow \mathcal{E}(a, w)$
Return $((a_{u,i} Y_{u,i}), K^b_{u,i})$

$\textsc{SimDecap}(u, a, Y)$
$(v, j) \leftarrow g^{-1}(a)$
$X_u \leftarrow \textsc{New}(u)$
$Y_{u,j} \leftarrow \textsc{Chal}(u, j)$
If $u = v \wedge Y = Y_{u,j} \wedge b = 0$
    Return $\mathcal{E}(a, \textsc{Pair}(X_u, Y_{u,j}))$
$K \leftarrow \tilde{\mathcal{H}}(a, \textsc{Pair}(X_u, Y_{u,j}))$
Return $K$
```

Fig. 12. Adversary $\mathcal{B}$ for Theorem 1; $\mathrm{O} = \{\textsc{New}, \textsc{SimEncap}_b, \textsc{SimDecap}, \textsc{SimH}\}$.

4.2 Augmented Cramer-Shoup KEM

Augmented Version. In this section we present a memory-tight reduction for an augmented version of the Cramer-Shoup KEM [8]. The augmented Cramer-Shoup key encapsulation mechanism $\mathsf{aCS}[\mathbb{G}, \mathcal{K}, l]$ is parameterized by a group $\mathbb{G} = (\mathbf{g}, p, \circ)$, key space $\mathcal{K}$, and length of the random string, l. The parameters $\mathbb{G}, \mathcal{K}$, and l are constants for any instantiation of the Cramer-Shoup KEM and hence, we override notation and use aCS and $\mathsf{aCS}[\mathbb{G}, \mathcal{K}, l]$ interchangably. We let $\mathsf{aCS.IM} = \mathsf{Fcs}(\{0,1\}^l \times \mathbb{G}^2, \mathcal{K})$ and define the scheme as follows.

```
$\mathsf{aCS.K}$
$x \leftarrow_\$ \mathbb{Z}_p^*$
$ek \leftarrow \mathbf{g}^x$
$dk \leftarrow x$
Return $(ek, dk)$

$\mathsf{aCS.E}^{\mathcal{H}}(ek)$
$a \leftarrow_\$ \{0,1\}^l$
$y \leftarrow_\$ \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow ek^y$
$K \leftarrow \mathcal{H}(a, Y, Z)$
Return $((a, Y), K)$

$\mathsf{aCS.D}^{\mathcal{H}}(dk, (a, Y))$
$Z \leftarrow Y^{dk}$
$K \leftarrow \mathcal{H}(a, Y, Z)$
Return $K$
```

Overview of existing techniques and associated challenges. A traditional security reduction for the Cramer-Shoup KEM from the Strong CDH problem in the single-user, single-challenge setting would use $l = 0$ and the lazy sampling technique to simulate $\mathcal{H}$ as a random oracle. The reduction would maintain a table T to store $\mathcal{H}$ queries and corresponding responses. When the adversary makes a decapsulation query on Y, the reduction would check the table to see if an entry $T[Y, Z]$ exists such that $\textsc{Gap}(X, Y, Z) = \mathsf{true}$ where $X = \mathbf{g}^x$ is the public key. If the entry exists, it would return the corresponding value, and if it does not exist, the reduction would sample a new uniformly random element K from the key set $\mathcal{K}$. The reduction would then store $T[Y, \_] \leftarrow K$ and return K. The second entry would be filed in the table T when the adversary makes a hash query for (Y, Z) such that $\textsc{Gap}(X, Y, Z) = \mathsf{true}$. The reduction wins the Strong CDH game if it outputs a Z such that $Z = \mathbf{g}^{xy}$, which it does by waiting for the Cramer-Shoup adversary to query its hash oracle on inputs (Y, Z) such that $\textsc{Gap}(X, Y, Z) = \mathsf{true}$.

Like with ECIES, the random oracle simulation using PRF technique cannot be used here as it is not possible for the reduction to simulate decapsulation queries using the PRF. Bhattacharya avoided this issue using the map-then-prf technique, defining $\mathcal{H}(Y, Z)$ so that when $Z = Y^{dk}$, $\mathcal{H}(Y, Z)$ is computable from ek, Y. This allows properly responding to decapsulation queries when the reduction only has access to Y and cannot compute $Z = Y^{dk}$.

This proof breaks in the multi-challenge setting because it is not clear how to identify and respond to challenge ciphertexts without simply storing them all. Once again, augmentation with a random string a allows encoding the information needed to respond to queries appropriately. Our result is captured by the following theorem. The proof of Theorem 2 is given in the full version [23].

Theorem 2 (Strong CDH ⇒ \$CCA). *Let $\mathbb{G} = (\mathbf{g}, p, \circ)$ be a group of prime order p. Let $\mathcal{K}$ and l be fixed. Define* $\mathsf{aCS} = \mathsf{aCS}[\mathbb{G}, \mathcal{K}, l]$.

Let $D_1 = \{0,1\}^l \times \mathbb{G} \times \mathbb{G} \cup \{\star\}$ and $D_2 = \mathcal{U} \times [q_{\text{Encap}}]$. Let $\mathcal{A}$ be a mu-\$cca *adversary with* $\mathbf{Query}(\mathcal{A}) = (q_{\text{New}}, q_{\text{Encap}}, q_{\text{Decap}}, q_{\text{H}})$ *and assume $2^l > 2q_{\text{Encap}}$. We construct a* $(\mathsf{Fcs}(D_1, \mathcal{K}) \times \mathsf{Inj}(D_2, \{0,1\}^l))$*-oracle adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\text{mu-\$cca}}_{\mathsf{aCS}}(\mathcal{A}) \leqslant 2\mathsf{Adv}^{\mathsf{scdh}}_{\mathbb{G}}(\mathcal{B}) + \frac{q^2_{\text{Encap}}}{2^l} + \frac{2q_{\text{Encap}}(2q_{\text{Decap}} + |\mathcal{U}| \cdot q_{\text{H}})}{2^l(p-1)}$$

$$\mathbf{Query}(\mathcal{B}) = (q_{\text{New}}, (q_{\text{Encap}} + q_{\text{Decap}} + q_{\text{H}}), q_{\text{H}})$$

$$\mathbf{Time}^*(\mathcal{B}) = O(\mathbf{Time}(\mathcal{A})) \text{ and } \mathbf{Mem}^*(\mathcal{B}) = O(\mathbf{Mem}(\mathcal{A})).$$

Intuition. For this proof, we program the random string a as the output of a random injection g applied to a user identity u and a counter i and simulate the random oracle H using the random function $\tilde{\mathcal{H}}(\alpha(a, Y, Z))$ where α is an injection from $\{0,1\}^l \times \mathbb{G} \times \mathbb{G} \cup \{\star\}$ to $\{0,1\}^l \times \{0,1\} \times \mathbb{G}^2$ that can be computed with knowing Z when $Z = Y^{ek}$.

Our reduction adversary plays the Strong CDH game where it gets its challenge ciphertexts Y from the oracle $\textsc{Chal}(u, i)$. To determine if a decapsulation query (u, a, Y) is for a challenge ciphertext, the reduction first inverts a to obtain (v, j). If $u = v$, it queries $\textsc{Chal}(u, j)$ to obtain the ciphertext $Y_{u,j}$. Then it can simply check if $Y = Y_{u,j}$. Finally, when the adversary $\mathcal{A}$ queries the oracle H with (a, Y, Z) such that $g^{-1}(a) = (u, j)$ and $\textsc{Strong}(u, Y, Z) = \mathsf{true}$ and $Y = \textsc{Chal}(u, j)$, the reduction outputs (u, j, Z) and wins the Strong CDH game.

5 Fujisaki-Okamoto Transformation

The Fujisaki-Okamoto [14,15] transformations use a random oracle to construct an IND-CCA secure KEM from a weakly (IND-CPA) secure PKE scheme. Bhattacharyya presented memory-tight reductions for the modules analyzed in [21] in the single-user, single-challenge setting [7]. In our work, we use the message encoding technique along with map-then-rf technique to prove memory-tight reductions for one version of the Fujisaki-Okamoto transformations in the multi-user, multi-challenge setting. In the following subsections, we present the definitions and memory-tight reductions for the transformations T, aV, and $\mathsf{aU}^{\perp}$.

5.1 Transformation T [IND-CPA → OW-PCA]

The transformation T constructs a deterministic OW-PCA secure public key encryption scheme $\mathsf{TKE} = \mathsf{T}[\mathsf{PKE}]$ from an IND-CPA secure public key encryption scheme PKE. We define TKE as follows with $\mathsf{TKE.IM} = \mathsf{Fcs}(\mathsf{PKE}.\mathcal{M}, \mathsf{PKE}.\mathcal{R}) \times \mathsf{PKE.IM}$ and $\mathsf{TKE}.\mathcal{M} = \mathsf{PKE}.\mathcal{M}$.

```
$\underline{\mathsf{TKE.E}^{\mathcal{H}\times\mathcal{H}'}(ek, m)}$
$c \leftarrow \mathsf{PKE.E}^{\mathcal{H}'}(ek, m; \mathcal{H}(m))$
Return $c$

$\underline{\mathsf{TKE.D}^{\mathcal{H}\times\mathcal{H}'}((ek, dk), c)}$
$m' \leftarrow \mathsf{PKE.D}^{\mathcal{H}'}(dk, c)$
If $m' = \perp$ or $\mathsf{PKE.E}^{\mathcal{H}'}(ek, m'; \mathcal{H}(m')) \neq c$
    Return $\perp$
Return $m'$
```

For key generation, $\mathsf{TKE.K}$ samples $(ek, dk) \leftarrow\$ \mathsf{PKE.K}$ and outputs $(ek, (ek, dk))$. Note that TKE is *tidy*, meaning that if $m = \mathsf{TKE.D}^{\mathcal{H}\times\mathcal{H}'}((ek, dk), c)$, then $c = \mathsf{TKE.E}^{\mathcal{H}\times\mathcal{H}'}(ek, m)$.

We present a memory-tight reduction for T in the multi-user, multi-challenge setting using the randomness programming technique. Our result is captured in Theorem 3, whose proof is given in the full version [23].

Theorem 3 (IND-CPA ⇒ OW-PCA). *Let* $\mathsf{TKE} = \mathsf{T}[\mathsf{PKE}]$. *If* PKE *is* δ*-correct, then* TKE *is* δ'*-correct for* $\delta'(q) = (q+1)\delta(q)$. *Let* $\mathcal{A}$ *be an adversary against* TKE *with* $\mathbf{Query}(\mathcal{A}) = (q_{\mathrm{NEW}}, q_{\mathrm{CHAL}}, q_{\mathrm{PCO}}, q_{\mathrm{H}})$. *Assume* PKE*'s algorithms make at most* q_{PKE} *oracle queries and define* $q^* = q_{\mathrm{H}} + q_{\mathrm{CHAL}}(q_{\mathsf{PKE}} + 1) + q_{\mathrm{PCO}}(2q_{\mathsf{PKE}} + 1) + q_{\mathsf{PKE}}$. *We construct an* $(\mathsf{Fcs}(\mathsf{PKE}.\mathcal{M}, \mathsf{PKE}.\mathcal{R}), \mathsf{Inj}^{\pm}(\mathcal{U} \times \mathcal{I}, \mathsf{PKE}.\mathcal{M}))$*-oracle adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\text{mu-ow-pca}}_{\mathsf{TKE}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\text{mu-cpa}}_{\mathsf{PKE}}(\mathcal{B}) + |\mathcal{U}| \cdot \delta'(q^*) + \frac{0.5q^2_{\mathrm{CHAL}} + |\mathcal{U}||\mathcal{I}|(q_{\mathrm{H}} + q_{\mathrm{PCO}} + 1)}{|\mathsf{PKE}.\mathcal{M}|}$$

$$\mathbf{Query}(\mathcal{B}) = (q_{\mathrm{NEW}}, q_{\mathrm{CHAL}})$$
$$\mathbf{Time}^*(\mathcal{B}) = O(\mathbf{Time}(\mathcal{A}))$$
$$\mathbf{Mem}^*(\mathcal{B}) = O(\mathbf{Mem}(\mathcal{A})).$$

The notion of CPA security we require is interesting in that the adversary's encryption queries are of the form (u, i, m) where i is a "challenge identifier" and if it exactly repeats a query (u, i, m) it is given back the ciphertext from the earlier query.

Intuition. For this proof, we program the random messages m to be the output of a random injection g applied to user identifier u and counter i. Our reduction adversary simulates the Chal oracle for $\mathcal{A}$ by its own encryption oracle on $g(u, i)$. If any message that $\mathcal{A}$ queries to its random oracle or outputs at the end of execution is in the image of g, then out reduction assumes it is in the real world and outputs 1. In the ideal world, the view of $\mathcal{A}$ is independent of g so we can information theoretically bound the probability it finds such a message.

5.2 Transformation aV [OW-PCA → OW-PCVA]

The augmented transformation aV constructs a deterministic OW-PCVA secure public key encryption scheme $\mathsf{VKE} = \mathsf{aV}[\mathsf{TKE}]$ from a deterministic OW-PCA secure scheme TKE. The unaugmented V transformation was given (with a single-user, single-challenge memory-tight reduction) in [7]. Our augmentation adds a random string to the encryption key which is included with every hash function query. We define VKE as follows with $\mathsf{VKE.IM} = \mathsf{Fcs}(\{0,1\}^l \times \mathsf{TKE}.\mathcal{M}, \{0,1\}^\gamma) \times \mathsf{TKE.IM}$ and $\mathsf{VKE}.\mathcal{M} = \mathsf{TKE}.\mathcal{M}$, where l and γ are fixed.

```
$\underline{\mathsf{VKE.E}^{\mathcal{H}\times\mathcal{H}'}((a, ek), m)}$
$c_1 \leftarrow \mathsf{TKE.E}^{\mathcal{H}'}(ek, m)$
$c_2 \leftarrow \mathcal{H}(a, m)$
$c \leftarrow (c_1, c_2)$
Return $c$
```

```
$\underline{\mathsf{VKE.D}^{\mathcal{H}\times\mathcal{H}'}((a, ek, dk), c)}$
$(c_1, c_2) \leftarrow c$
$m' \leftarrow \mathsf{TKE.D}^{\mathcal{H}'}(dk, c_1)$
If $m' = \perp$ or $\mathcal{H}(a, m') \neq c_2$ or $\mathsf{TKE.E}^{\mathcal{H}'}(ek, m') \neq c_1$
    Return $\perp$
Return $m'$
```

For key generation, VKE.K samples $a \leftarrow_\$ \{0,1\}^l$ and $(ek, dk) \leftarrow_\$ \mathsf{TKE.K}$, then returns $((a, ek), (a, ek, dk))$. Note that aV is tidy and if T is tidy and δ'-correct, then aV is δ'-correct.

We present a memory-tight reduction for aV in the multi-user, multi-challenge setting using the randomness programming technique. Our result is captured in Theorem 4 whose proof is given in the full version of the paper [23].

Theorem 4 (OW-PCA ⇒ OW-PCVA). *Let* $\mathsf{VKE} = \mathsf{aV}[\mathsf{TKE}]$ *and suppose* TKE *is* δ'*-correct. Let* $\mathcal{A}$ *be an adversary against* VKE *with* $\mathbf{Query}(\mathcal{A}) = (q_{\mathrm{U}}, q_{\mathrm{CHAL}}, q_{\mathrm{PCO}}, q_{\mathrm{CVO}}, q_{\mathrm{H}})$*. Assume* TKE*'s algorithms make at most* q_{TKE} *oracle queries and define* $q^* = q_{\mathsf{TKE}}(q_{\mathrm{H}} + q_{\mathrm{CHAL}} + 2q_{\mathrm{PCO}} + 2q_{\mathrm{CVO}})$*. We construct an* $(\mathsf{Fcs}(\{0,1\}^l \times \mathsf{TKE}.\mathcal{M}, \{0,1\}^\gamma), \mathsf{Fcs}(\mathsf{TKE}.\mathcal{C}, \{0,1\}^\gamma), \mathsf{Inj}^{\pm}(\mathcal{U}, \{0,1\}^l))$*-oracle adversary* $\mathcal{B}$ *against* TKE *such that*

$$\mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}va}}_{\mathsf{VKE}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}pca}}_{\mathsf{TKE}}(\mathcal{B}) + 4|\mathcal{U}| \cdot \delta'(q^*) + 0.5|\mathcal{U}|^2/2^l + q_{\mathrm{CVO}}/2^\gamma$$
$$\mathbf{Query}(\mathcal{B}) = (q_{\mathrm{NEW}}, q_{\mathrm{CHAL}}, q_{\mathrm{PCO}}, q_{\mathrm{H}} \cdot q_{\mathsf{TKE}})$$
$$\mathbf{Time}^*(\mathcal{B}) = O(\mathbf{Time}(\mathcal{A})) \text{ and } \mathbf{Mem}^*(\mathcal{B}) = O(\mathbf{Mem}(\mathcal{A})).$$

5.3 Augmented Transformation $\mathsf{aU}^{\perp}$ [OW-PCVA $\rightarrow$ \$IND-CCA]

The transformation $\mathsf{aU}^{\perp}$ constructs an IND-CCA secure key encapsulation mechanism $\mathsf{aUEM} = \mathsf{aU}^{\perp}[\mathsf{VKE}]$ from a deterministic, OW-PCVA secure public key encryption scheme VKE. We define aUEM as follows where $\mathsf{aUEM.K} = \mathsf{VKE.K}$ and $\mathsf{aUEM.IM} = \mathsf{Fcs}(\{0,1\}^l \times \mathsf{VKE}.\mathcal{M} \times \mathsf{VKE}.\mathcal{C}, \mathcal{K}) \times \mathsf{VKE.IM}$ ($\mathcal{K}$ is an arbitrary set used as the key set of aUEM).

```
$\mathsf{aUEM.E}^{\mathcal{H}\times\mathcal{H}'}(ek)$
$m \leftarrow_\$ \mathsf{VKE}.\mathcal{M}$
$a \leftarrow_\$ \{0,1\}^l$
$c \leftarrow \mathsf{VKE.E}^{\mathcal{H}'}(ek, m)$
$K \leftarrow \mathcal{H}(a, m, c)$
Return $((a, c), K)$
```

```
$\mathsf{aUEM.D}^{\mathcal{H}\times\mathcal{H}'}(dk, (a, c))$
$m \leftarrow \mathsf{VKE.D}^{\mathcal{H}'}(dk, c)$
If $m = \perp$
    Return $\perp$
$K \leftarrow \mathcal{H}(a, m, c)$
Return $K$
```

The following theorem gives our security result.

Theorem 5 (OW-PCVA $\Rightarrow$CCA). *Let* $\mathsf{aUEM} = \mathsf{aU}^{\perp}[\mathsf{VKE}]$ *where* VKE *is tidy and* δ'*-correct. Let* $\mathcal{A}$ *be an adversary against* aUEM *with* $\mathbf{Query}(\mathcal{A}) = (q_{\mathrm{NEW}}, q_{\mathrm{ENCAP}}, q_{\mathrm{DECAP}}, q_{\mathrm{H}})$. *Assume* VKE*'s algorithms make at most* q_{VKE} *oracle queries and define* $q^* = q_{\mathsf{VKE}}(2q_{\mathrm{H}} + q_{\mathrm{ENCAP}} + 2q_{\mathrm{DECAP}})$. *Let* $D_1 = \{0,1\}^l \times \mathsf{VKE}.\mathcal{M} \cup \{\star\} \times \mathsf{VKE}.\mathcal{C}$ *and* $D_2 = \mathcal{U} \times \mathcal{I}$,. *We construct an* $(\mathsf{Inj}^{\pm}(D_2, \{0,1\}^l), \mathsf{Fcs}(D_1, \mathcal{K}), \mathsf{Fcs}(D_1, \mathcal{K}))$*-oracle adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathsf{mu\text{-}cca}}_{\mathsf{aUEM}}(\mathcal{A}) \leqslant 2\mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}va}}_{\mathsf{VKE}}(\mathcal{B}) + 2|\mathcal{U}| \cdot \delta'(q^*) + 6|\mathcal{U}| \cdot |\mathcal{I}|/2^l$$
$$\mathbf{Query}(\mathcal{B}) = (q_{\mathrm{NEW}}, (q_{\mathrm{ENCAP}} + q_{\mathrm{DECAP}} + q_{\mathrm{H}}), q_{\mathrm{H}}, q_{\mathrm{DECAP}})$$
$$\mathbf{Time}^*(\mathcal{B}) = O(\mathbf{Time}(\mathcal{A})) \text{ and } \mathbf{Mem}^*(\mathcal{B}) = O(\mathbf{Mem}(\mathcal{A})).$$

INTUITION. This proof is very similar to the proof of Theorem 2. Once again, we program the random string a to be the output of the injective function $g(u, i)$, and the message m to be the output of the random function $h(u, i)$. We use the map-then-rf technique to simulate the oracle H using the random function $\tilde{\mathcal{H}}(\alpha(a, m, c))$ where α is an injective function.

The reduction adversary gets challenge ciphertexts c from the CHAL oracle. To simulate the H oracle it uses the PCO oracle and to simulate the DECAP oracle, it uses its CVO oracle. When the aUEM adversary queries the H oracle with a tuple (a, m, c) such that $\mathrm{PCO}(u, m, c) = 1$ and $c = \mathrm{CHAL}(u, j)$ where $(u, j) = g^{-1}(a)$, the reduction outputs (u, j, m).

6 Memory-Tight Reduction for PKE Schemes

In this section, we provide a modified version of the TAM-tight security proof from [17] to show the security of the KEM/DEM construction of public key encryption. Thus, combining one of the KEMs studied in the rest of the paper with an appropriate symmetric encryption scheme gives a PKE scheme with a TAM-tight reduction in the multi-user, multi-challenge setting.

KEM/DEM Scheme. Let SKE be a symmetric key encryption scheme and KEM be a key encapsulation mechanism. Then the KEM/DEM encryption scheme $\mathsf{KD} = \mathsf{KD}[\mathsf{KEM}, \mathsf{SKE}]$ is defined as follows, with $\mathsf{KD.IM} = \mathsf{KEM.IM} \times \mathsf{SKE.IM}$.

```
$\underline{\mathsf{KD}[\mathsf{KEM},\mathsf{SKE}].\mathsf{K}}$
$(ek, dk) \leftarrow_{\$} \mathsf{KEM.K}$
Return $(ek, dk)$

$\underline{\mathsf{KD}[\mathsf{KEM},\mathsf{SKE}].\mathsf{E}^{\mathcal{H}\times\mathcal{H}'}(ek, m)}$
$(c^k, K) \leftarrow_{\$} \mathsf{KEM.E}^{\mathcal{H}}(ek)$
$c^d \leftarrow_{\$} \mathsf{SKE.E}^{\mathcal{H}'}(K, m)$
Return $(c^k, c^d)$

$\underline{\mathsf{KD}[\mathsf{KEM},\mathsf{SKE}].\mathsf{D}^{\mathcal{H}\times\mathcal{H}'}(dk, c)}$
$(c^k, c^d) \leftarrow c$
$K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk, c^k)$
If $K = \perp$ then return $\perp$
Return $\mathsf{SKE.D}^{\mathcal{H}'}(K, c^d)$
```

The following theorem TAM-tightly proves the security of KD.

Theorem 6. *Let* SKE *be a symmetric key encryption scheme,* KEM *be a* ϵ-*uniform key encapsulation mechanism, and* $\mathsf{KD} = \mathsf{KD}[\mathsf{SKE}, \mathsf{KEM}]$. *Let* $T' = \mathcal{U} \times \bigcup_{ek \in \mathsf{KEM.Ek}} \mathsf{KEM}.\mathcal{C}(ek) \times \mathbb{N}$, *Let* $T = \mathcal{U} \times \mathsf{KEM.Ek}$, $D_{(u,ek)} = [q_{\text{Enc}}]$ *and* $R_{(u,ek)} = \mathsf{KEM}.\mathcal{C}(ek)$. $D'_{(u,c^k,l)} = \{0,1\}^l$, *and* $R'_{(u,c^k,l)} = \{0,1\}^{\mathsf{SKE.cl}(l)}$. *Let* $\mathcal{A}$ *be a* $\mathsf{mu\text{-}\$cca}$ *adversary against* KD *with* $\mathbf{Query}(\mathcal{A}) = (q_{\text{New}}, q_{\text{Enc}}, q_{\text{Dec}}, q_{\text{H}})$. *Then we can construct a* $(\mathsf{SKE.IM}, \mathsf{Inj}^{\pm}(T', D', R'))$-*oracle adversary* $\mathcal{B}_{\mathsf{KEM}}$ *and* $(\mathsf{KEM.IM}, \mathsf{Fcs}(\mathcal{U}, \mathsf{KEM}.\mathcal{R}), \mathsf{Inj}^{\pm}(T, D, R))$-*oracle adversary against* $\mathcal{B}_{\mathsf{SKE}}$ *such that*

$$\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD}}(\mathcal{A}) \leqslant 2\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE}}(\mathcal{B}_{\mathsf{SKE}}) + q_{\text{Enc}} \cdot \epsilon$$
$$+ \frac{1.5q_{\text{Enc}}^2 + 2q_{\text{Enc}}q_{\text{Dec}}}{|\mathsf{KEM}.\mathcal{C}|} + \frac{q_{\text{Enc}}^2 + 2q_{\text{Dec}}}{2^{\mathsf{SKE.xl}}}.$$

The complexities of $\mathcal{B}_{\mathsf{KEM}}$ *and* $\mathcal{B}_{\mathsf{SKE}}$ *basically match those of* $\mathcal{A}$. *Moreover,* $\mathcal{B}_{\mathsf{SKE}}$ *makes at most one encryption query per user.*

Instantiating KD. This result proves the multi-challenge, multi-user security of KD, but requires appropriate choices of KEM and SKE. Naturally, one could choose any of the KEMs studied earlier in this work for the first component.[6] For the symmetric encryption we need a scheme which achieves single-challenge, multi-user security. We are not aware of any TAM-tight multi-user analysis of symmetric encryption scheme, so one instead needs to pick a scheme whose multi-user, single challenge security is sufficiently strong against memory-unbounded adversaries. One reasonable option could be GCM with a random nonce. In the ideal cipher model, Hoang, Tessaro, and Thiruvengadam [20] showed a strong bound for this setting which is essentially independent of the number of users.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: DHIES: an encryption scheme based on the Diffie-Hellman problem. Contributions to IEEE P1363a, September 1998

[6] For using the Fujisaki-Okamoto Transformation, which we proved CCA secure, note that $|\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) - \mathsf{Adv}^{\mathsf{mu\text{-}cca}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}})| \leqslant q_{\text{Enc}} \cdot \epsilon$.

2. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45353-9_12
3. Auerbach, B., Cash, D., Fersch, M., Kiltz, E.: Memory-tight reductions. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 101–132. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_4
4. Bauer, B., Fuchsbauer, G., Loss, J.: A classification of computational assumptions in the algebraic group model. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12171, pp. 121–151. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56880-1_5
5. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_18
6. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_25
7. Bhattacharyya, R.: Memory-tight reductions for practical key encapsulation mechanisms. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12110, pp. 249–278. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45374-9_9
8. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. Cryptology ePrint Archive (2001). https://eprint.iacr.org/2001/108
9. Dai, W., Tessaro, S., Zhang, X.: Super-linear time-memory trade-offs for symmetric encryption. In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12552, pp. 335–365. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64381-2_12
10. Diemert, D., Gellert, K., Jager, T., Lyu, L.: Digital signatures with memory-tight security in the multi-challenge setting. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13093, pp. 403–433. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92068-5_14
11. Dinur, I.: On the streaming indistinguishability of a random permutation and a random function. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 433–460. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_15
12. Dinur, I.: Tight time-space lower bounds for finding multiple collision pairs and their applications. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 405–434. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_15
13. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 33–62. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_2
14. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_34
15. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. J. Cryptol. **26**(1), 80–101 (2011). https://doi.org/10.1007/s00145-011-9114-1

16. Gay, R., Hofheinz, D., Kiltz, E., Wee, H.: Tightly CCA-secure encryption without pairings. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 1–27. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_1
17. Ghoshal, A., Ghosal, R., Jaeger, J., Tessaro, S.: Hiding in plain sight: memory-tight proofs via randomness programming. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022. LNCS, vol. 13276, pp. 706–735. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-07085-3_24. https://eprint.iacr.org/2021/1409
18. Ghoshal, A., Jaeger, J., Tessaro, S.: The memory-tightness of authenticated encryption. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12170, pp. 127–156. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_5
19. Ghoshal, A., Tessaro, S.: On the memory-tightness of hashed ElGamal. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 33–62. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_2
20. Hoang, V.T., Tessaro, S., Thiruvengadam, A.: The multi-user security of GCM, revisited: tight bounds for nonce randomization. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018, pp. 1429–1440. ACM Press, October 2018
21. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 341–371. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_12
22. Hofheinz, D., Jager, T.: Tightly secure signatures and public-key encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 590–607. Springer, Heidelberg (2012). https://doi.org/10.1007/s00145-020-09356-x
23. Jaeger, J., Kumar, A.: Memory-tight multi-challenge security of public-key encryption. Cryptology ePrint Archive (2022). https://eprint.iacr.org/2022/
24. Jaeger, J., Tessaro, S.: Tight time-memory trade-offs for symmetric encryption. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 467–497. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_16
25. Libert, B., Joye, M., Yung, M., Peters, T.: Concise multi-challenge CCA-secure encryption and signatures with almost tight security. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 1–21. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_1
26. Maurer, U.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005). https://doi.org/10.1007/11586821_1
27. Okamoto, T., Pointcheval, D.: The gap-problems: a new class of problems for the security of cryptographic schemes. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44586-2_8
28. Shahaf, I., Ordentlich, O., Segev, G.: An information-theoretic proof of the streaming switching lemma for symmetric encryption. In 2020 IEEE International Symposium on Information Theory (ISIT), pp. 858–863 (2020)
29. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_18

30. Tessaro, S., Thiruvengadam, A.: Provable time-memory trade-offs: symmetric cryptography against memory-bounded adversaries. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11239, pp. 3–32. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03807-6_1
31. Wang, Y., Matsuda, T., Hanaoka, G., Tanaka, K.: Memory lower bounds of reductions revisited. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 61–90. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_3