Memory-Tight Multi-Challenge Security of Public-Key Encryption*

Joseph Jaeger and Akshaya Kumar

School of Cybersecurity and Privacy
Georgia Institute of Technology, Atlanta, Georgia, US
{josephjaeger, akshayakumar}@gatech.edu
https://cc.gatech.edu/~josephjaeger/
https://cc.gatech.edu/~akumar805/

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 and/or keys.

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: Provable security, Memory-tightness, Public-key cryptography

Contents

[*] A preliminary version of this paper appeared in the proceedings of ASIACRYPT 2022 [JK22] (© IACR 2022). This is the full version, which corrects several erroneous claims from the earlier version.

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 from other notions of tightness.

Auerbach, Cash, Fersch, and Kiltz [ACFK17] argued that the memory usage of an attacker can be crucial in determining its likelihood of success. This kicked of a line of works [WMHT18, TT18, JT19, Din20b, Din20a, Bha20, GT20, GJT20, DTZ20, SOS20, DGJL21, GGJT22] 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 and/or keys with random strings that are included in hash function calls.

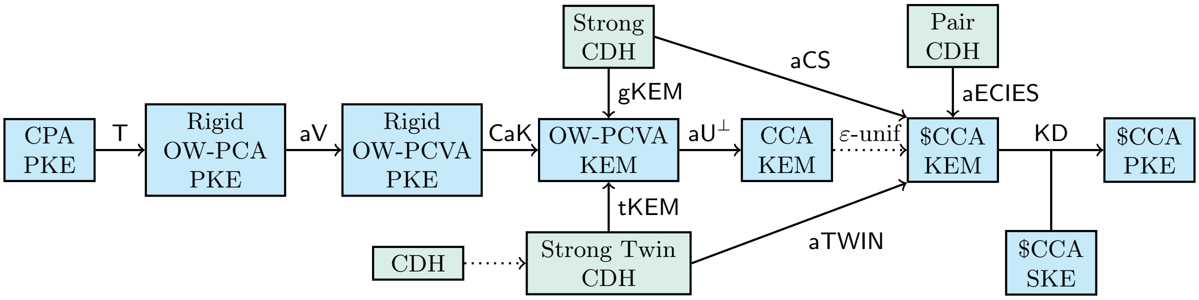
Multi-challenge Setting. As mentioned, our focus in this work is on multi-challenge and multi-user security. This is simply motivated by the fact that encryption schemes get deployed across many different users each of whom will encrypt many messages, so it is important to understand how the security 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 [HJ12, LJYP14, GHKW16] have looked at this in the non-memory-aware setting, providing schemes with TA-tight proofs of CCA 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 this 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. [ACFK17] 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 [Bha20] and Ghoshal and Tessaro [GT20] 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 [ABR01] 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 efficient pairing. However, for efficiency it is preferable to implement schemes using groups for which efficient pairings are believed not to exist.

[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-aware setting [GJT20, GGJT22].



Scheme/Transform	Assumption	Result	Augmentation
Augmented Hashed ElGamal (aECIES)	Pair CDH	$CCA KEM	c
Augmented Cramer-Shoup ElGamal (aCS)	Strong CDH	$CCA KEM	ek, c
Augmented Twin ElGamal (aTWIN)	CDH	$CCA KEM	ek, c
Transformation T	CPA PKE	Rigid OW-PCA PKE	-
Transformation aV	Rigid OW-PCA PKE	Rigid OW-PCVA PKE	ek
Transformation aU$^\perp$	Rigid OW-PCVA KEM or PKE	CCA KEM	ek, c
KEM/DEM (KD)	$CCA KEM and SKE	$CCA PKE	-

Fig. 1. TAM-tight reductions we provide. Applying transformations T, aV, and aU$^\perp$ in sequence gives a "Fujisaki-Okomoto-style" construction of a CCA secure KEM. Results are for multi-user, multi-challenge security. $CCA security requires ciphertexts be indistinguishable from random ciphertext. Augmentation ek means a random string is added to each encryption key and augmentation c means a random string is added to each ciphertext. These random strings are included in every relevant random oracle query.

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 [GT20, 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. The results in [GT20] complement ours in the sense that any reduction from Pair CDH to Strong CDH is likely to incur the memory lower bound presented in [GT20]

1.1 Our Results

We summarize our results in Fig. 1. In short, we obtain three constructions of $CCA secure KEMs from CDH assumptions (aECIES, aCS, and aTWIN) as well as one construction of a CCA secure KEM from a CPA secure public encryption scheme (by applying T, aV, and aU$^\perp$).

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 [ABR98] and the Cramer-Shoup [CS03] variants. Additionally we will look at a "Twin ElGamal" construction by applying the ideas of Bhattacharyya to a construction of Cash, Kiltz, and Shoup [CKS09]. The negative results of Ghoshal and Tessaro apply only to the ECIES variant.

First we discuss ECIES and Cramer-Shoup. In both, the decapsulation 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

[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.

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 $(a, \mathbf{g}^y)$. For the Cramer-Shoup version, the keys are also augmented and are of the form $(\alpha, x), (\alpha, \mathbf{g}^x)$. Here, a and α are uniformly random bitstrings used as additional input(s) to the hash function. We model H as a random oracle.

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 gives X as the encapsulation key, Y as the challenge ciphertext, simulates random oracle and decapsulation queries, and checks if any of the random oracle queries are X^y in which case it returns that and wins its game. 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 $C = X^{x_u \cdot y \cdot y_i}$, we have $C^{1/(x_u \cdot y_i)} = X^y$. 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 [GGJT22], 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.

Cramer-Shoup and Twin Constructions. For the (unaugmented) Cramer-Shoup construction hash queries have the form $H(Y, Z)$. Bhattacharyya's main technique for TAM-tightness with this construction was to simulate the random oracle as $h(Y, \mathbf{g})$ if $Z = X^{\mathrm{dlog}\, Y}$ and as $h(Y, Z)$ otherwise. Here h is random function. The reduction adversary uses its Strong DH oracle to check if $Z = X^{\mathrm{dlog}\, Y}$ holds.

There are two challenges in extending this to the multi-challenge, multi-user setting. The first is analogous to the discussion above, that we want to be able to identify challenge queries forwarded to decryption. As before, we use the auxiliary string included in each ciphertext for this. Namely, we select a as an "encryption" of i for the i-th challenge.

The second challenge is that for the multi-user setting there are multiple encapsulation keys X_u – one for each user. Thus, when simulating the random oracle we want to check whether $Z = X_u^{\mathrm{dlog}\, Y}$ holds which is ill-defined because its not clear which u should be used. This is where the second augmentation comes in, namely the random string α included with each users' encapsulation key. It acts as a domain separator between each user's queries to the random oracle. Moreover, in our proof our reduction adversary will sample α as an "encryption" of u so that u can be recovered efficiently when simulating random oracle queries.

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

In the Twin ElGamal construction, public keys are a pair $(X = \mathbf{g}^x, W = \mathbf{g}^w)$ and the hash computed in encapsulation is $H(\mathbf{g}^y, X^y, W^y)$. By augmenting this in the same way as the Cramer-Shoup we can use basically the same proof, replacing Strong CDH with Strong Twin CDH. Note that Strong Twin CDH was shown to be equivalent to plain CDH by Cash, Kiltz, and Shoup [CKS09].

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 (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 [Sho97, Mau05, FKL18]). 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 style transformations which start with a CPA secure public key encryption scheme and apply several random oracle based transformations to construct a CCA secure KEM. Hofheinz, Hövelmanns, and Kiltz [HHK17] gave a nice modular approach for proving the security of several variants of this transformation. Bhattacharyya showed how to make some of 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, focusing on one particular sequence of transformations.

For two of the transforms, we need to consider augmented versions in which random strings are added to the keys and to each ciphertext and incorporated into the hash queries. As with our CDH-based constructions, our reduction samples these strings as the encryption of the pertinent information it would need to identify challenge ciphertexts and responds 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 [GGJT22], 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.

1.2 Changes From Proceedings Version

A version of this article previously appeared at Asiacrypt 2022 [JK22]. This is a major revision of that article. That version omitted the proof of most results. While preparing this article, we found bugs in some of the omitted proofs and needed to change our constructions to fix these bugs. In the process of correcting these issues, we decided to improve and extend some of the results being proven.

Fixing Erroneous Claims. Theorems 2 and 5 in [JK22] are incorrect. They claim to provide TAM-tight security results in the multi-user, multi-challenge setting for versions of the Cramer-Shoup and $\mathsf{U}^{\perp}$ KEMs that augment the ciphertexts, *but not keys* with random strings. We originally wrote single-user, multi-challenge proofs for these constructions. When attempting to extend them to the multi-user setting we introduced subtle flaws to the proofs which we unfortunately missed. The current article fixes these issues by defining schemes aCS (Sec. 4.2) and $\mathsf{aU}^{\perp}$ (Sec. 5) to augment the keys with random strings as well. Single-user, multi-challenge security proofs for the schemes as defined in [JK22] are captured as a special case of the proofs in this version by setting this key augmentation string to have length 0.

Improved/Extended Results. In the process of fixing the incorrect proofs we decided to strengthen our results. First, we now consider a stronger version of security for encryption/encapsulation schemes that allows attackers to repeat prior queries to the challenge oracle and be given the same ciphertext from the previous time that query was made. When not considering memory, this change makes no difference as an attacker can simply remember all prior queries. The attacker cannot do so in the memory-aware setting, so requiring the game to store this information for the attacker strengthens the security definition. Second, we add analysis of the Twin ElGamal construction aTWIN which was not present in the prior version. Third, we also recast $\mathsf{aU}^{\perp}$ as a transformation applied to a KEM, rather than a PKE scheme. This allows us to observe that our security results for aCS and aTWIN can actually be viewed as specific instances of the security result for $\mathsf{aU}^{\perp}$. By considering the canonical way of constructing a KEM from a PKE scheme (namely, picking the input message at random and considering that to be the encapsulated key), we still capture the application of $\mathsf{aU}^{\perp}$ to PKE schemes. Finally, we now give two security theorems for the KEM/DEM construction, one that achieves the standard version of \$CCA security where the adversary is not allowed to repeat queries to the challenge oracle and the other where the adversary is allowed to repeat queries to challenge oracle. The proof of the latter is more involved and has a loss in concrete security, requiring longer ciphertexts.

2 Preliminaries

2.1 Notation

We recall basic notation and simple lemmas we will use in our paper.

Pseudocode. For our proofs, we use the code based framework of [BR06]. 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, $x \leftarrow\!\!\$\ S$ denotes sampling x uniformly from S, and $S_\star$ denotes $S \cup \{\star\}$. We use the symbol $\perp$ to indicate rejection. Sets are assumed not to include $\star$ or $\perp$ when not specified otherwise. When not specified, tables are initialized empty, integers are initialized to 0, and booleans are initialized to false. Code of the form $(x_{(\cdot)}, y_{(\cdot)}, \ldots) \leftarrow\!\!\$\ X$ initializes $x, y, \ldots$ as "lazily sampled" tables where $x_u, y_u, \ldots$ are sampled by $(x_u, y_u, \ldots) \leftarrow\!\!\$\ X$ whenever one of them is first accessed. The code $\mathsf{OUTPUT}(x_1, x_2, \ldots)$ indicates that an adversary halts immediately with the output $(x_1, x_2, \ldots)$.

Security notions are defined with games such as the one in Fig. 3. The probability that the game G outputs 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 [ACFK17], 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 $|\Pr[\mathcal{A}^f \Rightarrow 1] - \Pr[\mathcal{A}^g \Rightarrow 1]| \leqslant 0.5 \cdot q(q-1)/N \leqslant 0.5 \cdot q^2/N$, where the probability is taken over the randomness of $\mathcal{A}$, sampling $f \leftarrow\$ \mathsf{Fcs}(T, D, R)$, and sampling $g \leftarrow\$ \mathsf{Inj}(T, D, R)$.*

Simple Probability Bounding. At various points we will bound a probability by some fraction $(x - n)/(y - n)$. The following observation allow us a simple method for simplifying such fractions.

Lemma 2. *$(x - n)/(y - n) \leqslant x/y$ whenever $y \geqslant x$ and $y > n \geqslant 0$.*

The second condition, $y > n \geqslant 0$, will always hold for us and if the first, $y \geqslant x$, is false then x/y is a vacuous bound for a probability anyway. As such, we will apply the bound $(x - n)/(y - n) \leqslant x/y$ as if it were unconditional in our proofs.

2.2 Memory-tightness background

$\mathcal{F}$-oracle Adversaries. We adopt GGJT's [GGJT22] 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 [GGJT22], we present theorems in terms of the *reduced complexity* of an oracle aided adversary which is defined as

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

We refer readers to Lemma 2 of [GGJT22] 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 [GGJT22] 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.

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 $\mathcal{H}$.

Message Encoding Techniques. The message encoding technique proposed by GGJT in [GGJT22] 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 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 [Bha20], 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 ($\mathsf{PKE.K}$) that returns a pair of keys (ek, dk) where ek is the encryption key and dk is the corresponding decryption key, the encryption algorithm ($\mathsf{PKE.E}$) that takes the encryption key ek and a message m and returns ciphertext c, and the decryption algorithm ($\mathsf{PKE.D}$) 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) $\mathcal{D}$,

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

if 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 $\mathsf{PKE.E}$. When not stated otherwise, schemes are assumed to be perfectly correct.

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 $c \in \mathsf{PKE}.\mathcal{C}(ek, n)$. 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. We let $\mathsf{PKE}.\mathcal{R}$ denote the set from which $\mathsf{PKE.E}$ draws its randomness.

Suppose $\mathsf{PKE.E}$ is deterministic. Then we say PKE is *rigid* if $\mathsf{PKE.D}^{\mathcal{H}}(dk, c) = m \neq \perp$ implies that $c = \mathsf{PKE.E}^{\mathcal{H}}(ek, m)$ for any (ek, dk) output by $\mathsf{PKE.K}$. Bernstein and Persichetti [BP18] introduced rigidity, noting that (at the time) some results of [HHK17] analyzing Fujisaki-Okamoto-style transforms were incorrect

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

for having omitted rigidity as an assumption.[5] Bhattacharyya [Bha20] replaced rigidity with a property that for every message there exists at most one ciphertext that decrypts to it, which they fold into their definition of deterministic. In proofs, it can be combined with correctness to give a form of rigidity.[6] As we are not aware of any schemes of interest that do not achieve perfect rigidity, but do achieve this "uniqueness" property we stick with rigidity in our proofs.

Indistinguishable From Random Security. We consider indistinguishable from random, chosen ciphertext attack ($CCA) security as captured by Fig. 3. The definition is multi-user and multi-challenge (allowing multiple challenges per user). It requires that 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 (drawn from some set $\mathcal{U}$) by querying $\textsc{New}(u)$. It makes an encryption query $\textsc{Enc}(u, i, m)$ to receive a challenge encryption of m by u and a decryption query $\textsc{Dec}(u, c)$ to have u decrypt c. Here i is a "challenge identifier" drawn from some set $\mathcal{I}$. The attacker can repeat a query $\textsc{Enc}(u, i, m)$ to receive the same ciphertext that it was given the first time. At the end of this section, we discuss some subtleties regarding the choice of $\mathcal{U}$ that are unique to the memory-aware setting and our decision to allow repeated queries (Sec. 2.6).

The adversary needs to distinguish between the real world ($b = 1$) in which a query to $\textsc{Enc}(u, i, m)$ 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 [GT20, GGJT22] considered other ways a decryption oracle might respond to challenge ciphertexts and argued that this is the "correct" convention. The advantage of an adversary $\mathcal{A}$ is $\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{PKE}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{PKE},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{PKE},0}(\mathcal{A})]$.

CCA security is captured by the game $\mathsf{G}^{\mathsf{mu\text{-}cca}}_{\mathsf{PKE},b}$ which is defined the same, except c_0 is the encryption of the all zeros string and CPA security is captured by restricting the CCA security adversary to not query its decryption oracle. Their advantage functions $\mathsf{Adv}^{\mathsf{mu\text{-}cca}}$ and $\mathsf{Adv}^{\mathsf{mu\text{-}cpa}}$ are defined as above.

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 [BBM00] who provided a definition for CPA secure public-key encryption.

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 $(c, K) \in [\mathsf{KEM.E}^{\mathcal{H}}(ek)]$. The weaker notion of δ-correctness requires that,

$$\Pr[\mathsf{KEM.D}^{\mathcal{H}}(dk, c) \neq K \;:\; (c, K) \leftarrow\!\!\$\; \mathsf{KEM.E}^{\mathcal{H}}(ek)] \leqslant \delta.$$

The probability is over $(ek, dk) \leftarrow\!\!\$\; \mathsf{KEM.K}$, $\mathcal{H} \leftarrow\!\!\$\; \mathsf{KEM.IM}$, and the coins of $\mathsf{KEM.E}$. When not stated otherwise, schemes are assumed to be perfectly correct.

[5] The proofs in the ePrint version of [HHK17] have been updated to account for these bugs. Some proofs are corrected further in [Höv21, Sec. 2.1-2.2] which adds missing advantage terms due to imperfect correctness of schemes. The proofs of their Theorems 2.1.5 and 2.1.7 technically miscount the number of random oracle queries, but the proofs can be written so this does not affect the bound.

[6] Bhattacharyya's definition originally required exactly one ciphertext rather than at most one, but with that notion the schemes which they claim achieve this property do not necessarily do so.

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

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

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

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

Game $\mathsf{G}^{\mathsf{mu\text{-}cca}}_{\mathsf{PKE},b}(\mathcal{A})$
Same except:
$c_0 \leftarrow\!\!\$\ \mathsf{PKE.E}^{\mathcal{H}}(ek_u, 0^{|m|})$
```

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

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

$\textsc{Encap}(u, i)$
If $C[u, i] \neq \perp$:
    $c \leftarrow C[u, i]$
    Return $(c, T[u, c])$
$(c_1, K_1) \leftarrow\!\!\$\ \mathsf{KEM.E}^{\mathcal{H}}(ek_u)$
$c_0 \leftarrow\!\!\$\ \mathsf{KEM}.\mathcal{C}(ek_u)$
$K_0 \leftarrow\!\!\$\ \mathsf{KEM}.\mathcal{K}$
$T[u, c_b] \leftarrow K_b$
$C[u, i] \leftarrow c_b$
Return $(c_b, K_b)$

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

Game $\mathsf{G}^{\mathsf{mu\text{-}cca}}_{\mathsf{KEM},b}(\mathcal{A})$
Same except:
$c_0 \leftarrow c_1$
```

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

$\textsc{Enc}(u, i, m)$
If $C[u, i, m] \neq \perp$:
    Return $C[u, i, m]$
$c_1 \leftarrow\!\!\$\ \mathsf{SKE.E}^{\mathcal{H}}(K_u, m)$
$c_0 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SKE.cl}(|m|)}$
$M[u, c_b] \leftarrow m$
$C[u, i, m] \leftarrow c_b$
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 $m$

Game $\mathsf{G}^{\mathsf{mu\text{-}cca}}_{\mathsf{KEM},b}(\mathcal{A})$
Same except:
$c_0 \leftarrow\!\!\$\ \mathsf{SKE.E}^{\mathcal{H}}(K_u, 0^{|m|})$
```

```
Game $\mathsf{G}^{\mathsf{mu\text{-}ow\text{-}}w}_{\Pi}(\mathcal{A})$
$\mathcal{H} \leftarrow\!\!\$\ \Pi.\mathsf{IM}$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\!\!\$\ \Pi.\mathsf{K}$
$\mathrm{O} \leftarrow \perp$ //$w = \mathsf{pasa}$
$\mathrm{O} \leftarrow \mathrm{PC}$ //$w = \mathsf{pca}$
$\mathrm{O} \leftarrow (\mathrm{PC}, \mathrm{CV})$ //$w = \mathsf{pcva}$
$(m, u, i) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{New},\textsc{Chal},\mathrm{O},\mathcal{H}}$
Return $\mathrm{PC}(u, m, \textsc{Chal}(u, i))$

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

$\textsc{Chal}(u, i)$
If $C[u, i] \neq \perp$:
    Return $C[u, i]$
$K \leftarrow\!\!\$\ \Pi.\mathcal{M}$ //$\Pi = \mathsf{PKE}$
$c \leftarrow\!\!\$\ \Pi.\mathsf{E}^{\mathcal{H}}(ek_u, K)$ //$\Pi = \mathsf{PKE}$
$(c, K) \leftarrow\!\!\$\ \Pi.\mathsf{E}^{\mathcal{H}}(ek_u)$ //$\Pi = \mathsf{KEM}$
$C[u, i] \leftarrow c$
Return $c$

$\mathrm{PC}(u, K, c)$
$K' \leftarrow \Pi.\mathsf{D}^{\mathcal{H}}(dk_u, c)$
Return $(K = K')$

$\mathrm{CV}(u, c)$
$K' \leftarrow \Pi.\mathsf{D}^{\mathcal{H}}(dk_u, c)$
Return $(K' \neq \perp)$
```

Fig. 3. (**Left Three:**) Games defining multi-user, multi-challenge \$CCA security of a public key encryption scheme PKE, a key encapsulation mechanism KEM, or a symmetric encryption scheme SKE. The bottom shows how to change the sampling of ciphertext c_0 for CCA security. For \$CPA or CPA security the adversary is not given access to decryption/decapsulation. (**Far Right:**) Games defining multi-user, multi-challenge one-wayness security of PKE or KEM. The parameter $w \in \{\mathsf{pasa}, \mathsf{pca}, \mathsf{pcva}\}$ determines which oracles the attacker is given.

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 from which $\mathsf{KEM.K}$ 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

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

Note that the key produced by $\mathsf{KEM.E}$ is not being given to $\mathcal{D}$.

We define the min-entropy of KEM to be the largest real value $\mathsf{H}_\infty(\mathsf{KEM})$ which satisfies

$$\Pr[c = c^* : (c, \cdot) \leftarrow\!\!\$\ \mathsf{KEM.E}^{\mathcal{H}}(ek)] \leqslant 2^{-\mathsf{H}_\infty(\mathsf{KEM})}$$

for all $ek \in \mathsf{KEM.Ek}$, $\mathcal{H} \in \mathsf{KEM.IM}$, and $c^* \in \mathsf{KEM}.\mathcal{C}$.

Indistinguishable From Random Security. Our notion of \$CCA security for KEMs is presented in Fig. 3, 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 $\mathsf{KEM.E}$ and the ideal world ($b = 0$) where it returns a pair (c, K) chosen uniformly at random from $\mathsf{KEM}.\mathcal{C}(ek) \times \mathsf{KEM}.\mathcal{K}$. As with the PKE games, users are identified by $u \in \mathcal{U}$ and encapsulation queries use $i \in \mathcal{I}$ to allow repeated queries.

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 $\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM},0}(\mathcal{A})]$. We also define CCA security (via $\mathsf{Adv}^{\mathsf{mu\text{-}cca}}_{\mathsf{KEM}}$ and $\mathsf{G}^{\mathsf{mu\text{-}cca}}_{\mathsf{KEM},b}$) analogously to \$CCA security, except in the Encap oracle c_0 is set to equal c_1 rather than being sampled at random.

We note that $|\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) - \mathsf{Adv}^{\mathsf{mu\text{-}cca}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}})| \leqslant q_{\text{Encap}} \cdot \varepsilon$ holds for all ε-uniform KEM and $\mathcal{B}_{\mathsf{KEM}}$ making at most q_{Encap} queries to Encap.

One-Wayness Security. We consider multi-user, multi-challenge variants of the one-wayness security definitions from [HHK17]. The security games are given in Fig. 3. We will use One-Wayness under Plaintext Checking Attacks (OW-PCA) and One-Wayness under Plaintext and Validity Checking Attacks (OW-PCVA), but have also included One-Wayness under Passive attacks for completeness (OW-PASA).[7]

The games are written to use either a KEM or a PKE scheme with messages sampled randomly from $\mathsf{PKE}.\mathcal{M}$. In either case, the adversary is tasked with finding the decryption of a challenge ciphertext. The difference between each variant $w \in \{\mathsf{pasa}, \mathsf{pca}, \mathsf{pcva}\}$ is in the auxilliary oracle(s) O that the adversary is given access to. In the game $\mathsf{G}^{\mathsf{mu\text{-}ow\text{-}pca}}_{\Pi}$, the adversary has access to the Plaintext Checking Oracle PC which takes as input a message-ciphertext pair, and returns true if the message is a valid decryption of the ciphertext and false otherwise. It is implicitly required that messages input to this oracle not be $\perp$. The adversary has access to both oracles, PC and CV, in $\mathsf{G}^{\mathsf{mu\text{-}ow\text{-}pcva}}_{\mathsf{PKE}}$ where CV takes as input a ciphertext, and returns true if the ciphertext decrypts to a valid message.

The attacker's queries specify a particular user $u \in \mathcal{U}$ and its challenge queries additionally specify a challenge identifier $i \in \mathcal{I}$. It may re-query $\text{Chal}(u, i)$ to get back the same ciphertext. For each variant, we let Π denote either a KEM KEM or PKE scheme PKE and we define $\mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}}w}_{\Pi}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{mu\text{-}ow\text{-}}w}_{\Pi}]$.

Using a PKE scheme PKE with a randomly chosen message $m \in \mathsf{PKE}.\mathcal{M}$ can be viewed as a canonical way of constructing a KEM. We denote this by $\mathsf{CaK[PKE]}$ where $\mathsf{CaK[PKE].IM} = \mathsf{PKE.IM}$, $\mathsf{CaK[PKE].K} = \mathsf{PKE.K}$, $\mathsf{CaK[PKE].D} = \mathsf{PKE.D}$, and $\mathsf{CaK[PKE].E}^{\mathcal{H}}(ek)$ samples $K \leftarrow\$ \mathsf{PKE}.\mathcal{M}$; computes $c \leftarrow\$ \mathsf{PKE.E}^{\mathcal{H}}(ek, K)$; and returns (c, K). It is straightforward to see that $\mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}}w}_{\mathsf{PKE}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}}w}_{\mathsf{CaK[PKE]}}(\mathcal{A})$, as the underlying games are identical. If PKE is δ-correct then $\mathsf{CaK[PKE]}$ is $\delta(0)$-correct. The uniformity and min-entropy of $\mathsf{CaK[PKE]}$ depend on how PKE behaves on random messages.

2.5 Symmetric Key Encryption

Syntax. A symmetric key encryption scheme, SKE, consists of three algorithms - the key generation algorithm ($\mathsf{SKE.K}$) that returns a key K, the encryption algorithm ($\mathsf{SKE.E}$) that takes the key K and a message m and returns ciphertext c, and the decryption algorithm $\mathsf{SKE.D}$ 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.

Indistinguishable From Random CCA Security. Our notion of \$CCA security for SKE schemes is captured by Fig. 3, 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. As before $u \in \mathcal{U}$ is used to identify users and $i \in \mathcal{I}$ is used to identify challenges. 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.

[7] The last of these is commonly referred to as OW-CPA. We have chosen to use the "passive" nomenclature of [BP18].

2.6 Modeling Details: Naming Users and Repeating Queries

Naming Users. Throughout this paper we provide multi-user definitions of security. In these definitions, the attacker makes queries about a particular user with oracle queries of the form $\mathrm{O}(u, \dots)$. Here u is acting as an identifier for the user which is drawn from some set $\mathcal{U}$. This raises a natural question: what choices of $\mathcal{U}$ should be allowed? The challenge identifier drawn from set $\mathcal{I}$ can be considered analogously.

In one direction, for proving that some scheme satisfies our security notion is may be convenient to insist that $\mathcal{U} = \{1, \dots, q_u\}$ where q_u is the total number of different users that $\mathcal{A}$ will make queries for. For proving that our security notion suffices for proving the security of some higher-level protocol it may be desirable to allow arbitrary identifiers so that reduction adversaries can reference them with semantically meaningful strings. This motivates the most general choice of $\mathcal{U} = \{0, 1\}^*$.

A priori, it may seem that this is all a moot distinction, as all choices of $\mathcal{U}$ are equivalent as long as they are large enough. Let us consider the natural proof approach for this fact. Suppose $\mathcal{A}$ is a multi-user attacker for some security notion using $\mathcal{U}$. We will construct $\mathcal{A}'$ using $\mathcal{U}'$ such that $\mathsf{Adv}(\mathcal{A}) = \mathsf{Adv}(\mathcal{A}')$. Namely, $\mathcal{U}'$ will adaptively construct a mapping M from $\mathcal{U}$ to $\mathcal{U}'$. Adversary $\mathcal{A}'$ runs $\mathcal{A}$ and whenever the latter makes an oracle query of the form $\mathrm{O}(u, \dots)$ it checks if $M[u]$ is already defined. If not, it defines $M[u]$ to equal some unused $u' \in \mathcal{U}'$. Then it queries $\mathrm{O}(M[u], \dots)$ and returns the result to $\mathcal{A}$. In all of the games we consider, the game is "agnostic" to what u looks like and so $\mathcal{A}'$ will be perfectly simulating the view expecting by $\mathcal{A}$, giving the claimed result.

However, when we consider this from a concrete, memory-aware perspective we run into an issue. The above proof required storing a mapping M of size equal to the number of different users queried by $\mathcal{A}$. To avoid this issue, we can think of hashing the elements of $\mathcal{U}$ into $\mathcal{U}'$. Let $\mathcal{A}'$ be a $\mathsf{Fcs}(\mathcal{U},\mathcal{U}')$-oracle adversary which behaves as above except $M[u]$ is defined to equal $f(u)$ for $f \leftarrow\!\!\$\ \mathsf{Fcs}(\mathcal{U},\mathcal{U}')$. As long as $\mathcal{A}$ never find a collision in f, its simulated view is correct. For typical advantage functions, this gives us $\mathsf{Adv}(\mathcal{A}) \leqslant \mathsf{Adv}(\mathcal{A}') + 0.5q_u^2/|\mathcal{U}'|$.

One can similarly hash down from $\mathcal{I}$ to a smaller $\mathcal{I}'$. As we do not care about collisions "across users" this would add a term $0.5q_u q_i^2/|\mathcal{I}'|$ to the advantage where q_i upperbounds the number of distinct challenges that any one users receives.

Repeating Queries. In our definitions we have made the choice that on repeat queries to encryption or encapsulation the security game will return the same results from the previous query. A more traditional definition would not allow this. This can be captured by restricting attention to "challenge-respecting" adversaries that never repeat (u, i)'s. When considering such adversaries, we can without loss of generality restrict attention to attackers which use a global counter for i which then increment after each query. In a non-memory-aware setting, this change would be without loss of generality as an attacker could simply remember all of its prior queries. In the memory-aware setting we cannot do so.

For the proof of higher-level primitives having the game remember such information for us can be useful. Indeed, we switched to using this definitional style on realizing that the proof Theorem 4 (which proves CCA security of a scheme under the assumption of OW-PCVA security) *requires* the ability to repeat queries in the OW-PCVA game, even if we were using a version of CCA security that did not allow this. Moving backwards, the schemes intended to be used "on the way" to this theorem (those of Theorem 5 and Theorem 6) must also assume security notions where queries can be repeated. For our proof flows based on Diffie-Hellman assumptions, we will explicitly prove that we can achieve such security notions.

We give two different quantitative results for the KEM-DEM paradigm depending on whether or not we restrict our attention to challenge-respecting adversaries.

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 its single-challenge variant. The second is a new definition we introduce, Pair CDH, which gives the adversary oracle access to a pairing from the group

```
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. 4. 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.

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 [Sho97, Mau05, FKL18] 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 $\langle \mathbf{g} \rangle$.

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. 4. 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 [OP01] 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 $\mathsf{G}^{\mathsf{scdh}}$ [ABR01] 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 $f(ab)$ 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 $\mathbb{G}_2 = (\mathbf{g}_2, p, \circ_2)$

```
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. 5. Adversary used for Lemma 3, proving that single-challenge CDH security TAM-tightly implies multi-challenge CDH security.

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 $\circ_2$.

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 picks the values used for rerandomization as the output of a random function, rather than picking them randomly and storing them.

Lemma 3 (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 $\mathsf{Fcs}(\mathcal{U}, \mathbb{Z}_p^*) \times \mathsf{Fcs}(\mathcal{U} \times \mathcal{I}, \mathbb{Z}_p^*)$-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 3). Consider the adversary $\mathcal{B}_{\mathsf{x}}$ shown in Fig. 5. 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 $x_u = xx'_u$, 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 $y_{u,i} = yy'_{u,i}$, 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 $Z^{1/(x'_u y'_{u,i})} = \mathbf{g}^{xy}$.

The claims on the efficiency of $\mathcal{B}_{\mathsf{x}}$ are easy to verify. □

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 3, 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 $\textsc{Gap}(A, B, C) = \mathsf{true}$ if and only if $\textsc{Pair}(A, B) = \textsc{Pair}(\mathbf{g}, C)$. 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 $a = \mathrm{dlog}(A)$ and $b = \mathrm{dlog}(B)$, 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 how 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 $x = \mathrm{dlog}(X)$ and $y = \mathrm{dlog}(Y)$. 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 [BFL20], we can use the ability to find such "colliding" polynomials to solve the discrete log problem. We provide details of this analysis in Appendix A.

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 [ACFK17]. Namely, when 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 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 [Sho97, Mau05] 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.

3.4 (Strong) Twin CDH Assumption

Now we recall the Strong Twin CDH Assumption and extend the proof of Cash, Kiltz, and Shoup [CKS09] to show in the multi-user, multi-challenge setting it is TAM-tightly implied by CDH.

Consider the games shown in Fig. 6. Therein, the attack can make $\textsc{New}(u)$ queries to receive group elements W_u, X_u and $\textsc{Chal}(u, i)$ queries to receive $Y_{u,i} = \mathbf{g}^{y_{u,i}}$. The attacks goal is to compute $W_u^{y_{u,i}}, X_u^{y_{u,i}}$ for some u, i. The basic notion of Twin CDH security is captured by the game $\mathsf{G}^{\mathsf{tcdh}}$ in which the attacker has no additional oracle. The Strong Twin CDH security notion is capture by $\mathsf{G}^{\mathsf{stcdh}}$ in which the attacker's

```
Game $\mathsf{G}^{\mathsf{x}}_{\mathbb{G}}(\mathcal{A})$
$(\mathbf{g}, p, \circ) \leftarrow \mathbb{G}$
$w_{(\cdot)} \leftarrow_{\$} \mathbb{Z}_p^*$; $x_{(\cdot)} \leftarrow_{\$} \mathbb{Z}_p^*$
$y_{(\cdot,\cdot)} \leftarrow_{\$} \mathbb{Z}_p^*$
$\mathrm{O} \leftarrow \perp$  $//\mathsf{G}^{\mathsf{tcdh}}$
$\mathrm{O} \leftarrow \textsc{Strong}$  $//\mathsf{G}^{\mathsf{stcdh}}$
$(u, i, Z, \Psi) \leftarrow_{\$} \mathcal{A}^{\textsc{New},\textsc{Chal},\mathrm{O}}$
Return $(Z = \mathbf{g}^{w_u y_{u,i}})$ and $(\Psi = \mathbf{g}^{x_u y_{u,i}})$

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

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

$\textsc{Strong}(u, Y, Z, \Psi)$
Return $(Z = Y^{w_u})$ and $(\Psi = Y^{x_u})$
```

Fig. 6. Security games capturing two variants of the twin computational Diffie-Hellman problem, namely Twin CDH and Strong Twin CDH.

oracle Strong is a natural extension of the oracle from Strong CDH. Given (u, Y, Z, Ψ), the oracle tells the attacker whether (W_u, Y, Z) and (X_u, Y, Ψ) are *both* valid CDH tuples.

For $\mathsf{x} \in \{\mathsf{tcdh}, \mathsf{stcdh}\}$ we define $\mathsf{Adv}^{\mathsf{x}}_{\mathbb{G}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{x}}_{\mathbb{G}}(\mathcal{A})]$. We assume that user identifiers, u, are drawn from some fixed set $\mathcal{U}$ and challenge identifiers, i, are drawn from a fixed set $\mathcal{I}$.

CDH ⇒ Strong Twin CDH. Next we show that CDH security TAM-tightly implies Strong Twin CDH security. This is a relatively straightforward extension of Theorem 3 from [CKS09] to cover the multi-user, multi-challenge setting and to take memory-tightness into account.[8]

Lemma 4 (CDH ⇒ Strong Twin CDH). *Let $\mathbb{G}$ be a group and $\mathcal{A}_{\mathsf{stcdh}}$ be an adversary for $\mathsf{G}^{\mathsf{stcdh}}_{\mathbb{G}}$ with $(q_{\textsc{New}}, q_{\textsc{Chal}}, q_{\mathrm{O}}) = \mathbf{Query}(\mathcal{A})$. Then we can construct a $\mathsf{Fcs}(\mathcal{U}, \mathbb{Z}_p) \times \mathsf{Fcs}(\mathcal{U}, \mathbb{Z}_p)$-oracle adversary $\mathcal{B}_{\mathsf{cdh}}$ (given in the proof) such that*

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{stcdh}}_{\mathbb{G}}(\mathcal{A}_{\mathsf{stcdh}}) &\leqslant \mathsf{Adv}^{\mathsf{cdh}}_{\mathbb{G}}(\mathcal{B}_{\mathsf{cdh}}) + (q_{\textsc{New}} + 2q_{\mathrm{O}} + 1)/p\\
\mathbf{Query}(\mathcal{B}_{\mathsf{cdh}}) &= (q_{\textsc{New}}, q_{\textsc{Chal}}, 0)\\
\mathbf{Time}^*(\mathcal{B}_{\mathsf{cdh}}) &= \mathbf{Time}(\mathcal{A}_{\mathsf{stcdh}}) + O(q_{\textsc{New}} + q_{\mathrm{O}})\mathbf{Time}(\mathbb{G})\\
\mathbf{Mem}^*(\mathcal{B}_{\mathsf{cdh}}) &= \mathbf{Mem}(\mathcal{A}_{\mathsf{stcdh}}) + O(\mathbf{Mem}(\mathbb{G})).
\end{aligned}$$

The version of this theorem given in [CKS09] erroneously has an additional claim regarding the conditional probability that (its version of) $\mathcal{B}_{\mathsf{cdh}}$ succeeds given that it has not set the flag false.

The following lemma (taken from [CKS09] with minor modifications), gives a "trapdoor" method for simulating the Strong Twin CDH Strong oracle for an algorithm allowed to pick the value of X. We will implicitly use this in our proof of Lemma 4. To be self-contained we embed a proof of Lemma 5 into our proof, rather than making explicit use of it.

Lemma 5 (Trapdoor Test, Thm. 2 from [CKS09]). *Let $\mathbb{G} = (\mathbf{g}, p, \circ)$ be a group. Fix $W \in \langle \mathbf{g} \rangle$ and suppose X, r, s are sampled by $r \leftarrow_{\$} \mathbb{Z}_p$, $s \leftarrow_{\$} \mathbb{Z}_p$, $X \leftarrow \mathbf{g}^s W^{-r}$. Further suppose that Y, Z, Ψ are random variables taking values in $\langle \mathbf{g} \rangle$, which are defined as some function of W, X. Then we have:*

(i) X is uniformly distributed in $\langle \mathbf{g} \rangle$;
(ii) W and X are independent;
(iii) If $w = \mathrm{dlog}(W)$ and $x = \mathrm{dlog}(X)$, then the probability that the truth value of $Z^r \Psi = Y^s$ does not agree with the truth value of $Z = Y^w \wedge \Psi = Y^x$ is at most $1/p$. Moreover, if the latter is true, then the former necessarily is.

[8] We could alternatively have used the single challenge version of this result from CKS and then given a variation on Lemma 3 to show the single challenge version of Strong Twin CDH implies the multi-challenge version.

```
Adversary $\mathcal{B}_{\mathsf{cdh}}^{\textsc{New},\textsc{Chal},\mathrm{O}}$
$(\mathbf{g}, p, \circ) \leftarrow \mathbb{G}$
$g \leftarrow\$ \mathsf{Fcs}(\mathcal{U}, \mathbb{Z}_p)$
$h \leftarrow\$ \mathsf{Fcs}(\mathcal{U}, \mathbb{Z}_p)$
$(u, i, Z, \Psi) \leftarrow\$ \mathcal{A}_{\mathsf{stcdh}}^{\textsc{SimNew},\textsc{Chal},\textsc{SimStrong}}$
$r_u \leftarrow g(u);\ s_u \leftarrow h(u)$
If $(Z^{r_u}\Psi \neq Y^{s_u})$
    $\mathsf{fail} \leftarrow \mathsf{true}$
Return $(u, i, Z)$
```

```
$\textsc{SimNew}(u)$
$W_u \leftarrow \textsc{New}(u)$
$r_u \leftarrow g(u)$
$s_u \leftarrow h(u)$
$X_u \leftarrow \mathbf{g}^{s_u} W^{-r_u}$
Return $(W_u, X_u)$
```

```
$\textsc{SimStrong}(u, Y, Z, \Psi)$
$r_u \leftarrow g(u)$
$s_u \leftarrow h(u)$
Return $(Z^{r_u}\Psi = Y^{s_u})$
```

```
Hybrids $\mathsf{H}_\kappa$ for $0 \leqslant \kappa \leqslant 3$
$(\mathbf{g}, p, \circ) \leftarrow \mathbb{G}$
$w_{(\cdot)} \leftarrow\$ \mathbb{Z}_p^*;\ y_{(\cdot,\cdot)} \leftarrow\$ \mathbb{Z}_p^*$
$x_{(\cdot)} \leftarrow\$ \mathbb{Z}_p$ $//\mathsf{H}_{[0,3)}$
$r_{(\cdot)} \leftarrow\$ \mathbb{Z}_p$ $//\mathsf{H}_{[1,3)}$
$s_{(\cdot)} \leftarrow x_{(\cdot)} + r_{(\cdot)} \cdot w_{(\cdot)}$ $//\mathsf{H}_{[1,3)}$
$s_{(\cdot)} \leftarrow\$ \mathbb{Z}_p;\ r_{(\cdot)} \leftarrow\$ \mathbb{Z}_p$ $//\mathsf{H}_{[3,\infty)}$
$x_{(\cdot)} \leftarrow s_{(\cdot)} - r_{(\cdot)} \cdot w_{(\cdot)}$ $//\mathsf{H}_{[3,\infty)}$
$(u, i, Z, \Psi) \leftarrow\$ \mathcal{A}^{\textsc{New},\textsc{Chal},\textsc{Strong}}$
Return $(Z = \mathbf{g}^{w_u y_{u,i}})$ and $(\Psi = \mathbf{g}^{x_u y_{u,i}})$
```

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

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

$\textsc{Strong}(u, Y, Z, \Psi)$
$\mathsf{bool} \leftarrow ((Z = Y^{w_u})$ and $(\Psi = Y^{x_u}))$
Return $\mathsf{bool}$ $//\mathsf{H}_{[0,1)}$
If $\mathsf{bool} \neq (Z^{r_u}\Psi = Y^{s_u})$: $//\mathsf{H}_{[1,\infty)}$
    $\mathsf{bad} \leftarrow \mathsf{true}$ $//\mathsf{H}_{[1,\infty)}$
    Return $\mathsf{bool}$ $//\mathsf{H}_{[1,2)}$
Return $(Z^{r_u}\Psi = Y^{s_u})$ $//\mathsf{H}_{[1,\infty)}$
```

Fig. 7. Adversary and hybrids used for proof of Lemma 4.

Proof (of Lemma 4). Consider the adversary $\mathcal{B}_{\mathsf{cdh}}$ shown in Fig. 7. It runs $\mathcal{A}_{\mathsf{stcdh}}$, giving it direct access to Chal and simulating its New and Strong oracles. To simulate New it first queries its own oracle New to obtain W_u and then locally picks its own r_u, s_u, using random functions g, h, from which it derives X_u in a manner mirroring the setup of Lemma 5. Then the trapdoor test from that lemma is used to simulate Strong. Once $\mathcal{A}_{\mathsf{stcdh}}$ outputs (u, i, Z, Ψ), adversary $\mathcal{B}_{\mathsf{cdh}}$ sets the flag fail if Z, Ψ fail the trapdoor test and outputs (u, i, Z) as its final response. We do not make use of fail.

The claims about $\mathcal{B}_{\mathsf{cdh}}$'s efficiency are clear from its code. Adversary $\mathcal{B}_{\mathsf{cdh}}$ perfectly simulates the expected view of $\mathcal{A}_{\mathsf{stcdh}}$ except that X_u is uniform in $\langle \mathbf{g} \rangle$ rather than $\langle \mathbf{g} \rangle \setminus \{1_{\mathbb{G}}\}$ and SimStrong will sometimes return true on inputs for which Strong would have returned false.

Let H_0 be a hybrid game which is identical to $\mathsf{G}_{\mathbb{G}}^{\mathsf{stcdh}}$ except each x_u is sampled from $\mathbb{Z}_p$ rather than $\mathbb{Z}_p^*$, as defined by Fig. 7. Then $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{stcdh}}(\mathcal{A}_{\mathsf{stcdh}}) = \Pr[\mathsf{G}_{\mathbb{G}}^{\mathsf{stcdh}}(\mathcal{A}_{\mathsf{stcdh}})] \leqslant \Pr[\mathsf{H}_0] + (q_{\textsc{New}} + q_{\mathrm{O}} + 1)/p$ where $(q_{\textsc{New}} + q_{\mathrm{O}} + 1)$ is an upper bound on the total number of x_u values used by the game.

Now consider H_1, wherein variables $r_{(\cdot)}$ and $s_{(\cdot)}$ are introduced. In Strong, we attempt to return the boolean $(Z^{r_u}\Psi = Y^{s_u})$ in place of $((Z = Y^{w_u})$ and $(\Psi = Y^{x_u}))$, but set a bad flag and return the latter if these two differ. None of this changes the behavior of the game, so $\Pr[\mathsf{H}_0] = \Pr[\mathsf{H}_1]$.

In H_2, the Strong oracle always returns $(Z^{r_u}\Psi = Y^{s_u})$ and so is identical to H_1 unless bad is set, giving $\Pr[\mathsf{H}_2] \leqslant \Pr[\mathsf{H}_1] + \Pr[\mathsf{H}_1 \text{ sets } \mathsf{bad}]$. Note that the view of $\mathcal{A}$ is independent of $r_{(}\cdot)$ in H_1. We will show that each Strong query has a probability of $1/p$ of setting bad, giving an overall bound of $\Pr[\mathsf{H}_1 \text{ sets } \mathsf{bad}] \leqslant q_{\mathrm{O}}/p$. For a given query let $y = \mathrm{dlog}(Y)$, $z = \mathrm{dlog}(Z)$, and $\psi = \mathrm{dlog}(\psi)$. Then we care about when the boolean (a) $(zr_u + \psi = ys_u)$ differs from (b) $((z = yw_u)$ and $(\psi = yx_u))$, where these are following equations are evaluated mod p. The following calculation establishes that (a) must hold if (b) does

$$zr_u + \psi = yw_ur_u + yx_u = y(w_ur_u + x_u) = ys_u.$$

The following calculation establishes that (a) must not hold if $z = yw_u$ and $\psi \neq yx_u$

$$zr_u + \psi = yw_ur_u + \psi \neq yw_ur_u + yx_u = y(w_ur_u + x_u) = ys_u.$$

Now suppose that (a) holds, but $z \neq yw_u$ or equivalently $z - yw_u \neq 0$. Then $zr_u + \psi = ys_u = y(w_ur_u + x_u)$. Solving this for r_u gives

$$r_u = (yx_u - \psi)/(z - yw_u)$$

which is well defined because $z - yw_u \neq 0$. As the view of $\mathcal{A}$ is independent of r_u we can think of it being sampled after $\mathcal{A}$ makes its query, in which case there is a $1/p$ chance it happens to equal $(yx_u - \psi)/(z - yw_u)$.

Finally, in H_3 we switch from $s_{(\cdot)}$ being defined in terms of $x_{(\cdot)}$ to $s_{(\cdot)}$ being uniform and $x_{(\cdot)}$ defined in terms of it. This is equivalent, so $\Pr[\mathsf{H}_3] = \Pr[\mathsf{H}_2]$. Now the view $\mathcal{B}_{\mathsf{cdh}}$ gives to $\mathcal{A}_{\mathsf{stcdh}}$ is identical to its view in $\Pr[\mathsf{H}_3]$. Note its choice of X_u in SimNew implicitly sets $x_u = \mathrm{dlog}(X_u)$ to have the desired value. If $\mathcal{A}_{\mathsf{stcdh}}$ would win, then $Z = \mathbf{g}^{w_u y_{u,i}}$ and so $\mathcal{B}_{\mathsf{cdh}}$ will win. Hence, $\Pr[\mathsf{H}_3] \leqslant \mathsf{Adv}^{\mathsf{cdh}}_{\mathbb{G}}(\mathcal{B}_{\mathsf{cdh}})$. □

4 Hashed ElGamal KEMs

In this section we present the first example of KEMs with TAM-tight proofs in the multi-user, 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/or keys which are subsequently included in 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 [ABR98] 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 instance of augmented 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 $\mathsf{aECIES}.\mathsf{IM} = \mathsf{Fcs}(\{0,1\}^l \times \mathbb{G}, \mathcal{K})$. Its ciphertext set is defined by $\mathsf{aECIES}.\mathcal{C}(ek) = \{0,1\}^l \times (\langle \mathbf{g} \rangle \setminus \{1_{\mathbb{G}}\})$

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

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

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

Overview of Existing Techniques and Associated Challenges. Bhattacharyya [Bha20] studied ECIES in the memory-aware setting. They pointed out that the technique of simulating random oracles with PRFs introduced in [ACFK17] 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 as $\mathsf{F}(k, Z)$ instead of the random oracle $\mathcal{H}(Z)$, for a decapsulation query Y the reduction needs to return $\mathsf{F}(k, Y^{dk})$ which it cannot compute.[9]

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

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

as $\mathsf{F}(k, e(ek, Y))$ for all non-challenge ciphertexts. The reduction remembers the challenge ciphertext and responds 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 [GGJT22] 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 ⇒ $CCA). *Let* $\mathsf{aECIES} = \mathsf{aECIES}[\mathbb{G}, \mathcal{K}, l]$ *where* $\mathbb{G} = (\mathbf{g}, p, \circ)$ *is a prime order group. 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* $2^l \geqslant |\mathcal{U} \times \mathcal{I}|$*. Then Fig. 9 gives a* $\mathsf{Fcs}(\{0,1\}^l \times \mathbb{Z}_p, \mathcal{K}) \times \mathsf{Inj}^{\pm}(\mathcal{U} \times \mathcal{I}, \{0,1\}^l)$*-oracle adversary* $\mathcal{B}_{\mathsf{pcdh}}$ *such that*

$$\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aECIES}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{pcdh}}_{\mathbb{G}}(\mathcal{B}_{\mathsf{pcdh}}) + \frac{q_{\text{ENCAP}}^2}{2^l} + \frac{2(|\mathcal{I}| + q_{\text{ENCAP}}) \cdot q_{\text{DECAP}}}{2^l(p-1)}$$

$$\mathbf{Query}(\mathcal{B}_{\mathsf{pcdh}}) = ((q_{\text{NEW}} + q_{\text{ENCAP}} + q_{\text{DECAP}} + q_{\text{H}}), (q_{\text{ENCAP}} + q_{\text{DECAP}} + q_{\text{H}}), (q_{\text{ENCAP}} + 2q_{\text{DECAP}} + 2q_{\text{H}}))$$

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

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 $q_{\text{ENCAP}}^2/2^l$ term to dominate the information theoretic part of the bound. Suppose we consider 2^{32} users receiving 2^{32} ciphertexts each (e.g. $|\mathcal{U}| = |\mathcal{I}| \approx 2^{32}$ and $q_{\text{ENCAP}} = q_{\text{DECAP}} \approx 2^{64}$) with a group of size $p \approx 2^{256}$. With a cautious choice of $l = 256$, the size of the ciphertext is not too significantly increased, we get $q_{\text{ENCAP}}^2/2^l \approx 2^{-128}$. If we aim for bounding the information theoretic term to around 2^{-64}, then we can pick $l = 192$.

However, these estimates assumed that $|\mathcal{U}|$ and $|\mathcal{I}|$ were roughly equal to the number of users and ciphertexts per user. As discussed in Sec. 2.6, the *names* for users/challenges might come from a set which is much larger than the total number of users/challenge. At an extreme, one might have $\mathcal{U} = \mathcal{I} = \{0,1\}^*$. To resolve this we can consider the technique described in that section of hashing these identifiers down to smaller sets $\mathcal{U}'$ and $\mathcal{I}'$, which will require larger choices for l. This adds advantage terms $0.5q_u^2/|\mathcal{U}'|$ and $0.5q_u q_i^2/|\mathcal{I}'|$ where q_u and q_i are, respectively, the number of distinct users and encryptions per user. Using the parameter estimates from above (with $q_u = q_i \approx 2^{32}$) we can get a bound around 2^{-64} by picking $|\mathcal{U}'| = 2^{128}$ and $|\mathcal{I}'| = 2^{160}$ or a bound around 2^{-128} by picking $|\mathcal{U}'| = 2^{192}$ and $|\mathcal{I}'| = 2^{244}$. For the security bound, $l = 192$ or $l = 256$ would suffice, however, we additionally require $2^l \geqslant |\mathcal{U} \times \mathcal{I}|$ so that the injection is well-defined. Thus, in this case we would require $l = 288$ or $l = 436$.

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 challenge identifier 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 $\text{PAIR}(\mathbf{g}, \mathbf{g}^{xy}) = \text{PAIR}(\mathbf{g}^x, \mathbf{g}^y)$.

Our adversary simulates the i-th challenge ciphertext for u 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, i). If $v = u$, it re-queries $\text{CHAL}(u, i)$ to obtain the corresponding ciphertext $Y_{u,i}$. If $Y = Y_{u,i}$, the reduction assumes this was a challenge ciphertext. Finally, when the adversary $\mathcal{A}$ queries the oracle H with (a, Z) such that $a^{-1} = (u, i)$ and $\text{PAIR}(\mathbf{g}, Z) = \text{PAIR}(\text{NEW}(u), \text{CHAL}(u, i))$, the reduction outputs Z and wins the Pair CDH game.

Proof (of Theorem 1). We use a sequence of hybrids H^1_0 through H^1_3, H^2_0 through H^2_1, and H^3_0 through H^3_1 presented in Fig. 8 where we establish the following claims that upper bound the advantage of $\mathcal{A}$.

1. $\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aECIES}}(\mathcal{A}) = 2\Pr[\mathsf{H}^1_0] - 1$
2. $\Pr[\mathsf{H}^1_0] \leqslant \Pr[\mathsf{H}^1_3] + q_{\text{ENCAP}}^2/2^{l+1}$
3. $\Pr[\mathsf{H}^1_3] = \Pr[\mathsf{H}^2_1]$
4. $\Pr[\mathsf{H}^2_1] \leqslant \Pr[\mathsf{H}^3_0] + (|\mathcal{I}| + q_{\text{ENCAP}}) \cdot q_{\text{DECAP}}/(2^l \cdot (p-1))$
5. $\Pr[\mathsf{H}^3_0] \leqslant \Pr[\mathsf{H}^3_1] + \Pr[b = 0 \wedge \mathsf{H}^3_1 \text{ sets bad}]$
6. $\Pr[\mathsf{H}^3_1] \leqslant \frac{1}{2}$
7. $\Pr[b = 0 \wedge \mathsf{H}^3_1 \text{ sets bad}] \leqslant \mathsf{Adv}^{\mathsf{pcdh}}_{\mathbb{G}}(\mathcal{B}_{\mathsf{pcdh}})/2$

```
Hybrids $\mathsf{H}^{\iota}_{\kappa}$
$b \leftarrow\!\$ \{0,1\}$
$x_{(\cdot)} \leftarrow\!\$ \mathbb{Z}_p^*$; $X_{(\cdot)} \leftarrow \mathbf{g}^{x(\cdot)}$
$\mathcal{H}_1, \mathcal{H}_0 \leftarrow\!\$ \mathsf{aCS.IM}$
$D_1 \leftarrow \{0,1\}^l \times \mathbb{Z}_p$
$\tilde{\mathcal{H}} \leftarrow\!\$ \mathsf{Fcs}(\{0,1\}, D_1, \mathcal{K})$
$g^{\pm} \leftarrow\!\$ \mathsf{Inj}^{\pm}(\mathcal{U} \times \mathcal{I}, \{0,1\}^l)$
$h \leftarrow\!\$ \mathsf{Fcs}(\mathcal{U} \times \mathcal{I}, \mathbb{Z}_p^*)$
$q \leftarrow\!\$ \mathsf{Inj}(\mathbb{Z}_p, \mathbb{Z}_p)$
$b' \leftarrow\!\$ \mathcal{A}^{\textsc{New},\textsc{Encap},\textsc{Decap},\mathrm{H}}$
Return $(b' = b)$

$\textsc{New}(u)$ //$\mathsf{H}^1, \mathsf{H}^2, \mathsf{H}^3$
Return $X_u$
```

```
$\textsc{Encap}(u,i)$ //$\mathsf{H}^1$
If $C[u,i] \neq \perp$:
    $(a,Y) \leftarrow C[u,i]$
    Return $((a,Y), T[u,a,Y])$
$I[u] \leftarrow I[u] \cup \{i\}$
$a \leftarrow \{0,1\}^l$ //$\mathsf{H}^1_{[0,1)}$
$a \leftarrow g(u,i)$ //$\mathsf{H}^1_{[1,\infty)}$
$y \leftarrow\!\$ \mathbb{Z}_p^*$ //$\mathsf{H}^1_{[0,1)}$
$y \leftarrow h(u,i)$ //$\mathsf{H}^1_{[1,\infty)}$
$Y \leftarrow \mathbf{g}^y$; $Z \leftarrow X_u^y$
$K^1 \leftarrow \mathcal{H}_1(a,Z)$
$K^0 \leftarrow\!\$ \mathcal{K}$ //$\mathsf{H}^1_{[0,2)}$
$K^0 \leftarrow \mathcal{H}_0(a,Z)$ //$\mathsf{H}^1_{[2,\infty)}$
$T[u,a,Y] \leftarrow K^b$
$C[u,i] \leftarrow (a,Y)$
Return $((a,Y), K^b)$
```

```
$\textsc{Decap}(u,a,Y)$ //$\mathsf{H}^1$
$(v,i) \leftarrow g^{-1}(a)$ //$\mathsf{H}^1_{[3,\infty)}$
If $T[u,a,Y] \neq \perp$ //$\mathsf{H}^1_{[0,3)}$
If $v = u$ and $i \in I[u]$ and $(a,Y) = C[u,i]$: //$\mathsf{H}^1_{[3,\infty)}$
    Return $T[u,a,Y]$
$Z \leftarrow Y^{x_u}$
$K \leftarrow \mathcal{H}_1(a,Z)$
Return $K$

$\mathrm{H}(a,Z)$ //$\mathsf{H}^1$
Return $\mathcal{H}_1(a,Z)$
```

```
$\textsc{Encap}(u,i)$ //$\mathsf{H}^2$
If $C[u,i] \neq \perp$:
    $(a,Y) \leftarrow C[u,i]$
    Return $((a,Y), T[u,a,Y])$
$I[u] \leftarrow I[u] \cup \{i\}$
$a \leftarrow g(u,i)$
$y \leftarrow h(u,i)$
$Y \leftarrow \mathbf{g}^y$; $Z \leftarrow X_u^y$
$K^b \leftarrow \tilde{\mathcal{H}}_b(a, \textsc{Pair}(\mathbf{g}, Z))$ //$\mathsf{H}^2_{[0,1)}$
$K^b \leftarrow \tilde{\mathcal{H}}_b(a, \textsc{Pair}(X_u, Y))$ //$\mathsf{H}^2_{[1,\infty)}$
$T[u,a,Y] \leftarrow K^b$
$C[u,i] \leftarrow (a,Y)$
Return $((a,Y), K^b)$
```

```
$\textsc{Decap}(u,a,Y)$ //$\mathsf{H}^2$
$(v,i) \leftarrow g^{-1}(a)$
If $i \in I[u]$ and $(a,Y) = C[u,i]$:
    Return $T[u,a,Y]$
    Return $\tilde{\mathcal{H}}_b(a, \textsc{Pair}(X_u, Y))$ //$\mathsf{H}^2_{[1,\infty)}$
$Z \leftarrow Y^{x_u}$ //$\mathsf{H}^2_{[0,1)}$
$K \leftarrow \tilde{\mathcal{H}}_1(a, \textsc{Pair}(\mathbf{g}, Z))$ //$\mathsf{H}^2_{[0,1)}$
$K \leftarrow \tilde{\mathcal{H}}_1(a, \textsc{Pair}(X_u, Y))$ //$\mathsf{H}^2_{[1,\infty)}$
Return $K$

$\textsc{Pair}(X,Y)$ //Internal, $\mathsf{H}^2$
$x \leftarrow \mathrm{dlog}(X)$; $y \leftarrow \mathrm{dlog}(Y)$
Return $q(xy)$
```

```
$\mathrm{H}(a,Z)$ //$\mathsf{H}^2$
Return $\tilde{\mathcal{H}}_1(a, \textsc{Pair}(\mathbf{g}, Z))$
```

```
$\textsc{Encap}(u,i)$ //$\mathsf{H}^3$
$a \leftarrow g(u,i)$
$Y \leftarrow \mathbf{g}^{h(u,i)}$
$K^b \leftarrow \tilde{\mathcal{H}}_b(a, \textsc{Pair}(X_u, Y))$
Return $((a,Y), K^b)$
```

```
$\textsc{Decap}(u,a,Y)$ //$\mathsf{H}^3$
$(v,i) \leftarrow g^{-1}(a)$
If $i \neq \perp$ and $a = g(u,i)$ and $Y = \mathbf{g}^{h(u,i)}$:
    Return $\tilde{\mathcal{H}}_b(a, \textsc{Pair}(X_u, Y))$
$K \leftarrow \tilde{\mathcal{H}}_1(a, \textsc{Pair}(X_u, Y))$
Return $K$

$\textsc{Pair}(X,Y)$ //Internal, $\mathsf{H}^3$
$x \leftarrow \mathrm{dlog}(X)$; $y \leftarrow \mathrm{dlog}(Y)$
Return $q(xy)$
```

```
$\mathrm{H}(a,Z)$ //$\mathsf{H}^3$
$(u,i) \leftarrow g^{-1}(a)$ //$\mathsf{H}^3_{[1,\infty)}$
If $(u,i) \neq \perp$: //$\mathsf{H}^3_{[1,\infty)}$
    $Y \leftarrow \mathbf{g}^{h(u,i)}$ //$\mathsf{H}^3_{[1,\infty)}$
    If $\textsc{Pair}(\mathbf{g}, Z) = \textsc{Pair}(X_u, Y)$: //$\mathsf{H}^3_{[1,\infty)}$
        $\mathsf{bad} \leftarrow \mathsf{true}$ //$\mathsf{H}^3_{[1,\infty)}$
        Return $\tilde{\mathcal{H}}_b(a, \textsc{Pair}(\mathbf{g}, Z))$ //$\mathsf{H}^3_{[1,\infty)}$
Return $\tilde{\mathcal{H}}_1(a, \textsc{Pair}(\mathbf{g}, Z))$
```

Fig. 8. Hybrids games used in proof of Theorem 1. Note that New is shared between all hybrids. Oracles labelled "internal" are not accessible to the adversary. Grey highlighting indicates changes from earlier games.

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. 3) if b is chosen uniformly. We obtained H_0^1 by substituting the code of aECIES into $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}$ and making some notational changes. 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_3^1. First we transition to H_1^1, where we use a random injection g to sample a and a random function h to sample y. Since the inputs to g and h never repeat, the switching lemma gives us $\Pr[\mathsf{H}_0^1] \leqslant \Pr[\mathsf{H}_1^1] + q^2_{\textsc{Encap}}/(2 \cdot 2^l)$.

In H_2^1 we switch from sampling K_0 uniformly at random to assigning it the output of the random function $\mathcal{H}_0$. As the input to $\mathcal{H}_0$ never repeats (in particular the $a = g(u,i)$ component), this does not change the view of the adversary and $\Pr[\mathsf{H}_1^1] = \Pr[\mathsf{H}_2^1]$.

In H_3^1, we modify the Decap oracle where we switch the conditional from checking whether $T[u,a,Y] \neq \bot$ to evaluating the boolean ($v = u$ and $i \in I[u]$ and $(a,Y) = C[u,i]$). Here $I[u]$ tracks the i values for which $\textsc{Encap}(u,i)$ has been queried. The check $(a,Y) = C[u,i]$ implicitly checks if $u = v$ because the first component of $C[u,i]$ is $g(u,i)$, so in the next game transition we will drop the $v = u$ check. These conditions are equivalent since $T[u,a,Y] \neq \bot$ iff the attacker queried $\textsc{Encap}(u,i)$ (i.e., $i \in I[u]$) which produced the ciphertext (a,Y) (i.e., $(a,Y) = C[u,i] = (g(u,i), \mathbf{g}^{h(u,i)})$). Hence, $\Pr[\mathsf{H}_2^1] = \Pr[\mathsf{H}_3^1]$.

Transition H_3^1 to H_0^2 (map-then-rf). Next we transition to hybrid H_0^2. We have highlighted the ways in which H_0^2 differs from H_3^1. We have replaced the hash functions $\mathcal{H}_b$ with $\tilde{\mathcal{H}}_b \circ \lambda$ where λ is defined by $\lambda(a,Z) = (a, \textsc{Pair}(\mathbf{g}, Z))$. Note that λ is an injection because $\textsc{Pair}(\mathbf{g}, \cdot)$ is. As $\tilde{\mathcal{H}}$ is a random function and λ is an injection, their composition is a random function. Hence, $\Pr[\mathsf{H}_3^1] = \Pr[\mathsf{H}_0^2]$.

Transition H_0^2 to H_1^2. In game H_1^2, we use the bilinearity of Pair to compute $\textsc{Pair}(\mathbf{g}, Z)$ without knowing Z in Encap and Decap. We use the X_u and Y from which Z was derived in a way that ensures $\mathrm{dlog}(Z) = \mathrm{dlog}(X_u) \cdot \mathrm{dlog}(Y)$. We have $\Pr[\mathsf{H}_0^2] = \Pr[\mathsf{H}_1^2]$.

Transition H_1^2 to H_0^3. Next we transition to the final set of hybrids H^3, starting with H_0^3. We highlighted important ways in which H_0^3 is different from H_1^2. In H_0^3, have removed the tables I, C, and T. Table T was not being used anywhere. Where C was being used we instead recompute $a = g(u,i)$ and $Y = \mathbf{g}^{h(u,i)}$. The removal of I is the one place where our change modifies the behavior of the game. In Decap, the check if $i \in I[u]$ has been replaced with simply checking for $i \neq \bot$. Hence, the games differ if the adversary makes a query $\textsc{Decap}(u,a,Y)$ with $a = g(u,i)$ and $Y = \mathbf{g}^{h(u,i)}$ despite having not queried $\textsc{Encap}(u,i)$.

We can information theoretically bound the probability of this. We analyze this probability in H_1^2 where the view of the adversary only depends on values of $g(v,i)$ and $h(v,i)$ for which $i \in I[v]$. Consider some fixed point in time when $\mathcal{A}$ makes a $\textsc{Decap}(u,a,Y)$ query with an a not previously returned by encapsulation. Using this view, the adversary must guess some $a = g(u,i)$ such that $i \in \mathcal{I} \setminus I[u]$ along with the corresponding $Y = \mathbf{g}^{h(u,i)}$. There are $|\mathcal{I}|$ points in the image of $g(u,\cdot)$, of which the adversary has seen $|I[u]|$ from Encap. There are 2^l points in the codomain of $g(\cdot,\cdot)$, of which the adversary has seen $|I[\mathcal{U}]|$ from Encap, where we define $I[\mathcal{U}] = \{\, (u,i) \;:\; i \in I[u] \,\}$. Thus we can bound the probability that the adversary picks such an $a \in g(u,\cdot)$ by

$$\frac{|g(u,\mathcal{I}) \setminus g(u, I[u])|}{|\{0,1\}^l \setminus g(I[\mathcal{U}])|} = \frac{|\mathcal{I}| - |I[u]|}{2^l - |I[\mathcal{U}]|} \leqslant \frac{|\mathcal{I}| + |I[\mathcal{U}]| - |I[u]|}{2^l} \leqslant \frac{|\mathcal{I}| + q_{\textsc{Encap}}}{2^l}.$$

In the last inequality we've used that $|I[\mathcal{U}]| \leqslant q_{\textsc{Encap}}$. The adversary must additionally have guessed the correct $\mathbf{g}^{h(u,i)}$, which it has an $1/(p-1)$ chance of having done (as h is a random function). Applying a union bound across all Decap queries gives the bound $\Pr[\mathsf{H}_1^2] \leqslant \Pr[\mathsf{H}_0^3] + (|\mathcal{I}| + q_{\textsc{Encap}}) \cdot q_{\textsc{Decap}}/(2^l \cdot (p-1))$.

Transition H_0^3 to H_1^3. In H_1^3, we have modified H to add a bad condition. In particular, if the attacker every makes a $\mathrm{H}(a,Z)$ query that could possibly correspond to the calculation of a key in Encap, then the oracle uses $\tilde{\mathcal{H}}_b$ instead of $\tilde{\mathcal{H}}_1$. Games H_0^3 and H_1^3 only differ when $b = 0$ and the bad flag gets set. By the fundamental lemma of game playing proofs, $\Pr[\mathsf{H}_0^3] \leqslant \Pr[\mathsf{H}_1^3] + \Pr[b = 0 \wedge \mathsf{H}_1^3 \text{ sets } \mathsf{bad}]$.

Now in H_1^3, the bit b is only used for determining which of $\tilde{\mathcal{H}}_1$ and $\tilde{\mathcal{H}}_0$ is to be used for hash evaluations based on challenge ciphertexts inside of Encap, Decap, and H. Because this use is consistent between all oracles, the adversary's view is independent of the bit b and $\Pr[\mathsf{H}_1^3] \leqslant 1/2$.

```
Adversary $\mathcal{B}_{\mathsf{pcdh}}^{\text{New},\text{Chal},\text{Pair}}$
    $\tilde{\mathcal{H}} \leftarrow\$ \mathsf{Fcs}(\{0,1\}^l \times \mathbb{Z}_p, \mathcal{K})$
    $g \leftarrow\$ \mathsf{Inj}(\mathcal{U} \times \mathcal{I}, \{0,1\}^l)$
    $b' \leftarrow\$ \mathcal{A}^{\text{New},\text{SimEncap},\text{SimDecap},\text{SimH}}$
    Return $\perp$

SimEncap$(u, i)$
    $a \leftarrow g(u, i)$
    $X_u \leftarrow \text{New}(u)$
    $Y \leftarrow \text{Chal}(u, i)$
    $K \leftarrow \tilde{\mathcal{H}}(a, \text{Pair}(X_u, Y))$
    Return $((a, Y), K^b)$

SimDecap$(u, a, Y)$
    $(v, i) \leftarrow g^{-1}(a)$
    $X_u \leftarrow \text{New}(u)$
    $Y_{u,i} \leftarrow \text{Chal}(u, i)$
    If $i \neq \perp$ and $a = g(u, i)$ and $Y = Y_{u,i}$:
        Return $\tilde{\mathcal{H}}(a, \text{Pair}(X_u, Y))$
    $K \leftarrow \tilde{\mathcal{H}}(a, \text{Pair}(X_u, Y))$
    Return $K$

SimH$(a, Z)$
    $(u, i) \leftarrow g^{-1}(a)$
    If $(u, i) \neq \perp$:
        $X_u \leftarrow \text{New}(u)$
        $Y \leftarrow \text{Chal}(u, i)$
        If $\text{Pair}(\mathbf{g}, Z) = \text{Pair}(X_u, Y)$:
            $\mathsf{OUTPUT}(u, i, Z)$
    Return $\tilde{\mathcal{H}}(a, \text{Pair}(\mathbf{g}, Z))$
```

Fig. 9. Adversary $\mathcal{B}_{\mathsf{pcdh}}$ for Theorem 1 proving the security of aECIES.

To bound $\Pr[b = 0 \wedge \mathsf{H}_1^3$ sets $\mathsf{bad}]$, we construct an adversary $\mathcal{B}_{\mathsf{pcdh}}$ given in Fig. 9 against the Pair CDH security of $\mathbb{G}$. It uses New for obtaining users' public keys, Chal for simulating challenge ciphertexts, and its oracle Pair wherever the internal Pair was used in H_1^3 – perfectly simulating $\mathcal{A}$'s view from H_1^3 when $b = 0$. Following the argument above about $\tilde{\mathcal{H}}_1$ and $\tilde{\mathcal{H}}_0$ being indistinguishable in H_1^3, the adversary simply uses a single hash function $\mathcal{H}$. Whenever the flag bad is set, $\mathcal{B}_{\mathsf{pcdh}}$ outputs the corresponding (u, i, Z) and wins the Pair CDH game. Therefore, $\Pr[\mathsf{H}_1^3$ sets $\mathsf{bad} | b = 0] \leqslant \mathsf{Adv}_{\mathbb{G}}^{\mathsf{pcdh}}(\mathcal{B}_{\mathsf{pcdh}})$. Clearly $\Pr[b = 0] = 1/2$. □

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 [CS03]. The augmented Cramer-Shoup key encapsulation mechanism $\mathsf{aCS}[\mathbb{G}, \mathcal{K}, l_1, l_2]$ is parameterized by a group $\mathbb{G} = (\mathbf{g}, p, \circ)$, key space $\mathcal{K}$, and lengths of a random strings, l_1, l_2. We often think of the parameters as fixed and use aCS in place of $\mathsf{aCS}[\mathbb{G}, \mathcal{K}, l_1, l_2]$. We let $\mathsf{aCS.IM} = \mathsf{Fcs}(\{0,1\}^{l_1+l_2} \times \mathbb{G}^2, \mathcal{K})$ and define the scheme as follows. Its keyspace is defined by $\mathsf{aCS}.\mathcal{K} = \mathcal{K}$ and its ciphertext set by $\mathsf{aCS}.\mathcal{C}(ek) = \{0,1\}^{l_2} \times (\langle \mathbf{g} \rangle \setminus \{1_{\mathbb{G}}\})$.

```
aCS.K
    $\alpha \leftarrow\$ \{0,1\}^{l_1}$
    $x \leftarrow\$ \mathbb{Z}_p^*$
    $ek \leftarrow (\alpha, \mathbf{g}^x)$
    $dk \leftarrow (\alpha, x)$
    Return $(ek, dk)$

aCS.E$^{\mathcal{H}}((\alpha, X))$
    $a \leftarrow\$ \{0,1\}^{l_2}$
    $y \leftarrow\$ \mathbb{Z}_p^*$
    $Y \leftarrow \mathbf{g}^y$
    $Z \leftarrow X^y$
    $K \leftarrow \mathcal{H}(\alpha, a, Y, Z)$
    Return $((a, Y), K)$

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

Overview of Existing Techniques and Associated Challenges. A traditional security reduction for the Cramer-Shoup KEM (i.e. aCS with $l_1 = l_2 = 0$) from the Strong CDH problem in the single-user, single-challenge setting uses the lazy sampling technique to simulate $\mathcal{H}$ as a random oracle. The reduction maintains a table T to store $\mathcal{H}$ queries and corresponding responses. When the adversary makes a decapsulation query on Y, the reduction checks the table to see if an entry $T[Y, Z]$ exists such that $\text{Gap}(X, Y, Z) = \mathsf{true}$ where $X = \mathbf{g}^x$ is the public key. If the entry exists, it returns the corresponding value. Otherwise, the reduction samples a new uniformly random element K from the key set $\mathcal{K}$, stores $T[Y, \_] \leftarrow K$ and returns K. The second entry is filled in the table T when the adversary makes a hash query for (Y, Z) such that $\text{Gap}(X, Y, Z) = \mathsf{true}$. 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 $\text{Gap}(X, Y, Z) = \mathsf{true}$. Due to the use of the table T, this reduction is not memory- or time-tight.

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. Bhattacharyya avoided this issue by 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 (all non-challenge) decapsulation queries when the reduction only has access to Y and cannot compute $Z = Y^{dk}$. Since there is only one challenge ciphertext, the reduction simply remembers the challenge ciphertext so it can respond correctly when the adversary forwards the challenge ciphertext.

This proof breaks in the multi-user/multi-challenge setting because it is not clear how to identify and respond to challenge ciphertexts without simply storing them all. Augmenting the scheme with random strings α and a allows encoding the information needed to appropriately respond to queries. Our result is captured by the following theorem.

Theorem 2 (Strong CDH ⇒ \$CCA). *Let $\mathbb{G} = (\mathbf{g}, p, \circ)$ be a group of prime order p. Let $\mathcal{K}$, l_1 and l_2 be fixed. Define $\mathsf{aCS} = \mathsf{aCS}[\mathbb{G}, \mathcal{K}, l_1, l_2]$.*

Let $\mathcal{A}$ be an adversary with $\mathbf{Query}(\mathcal{A}) = (q_{\text{New}}, q_{\text{Encap}}, q_{\text{Decap}}, q_{\text{H}})$. Assume that $2^{l_1} \geqslant |\mathcal{U}|$ and $2^{l_2} \geqslant |\mathcal{I}|$. We construct a $\mathsf{Fcs}(\{0,1\}^{l_1+l_2} \times \mathbb{G} \times \mathbb{G}_\star, \mathcal{K}) \times \mathsf{Inj}^\pm(\mathcal{U}, \{0,1\}^{l_1}) \times \mathsf{Inj}^\pm(\mathcal{U}, \mathcal{I}, \{0,1\}^{l_2})$-oracle adversary $\mathcal{B}$ such that

$$\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aCS}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{scdh}}_{\mathbb{G}}(\mathcal{B}) + \frac{|\mathcal{U}|(|\mathcal{U}|-1)}{2^{l_1}} + \frac{q^2_{\text{Encap}}}{2^{l_2}} + \frac{2q_{\text{Decap}}|\mathcal{I}|}{2^{l_2}(p-1)}$$

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

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

The proof of this result is given in Appendix B. We will later observe that this result can be captured as a special case of our Theorem 4.

The proceedings version of this work claimed security for a version of this scheme captured by setting $l_1 = 0$. There was a bug in our proof of that result and we do not know how to TAM-tightly prove multi-user security of that scheme. TAM-tight single-user, multi-challenge security is captured by the above with $l_1 = 0$ and $|\mathcal{U}| = 1$.

In the proof we observe that the bound could more precisely be written as

$$\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aCS}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{scdh}}_{\mathbb{G}}(\mathcal{B}) + \frac{q_u(q_u-1)}{2^{l_1}} + \frac{q_u q_i^2}{2^{l_2}} + \frac{2q_{\text{Decap}} \cdot |\mathcal{I}|}{2^{l_2+\mathsf{H}_\infty(\mathsf{KEM})}}$$

where q_u is the number of distinct values of u that $\mathcal{A}$ queries to its New, Encap, and Decap oracles and q_i is the maximum number of distinct values of i that $\mathcal{A}$ queries to any user's $\text{Encap}(u,\cdot)$ oracle. Alternatively, we could use $q^2_{u,i}$ in place of $q_u q_i^2$, where $q_{u,i}$ is the number of distinct values of (u,i) that $\mathcal{A}$ queries to its Encap oracle.

Intuition. In this proof, our Strong CDH adversary programs the random string α as the output of a random injection f applied to user identity u and the random string a as the output of a tweakable random injection g_u (with the user identity u as the tweak) applied to challenge identifier i. It simulates the random oracle $\mathcal{H}(f(u), a, Y, Z)$ as $\mathcal{H}(f(u), a, Y, \star)$ when $\text{Strong}(u, Y, Z)$ is true. This allows us to simulate decapsulations because $\text{Strong}(u, Y, Z)$ always holds in Decap.

The adversary simulates challenge ciphertexts using its own $\text{Chal}(u, i)$. To determine if a decapsulation query (u, a, Y) is for a challenge ciphertext, the reduction first inverts a to obtain i. If $i \neq \bot$, it re-queries $\text{Chal}(u, i)$ obtain the corresponding ciphertext $Y_{u,i}$. If $Y = Y_{u,i}$, the reduction assumes this was a challenge ciphertext. Finally, when the adversary $\mathcal{A}$ queries the oracle H with (α, a, Y, Z) such that $f^{-1}(\alpha) = u$, $g_u^{-1}(a) = i$, $\text{Strong}(u, Y, Z) = \mathsf{true}$, and $Y = \text{Chal}(u, i)$, the reduction outputs (u, i, Z) and wins the Strong CDH game.

Removing Key Augmentation with Known Users. In aCS (and schemes to come) we augment the keys with a string α which is included in each random oracle query. The proof uses this as a way to identify which user a given random oracle query "belongs to". In a setting where user identifiers are known (and in particular can be included as an input to the algorithms of the KEM) one could omit refrain from augmenting the keys and instead directly include u in each random oracle query.

Choosing the Auxiliary String Length. We consider how the lengths l_1 and l_2 affect the tightness of our result. Assume again 2^{32} users receiving 2^{32} ciphertexts each (so $q_u = q_i \approx 2^{32}$ and $q_{\text{Decap}} \approx 2^{64}$) and aim for the information theoretic term of our bound to be in the ballpark of 2^{-64} or 2^{-128}. For the setting that $|\mathcal{U}| = |\mathcal{I}| \approx 2^{32}$, we can use the alternate statement of the bound to derive that $(l_1, l_2) \approx (128, 160)$ gives 2^{-64} and $(l_1, l_2) \approx (192, 224)$ gives 2^{-128}. For the setting that $|\mathcal{U}|$ and $|\mathcal{I}|$ are large so we have to hash down to $\mathcal{U}'$ and $\mathcal{I}'$, we derive that $(|\mathcal{U}'|, |\mathcal{I}'|, l_1, l_2) \approx (2^{128}, 2^{160}, 128, 160)$ gives 2^{-64} and $(|\mathcal{U}'|, |\mathcal{I}'|, l_1, l_2) \approx (2^{192}, 2^{224}, 192, 224)$ gives 2^{-128}.

4.3 Augmented Twin ElGamal KEM

Now we give a TAM-tight proof for an augmented version of the Twin ElGamal KEM from plain CDH security. The original security proof for the Twin ElGamal KEM given by Cash, Kiltz, and Shoup [CKS09] (CKS) closely mirrors the security proof for the Cramer-Shoup KEM based on Strong CDH. We modify our proof for the security of the Cramer-Shoup KEM (Theorem 2) to use Strong Twin CDH to TAM-tightly establish the security of the augmented Twin ElGamal KEM.

Augmented Version. In this section we present a memory-tight reduction for an augmented version of the Twin ElGamal KEM [CKS09]. The augmented Twin ElGamal KEM $\mathsf{aTWIN}[\mathbb{G}, \mathcal{K}, l_1, l_2]$ is parameterized by a group $\mathbb{G} = (\mathbf{g}, p, \circ)$, key space $\mathcal{K}$, and lengths of the random strings, l_1, l_2. We treat the parameters $\mathbb{G}, \mathcal{K},$, l_1, and l_2 as constants and often write aTWIN in place of $\mathsf{aTWIN}[\mathbb{G}, \mathcal{K}, l_1, l_2]$. We set $\mathsf{aTWIN.IM}$ as $\mathsf{Fcs}(\{0,1\}^{l_1+l_2} \times \mathbb{G}^3, \mathcal{K})$ and define the scheme as follows. Its keyspace is defined by $\mathsf{aTWIN}.\mathcal{K} = \mathcal{K}$ and its ciphertext set by $\mathsf{aTWIN}.\mathcal{C}(ek) = \{0,1\}^{l_2} \times (\langle \mathbf{g} \rangle \setminus \{1_{\mathbb{G}}\}) \times (\langle \mathbf{g} \rangle \setminus \{1_{\mathbb{G}}\})$.

```
$\mathsf{aTWIN.K}$
$\alpha \leftarrow_\$ \{0,1\}^{l_1}$
$w, x \leftarrow_\$ \mathbb{Z}_p^*$
$ek \leftarrow (\alpha, \mathbf{g}^w, \mathbf{g}^x)$
$dk \leftarrow (\alpha, w, x)$
Return $(ek, dk)$
```

```
$\mathsf{aTWIN.E}^{\mathcal{H}}((\alpha, W, X))$
$a \leftarrow_\$ \{0,1\}^{l_2}$
$y \leftarrow_\$ \mathbb{Z}_p^*$
$Y \leftarrow \mathbf{g}^y$
$Z \leftarrow W^y, \Psi \leftarrow X^y$
$K \leftarrow \mathcal{H}(\alpha, a, Y, Z, \Psi)$
Return $((a, Y), K)$
```

```
$\mathsf{aTWIN.D}^{\mathcal{H}}((\alpha, w, x), (a, Y))$
$Z \leftarrow Y^w$; $\Psi \leftarrow Y^x$
$K \leftarrow \mathcal{H}(\alpha, a, Y, Z, \Psi)$
Return $K$
```

The following theorem captures our TAM-tight security result for this scheme.

Theorem 3 (Strong Twin CDH ⇒ \$CCA). *Let $\mathbb{G} = (\mathbf{g}, p, \circ)$ be a group of prime order p. Let $\mathcal{K}, l_1$, and l_2 be fixed. Define $\mathsf{aTWIN} = \mathsf{aTWIN}[\mathbb{G}, \mathcal{K}, l_1, l_2]$.*

Let $\mathcal{A}$ be a adversary with $\mathbf{Query}(\mathcal{A}) = (q_{\text{New}}, q_{\text{Encap}}, q_{\text{Decap}}, q_{\text{H}})$. Assume that $2^{l_1} \geqslant |\mathcal{U}|$ and $2^{l_1} \geqslant |\mathcal{I}|$. We construct a $\mathsf{Fcs}(\{0,1\}^{l_1+l_2} \times \mathbb{G} \times \mathbb{G}_\star^2, \mathcal{K}) \times \mathsf{Inj}^{\pm}(\mathcal{U}, \{0,1\}^{l_1}) \times \mathsf{Inj}^{\pm}(\mathcal{U}, \mathcal{I}, \{0,1\}^{l_2})$-oracle adversary $\mathcal{B}$ as defined in Fig. 15 such that

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aTWIN}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{stcdh}}_{\mathbb{G}}(\mathcal{B}) + \frac{|\mathcal{U}|(|\mathcal{U}|-1)}{2^{l_1}} + \frac{q_{\text{Encap}}^2}{2^{l_2}} + \frac{2q_{\text{Decap}}|\mathcal{I}|}{2^{l_2}(p-1)} \\
&\mathbf{Query}(\mathcal{B}) = (q_{\text{New}}, (q_{\text{Encap}} + q_{\text{Decap}} + q_{\text{H}}), q_{\text{H}}) \\
&\mathbf{Time}^*(\mathcal{B}) = O(\mathbf{Time}(\mathcal{A})) \\
&\mathbf{Mem}^*(\mathcal{B}) = O(\mathbf{Mem}(\mathcal{A})).
\end{aligned}$$

The proof of this result is given in Appendix C. It is basically identical to the proof of Theorem 2, just with the two group elements Z and Ψ wherever the prior proof had Z. Consequently, we explicitly write the hybrids games and relationships between them, but leave the reader to read the explanations from the proof of Theorem 2 to understand them. Again, we will later observe that this result can be captured as a special case of our Theorem 4.

5 Fujisaki-Okamoto Transformation

The Fujisaki-Okamoto [FO99, FO13] transformations use a random oracle to construct an CCA secure KEM from a weakly (CPA) secure PKE scheme. Hofheinz, Hövelmanns, and Kiltz [HHK17] gave a modular treatment of several Fujisaki-Okamoto-style transformations by splitting them into modules that could be analyzed individually. Bhattacharyya presented memory-tight reductions for several modules analyzed in [HHK17] in the single-user, single-challenge setting [Bha20]. In our work, we present a general construction for CCA-secure KEMs from weakly secure KEM/PKE using Fujisaki-Okamoto-style transformation. We use the message encoding technique along with the map-then-rf technique to prove memory-tight reductions of the Fujisaki-Okamoto-style transformations in the multi-user, multi-challenge setting.

We divide this section into three subsections. First, we introduce the augmented transform $\mathsf{aU}^{\perp}$ that takes as input a weakly (OW-PCVA) secure KEM and transforms it to a CCA secure KEM by using a random oracle. We then present a memory-tight proof for $\mathsf{aU}^{\perp}$ in the multi-user, multi-challenge setting. We show that two of the Hashed-ElGamal KEMs discussed in Sec. 4 can be obtained as instances of the $\mathsf{aU}^{\perp}$ transform. In the following subsections, we introduce the transforms T and aV that can be used in succession to construct a OW-PCVA secure PKE scheme from a weakly (CPA) secure PKE scheme. After canonically constructing a OW-PCVA secure KEM from the OW-PCVA secure PKE (see Sec. 2.4), we can apply the $\mathsf{aU}^{\perp}$ transform to obtain an augmented version of the scheme $\mathsf{QFO}^{\perp}$ considered by [HHK17, Bha20]. We give memory-tight proofs for T and aV.

5.1 Augmented Transformation $\mathsf{aU}^{\perp}$ [OW-PCVA → CCA]

The transformation $\mathsf{aU}^{\perp}$ constructs a CCA secure key encapsulation mechanism $\mathsf{aUEM} = \mathsf{aU}^{\perp}[\mathsf{KEM}, \mathcal{K}, l_1, l_2]$ from a OW-PCVA secure key encapsulation mechanism KEM, where the key set $\mathcal{K}$ and lengths l_1, l_2 are fixed parameters of aUEM. As done before, we use aUEM instead of $\mathsf{aUEM}[\mathsf{KEM}, \mathcal{K}, l_1, l_2]$ for notational convenience. We define aUEM as follows where $\mathsf{aUEM.IM} = \mathsf{Fcs}(\{0,1\}^{l_1+l_2} \times \mathsf{KEM}.\mathcal{K} \times \mathsf{KEM}.\mathcal{C}, \mathsf{aUEM}.\mathcal{K}) \times \mathsf{KEM.IM}$.

```
$\underline{\mathsf{aUEM.K}}$
$\alpha \leftarrow\!\!\$\ \{0,1\}^{l_1}$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{KEM.K}$
Return $((\alpha, ek), (\alpha, dk))$
```

```
$\underline{\mathsf{aUEM.E}^{\mathcal{H}\times\mathcal{H}'}((\alpha, ek))}$
$a \leftarrow\!\!\$\ \{0,1\}^{l_2}$
$(c, k) \leftarrow\!\!\$\ \mathsf{KEM.E}^{\mathcal{H}'}(ek)$
$K \leftarrow \mathcal{H}(\alpha, a, c, k)$
Return $((a, c), K)$
```

```
$\underline{\mathsf{aUEM.D}^{\mathcal{H}\times\mathcal{H}'}((\alpha, dk), (a, c))}$
$k \leftarrow \mathsf{KEM.D}^{\mathcal{H}'}(dk, c)$
If $k = \perp$ then return $\perp$
$K \leftarrow \mathcal{H}(\alpha, a, c, k)$
Return $K$
```

We note that aUEM is δ-correct and ε-uniform (for $\mathsf{aUEM}.\mathcal{C}(ek) = \{0,1\}^{l_2} \times \mathsf{KEM}.\mathcal{C}(ek)$) if KEM is. It has min-entropy $\mathsf{H}_{\infty}(\mathsf{aUEM}) = \mathsf{H}_{\infty}(\mathsf{KEM})$.

We present a memory-tight reduction for the augmented transformation $\mathsf{aU}^{\perp}$ in the multi-user, multi-challenge setting. Our result is captured in the following theorem which we prove in Appendix D.

Theorem 4 (OW-PCVA ⇒ CCA). *Let* $\mathsf{aUEM} = \mathsf{aU}^{\perp}[\mathsf{KEM}, \mathcal{K}, l_1, l_2]$ *where* KEM *is* δ*-correct. Let* $\mathcal{A}$ *be an adversary against* aUEM *with* $\mathbf{Query}(\mathcal{A}) = (q_{\text{New}}, q_{\text{Encap}}, q_{\text{Decap}}, q_{\text{H}})$. *Assume* $2^{l_1} \geqslant |\mathcal{U}|$ *and* $2^{l_2} \geqslant |\mathcal{I}|$. *We construct a* $\mathsf{Fcs}(\{0,1\}^{l_1+l_2} \times \mathsf{KEM}.\mathcal{C} \times \mathsf{KEM}.\mathcal{K}_{\star}, \mathcal{K}) \times \mathsf{Inj}^{\pm}(\mathcal{U}, \{0,1\}^{l_1}) \times \mathsf{Inj}^{\pm}(\mathcal{U}, \mathcal{I}, \{0,1\}^{l_1})$*-oracle adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathsf{mu\text{-}cca}}_{\mathsf{aUEM}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}pcva}}_{\mathsf{KEM}}(\mathcal{B}) + 2q_{\text{Encap}} \cdot \delta + \frac{|\mathcal{U}|(|\mathcal{U}|-1)}{2^{l_1}} + \frac{q^2_{\text{Encap}}}{2^{l_2}} + \frac{2q_{\text{Decap}} \cdot |\mathcal{I}|}{2^{l_2+\mathsf{H}_{\infty}(\mathsf{KEM})}}$$

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

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

In the proof we observe that the bound could more precisely be written as

$$\mathsf{Adv}^{\mathsf{mu\text{-}cca}}_{\mathsf{aUEM}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}pcva}}_{\mathsf{KEM}}(\mathcal{B}) + 2q_{u,i} \cdot \delta + \frac{q_u(q_u-1)}{2^{l_1}} + \frac{q^2_{u,i}}{2^{l_2}} + \frac{2q_{\text{Decap}} \cdot |\mathcal{I}|}{2^{l_2+\mathsf{H}_{\infty}(\mathsf{KEM})}}$$

where q_u is the number of distinct values of u that $\mathcal{A}$ queries to its New, Encap, and Decap oracles and $q_{u,i}$ is the number of distinct values of (u,i) that $\mathcal{A}$ queries to its Encap oracle. Alternatively, we could use $q_u q_i^2$ in place of $q_{u,i}^2$, where q_i is the maximum number of distinct values of i that $\mathcal{A}$ queries to any user's $\text{Encap}(u,\cdot)$ oracle.

The proceedings version of this work claimed security for a version of this scheme captured by setting $l_1 = 0$. There was a bug in our proof of that result and we do not know how to TAM-tightly prove multi-user security of that scheme. TAM-tight single-user, multi-challenge security is captured by the above with $l_1 = 0$ and $|\mathcal{U}| = 1$.

Intuition. This proof is very similar to the proof of Theorem 2. Once again, our OW-PCVA adversary programs the random string α to be the output of the injective function $f(u)$ and the random string a to be the output of the tweakable random injection $g_u(i)$. We use the map-then-rf technique to simulate the oracle $\mathrm{H}(\alpha, a, c, k)$ as $\mathcal{H}(\alpha, a, c, \star)$ when k is the decapsulation of c.

The adversary simulates the challenge i ciphertext for user u as $\text{Chal}(u,i)$. To determine if a decapsulation query (u,a,c) is for a challenge ciphertext, the reduction first inverts a to obtain i. If $i \neq \perp$, it queries $\text{Chal}(u,i)$ and checks if this outputs c. To check for invalid ciphertexts in the simulation of Decap, it uses its CV oracle. Finally, when the aUEM adversary queries the H oracle with a tuple (α, a, c, k) such that $\mathrm{PC}(u,k,c) = \mathsf{true}$ and $c = \text{Chal}(u,i)$ where $u = f^{-1}(\alpha), i = g_u^{-1}(a)$, the reduction outputs (u,i,k) and wins its game.

Hashed ElGamal KEMs. The Hashed ElGamal KEMs aCS and aTWIN can be captured by applying the transformation $\mathsf{aU}^{\perp}$ to weakly (OW-PCA) secure key encapsulation mechanisms gKEM and tKEM defined below for a group $\mathbb{G}$ of prime order p.

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

```
gKEM.E($X$)
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$c \leftarrow \mathbf{g}^y$
$k \leftarrow X^y$
Return $(c, k)$
```

```
gKEM.D($x, c$)
$k \leftarrow c^x$
Return $k$
```

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

```
tKEM.E($W, X$)
$y \leftarrow_{\$} \mathbb{Z}_p^*$
$c \leftarrow \mathbf{g}^y$
$k \leftarrow (W^y, X^y)$
Return $(c, k)$
```

```
tKEM.D($(w, x), c$)
$k \leftarrow (c^w, c^x)$
Return $k$
```

For these KEMs, the notions of OW-PCA and OW-PCVA security are equivalent because all $c \in \mathbb{G}$ are valid ciphertexts and we assume inclusion in the group can be efficiently checked. The OW-PCA security of gKEM or tKEM is, respectively, equivalent to the Strong CDH or Strong Twin CDH of the underlying group (with the strong oracle playing the role of the plaintext checking oracle). It is easy to see that $\mathsf{aCS} = \mathsf{aU}^{\perp}[\mathsf{gKEM}]$ and $\mathsf{aTWIN} = \mathsf{aU}^{\perp}[\mathsf{tKEM}]$. Both schemes are 0-uniform, are perfectly correct, and have min-entropy $\log_2(p-1)$ which allows us to recover Theorems 2 and 3 as special cases of Theorem 4.

5.2 Transformation T [CPA → OW-PCA]

The transformation T constructs a deterministic OW-PCA secure public key encryption scheme $\mathsf{TKE} = \mathsf{T}[\mathsf{PKE}]$ from a 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 $\mathsf{TKE}.\mathcal{M} = \mathsf{PKE}.\mathcal{M}$.

```
TKE.K
$(ek, dk) \leftarrow_{\$} \mathsf{PKE.K}$
Return $(ek, (ek, dk))$
```

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

```
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 $m'$
```

Note that the if statement in TKE.D ensures that TKE is rigid.

The following theorem gives a memory-tight reduction for T in the multi-user, multi-challenge setting using the randomness programming technique.

Theorem 5 (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* $\mathbf{Query}(\mathcal{A}) = (q_{\text{New}}, q_{\text{Chal}}, q_{\text{PC}}, q_{\text{H}})$. *Assume*

PKE*'s algorithms make at most* q_{PKE} *oracle queries and define* $q^* = q_{\mathrm{H}} + q_{\mathrm{CHAL}}(q_{\mathsf{PKE}} + 1) + q_{\mathrm{PC}}(2q_{\mathsf{PKE}} + 1)$. *Assume* $|\mathsf{PKE}.\mathcal{M}| \geqslant \mathcal{U} \times \mathcal{I}$. *We construct an* $\mathsf{Fcs}(\mathsf{PKE}.\mathcal{M}, \mathsf{PKE}.\mathcal{R}) \times \mathsf{Inj}^{\pm}(\mathcal{U} \times \mathcal{I}, \mathsf{PKE}.\mathcal{M})$*-oracle adversary* $\mathcal{B}$ *such that*

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

$$\mathbf{Query}(\mathcal{B}) = (q_{\mathrm{NEW}} + q_{\mathrm{PC}}, q_{\mathrm{CHAL}}, q_{\mathrm{H}} + q_{\mathrm{PC}} \cdot q_{\mathsf{PKE}})$$

$$\mathbf{Time}^*(\mathcal{B}) = O(\mathbf{Time}(\mathcal{A}))$$

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

The proof of this theorem is given in Appendix E. In the proof, we note that the correctness term could more precisely be written as $q_u \cdot \delta'(q^*)$ where q_u is the number of distinct values of u that $\mathcal{A}$ queries to its oracles or outputs at the end of execution.

INTUITION. For this proof, our CPA adversary programs the random messages m to be the output of a random injection $g(u, i)$. It simulates the challenge ciphertext i for user u by invoking its own encryption oracle on $g(u, i)$. It simulates the plaintext checking oracle by seeing if the given message encrypts to the given ciphertext. 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 our 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.

5.3 Augmented 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 [Bha20]. Our augmentation adds a random string to the keys 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.

```
VKE.K
$\alpha \leftarrow_\$ \{0,1\}^l$
$(ek, dk) \leftarrow_\$$ TKE.K
Return $((\alpha, ek), (\alpha, ek, dk))$
```

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

```
VKE.D^{H×H'}((α, ek, dk), c)
$(c_1, c_2) \leftarrow c$
$m' \leftarrow \mathsf{TKE.D}^{\mathcal{H}'}(dk, c_1)$
If $m' = \bot$ or $\mathcal{H}(\alpha, m') \neq c_2$ or $\mathsf{TKE.E}^{\mathcal{H}'}(ek, m') \neq c_1$:
  Return $\bot$
Return $m'$
```

Note that aV is rigid and if TKE is rigid and δ'-correct, then aV is δ'-correct. If TKE is rigid, then the re-encryption inside of $\mathsf{VKE.D}$ is superfluous.

We present a memory-tight reduction for aV in the multi-user, multi-challenge setting using the randomness programming technique.

Theorem 6 (OW-PCA ⇒ OW-PCVA). *Let* $\mathsf{VKE} = \mathsf{aV}[\mathsf{TKE}]$ *and suppose* TKE *is rigid and* δ'*-correct. Let* $\mathcal{A}$ *be an adversary against* VKE *with* $\mathbf{Query}(\mathcal{A}) = (q_{\mathrm{U}}, q_{\mathrm{CHAL}}, q_{\mathrm{PC}}, q_{\mathrm{CV}}, q_{\mathrm{H}})$. *Assume that* $2^l \geqslant |\mathcal{U}|$. *Assume* TKE*'s algorithms make at most* q_{TKE} *oracle queries and define* $q^* = q_{\mathsf{TKE}}(2q_{\mathrm{CHAL}} + 2q_{\mathrm{H}} + q_{\mathrm{PC}} + q_{\mathrm{CV}} + 1) + q_{\mathrm{H}}$. *We construct an* $\mathsf{Fcs}(\{0,1\}^l \times \mathsf{TKE}.\mathcal{M}, \{0,1\}^\gamma) \times \mathsf{Fcs}(\mathcal{U} \times \mathsf{TKE}.\mathcal{C}, \{0,1\}^\gamma) \times \mathsf{Inj}^{\pm}(\mathcal{U}, \{0,1\}^l)$*-oracle adversary* $\mathcal{B}$ *against* TKE *such that*

$$\mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}pcva}}_{\mathsf{VKE}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}pca}}_{\mathsf{TKE}}(\mathcal{B}) + 2|\mathcal{U}| \cdot \delta'(q^*) + \frac{|\mathcal{U}|^2}{2^{l+1}} + \frac{q_{\mathrm{CV}}}{2^\gamma}$$

$$\mathbf{Query}(\mathcal{B}) = (q_{\mathrm{NEW}}, q_{\mathrm{CHAL}}, q_{\mathrm{PC}}, q_{\mathrm{H}} \cdot q_{\mathsf{TKE}})$$

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

The proof is given in Appendix F.

Intuition. Our OW-PCA adversary for this proof programs the random string α as the output of the random injection f applied to user identity u and simulates hash oracle queries $\mathcal{H}(\alpha, m)$ as $\tilde{\mathcal{H}}(u, \mathsf{VKE.E}^{\mathcal{H}'}(ek_u, m))$ where $u = f^{-1}(\alpha)$. To simulate challenge ciphertexts, our adversary queries its own Chal oracle to obtain the first part of the ciphertext c_1, and computes the hash $\tilde{\mathcal{H}}(u, c_1)$ to obtain the second part c_2. Similarly, to simulate PC queries of the form $\mathrm{PC}(u, m, c)$, it invokes its own PC oracle to check if the first part c_1 of the input ciphertext c decrypts to m and if so, it checks if the second part c_2 equals $\tilde{\mathcal{H}}(u, c_1)$. To simulate the CV oracle, our adversary simply verifies the second part of the ciphertext c_2 by hashing the first part c_1. Finally, when the OW-PCVA adversary $\mathcal{A}$ outputs the tuple (m', u, i), our reduction outputs the same and wins its game whenever $\mathcal{A}$ wins.

Fujisaki-Okamoto Construction. The public-key encryption scheme VKE obtained as a result of applying the T and aV transforms can be used to construct a CCA secure KEM using the $\mathsf{aU}^{\perp}$ transform. For this, we first convert VKE to a KEM as $\mathsf{KEM} = \mathsf{CaK}[\mathsf{VKE}]$. As noted in Sec. 2.4, $\mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}pcva}}_{\mathsf{VKE}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}pcva}}_{\mathsf{CaK[VKE]}}(\mathcal{A})$ for any $\mathcal{A}$. Therefore, KEM is OW-PCVA secure, and can be used as an input to the $\mathsf{aU}^{\perp}$ transform to obtain a CCA secure KEM, aUEM. The construction $\mathsf{aUEM} = \mathsf{aU}^{\perp}[\mathsf{CaK}[\mathsf{aV}[\mathsf{T}[\mathsf{PKE}]]]]$ is essentially an augmented version of the QFO transform in [HHK17, Bha20], which one could call $\mathsf{aQFO}^{\perp}$.

6 Memory-Tight Reduction for PKE Schemes (via KEM/DEM)

In this section, we provide a modified version of the TAM-tight security proof from [GGJS12] to show the security of the KEM/DEM construction of public key encryption [CS03]. 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 $\mathsf{KD.IM} = \mathsf{KEM.IM} = \mathsf{SKE.IM}$. Assuming KEM and SKE use the same ideal model is without loss of generality as it could be the case that the set of functions is $\mathsf{F} \times \mathsf{G}$ where KEM only actually queries F and SKE only actually queries G. We assume that $\mathsf{SKE.K}$ outputs a uniformly random key from $\mathsf{KEM}.\mathcal{K}$.

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

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

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

We will give two different theorems capturing the TAM-tight security of KD. The first, Theorem 7, restricts attention to challenge respecting adversaries to cover the more typical notion of security in which repeated queries to encryption are disallowed. The second, Theorem 8, is in the more general setting where the attacker is allowed to repeat encryption queries, receiving back the same ciphertext from its original query. The proof of the second is harder and results in worse concrete security, corresponding to a need for the ciphertexts of the underlying schemes to be longer.

Theorem 7. *Let SKE be a symmetric key encryption scheme, KEM be a δ-correct, ε-uniform key encapsulation mechanism, and $\mathsf{KD} = \mathsf{KD}[\mathsf{SKE}, \mathsf{KEM}]$. Let $T = \mathcal{U} \times \mathsf{KEM.Ek}$, $D_{(u,ek)} = \mathcal{I}$ and $R_{(u,ek)} = \mathsf{KEM}.\mathcal{C}(ek)$. Let $T' = \mathcal{U} \times \bigcup_{ek \in \mathsf{KEM.Ek}} \mathsf{KEM}.\mathcal{C}(ek) \times \mathbb{N}$, $D'_{(u,c^k,l)} = \{0,1\}^l$, and $R'_{(u,c^k,l)} = \{0,1\}^{\mathsf{SKE.cl}(l)}$. Assume SKE's and KEM's algorithms make at most q' queries to the ideal model. Let $\mathcal{A}$ be a* challenge-respecting *adversary against KD with* **Query**$(\mathcal{A}) = (q_{\mathrm{New}}, q_{\mathrm{Enc}}, q_{\mathrm{Dec}}, q_{\mathrm{H}})$. *Then we can construct an $\mathsf{Inj}^{\pm}(T', D', R')$-oracle adversary $\mathcal{B}_{\mathsf{KEM}}$ and $\mathsf{Fcs}(\mathcal{U}, \mathsf{KEM}.\mathcal{R}) \times \mathsf{Inj}^{\pm}(T, D, R)$-oracle adversary against $\mathcal{B}_{\mathsf{SKE}}$ such that*

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

$$\begin{aligned} &\mathbf{Query}(\mathcal{B}_{\mathsf{KEM}}) = (q_{\mathrm{New}}, q_{\mathrm{Enc}}, q_{\mathrm{Dec}}, q_{\mathrm{H}} + (q_{\mathrm{Enc}} + q_{\mathrm{Dec}})q') && \mathbf{Query}(\mathcal{B}_{\mathsf{SKE}}) = (q_{\mathrm{Enc}}, q_{\mathrm{Dec}}, q_{\mathrm{H}} + 2q_{\mathrm{Dec}}q') \\ &\mathbf{Time}^*(\mathcal{B}_{\mathsf{KEM}}) = O(\mathbf{Time}(\mathcal{A})) && \mathbf{Time}^*(\mathcal{B}_{\mathsf{SKE}}) = O(\mathbf{Time}(\mathcal{A})) \\ &\mathbf{Mem}^*(\mathcal{B}_{\mathsf{KEM}}) = O(\mathbf{Mem}(\mathcal{A})) && \mathbf{Mem}^*(\mathcal{B}_{\mathsf{SKE}}) = O(\mathbf{Mem}(\mathcal{A})). \end{aligned}$$

$\mathcal{B}_{\mathsf{KEM}}$ *and* $\mathcal{B}_{\mathsf{SKE}}$ *are both challenge respecting and* $\mathcal{B}_{\mathsf{SKE}}$ *makes at most one encryption query per user.*

The proof of this theorem is given in Appendix G. In the proof, we are able to make a number of simplifying assumptions because the adversary is challenge respecting and so we do not have to worry about maintaining consistency between repeated queries. The following theorem captures the more general case when repeat queries are allowed, so we cannot make such simplifying assumptions. It differs from the prior theorem in the oracles used by the adversaries and the information theoretic part of the bound.

Theorem 8. *Let* SKE *be a symmetric key encryption scheme,* KEM *be a δ-correct, ε-uniform key encapsulation mechanism, and* KD = KD[SKE, KEM]. *Let* $T = \mathcal{U} \times \mathsf{KEM.Ek}$, $D_{(u,ek)} = \mathcal{I}$ *and* $R_{(u,ek)} = \mathsf{KEM}.\mathcal{C}(ek)$. *Let* $T' = \mathcal{U} \times \bigcup_{ek \in \mathsf{KEM.Ek}} \mathsf{KEM}.\mathcal{C}(ek) \times \mathbb{N}$, $D'_{(u,c^k,l)} = \{0,1\}^l$, *and* $R'_{(u,c^k,l)} = \{0,1\}^{\mathsf{SKE.cl}(l)}$. *Assume* SKE*'s and* KEM*'s algorithms make at most q' queries to the ideal model. Let $\mathcal{A}$ be an* arbitrary *adversary against* KD *with* $\mathbf{Query}(\mathcal{A}) = (q_{\mathrm{New}}, q_{\mathrm{Enc}}, q_{\mathrm{Dec}}, q_{\mathrm{H}})$. *Fix $\gamma \in \mathbb{N}$. Then we can construct an* $\mathsf{Fcs}(\mathcal{U} \times \mathcal{I} \times \{0,1\}^*, \mathsf{SKE}.\mathcal{R}) \times \mathsf{Inj}^{\pm}(T', D', R')$*-oracle adversary* $\mathcal{B}_{\mathsf{KEM}}$ *and* $\mathsf{Fcs}(\mathcal{U}, \mathsf{KEM}.\mathcal{R}) \times \mathsf{Inj}^{\pm}(T, D \times \{0,1\}^\gamma, R) \times \mathsf{Fcs}(\{0,1\}^*, \{0,1\}^\gamma)$*-oracle adversary against* $\mathcal{B}_{\mathsf{SKE}}$ *such that*

$$\begin{aligned} \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD}}(\mathcal{A}) \leqslant\ & 2\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE}}(\mathcal{B}_{\mathsf{SKE}}) + q_{\mathrm{Enc}} \cdot (2\delta + \varepsilon) \\ & + \frac{2q_{\mathrm{Enc}}^2 + 2q_{\mathrm{Enc}}q_{\mathrm{Dec}}}{|\mathsf{KEM}.\mathcal{C}|} + \frac{q_{\mathrm{Enc}}^2 + 2q_{\mathrm{Dec}}}{2^{\mathsf{SKE.xl}}} + \frac{q_{\mathrm{Enc}}^2}{2^\gamma} + \frac{2q_{\mathrm{Dec}} \cdot |\mathcal{I}| \cdot 2^\gamma}{|\mathsf{KEM}.\mathcal{C}|} \end{aligned}$$

$$\begin{aligned} &\mathbf{Query}(\mathcal{B}_{\mathsf{KEM}}) = (q_{\mathrm{New}}, q_{\mathrm{Enc}}, q_{\mathrm{Dec}}, q_{\mathrm{H}} + (q_{\mathrm{Enc}} + q_{\mathrm{Dec}})q') && \mathbf{Query}(\mathcal{B}_{\mathsf{SKE}}) = (q_{\mathrm{Enc}}, q_{\mathrm{Dec}}, q_{\mathrm{H}} + 2q_{\mathrm{Dec}}q') \\ &\mathbf{Time}^*(\mathcal{B}_{\mathsf{KEM}}) = O(\mathbf{Time}(\mathcal{A})) && \mathbf{Time}^*(\mathcal{B}_{\mathsf{SKE}}) = O(\mathbf{Time}(\mathcal{A})) \\ &\mathbf{Mem}^*(\mathcal{B}_{\mathsf{KEM}}) = O(\mathbf{Mem}(\mathcal{A})) && \mathbf{Mem}^*(\mathcal{B}_{\mathsf{SKE}}) = O(\mathbf{Mem}(\mathcal{A})). \end{aligned}$$

$\mathcal{B}_{\mathsf{SKE}}$ *may (with low probability) make multiple encryption queries to a user.*

The proof of this result is in Appendix H. Unsurprisingly, much of the proof overlaps with that of Theorem 7. Rather than repeat most of the logic, we give the explicit pseudocode for our hybrids and reductions as well as the associated security bounds. Then we explain specially the places where the games and analysis differs from the prior proof.

A primary difficulty in the proof comes from simulating the KEM ciphertext after it has been switched to random. In the proof of Theorem 7 we are able to pick it as the output of a random injection tweaked by the user u applied to the challenge identifier i. This allows us to try inverting the KEM ciphertext during decryption to see if it could correspond to a challenge ciphertext and react appropriately if so. Doing this in our proof of Theorem 8 would not work because it would result in two queries $\textsc{Enc}(u, i, m)$ and $\textsc{Enc}(u, i, m')$ for $m \neq m'$ having the same KEM ciphertext when they should have ciphertexts that look completely independent. So we need to make the KEM ciphertext depend on the message m. Using m as a tweak does not work, as in decryption we would not know which m to use when using the inverse of the inject. Using m as an input would be wholly unsatisfying as it would require the KEM ciphertext to be at least as long as the message! We resolve this issue in our proof by hashing the message down to a shorter string and including this as input to the injection. This works as long as the attacker does not find a collision in the hash.

The proof includes a few other minor adjustments required to allow the correct behavior of repeating ciphertexts when encryption queries repeat while giving seemingly independent outputs when they do not.

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 the KEMs studied earlier in this work for the first component. Note that $\mathsf{aU}^{\perp}$ was only proven CCA secure, where we require \$CCA security. Per an earlier observation, this gap can be bridged for any ε-uniform KEM.

For 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. Depending on the believed security of KEM, one reasonable option could be GCM with a random nonce. In the ideal cipher model, Hoang, Tessaro, and Thiruvengadam [HTT18] showed a strong bound for this setting which is essentially independent of the number of users.

References

ABR98. Michel Abdalla, Mihir Bellare, and Phillip Rogaway. DHIES: An encryption scheme based on the Diffie-Hellman problem. Contributions to IEEE P1363a, September 1998. 3, 18

ABR01. Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer, Heidelberg, April 2001. doi:10.1007/3-540-45353-9_12. 2, 13

ACFK17. Benedikt Auerbach, David Cash, Manuel Fersch, and Eike Kiltz. Memory-tight reductions. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 101–132. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63688-7_4. 2, 6, 15, 18, 34

BBM00. Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, Heidelberg, May 2000. doi:10.1007/3-540-45539-6_18. 9

BFL20. Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 121–151. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56880-1_5. 15, 34

Bha20. Rishiraj Bhattacharyya. Memory-tight reductions for practical key encapsulation mechanisms. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 249–278. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45374-9_9. 2, 8, 9, 18, 25, 27, 28

BP18. Daniel J. Bernstein and Edoardo Persichetti. Towards KEM unification. Cryptology ePrint Archive, Report 2018/526, 2018. https://eprint.iacr.org/2018/526. 8, 11

BR06. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. doi:10.1007/11761679_25. 6

CKS09. David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. *Journal of Cryptology*, 22(4):470–504, October 2009. doi:10.1007/s00145-009-9041-6. 3, 5, 15, 16, 24

CS03. Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003. doi:10.1137/S0097539702403773. 3, 22, 28

DGJL21. Denis Diemert, Kai Gellert, Tibor Jager, and Lin Lyu. Digital signatures with memory-tight security in the multi-challenge setting. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 403–433. Springer, Heidelberg, December 2021. doi:10.1007/978-3-030-92068-5_14. 2

Din20a. Itai Dinur. On the streaming indistinguishability of a random permutation and a random function. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 433–460. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45724-2_15. 2

Din20b. Itai Dinur. Tight time-space lower bounds for finding multiple collision pairs and their applications. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 405–434. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45721-1_15. 2

DTZ20. Wei Dai, Stefano Tessaro, and Xihu Zhang. Super-linear time-memory trade-offs for symmetric encryption. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 335–365. Springer, Heidelberg, November 2020. doi:10.1007/978-3-030-64381-2_12. 2

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96881-0_2. 5, 13

FO99. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_34. 25

FO13. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013. doi:10.1007/s00145-011-9114-1. 25

GGJS12. Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 99–116. Springer, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_8. 28

GGJT22. Ashrujit Ghoshal, Riddhi Ghosal, Joseph Jaeger, and Stefano Tessaro. Hiding in plain sight: Memory-tight proofs via randomness programming. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 706–735. Springer, Heidelberg, May / June 2022. doi:10.1007/978-3-031-07085-3_24. 2, 4, 5, 7, 8, 9, 19

GHKW16. Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Tightly CCA-secure encryption without pairings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 1–27. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49890-3_1. 2

GJT20. Ashrujit Ghoshal, Joseph Jaeger, and Stefano Tessaro. The memory-tightness of authenticated encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 127–156. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56784-2_5. 2

GT20. Ashrujit Ghoshal and Stefano Tessaro. On the memory-tightness of hashed ElGamal. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 33–62. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45724-2_2. 2, 3, 9

HHK17. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70500-2_12. 5, 8, 9, 11, 25, 28

HJ12. Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 590–607. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_35. 2

Höv21. Kathrin Hövelmanns. *Generic constructions of quantum-resistant cryptosystems*. PhD thesis, Dissertation, Bochum, Ruhr-Universität Bochum, 2020, 2021. doi:10.13154/294-7758. 9

HTT18. Viet Tung Hoang, Stefano Tessaro, and Aishwarya Thiruvengadam. The multi-user security of GCM, revisited: Tight bounds for nonce randomization. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1429–1440. ACM Press, October 2018. doi:10.1145/3243734.3243816. 30

JK22. Joseph Jaeger and Akshaya Kumar. Memory-tight multi-challenge security of public-key encryption. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part III*, volume 13793 of *LNCS*, pages 454–484. Springer, Heidelberg, December 2022. doi:10.1007/978-3-031-22969-5_16. 1, 5, 6

JT19. Joseph Jaeger and Stefano Tessaro. Tight time-memory trade-offs for symmetric encryption. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 467–497. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17653-2_16. 2

LJYP14. Benoît Libert, Marc Joye, Moti Yung, and Thomas Peters. Concise multi-challenge CCA-secure encryption and signatures with almost tight security. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 1–21. Springer, Heidelberg, December 2014. doi:10.1007/978-3-662-45608-8_1. 2

Mau05. Ueli Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *Cryptography and Coding*, pages 1–12, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/11586821_1. 5, 13, 15, 34

OP01. Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer, Heidelberg, February 2001. doi:10.1007/3-540-44586-2_8. 13

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18. 5, 13, 15, 34

SOS20. Ido Shahaf, Or Ordentlich, and Gil Segev. An information-theoretic proof of the streaming switching lemma for symmetric encryption. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 858–863, 2020. doi:10.1109/ISIT44484.2020.9174331. 2

TT18. Stefano Tessaro and Aishwarya Thiruvengadam. Provable time-memory trade-offs: Symmetric cryptography against memory-bounded adversaries. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 3–32. Springer, Heidelberg, November 2018. doi:10.1007/978-3-030-03807-6_1. 2

WMHT18. Yuyu Wang, Takahiro Matsuda, Goichiro Hanaoka, and Keisuke Tanaka. Memory lower bounds of reductions revisited. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 61–90. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78381-9_3. 2

```
Game $\mathsf{G}^{\mathsf{x}}_{\mathbb{G}}(\mathcal{A})$
$(\mathbf{g}, p, \circ) \leftarrow \mathbb{G}$
$x, y \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$X \leftarrow \mathbf{g}^x$; $Y \leftarrow \mathbf{g}^y$ $\mathrm{O} \leftarrow \perp$ $//\mathsf{G}^{\mathsf{acdh}}$
$\mathrm{O} \leftarrow \textsc{Pair}$ $//\mathsf{G}^{\mathsf{apcdh}}$
$f \leftarrow\!\!\$\ \mathsf{Inj}(\mathbb{Z}_p, \mathbb{Z}_p)$ $//\mathsf{G}^{\mathsf{apcdh}}$
$[Z]_{\vec{z}} \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{O}}(X, Y)$
Return $(Z = \mathbf{g}^{xy})$ and $(Z = g^{z_1} X^{z_2} Y^{z_3})$
```

```
$\textsc{Pair}([A]_{\vec{a}}, [B]_{\vec{b}})$
Require $A = g^{a_1} X^{a_2} Y^{a_3}$
Require $B = g^{b_1} X^{b_2} Y^{b_3}$
$a \leftarrow \mathrm{dlog}(A)$; $b \leftarrow \mathrm{dlog}(B)$
Return $f(ab)$
```

```
Game $\mathsf{G}^{\mathrm{dlog}}_{\mathbb{G}}(\mathcal{A})$
$(\mathbf{g}, p, \circ) \leftarrow \mathbb{G}$
$c \leftarrow\!\!\$\ \mathbb{Z}_p$
$c' \leftarrow\!\!\$\ \mathcal{A}(\mathbf{g}^c)$
Return $(c = c')$
```

Fig. 10. Security games G^{x} for (single-user, single-challenge) PairCDH and CDH in the algebraic group model and security game $\mathsf{G}^{\mathrm{dlog}}$ for discrete log.

A Pair Diffie-Hellman in Ideal Models

We will show that in the algebraic group model CDH security and discrete log security together imply Pair CDH security. As discrete log security is implied by CDH security in the algebraic group model, this implies that CDH and Pair CDH are equivalent in the algebraic group model. (Indeed, discrete log security is known to be equivalent to CDH security in the algebraic group model so all three notions are equivalent therein.) We work with single-user, single-challenge variants of CDH security games as Lem. 3 extends readily to show it TAM-tightly implies multi-user, multiple-challenge security in the algebraic group model.

In the algebraic group model, each time an adversary $\mathcal{A}$ outputs a group element (as an oracle query or final output) it must additionally output a vector over $\mathbb{Z}_p$ which "explains" how the group element can be derived from $\mathbf{g}$ and all other group elements previously given to the adversary. Security games for Pair CDH and CDH in the algebraic group model are given in Fig. 10. In these game we use the notation $[A]_{\vec{a}}$ to denote a tuple (A, a_1, a_2, a_2) for which $g^{a_1} X^{a_2} Y^{a_3} = A$. We perform this check explicitly in the code.

In the same figure we give the security game for the standard notion of discrete log security in which the adversary is given a random group element with the goal of producing its discrete logarithm. For $\mathsf{x} \in \{\mathsf{acdh}, \mathsf{apcdh}, \mathsf{dlog}\}$ we define the advantage function $\mathsf{Adv}^{\mathsf{x}}_{\mathbb{G}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{x}}_{\mathbb{G}}(\mathcal{A})]$.

The following theorem captures our security result in the algebraic model.

Theorem 9. *Let $\mathbb{G}$ be a group and $\mathcal{A}$ be an adversary for $\mathsf{G}^{\mathsf{apcdh}}_{\mathbb{G}}$. Then we can construct $\mathsf{Fcs}(\mathbb{Z}_p^6, \mathbb{Z}_p)$-oracle adversaries $\mathcal{B}$ for $\mathsf{G}^{\mathsf{acdh}}_{\mathbb{G}}$ and $\mathcal{C}$ for $\mathsf{G}^{\mathrm{dlog}}_{\mathbb{G}}$ (both given in the proof) such that*

$$\mathsf{Adv}^{\mathsf{apcdh}}_{\mathbb{G}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{acdh}}_{\mathbb{G}}(\mathcal{B}) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathbb{G}}(\mathcal{C}) + (0.5 \cdot q^2_{\textsc{Pair}} + 4)/(p-1)$$

For the theorem we have simplified by not focusing on memory or time tightness. After the proof we describe how to modify the adversary $\mathcal{C}$ to achieve {AM,TM}-tight tightness.

Proof. We start by considering the adversary $\mathcal{B}$ defined in Fig. 11. It uses a random function $f' \in \mathsf{Fcs}(\mathbb{Z}_p^6, \mathbb{Z}_p)$ to simulate the $\textsc{Pair}$ oracle of $\mathcal{A}$.

```
Adversary $\mathcal{B}(X,Y)$
$f' \leftarrow\$ \mathsf{Fcs}(\mathbb{Z}_p^6, \mathbb{Z}_p)$
$[Z]_{\vec{z}} \leftarrow\$ \mathcal{A}^{\textsc{SimPair}}(X,Y)$
Return $[Z]_{\vec{z}}$

$\textsc{SimPair}([A]_{\vec{a}}, [B]_{\vec{b}})$
Require $A = g^{a_1} X^{a_2} Y^{a_3}$
Require $B = g^{b_1} X^{b_2} Y^{b_3}$
$c_1 \leftarrow a_1 b_1$
$c_2 \leftarrow (a_1 b_2 + a_2 b_1)$
$c_3 \leftarrow (a_1 b_3 + b_1 a_3)$
$c_4 \leftarrow (a_2 b_3 + a_3 b_2)$
$c_5 \leftarrow a_2 b_2$
$c_6 \leftarrow a_3 b_3$
Return $f'(c_1, c_2, c_3, c_4, c_5, c_6)$
```

```
Adversary $\mathcal{C}(U)$
$f' \leftarrow\$ \mathsf{Fcs}(\mathbb{Z}_p^6, \mathbb{Z}_p)$
$w_x, w_y \leftarrow\$ \mathbb{Z}_p^*$
$v_x, v_y \leftarrow\$ \mathbb{Z}_p$
$X \leftarrow U^{w_x} \mathbf{g}^{v_x}$
$Y \leftarrow U^{w_y} \mathbf{g}^{v_y}$
$\mathcal{P} \leftarrow \varnothing$
$([Z]_{\vec{z}}) \leftarrow\$ \mathcal{A}^{\textsc{SimPair}}(X,Y)$
For $P, P' \in \mathcal{P}$ do
    $Q \leftarrow [P - P'](w_x \cdot \mathbf{u} + v_x, w_y \cdot \mathbf{u} + v_h)$
    $(u_1, u_2) \leftarrow \mathsf{Zeros}(Q)$
    If $\mathbf{g}^{u_1} = U$ then return $u_1$
    If $\mathbf{g}^{u_2} = U$ then return $u_2$
Return 0
```

```
$\textsc{SimPair}([A]_{\vec{a}}, [B]_{\vec{b}})$
Require $A = g^{a_1} X^{a_2} Y^{a_3}$
Require $B = g^{b_1} X^{b_2} Y^{b_3}$
$\mathcal{P} \leftarrow \mathcal{P} \cup \{P_{\vec{a},\vec{b}}\}$
$c_1 \leftarrow a_1 b_1$
$c_2 \leftarrow (a_1 b_2 + a_2 b_1)$
$c_3 \leftarrow (a_1 b_3 + b_1 a_3)$
$c_4 \leftarrow (a_2 b_3 + a_3 b_2)$
$c_5 \leftarrow a_2 b_2 x^2$
$c_6 \leftarrow a_3 b_3 y^2$
Return $f'(c_1, c_2, c_3, c_4, c_5, c_6)$
```

Fig. 11. Adversaries used for proof of Thm. 9. Polynomial $P_{\vec{a},\vec{b}}$ is defined in the text of the proof. Zeros computes the (at most two) zeros of the quadratic polynomial Q.

The true Pair oracle uses a function $f \in \mathsf{Inj}(\mathbb{Z}_p, \mathbb{Z}_p)$ and returns $f(\mathrm{dlog}(A) \cdot \mathrm{dlog}(B))$. Using the representations of A and B provided by $\mathcal{A}$ when querying this oracle, we can write this as $f((a_1 + a_2 x + a_3 y)(b_1 + b_2 x + b_3 y))$. We can expand that input to f as $a_1b_1 + (a_1b_2 + a_2b_1)x + (a_1b_3 + b_1a_3)y + (a_2b_3 + a_3b_2)xy + a_2b_2x^2 + a_3b_3y^2$. We think of this as a degree-two polynomial $P_{\vec{a},\vec{b}} \in \mathbb{Z}_p[\mathbf{x}, \mathbf{y}]$ evaluated at the point $(\mathbf{x}, \mathbf{y}) = (x, y)$. Adversary $\mathcal{C}$ computes the coefficients $(c_1, \ldots, c_6)$ of this polynomial and returns $f'(c_1, \ldots, c_6)$.

There are two ways that $\mathcal{A}$ could recognize the imperfection of this simulation, by detecting that $\mathcal{C}$ is using a random function, rather than an injection or by detecting that $\mathcal{C}$ has changed the input to this function. Letting H_0 denote a hybrid game which is identical to $\mathsf{G}^{\mathsf{apcdh}}_{\mathbb{G}}(\mathcal{A})$ except that f is drawn from $\mathsf{Fcs}(\mathbb{Z}_p, \mathbb{Z}_p)$ rather than $\mathsf{Inj}(\mathbb{Z}_p, \mathbb{Z}_p)$. By the switching lemma (Lemma 1), we have that $\Pr[\mathsf{G}^{\mathsf{apcdh}}_{\mathbb{G}}(\mathcal{A})] \leqslant \Pr[\mathsf{H}_0] + 0.5 \cdot q^2_{\mathrm{PAIR}}/p$.

Now let H_1 denote a hybrid game in which the pair oracle is as simulated by $\mathcal{C}$. The only way that H_0 and H_1 can be distinguished is if $\mathcal{A}$ makes oracle queries which give distinct polynomials that have the same output on (x, y). Stated more formally, if $\mathcal{A}$ makes a pair queries with inputs $([A]_{\vec{a}}, [B]_{\vec{b}})$ and $([A']_{\vec{a}'}, [B']_{\vec{b}'})$ such that $P_{\vec{a},\vec{b}} \neq P_{\vec{a}',\vec{b}'}$, yet $P_{\vec{a},\vec{b}}(x, y) = P_{\vec{a}',\vec{b}'}(x, y)$. Letting bad denote the event that this condition holds in H_1 for any pair of queries made by $\mathcal{A}$, we have that $\Pr[\mathsf{H}_0] \leqslant \Pr[\mathsf{H}_1] + \Pr[\mathsf{bad}]$.

Let H'_1 denote a game which is identical to H_1 except x, y are sampled from $\mathbb{Z}_p$, rather than $\mathbb{Z}_p^*$ and let bad' denote the probability of the bad event in this game. Then $\Pr[\mathsf{bad}] \leqslant \Pr[\mathsf{bad}'] + 2/p$.

So to complete the proof, we need only provide a bound on $\Pr[\mathsf{bad}']$. We do so using the discrete log adversary $\mathcal{C}$ shown in Fig. 11. Given a discrete log challenge U it samples its own $w_x, w_y \leftarrow\$ \mathbb{Z}_p^*$ and $v_x, v_y \leftarrow\$ \mathbb{Z}_p$ and defines $X \leftarrow U^{w_x}\mathbf{g}^{v_x}$ and $Y \leftarrow U^{w_y}\mathbf{g}^{v_y}$. It then runs $\mathcal{A}$ on input X, Y and simulates its oracle exactly as $\mathcal{B}$ does, thus perfectly mirroring the view of $\mathcal{A}$ in H'_1.

While simulating the view of $\mathcal{A}$, it uses a set $\mathcal{P}$ to store the polynomials $P_{\vec{a},\vec{b}}$ corresponding to each of $\mathcal{A}$'s oracle queries. If bad' ever occurs, then $\mathcal{P}$ will contain polynomials P, P' for which $P \neq P'$, yet $P(x, y) = P'(x, y)$. At the end of its execution, $\mathcal{C}$ iterates over all pairs of polynomials in $\mathcal{P}$, hoping that the above holds for some pair and trying to use them to solve for u.

Letting $u = \mathrm{dlog}(U)$, note that $x = uw_x + v_x$ and $y = uw_y + v_y$. Let P, P' denote a pair of polynomials satisfying the above conditions. When it reaches them in the loop, adversary $\mathcal{C}$ defines a new polynomial Q by $[P - P'](w_x \cdot \mathbf{u} + v_x, w_y \cdot \mathbf{u} + v_h)$. Thus Q is (at most) quadratic polynomial in the formal variable $\mathbf{u}$. Because $P(x, y) = P'(x, y)$, we have that $Q(u) = 0$. Assuming Q is not the zero polynomial, it has at most two zeros. So $\mathcal{C}$ can factor Q to obtain its zeros and check which is u. We use Zero in pseudocode to denote this process (assuming it outputs $u_1 = u_2 = 0$ if $Q = 0$).

It remains to bound the probability that $Q = 0$ when $P \neq P'$. We can first think of Q as a polynomial from $(\mathbb{Z}_p[\mathbf{w}_x, \mathbf{v}_x, \mathbf{w}_y, \mathbf{v}_y])[\mathbf{u}]$, i.e. treat w_x, v_x, w_y, v_y as formal variables so that Q is a polynomial of $\mathbf{u}$ with coefficients that are polynomials of $\mathbf{w}_x, \mathbf{v}_x, \mathbf{w}_y, \mathbf{v}_y$. From [BFL20, Lemma 2.1], the coefficient of Q of highest degree is a polynomial of $\mathbf{w}_x, \mathbf{w}_y$ with degree equal to that of $P - P'$ (which is non-zero from our assumed conditions on P, P'). We can bound the probability that Q is zero by the probability this coefficient is zero. For this, note that v_x, v_y act as "one-time pads", so the view of $\mathcal{A}$ is independent of $\mathbf{w}_x, \mathbf{w}_y$. Thus we can use the Schwartz-Zippel lemma to bound the probability Q is zero by $2/(p-1)$. Putting this reasoning together gives $\Pr[\mathsf{bad}'] \leqslant \mathsf{Adv}^{\mathrm{dlog}}_{\mathbb{G}}(\mathcal{C}) + 2/(p-1)$. □

Achieving Limited Tightness. The result above is advantage-tight, but neither time- nor memory-tight because $\mathcal{C}$ uses additional memory to store all of the polynomials and then iterates over all pairs of them at the end of execution.

We could modify $\mathcal{C}$ to achieve TM-tightness by having it pick two of the Pair oracle queries at random and assume that they give colliding polynomials. Storing only those to polynomials in $\mathcal{P}$ would make it use only slightly more time and memory than $\mathcal{A}$, but we would have only been able to show that $\Pr[\mathsf{bad}'] \leqslant q^2_{\mathrm{Pair}} \mathsf{Adv}^{\mathrm{dlog}}_{\mathbb{G}}(\mathcal{C}^*) + 2/(p-1)$ where $\mathcal{C}^*$ is this modified version of $\mathcal{C}$.

To achieve AM-tightness, we can check every pair of queries for collisions using the memory-tight rewinding technique of Auerbach, et al [ACFK17]. Namely, we have $\mathcal{C}$ remember all the coins it uses by picking them using a random function. Then each time $\mathcal{A}$ makes a new Pair oracle query, $\mathcal{C}$ pauses $\mathcal{A}$ and runs a second copy of that adversary from the start of its execution until it reaches the same Pair query. While running this second copy, each time it makes a Pair oracle query we try to solve for u using the polynomials corresponding to this query and to the Pair query in the first copy of $\mathcal{A}$ that we paused at. If $\mathcal{A}$ makes q oracle queries, then the runtime of this $\mathcal{C}$ should be $O(q \cdot \mathbf{Time}(\mathcal{A}))$.

Extending to Generic Group Models. To obtain bound in a generic group model [Sho97, Mau05], we can follow the same general line of reasoning as above. First, we use standard techniques to convert a given generic adversary to an algebraic, generic adversary with the same advantage. Then we can bound its advantage by the probability that $\mathcal{B}$ from above succeeds plus the probability that it causes bad to occur. Using the typical Schwartz-Zippel type of analysis applied to the generic group model, we can bound both these by $O((q^2_{\mathrm{OP}} + q^2_{\mathrm{Pair}})/p)$, where q_{OP} is the number of generic group operations the adversary performs and q_{Pair} is the number of oracle queries it makes.

B Proof of Theorem 2 (Augmented Cramer-Shoup)

We prove the security of aCS through a sequence of hybrids presented in Fig. 12. In particular, we establish the following claims that upper bound the advantage of adversary $\mathcal{A}$ as claimed in the theorem.

1. $\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aCS}}(\mathcal{A}) = 2\Pr[\mathsf{H}^1_0] - 1$
2. $\Pr[\mathsf{H}^1_0] \leqslant \Pr[\mathsf{H}^1_4] + |\mathcal{U}|(|\mathcal{U}|-1)/2^{l_1+1} + q^2_{\mathrm{Encap}}/2^{l_2+1}$
3. $\Pr[\mathsf{H}^1_4] = \Pr[\mathsf{H}^2_1]$
4. $\Pr[\mathsf{H}^2_1] \leqslant \Pr[\mathsf{H}^3_0] + q_{\mathrm{Decap}} \cdot |\mathcal{I}|/(2^{l_2}(p-1))$
5. $\Pr[\mathsf{H}^3_0] \leqslant \Pr[\mathsf{H}^3_1] + \Pr[b = 0 \wedge \mathsf{H}^3_1 \text{ sets } \mathsf{bad}]$
6. $\Pr[\mathsf{H}^3_1] \leqslant 1/2$
7. $\Pr[b = 0 \wedge \mathsf{H}^3_1 \text{ sets } \mathsf{bad}] \leqslant \mathsf{Adv}^{\mathsf{scdh}}_{\mathbb{G}}(\mathcal{B}_{\mathsf{scdh}})/2$

Transition to H^1_0. We claim that $\mathcal{A}$'s view in H^1_0 is identical to $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aCS},b}$ (from Fig. 3), if b is chosen uniformly. The code of this game was obtained by plugging the code of aCS into $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}$, then making some small notational simplifications in preparation for future hybrids.

Note that H^1_0's final output is whether $b' = b$, so standard conditional probability calculations give that $\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{aCS}}(\mathcal{A}) = 2\Pr[\mathsf{H}^1_0] - 1$.

Transition H^1_0 to H^1_3. To transition from H^1_0 to H^1_3, we first transition to H^1_1. There each use of α_u (in Encap, Decap, or New) is chosen as the output of random injection f, rather than being sampled at random. From the Switching Lemma we get that $\Pr[\mathsf{H}^1_0] \leqslant \Pr[\mathsf{H}^1_1] + |\mathcal{U}|(|\mathcal{U}|-1)/2^{l_1+1}$. This term could more precisely be written as $q_u(q_u-1)/2^{l_1+1}$ where q_u is the number of distinct values of u that $\mathcal{A}$ queries to its oracles.

Now in switching to H^1_2, we use a tweakable random injection g to sample a and using a tweakable random function h to sample $y_{u,i}$ in Encap. The inputs to g and h do not repeat so by the Switching Lemma we get

```
Hybrids $\mathsf{H}^{\iota}_{\kappa}$
$b \leftarrow_{\$} \{0,1\}$
$x_{(\cdot)} \leftarrow_{\$} \mathbb{Z}_p^*$
$\mathcal{H}_0, \mathcal{H}_1 \leftarrow_{\$} \mathsf{aCS.IM}$
$D_1 \leftarrow \{0,1\}^{l_1+l_2} \times \mathbb{G} \times \mathbb{G}_\star$
$\tilde{\mathcal{H}} \leftarrow_{\$} \mathsf{Fcs}(\{0,1\}, D_1, \mathcal{K})$
$\alpha_{(\cdot)} \leftarrow_{\$} \{0,1\}^{l_1}$
$f^{\pm} \leftarrow_{\$} \mathsf{Inj}^{\pm}(\mathcal{U}, \{0,1\}^{l_1})$
$g^{\pm} \leftarrow_{\$} \mathsf{Inj}^{\pm}(\mathcal{U}, \mathcal{I}, \{0,1\}^{l_2})$
$h \leftarrow_{\$} \mathsf{Fcs}(\mathcal{U}, \mathcal{I}, \mathbb{Z}_p^*)$
$b' \leftarrow_{\$} \mathcal{A}^{\text{New},\text{Encap},\text{Decap},\text{H}}$
Return $(b' = b)$
```

```
Encap$(u,i)$ //$\mathsf{H}^1$
If $C[u,i] \neq \bot$:
    $(a,Y) \leftarrow C[u,i]$
    Return $((a,Y), T[u,a,Y])$
$I[u] \leftarrow I[u] \cup \{i\}$
$a \leftarrow_{\$} \{0,1\}^{l_2}$ //$\mathsf{H}^1_{[0,2)}$
$a \leftarrow g_u(i)$ //$\mathsf{H}^1_{[2,\infty)}$
$y \leftarrow_{\$} \mathbb{Z}_p^*$ //$\mathsf{H}^1_{[0,2)}$
$y \leftarrow h_u(i)$ //$\mathsf{H}^1_{[2,\infty)}$
$Y \leftarrow \mathbf{g}^y$; $Z \leftarrow Y^{x_u}$
$K^1 \leftarrow \mathcal{H}_1(\alpha_u, a, Y, Z)$ //$\mathsf{H}^1_{[0,1)}$
$K^1 \leftarrow \mathcal{H}_1(f(u), a, Y, Z)$ //$\mathsf{H}^1_{[1,\infty)}$
$K^0 \leftarrow_{\$} \mathcal{K}$ //$\mathsf{H}^1_{[0,3)}$
$K^0 \leftarrow \mathcal{H}_0(f(u), a, Y, Z)$ //$\mathsf{H}^1_{[3,\infty)}$
$T[u,a,Y] \leftarrow K^b$
$C[u,i] \leftarrow (a,Y)$
Return $((a,Y), K^b)$
```

```
Decap$(u,a,Y)$ //$\mathsf{H}^1$
$i \leftarrow g_u^{-1}(a)$ //$\mathsf{H}^1_{[4,\infty)}$
If $i \in I[u]$ and $(a,Y) = C[u,i]$: //$\mathsf{H}^1_{[4,\infty)}$
If $T[u,a,Y] \neq \bot$: //$\mathsf{H}^1_{[0,4)}$
    Return $T[u,a,Y]$
$Z \leftarrow Y^{x_u}$
Return $\mathcal{H}_1(\alpha_u, a, Y, Z)$ //$\mathsf{H}^1_{[0,1)}$
Return $\mathcal{H}_1(f(u), a, Y, Z)$ //$\mathsf{H}^1_{[1,\infty)}$

New$(u)$ //$\mathsf{H}^1$
Return $(\alpha_u, \mathbf{g}^{x_u})$ //$\mathsf{H}^1_{[0,1)}$
Return $(f(u), \mathbf{g}^{x_u})$ //$\mathsf{H}^1_{[1,\infty)}$

H$(\alpha,a,Y,Z)$ //$\mathsf{H}^1$
Return $\mathcal{H}_1(\alpha, a, Y, Z)$
```

```
Encap$(u,i)$ //$\mathsf{H}^2$
If $C[u,i] \neq \bot$:
    $(a,Y) \leftarrow C[u,i]$
Else
    $a \leftarrow g_u(i)$; $y \leftarrow h_u(i)$
    $Y \leftarrow \mathbf{g}^y$
$I[u] \leftarrow I[u] \cup \{i\}$
$Z \leftarrow Y^{x_u}$
$K^b \leftarrow \tilde{\mathcal{H}}_b(\lambda(f(u), a, Y, Z))$ //$\mathsf{H}^2_{[0,1)}$
$K^b \leftarrow \tilde{\mathcal{H}}_b(f(u), a, Y, \star)$ //$\mathsf{H}^2_{[1,\infty)}$
$T[u,a,Y] \leftarrow K^b$
$C[u,i] \leftarrow (a,Y)$
Return $((a,Y), K^b)$
```

```
Decap$(u,a,Y)$ //$\mathsf{H}^2$
$i \leftarrow g_u^{-1}(a)$
If $i \in I[u]$ and $(a,Y) = C[u,i]$:
    Return $T[u,a,Y]$ //$\mathsf{H}^2_{[0,1)}$
    Return $\tilde{\mathcal{H}}_b(f(u), a, Y, \star)$ //$\mathsf{H}^2_{[1,\infty)}$
$Z \leftarrow Y^{x_u}$
Return $\tilde{\mathcal{H}}_1(\lambda(f(u), a, Y, Z))$ //$\mathsf{H}^2_{[0,1)}$
Return $\tilde{\mathcal{H}}_1(f(u), a, Y, \star)$ //$\mathsf{H}^2_{[1,\infty)}$

New$(u)$ //$\mathsf{H}^2$
Return $(f(u), \mathbf{g}^{x_u})$
```

```
H$(\alpha,a,Y,Z)$ //$\mathsf{H}^2$
Return $\tilde{\mathcal{H}}_1(\lambda(\alpha, a, Y, Z))$ //$\mathsf{H}^2_{[0,1)}$
$u \leftarrow f^{-1}(\alpha)$ //$\mathsf{H}^2_{[1,\infty)}$
If $u \neq \bot$ and $Z = Y^{x_u}$: //$\mathsf{H}^2_{[1,\infty)}$
    Return $\tilde{\mathcal{H}}_1(\alpha, a, Y, \star)$ //$\mathsf{H}^2_{[1,\infty)}$
Return $\tilde{\mathcal{H}}_1(\alpha, a, Y, Z)$ //$\mathsf{H}^2_{[1,\infty)}$

Injection $\lambda(\alpha,a,Y,Z)$ //Internal, $\mathsf{H}^2$
$u \leftarrow f^{-1}(\alpha)$
If $u \neq \bot$ and $Z = Y^{x_u}$:
    Return $(\alpha, a, Y, \star)$
Return $(\alpha, a, Y, Z)$
```

```
Encap$(u,i)$ //$\mathsf{H}^3$
$a \leftarrow g_u(i)$
$Y \leftarrow \mathbf{g}^{h_u(i)}$
$K^b \leftarrow \tilde{\mathcal{H}}_b(f(u), a, Y, \star)$
Return $((a,Y), K^b)$
```

```
Decap$(u,a,Y)$ //$\mathsf{H}^3$
$i \leftarrow g_u^{-1}(a)$; $Y_{u,i} \leftarrow \mathbf{g}^{h_u(i)}$
If $i \neq \bot$ and $a = g_u(i)$ and $Y = Y_{u,i}$:
    Return $\tilde{\mathcal{H}}_b(f(u), a, Y, \star)$
Return $\tilde{\mathcal{H}}_1(f(u), a, Y, \star)$

New$(u)$
Return $(f(u), \mathbf{g}^{x_u})$
```

```
H$(\alpha,a,Y,Z)$ //$\mathsf{H}^3$
$u \leftarrow f^{-1}(\alpha)$
If $u \neq \bot$ and $Z = Y^{x_u}$
    $i \leftarrow g_u^{-1}(a)$ //$\mathsf{H}^3_{[1,\infty)}$
    If $i \neq \bot$ and $Y = \mathbf{g}^{h_u(i)}$: //$\mathsf{H}^3_{[1,\infty)}$
        $\mathsf{bad} \leftarrow \mathsf{true}$ //$\mathsf{H}^3_{[1,\infty)}$
        Return $\tilde{\mathcal{H}}_b(\alpha, a, Y, \star)$ //$\mathsf{H}^3_{[1,\infty)}$
    Return $\tilde{\mathcal{H}}_1(\alpha, a, Y, \star)$
Return $\tilde{\mathcal{H}}_1(\alpha, a, Y, Z)$
```

Fig. 12. Hybrids games used in proof of Theorem 2 (TAM-tight security of augmented Cramer-Shoup). Oracles labelled "internal" are not accessible to the adversary. Grey highlighting indicates changes from earlier games.

```
Adversary $\mathcal{B}_{\mathsf{scdh}}^{\textsc{New},\textsc{Chal},\textsc{Strong}}$
$D_1 \leftarrow \{0,1\}^{l_1+l_2} \times \mathbb{G} \times \mathbb{G}_\star$
$\tilde{\mathcal{H}} \leftarrow\$ \mathsf{Fcs}(D_1, \mathcal{K})$
$f^\pm \leftarrow\$ \mathsf{Inj}^\pm(\mathcal{U}, \{0,1\}^{l_1})$
$g^\pm \leftarrow\$ \mathsf{Inj}^\pm(\mathcal{U}, \mathcal{I}, \{0,1\}^{l_2})$
$b' \leftarrow\$ \mathcal{A}^{\textsc{SimNew},\textsc{SimEncap},\textsc{SimDecap},\textsc{SimH}}$
Return $\perp$

$\textsc{SimNew}(u)$
Return $(f(u), \textsc{New}(u))$
```

```
$\textsc{SimEncap}(u, i)$
$a \leftarrow g_u(i)$
$Y \leftarrow \textsc{Chal}(u, i)$
$K \leftarrow \tilde{\mathcal{H}}(f(u), a, Y, \star)$
Return $((a, Y), K)$

$\textsc{SimDecap}(u, a, Y)$
$i \leftarrow g_u^{-1}(a)$; $Y_{u,i} \leftarrow \textsc{Chal}(u, i)$
If $i \neq \perp$ and $a = g_u(i)$ and $Y = Y_{u,i}$:
  Return $\tilde{\mathcal{H}}(f(u), a, Y, \star)$
Return $\tilde{\mathcal{H}}(f(u), a, Y, \star)$
```

```
$\textsc{SimH}(\alpha, a, Y, Z)$
$u \leftarrow f^{-1}(\alpha)$
If $u \neq \perp$ and $\textsc{Strong}(u, Y, Z)$:
  $i \leftarrow g_u^{-1}(a)$
  If $i \neq \perp$ and $Y = \textsc{Chal}(u, i)$:
    $\mathsf{OUTPUT}(u, i, Z)$
  Return $\tilde{\mathcal{H}}(\alpha, a, Y, \star)$
Return $\tilde{\mathcal{H}}(\alpha, a, Y, Z)$
```

Fig. 13. Strong CDH adversary $\mathcal{B}_{\mathsf{scdh}}$ used for proof Theorem 2 (TAM-tight security of augmented Cramer-Shoup).

that $\Pr[\mathsf{H}_1^1] \leqslant \Pr[\mathsf{H}_2^1] + q_{\textsc{Encap}}^2/2^{l_2+1}$. This term could more precisely be written as $q_{u,i}^2/2^{l_2+1}$ or $q_u q_i^2/2^{l_2+1}$ where $q_{u,i}$ is the number of distinct values of (u, i) that $\mathcal{A}$ queries to its $\textsc{Encap}$ oracle and q_i is the maximum number of distinct values of i that $\mathcal{A}$ queries to any user's $\textsc{Encap}(u, \cdot)$ oracle.

In H_3^1, we switch from sampling K_0 uniformly at random to assigning it the output of the random function $\mathcal{H}_0(f(u), a, Y, Z)$. Note that both f and g_u are random injections and the condition at the beginning of $\textsc{Encap}$ ensures that the inputs to $\mathcal{H}_0$ do not repeat. Therfore, $\Pr[\mathsf{H}_2^1] = \Pr[\mathsf{H}_3^1]$.

Finally, in the transition to H_4^1 we replace the check $T[u, a, Y] \neq \perp$ with ($i \in I[u]$ and $(a, Y) = C[u, i]$), where $i = g_u^{-1}(a)$. Both of these booleans capture exactly whether (a, Y) was a challenge ciphertext returned by an earlier query $\textsc{Encap}(u, i)$ and so are identical. Hence $\Pr[\mathsf{H}_3^1] = \Pr[\mathsf{H}_4^1]$.

Transition H_4^1 to H_1^2 (Map-then-rf). Next we move to the hybrid H_0^2. We have highlighted the interesting ways in which H_0^2 differs from H_4^1. We've also reorganized $\textsc{Encap}$ slightly. The main difference is that $\mathcal{H}$ hash been replaced by $\tilde{\mathcal{H}}_1 \circ \lambda$. Here λ is the defined by

$$\lambda(\alpha, a, Y, Z) = \begin{cases} (\alpha, a, Y, \star), & \text{if } Z = Y^x, \text{ where } x = x_{f^{-1}(\alpha)}. \\ (\alpha, a, Y, Z), & \text{otherwise.} \end{cases}$$

As $\tilde{\mathcal{H}}_1$ is random and λ is an injection, this is equivalent to using the random function $\mathcal{H}$ so $\Pr[\mathsf{H}_3^1] = \Pr[\mathsf{H}_0^2]$.

In H_1^2, we apply the logic of λ. In both $\textsc{Encap}$ and $\textsc{Decap}$, the value Z was just defined as Y^{x_u} so we hit the first case, getting $\lambda(f(u), a, Y, Z) = (f(u), a, Y, \star)$. In H, we've directly added code to check if the first or the second condition of λ holds. Therefore, this doesn't change the adversary's view.

Inside of $\textsc{Decap}$ of H_1^2, when $T[u, a, Y] \neq \perp$ we return $\tilde{\mathcal{H}}_b(f(u), a, Y, \star)$ in place of $T[u, a, Y]$. In $\textsc{Encap}$, we always assign the value $\tilde{\mathcal{H}}_b(f(u), a, Y, \star)$ to $T[u, a, Y]$, so this is equivalent. Hence, $\Pr[\mathsf{H}_0^2] = \Pr[\mathsf{H}_1^2]$.

Transition H_1^2 to H_0^3. Next we transition to the final set of hybrids H^3, starting with H_0^3. We highlighted important ways in which H_0^3 is different from H_1^2. In H_0^3, have removed the tables I, C, and T. Table T was not being used anywhere. Where C was being used we instead recompute $a = g_u(i)$ and $Y = \mathbf{g}^{h_u(i)}$. We could have omitted this in the decapsulation oracle as by definition of i, the check $a = g_u(i)$ will necessarily hold when $i \neq \perp$. The removal of I is the one place where our change modifies the behavior of the game. In $\textsc{Decap}$, checking if $i \in I[u]$ has been replaced with checking for $i \neq \perp$. Hence, the games differ if the adversary makes a query $\textsc{Decap}(u, a, Y)$ with $a = g_u(i)$ and $Y = \mathbf{g}^{h_u(i)}$ despite having not queried $\textsc{Encap}(u, i)$.

We can information theoretically bound the probability of this. We analyze this probability in H_1^2 where the view of the adversary only depends on values of $g_u(i)$ and $h_u(i)$ for which $i \in I[u]$. Consider a fixed point in time when $\mathcal{A}$ makes a $\textsc{Decap}(u, a, Y)$ query with an a not previously returned by an encapsulation query to u. Using this view, the adversary must guess some $a = g_u(i)$ such that $i \in \mathcal{I} \setminus I[u]$ along with the corresponding $Y = \mathbf{g}^{h_u(i)}$. There are $|\mathcal{I}|$ points in the image of g_u and 2^{l_2} points in its codomain, of which the adversary has seen $|I[u]|$ from $\textsc{Encap}$. Thus we can bound the probability that the adversary picks such an $a \in g(\mathcal{I} \backslash I[u])$ by

$$\frac{|g_u(\mathcal{I}) \smallsetminus g_u(I[u])|}{|\{0,1\}^{l_2} \smallsetminus g_u(I[u])|} = \frac{|\mathcal{I}| - |I[u]|}{2^{l_2} - |I[u]|} \leqslant \frac{|\mathcal{I}|}{2^{l_2}}.$$

The adversary must additionally have guessed the correct $\mathbf{g}^{h_u(i)}$, which it has an a $1/(p-1)$ chance of having done (as h is a random function). Applying a union bound across all Decap queries gives the bound $\Pr[\mathsf{H}_1^2] \leqslant \Pr[\mathsf{H}_0^3] + q_{\text{Decap}} \cdot |\mathcal{I}|/(2^{l_2}(p-1))$.

Transition H_0^3 to H_1^3. Finally, in H_1^3 the code of H has been modified to add an extra conditional. The conditional checks if the hash query corresponds to any ciphertexts that could possibly be returned by Encap. If so, the flag bad is set and the oracle returns $\tilde{\mathcal{H}}_b(\alpha, a, Y, \star)$. Note this only changes anything when $b = 0$. Thus H_0^3 and H_1^3 are identical-until-bad which implies that $\Pr[\mathsf{H}_0^3] \leqslant \Pr[\mathsf{H}_1^3] + \Pr[b = 0 \wedge \mathsf{H}_1^3 \text{ sets } \mathsf{bad}]$.

Now in H_1^3, the games only depends on b for determining which $\tilde{\mathcal{H}}_b$ is used in Encap, Decap, and H. In all cases this is for inputs on the form $(f(u), g_u(i), \mathbf{g}^{h_u(i)}, \star)$. These oracles are always consistent in which $\tilde{\mathcal{H}}_b$ to use for such inputs so the view of $\mathcal{A}$ is independent of b and $\Pr[\mathsf{H}_1^3] \leqslant 1/2$.

Probability of Bad. To conclude, we bound $\Pr[b = 0 \wedge \mathsf{H}_1^3 \text{ sets } \mathsf{bad}]$. When bad is set, the value Z is equal to $\mathbf{g}^{x_u \cdot y_{u,i}}$ where x_u is a secret decryption key and $y_{u,i}$ is a secret exponent chosen inside an encapsulation query $\text{Encap}(u,i)$. We can then use the ability to set bad to win the Strong CDH game. This is captured by the adversary $\mathcal{B}_{\mathsf{scdh}}$ given in Fig. 13. It simulates the view of $\mathcal{A}$ in H_1^3, using its own New oracle to produce encryption keys of users, its Chal oracle to obtain $\mathbf{g}^{y_{u,i}}$ values, and Strong in simulating the hash function to check when $Z = Y^{x_u}$. Whenever bad would be set, $\mathcal{B}_{\mathsf{scdh}}$ halts immediately. By comparing the code we can see that $\mathcal{B}_{\mathsf{scdh}}$ perfectly simulates the view of $\mathcal{A}$ (up until bad would be set)[10] and it wins its own game whenever bad would be set. Hence, $\Pr[\mathsf{H}_1^3 \text{ sets } \mathsf{bad} | b = 0] \leqslant \mathsf{Adv}_{\mathbb{G}}^{\mathsf{scdh}}(\mathcal{B}_{\mathsf{scdh}})$. Clearly $\Pr[b = 0] = 1/2$. □

C Proof of Theorem 3 (Augmented Twin ElGamal)

We prove Theorem 3 through a sequence of hybrids H_0^1 through H_4^1, H_0^2 through H_1^2, and H_0^3 through H_1^3 presented in Fig. 14. The following claims hold and upper bound the advantage of adversary $\mathcal{A}$ as claimed in the theorem.

1. $\mathsf{Adv}_{\mathsf{aTWIN}}^{\mathsf{mu\text{-}\$cca}}(\mathcal{A}) = 2\Pr[\mathsf{H}_0^1] - 1$
2. $\Pr[\mathsf{H}_0^1] \leqslant \Pr[\mathsf{H}_4^1] + |\mathcal{U}|(|\mathcal{U}|-1)/2^{l_1+1} + q_{\text{Encap}}^2/2^{l_2+1}$
3. $\Pr[\mathsf{H}_4^1] = \Pr[\mathsf{H}_1^2]$
4. $\Pr[\mathsf{H}_1^2] \leqslant \Pr[\mathsf{H}_0^3] + q_{\text{Decap}} \cdot |\mathcal{I}|/(2^{l_2}(p-1))$
5. $\Pr[\mathsf{H}_0^3] \leqslant \Pr[\mathsf{H}_1^3] + \Pr[b = 0 \wedge \mathsf{H}_1^3 \text{ sets } \mathsf{bad}]$
6. $\Pr[\mathsf{H}_1^3] = 1/2$
7. $\Pr[b = 0 \wedge \mathsf{H}_1^3 \text{ sets } \mathsf{bad}] \leqslant \mathsf{Adv}_{\mathbb{G}}^{\mathsf{stcdh}}(\mathcal{B}_{\mathsf{stcdh}})/2$

The sequence of hybrids is basically identical to those used in the previous section (Sec. B) to prove the security of aCS. The only changes are that where those earlier hybrids used the group element Z, the new hybrids now have both Z and Ψ. Because of this similarity, we do not provide individual descriptions to justify the claimed bounds. The interested reader can understand each transition by reading the corresponding text from the prior section while considering Figure 14. □

D Proof of Theorem 4 (Augmented aU$^{\perp}$ Transform)

We prove Theorem 4 through a sequence of hybrids H_0^1 through $\mathsf{H}_4^1, \mathsf{H}_0^2$ through H_1^2, and H_0^3 through H_1^3 presented in Fig. 16 where we establish the following claims that upper bound the advantage of adversary $\mathcal{A}$ as stated in the theorem.

1. $\mathsf{Adv}_{\mathsf{aUEM}}^{\mathsf{mu\text{-}cca}}(\mathcal{A}) = 2\Pr[\mathsf{H}_0^1] - 1$
2. $\Pr[\mathsf{H}_0^1] \leqslant \Pr[\mathsf{H}_4^1] + |\mathcal{U}|(|\mathcal{U}|-1)/2^{l_1+1} + q_{\text{Encap}}^2/2^{l_2+1}$
3. $\Pr[\mathsf{H}_4^1] = \Pr[\mathsf{H}_0^2]$
4. $\Pr[\mathsf{H}_0^2] \leqslant \Pr[\mathsf{H}_1^2] + q_{\text{Encap}} \cdot \delta$
5. $\Pr[\mathsf{H}_1^2] \leqslant \Pr[\mathsf{H}_0^3] + q_{\text{Decap}} \cdot |\mathcal{I}|/2^{l_2 + \mathsf{H}_\infty(\mathsf{KEM})}$
7. $\Pr[\mathsf{H}_0^3] \leqslant \Pr[\mathsf{H}_1^3] + \Pr[b = 0 \wedge \mathsf{H}_1^3 \text{ sets } \mathsf{bad}]$
8. $\Pr[\mathsf{H}_1^3] \leqslant 1/2$
9. $\Pr[b = 0 \wedge \mathsf{H}_1^3 \text{ sets } \mathsf{bad}] \leqslant \mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{mu\text{-}ow\text{-}pcva}}(\mathcal{B})/2$

[10] Note that $\mathcal{B}_{\mathsf{scdh}}$ uses a single function $\tilde{\mathcal{H}}$, rather than separate $\tilde{\mathcal{H}}_1$ and $\tilde{\mathcal{H}}_0$. This is undetectable by the same logic used to establish that $\Pr[\mathsf{H}_1^3] = 1/2$.

```
Hybrids $\mathsf{H}^{\iota}_{\kappa}$
$b \leftarrow\!\$\ \{0,1\}$
$w_{(\cdot)}, x_{(\cdot)} \leftarrow\!\$\ \mathbb{Z}_p^*$
$\mathcal{H}_0, \mathcal{H}_1 \leftarrow\!\$\ \mathsf{aTWIN.IM}$
$D_1 \leftarrow \{0,1\}^{l_1+l_2} \times \mathbb{G} \times (\mathbb{G}_\star)^2$
$\tilde{\mathcal{H}} \leftarrow\!\$\ \mathsf{Fcs}(\{0,1\}, D_1, \mathcal{K})$
$\alpha_{(\cdot)} \leftarrow\!\$\ \{0,1\}^{l_1}$
$f^{\pm} \leftarrow\!\$\ \mathsf{Inj}^{\pm}(\mathcal{U}, \{0,1\}^{l_1})$
$g^{\pm} \leftarrow\!\$\ \mathsf{Inj}^{\pm}(\mathcal{U}, \mathcal{I}, \{0,1\}^{l_2})$
$h \leftarrow\!\$\ \mathsf{Fcs}(\mathcal{U}, \mathcal{I}, \mathbb{Z}_p^*)$
$b' \leftarrow\!\$\ \mathcal{A}^{\textsc{New},\textsc{Encap},\textsc{Decap},\mathrm{H}}$
Return $(b' = b)$
```

```
$\textsc{Encap}(u,i)$ $//\mathsf{H}^1$
If $C[u,i] \neq \bot$:
   $(a,Y) \leftarrow C[u,i]$
   Return $((a,Y), T[u,a,Y])$
$I[u] \leftarrow I[u] \cup \{i\}$
$a \leftarrow\!\$\ \{0,1\}^{l_2}$ $//\mathsf{H}^1_{[0,2)}$
$a \leftarrow g_u(i)$ $//\mathsf{H}^1_{[2,\infty)}$
$y \leftarrow\!\$\ \mathbb{Z}_p^*$ $//\mathsf{H}^1_{[0,2)}$
$y \leftarrow h_u(i)$ $//\mathsf{H}^1_{[2,\infty)}$
$Y \leftarrow \mathbf{g}^y$; $Z \leftarrow Y^{w_u}$; $\Psi \leftarrow Y^{x_u}$
$K^1 \leftarrow \mathcal{H}_1(\alpha_u, a, Y, Z, \Psi)$ $//\mathsf{H}^1_{[0,1)}$
$K^1 \leftarrow \mathcal{H}_1(f(u), a, Y, Z, \Psi)$ $//\mathsf{H}^1_{[1,\infty)}$
$K^0 \leftarrow\!\$\ \mathcal{K}$ $//\mathsf{H}^1_{[0,3)}$
$K^0 \leftarrow\!\$\ \mathcal{H}_0(f(u), a, Y, Z, \Psi)$ $//\mathsf{H}^1_{[0,3)}$
$T[u,a,Y] \leftarrow K^b$
$C[u,i] \leftarrow (a,Y)$
Return $((a,Y), K^b)$
```

```
$\textsc{Decap}(u,a,Y)$ $//\mathsf{H}^1$
$i \leftarrow g_u^{-1}(a)$ $//\mathsf{H}^1_{[4,\infty)}$
If $i \in I[u]$ and $(a,Y) = C[u,i]$:$//\mathsf{H}^1_{[4,\infty)}$
If $T[u,a,Y] \neq \bot$: $//\mathsf{H}^1_{[0,4)}$
   Return $T[u,a,Y]$
$Z \leftarrow Y^{w_u}; \Psi \leftarrow Y^{x_u}$
Return $\mathcal{H}_1(\alpha_u, a, Y, Z, \Psi)$ $//\mathsf{H}^1_{[0,1)}$
Return $\mathcal{H}_1(f(u), a, Y, Z, \Psi)$ $//\mathsf{H}^1_{[1,\infty)}$

$\textsc{New}(u)$ $//\mathsf{H}^1$
Return $(\alpha_u, \mathbf{g}^{w_u}, \mathbf{g}^{x_u})$ $//\mathsf{H}^1_{[0,1)}$
Return $(f(u), \mathbf{g}^{w_u}, \mathbf{g}^{x_u})$ $//\mathsf{H}^1_{[1,\infty)}$

$\mathrm{H}(\alpha, a, Y, Z, \Psi)$ $//\mathsf{H}^1$
Return $\mathcal{H}_1(\alpha, a, Y, Z, \Psi)$
```

```
$\textsc{Encap}(u,i)$ $//\mathsf{H}^2$
If $C[u,i] \neq \bot$:
   $(a,Y) \leftarrow C[u,i]$
Else
   $a \leftarrow g_u(i)$; $y \leftarrow h_u(i)$
   $Y \leftarrow \mathbf{g}^y$
$I[u] \leftarrow I[u] \cup \{i\}$
$Z \leftarrow Y^{w_u}$; $\Psi \leftarrow Y^{x_u}$
$K^b \leftarrow \tilde{\mathcal{H}}_b(\lambda(f(u), a, Y, Z, \Psi))$ $//\mathsf{H}^2_{[0,1)}$
$K^b \leftarrow \tilde{\mathcal{H}}_b(f(u), a, Y, \star, \star)$ $//\mathsf{H}^2_{[1,\infty)}$
$T[u,a,Y] \leftarrow K^b$
$C[u,i] \leftarrow (a,Y)$
Return $((a,Y), K^b)$

Injection $\lambda(\alpha, a, Y, Z, \Psi)$ //Internal, $\mathsf{H}^2$
$u \leftarrow f^{-1}(\alpha)$
If $u \neq \bot$ and $Z = Y^{w_u}$ and $\Psi = Y^{x_u}$:
   Return $(\alpha, a, Y, \star, \star)$
Return $(\alpha, a, Y, Z, \Psi)$
```

```
$\textsc{Decap}(u,a,Y)$ $//\mathsf{H}^2$
$i \leftarrow g_u^{-1}(a)$
If $i \in I[u]$ and $(a,Y) = C[u,i]$:
   Return $T[u,a,Y]$ $//\mathsf{H}^2_{[0,1)}$
   Return $\tilde{\mathcal{H}}_b(f(u), a, Y, \star, \star)$ $//\mathsf{H}^2_{[1,\infty)}$
$Z \leftarrow Y^{w_u}; \Psi \leftarrow Y^{x_u}$
Return $\tilde{\mathcal{H}}_1(\lambda(f(u), a, Y, Z, \Psi))$ $//\mathsf{H}^2_{[0,1)}$
Return $\tilde{\mathcal{H}}_1(f(u), a, Y, \star, \star)$ $//\mathsf{H}^2_{[1,\infty)}$

$\textsc{New}(u)$ $//\mathsf{H}^2$
Return $(f(u), \mathbf{g}^{w_u}, \mathbf{g}^{x_u})$

$\mathrm{H}(\alpha, a, Y, Z, \Psi)$ $//\mathsf{H}^2$
Return $\tilde{\mathcal{H}}_1(\lambda(\alpha, a, Y, Z, \Psi))$ $//\mathsf{H}^2_{[0,1)}$
$u \leftarrow f^{-1}(\alpha)$ $//\mathsf{H}^2_{[1,\infty)}$
If $u \neq \bot$ and $Z = Y^{w_u}$ and $\Psi = Y^{x_u}$: $//\mathsf{H}^2_{[1,\infty)}$
   Return $\tilde{\mathcal{H}}_1(\alpha, a, Y, \star, \star)$ $//\mathsf{H}^2_{[1,\infty)}$
Return $\tilde{\mathcal{H}}_1(\alpha, a, Y, Z, \Psi)$ $//\mathsf{H}^2_{[1,\infty)}$
```

```
$\textsc{Encap}(u,i)$ $//\mathsf{H}^3$
$a \leftarrow g_u(i)$
$Y \leftarrow \mathbf{g}^{h_u(i)}$
$K^b \leftarrow \tilde{\mathcal{H}}_b(f(u), a, Y, \star, \star)$
Return $((a,Y), K^b)$
```

```
$\textsc{Decap}(u,a,Y)$ $//\mathsf{H}^3$
$i \leftarrow g_u^{-1}(a)$; $Y_{u,i} \leftarrow \mathbf{g}^{h_u(i)}$
If $i \neq \bot$ and $a = g_u(i)$ and $Y = Y_{u,i}$:
   Return $\tilde{\mathcal{H}}_b(f(u), a, Y, \star, \star)$
Return $\tilde{\mathcal{H}}_1(f(u), a, Y, \star, \star)$

$\textsc{New}(u)$ $//\mathsf{H}^3$
Return $(f(u), \mathbf{g}^{w_u}, \mathbf{g}^{x_u})$
```

```
$\mathrm{H}(\alpha, a, Y, Z, \Psi)$ $//\mathsf{H}^3$
$u \leftarrow f^{-1}(\alpha)$
If $u \neq \bot$ and $Z = Y^{w_u}$ and $\Psi = Y^{x_u}$:
   $i \leftarrow g_u^{-1}(a)$ $//\mathsf{H}^3_{[1,\infty)}$
   If $i \neq \bot$ and $Y = \mathbf{g}^{h_u(i)}$: $//\mathsf{H}^3_{[1,\infty)}$
      $\mathsf{bad} \leftarrow \mathsf{true}$ $//\mathsf{H}^3_{[1,\infty)}$
      Return $\tilde{\mathcal{H}}_b(\alpha, a, Y, \star, \star)$ $//\mathsf{H}^3_{[1,\infty)}$
   Return $\tilde{\mathcal{H}}_1(\alpha, a, Y, \star, \star)$
Return $\tilde{\mathcal{H}}_1(\alpha, a, Y, Z, \Psi)$
```

Fig. 14. Hybrids games used in proof of Theorem 3 (TAM-tight security of augmented Twin ElGamal). Oracles labelled "internal" are not accessible to the adversary. Grey highlighting indicates changes from earlier games.

```
Adversary $\mathcal{B}_{\mathsf{stcdh}}^{\textsc{New},\textsc{Chal},\textsc{Strong}}$
$D_1 \leftarrow \{0,1\}^{l_1+l_2} \times \mathbb{G} \times \mathbb{G}_\star^2$
$\tilde{\mathcal{H}} \leftarrow\$\ \mathsf{Fcs}(D_1, \mathcal{K})$
$f^\pm \leftarrow\$\ \mathsf{Inj}^\pm(\mathcal{U}, \{0,1\}^{l_1})$
$g^\pm \leftarrow\$\ \mathsf{Inj}^\pm(\mathcal{U}, \mathcal{I}, \{0,1\}^{l_2})$
$b' \leftarrow\$\ \mathcal{A}^{\textsc{SimNew},\textsc{SimEncap},\textsc{SimDecap},\textsc{SimH}}$
Return $\perp$

$\textsc{SimNew}(u)$
Return $(f(u), \textsc{New}(u))$
```

```
$\textsc{SimEncap}(u, i)$
$a \leftarrow g_u(i)$
$Y \leftarrow \textsc{Chal}(u, i)$
$K \leftarrow \tilde{\mathcal{H}}(f(u), a, Y, \star, \star)$
Return $((a, Y), K)$

$\textsc{SimDecap}(u, a, Y)$
$i \leftarrow g_u^{-1}(a)$; $Y_{u,i} \leftarrow \textsc{Chal}(u, i)$
If $i \neq \perp$ and $a = g_u(i)$ and $Y = Y_{u,i}$:
    Return $\tilde{\mathcal{H}}(f(u), a, Y, \star, \star)$
Return $\tilde{\mathcal{H}}(\alpha, a, Y, \star, \star)$
```

```
$\textsc{SimH}(\alpha, a, Y, Z, \Psi)$
$u \leftarrow f^{-1}(\alpha)$
If $u \neq \perp$ and $\textsc{Strong}(u, Y, Z, \Psi)$:
    $i \leftarrow g_u^{-1}(a)$
    If $i \neq \perp$ and $Y = \textsc{Chal}(u, i)$:
        $\mathsf{OUTPUT}(u, i, Z, \Psi)$
    Return $\tilde{\mathcal{H}}(\alpha, a, Y, \star, \star)$
Return $\tilde{\mathcal{H}}(\alpha, a, Y, Z, \Psi)$
```

Fig. 15. Adversary $\mathcal{B}_{\mathsf{stcdh}}$ used for Theorem 3 (TAM-tight security of augmented Twin ElGamal).

Transition to H_0^1. The hybrid H_0^1 is obtained by substituting the code for aUEM into the game $\mathsf{G}_{\mathsf{aUEM},b}^{\mathsf{mu\text{-}cca}}$. For notational convenience, we wrote KEM's ideal model $\mathcal{H}'$ as a separate oracle from the new random oracle added by $\mathsf{aU}^{\perp}$. The bit b is chosen uniformly and H_0^1 checks if $b = b'$, so standard probability calculations give $\mathsf{Adv}_{\mathsf{aUEM}}^{\mathsf{mu\text{-}cca}}(\mathcal{A}) = 2\Pr[\mathsf{H}_0^1] - 1$.

Transitions H_0^1 to H_4^1. We first transition to H_1^1, where instead of sampling the string α at random, we assign it as the output of the random injection $f(u)$. Using the Switching Lemma, we have $\Pr[\mathsf{H}_0^1] \leqslant \Pr[\mathsf{H}_1^1] + |\mathcal{U}|(|\mathcal{U}| - 1)/2^{l_1+1}$. This term could more precisely be written as $q_u(q_u - 1)/2^{l_1+1}$ where q_u is the number of distinct values of u that $\mathcal{A}$ queries to its oracles.

In H_2^1, we modify the Encap oracle where we switch from sampling a uniformly to assigning it the output of a tweakable random injection $g_u(\cdot)$. As the input to g_u does not repeat, the Switching Lemma gives $\Pr[\mathsf{H}_1^1] \leqslant \Pr[\mathsf{H}_2^1] + q_{\textsc{Encap}}^2/2^{l_2+1}$. This term could more precisely be written as $q_{u,i}^2/2^{l_2+1}$ or $q_u q_i^2/2^{l_2+1}$ where $q_{u,i}$ is the number of distinct values of (u, i) that $\mathcal{A}$ queries to its Encap oracle and q_i is the maximum number of distinct values of i that $\mathcal{A}$ queries to any user's $\textsc{Encap}(u, \cdot)$ oracle.

In hybrid H_3^1, we switch from sampling K_0 at random to assigning it the output of a random function $\mathcal{H}_0(f(u), a, c, k)$. Because f and g_u are injections applied to u and i respectively, the inputs to $\mathcal{H}_0$ never repeat. Therefore, sampling K_0 at random is the same as computing it as the output of $\mathcal{H}_0$ and $\Pr[\mathsf{H}_2^1] = \Pr[\mathsf{H}_3^1]$.

Finally, in the transition to H_4^1, we modify the Decap oracle where we switch from checking if $T[u, a, c] \neq \perp$ to checking if ($i \in I[u]$ and $C[u, i] = (a, c)$) where $i = g_u^{-1}(a)$. These both check if (a, c) was returned by Encap for some user u and are equivalent. Therefore, $\Pr[\mathsf{H}_3^1] = \Pr[\mathsf{H}_4^1]$.

Transitions H_0^2 to H_1^2 (map-then-rf). We have highlighted the interesting differences between H_4^1 and H_0^2 in grey. We have replaced all invocations of $\mathcal{H}_b$ with $\tilde{\mathcal{H}}_b \circ \lambda$ where λ is the injection

$$\lambda(\alpha, a, c, k) = \begin{cases} (\alpha, a, c, \star), & \text{if } \mathrm{PC}(u, k, c) = \mathsf{true} \text{ where } u = f^{-1}(\alpha) \\ (\alpha, a, c, k), & \text{otherwise.} \end{cases}$$

Here, $\tilde{\mathcal{H}}_b$ are random functions and PC evaluates the boolean $k = \mathsf{KEM.D}^{\mathcal{H}'}(dk_u, c)$. As the composition of a random function and an injection is a random function, this does not change the behavior of the game. We also re-organized Encap slightly so that the table T does not have to be read by the oracle. This does not change the oracle's behavior so, $\Pr[\mathsf{H}_4^1] = \Pr[\mathsf{H}_0^2]$.

Next, we transition to H_1^2 where we have replaced calls to λ with its output. At the end of Decap, k is the decapsulation of c by definition so $\mathrm{PC}(u, k, c)$ will necessarily hold and $\lambda(f(u), a, k, c) = (f(u), a, c, \star)$. In H we have added code to check which of the cases of λ an input (α, a, c, k) corresponds to. In Encap, c is the encapsulation of k. So, $\mathrm{PC}(u, k, c)$ is true unless we've found a correctness failure in KEM. The probability of finding a correctness failures can clearly be bounded by $q_{\textsc{Encap}} \cdot \delta$. In Decap we replace the use of an entry in table T with the hash value we know will necessarily be stored it in if it is non-$\perp$ Therefore, $\Pr[\mathsf{H}_0^2] \leqslant \Pr[\mathsf{H}_1^2] + q_{\textsc{Encap}} \cdot \delta$. This term could more precisely be written as $q_{u,i} \cdot \delta$ where $q_{u,i}$ is the number of distinct values of (u, i) that $\mathcal{A}$ queries to its Encap oracle.

```
Hybrids $\mathsf{H}^{\iota}_{\kappa}$
$b \leftarrow\$ \{0,1\}$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\$ \mathsf{KEM.K}$
$\mathcal{H}_0, \mathcal{H}_1 \leftarrow\$ \mathsf{aUEM.IM}$
$\mathcal{H}' \leftarrow\$ \mathsf{KEM.IM}$
$\mathcal{K} \leftarrow \mathsf{KEM}.\mathcal{K}$; $\mathcal{C} \leftarrow \mathsf{KEM}.\mathcal{C}$
$D_1 \leftarrow \{0,1\}^{l_1+l_2} \times \mathcal{C} \times \mathcal{K}_\star$
$\tilde{\mathcal{H}} \leftarrow\$ \mathsf{Fcs}(\{0,1\}, D_1, \mathsf{aUEM}.\mathcal{K})$
$\alpha_{(\cdot)} \leftarrow\$ \{0,1\}^{l_1}$
$f^{\pm} \leftarrow\$ \mathsf{Inj}^{\pm}(\mathcal{U}, \{0,1\}^{l_1})$
$g^{\pm} \leftarrow\$ \mathsf{Inj}^{\pm}(\mathcal{U}, \mathcal{I}, \{0,1\}^{l_2})$
$b' \leftarrow\$ \mathcal{A}^{\text{NEW},\text{ENCAP},\text{DECAP},\text{H},\mathcal{H}'}$
Return $(b' = b)$
```

```
ENCAP$(u,i)$ //$\mathsf{H}^1$
If $C[u,i] \neq \bot$:
    $(a,c) \leftarrow C[u,i]$
    Return $((a,c), T[u,a,c])$
$I[u] \leftarrow I[u] \cup \{i\}$
$a \leftarrow\$ \{0,1\}^{l_2}$ //$\mathsf{H}^1_{[0,2)}$
$a \leftarrow g_u(i)$ //$\mathsf{H}^1_{[2,\infty)}$
$(c,k) \leftarrow\$ \mathsf{KEM.E}^{\mathcal{H}'}(ek_u)$
$K^1 \leftarrow \mathcal{H}_1(\alpha_u, a, c, k)$ //$\mathsf{H}^1_{[0,1)}$
$K^1 \leftarrow \mathcal{H}_1(f(u), a, c, k)$ //$\mathsf{H}^1_{[1,\infty)}$
$K^0 \leftarrow\$ \mathcal{K}$ //$\mathsf{H}^1_{[0,3)}$
$K^0 \leftarrow \mathcal{H}_0(f(u), a, c, k)$ //$\mathsf{H}^1_{[3,\infty)}$
$T[u,a,c] \leftarrow K^b$
$C[u,i] \leftarrow (a,c)$
Return $((a,c), K^b)$
```

```
DECAP$(u,a,c)$ //$\mathsf{H}^1$
$i \leftarrow g_u^{-1}(a)$ //$\mathsf{H}^1_{[4,\infty)}$
If $i \in I[u]$ and $(a,c) = C[u,i]$://$\mathsf{H}^1_{[4,\infty)}$
If $T[u,a,c] \neq \bot$: //$\mathsf{H}^1_{[0,4)}$
    Return $T[u,a,c]$
$k \leftarrow \mathsf{KEM.D}^{\mathcal{H}'}(dk_u, c)$
If $k = \bot$ then return $\bot$
Return $\mathcal{H}_1(\alpha_u, a, c, k)$ //$\mathsf{H}^1_{[0,1)}$
Return $\mathcal{H}_1(f(u), a, c, k)$ //$\mathsf{H}^1_{[1,\infty)}$

NEW$(u)$ //$\mathsf{H}^1$
Return $(\alpha_u, ek_u)$ //$\mathsf{H}^1_{[0,1)}$
Return $(f(u), ek_u)$ //$\mathsf{H}^1_{[1,\infty)}$

H$(\alpha, a, c, k)$ //$\mathsf{H}^1$
Return $\mathcal{H}_1(\alpha, a, c, k)$
```

```
ENCAP$(u,i)$ //$\mathsf{H}^2$
If $C[u,i] \neq \bot$:
    $(a,c) \leftarrow C[u,i]$; $k \leftarrow K[u,i]$
Else
    $a \leftarrow g_u(i)$
    $(c,k) \leftarrow\$ \mathsf{KEM.E}^{\mathcal{H}'}(ek_u)$
$I[u] \leftarrow I[u] \cup \{i\}$
$K^b \leftarrow \tilde{\mathcal{H}}_b(\lambda(f(u), a, c, k))$ //$\mathsf{H}^2_{[0,1)}$
$K^b \leftarrow \tilde{\mathcal{H}}_b(f(u), a, c, \star)$ //$\mathsf{H}^2_{[1,\infty)}$
$T[u,a,c] \leftarrow K^b$
$C[u,i] \leftarrow (a,c)$
$K[u,i] \leftarrow k$
Return $((a,c), K^b)$
```

```
DECAP$(u,a,c)$ //$\mathsf{H}^2$
$i \leftarrow g_u^{-1}(a)$
If $i \in I[u]$ and $(a,c) = C[u,i]$:
    Return $T[u,a,c]$ //$\mathsf{H}^2_{[0,1)}$
    Return $\tilde{\mathcal{H}}_b(f(u), a, c, \star)$ //$\mathsf{H}^2_{[1,\infty)}$
$k \leftarrow \mathsf{KEM.D}^{\mathcal{H}'}(dk_u, c)$
If $k = \bot$ then return $\bot$
Return $\tilde{\mathcal{H}}_1(\lambda(f(u), a, c, k))$ //$\mathsf{H}^2_{[0,1)}$
Return $\tilde{\mathcal{H}}_1(f(u), a, c, \star)$ //$\mathsf{H}^2_{[1,\infty)}$

NEW$(u)$ //$\mathsf{H}^2$
Return $(f(u), ek_u)$
```

```
H$(\alpha, a, c, k)$ //$\mathsf{H}^2$
Return $\tilde{\mathcal{H}}_1(\lambda(\alpha, a, c, k))$ //$\mathsf{H}^2_{[0,1)}$
$u \leftarrow f^{-1}(\alpha)$ //$\mathsf{H}^2_{[1,\infty)}$
If $u \neq \bot$ and PC$(u,k,c)$: //$\mathsf{H}^2_{[1,\infty)}$
    Return $\tilde{\mathcal{H}}_1(\alpha, a, c, \star)$ //$\mathsf{H}^2_{[1,\infty)}$
Return $\tilde{\mathcal{H}}_1(\alpha, a, c, k)$ //$\mathsf{H}^2_{[1,\infty)}$

Injection $\lambda(\alpha, a, c, k)$ //Internal, $\mathsf{H}^2$
$u \leftarrow f^{-1}(\alpha)$
If $u \neq \bot$ and PC$(u,k,c)$:
    Return $(\alpha, a, c, \star)$
Return $(\alpha, a, c, k)$
```

```
ENCAP$(u,i)$ //$\mathsf{H}^3$
$a \leftarrow g_u(i)$
$c \leftarrow$ CHAL$(u,i)$
$K^b \leftarrow \tilde{\mathcal{H}}_b(f(u), a, c, \star)$
Return $((a,c), K^b)$

CHAL$(u,i)$ //Internal, $\mathsf{H}^3$
If $C[u,i] = \bot$:
    $(c,k) \leftarrow\$ \mathsf{KEM.E}^{\mathcal{H}'}(ek_u)$
    $C[u,i] \leftarrow c$
Return $C[u,i]$
```

```
DECAP$(u,a,c)$ //$\mathsf{H}^3$
$i \leftarrow g_u^{-1}(a)$
If $i \neq \bot$ and $c =$ CHAL$(u,i)$:
    Return $\tilde{\mathcal{H}}_b(f(u), a, c, \star)$
$k \leftarrow \mathsf{KEM.D}^{\mathcal{H}'}(dk_u, c)$
If $k = \bot$ then return $\bot$
Return $\tilde{\mathcal{H}}_1(f(u), a, c, \star)$

NEW$(u)$ //$\mathsf{H}^3$
Return $(f(u), ek_u)$
```

```
H$(\alpha, a, k, c)$ //$\mathsf{H}^3$
$u \leftarrow f^{-1}(\alpha)$
If $u \neq \bot$ and PC$(u,k,c)$:
    $i \leftarrow g_u^{-1}(a)$ //$\mathsf{H}^3_{[1,\infty)}$
    If $i \neq \bot$ and $c =$ CHAL$(u,i)$: //$\mathsf{H}^3_{[1,\infty)}$
        $\mathsf{bad} \leftarrow \mathsf{true}$; Return $\tilde{\mathcal{H}}_b(\alpha, a, c, \star)$ //$\mathsf{H}^3_{[1,\infty)}$
    Return $\tilde{\mathcal{H}}_1(\alpha, a, c, \star)$
Return $\tilde{\mathcal{H}}_1(\alpha, a, c, k)$
```

Fig. 16. Hybrids used in proof of Theorem 4 (TAM-tight security of $\mathsf{aU}^{\perp}$). Oracles labelled “internal” are not accessible to the adversary. Grey highlighting indicates changes from earlier games. The code PC(u,k,c) is shorthand for the boolean $(k = \mathsf{KEM.D}^{\mathcal{H}'}(dk_u, c))$.

```
Adversary $\mathcal{B}_{\mathsf{mu\text{-}ow\text{-}pcva}}^{\text{New},\text{Chal},\text{PC},\text{CV},\mathcal{H}'}$
$D_1 \leftarrow \{0,1\}^{l_1+l_2} \times \mathsf{KEM}.\mathcal{C} \times \mathsf{KEM}.\mathcal{K}_\star$
$\tilde{\mathcal{H}} \leftarrow_\$ \mathsf{Fcs}(D_1, \mathsf{aUEM}.\mathcal{K})$
$f^\pm \leftarrow \mathsf{Inj}^\pm(\mathcal{U}, \{0,1\}^{l_1})$
$g^\pm \leftarrow \mathsf{Inj}^\pm(\mathcal{U}, \mathcal{I}, \{0,1\}^{l_2})$
$b' \leftarrow \mathcal{A}^{\text{SimNew},\text{SimEncap},\text{SimDecap},\text{SimH},\mathcal{H}'}$
Return $\perp$

SimNew$(u)$
$ek_u \leftarrow$ New$(u)$
Return $(f(u), ek_u)$

SimEncap$(u, i)$
$a \leftarrow g_u(i)$
$c \leftarrow$ Chal$(u, i)$
$K \leftarrow \tilde{\mathcal{H}}(f(u), a, c, \star)$
Return $((a, c), K)$

SimDecap$(u, a, c)$
$i \leftarrow g_u^{-1}(a)$
If $i \neq \perp$ and $c =$ Chal$(u, i)$:
    Return $\tilde{\mathcal{H}}(f(u), a, c, \star)$
If CV$(u, c) = \mathsf{false}$ then return $\perp$
Return $\tilde{\mathcal{H}}(f(u), a, c, \star)$

SimH$(\alpha, a, c, k)$
$u \leftarrow f^{-1}(\alpha)$
If $u \neq \perp$ and PC$(u, k, c)$:
    $i \leftarrow g_u^{-1}(a)$
    If $i \neq \perp$ and $c =$ Chal$(u, i)$:
        $\mathsf{OUTPUT}(u, i, k)$
    Return $\tilde{\mathcal{H}}(\alpha, a, c, \star)$
Return $\tilde{\mathcal{H}}(\alpha, a, c, k)$
```

Fig. 17. Adversary used for Theorem 4 (TAM-tight security of $\mathsf{aU}^{\perp}$).

Transitions H_0^3 to H_1^3. We have highlighted the most interesting differences between H_1^2 and H_0^3 in grey. In H_0^3, we have abstracted out the part of code that computes the challenge ciphertext into the subroutine Chal. Tables T and K are unused in H_1^2 so have been removed. Table C is modified to only store c; the auxiliary string a is always recomputed using g. The Encap oracles in H_1^2 and H_0^3 are equivalent.

In the Decap oracle, instead of using the table C to check whether the input ciphertext (a, c) was previously returned by Encap, we use the subroutine Chal to perform the same check. The calculation of i from a ensures we do not have to check that part of the ciphertext. Note that the comparison now looks at if $i \neq \perp$ rather than $i \in I[u]$ (indeed, the table I has been removed). These two checks can be distinguished if an adversary queries Decap with a "challenge" ciphertext that it hasn't yet seen from Encap. We can information theoretically bound the probability that the adversary does so. We do so by analyzing this probability in H_1^2 where the adversary's view only depends on "challenge" values a in the set $g_u(I[u])$. So, to distinguish, the adversary must guess some $a = g_u(i)$ such that $i \in \mathcal{I} \setminus I[u]$ along with the corresponding c such that $(c, k) \leftarrow_\$ \mathsf{KEME}^{\mathcal{H}'}(ek_u)$ for some k *would* be calculated by Encap(u, i).

We can bound the probability that some a the adversary choses is contained in the set $g_u(\mathcal{I} \setminus I[u])$ by

$$\frac{|g(\mathcal{I}) \setminus g(I[u])|}{|\{0,1\}^{l_2} \setminus g_u(I[u])|} = \frac{|\mathcal{I}| - |I[u]|}{2^{l_2} - |I[u]|} \leqslant \frac{|\mathcal{I}|}{2^{l_2}}.$$

Conditioned on having guessed a valid a, the adversary must also guess the corresponding c. Since the output of $\mathsf{KEM.E}$ is independent of a, this probability can be bounded by the min-entropy of KEM. Hence, the probability that a particular Decap query can distinguish these two games is $|\mathcal{I}|/(2^{l_2} \cdot 2^{\mathsf{H}_\infty(\mathsf{KEM})})$. Applying a union bound over all Decap queries, we get $\Pr[\mathsf{H}_1^2] \leqslant \Pr[\mathsf{H}_0^3] + q_{\text{Decap}} \cdot |\mathcal{I}|/2^{l_2 + \mathsf{H}_\infty(\mathsf{KEM})}$.

In H_1^3, we have added code to set the flag bad and use $\tilde{\mathcal{H}}_b$ rather than $\tilde{\mathcal{H}}_1$ if the H oracle is queried on a challenge ciphertext (and its decapsulation). Games H_0^3 and H_1^3 only differ under the event that the bad flag gets set and $b = 0$. By the fundamental lemma of game playing, $\Pr[\mathsf{H}_0^3] \leqslant \Pr[\mathsf{H}_1^3] + \Pr[b = 0 \wedge \mathsf{H}_1^3 \text{ sets } \mathsf{bad}]$.

In H_1^3, the only use of b is for determining whether $\tilde{\mathcal{H}}_1$ or $\tilde{\mathcal{H}}_0$ is used for challenge ciphertexts in Encap, Decap, and H. The hash functions both receive inputs of the same form $(f(u), a, c, \star)$ and are both random functions so their outputs are indistinguishable to the adversary. The three oracles are consistent in which of the two hash functions they use for such values. Therefore, the adversary's view in H_1^3 is independent of the bit b. Hence, $\Pr[\mathsf{H}_1^3] \leqslant 1/2$.

Probability of Bad. To bound $\Pr[b = 0 \wedge \mathsf{H}_1^3 \text{ sets } \mathsf{bad}]$, we let $\mathcal{B}$ be the adversary $\mathcal{B}_{\mathsf{mu\text{-}ow\text{-}pcva}}$ given in Fig. 17 against the OW-PCVA security of KEM. Note that bad is set in H_1^3 when PC(u, k, c) is true. Adversary $\mathcal{B}$ can now use this to win its OW-PCVA game. We claim that $\mathcal{B}$ perfectly simulates H_1^3 with $b = 0$ for $\mathcal{A}$. It only uses a single hash function $\tilde{\mathcal{H}}$ which is undetectable by the above arguments about $\mathcal{A}$'s view being independent of b in H_1^3. It uses its own challenge oracle Chal to compute ciphertexts while simulating oracles, its PC oracle to simulate H, and its CV oracle to check the validity of the input ciphertext to simulate Decap.

```
$\underline{\text{Hybrids } \mathsf{H}_\kappa \text{ for } 0 \leqslant \kappa \leqslant 4}$
$\mathcal{H} \times \mathcal{H}' \leftarrow\$ \mathsf{TKE.IM}$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\$ \mathsf{TKE.K}$
$g^{\pm} \leftarrow\$ \mathsf{Inj}^{\pm}(\mathcal{U} \times \mathcal{I}, \mathsf{PKE}.\mathcal{M})$
$\tilde{\mathcal{H}} \leftarrow\$ \mathsf{Fcs}(\mathcal{U} \times \mathcal{I}, \mathsf{PKE}.\mathcal{R})$ $//\mathsf{H}_{[2,4)}$
$(m, u, i) \leftarrow\$ \mathcal{A}^{\textsc{New},\textsc{Chal},\mathrm{PC},\mathrm{H},\mathcal{H}'}$
Return $\mathrm{PC}(u, m, \textsc{Chal}(u, i))$ $//\mathsf{H}_{[0,1)}$
Return $m = g(u, i)$ $//\mathsf{H}_{[1,\infty)}$

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

$\underline{\mathrm{H}(m)}$
If $g^{-1}(m) \neq \bot$: $//\mathsf{H}_{[2,\infty)}$
    Return $\tilde{\mathcal{H}}(g^{-1}(m))$ $//\mathsf{H}_{[2,3)}$
    $\mathsf{OUTPUT}(\mathsf{true})$ $//\mathsf{H}_{[3,\infty)}$
Return $\mathcal{H}(m)$
```

```
$\underline{\textsc{Chal}(u, i)}$
If $C[u, i] \neq \bot$ then return $C[u, i]$
$m \leftarrow g(u, i)$ $//\mathsf{H}_{[0,4)}$
$c \leftarrow \mathsf{PKE.E}^{\mathcal{H}'}(ek_u, m; \mathcal{H}(m))$ $//\mathsf{H}_{[0,2)}$
$c \leftarrow \mathsf{PKE.E}^{\mathcal{H}'}(ek_u, m; \tilde{\mathcal{H}}(u, i))$ $//\mathsf{H}_{[2,4)}$
$c \leftarrow\$ \mathsf{PKE.E}^{\mathcal{H}'}(ek_u, 0^{|m|})$ $//\mathsf{H}_{[4,\infty)}$
$C[u, i] \leftarrow c$
Return $c$

$\underline{\mathrm{PC}(u, m, c)}$
$m' \leftarrow \mathsf{TKE.D}^{\mathcal{H} \times \mathcal{H}'}((ek_u, dk_u), c)$ $//\mathsf{H}_{[0,1)}$
Return $(m = m')$ $//\mathsf{H}_{[0,1)}$
Return $(c = \mathsf{TKE.E}^{\mathrm{H} \times \mathcal{H}'}(ek_u, m))$ $//\mathsf{H}_{[1,\infty)}$
```

```
$\underline{\text{Adversary } \mathcal{B}^{\textsc{New},\textsc{Enc},\mathcal{H}'}}$
$\mathcal{H} \leftarrow\$ \mathsf{Fcs}(\mathsf{PKE}.\mathcal{M}, \mathsf{PKE}.\mathcal{R})$
$g^{\pm} \leftarrow\$ \mathsf{Inj}^{\pm}(\mathcal{U} \times \mathcal{I}, \mathsf{PKE}.\mathcal{M})$
$(m, u, i) \leftarrow\$ \mathcal{A}^{\textsc{New},\textsc{SimChal},\textsc{SimPC},\textsc{SimH},\mathcal{H}'}$
If $(m = g(u, i))$ then return 1
Return 0
```

```
$\underline{\textsc{SimH}(m)}$
If $g^{-1}(m) \neq \bot$:
    $\mathsf{OUTPUT}(1)$
Return $\mathcal{H}(m)$
```

```
$\underline{\textsc{SimChal}(u, i)}$
$c \leftarrow \textsc{Enc}(u, i, g(u, i))$
Return $c$

$\underline{\textsc{SimPC}(u, m, c)}$
Return $(c = \mathsf{TKE.E}^{\textsc{SimH} \times \mathcal{H}'}(\textsc{New}(u), m))$
```

Fig. 18. Hybrids H_κ and adversary $\mathcal{B}$ used for proof of Theorem 5 (TAM-tight security of T).

Whenever bad is set, $\mathcal{B}$ halts and wins its OW-PCVA game. Hence, $\Pr[\mathsf{H}_1^3 \text{ sets } \mathsf{bad} | b = 0] \leqslant \mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{mu\text{-}ow\text{-}pcva}}(\mathcal{B})$. Clearly $\Pr[b = 0] = 1/2$. This completes the proof. □

E Proof of Theorem 5 (T Transform)

For the correctness claim, consider an algorithm $\mathcal{D}$ that makes q oracle queries to its oracle $\mathcal{H} \times \mathcal{H}' \in \mathsf{TKE.IM}$. Speaking slightly informally, we can assume it "wins" and halts if makes an oracle query to $\mathcal{H}$ on an m such that correctness fails for TKE on m. Then we can apply a "union" bound over all of $\mathcal{D}$'s queries and its final output to obtain the claimed correctness bound of $(q + 1)\delta(q)$.

We prove Theorem 5 through a sequence of hybrids H_0 through H_4 presented in Fig. 18, where we establish the following claims that upper bound the advantage of adversary $\mathcal{A}$ as claimed in the theorem.

1. $\mathsf{Adv}_{\mathsf{TKE}}^{\mathsf{mu\text{-}ow\text{-}pca}}(\mathcal{A}) \leqslant \Pr[\mathsf{H}_0] + \frac{(q_{\textsc{Chal}}+1)^2}{2 \cdot |\mathsf{PKE}.\mathcal{M}|}$
2. $\Pr[\mathsf{H}_0] \leqslant \Pr[\mathsf{H}_1] + |\mathcal{U}| \cdot \delta'(q^*)$
3. $\Pr[\mathsf{H}_1] = \Pr[\mathsf{H}_2]$
4. $\Pr[\mathsf{H}_2] \leqslant \Pr[\mathsf{H}_3]$
5. $\Pr[\mathsf{H}_3] \leqslant \Pr[\mathsf{H}_4] + \mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{mu\text{-}cpa}}(\mathcal{B})$
6. $\Pr[\mathsf{H}_4] \leqslant (q_{\mathrm{H}} + q_{\mathrm{PC}} + 1) \cdot \frac{|\mathcal{U}| \cdot |\mathcal{I}|}{|\mathsf{PKE}.\mathcal{M}|}$

Transition H_0. Game H_0 was created by plugging code of TKE into $\mathsf{G}_{\mathsf{TKE}}^{\mathsf{mu\text{-}ow\text{-}pca}}$ and then switching from sampling the message m uniformly at random to assigning it as the output of a random injection g. The inputs to g do not repeat so, $\mathsf{Adv}_{\mathsf{TKE}}^{\mathsf{mu\text{-}ow\text{-}pca}}(\mathcal{A}) \leqslant \Pr[\mathsf{H}_0] + (q_{\textsc{Chal}} + 1)^2/(2 \cdot |\mathsf{PKE}.\mathcal{M}|)$.

Transition H_0 to H_1. In game H_1, the PC oracle is modified to check if c is the encryption of m, rather than if m is the decryption of c. Additionally the end of the game directly checks if $m = g(u, i)$, rather than if m is the decryption of the encryption of m. In both places, causing the behavior of H_0 and H_1 to differ gives a message for which the correctness of TKE breaks. Let PC_κ denote PC as defined in H_κ. Because TKE is rigid, it is not possible for PC_0 to return true when PC_1 would return false. If PC_1 returns true

when PC_0 would return false, then m encrypts to c, but c decrypts to $m' \neq m$, a correctness break. The end of game check in H_0 is whether $m = \mathsf{TKE.D}^{\mathcal{H}\times\mathcal{H}'}(dk_u, \mathsf{TKE.E}^{\mathcal{H}\times\mathcal{H}'}(ek_u, g(u,i)))$ and in H_1 is whether $m = g(u,i)$. The only way for these to differ is if $g(u,i) \neq \mathsf{TKE.D}^{\mathcal{H}\times\mathcal{H}'}(dk_u, \mathsf{TKE.E}^{\mathcal{H}\times\mathcal{H}'}(ek_u, g(u,i)))$, a correctness break.

Consider an algorithm $\mathcal{D}$ (a "δ'-correctness adversary") that simulates H_1 using its input keys as a random user u's keys and its oracles for the ideal model oracle. It checks if each $\mathrm{PC}(u,\cdot,\cdot)$ query gives a correctness break, outputting m if so. If $\mathcal{A}$ halts and outputs (m,u,i), then $\mathcal{D}$ outputs $g(u,i)$ as a potential correctness break. If $\mathcal{A}$ causes behavior that differentiates H_0 and H_1, there is a $1/|\mathcal{U}|$ chance this is for the chosen u, in which case $\mathcal{D}$ will. More formally, $\mathcal{D}$ can be written as follows to perfectly simulate H_0. To save on oracle queries, we omit having $\mathsf{TKE.D}$ recompute encryptions in the case that $m \neq m'$.

```
Adversary $\mathcal{D}^{\mathcal{H},\mathcal{H}'}(ek^*, dk^*)$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow_{\$} \mathsf{TKE.K}$
$u^* \leftarrow_{\$} \mathcal{U}$
$(ek_{u^*}, dk_{u^*}) \leftarrow (ek^*, dk^*)$
$g^{\pm} \leftarrow_{\$} \mathsf{Inj}^{\pm}(\mathcal{U}\times\mathcal{I}, \mathsf{PKE}.\mathcal{M})$
$(m,u,i) \leftarrow_{\$} \mathcal{A}^{\textsc{SimNew},\textsc{SimChal},\textsc{SimPC},\mathcal{H},\mathcal{H}'}$
$\mathsf{OUTPUT}(g(u,i))$

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

$\textsc{Chal}(u,i)$
If $C[u,i] \neq \bot$ then return $C[u,i]$
$C[u,i] \leftarrow \mathsf{PKE.E}^{\mathcal{H}'}(ek_u, g(u,i); \mathcal{H}(g(u,i)))$
Return $C[u,i]$

$\mathrm{PC}(u,m,c)$
$m' \leftarrow \mathsf{PKE.D}^{\mathcal{H}'}(dk_u, c)$
$c' \leftarrow \mathsf{PKE.E}^{\mathcal{H}'}(ek_u, m; \mathcal{H}(m))$
If $c' = c$ and $m' = m$ then return $\mathsf{true}$
If $u = u^*$ and $c' = c$ and $m' \neq m$ then $\mathsf{OUTPUT}(m)$
Return $\mathsf{false}$
```

The probability that $\mathcal{D}$ succeeds in finding a correctness break must be bounded by $\delta'(q^*)$ where $q^* = q_{\mathrm{H}} + q_{\textsc{Chal}}(q_{\mathsf{PKE}}+1) + q_{\mathrm{PC}}(2q_{\mathsf{PKE}}+1)$ is the number of oracle queries to $\mathcal{H}$ or $\mathcal{H}'$ made by $\mathcal{D}$. This gives $\Pr[\mathsf{H}_0] \leqslant \Pr[\mathsf{H}_1] + |\mathcal{U}|\cdot\delta'(q^*)$. This term could more precisely be written as $q_u \cdot \delta'(q^*)$ where q_u is the number of distinct values of u that $\mathcal{A}$ queries to its oracles or outputs at the end of execution. For this $\mathcal{D}$ would be modified to select $j \leftarrow_{\$} \{1, 2, \ldots, q_u\}$ and use its input keys for the j-th user that $\mathcal{A}$ accesses.

Transition H_1 to H_2. In the next transition we redefine $\mathcal{H}$ on inputs m in the image of g to be $\tilde{\mathcal{H}}(g^{-1}(m))$. This redefinition is done for both queries to H and the direct use of $\mathcal{H}$ in $\textsc{Chal}$. As g^{-1} is an injection and $\tilde{\mathcal{H}}$ is random, this does not change the behavior of the game, giving $\Pr[\mathsf{H}_1] = \Pr[\mathsf{H}_2]$.

Transition H_2 to H_3. In H_3, we halt the game early and return true if H is ever queried (directly by the adversary or through PC) with an m in the image of g. The game halting early and outputting true can only increase the probability that true is returned so $\Pr[\mathsf{H}_2] \leqslant \Pr[\mathsf{H}_3]$.

Transition H_3 to H_4. Note that in H_3, the random function $\tilde{H}$ is only ever used inside of $\textsc{Chal}$ to pick the coins for encryption. Thus for each (u,i), the message $m = g(u,i)$ is simply being encrypted by PKE with fresh coins. We can apply the CPA security of PKE to replace this with an encryption of $0^{|m|}$. This is captured by the adversary $\mathcal{B}$ shown in the same figure. It simulates the view of $\mathcal{A}$, using its encryption oracle for the encryptions in the challenge oracle. If $\mathcal{A}$ would cause its game to output true, then $\mathcal{B}$ outputs 1. Otherwise it outputs 0. When $\mathcal{B}$'s oracle is real it correctly simulates H_3 and when the oracle returns encryptions of zeros, it correctly simulates H_4. Thus, $\Pr[\mathsf{H}_3] = \Pr[\mathsf{H}_4] + \mathsf{Adv}^{\mathsf{mu\text{-}cpa}}_{\mathsf{PKE}}(\mathcal{B})$.

Game H_4. Finally we can information theoretically bound the probability that $\mathcal{A}$ succeeds in H_4. To win $\mathcal{A}$ must produce a m that is in the image of g as part of a query to H or PC, or as its final output. So $\mathcal{A}$ gets $q_{\mathrm{H}} + q_{\mathrm{PC}} + 1$ "guesses" and its view is otherwise independent of g. A simple union bound give that $\Pr[\mathsf{H}_4] \leqslant (q_{\mathrm{H}} + q_{\mathrm{PC}} + 1)\cdot|\mathcal{U}|\cdot|\mathcal{I}|/|\mathsf{PKE}.\mathcal{M}|$. This term could more precisely be written with $q_{\mathcal{H}}$ in place of q_{H}, where $q_{\mathcal{H}}$ is the number of oracle queries that $\mathcal{A}$ makes to $\mathcal{H}$. (Note that q_{H} additionally includes queries to $\mathcal{H}'$.) For query restricting adversaries that use a global counter for i it would have sufficed for to replace $g(u,i)$ with $g(i)$, in which case $|\mathcal{U}|\cdot|\mathcal{I}|$ could be replaced with $q_{\textsc{Chal}}$. □

F Proof of Theorem 6 (Augmented V Transform)

We prove Theorem 6 through a sequence of hybrids H^1_0 through H^1_2 and H^2_0 through H^2_3 presented in Fig. 19. We establish the following claims that upper bound the advantage of adversary $\mathcal{A}$ as claimed in the theorem.

1. $\mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}pcva}}_{\mathsf{VKE}}(\mathcal{A}) = \Pr[\mathsf{H}^1_0]$
2. $\Pr[\mathsf{H}^1_0] \leqslant \Pr[\mathsf{H}^1_1] + |\mathcal{U}|^2/2^{l+1}$
3. $\Pr[\mathsf{H}^1_1] \leqslant \Pr[\mathsf{H}^1_2] + |\mathcal{U}| \cdot \delta'(q_1)$
4. $\Pr[\mathsf{H}^1_2] = \Pr[\mathsf{H}^2_0]$
5. $\Pr[\mathsf{H}^2_0] = \Pr[\mathsf{H}^2_1]$
6. $\Pr[\mathsf{H}^2_1] \leqslant \Pr[\mathsf{H}^2_2] + \Pr[\mathsf{H}^2_2 \text{ sets } \mathsf{bad}]$
7. $\Pr[\mathsf{H}^2_2] \leqslant \mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}pca}}_{\mathsf{TKE}}(\mathcal{B})$
8. $\Pr[\mathsf{H}^2_2 \text{ sets } \mathsf{bad}] \leqslant |\mathcal{U}| \cdot \delta'(q_2) + q_{\mathrm{CV}}/2^{\gamma}$

Here $q_1 = q_{\mathsf{TKE}}(q_{\textsc{Chal}} + q_{\mathrm{H}} + q_{\mathrm{PC}} + q_{\mathrm{CV}} + 1) + q_{\mathrm{H}}$ and $q_2 = q_{\mathsf{TKE}}(2q_{\textsc{Chal}} + 2q_{\mathrm{H}} + q_{\mathrm{PC}}) + q_{\mathrm{H}}$. Note that both are less than $q^* = q_{\mathsf{TKE}}(2q_{\textsc{Chal}} + 2q_{\mathrm{H}} + q_{\mathrm{PC}} + q_{\mathrm{CV}} + 1) + q_{\mathrm{H}}$.

Transitions H^1_0 to H^1_2. We start with the hybrid H^1_0 which is obtained by plugging in the code for VKE in the game $\mathsf{G}^{\mathsf{mu\text{-}ow\text{-}pcva}}_{\mathsf{VKE}}$ from Fig. 3. As TKE is tidy, we've omitted the re-encryption done by $\mathsf{VKE.D}$ in PC and CV. Hence, $\mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}pcva}}_{\mathsf{VKE}}(\mathcal{A}) = \Pr[\mathsf{H}^1_0]$.

Next we transition to H^1_1. In H^1_1, instead of sampling the random string α uniformly at random, we assign it as the output of the random injection f. Using the switching lemma, we get $\Pr[\mathsf{H}^1_0] \leqslant \Pr[\mathsf{H}^1_1] + 0.5|\mathcal{U}|^2/2^l$. This term could more precisely be written as $0.5q_u^2/2^l$ where q_u is the number of distinct values of u that $\mathcal{A}$ queries to its oracles.

In H^1_2, the random function $\mathcal{H}$ is replaced with the random function $\tilde{\mathcal{H}}$ according to the definition $\mathcal{H}(\alpha, m) = \tilde{\mathcal{H}}(u, \mathsf{TKE.E}^{\mathcal{H}'}(ek_u, m))$ where $u = f^{-1}(\alpha)$. When $f^{-1}(\alpha) = \bot$, we still used $\mathcal{H}$. Inside PC and CV, the messages in question were obtained by decrypting a ciphertext c_1. By rigidity we know the message will re-encrypt to c_1 so we use c_1 directly, rather than re-encrypting. This change can only be detected is by querying $\mathcal{H}$ on $(f(u), m)$ and $(f(u), m')$ where $m \neq m'$, but $\mathsf{TKE.E}^{\mathcal{H}'}(ek_u, m) = \mathsf{TKE.E}^{\mathcal{H}'}(ek_u, m')$. Then at least one of m and m' must result in a correctness failure for TKE.

In Fig. 20, we design a correctness adversary $\mathcal{D}_1$ to bound the probability of this. We've used highlighting to indicate the interesting differences between it and H^1_2. In particular, it sets the keys of a random user u^* to be the keys it was given as input. Any time it would make a query $\tilde{\mathcal{H}}(u^*, c_1)$ where c_1 is known to be an encryption of m it queries its internal oracle $\textsc{Check}(m, c)$. This oracle uses a table T to store a mapping from ciphertexts that were queries to $\tilde{H}$ and the messages they were the encryptions of. If a collision in T is ever found, there must have been correctness error. The oracle checks whether this was for m or $T[c]$ and outputs the result. Note that $\mathcal{D}_1$ makes $q_1 = q_{\mathsf{TKE}}(q_{\textsc{Chal}} + q_{\mathrm{H}} + q_{\mathrm{PC}} + q_{\mathrm{CV}} + 1) + q_{\mathrm{H}}$ queries to $\mathcal{H}'$. Hence, it establishes that $\Pr[\mathsf{H}^1_1] \leqslant \Pr[\mathsf{H}^1_2] + |\mathcal{U}| \cdot \delta'(q_1)$.

Transition H^1_2 to H^2_0. The only change from game H^1_2 to H^2_0 is a slight reorganization of the code of PC which can easily be seen not to change the behavior of that oracle. Hence, $\Pr[\mathsf{H}^1_2] = \Pr[\mathsf{H}^2_0]$.

Transitions H^2_0 to H^2_2. The changes in this set of transitions are all to the CV oracle. In H^2_1, we rearrange the code of the CV oracle and set the bad flag whenever $c_2 = c_2'$ but c_1 is an invalid ciphertext (i.e., attempting to decrypt it fails). Despite this reorganization, the oracle in H^2_1 returns the same values as the CV oracle in H^2_0. Therefore, $\Pr[\mathsf{H}^2_0] = \Pr[\mathsf{H}^2_1]$.

The CV oracle in H^2_2 is the same as that in H^2_1 except when the flag bad is set at which point it returns true, rather than false. By the fundamental lemma of game playing proofs, $\Pr[\mathsf{H}^2_1] \leqslant \Pr[\mathsf{H}^2_2] + \Pr[\mathsf{H}^2_2 \text{ sets } \mathsf{bad}]$.

To bound $\Pr[\mathsf{H}^2_2]$, we construct the adversary $\mathcal{B}$ against the $\mathsf{mu\text{-}ow\text{-}pca}$ security of the underlying scheme TKE in Fig. 19. Adversary $\mathcal{B}$ can readily be seen to perfectly simulate game H^2_2 for adversary $\mathcal{A}$. Therefore, $\Pr[\mathsf{H}^2_2] \leqslant \mathsf{Adv}^{\mathsf{mu\text{-}ow\text{-}pca}}_{\mathsf{TKE}}(\mathcal{B})$.

Now we analyze the event that the flag bad gets set in H^2_2. We note from Fig. 19 that bad is set when $c_2 = c_2'$ but $m = \bot$ where $m = \mathsf{TKE.D}^{\mathcal{H}'}(dk_u, c_1)$. We can define an additional hybrid H^2_3 to be equivalent to H^2_2 except it halts execution immediately if $\textsc{Chal}$, H, or PC ever query $\tilde{\mathcal{H}}(u, c_1)$ for some c_1 such that $\mathsf{TKE.D}^{\mathcal{H}'}(dk_u, c_1) = \bot$. Note that this is not possible in PC because it only queries $\tilde{\mathcal{H}}$ when c_1 decrypts to the non-$\bot$ message it was given as input. In $\textsc{Chal}$ or H, the ciphertext c_1 was obtained by encrypting a message m so this can only happen if there was a correctness failure for TKE. In Fig. 20, we design a correctness adversary $\mathcal{D}_2$ to bound the probability of this. We've used highlighting to indicate the interesting differences between it and H^2_2. In particular, it sets the keys of a random user u^* to be the keys it was given as input.

```
Hybrids $\mathsf{H}^{\iota}_{\kappa}$
$\mathcal{H} \times \mathcal{H}' \leftarrow_{\$} \mathsf{VKE.IM}$
$\tilde{\mathcal{H}} \leftarrow_{\$} \mathsf{Fcs}(\mathcal{U} \times \mathsf{TKE}.\mathcal{C}, \{0,1\}^{\gamma})$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow_{\$} \mathsf{TKE}.\mathcal{K}$
$\alpha_{(\cdot)} \leftarrow_{\$} \{0,1\}^{l_1}$
$f^{\pm} \leftarrow_{\$} \mathsf{Inj}^{\pm}(\mathcal{U}, \{0,1\}^{l})$
$(m, u, i) \leftarrow_{\$} \mathcal{A}^{\textsc{New},\textsc{Chal},\mathrm{PC},\mathrm{CV},\mathrm{H},\mathcal{H}'}$
Return $\mathrm{PC}(u, m, \textsc{Chal}(u, i))$

$\textsc{New}(u)$ $//\mathsf{H}^1, \mathsf{H}^2$
Return $(\alpha_u, ek_u)$ $//\mathsf{H}^1_{[0,1)}$
Return $(f(u), ek_u)$ $//\mathsf{H}^1_{[1,\infty)}, \mathsf{H}^2_{[0,\infty)}$
```

```
$\textsc{Chal}(u, i)$ $//\mathsf{H}^1$
If $C[u, i] \neq \bot$:
   Return $C[u, i]$
$m \leftarrow_{\$} \mathsf{VKE}.\mathcal{M}$
$c_1 \leftarrow \mathsf{TKE.E}^{\mathcal{H}'}(ek_u, m)$
$c_2 \leftarrow \mathcal{H}(\alpha_u, m)$ $//\mathsf{H}^1_{[0,1)}$
$c_2 \leftarrow \mathcal{H}(f(u), m)$ $//\mathsf{H}^1_{[1,2)}$
$c_2 \leftarrow \tilde{\mathcal{H}}(u, c_1)$ $//\mathsf{H}^1_{[2,\infty)}$
$c \leftarrow (c_1, c_2)$
$C[u, i] \leftarrow c$
Return $c$

$\mathrm{H}(\alpha, m)$ $//\mathsf{H}^1$
If $f^{-1}(\alpha) \neq \bot$: $//\mathsf{H}^1_{[2,\infty)}$
   $u \leftarrow f^{-1}(\alpha)$ $//\mathsf{H}^1_{[2,\infty)}$
   $c_1 \leftarrow \mathsf{TKE.E}^{\mathcal{H}'}(ek_u, m)$ $//\mathsf{H}^1_{[2,\infty)}$
   Return $\tilde{\mathcal{H}}(u, c_1)$ $//\mathsf{H}^1_{[2,\infty)}$
Return $\mathcal{H}(\alpha, m)$
```

```
$\mathrm{PC}(u, m, c)$ $//\mathsf{H}^1$
$(c_1, c_2) \leftarrow c$
$m' \leftarrow \mathsf{TKE.D}^{\mathcal{H}'}(dk_u, c_1)$
If $m' = \bot$ then return $\mathsf{false}$
$c_2' \leftarrow \mathcal{H}(\alpha_u, m')$ $//\mathsf{H}^1_{[0,1)}$
$c_2' \leftarrow \mathcal{H}(f(u), m')$ $//\mathsf{H}^1_{[1,2)}$
$c_2' \leftarrow \tilde{\mathcal{H}}(u, c_1)$ $//\mathsf{H}^1_{[2,\infty)}$
Return $(m = m')$ and $(c_2 = c_2')$

$\mathrm{CV}(u, c)$ $//\mathsf{H}^1$
$(c_1, c_2) \leftarrow c$
$m \leftarrow \mathsf{TKE.D}^{\mathcal{H}'}(dk_u, c_1)$
If $m = \bot$ then return $\mathsf{false}$
$c_2' \leftarrow \mathcal{H}(\alpha_u, m)$ $//\mathsf{H}^1_{[0,1)}$
$c_2' \leftarrow \mathcal{H}(f(u), m)$ $//\mathsf{H}^1_{[1,2)}$
$c_2' \leftarrow \tilde{\mathcal{H}}(u, c_1)$ $//\mathsf{H}^1_{[2,\infty)}$
Return $(m \neq \bot)$ and $(c_2 = c_2')$
```

```
$\textsc{Chal}(u, i)$ $//\mathsf{H}^2$
If $C[u, i] \neq \bot$:
   Return $C[u, i]$
$m \leftarrow_{\$} \mathsf{VKE}.\mathcal{M}$
$c_1 \leftarrow \mathsf{TKE.E}^{\mathcal{H}'}(ek_u, m)$
$c_2 \leftarrow \tilde{\mathcal{H}}(u, c_1)$
$c \leftarrow (c_1, c_2)$
$C[u, i] \leftarrow c$
Return $c$
```

```
$\mathrm{H}(\alpha, m)$ $//\mathsf{H}^2$
If $f^{-1}(\alpha) \neq \bot$:
   $u \leftarrow f^{-1}(\alpha)$
   $c_1 \leftarrow \mathsf{TKE.E}^{\mathcal{H}'}(ek_u, m)$
   Return $\tilde{\mathcal{H}}(u, c_1)$
Return $\mathcal{H}(\alpha, m)$

$\mathrm{PC}(u, m, c)$ $//\mathsf{H}^2$
$(c_1, c_2) \leftarrow c$
If $m = \mathsf{TKE.D}^{\mathcal{H}'}(dk_u, c_1)$:
   $c_2' \leftarrow \tilde{\mathcal{H}}(u, c_1)$
   Return $(c_2 = c_2')$
Return $\mathsf{false}$
```

```
$\mathrm{CV}(u, c)$ $//\mathsf{H}^2$
$(c_1, c_2) \leftarrow c$
$m \leftarrow \mathsf{TKE.D}^{\mathcal{H}'}(dk_u, c_1)$ $//\mathsf{H}^2_{[0,2)}$
$c_2' \leftarrow \tilde{\mathcal{H}}(u, c_1)$
If $c_2 = c_2'$: $//\mathsf{H}^2_{[1,\infty)}$
   If $m = \bot$: $//\mathsf{H}^2_{[1,2)}$
      $\mathsf{bad} \leftarrow \mathsf{true}$ $//\mathsf{H}^2_{[1,2)}$
      Return $\mathsf{false}$ $//\mathsf{H}^2_{[1,2)}$
   Return $\mathsf{true}$ $//\mathsf{H}^2_{[1,\infty)}$
Return $(m \neq \bot)$ and $(c_2 = c_2')$ $//\mathsf{H}^2_{[0,1)}$
Return $\mathsf{false}$ $//\mathsf{H}^2_{[1,\infty)}$
```

```
Adversary $\mathcal{B}^{\textsc{New},\textsc{Chal},\mathrm{PC},\mathcal{H}'}$
$\mathcal{H} \leftarrow_{\$} \mathsf{Fcs}(\{0,1\}^{l} \times \mathsf{TKE}.\mathcal{M}, \{0,1\}^{\gamma})$
$\tilde{\mathcal{H}} \leftarrow_{\$} \mathsf{Fcs}(\mathcal{U} \times \mathsf{TKE}.\mathcal{C}, \{0,1\}^{\gamma})$
$f^{\pm} \leftarrow_{\$} \mathsf{Inj}^{\pm}(\mathcal{U}, \{0,1\}^{l})$
$(m, u, i) \leftarrow_{\$} \mathcal{A}^{\mathrm{O}}$
Return $(m, u, i)$

$\textsc{SimNew}(u)$
Return $(f(u), \textsc{New}(u))$
```

```
$\textsc{SimChal}(u, i)$
$c_1 \leftarrow \textsc{Chal}(u, i)$
$c_2 \leftarrow \tilde{\mathcal{H}}(u, c_1)$
$c \leftarrow (c_1, c_2)$
Return $c$

$\textsc{SimH}(\alpha, m)$
If $f^{-1}(\alpha) \neq \bot$:
   $u \leftarrow f^{-1}(\alpha)$; $ek \leftarrow \textsc{New}(u)$
   Return $\tilde{\mathcal{H}}(u, \mathsf{TKE.E}^{\mathcal{H}'}(ek, m))$
Return $\mathcal{H}(\alpha, m)$
```

```
$\textsc{SimPC}(u, m, c)$
$(c_1, c_2) \leftarrow c$
If $\mathrm{PC}(u, m, c_1)$:
   $c_2' \leftarrow \tilde{\mathcal{H}}(u, c_1)$
   Return $(c_2 = c_2')$
Return $\mathsf{false}$

$\textsc{SimCV}(u, c)$
$(c_1, c_2) \leftarrow c$
$c_2' \leftarrow \tilde{\mathcal{H}}(u, c_1)$
Return $(c_2 = c_2')$
```

Fig. 19. Hybrid games and the adversary $\mathcal{B}$ used in proof of Theorem 6 (TAM-tight security of augmented V transform). Grey highlighting indicates changes from earlier games.

```
Adversary $\mathcal{D}_1^{\mathcal{H}'}(ek^*, dk^*)$
$\mathcal{H} \leftarrow_\$ \mathsf{Fcs}(\{0,1\}^l \times \mathsf{TKE}.\mathcal{M}, \{0,1\}^\gamma)$
$\tilde{\mathcal{H}} \leftarrow_\$ \mathsf{Fcs}(\mathcal{U} \times \mathsf{TKE}.\mathcal{C}, \{0,1\}^\gamma)$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow_\$ \mathsf{TKE}.\mathcal{K}$
$u^* \leftarrow_\$ \mathcal{U}$
$(ek_{u^*}, dk_{u^*}) \leftarrow (ek^*, dk^*)$
$f^\pm \leftarrow_\$ \mathsf{Inj}^\pm(\mathcal{U}, \{0,1\}^l)$
$(m, u, i) \leftarrow_\$ \mathcal{A}^{\textsc{SimNew},\textsc{SimChal},\textsc{SimPC},\textsc{SimCV},\textsc{SimH},\mathcal{H}'}$
Return $\mathsf{OUTPUT}(\bot)$

$\textsc{SimNew}(u)$
Return $(f(u), ek_u)$

$\textsc{Check}(m, c)$ //Internal
If $T[c] \neq \bot$ and $T[c] \neq m$:
    $m' \leftarrow \mathsf{TKE.D}^{\mathcal{H}'}(dk^*, c)$
    If $m' \neq m$ then $\mathsf{OUTPUT}(m)$
    $\mathsf{OUTPUT}(T[c])$
$T[c] \leftarrow m$

$\textsc{SimChal}(u, i)$
If $C[u, i] \neq \bot$:
    Return $C[u, i]$
$m \leftarrow_\$ \mathsf{VKE}.\mathcal{M}$
$c_1 \leftarrow \mathsf{TKE.E}^{\mathcal{H}'}(ek_u, m)$
If $u = u^*$ then $\textsc{Check}(m, c_1)$
$c_2 \leftarrow \tilde{\mathcal{H}}(u, c_1)$
$c \leftarrow (c_1, c_2)$
$C[u, i] \leftarrow c$
Return $c$

$\textsc{SimH}(\alpha, m)$
If $f^{-1}(\alpha) \neq \bot$:
    $u \leftarrow f^{-1}(\alpha)$
    $c_1 \leftarrow \mathsf{TKE.E}^{\mathcal{H}'}(ek_u, m)$
    If $u = u^*$ then $\textsc{Check}(m, c_1)$
    Return $\tilde{\mathcal{H}}(u, c_1)$
Return $\mathcal{H}(\alpha, m)$

$\textsc{SimPC}(u, m, c)$
$(c_1, c_2) \leftarrow c$
$m' \leftarrow \mathsf{TKE.D}^{\mathcal{H}'}(dk_u, c_1)$
If $m' = \bot$ then return $\mathsf{false}$
If $u = u^*$ then $\textsc{Check}(m', c_1)$
$c_2' \leftarrow \tilde{\mathcal{H}}(u, c_1)$
Return $(m = m')$ and $(c_2 = c_2')$

$\textsc{SimCV}(u, c)$
$(c_1, c_2) \leftarrow c$
$m \leftarrow \mathsf{TKE.D}^{\mathcal{H}'}(dk_u, c_1)$
If $m = \bot$ then return $\mathsf{false}$
If $u = u^*$ then $\textsc{Check}(m, c_1)$
$c_2' \leftarrow \tilde{\mathcal{H}}(u, c_1)$
Return $(m \neq \bot)$ and $(c_2 = c_2')$
```

```
Adversary $\mathcal{D}_2^{\mathcal{H}'}(ek^*, dk^*)$
$\mathcal{H} \leftarrow_\$ \mathsf{Fcs}(\{0,1\}^l \times \mathsf{TKE}.\mathcal{M}, \{0,1\}^\gamma)$
$\tilde{\mathcal{H}} \leftarrow_\$ \mathsf{Fcs}(\mathcal{U} \times \mathsf{TKE}.\mathcal{C}, \{0,1\}^\gamma)$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow_\$ \mathsf{TKE}.\mathcal{K}$
$u^* \leftarrow_\$ \mathcal{U}$
$(ek_{u^*}, dk_{u^*}) \leftarrow (ek^*, dk^*)$
$f^\pm \leftarrow_\$ \mathsf{Inj}^\pm(\mathcal{U}, \{0,1\}^l)$
$(m, u, i) \leftarrow_\$ \mathcal{A}^{\textsc{SimNew},\textsc{SimChal},\textsc{SimPC},\textsc{SimCV},\textsc{SimH},\mathcal{H}'}$
Return $\mathsf{OUTPUT}(\bot)$

$\textsc{SimNew}(u)$
Return $(f(u), ek_u)$

$\textsc{Check}(m, c)$ //Internal
$m' \leftarrow \mathsf{TKE.D}^{\mathcal{H}'}(dk^*, c)$
If $m' = \bot$ then $\mathsf{OUTPUT}(m)$

$\textsc{SimChal}(u, i)$
If $C[u, i] \neq \bot$:
    Return $C[u, i]$
$m \leftarrow_\$ \mathsf{VKE}.\mathcal{M}$
$c_1 \leftarrow \mathsf{TKE.E}^{\mathcal{H}'}(ek_u, m)$
If $u = u^*$ then $\textsc{Check}(m, c_1)$
$c_2 \leftarrow \tilde{\mathcal{H}}(u, c_1)$
$c \leftarrow (c_1, c_2)$
$C[u, i] \leftarrow c$
Return $c$

$\textsc{SimCV}(u, c)$
$(c_1, c_2) \leftarrow c$
$c_2' \leftarrow \tilde{\mathcal{H}}(u, c_1)$
Return $(c_2 = c_2')$

$\textsc{SimH}(\alpha, m)$
If $f^{-1}(\alpha) \neq \bot$:
    $u \leftarrow f^{-1}(\alpha)$
    $c_1 \leftarrow \mathsf{TKE.E}^{\mathcal{H}'}(ek_u, m)$
    If $u = u^*$ then $\textsc{Check}(m, c_1)$
    Return $\tilde{\mathcal{H}}(u, c_1)$
Return $\mathcal{H}(\alpha, m)$

$\textsc{SimPC}(u, m, c)$
$(c_1, c_2) \leftarrow c$
If $m = \mathsf{TKE.D}^{\mathcal{H}'}(dk_u, c_1)$:
    $c_2' \leftarrow \tilde{\mathcal{H}}(u, c_1)$
    Return $(c_2 = c_2')$
Return $\mathsf{false}$
```

Fig. 20. Correctness adversaries used for proof of Theorem 6 (TAM-tight security of augmented V transform).

Any time it would make a query $\tilde{\mathcal{H}}(u^*, c_1)$ in SimChal or SimH it queries its internal oracle Check(m, c_1). This oracle outputs m if c_1 decrypts to $\bot$. Note that $\mathcal{D}_2$ makes $q_2 = q_{\mathsf{TKE}}(2q_{\text{Chal}} + 2q_{\text{H}} + q_{\text{PC}}) + q_{\text{H}}$ queries to $\mathcal{H}'$. Hence, $\Pr[\mathsf{H}_2^2 \text{ sets } \mathsf{bad}] \leqslant \Pr[\mathsf{H}_3^2 \text{ sets } \mathsf{bad}] + |\mathcal{U}| \cdot \delta'(q_2)$.

Now in H_3^2 the only way the adversary's view can depend on $\tilde{\mathcal{H}}(u, c_1)$ for such c_1 is by making queries $\text{CV}(u, (c_1, c_2))$ and learning if $\tilde{\mathcal{H}}(u, c_1) = c_2$. So standard analysis gives $\Pr[\mathsf{H}_3^2 \text{ sets } \mathsf{bad}] \leqslant q_{\text{CV}}/2^\gamma$. This completes the proof. □

G Proof of Theorem 7 (KEM/DEM Construction, Challenge Respecting)

We prove Theorem 7 via a sequence of hybrid games shown in Fig. 21, 22, 23, and 24. We establish the following bounds relating the different hybrids.

```
Hybrids $\mathsf{H}^1_\kappa$ for $0 \leqslant \kappa \leqslant 1$
$\mathcal{H} \leftarrow\!\!\$\ \mathsf{KD.IM}$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\!\!\$\ \mathsf{KEM.K}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{New},\textsc{Enc},\textsc{Dec},\mathcal{H}}$
Return $(b' = 1)$

$\textsc{Enc}(u, i, m)$
$(c^k, K) \leftarrow\!\!\$\ \mathsf{KEM.E}^{\mathcal{H}}(ek_u)$   $//\mathsf{H}^1_{[0,1)}$
$c^k \leftarrow\!\!\$\ \mathsf{KEM}.\mathcal{C}(ek_u)$   $//\mathsf{H}^1_{[1,\infty)}$
$K \leftarrow\!\!\$\ \mathsf{KEM}.\mathcal{K}$   $//\mathsf{H}^1_{[1,\infty)}$
$c^d \leftarrow\!\!\$\ \mathsf{SKE.E}^{\mathcal{H}'}(K, m)$
$T[u, c^k] \leftarrow K$
Return $(c^k, c^d)$
```

```
$\textsc{Dec}(u, c)$
$(c^k, c^d) \leftarrow c$
If $T[u, c^k] \neq \bot$
    $K \leftarrow T[u, c^k]$
    Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$
$K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$
If $K = \bot$ then return $\bot$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$

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

```
Adversary $\mathcal{B}_0^{\textsc{New},\textsc{Encap},\textsc{Decap},\mathcal{H}}$
$d' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{New},\textsc{SimEnc},\textsc{SimDec},\mathcal{H}}$
Return $d'$

$\textsc{SimEnc}(u, i, m)$
$(c^k, K) \leftarrow \textsc{Encap}(u, i)$
$c^d \leftarrow\!\!\$\ \mathsf{SKE.E}^{\mathcal{H}}(K, m)$
Return $(c^k, c^d)$

$\textsc{SimDec}(u, c)$
$(c^k, c^d) \leftarrow c$
$K \leftarrow \textsc{Decap}(u, c^k)$
If $K = \bot$ then return $\bot$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$
```

Fig. 21. Hybrids H^1_κ (**Left**) and adversary $\mathcal{B}_0$ (**Right**) used for proof of Theorem 7.

1. $\Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD},1}(\mathcal{A})] \leqslant \Pr[\mathsf{H}^1_0] + q_{\text{Enc}} \cdot \delta$
2. $\Pr[\mathsf{H}^1_0] = \Pr[\mathsf{H}^1_1] + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_0)$
3. $\Pr[\mathsf{H}^1_1] \leqslant \Pr[\mathsf{H}^2_0] + 0.5q^2_{\text{Enc}}/|\mathsf{KEM}.\mathcal{C}|$
4. $\Pr[\mathsf{H}^2_0] = \Pr[\mathsf{H}^2_1] = \Pr[\mathsf{H}^2_2]$
5. $\Pr[\mathsf{H}^2_2] \leqslant \Pr[\mathsf{H}^2_3] + q_{\text{Enc}}q_{\text{Dec}}/|\mathsf{KEM}.\mathcal{C}|$
6. $\Pr[\mathsf{H}^2_3] = \Pr[\mathsf{H}^2_4] + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE}}(\mathcal{B}_{\mathsf{SKE}})$
7. $\Pr[\mathsf{H}^2_4] \leqslant \Pr[\mathsf{H}^2_5] + q_{\text{Enc}}q_{\text{Dec}}/|\mathsf{KEM}.\mathcal{C}|$
8. $\Pr[\mathsf{H}^2_5] = \Pr[\mathsf{H}^3_0] = \Pr[\mathsf{H}^3_1]$
9. $\Pr[\mathsf{H}^3_1] = \Pr[\mathsf{H}^3_2] = \Pr[\mathsf{H}^3_3] = \Pr[\mathsf{H}^3_4]$
10. $\Pr[\mathsf{H}^3_4] \leqslant \Pr[\mathsf{H}^3_5] + 0.5q^2_{\text{Enc}}/|\mathsf{KEM}.\mathcal{C}|$
11. $\Pr[\mathsf{H}^3_5] \leqslant \Pr[\mathsf{H}^4_0] + 0.5q^2_{\text{Enc}}/|\mathsf{KEM}.\mathcal{C}| + 0.5q^2_{\text{Enc}}/2^{\mathsf{SKE.xl}}$
12. $\Pr[\mathsf{H}^4_0] \leqslant \Pr[\mathsf{H}^4_1] + q_{\text{Dec}}/2^{\mathsf{SKE.xl}}$
13. $\Pr[\mathsf{H}^4_1] = \Pr[\mathsf{H}^4_2] + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_1)$
14. $\Pr[\mathsf{H}^4_2] \leqslant \Pr[\mathsf{H}^4_3] + q_{\text{Enc}} \cdot \delta$
15. $\Pr[\mathsf{H}^4_3] \leqslant \Pr[\mathsf{H}^4_4] + q_{\text{Enc}} \cdot \varepsilon$
16. $\Pr[\mathsf{H}^4_4] \leqslant \Pr[\mathsf{H}^4_5] + q_{\text{Dec}}/2^{\mathsf{SKE.xl}}$
17. $\Pr[\mathsf{H}^4_5] \leqslant \Pr[\mathsf{H}^4_6] + 0.5q^2_{\text{Enc}}/|\mathsf{KEM}.\mathcal{C}| + 0.5q^2_{\text{Enc}}/2^{\mathsf{SKE.xl}}$
18. $\Pr[\mathsf{H}^4_6] = \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD},0}(\mathcal{A})]$

Applying these in sequence to $\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD},0}(\mathcal{A})]$ gives

$$\begin{aligned}\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD}}(\mathcal{A}) \leqslant{} & \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_0) + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE}}(\mathcal{B}_{\mathsf{SKE}}) + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_1) + q_{\text{Enc}} \cdot (2\delta + \varepsilon) \\ & + (2q^2_{\text{Enc}} + 2q_{\text{Enc}}q_{\text{Dec}})/|\mathsf{KEM}.\mathcal{C}| + (q^2_{\text{Enc}} + 2q_{\text{Dec}})/2^{\mathsf{SKE.xl}}.\end{aligned}$$

The adversary $\mathcal{B}_{\mathsf{KEM}}$ considered in the theorem statement picks $d \leftarrow\!\!\$\ \{0,1\}$ and then runs $\mathcal{B}_d$. This gives $2\mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) = \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_0) + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_1)$. Statements about the adversaries' complexity can be verified by examining their code.

General summary. We can broadly view this proof as occurring in three phases, corresponding to the three reductions we provide. The first phase ($\mathsf{H}^1_0 \rightarrow \mathsf{H}^1_1$) is fairly immediate and quick; we apply the security of KEM to replace its ciphertexts with random. For the second phase ($\mathsf{H}^2_0 \rightarrow \mathsf{H}^2_3$), we want to apply the security of SKE. Doing so in the naive way requires storing the encryption query that each randomly chosen KEM ciphertext came from. To avoid this, we carefully re-arrange the game to pick these KEM ciphertexts as the output of an injection applied to the information we need to remember, allowing us to make the reduction to SKE TAM-tight.

A priori, it is surprising that a third phase ($\mathsf{H}^2_4 \rightarrow \mathsf{H}^4_3$) is needed at all. Haven't we already applied the two reductions needed to switch both components of the ciphertext with random? It turns out there is a subtle technical issue with the decryption oracle. The desired behavior is to "automatically" return the correct message for all challenge ciphertexts forwarded from encryption and to decrypt all other ciphertexts with KD.D. However, our decryption oracle after the second phase does not do this. In particular, when given a ciphertext whose KEM component is from a challenge ciphertext, but whose SKE component is fresh, the decryption oracle will decrypt the SKE ciphertext with a random key when it should be using the key encapsulated in the (random) KEM ciphertext. Switching this to the correct behavior requires a careful reduction to the \$CCA security of KEM. To perform this reduction in a memory-tight manner, again, we

```
Hybrids $\mathsf{H}^2_\kappa$ for $0 \leqslant \kappa \leqslant 5$
$\mathcal{H} \leftarrow\$ \mathsf{KD.IM}$
$K_{(\cdot)} \leftarrow\$ \mathsf{SKE.K}$
$r \leftarrow\$ \mathsf{Fcs}(\mathcal{U}, \mathsf{KEM}.\mathcal{R})$
$h^{\pm} \leftarrow\$ \mathsf{Inj}^{\pm}(T, D, R)$
$b' \leftarrow\$ \mathcal{A}^{\textsc{New},\textsc{Enc},\textsc{Dec},\mathcal{H}}$
Return $(b' = 1)$

$\textsc{Enc}(u, i, m)$
$I[u] \leftarrow I[u] \cup \{i\}$ $//\mathsf{H}^2_{[1,\infty)}$
$(ek_u, \cdot) \leftarrow \mathsf{KEM.K}(r(u))$
$c^k \leftarrow h_{u,ek_u}(i)$
$c^d \leftarrow\$ \mathsf{SKE.E}^{\mathcal{H}}(K_i, m)$ $//\mathsf{H}^2_{[0,4)}$
$c^d \leftarrow\$ \{0,1\}^{\mathsf{SKE.cl}(|m|)}$ $//\mathsf{H}^2_{[4,\infty)}$
$T[u, c^k] \leftarrow K_i$
$M[i, c^d] \leftarrow m$ $//\mathsf{H}^2_{[2,\infty)}$
Return $(c^k, c^d)$
```

```
$\textsc{Dec}(u, c)$
$(c^k, c^d) \leftarrow c$
$(ek_u, dk_u) \leftarrow \mathsf{KEM.K}(r(u))$
$i \leftarrow h^{-1}_{u,ek_u}(c^k)$ $//\mathsf{H}^2_{[1,\infty)}$
If $T[u, c^k] \neq \perp$ $//\mathsf{H}^2_{[0,1)}$
If $i \in I[u]$ $//\mathsf{H}^2_{[1,3)}$, $\mathsf{H}^2_{[5,\infty)}$
If $i \neq \perp$ $//\mathsf{H}^2_{[3,5)}$
    If $M[i, c^d] \neq \perp$ $//\mathsf{H}^2_{[2,\infty)}$
        Return $M[i, c^d]$ $//\mathsf{H}^2_{[2,\infty)}$
    Return $\mathsf{SKE.D}^{\mathcal{H}}(T[u, c^k], c^d)$ $//\mathsf{H}^2_{[0,1)}$
    Return $\mathsf{SKE.D}^{\mathcal{H}}(K_i, c^d)$ $//\mathsf{H}^2_{[1,\infty)}$
$K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$
If $K = \perp$ then return $\perp$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$

$\textsc{New}(u)$
$(ek_u, \cdot) \leftarrow \mathsf{KEM.K}(r(u))$
Return $ek_u$
```

```
Adversary $\mathcal{B}^{\textsc{Enc},\textsc{Dec},\mathcal{H}}_{\mathsf{SKE}}$
$r \leftarrow\$ \mathsf{Fcs}(\mathcal{U}, \mathsf{KEM}.\mathcal{R})$
$h^{\pm} \leftarrow\$ \mathsf{Inj}^{\pm}(T, D, R)$
$b' \leftarrow\$ \mathcal{A}^{\textsc{SimNew},\textsc{SimEnc},\textsc{SimDec},\mathcal{H}}$
Return $b'$

$\textsc{SimNew}(u)$
$(ek_u, \cdot) \leftarrow \mathsf{KEM.K}(r(u))$
Return $ek_u$

$\textsc{SimEnc}(u, i, m)$
$(ek_u, \cdot) \leftarrow \mathsf{KEM.K}(r(u))$
$c^k \leftarrow h_{u,ek_u}(i)$
$c^d \leftarrow \textsc{Enc}(i, i, m)$
Return $(c^k, c^d)$

$\textsc{SimDec}(u, c)$
$(c^k, c^d) \leftarrow c$
$(ek_u, dk_u) \leftarrow \mathsf{KEM.K}(r(u))$
$i \leftarrow h^{-1}_{u,ek_u}(c^k)$
If $i \neq \perp$
    Return $\textsc{Dec}(i, c^d)$
$K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$
If $K = \perp$ then return $\perp$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$
```

Fig. 22. Hybrids H^2_κ (**Left**) and adversary $\mathcal{B}_{\mathsf{SKE}}$ (**Right**) used for proof of Theorem 7. Adversary $\mathcal{B}_{\mathsf{SKE}}$ is used to transition from H^2_3 to H^2_4. Recall that $T = \mathcal{U} \times \mathsf{KEM.Ek}$, $D_{(u,ek)} = \mathcal{I} = [q_{\textsc{Enc}}]$, and $R_{(u,ek)} = \mathsf{KEM}.\mathcal{C}(ek)$.

need to first carefully re-write the game, this time to have the random SKE ciphertexts be the output of a random injection applied to the message they are supposed to encrypt.

While we do not refer to it as its own phase, there are a few further small transitions ($\mathsf{H}^4_4 \to \mathsf{H}^4_6$) after the last reduction which undo some of our memory-tightness tricks to reach the conclusion of the proof.

Transition to H^1_0 (start phase 1). We compare the hybrid game H^1_0 to the game $\mathsf{G}^{\text{mu-\$cca}}_{\mathsf{KD},1}$ defined in Fig. 3. In both games, the encryption oracle returns "real" encryptions of the message. In $\mathsf{G}^{\text{mu-\$cca}}_{\mathsf{KD},1}$, the messages underlying challenge ciphertexts are stored in a table which is used to respond to decryption queries on these ciphertexts. In H^1_0, we instead use a table to store the key underlying the KEM component of the ciphertext. This key is used for any decryption query involving that part of the ciphertext. The behavior of the game only differs if there is a correctness error in any ciphertext produced by the encryption algorithm. We assume δ-correctness of KEM and perfect correctness of SKE, so we get $\Pr[\mathsf{G}^{\text{mu-\$cca}}_{\mathsf{KD},1}(\mathcal{A})] \leqslant \Pr[\mathsf{H}^1_0] + q_{\textsc{Enc}} \cdot \delta$.

Transition from H^1_0 to H^1_1 (end phase 1). In game H^1_1, we switch from using the real output of the encapsulation algorithm to sampling the KEM ciphertext and generated key at random. To bound the difference between the games H^1_0 and H^1_1 we use the security of KEM. We construct the adversary $\mathcal{B}_0$ in Fig. 21. Adversary $\mathcal{B}_0$ perfectly simulates game H^1_κ for adversary $\mathcal{A}$ when the bit in its game is $1 - \kappa$. Note that the table T in games H^1_κ is the same as the table T used of the game played by $\mathcal{B}_0$. Therefore, $\Pr[\mathsf{H}^1_0] = \Pr[\mathsf{H}^1_1] + \mathsf{Adv}^{\text{mu-\$cca}}_{\mathsf{KEM}}(\mathcal{B}_0)$.

Transition from H^1_1 to H^2_0 (start phase 2). Now we start phase 2 of the proof where we move toward using the security of SKE to replace its ciphertexts with random strings of the same length. We want to think of the random key K used by SKE during an encryption query of the form $(u, i, \cdot)$ as belonging to an SKE user whose identity is i. Here we are using the fact that $\mathcal{A}$ is challenge respecting so we know i will never repeat across queries. To be able to properly make queries for user i in Dec, we will first have to transition

```
Hybrids $\mathsf{H}^3_\kappa$ for $0 \leqslant \kappa \leqslant 5$
$\mathcal{H} \leftarrow\!\!\$\ \mathsf{KD.IM}$
$K_{(\cdot)} \leftarrow\!\!\$\ \mathsf{SKE.K}$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\!\!\$\ \mathsf{KEM.K}$
$h^{\pm} \leftarrow\!\!\$\ \mathsf{Inj}^{\pm}(T, D, R)$ $//\mathsf{H}^3_{[0,5)}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{New},\textsc{Enc},\textsc{Dec},\mathcal{H}}$
Return $(b' = 1)$
```

```
$\textsc{Enc}(u, i, m)$
$I[u] \leftarrow I[u] \cup \{i\}$
$c^k \leftarrow h_{u,ek_u}(i)$ $//\mathsf{H}^3_{[0,5)}$
$c^k \leftarrow\!\!\$\ \mathsf{KEM}.\mathcal{C}(ek_u)$ $//\mathsf{H}^3_{[5,\infty)}$
$c^d \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SKE.cl}(|m|)}$
$T[u, c^k] \leftarrow K_i$
$M[\mathbf{lab}(u, c^k), c^d] \leftarrow m$
Return $(c^k, c^d)$

Label function lab
$\mathbf{lab}(u, c^k) = h^{-1}_{u,ek_u}(c^k)$ $//\mathsf{H}^3_{[0,1)}$
$\mathbf{lab}(u, c^k) = u, h^{-1}_{u,ek_u}(c^k)$ $//\mathsf{H}^3_{[1,3)}$
$\mathbf{lab}(u, c^k) = u, c^k$ $//\mathsf{H}^3_{[3,\infty)}$
```

```
$\textsc{Dec}(u, c)$
$(c^k, c^d) \leftarrow c$
$i \leftarrow h^{-1}_{u,ek_u}(c^k)$
If $i \in I[u]$ and $M[\mathbf{lab}(u, c^k), c^d] \neq \bot$ $//\mathsf{H}^3_{[0,2)}$
If $M[\mathbf{lab}(u, c^k), c^d] \neq \bot$ $//\mathsf{H}^3_{[2,\infty)}$
    Return $M[\mathbf{lab}(u, c^k), c^d]$
If $i \in I[u]$ then $K \leftarrow K_i$ $//\mathsf{H}^3_{[0,4)}$
If $T[u, c^k] \neq \bot$ then $K \leftarrow T[u, c^k]$ $//\mathsf{H}^3_{[4,\infty)}$
Else $K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$
If $K = \bot$ then return $\bot$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$

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

Fig. 23. Hybrids H^3_κ for proof of Theorem 7. Recall that $T = \mathcal{U} \times \mathsf{KEM.Ek}$, $D_{(u,ek)} = [q_{\textsc{Enc}}]$ and $R_{(u,ek)} = \mathsf{KEM}.\mathcal{C}(ek)$.

to a hybrid where c^k is chosen using an injection applied to i so that i can later be recovered when c^k is queried to Dec. This will occur over several game hops.

The ways in which game H^2_0 (Fig. 22) differs from game H^1_1 have been highlighted grey in H^2_0. We summarize the interesting changes. To succinctly remember (ek_u, dk_u) for each u we generate them as $\mathsf{KEM.K}(r(u))$ where r is a random function. We can then re-derive these keys whenever needed. We've removed the table C. As $\mathcal{A}$ is challenge respecting, this table was unused.

The change which does affect its view is the switch from sampling the ciphertext c^k at random to assigning it as the output of a random injection $h_{u,ek_u}(\cdot)$ from $\mathcal{I} = [q_{\textsc{Enc}}]$ to $\mathsf{KEM}.\mathcal{C}(ek_u)$. Because $\mathcal{A}$ is challenge respecting, it never repeats i across queries to Enc and we can use the switching lemma to get $\Pr[\mathsf{H}^1_1] \leqslant \Pr[\mathsf{H}^2_0] + 0.5 q^2_{\textsc{Enc}}/|\mathsf{KEM}.\mathcal{C}|$.

Transitions from H^2_0 through H^2_3. Next we move toward adding a look-up table M to mirror the table in $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE},b}$. It will be indexed by a SKE user identifier i and challenge ciphertext c^d and store the message encrypted by that ciphertext.

In H^2_1, we start by getting rid of the table T. We start storing sets $I[u]$ which contain i iff the i-th encryption query was to user u. In Dec, the check whether $T[u, c^k]$ is $\bot$ is replaced with a check whether $i = h^{-1}_{u,ek_u}(c^k) \in I[u]$ and if this check passes we directly use K_i rather than recovering it from T. If $T[u, c^k] \neq \bot$ and $c^k = h_{u,ek_u}(i)$ then it must have been set by the i-th encryption query of the form $\textsc{Enc}(u, i, \cdot)$, and so $i \in I[u]$. The same change of logic works in reverse, noting that this i-th encryption query for user u would have picked $c^k = h_{u,ek_u}(i)$. Hence, $\Pr[\mathsf{H}^2_0] = \Pr[\mathsf{H}^2_1]$.

Next, in H^2_2, we add the aforementioned table M to Enc and Dec. If $i \in I[u]$, then the i-th encryption query had the form $(\cdot, c^d) = \textsc{Enc}(u, i, m)$. Then, by the perfect correctness of SKE, $M[i, c^d] = m = \mathsf{SKE.D}^{\mathcal{H}'}(K_i, c^d)$, so the use of M does not change the behavior of the game. We get $\Pr[\mathsf{H}^2_1] = \Pr[\mathsf{H}^2_2]$.

For H^2_3, we make a modification that does change the behavior of the game. The check whether $i \in I[u]$ is replaced with a check whether $i = h^{-1}_{u,ek_u}(c^k)$ is non-$\bot$. This *can* cause misbehavior of the game, but the only way to trigger the difference is if the i-th encryption query was not to user u (or has not occurred yet), but $\mathcal{A}$ anyway manages to make a query $\textsc{Dec}(u, (h_{u,ek_u}(i), \cdot))$.

We bound this using a union bound over the decryption queries. In H^2_2, the view of $\mathcal{A}$ can only depend on h_{u,ek_u} applied to inputs in $I[u]$. For a query $\textsc{Dec}(u, (c^k, \cdot))$ where $c^k \notin h_{u,ek_u}(I[u])$ we can bound the probability of $i \neq \bot$ by

$$\frac{|h_{u,ek_u}([q_{\textsc{Encap}}]) \setminus h_{u,ek_u}(I[u])|}{|\mathsf{KEM}.\mathcal{C}(ek_u) \setminus h_{u,ek_u}(I[u])|} = \frac{q_{\textsc{Enc}} - |I[u]|}{|\mathsf{KEM}.\mathcal{C}(ek_u)| - |I[u]|} \leqslant \frac{q_{\textsc{Enc}}}{|\mathsf{KEM}.\mathcal{C}|}.$$

Thus we get $\Pr[\mathsf{H}^2_2] \leqslant \Pr[\mathsf{H}^2_3] + q_{\textsc{Enc}} q_{\textsc{Dec}}/|\mathsf{KEM}.\mathcal{C}|$.

Transition from H^2_3 to H^2_4 (end phase 2). In game H^2_4, we change Enc so that c^d is sampled at random rather than produced by $\mathsf{SKE.E}$. To bound the difference between H^2_3 and H^2_4, we construct the adversary $\mathcal{B}_{\mathsf{SKE}}$ in Fig. 22 against the security of SKE.

For encryption and decryption of c^d, the reduction adversary invokes its own encryption and decryption oracles. Because i is incremented with each encryption, $\mathcal{B}_{\mathsf{SKE}}$ is making at most one encryption query per user as claimed in the theorem. The adversary uses the random function r to sample (ek_u, dk_u) and the random injection h to sample c^k while using little memory.

We claim that when interacting with $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE},b}$, $\mathcal{B}_{\mathsf{SKE}}$ perfectly simulates game H^2_{4-b} for $\mathcal{A}$. These claims were verified by manually plugging the code of $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE},b}$ into $\mathcal{B}_{\mathsf{SKE}}$'s oracle queries to compare with H^2_{4-b}. Therefore, $\Pr[\mathsf{H}^2_3] = \Pr[\mathsf{H}^2_4] + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE}}(\mathcal{B}_{\mathsf{SKE}})$.

Transition from H^2_4 to H^2_5 (start phase 3). Phase 3 of the proof will begin in earnest with H^3_0. Before we get there we quickly switch back from Dec checking if $i \neq \perp$ to checking if $i \in I[u]$. Using analogous analysis to when we switched the other way we get that $\Pr[\mathsf{H}^2_4] \leqslant \Pr[\mathsf{H}^2_5] + q_{\text{Enc}} q_{\text{Dec}} / |\mathsf{KEM}.\mathcal{C}|$.

Transition from H^2_5 to H^3_0. Now we move to H^3_0 (Fig. 23) where grey highlighting has been used to indicate differences from H^2_5 that were made to set up for future modifications. First, we switched away from using r to sample (ek_u, dk_u). Second, we rewrote the indexing into table M to make use of a “label” function **lab**. For the definition of **lab** in H^3_0, this results in M being indexed no differently than it was before. Finally, we re-organized the conditional logic in Dec. The initial nested if statements were split into two separate if conditions, the first for when both of the nested if’s are true and the second for if just the outer if statement is true. We rewrote the later part of Dec so there is a single invocation of $\mathsf{SKE.D}$ with proceeding code deciding which key to use. The modifications do not change the behavior of the game and so $\Pr[\mathsf{H}^2_5] = \Pr[\mathsf{H}^3_0]$

Transitions H^3_0 through H^3_4. In the next steps toward transitioning toward our game matching $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD},0}$, we rewrite the behavior of our M table to match the table in that game which means that it should be indexed by (u, c^k, c^d). Moreover, we switch c^k back to being sampled at random. These changes occur over multiple transitions.

In these transitions we change the definition of the “label function” **lab** used to index into table M. For the first transition, we add u to the output of the function and claim $\Pr[\mathsf{H}^3_0] = \Pr[\mathsf{H}^3_1]$. This follows because if the i-th encryption query is of the form $\text{Enc}(v, i, \cdot)$, then in H^3_0 the table entries $M[i, \cdot]$ will only be accessed in that encryption query and in decryption queries of the form $\text{Dec}(v, \cdot)$. It is clear that $M[i, \cdot]$ is only written into during the i-th encryption query. To see that no $\text{Dec}(u, \cdot)$ query with $u \neq v$ will read from $M[i, \cdot]$ note that Dec only accesses $M[i, \cdot]$ after the check $i \in I[u]$ succeeds and $I[v]$ will be the only set containing i. Hence, $\Pr[\mathsf{H}^3_0] = \Pr[\mathsf{H}^3_1]$

For the transition to H^3_2, the $i \in I[u]$ check is removed from the first if statement in Dec. Note that $M[u, i, c^d]$ can only be assigned a non-$\perp$ by the i-th Enc query being to u. This query would have added i to $I[u]$ and so $\Pr[\mathsf{H}^3_1] = \Pr[\mathsf{H}^3_2]$.

For the transition to H^3_3, **lab**’s output $u, h^{-1}_{u,ek_u}(c^k)$ is changed to u, c^k. Note that with u fixed, h^{-1}_{u,ek_u} is an injection. Hence, indexing into the table with $h^{-1}_{u,ek_u}(c^k)$ or c^k is equivalent. This gives $\Pr[\mathsf{H}^3_2] = \Pr[\mathsf{H}^3_3]$.

In H^3_4, we switch from the conditional “If $i \in I[u]$ then $K \leftarrow K_i$” in Dec to “If $T[u, c^k] \neq \perp$ then $K \leftarrow T[u, c^k]$”. By examining Enc we can verify that $i \in I[u]$ iff $T[u, h_{u,ek_u}(i)] = K_i$ and otherwise $T[u, h_{u,ek_u}(i)]$ is $\perp$. Hence $\Pr[\mathsf{H}^3_3] = \Pr[\mathsf{H}^3_4]$.

Finally, moving to H^3_5 we switch from using the injection h to pick c^k to picking it uniformly at random from $\mathsf{KEM}.\mathcal{C}$. By the switching lemma, $\Pr[\mathsf{H}^3_4] \leqslant \Pr[\mathsf{H}^3_5] + 0.5 q^2_{\text{Enc}} / |\mathsf{KEM}.\mathcal{C}|$.

Transition from H^3_5 to H^4_0. It is tempting at this point to believe that we are done, as (c^k, c^d) are sampled at random and the table M ensures that decapsulation queries are properly responded to as in $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD},0}$. Unfortunately, there are some ciphertexts that Dec handles improperly.[11] If the i-th encryption query was of the form $(c^k, c^d) \leftarrow \text{Enc}(v, i, m)$, then any decryption query $\text{Dec}(v, (c^k, c'))$ for $c' \neq c^d$ will

[11] This is a surprisingly subtle detail and easy to miss when not writing the proof with careful detail.

```
Hybrids $\mathsf{H}^4_\kappa$ for $0 \leqslant \kappa \leqslant 6$
$\mathcal{H} \leftarrow\!\!\$\ \mathsf{KD.IM}$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\!\!\$\ \mathsf{KEM.K}$
$g^{\pm} \leftarrow\!\!\$\ \mathsf{Inj}^{\pm}(T', D', R')$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{New},\textsc{Enc},\textsc{Dec},\mathcal{H}}$
Return $(b' = 1)$

$\textsc{Enc}(u, i, m)$
$c^k \leftarrow\!\!\$\ \mathsf{KEM}.\mathcal{C}(ek_u)$ $//\mathsf{H}^4_{[0,2)}, \mathsf{H}^4_{[4,\infty)}$
$K \leftarrow\!\!\$\ \mathsf{KEM}.\mathcal{K}$ $//\mathsf{H}^4_{[0,2)}$
$(c^k, K) \leftarrow\!\!\$\ \mathsf{KEM.E}^{\mathcal{H}}(ek_u)$ $//\mathsf{H}^4_{[2,4)}$
$c^d \leftarrow g_{u,c^k,|m|}(m)$ $//\mathsf{H}^4_{[0,6)}$
$c^d \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SKE.cl}(|m|)}$ $//\mathsf{H}^4_{[6,\infty)}$
$T[u, c^k] \leftarrow K$ $//\mathsf{H}^4_{[0,3)}$
$M[u, c^k, c^d] \leftarrow m$
Return $(c^k, c^d)$
```

```
$\textsc{Dec}(u, c)$
$(c^k, c^d) \leftarrow c$
If $M[u, c^k, c^d] \neq \bot$ $//\mathsf{H}^4_{[0,1)}, \mathsf{H}^4_{[5,\infty)}$
    Return $M[u, c^k, c^d]$ $//\mathsf{H}^4_{[0,1)}, \mathsf{H}^4_{[5,\infty)}$
$m \leftarrow g^{-1}_{u,c^k,\mathsf{SKE.ml}(|c^d|)}(c^d)$ $//\mathsf{H}^4_{[1,5)}$
If $m \neq \bot$ then return $m$ $//\mathsf{H}^4_{[1,5)}$
If $T[u, c^k] \neq \bot$ then $K \leftarrow T[u, c^k]$ $//\mathsf{H}^4_{[0,3)}$
Else $K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$ $//\mathsf{H}^4_{[0,3)}$
$K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$ $//\mathsf{H}^4_{[3,\infty)}$
If $K = \bot$ then return $\bot$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$

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

```
Adversary $\mathcal{B}_1^{\textsc{New},\textsc{Encap},\textsc{Decap},\mathcal{H}}$
$g^{\pm} \leftarrow\!\!\$\ \mathsf{Inj}^{\pm}(T', D', R')$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{New},\textsc{SimEnc},\textsc{SimDec},\mathcal{H}}$
Return $1 - b'$

$\textsc{SimEnc}(u, i, m)$
$(c^k, \cdot) \leftarrow \textsc{Encap}(u, i)$
$c^d \leftarrow g_{u,c^k,|m|}(m)$
Return $(c^k, c^d)$

$\textsc{SimDec}(u, c)$
$(c^k, c^d) \leftarrow c$
$m \leftarrow g^{-1}_{u,c^k,\mathsf{SKE.ml}(|c^d|)}(c^d)$
If $m \neq \bot$ then return $m$
$K \leftarrow \textsc{Decap}(u, c^k)$
If $K = \bot$ then return $\bot$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$
```

Fig. 24. Hybrids H^4_κ (**Left**) and adversary $\mathcal{B}_1$ (**Right**) used for proof of Theorem 7. Adversary $\mathcal{B}_1$ is used to transition from H^4_1 to H^4_2. Recall that $T' = \mathcal{U} \times \bigcup_{ek \in \mathsf{KEM.Ek}} \mathsf{KEM}.\mathcal{C}(ek) \times \mathbb{N}$, $D'_{(u,c^k,l)} = \{0,1\}^l$, and $R'_{(u,c^k,l)} = \{0,1\}^{\mathsf{SKE.cl}(l)}$.

decrypt c^d using the randomly chosen key K_i because $T[u, c^k] \neq \bot$.[12] The decryption oracle in $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD},0}$ would instead decrypt c^d using $K = \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$.

To change this decryption behavior we are, perhaps surprisingly, going to again make use of the security of KEM. For this reduction to be memory-tight we cannot store the table M, so our next game transitions are aimed at allowing simulation of it in a memory-tight manner. In particular, we are going to switch to c^d being the output of an injection g applied to the message. Then in decryption we can use g^{-1} in place of M.

To start, we transition to H^4_0 (Fig. 24), the first hybrid in our final set of hybrid games. We used grey highlighting to indicate the changes. The only change which is not simple rewriting is the treatment of c^d which is now set to $g_{u,c^k,|m|}(m)$ where g is an injection. Let us first consider an intermediate hybrid H^4_{-1} where instead g was a random *function* chosen as $g \leftarrow\!\!\$\ \mathsf{Fcs}^{\pm}(T', D', R')$. The switch to this hybrid would be detectable only if g is called with the same inputs twice, which requires the same random c^k to be sampled twice. This gives that $\Pr[\mathsf{H}^3_5] \leqslant \Pr[\mathsf{H}^4_{-1}] + 0.5q^2_{\textsc{Enc}}/|\mathsf{KEM}.\mathcal{C}|$.[13] Now to move to H^4_0 we apply the switching lemma to use a random injection instead which gives $\Pr[\mathsf{H}^4_{-1}] \leqslant \Pr[\mathsf{H}^4_0] + 0.5q^2_{\textsc{Enc}}/2^{\mathsf{SKE.xl}}$.

Transition from H^4_0 to H^4_1. In H^4_1, we replace the use of $M[u, c^k, c^d]$ in $\textsc{Dec}$ with $g^{-1}_{u,c^k,\mathsf{SKE.ml}(|c^d|)}(c^d)$. Examining, $\textsc{Enc}$ we can see that if $M[u, c^k, c^d] \neq \bot$, then $g^{-1}_{u,c^k,\mathsf{SKE.ml}(|c^d|)}(c^d) = M[u, c^k, c^d]$.[14] The other direction does not necessarily hold, so $\mathcal{A}$ can detect this game transition if it is able to query $\textsc{Dec}(u, (c^k, c^d))$ where (c^k, c^d) was not an earlier response to an $\textsc{Enc}(u, \cdot)$ query (so $M[u, c^k, c^d] = \bot$), yet $g^{-1}_{u,c^k,\mathsf{SKE.ml}(|c^d|)}(c^d) \neq \bot$. In H^4_0, the view of $\mathcal{A}$ only depends on g through queries to $\textsc{Enc}$ so we analyze the probability of this bad event there, using a union bound over $\textsc{Dec}$ queries. During a $\textsc{Dec}(u, (c^k, c^d))$ query where $M[u, c^k, c^d] = \bot$, let n denote the number of earlier $\textsc{Enc}(u, m)$ queries with $|m| = \mathsf{SKE.ml}(|c^d|)$ which returned some ciphertext of the form $(c^k, \cdot)$. Then we can bound the probability of $g^{-1}_{u,c^k,\mathsf{SKE.ml}(|c^d|)}(c^d) \neq \bot$ by

$$\frac{2^{\mathsf{SKE.ml}(|c^d|)} - n}{2^{|c^d|} - n} \leqslant \frac{2^{\mathsf{SKE.ml}(|c^d|)}}{2^{|c^d|}} \leqslant \frac{1}{2^{\mathsf{SKE.xl}}}.$$

Thus we get $\Pr[\mathsf{H}^4_0] \leqslant \Pr[\mathsf{H}^4_1] + q_{\textsc{Dec}}/2^{\mathsf{SKE.xl}}$.

[12] Technically, c^k may have been returned by multiple encryption queries. We refer to the most recent i.

[13] With careful analysis we could combine this with the H^3_4 to H^3_5 transition to shave $q^2_{\textsc{Enc}}/|\mathsf{KEM}.\mathcal{C}|$ off of our bound.

[14] For simplicity we assume ciphertexts of all lengths are possible for SKE.

Transition from H^4_1 to H^4_2 (end phase 3). Now we switch in Enc from (c^d, K) being randomly sampled to them being the output of $\mathsf{KEM.E}$. This is achieved using a reduction to the security of KEM. Consider the adversary $\mathcal{B}_1$ shown in Fig. 24.

For encryption queries, it uses its Encap oracle to obtain c^k while ignoring the output key and picks c^d using g. For decryption queries it first tries to "decrypt" c^d with g. Failing that it queries Decap to obtain a key it uses to decrypt c^d. The table T used in H^4 matches the corresponding table inside of $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM},b}$.

By plugging the code of $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM},b}$ we verify that the view of $\mathcal{A}$ when run by $\mathcal{B}_{\mathsf{KEM}}$ in that game perfectly matches its view in game H^4_{b+1}. Note that $\mathcal{B}_{\mathsf{KEM}}$ flips the bit output by $\mathcal{A}$. This give us that $\Pr[\mathsf{H}^4_{b+1}] = 1 - \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM},b}(\mathcal{B}_{\mathsf{KEM}})]$ and so $\Pr[\mathsf{H}^4_1] = \Pr[\mathsf{H}^4_2] + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}})$.

Transitions H^4_2 through H^4_7 (clean up). The rest of the proof is a matter of "cleaning up" various changes we made for the above reduction. First, unless there is a correctness error in KEM, if $T[u, c^k] \neq \perp$ then it has the value $\mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$. Thus both branches of the if statement based on T in Dec are equivalent when correctness errors do not occur so we simplify this is H^4_3. Using δ-correctness, we bound $\Pr[\mathsf{H}^4_2] \leqslant \Pr[\mathsf{H}^4_3] + q_{\text{Enc}} \cdot \delta$.

Now, we switch c^k back from being generated by $\mathsf{KEM.E}$ to being randomly sampled. Note that K is unused now that T has been removed. Hence, the ε-uniformity of KEM gives $\Pr[\mathsf{H}^4_3] \leqslant \Pr[\mathsf{H}^4_4] + q_{\text{Enc}} \cdot \varepsilon$.

In H^4_6, we undo the switch from $M[u, c^k, c^d]$ to $g^{-1}_{u,c^k,\mathsf{SKE.ml}(|c^d|)}(c^d)$ in Dec. Then in H^4_7, we undo the switch from c^d being random to being the output of g. By the same logic as before, $\Pr[\mathsf{H}^4_4] \leqslant \Pr[\mathsf{H}^4_5] + q_{\text{Dec}}/2^{\mathsf{SKE.xl}}$ and $\Pr[\mathsf{H}^4_5] \leqslant \Pr[\mathsf{H}^4_6] + 0.5q^2_{\text{Enc}}/2^{\mathsf{SKE.xl}} + 0.5q^2_{\text{Enc}}/|\mathsf{KEM}.\mathcal{C}|)$.

Now we examine H^4_7. In Enc it picks (c^k, c^d) at random and stores the message in $M[u, c^k, c^d]$. This message is returned using M if a query $\text{Dec}(u, (c^k, c^d))$ is made. Otherwise, Dec honestly decrypts. This is exactly the behavior of $\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD},0}$ and so $\Pr[\mathsf{H}^4_6] = \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD},0}(\mathcal{A})]$, concluding the proof. □

H Proof of Theorem 8 (KEM/DEM Construction, Challenge Repeating)

We prove Theorem 8 via a sequence of hybrid games shown in Fig. 25, 25, 26, and 27. We establish the following bounds relating the different hybrids.

1. $\Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD},1}(\mathcal{A})] \leqslant \Pr[\mathsf{H}^1_0] + q_{\text{Enc}} \cdot \delta$
2. $\Pr[\mathsf{H}^1_0] = \Pr[\mathsf{H}^1_1] + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_0)$
3. $\Pr[\mathsf{H}^1_1] \leqslant \Pr[\mathsf{H}^2_0] + q^2_{\text{Enc}}/2^{\gamma+1} + q^2_{\text{Enc}}/(2|\mathsf{KEM}.\mathcal{C}|)$
4. $\Pr[\mathsf{H}^2_0] = \Pr[\mathsf{H}^2_1] = \Pr[\mathsf{H}^2_2]$
5. $\Pr[\mathsf{H}^2_2] \leqslant \Pr[\mathsf{H}^2_3] + q_{\text{Dec}}|\mathcal{I}| \cdot 2^{\gamma}/|\mathsf{KEM}.\mathcal{C}|$
6. $\Pr[\mathsf{H}^2_3] = \Pr[\mathsf{H}^2_4] + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{SKE}}(\mathcal{B}_{\mathsf{SKE}})$
7. $\Pr[\mathsf{H}^2_4] \leqslant \Pr[\mathsf{H}^2_5] + q_{\text{Dec}}|\mathcal{I}| \cdot 2^{\gamma}/|\mathsf{KEM}.\mathcal{C}|$
8. $\Pr[\mathsf{H}^2_5] = \Pr[\mathsf{H}^3_0] = \Pr[\mathsf{H}^3_1]$
9. $\Pr[\mathsf{H}^3_1] = \Pr[\mathsf{H}^3_2] = \Pr[\mathsf{H}^3_3] = \Pr[\mathsf{H}^3_4]$

10. $\Pr[\mathsf{H}^3_4] \leqslant \Pr[\mathsf{H}^3_5] + 0.5q^2_{\text{Enc}}/|\mathsf{KEM}.\mathcal{C}|$

10.5 $\Pr[\mathsf{H}^3_5] \leqslant \Pr[\mathsf{H}^3_6] + q^2_{\text{Enc}}/2^{\gamma+1}$

11. $\Pr[\mathsf{H}^3_6] \leqslant \Pr[\mathsf{H}^4_0] + q^2_{\text{Enc}}/|\mathsf{KEM}.\mathcal{C}| + q^2_{\text{Enc}}/2^{\mathsf{SKE.xl}+1}$
12. $\Pr[\mathsf{H}^4_0] \leqslant \Pr[\mathsf{H}^4_1] + q_{\text{Dec}}/2^{\mathsf{SKE.xl}}$
13. $\Pr[\mathsf{H}^4_1] = \Pr[\mathsf{H}^4_2] + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KEM}}(\mathcal{B}_1)$
14. $\Pr[\mathsf{H}^4_2] \leqslant \Pr[\mathsf{H}^4_3] + q_{\text{Enc}} \cdot \delta$
15. $\Pr[\mathsf{H}^4_3] \leqslant \Pr[\mathsf{H}^4_4] + q_{\text{Enc}} \cdot \varepsilon$
16. $\Pr[\mathsf{H}^4_4] \leqslant \Pr[\mathsf{H}^4_5] + q_{\text{Dec}}/2^{\mathsf{SKE.xl}}$
17. $\Pr[\mathsf{H}^4_5] \leqslant \Pr[\mathsf{H}^4_6] + q^2_{\text{Enc}}/(2|\mathsf{KEM}.\mathcal{C}|) + q^2_{\text{Enc}}/2^{\mathsf{SKE.xl}+1}$
18. $\Pr[\mathsf{H}^4_6] = \Pr[\mathsf{G}^{\mathsf{mu\text{-}\$cca}}_{\mathsf{KD},0}(\mathcal{A})]$

The adversary $\mathcal{B}_{\mathsf{KEM}}$ is obtained by randomly choosing to run either $\mathcal{B}_0$ or $\mathcal{B}_1$. Combining these gives the claimed bound. The claimed complexity of the attackers can be seen by considering their code.

These hybrids are largely based off of the hybrids used the prove Theorem 7 in Appendix G. The only changes are because the prior proof considered a challenge-respecting adversary, whereas the proof in this section allows the adversary to repeat queries to its Enc oracle. We have made an effort to make "matching" games have the same name (at times adding games with negative indices or skipping some indices to enable this). In the hybrids and adversary we use blue text to indicate specific places where the code differs in our new hybrids. Rather than giving a self-contained proof of each of the probability claims above, we emphasize only the places that the analysis differs from the earlier proof. The reader can understand skipped explanations by referring to the corresponding text from the previous proof.

```
$\underline{\text{Hybrids } \mathsf{H}^1_\kappa \text{ for } 0 \leqslant \kappa \leqslant 1}$
$\mathcal{H} \leftarrow\!\!\$ \ \mathsf{KD.IM}$
$\tilde{r} \leftarrow\!\!\$ \ \mathsf{Fcs}(\mathcal{U} \times \mathcal{I} \times \{0,1\}^*, \mathsf{SKE}.\mathcal{R})$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\!\!\$ \ \mathsf{KEM.K}$
$b' \leftarrow\!\!\$ \ \mathcal{A}^{\textsc{New},\textsc{Enc},\textsc{Dec},\mathcal{H}}$
Return $(b' = 1)$

$\underline{\textsc{Enc}(u, i, m)}$
If $C[u,i,m] \neq \perp$
    Return $C[u,i,m]$
$(c^k, K) \leftarrow\!\!\$ \ \mathsf{KEM.E}^{\mathcal{H}}(ek_u)$ $//\mathsf{H}^1_{[0,1)}$
$c^k \leftarrow\!\!\$ \ \mathsf{KEM}.\mathcal{C}(ek_u)$ $//\mathsf{H}^1_{[1,\infty)}$
$K \leftarrow\!\!\$ \ \mathsf{KEM}.\mathcal{K}$ $//\mathsf{H}^1_{[1,\infty)}$
$c^d \leftarrow \mathsf{SKE.E}^{\mathcal{H}}(K, m; \tilde{r}(u,i,m))$
$T[u, c^k] \leftarrow K$
$C[u,i,m] \leftarrow (c^k, c^d)$
Return $(c^k, c^d)$

$\underline{\textsc{Dec}(u, c)}$
$(c^k, c^d) \leftarrow c$
If $T[u, c^k] \neq \perp$
    $K \leftarrow T[u, c^k]$
    Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$
$K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$
If $K = \perp$ then return $\perp$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$

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

$\underline{\text{Adversary } \mathcal{B}_0^{\textsc{New},\textsc{Encap},\textsc{Decap},\mathcal{H}}}$
$\tilde{r} \leftarrow\!\!\$ \ \mathsf{Fcs}(\mathcal{U} \times \mathcal{I} \times \{0,1\}^*, \mathsf{SKE}.\mathcal{R})$
$d' \leftarrow\!\!\$ \ \mathcal{A}^{\textsc{New},\textsc{SimEnc},\textsc{SimDec},\mathcal{H}}$
Return $d'$

$\underline{\textsc{SimEnc}(u, i, m)}$
$(c^k, K) \leftarrow \textsc{Encap}(u, (i, m))$
$c^d \leftarrow\!\!\$ \ \mathsf{SKE.E}^{\mathcal{H}}(K, m)$
$c^d \leftarrow \mathsf{SKE.E}^{\mathcal{H}}(K, m; \tilde{r}(u,i,m))$
Return $(c^k, c^d)$

$\underline{\textsc{SimDec}(u, c)}$
$(c^k, c^d) \leftarrow c$
$K \leftarrow \textsc{Decap}(u, c^k)$
If $K = \perp$ then return $\perp$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$
```

```
$\underline{\text{Hybrids } \mathsf{H}^2_\kappa \text{ for } -2 \leqslant \kappa \leqslant 5}$
$\mathcal{H} \leftarrow\!\!\$ \ \mathsf{KD.IM}$
$\tilde{r} \leftarrow\!\!\$ \ \mathsf{Fcs}(\mathcal{U} \times \mathcal{I} \times \{0,1\}^*, \mathsf{SKE}.\mathcal{R})$
$K_{(\cdot,\cdot,\cdot)} \leftarrow\!\!\$ \ \mathsf{SKE.K}$
$r \leftarrow\!\!\$ \ \mathsf{Fcs}(\mathcal{U}, \mathsf{KEM}.\mathcal{R})$
$h \leftarrow\!\!\$ \ \mathsf{Fcs}(T, D \times \{0,1\}^*, R)$ $//\mathsf{H}^2_{[-2,0)}$
$t \leftarrow \mathbb{I}$ $//\mathsf{H}^2_{[-2,-1)}$
$h^{\pm} \leftarrow\!\!\$ \ \mathsf{Inj}^{\pm}(T, D \times \{0,1\}^{\gamma}, R)$ $//\mathsf{H}^2_{[-0,\infty)}$
$t \leftarrow\!\!\$ \ \mathsf{Fcs}(\{0,1\}^*, \{0,1\}^{\gamma})$ $//\mathsf{H}^2_{[-1,\infty)}$
$g \leftarrow\!\!\$ \ \mathsf{Fcs}(T', D', R')$
$b' \leftarrow\!\!\$ \ \mathcal{A}^{\textsc{New},\textsc{Enc},\textsc{Dec},\mathcal{H}}$
Return $(b' = 1)$

$\underline{\textsc{Enc}(u, i, m)}$
$I[u] \leftarrow I[u] \cup \{(i, t(m))\}$ $//\mathsf{H}^2_{[1,\infty)}$
$(ek_u, \cdot) \leftarrow \mathsf{KEM.K}(r(u))$
$c^k \leftarrow h_{u,ek_u}(i, t(m))$
$c^d \leftarrow \mathsf{SKE.E}^{\mathcal{H}}(K_{u,i,t(m)}, m; \tilde{r}(u,i,m))$ $//\mathsf{H}^2_{[-2,4)}$
$c^d \leftarrow g_{u,c^k,|m|}(m)$ $//\mathsf{H}^2_{[4,\infty)}$
$T[u, c^k] \leftarrow K_{u,i,t(m)}$
$M[(u,i,t(m)), c^d] \leftarrow m$ $//\mathsf{H}^2_{[2,\infty)}$
Return $(c^k, c^d)$

$\underline{\textsc{Dec}(u, c)}$
$(c^k, c^d) \leftarrow c$
$(ek_u, dk_u) \leftarrow \mathsf{KEM.K}(r(u))$
$(i, \tau) \leftarrow h^{-1}_{u,ek_u}(c^k)$ $//\mathsf{H}^2_{[1,\infty)}$
If $T[u, c^k] \neq \perp$ $//\mathsf{H}^2_{[-2,1)}$
If $(i, \tau) \in I[u]$ $//\mathsf{H}^2_{[1,3)}$, $\mathsf{H}^2_{[5,\infty)}$
If $(i, \tau) \neq \perp$ $//\mathsf{H}^2_{[3,5)}$
    If $M[(u,i,\tau), c^d] \neq \perp$ $//\mathsf{H}^2_{[2,\infty)}$
        Return $M[(u,i,\tau), c^d]$ $//\mathsf{H}^2_{[2,\infty)}$
    Return $\mathsf{SKE.D}^{\mathcal{H}}(T[u, c^k], c^d)$ $//\mathsf{H}^2_{[-2,1)}$
    Return $\mathsf{SKE.D}^{\mathcal{H}}(K_{u,i,\tau}, c^d)$ $//\mathsf{H}^2_{[1,\infty)}$
$K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$
If $K = \perp$ then return $\perp$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$

$\underline{\textsc{New}(u)}$
$(ek_u, \cdot) \leftarrow \mathsf{KEM.K}(r(u))$
Return $ek_u$
```

```
$\underline{\text{Adversary } \mathcal{B}_{\mathsf{SKE}}^{\textsc{Enc},\textsc{Dec},\mathcal{H}}}$
$r \leftarrow\!\!\$ \ \mathsf{Fcs}(\mathcal{U}, \mathsf{KEM}.\mathcal{R})$
$h^{\pm} \leftarrow\!\!\$ \ \mathsf{Inj}^{\pm}(T, D \times \{0,1\}^{\gamma}, R)$
$t \leftarrow\!\!\$ \ \mathsf{Fcs}(\{0,1\}^*, \{0,1\}^{\gamma})$
$b' \leftarrow\!\!\$ \ \mathcal{A}^{\textsc{SimNew},\textsc{SimEnc},\textsc{SimDec},\mathcal{H}}$
Return $b'$

$\underline{\textsc{SimNew}(u)}$
$(ek_u, \cdot) \leftarrow \mathsf{KEM.K}(r(u))$
Return $ek_u$

$\underline{\textsc{SimEnc}(u, i, m)}$
$(ek_u, \cdot) \leftarrow \mathsf{KEM.K}(r(u))$
$c^k \leftarrow h_{u,ek_u}(i, t(m))$
$c^d \leftarrow \textsc{Enc}((u, i, t(m)), 0, m)$
Return $(c^k, c^d)$

$\underline{\textsc{SimDec}(u, c)}$
$(c^k, c^d) \leftarrow c$
$(ek_u, dk_u) \leftarrow \mathsf{KEM.K}(r(u))$
$(i, \tau) \leftarrow h^{-1}_{u,ek_u}(c^k)$
If $(i, \tau) \neq \perp$
    Return $\textsc{Dec}((u, i, \tau), c^d)$
$K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$
If $K = \perp$ then return $\perp$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$
```

Fig. 25. Hybrids H^1_κ (**Top Left**), adversary $\mathcal{B}_0$ (**Top Right**), hybrids H^2_κ (**Center**), and adversary $\mathcal{B}_{\mathsf{SKE}}$ (**Bottom**) used for proof of Theorem 8. Adversary $\mathcal{B}_{\mathsf{SKE}}$ is used to transition from H^2_3 to H^2_4. Recall that $T = \mathcal{U} \times \mathsf{KEM.Ek}$, $D_{(u,ek)} = \mathcal{I}$, and $R_{(u,ek)} = \mathsf{KEM}.\mathcal{C}(ek)$. Recall that $T' = \mathcal{U} \times \bigcup_{ek \in \mathsf{KEM.Ek}} \mathsf{KEM}.\mathcal{C}(ek) \times \mathbb{N}$, $D'_{(u,c^k,l)} = \{0,1\}^l$, and $R'_{(u,c^k,l)} = \{0,1\}^{\mathsf{SKE.cl}(l)}$. By $\mathbb{I}$ we denote the identity function.

```
Hybrids $\mathsf{H}^3_\kappa$ for $1 \leqslant \kappa \leqslant 6$
$\mathcal{H} \leftarrow\$ \mathsf{KD.IM}$
$K_{(\cdot,\cdot,\cdot)} \leftarrow\$ \mathsf{SKE.K}$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\$ \mathsf{KEM.K}$
$h^{\pm} \leftarrow\$ \mathsf{Inj}^{\pm}(T, D \times \{0,1\}^{\gamma}, R)$ $//\mathsf{H}^3_{[1,5)}$
$h \leftarrow\$ \mathsf{Fcs}(T, D \times \{0,1\}^*, R)$ $//\mathsf{H}^3_{[5,\infty)}$
$t \leftarrow\$ \mathsf{Fcs}(\{0,1\}^*, \{0,1\}^{\gamma})$ $//\mathsf{H}^3_{[1,6)}$
$t \leftarrow \mathbb{I}$ $//\mathsf{H}^3_{[6,\infty)}$
$g \leftarrow\$ \mathsf{Fcs}(T', D', R')$
$b' \leftarrow\$ \mathcal{A}^{\text{New},\text{Enc},\text{Dec},\mathcal{H}}$
Return $(b' = 1)$

Enc$(u, i, m)$
$I[u] \leftarrow I[u] \cup \{(i, t(m)\}$
$c^k \leftarrow h_{u,ek_u}(i, t(m))$
$c^d \leftarrow g_{u,c^k,|m|}(m)$
$T[u, c^k] \leftarrow K_{u,i,t(m)}$
$M[\mathbf{lab}(u, c^k), c^d] \leftarrow m$
Return $(c^k, c^d)$
```

```
Dec$(u, c)$
$(c^k, c^d) \leftarrow c$
$(i, \tau) \leftarrow h^{-1}_{u,ek_u}(c^k)$ $//\mathsf{H}^3_{[1,5)}$
If $(i, \tau) \in I[u]$ and $M[\mathbf{lab}(u, c^k), c^d] \neq \bot$ $//\mathsf{H}^3_{[1,2)}$
If $M[\mathbf{lab}(u, c^k), c^d] \neq \bot$ $//\mathsf{H}^3_{[2,\infty)}$
    Return $M[\mathbf{lab}(u, c^k), c^d]$
If $(i, \tau) \in I[u]$ then $K \leftarrow K_{u,i,\tau}$ $//\mathsf{H}^3_{[1,4)}$
If $T[u, c^k] \neq \bot$ then $K \leftarrow T[u, c^k]$ $//\mathsf{H}^3_{[4,\infty)}$
Else $K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$
If $K = \bot$ then return $\bot$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$

New$(u)$
Return $ek_u$

Defining label function lab
$\mathbf{lab}(u, c^k) = u, h^{-1}_{u,ek_u}(c^k)$ $//\mathsf{H}^3_{[1,3)}$
$\mathbf{lab}(u, c^k) = u, c^k$ $//\mathsf{H}^3_{[3,\infty)}$
```

Fig. 26. Hybrids H^3_κ for proof of Theorem 8. Recall that $T = \mathcal{U} \times \mathsf{KEM.Ek}$, $D_{(u,ek)} = \mathcal{I}$ and $R_{(u,ek)} = \mathsf{KEM}.\mathcal{C}(ek)$.

Hybrids H^1. We start with the H^1 hybrids and corresponding adversary given in Fig. 25. The only difference from the equivalents in Fig. 21 is the use of a random function $\tilde{r}$. In hybrid H^1_0, the table C keeps track of all queries to Enc and returns the same output as before if a query is repeated. The random function $\tilde{r}$ is used to explicitly generate randomness for $\mathsf{SKE.E}$. This serves as a way for $\mathcal{B}_0$ to return consistent results for repeated queries without having to store C. As the inputs to $\tilde{r}$ only repeat on repeated queries, this modification does not change the analysis.

Hybrids H^2. Next we move to the H^2 hybrids and corresponding adversary given in Fig. 25. These contain more significant deviations from their equivalents in Fig. 22. In several places where the equivalent games were able to reference different encryption queries with just the challenge value i (as it was guaranteed to always differ) we instead use either $(u, i, t(m))$ or $(i, t(m))$. We use the former for indexing per-encryption symmetric keys and indexing into table M. We use the latter for storing in $I[u]$ and as input to h.

In H^2_{-2}, we start with h being a random function and t being the identity function. This ensures proper behavior of outputs from Enc such that outputs repeat when encryption queries repeat and seem independent otherwise, giving $\Pr[\mathsf{H}^1_1] = \Pr[\mathsf{H}^2_{-2}]$. In H^2_{-2}, we switch to t being a random function. As $t(m)$ is being used only as input to the random function h and indexing into the tables K and M, this change is only detectable if a collision in t is found. Hence, $\Pr[\mathsf{H}^2_{-2}] \leqslant \Pr[\mathsf{H}^2_{-1}] + q^2_{\text{Enc}}/2^{\gamma+1}$. Finally, in switching to H^2_0 we switch h to being an injection, incurring the same switching lemma term $\Pr[\mathsf{H}^2_{-1}] = \Pr[\mathsf{H}^2_0] + 0.5q^2_{\text{Enc}}/|\mathsf{KEM}.\mathcal{C}|$ from the transition to the equivalent of H^2_0 in Appendix G.

The next transition requiring new analysis is the transition to H^2_3. This is the transition where we switch from checking if $(i, \tau) \in I[u]$ to checking if $(i, \tau) = \bot$ in Dec. This corresponds to the attacker guessing a point in the imagine of h which they have not been shown through Enc. We calculate

$$\frac{|h_{u,ek_u}(\mathcal{I} \times \{0,1\}^{\gamma}) \setminus h_{u,ek_u}(I[u])|}{|\mathsf{KEM}.\mathcal{C}(ek_u) \setminus h_{u,ek_u}(I[u])|} = \frac{|\mathcal{I}| \cdot 2^{\gamma} - |I[u]|}{|\mathsf{KEM}.\mathcal{C}(ek_u)| - |I[u]|} \leqslant \frac{|\mathcal{I}| \cdot 2^{\gamma}}{|\mathsf{KEM}.\mathcal{C}|}.$$

$\Pr[\mathsf{H}^2_2] \leqslant \Pr[\mathsf{H}^2_3] + q_{\text{Dec}}|\mathcal{I}| \cdot 2^{\gamma}/|\mathsf{KEM}.\mathcal{C}|$. We incur the same loss between H^2_4 and H^2_5.

In H^2_4, rather than picking c^d uniformly at random for each query, we pick it as $g_{u,c^k,|m|}(m)$ where g is a random function. Note that c^k is chosen such that when u is fixed, c^k uniquely determines the challenge value i. Hence c^d repeats when (u, i, m) does, but is uniform and independent for otherwise, as desired.

```
Hybrids $\mathsf{H}^4_\kappa$ for $0 \leqslant \kappa \leqslant 6$
$\mathcal{H} \leftarrow\!\!\$\ \mathsf{KD.IM}$
$\tilde{r} \leftarrow\!\!\$\ \mathsf{Fcs}(\mathcal{U} \times \mathcal{I} \times \{0,1\}^*, \mathsf{KEM}.\mathcal{R})$
$K_{(\cdot,\cdot,\cdot)} \leftarrow\!\!\$\ \mathsf{SKE.K}$
$(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\!\!\$\ \mathsf{KEM.K}$
$h \leftarrow\!\!\$\ \mathsf{Fcs}(T, D \times \{0,1\}^*, R)$
$g^\pm \leftarrow\!\!\$\ \mathsf{Inj}^\pm(T', D', R')$ $//\mathsf{H}^4_{[0,6)}$
$g \leftarrow\!\!\$\ \mathsf{Fcs}(T', D', R')$ $//\mathsf{H}^4_{[6,\infty)}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{New},\textsc{Enc},\textsc{Dec},\mathcal{H}}$
Return $(b' = 1)$

$\textsc{Enc}(u, i, m)$
$c^k \leftarrow h_{u,ek_u}(i, m)$ $//\mathsf{H}^4_{[0,2)}, \mathsf{H}^4_{[4,\infty)}$
$K \leftarrow K_{u,i,m}$ $//\mathsf{H}^4_{[0,2)}$
$(c^k, K) \leftarrow \mathsf{KEM.E}^{\mathcal{H}}(ek_u; \tilde{r}(u, i, m))$ $//\mathsf{H}^4_{[2,4)}$
$c^d \leftarrow g_{u,c^k,|m|}(m)$
$T[u, c^k] \leftarrow K$ $//\mathsf{H}^4_{[0,3)}$
$M[u, c^k, c^d] \leftarrow m$
Return $(c^k, c^d)$
```

```
$\textsc{Dec}(u, c)$
$(c^k, c^d) \leftarrow c$
If $M[u, c^k, c^d] \neq \perp$ $//\mathsf{H}^4_{[0,1)}, \mathsf{H}^4_{[5,\infty)}$
    Return $M[u, c^k, c^d]$ $//\mathsf{H}^4_{[0,1)}, \mathsf{H}^4_{[5,\infty)}$
$m \leftarrow g^{-1}_{u,c^k,\mathsf{SKE.ml}(|c^d|)}(c^d)$ $//\mathsf{H}^4_{[1,5)}$
If $m \neq \perp$ then return $m$ $//\mathsf{H}^4_{[1,5)}$
If $T[u, c^k] \neq \perp$ then $K \leftarrow T[u, c^k]$ $//\mathsf{H}^4_{[0,3)}$
Else $K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$ $//\mathsf{H}^4_{[0,3)}$
$K \leftarrow \mathsf{KEM.D}^{\mathcal{H}}(dk_u, c^k)$ $//\mathsf{H}^4_{[3,\infty)}$
If $K = \perp$ then return $\perp$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$

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

```
Adversary $\mathcal{B}_1^{\textsc{New},\textsc{Encap},\textsc{Decap},\mathcal{H}}$
$g^\pm \leftarrow\!\!\$\ \mathsf{Inj}^\pm(T', D', R')$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{New},\textsc{SimEnc},\textsc{SimDec},\mathcal{H}}$
Return $1 - b'$

$\textsc{SimEnc}(u, i, m)$
$(c^k, \cdot) \leftarrow \textsc{Encap}(u, i)$
$c^d \leftarrow g_{u,c^k,|m|}(m)$
Return $(c^k, c^d)$

$\textsc{SimDec}(u, c)$
$(c^k, c^d) \leftarrow c$
$m \leftarrow g^{-1}_{u,c^k,\mathsf{SKE.ml}(|c^d|)}(c^d)$
If $m \neq \perp$ then return $m$
$K \leftarrow \textsc{Decap}(u, c^k)$
If $K = \perp$ then return $\perp$
Return $\mathsf{SKE.D}^{\mathcal{H}}(K, c^d)$
```

Fig. 27. Hybrids H^4_κ (**Left**) and adversary $\mathcal{B}_1$ (**Right**) used for proof of Theorem 8. Adversary $\mathcal{B}_1$ is used to transition from H^4_1 to H^4_2. Recall that $T' = \mathcal{U} \times \bigcup_{ek \in \mathsf{KEM.Ek}} \mathsf{KEM}.\mathcal{C}(ek) \times \mathbb{N}$, $D'_{(u,c^k,l)} = \{0,1\}^l$, and $R'_{(u,c^k,l)} = \{0,1\}^{\mathsf{SKE.cl}(l)}$.

Hybrids H^3. Next we move to the H^3 hybrids given in Fig. 26. Their deviation from their equivalents in Fig. 23 are generally inherited from the earlier games.

We can skip the hybrid H^3_0 as we have already been using u when indexing into M. At the end of the H^3 hybrids we switch h to being a random function (consistent with its equivalent in Appendix G switching to picking the output of h uniformly). In H^3_5, we switch h to being a random function where the equivalent game got rid of h entirely. We require using h still to maintain the proper consistency between queries. In the extra hybrid H^3_6, we switch t from being a random function to the identity function. This incurs the loss of $\Pr[\mathsf{H}^3_5] \leqslant \Pr[\mathsf{H}^3_6] + q^2_{\text{Enc}}/2^{\gamma+1}$.

Hybrids H^4. Finally we consider the H^3 hybrids and corresponding adversary given in Fig. 27. Their deviation from their equivalents in Fig. 24 are generally straightforward. The ciphertext c^k is picked via random function h when the equivalent game sampled it at random. A random function $\tilde{r}$ is introduced to gives random coins to KEM that are consistent for repeated queries. The injection g is replaced with a random function rather than being replaced with random sampling. These modifications ensure the behavior of repeating when encryption queries do, but looking independent otherwise. We again have $\Pr[\mathsf{H}^4_6] = \Pr[\mathsf{G}^{\text{mu-\$cca}}_{\mathsf{KD},0}(\mathcal{A})]$. □