Hiding in Plain Sight: Memory-tight Proofs via Randomness Programming*

Ashrujit Ghoshal[1], Riddhi Ghosal[2], Joseph Jaeger[3], and Stefano Tessaro[1]

[1] Paul G. Allen School of Computer Science & Engineering
University of Washington, Seattle, Washington, US
ashrujit@cs.washington.edu,tessaro@cs.washington.edu
[2] University of California, Los Angeles, US
riddhi@cs.ucla.edu
[3] Georgia Institute of Technology, Atlanta, Georgia, US
josephjaeger@gatech.edu

Abstract. This paper continues the study of *memory-tight reductions* (Auerbach et al, CRYPTO '17). These are reductions that only incur minimal memory costs over those of the original adversary, allowing precise security statements for memory-bounded adversaries (under appropriate assumptions expressed in terms of adversary time and memory usage). Despite its importance, only a few techniques to achieve memory-tightness are known and impossibility results in prior works show that even basic, textbook reductions cannot be made memory-tight.

This paper introduces a new class of memory-tight reductions which leverage random strings in the interaction with the adversary to hide state information, thus shifting the memory costs to the adversary. We exhibit this technique with several examples. We give memory-tight proofs for digital signatures allowing many forgery attempts when considering randomized message distributions or probabilistic RSA-FDH signatures specifically. We prove security of the authenticated encryption scheme Encrypt-then-PRF with a memory-tight reduction to the underlying encryption scheme. By considering specific schemes or restricted definitions we avoid generic impossibility results of Auerbach et al. (CRYPTO '17) and Ghoshal et al. (CRYPTO '20).

As a further case study, we consider the textbook equivalence of CCA-security for public-key encryption for one or multiple encryption queries. We show two qualitatively different memory-tight versions of this result, depending on the considered notion of CCA security.

Keywords: Provable security, memory-tightness, time-memory trade-offs

1 Introduction

The aim of concrete security proofs is to lower bound, as precisely as possible, the resources needed to break a cryptographic scheme of interest, under some plausible assumptions. The traditional resource used in provable security is *time complexity* (as well as related metrics, like data complexity). Recent works [1,27,26,21,14,13,8,18,17,11,25,12] have focused on additionally taking the *memory* costs of the adversary into account. This is important, as the amount of available memory can seriously impact the feasibility of an attack.

This paper presents new techniques for *memory-tight reductions*, a notion introduced by Auerbach et al. [1] to relate the assumed time-memory hardness of an underlying computational problem to the security of a scheme. More precisely, the end goal is to prove, via a reduction, that any adversary running in time t and with s bits of memory can achieve at most advantage $\epsilon = \epsilon(t, s)$ in compromising a scheme, by assuming that some underlying computational problem can only be solved with advantage $\delta = \delta(t', s')$ by algorithms running in time t' and with memory s'. A memory-tight reduction guarantees that $s \approx s'$, and usually, we want this to be tight also according to other parameters, i.e., $t \approx t'$ and $\epsilon \approx \delta$.

* A preliminary version of this paper appears in the proceedings of EUROCRYPT 2022. This is the full version.

Memory-tight reductions are of value whenever the underlying problem is (conjectured to) be *memory sensitive*, i.e., the time needed to solve it grows as the amount of memory available to the adversary is reduced. Examples of memory-sensitive problems include classical ones in the public-key setting, such as breaking RSA and factoring, lattice problems and LPN, solving discrete logarithms over finite fields,[4] as well as problems in the secret-key setting, such as finding k-way collisions (for $k > 2$), finding several collisions at once [13], and distinguishing random permutations from random functions [21,14,25].

Developing memory-tight reductions is not always easy, and can be (provably) impossible [1,27,18,17]. This makes it fundamental to develop as many techniques as possible to obtain such reductions. In this paper, we identify a class of examples which admit a new kind of memory-tight reductions. Our approach relies on the availability of random strings exchanged between the adversary and the security game, and which the reduction can leverage to encode state which can be recovered from later queries of the adversary, without the need to store this information locally, and thus saving memory. (In particular, the burden of keeping this information remains on the adversary, which needs to reproduce this random string for this state information to be relevant.) We present these techniques abstractly in the next section, with the help of a motivating example, and then move on to an overview of our specific results.

1.1 Our Techniques - An Overview

As a motivating example, consider the standard UFCMA security notion for signatures. It is defined via a game where the attacker, given the verification key vk, obtains signatures for chosen messages $m_1, m_2, \ldots$, after which it outputs a candidate message-signature pair (m^*, σ^*), and wins if m^* was not signed before, and σ^* is valid for m^*. When ignoring memory, this notion is *tightly* equivalent to one (which we refer to as mUFCMA) that allows for an arbitrary number of "forgery attempts" for pairs (m^*, σ^*), and the adversary wins if one of them succeeds in the above sense. This is convenient: we generally target mUFCMA, but only need to deal with proving the simpler UFCMA notion.

The classical reduction transforms any mUFCMA adversary into a roughly equally efficient UFCMA adversary, which wins with the same probability, by (1) simulating forgery queries using the verification key, and (2) outputting the first forgery query (m^*, σ^*) which validates and such that m^* is fresh. This reduction is however *not* memory-tight, as we need to ensure the freshness of m^*, which requires remembering the previously signed messages. ACFK [1] prove that this is in some sense necessary, by showing that a (restricted) class of reductions cannot be memory-tight via a reduction to streaming lower bounds.

Our idea: Efficient tagging. To illustrate our new technique, which we refer to as *efficient tagging*, imagine now that we only use the signature scheme to sign *random* messages $m_1, m_2, \ldots, m_q \leftarrow\$ \{0,1\}^\ell$, and consider a corresponding variant of mUFCMA security, which we want to reduce to (plain) UFCMA security. This, intuitively, does not seem to help resolve the above issue, because random messages are hardest to compress.

However, what is important here is that the reduction is responsible for simulating the random messages, and can simulate them in special ways, and *program* them so that they encode state information. For instance, assume that the reduction has access to an *injective* random function $f : [q] \to \{0,1\}^\ell$, with inverse f^{-1}, which can be simulated *succinctly* from a short key as a pseudorandom object. Then, the reduction to UFCMA can set $m_i \leftarrow f(i)$ for the i-th query, and upon simulating a forgery query for (m^*, σ^*), the reduction checks whether $f^{-1}(m^*) \in [q]$ to learn whether m^* is a fresh signing query or not.

Of course, the simulation is not perfect: The original m_i's are not necessarily distinct (this can be handled via the classical "switching lemma"). Also, the reduction could miss a valid forgery if the adversary outputs m_i *before* it is given to the adversary, but this again only occurs with small probability.

Inefficient tagging and non-time-tight reductions. In the above example, we can efficiently check that $f^{-1}(m^*) \in [q]$. However, in some cases we may not – again, consider an example where the messages to be signed are sampled as $m_i \leftarrow h(r_i)$, where h is a hard-to-invert function and r_i is random. Then, we

[4] However, the discrete logarithm problem in elliptic-curve groups, or any other group in which the best-known attacks are generic, is not memory sensitive, since optimal memory-less attacks are known.

could adapt our proof above by setting $m_i \leftarrow h(f(i))$, but now, to detect a prior signing query, we would have to check whether $m^* = h(f(i))$ for some $i \in [q]$, and this can only be done in linear time. The resulting UFCMA adversary runs in time $t' = t + \Theta(q_F \cdot q)$, where t is the running time of the original adversary and q_F is the number of forgery attempts. For example, if $q \approx q_F \approx t$, the reduction is not time tight, and the adversary runs in time $t' = O(t^2)$.

Are non-time-tight reductions useless? It turns out that such non-time-tight reductions can still be helpful to infer that breaking a scheme requires memory, although this ultimately depends on the concrete security of the problem targeted by the reduction. Say, for example, a reduction for a given scheme transforms a successful adversary running in time t and using memory s into an adversary running in time t^2 and using memory s breaking discrete logarithms over $\mathbb{F}_p$, for a 4096-bit prime p. It turns out that if we have fewer than 2^{78} bits of memory, no known discrete logarithm algorithm is better than a generic one (i.e. runs in time better than 2^{2048}), which means that our non-time-tight reduction is still sufficient to infer security for any $s < 2^{78}$ as long as $t < 2^{1024}$.

Message encoding. At the highest level, what happens is that the reduction is in control of certain random values which we can exploit to hide state information which can later be uniquely recovered, since triggering a situation where the reduction needs to remember requires the adversary to actually give back to the reduction this value. In the above, this state information is simple, namely whether the query is old or not. But as we will show below, the paradigm can be used to store complex information – we refer to this technique as *message encoding*, and discuss an example below.

A New Viewpoint: $\mathcal{F}$-oracle adversaries. In our technique described above we needed access to a large random injection, which we argue can be simulated pseudorandomly. Prior works have similarly used PRFs to pseudorandomly simulate random oracles [1,7] with low memory. The fact that one needs to decide how to simulate such objects when stating a memory-tight reduction is rather inconvenient: different instantiations seemingly lead to quantitatively different reductions, although this fact does not appear to be a reflection of any particular reality. In this paper, we propose (and advocate for) what we believe to be the "right" viewpoint: Our reductions are stated in terms of $\mathcal{F}$-oracle adversaries where $\mathcal{F}$ is a set of functions and such an adversary expects oracle access to a random $f \in \mathcal{F}$. Then, a memory-tightness theorem is obtained in one of two ways, by either (1) applying a generic lemma stating that f can be instantiated in low memory using an $\mathcal{F}$-pseudorandom function, or (2) assuming that the use of f does not functionally increase the success chances of the adversary because f is independent of the problem instance being solved (this is provably the case for some information theoretic problems). In particular, (1) is more conservative than (2), but it is very likely that (2) is also a viable approach which leads to cleaner result – indeed, we do not expect any of the considered memory-sensitive problems to become easier given access to an oracle from any natural class $\mathcal{F}$ – e.g., Factoring does not become easier given access to a random injection.

1.2 Our Results

We now move to an overview of our results (summarized in Fig. 1) which exemplify different applications of the tagging and message-encoding techniques.

Multi-challenge Security of Digital Signatures. Our first results consider the security of digital signatures in the face of multiple forgery attempts (i.e., challenge queries), generalizing the examples discussed above. We work with a notion we refer to as UFRMA (unforgeability under randomized message attack). This notion is parameterized by a *message distribution* D and when the attacker makes a signing query for m it receives a signature of $m' = \mathsf{D}(m; r)$ for a random r. If m and r can be extracted from m', giving the notion xUFRMA (or mxUFRMA for many forgery attempts), we can generalize our efficient tagging approach above by having the reduction to UFCMA choose $r = f(m, i)$ where each $f(m, \cdot)$ is a random injection. This setting can capture, e.g., the signatures used in key exchange protocols like TLS 1.3 where the server signs a transcript which includes a random 256-bit nonce. A version of our inefficient tagging example works when only m can be extracted from m' (wUFRMA); we pick $r = f(m, i)$ and in verification of a forgery query perform the linear time check of whether $m^* = \mathsf{D}(m; f(m, i))$ for some $i \in [q]$. This setting captures places where the message to be signed includes a fresh public key or ciphertext. This includes, for example, the use of signatures for signing certificates, in some key exchange protocols, and in signcryption.

We further prove mUFCMA security for particular schemes. First, we can randomize any digital signature scheme DS (obtaining a scheme we call RDS) by signing $m \| r$ for random r chosen by the signing algorithm and including r as part of the signature. An immediate implication of our mxUFCRA result is a tight reduction from the mUFCMA security of RDS to the UFCMA security of the underlying scheme. One particular instantiation of RDS is Probabilistic Full Domain Hash with RSA (RSA-PFDH) which was introduced by Coron [10] to provide a variant of Full Domain Hash [5] with an (advantage-) tighter security proof. Using our efficient tagging technique we obtain a fully tight proof of the *strong* mUFCMA security of RSA-PFDH from the RSA assumption.

In independent and concurrent work, Diemert, Gellert, Jager, and Lyu [12] studied the mUFCMA security of digital signature schemes. They also considered the RDS construction, proving that if DS can be proven strong UFCMA1 secure[5] with a restricted class of "canonical" memory-tight reductions then there is a memory-tight reduction for the strong mUFCMA security of RDS. This complements our result, showing memory-tight strong mUFCMA security of RDS based on a restricted class of schemes while our result proves memory-tight plain mUFCMA security based on any plain UFCMA scheme. They apply their RDS result to establish tight proofs for the strong mUFCMA security of RSA-PFDH (matching our direct proof in Theorem 4). as well as schemes based on lossy identification schemes and pairings.

Authenticated Encryption Security. Ghoshal, Jaeger, and Tessaro [17] have recently observed that in the context of authenticated encryption (AE), it is difficult to lift confidentiality of the scheme, in terms of INDR security, to full AE security, when additionally assuming ciphertext integrity, if we want to do so in a memory-tight way. This is well motivated, as several works establish tight time-memory trade-offs for INDR security [26,21,14,11,25], which we would like to lift to their AE security. The difficulty in the proof is that the INDR reduction must simulate a decryption oracle which rejects all ciphertexts *except those forwarded from an encryption query*. Recognizing these forwarded ciphertexts seems to require remembering state.

Here, we give a different take and show that for specific schemes – in particular, those obtained by adding integrity via a PRF, following the lines of [24,4,23] – a memory-tight reduction *can* be given. Our INDR reduction is applied after arguing that the PRF looks like a random function f and thus forgeries are unlikely to occur. It uses f in a version of our efficient tagging technique to identify whether a ciphertext queried to decryption is fresh.[6]

Chosen ciphertext security: One to many. A classical textbook result for public-key encryption shows that CCA-security against a *single* encryption query (1CCA) implies security against *multiple* queries (mCCA), with a quantitative advantage loss accounting to the number of such queries. ACFK [1] claim, incorrectly, that the associated reduction from 1CCA to mCCA is easy to make memory-tight, but this appears to be an oversight: No such reduction is known, and here we use our techniques to recover a memory-tight version of this result.

Let us consider concretely the "left-or-right" formulation of 1CCA/mCCA-security: The reduction from 1CCA to mCCA, given an adversary $\mathcal{A}$, picks a random $i \leftarrow\$ [q]$ (where q is the number of encryption queries) and simulates the multi-query challenger to $\mathcal{A}$ by answering its first $i-1$ encryption queries with an encryption of the left message, whereas the last $q-i$ queries are answered by encrypting the right message. Only the answer to the i-th query is answered by the single-query challenger. A problem arises when simulating the decryption queries: Indeed, we need to guarantee that a decryption query for *any* of the challenge ciphertexts $c_1^*, \ldots, c_q^*$ returns an error $\perp$, yet this suggests that we seemingly need to remember the extra challenge ciphertexts c_j^* for $j \neq i$.

We will resolve this in two ways. First, we give a new memory-tight reduction using the inefficient tagging method, with the same advantage loss as the original textbook reduction. Our reduction is non-time-tight,

5 The suffix '1' indicates a variant of UFCMA security in which the adversary can only obtain a single signature per message. The security game always returning the same signature if the adversary repeats signature queries.

6 Ghoshal et al. [17] in fact described three variants of AE with different conventions for how decryption responds to non-fresh queries. By our results, memory-tight reductions to INDR are possible for two of the three variants.

Assumption	Scheme	Result	Time	Memory	Advantage	New Technique
1UFCMA	Any	mxUFRMA	✓	✓	✓	Efficient tagging
	RDS	mUFCMA	✓	✓	✓	Efficient tagging
	Any	mwUFRMA	✗	✓	✓	Inefficient tagging
RSA	RSA-PFDH	mSUFCMA	✓	✓	✓	Efficient tagging
(PRF, INDR)	EtP	AE	(✓,✓)	(✗,✓)	(✓,✓)	Efficient tagging
1CCA	Any	mCCA	✗	✓	✗	Inefficient tagging
1$CCA	Any	m$CCA	✓	✓	✗	Message encoding

Fig. 1. Memory-tight reductions we provide. A 1 vs. an m prefix indicates whether one or many challenge queries are allowed. A ✓ vs. an ✗ indicates whether the reduction is tight with respect to that complexity metric. Reductions lacking tightness multiply running time/advantage by $O(q)$ or add $O(q)$ to the memory complexity, where q is the number of queries. An x vs. a w indicates whether the coins underlying the distribution of messages can be extracted from the message. RDS is randomization of any digital signature scheme by padding input messages with randomness. RSA-PFDH is probabilistic full-domain hash with RSA. EtP is the Encrypt-then-PRF AE construction.

---

so may not be suitable for all situations. The main idea here is that we use the *randomness* used to generate the challenge ciphertext as our tag.

To obtain a reduction which is also tight with respect to time, we resort to the observation that changing to a stronger (but still commonly achieved) definition of CCA-security allows for different memory-tight reductions. We give in particular a memory-tight *and* time-tight reduction (with the usual factor q advantage loss) from the notion of 1$CCA-`m` security to the notion of m$CCA-`m` security. These are variants of CCA security where (1) encryption queries are with respect to a *single* message, and return either the encryption of the message, or a random, independent ciphertext, and (2) decryption queries on a challenge ciphertext c_i^* returns the associated message.

Our reduction uses the full power of our message encoding approach, simulating random ciphertexts in a careful way which allows for recovering the associated challenge plaintext.

A FEW REMARKS. The above results on CCA security show us that the ability to give a memory-tight reduction is strongly coupled with definitional choices. In particular, different equivalent approaches to modeling the decryption oracle in the memory unbounded regime may not be equivalent in the memory-bounded setting. This means in particular that we need to exercise more care in choosing the right definition. We believe, for example, that the approach taken in m$CCA-`m` security is the more "natural" one (as it does not require artificially blocking the output of the decryption oracle, by always returning a message), but there may be contexts where other definitional choices are favored.

Another important lesson learnt from our AE result is that impossibility results, such as those in [1,27,18,17], do not preclude positive results in form of memory-tight reductions, either by leveraging the structure of specific schemes, or by considering restricted security notions.

1.3 Paper Outline

Section 2 introduces notation, our computational model, and basic cryptographic background. Section 3 discusses our convention of using $\mathcal{F}$-oracle adversaries. Section 4 gives our memory-tight reduction for digital signature schemes when many forgery attempts are allowed. In particular, the generic results are in Section 4.2, while the result specific to RSA-PFDH is in Section 4.5. Section 5 proves the security of Encrypt-then-PRF with a memory-tight reduction to the INDR security of the encryption scheme. Section 6 gives our results relating the one- and many-challenge query variants of CCA security. In particular, Section 6.1 gives our result for the traditional "left-vs.-right" notion and Section 6.2 gives our result for the "indistinguishable from random" variant.

2 Preliminaries

Let $\mathbb{N} = \{0, 1, \ldots\}$ and $[n] = \{1, \ldots, n\}$ for $n \in \mathbb{N}$. If $x \in \{0,1\}^*$ is a string, then $|x|$ denotes its length in bits. If S is a set, then $|S|$ denotes its size. We let $x \,\|\, y \,\|\, \ldots$ denote an encoding of the strings $x, y, \ldots$ from which the constituent strings can be unambiguously recovered. We identify bitstrings with integers in the standard way.

Functions. Let T be a set (called the tweak set) and for each $t \in T$ let D_t and R_t be sets. Then $\mathsf{Fcs}(T, D, R)$ denotes the set of all f such that for each $t \in T$, $f(t, \cdot)$ is a function from D_t to R_t. Similarly, $\mathsf{Inj}(T, D, R)$ denotes the set of all f such that for each $t \in T$, $f(t, \cdot)$ is an injection from D_t to R_t. When D_t or R_t are independent of the choice of t we may omit the subscript.

If $f \in \mathsf{Inj}(T, D, R)$, then its inverse f^{-1} is defined by $f^{-1}(t, f(t, x)) = x$ for all (t, x) and $f^{-1}(t, y) = \bot$ for $y \notin f(t, D_t)$. For such f we let $f^{\pm}$ denote the function defined by $f^{\pm}(+, x) = f(x)$ and $f^{\pm}(-, x) = f^{-1}(x)$. We let $\mathsf{Inj}^{\pm}(T, D, R) = \{f^{\pm} : f \in \mathsf{Inj}(T, D, R)\}$.

2.1 Computational Model

Pseudocode. We regularly use pseudocode inspired by the code-based framework of [6]. We think of algorithms as randomized RAMs when not specified otherwise. If $\mathcal{A}$ is an algorithm, then $y \leftarrow \mathcal{A}^{\mathrm{O}_1, \cdots}(x_1, \ldots; r)$ denotes running $\mathcal{A}$ on inputs $x_1, \ldots$ with coins r and access to the oracles $\mathrm{O}_1, \ldots$ to produce output y. When the coins are implicit we write $\leftarrow\!\!\$$ in place of $\leftarrow$ and omit r.

We let $x \leftarrow\!\!\$\ \mathcal{D}$ denote sampling x according to the distribution $\mathcal{D}$. If $\mathcal{D}$ is a set, we overload notation and let $\mathcal{D}$ also denote the uniform distribution over elements of $\mathcal{D}$. The domain of $\mathcal{D}$ is denoted by $[\mathcal{D}]$.

Security notions are defined via games; for an example see Fig. 2. The probability that G outputs true is denoted $\Pr[\mathsf{G}]$. In proofs we sometimes define a sequence of "hybrid" games in one figure, using comments of the form "$//\mathsf{H}_{[i,j)}$." A line of code commented thusly is only included in the hybrids H_k for $i \leqslant k < j$. (We are of course referring only to values of $k \in \mathbb{N}$.) By this convention to identify the differences between H_{k-1} and H_k one looks for comments $\mathsf{H}_{[i,k)}$ (code no longer included in the k-th hybrid) and $\mathsf{H}_{[k,j)}$ (code new to the k-th hybrid).

We let $\bot$ be a special symbol used to indicate rejection. If we do not explicitly include $\bot$ in a set, then $\bot$ is not contained in that set. If $\bot$ is an input to a function or algorithm, then we assume its output is $\bot$. We do not distinguish between $\bot$ and tuples $(\bot, \ldots, \bot)$. Algorithms cannot query $\bot$ to their oracles.

Complexity measures. To measure the complexity of algorithms we follow the conventions of measuring their local complexity, not including the complexity of whatever oracles they interact with. Local complexity was preferred by Auerbach et al. [1] for analyzing memory-limited adversaries so that analysis can be agnostic to minor details of security definitions' implementations. We focus on worst-case runtime $\mathbf{Time}(\mathcal{A})$ and memory complexity $\mathbf{Mem}(\mathcal{A})$ (i.e. how many bits of state it stores for local computation). These exclude the internal complexity of oracles queried by $\mathcal{A}$, but include the time and memory used to write the query and receive the response. If $\mathcal{A}$ expects access to n oracles then we let $\mathbf{Query}(\mathcal{A}) = (q_1, \ldots, q_n)$ where q_i is an upper bound on the number of queries to its i-th oracle. (Here we index from left to right, so for $\mathcal{A}^{\mathrm{O}_1, \ldots, \mathrm{O}_n}$ the i-th oracle is O_i.) If S is a scheme, then $\mathbf{Time}(\mathsf{S})$ and $\mathbf{Mem}(\mathsf{S})$ are the sums of the corresponding complexities over all of its algorithms. If G is a game, then we define $\mathbf{Time}(\mathsf{G})$ and $\mathbf{Mem}(\mathsf{G})$ to *exclude* the complexity of any adversaries embedded in the game.

2.2 Cryptographic Background

Ideal models. Some schemes we look at may be proven secure in ideal models (e.g. the random oracle or ideal cipher models). To capture this we can think of a scheme S as specifying a set of functions $\mathsf{S.I}$. At the beginning of a security game a function h will be sampled from this set. The adversary and all algorithms of S are given oracle access to h.

FUNCTION FAMILIES. A family of functions F specifies, for each $K \in \mathsf{F.K}$, an efficiently computable function $\mathsf{F}_K \in \mathsf{F.F}$. We refer to $\mathsf{F.F}$ as the function space of F. Pseudorandom (PR) security of F is captured by the game defined in Fig. 2. It measures how F with a random key can be distinguished from a random function in $\mathsf{F.F}$ via oracle access. We define $\mathsf{Adv}^{\mathsf{pr}}_{\mathsf{F}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{pr}}_{\mathsf{F},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\mathsf{pr}}_{\mathsf{F},0}(\mathcal{A})]$. The standard notions of (tweakable) pseudorandom functions/injections/permutations or strong injections/permutations are captured by appropriate choices of $\mathsf{F.F}$.

Game $\mathsf{G}^{\mathsf{pr}}_{\mathsf{F},b}(\mathcal{A})$	$\text{Ev}(x)$
$\mathrm{h} \leftarrow\!\$\; \mathsf{F.I}$	$y_1 \leftarrow \mathsf{F}^{\mathrm{h}}_K(x)$
$K \leftarrow\!\$\; \mathsf{F.K}$	$y_0 \leftarrow f(x)$
$f \leftarrow\!\$\; \mathsf{F.F}$	Return y_b
$b' \leftarrow\!\$\; \mathcal{A}^{\text{Ev},\mathrm{h}}$	
Return $b' = 1$	

Fig. 2. Security game capturing the pseudorandomness of function family F.

SWITCHING LEMMA. We make use of the following standard result which bounds how well a random function and a random injection can be distinguished.

Lemma 1 (Switching Lemma). *Fix T, D, and R. Let $N = \min_{t \in T} |R_t|$. Then for any adversary $\mathcal{A}$ with $q = \mathbf{Query}(\mathcal{A})$ we have that*

$$|\Pr[\mathcal{A}^f \Rightarrow 1] - \Pr[\mathcal{A}^g \Rightarrow 1]| \leqslant 0 \cdot q^2/N.$$

The probabilities are measured over the coins of $\mathcal{A}$, the uniform choice of f from $\mathsf{Fcs}(T, D, R)$, and the uniform choice of g from $\mathsf{Inj}(T, D, R)$.

Recent papers [21,13,25] have given improved versions of the switching lemma for adversaries with bounded memory complexity, as long as it does not repeat oracle queries. In our application of the switching lemma the adversary's memory complexity is too large for these bounds to provide any improvement.

OTHER PRIMITIVES. We recall relevant syntax and security definitions for digital signatures, nonce-based encryption, and public key encryption schemes in the sections where we consider them (Sections 4, 5, and 6 respectively).

3 Adversaries With Access to Random Functions

This paper proposes and adopts what we consider to be a better formalism to deal with memory-tight reductions. Namely, all of our reductions will require access to some variety of large random functions which it will query on a small number of inputs (specifically uniformly random functions and invertible random injections). That is, our reduction adversaries can be written in the form shown of the left below, for some set of functions $\mathcal{F}$ and algorithm $\mathcal{A}_2$. (On the right is a pseudorandom version of $\mathcal{A}$ which we will discuss momentarily.)

Adversary $\mathcal{A}^{\mathrm{O}}(in)$	Adversary $\mathcal{A}^{\mathrm{O}}_{\mathsf{F}}(in)$
$f \leftarrow\!\$\; \mathcal{F}$	$K \leftarrow\!\$\; \mathsf{F.K}$
$out \leftarrow\!\$\; \mathcal{A}_2^{\mathrm{O},f}(in)$	$out \leftarrow\!\$\; \mathcal{A}_2^{\mathrm{O},\mathsf{F}_K}(in)$
Return out	Return out

We refer to such an $\mathcal{A}$ as an $\mathcal{F}$-oracle adversary. In this section we will generally discuss such adversaries, rather than separately providing the discussion for such adversaries each time we apply them.

The time and memory complexity of any $\mathcal{F}$-oracle adversary must include the complexity of sampling, storing, and evaluating f. This will be significant if $\mathcal{F}$ is large. However, as we will argue, this additional state and time should be assumed to not significantly increase the advantage of $\mathcal{A}$. As such, we will define the *reduced complexity* of $\mathcal{A}$ by

$$\mathbf{Time}^*(\mathcal{A}) = \mathbf{Time}(\mathcal{A}_2) \text{ and } \mathbf{Mem}^*(\mathcal{A}) = \mathbf{Mem}(\mathcal{A}_2).$$

Later we state theorems in terms of reduced complexity.

Pseudorandom replacement. The most conservative justification of $\mathcal{F}$-oracle adversaries is to bound how much the oracle can help by replacing it with a pseudorandom version. This was the approach taken by Auerbach et al. [1] when they used pseudorandom functions for purposes such as emulating random oracles and storing the coins required by an adversary with low memory, and has been adopted by follow-up work [27,8,12]. If F is a function family with $\mathsf{F.F} = \mathcal{F}$, then the adversary $\mathcal{A}_{\mathsf{F}}$ we gave above does exactly this. It replaces $\mathcal{A}_2$'s oracle access to f with access to F_K for a random K. The following lemma is straightforward.

Lemma 2. *Let $\mathcal{A}$ be an $\mathcal{F}$-oracle adversary for a game G. Then for any function family F with $\mathsf{F.F} = \mathcal{F}$ we can define a pseudorandomness adversary $\mathcal{A}_{\mathsf{i}}$ such that*

$$\Pr[\mathsf{G}(\mathcal{A})] \leqslant \Pr[\mathsf{G}(\mathcal{A}_{\mathsf{F}})] + \mathsf{Adv}^{\mathsf{pr}}_{\mathsf{F}}(\mathcal{A}_{\mathsf{i}}), \quad \mathbf{Time}(\mathcal{A}_{\mathsf{i}}) = \mathbf{Time}^*(\mathcal{A}) + \mathbf{Time}(\mathsf{G}(\mathcal{A})),$$
$$\mathbf{Query}(\mathcal{A}_{\mathsf{i}}) = q, \text{ and} \quad \mathbf{Mem}(\mathcal{A}_{\mathsf{i}}) = \mathbf{Mem}^*(\mathcal{A}) + \mathbf{Mem}(\mathsf{G}(\mathcal{A})).$$

Here q is an upper bound on the number of queries $\mathcal{A}_2$ makes to its second oracle.

Note that the complexity of $\mathcal{A}_{\mathsf{F}}$ is given by $\mathbf{Time}(\mathcal{A}_{\mathsf{F}}) = \mathbf{Time}^*(\mathcal{A}) + q \cdot \mathbf{Time}(\mathsf{F})$ and $\mathbf{Mem}(\mathcal{A}_{\mathsf{F}}) = \mathbf{Mem}^*(\mathcal{A}) + \mathbf{Mem}(\mathsf{F})$. Thus the existence of an appropriate pseudorandom F ensures that the memory and time complexity excluded by $\mathbf{Time}^*$ and $\mathbf{Mem}^*$ cannot significantly aid an adversary. In the use of this technique by Auerbach et al. [1] the reduction $\mathcal{A}_{\mathsf{i}}$ was memory-tight. Note this is not strictly necessary as long as we are willing to assume the existence of F with sufficient security as a function of attackers' time and query complexities without regard to memory complexity.

We could have combined Lemma 2 with any of our coming theorems to obtain bounds in terms of **Time** and **Mem**, rather than their reduced version. However we find the use of reduced complexity cleaner as it simplifies our theorems, allowing us to focus on the conceptual core of the proofs without having to repeat the rote step of replacing random objects with pseudorandom ones.

When combining the lemma with a theorem, game G would correspond to the security game played by the reduction adversary. For our theorems, that game will have low time and memory overhead over that of $\mathcal{A}$, so the application of the lemma would be time- and memory-tight. That said, the tightness of this is less important than the tightness of the other components of the theorem we would apply it to. Note that the definition of $\mathcal{A}_{\mathsf{i}}$ is *independent* of the choice of F. Consequently, we can always choose F with a very high security threshold to counteract any looseness in the lemma. In Appendix A, we summarize the $\mathcal{F}$ used in our theorems and how they could be pseudorandomly instantiated.

Assumed independence. As a second observation why the storage of f may not help $\mathcal{A}$, note that f is completely "independent" of the problem $\mathcal{A}$ is trying to solve (as specified by *in* and the behavior of O). In various settings it seems likely that such independent state does not help. For example, it would be very surprising (or even a breakthrough) to show a better factoring or lattice algorithm given access to a random function f from a natural set. Indeed, cryptanalytic work often makes use of random oracles without significant comment (from which other types of random functions can be constructed).

Information theoretic settings. In some information theoretic settings, the "independence" of f from the problem can be made rigorous. Information theoretic results are typically depending only on the query complexity of the attacker or its memory usage, *ignoring code size*. In such settings, we expect bounds of the form $\mathsf{Adv}(\mathcal{A}) \leqslant \epsilon(\mathbf{Mem}(\mathcal{A}), \mathbf{Query}(\mathcal{A}))$ for some function ϵ. Because this bound *does not* depend on the code size of $\mathcal{A}$, if $\mathcal{A}$ is an $\mathcal{F}$-oracle adversary we should be able to prove $\mathsf{Adv}(\mathcal{A}) \leqslant \epsilon(\mathbf{Mem}^*(\mathcal{A}), \mathbf{Query}(\mathcal{A}))$ by a coin-fixing argument in which we fix the random choice of function ahead of time and embed it in the description of the adversary. This is, for example, the case for the recent time-memory tradeoffs shown for distinguishing between a random function and a random injection without repeating queries [21,13,25]. A coin-fixing readily shows that these tradeoffs hold when using $\mathbf{Mem}^*(\mathcal{A})$ in place of $\mathbf{Mem}(\mathcal{A}), \mathbf{Query}(\mathcal{A})$.

4 Multi-challenge Security of Digital Signature Schemes

In the context of memory-tightness, the security of digital signature schemes has been considered in several works [1,27,12]. The standard security notion for signatures asks the attacker, given examples, to come up with a forged signature on a fresh message. A straightforward proof shows (in the standard setting where memory efficiency is not a concern) that the security notion is equivalent whether the attacker is allowed one or many forgery attempts. However, Auerbach et al. [1] proved an impossibility result showing that a (certain form of black-box) reduction cannot be time, memory, and advantage tight. The difficulty faced by the reduction is in distinguishing between when the adversary has produced a novel forgery and when it is simply repeating a signature that it was given.

In this section we show a few ways that security against many forgery attempts (i.e., multiple challenges) can be proven to follow from security against a single forgery (i.e., a single challenge) in a memory-tight manner. Our first results consider a variant definition of digital signature security we introduce (called UFRMA) in which the adversary has only partial control over the messages being signed. Using our new techniques, we show that single challenge UFCMA security implies multi-challenge UFRMA security in a memory-tight manner (for some practically relevant distributions over messages). We also consider the security of the RSA full domain hash digital signature scheme. Auerbach et al. [1] gave a memory-, but not advantage-tight proof of the security of the standard version of this scheme in the single challenge setting. By considering a probabilistic variant of the scheme introduced by Coron [10] we are able to provide a memory-, time-, and advantage-tight proof of the many-forgery SUFCMA security of the variant.

4.1 Syntax and Security

Digital signature syntax. A digital signature scheme DS specifies a key generation algorithm $\mathsf{DS.K}$, a signing algorithm $\mathsf{DS.Sign}$, and a verification algorithm $\mathsf{DS.Ver}$. The syntaxes of these algorithms are shown in Fig. 3. We capture ideals models by providing $\mathsf{DS.Sign}$ and $\mathsf{DS.Ver}$ with oracle access to a function h drawn at random from the set $\mathsf{DS.I}$. When relevant we let $\mathsf{DS.M}$ denote the set of messages it accepts. The verification and signing keys are respectively denoted by vk and sk. The message to be signed is m, the signature produced is σ, and the decision is $d \in \{\mathsf{true}, \mathsf{false}\}$. Correctness requires $\mathsf{DS.Ver}^{\mathrm{h}}(vk, m, \sigma) = \mathsf{true}$ for all $\mathrm{h} \in \mathsf{DS.I}$, all $(vk, sk) \in [\mathsf{DS.K}]$, all $m \in \mathsf{DS.M}$, and all $\sigma \in [\mathsf{DS.Sign}^{\mathrm{h}}(sk, m)]$.

DS Syntax
$\mathrm{h} \leftarrow\$ \mathsf{DS.I}$
$(vk, sk) \leftarrow\$ \mathsf{DS.K}$
$\sigma \leftarrow\$ \mathsf{DS.Sign}^{\mathrm{h}}(sk, m)$
$d \leftarrow \mathsf{DS.Ver}^{\mathrm{h}}(vk, m, \sigma)$

Fig. 3. Syntax of digital signature scheme.

Message distribution syntax. One of the security notions we consider for digital signature schemes will be parameterized by a message distribution via which the adversary is given incomplete control over the messages which are signed. A message distribution D specifies sampling algorithm $\mathsf{D.S}$ which samples an output message m' based on parameters m given as input (written $m' \leftarrow\$ \mathsf{D.S}(m)$). The parameters m must be drawn from a set $\mathsf{D.M}$, which we typically leave implicit. When making the randomness of the sampling algorithm explicit we let $\mathsf{D.R}$ be the set from which its randomness is drawn and write $m' \leftarrow \mathsf{D.S}(m; r)$. If there exists an extraction algorithm $\mathsf{D.X}$ such that $\mathsf{D.X}(\mathsf{D.S}(m; r)) = (m, r)$ for all m, r then we say D is *extractable*. If $\mathsf{D.X}(\mathsf{D.S}(m; r)) = m$ for all m, r then we say D is *weakly extractable*. We assume that $\mathsf{D.X}(m') = \bot$ if $m' \neq \mathsf{D.S}(m; r)$ for all m, r. We define the min-entropy of D as

$$\mathsf{D.H}_\infty = -\lg \max_m \Pr[r \leftarrow\$ \mathsf{D.R} : \mathsf{D.S}(m; r) = m'] \ .$$

Unforgeability security. The unforgeability security notions we consider are defined in Fig. 4. The standard notion of UFCMA (unforgeability under chosen message attack) security is captured by $\mathsf{G}^{\mathsf{ufcma}}$ which includes the boxed but not the highlighted code, giving the adversary access to a regular signing oracle Sign. The goal of the adversary is to query Forge with a valid signature σ^* of a message m^* which was not previously included in a signing query (as stored by the set $\mathcal{S}$). We define $\mathsf{Adv}^{\mathsf{ufcma}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{ufcma}}_{\mathsf{DS}}(\mathcal{A})]$.

Our new security notion UFRMA (unforgeability under randomized message attack) is captured by the game $\mathsf{G}^{\mathsf{ufrma}}$ which is parameterized by a message distribution D. In this game the adversary is instead given access to the randomized signing oracle RSign where the message to be signed is chosen by D. Note that the coins used by D are returned to the adversary along with the signature. Otherwise this game matches that of UFCMA security. We define $\mathsf{Adv}^{\mathsf{ufrma}}_{\mathsf{DS,D}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{ufrma}}_{\mathsf{DS,D}}(\mathcal{A})]$.

Game $\boxed{\mathsf{G}^{\mathsf{ufcma}}_{\mathsf{DS}}(\mathcal{A})}$, $\mathsf{G}^{\mathsf{ufrma}}_{\mathsf{DS,D}}(\mathcal{A})$	$\text{Sign}(m)$	$\text{RSign}(m)$
$\mathrm{h} \leftarrow\!\!\$\ \mathsf{DS.I}$	$\mathcal{S} \leftarrow \mathcal{S} \cup \{m\}$	$r \leftarrow\!\!\$\ \mathsf{D.R}$
$(vk, sk) \leftarrow\!\!\$\ \mathsf{DS.K}$	$\sigma \leftarrow\!\!\$\ \mathsf{DS.Sign}^{\mathrm{h}}(sk, m)$	$m' \leftarrow \mathsf{D.S}(m; r)$
$\mathcal{S} \leftarrow \varnothing$	Return σ	$\mathcal{S} \leftarrow \mathcal{S} \cup \{m'\}$
$\mathsf{win} \leftarrow \mathsf{false}$	$\text{Forge}(m^*, \sigma^*)$	$\sigma \leftarrow\!\!\$\ \mathsf{DS.Sign}^{\mathrm{h}}(sk, m')$
$\boxed{\text{Run } \mathcal{A}^{\text{Sign,Forge,h}}(vk)}$	If $m^* \notin \mathcal{S}$:	Return (σ, r)
Run $\mathcal{A}^{\text{RSign,Forge,h}}(vk)$	If $\mathsf{DS.Ver}^{\mathrm{h}}(vk, m, \sigma)$:	
Return win	$\mathsf{win} \leftarrow \mathsf{true}$	

Fig. 4. Security games capturing the unforgeability of a digital signature scheme.

We will relate the advantage of attacks making only a single forgery attempt and those making many such attempts. When wanting to make the distinction explicit we prefix the abbreviation of a security notion with an 'm' or '1'. Strong UFCMA security, denoted SUFCMA, is captured by modifying $\mathsf{G}^{\mathsf{ufcma}}$ to store the tuple (σ, m) in $\mathcal{S}$ in Sign and checking $(m^*, \sigma^*) \notin \mathcal{S}$ in Forge. We denote this by $\mathsf{G}^{\mathsf{sufcma}}$ and the corresponding advantage by $\mathsf{Adv}^{\mathsf{sufcma}}$. We define SUFRMA, $\mathsf{G}^{\mathsf{sufrma}}$, and $\mathsf{Adv}^{\mathsf{sufrma}}$ analogously. We write xUFRMA when assuming that D is extractable and wUFRMA when assuming it is weakly extractable.

4.2 Multi-Challenge Security for Extractable Message Distributions

The first applications we show for our techniques are generic methods of tightly implying security of a digital signature scheme against multiple forgery attempts (i.e., multi-challenge security). Recall that Auerbach et al. [1] gave a lower bound showing that a black-box reduction proving that single UFCMA security implies many UFCMA cannot be made memory-tight and time-tight. We avoid this in two ways; first by considering mUFRMA, rather than mUFCMA, security and then by considering a particular choice of digital signature scheme.

High-level idea. The primary difficulty of a tight proof that 1UFCMA security implies mUFCMA security is that a successful mUFCMA attacker may have made many Forge queries which verify correctly, one of which is a valid forgery and the rest of which were just forwarded from its Sign oracle. A 1UFCMA reduction must then somehow be able to identify which of the queries is the true forgery so it can forward this to its own Forge oracle.

The technical core of the coming proof for mUFRMA is that our reduction adversary will use the random coins of the message distribution D to signal things to its future self. In particular, when $\mathcal{A}_{\mathsf{r}}$ makes a query $\text{RSign}(m)$, the reduction will choose coins for $\mathsf{D.S}$ via $r \leftarrow f(m, i)$ where i is a counter which is incremented with each query and f is a random tweakable function/injection. The coins then act as a sort of authentication tag for m. On a later $\text{Forge}(m^*, \sigma^*)$ query, if $m^* = \mathsf{D.S}(m; r)$ where $r = f(m, i)$ for some $i \in [q_{\text{Sign}}]$ the reduction can safely assume this message was signed by an earlier RSign query.

When D is fully extractable, we can perform the requisite check for Forge by having f be an injection. We extract m and r from m^* and then compute $i \leftarrow f^{-1}(m, r)$. This is the strategy used in Theorem 1. If we assume only that D is weakly extractable, we can extract m if D has a sufficient amount of entropy, and then individually check if $\mathsf{D.S}(m; f(m, i))$ holds for each choice of i. This reduction strategy, used in Theorem 3, obtains the same advantage at the cost of an extra runtime being needed to iterate over the possible choices of i in Forge.

Extractable Message Distribution. If the message distribution D is extractable, the following theorem captures that 1UFCMA security tightly implies mUFRMA security. The proof makes use of our efficient tagging technique.

Theorem 1 (1UFCMA ⇒ mxUFRMA). *Let* DS *be a digital signature scheme and* D *be an extractable message distribution. Let* $\mathcal{A}_{\mathsf{r}}$ *be an adversary with* $(q_{\text{Sign}}, q_{\text{Forge}}, q_{\mathrm{h}}) = \mathbf{Query}(\mathcal{A}_{\mathsf{r}})$ *and assume* $q_{\mathsf{Sign}} \leqslant$

Adversary $\mathcal{A}_{\mathsf{u}}^{\text{SIGN},\text{FORGE},\mathrm{h}}(vk)$	$\text{SIMRSIGN}(m)$	$\text{SIMFORGE}(m^*, \sigma^*)$
$i \leftarrow 0$	$i \leftarrow i+1$	$(m, r) \leftarrow \mathsf{D.X}(m^*)$
$f \leftarrow\$ \mathsf{Inj}^{\pm}(\mathsf{DS.M}, [q_{\text{SIGN}}], \mathsf{D.R})$	$r \leftarrow f(m, i)$	If $f^{-1}(m, r) \notin [q_{\text{SIGN}}]$:
Run $\mathcal{A}_{\mathsf{r}}^{\text{SIMRSIGN},\text{SIMFORGE},\mathrm{h}}(vk)$	$m' \leftarrow \mathsf{D.S}(m; r)$	If $\mathsf{DS.Ver}^{\mathrm{h}}(vk, m^*, \sigma^*)$:
	$\sigma \leftarrow \text{SIGN}(m')$	Query $\text{FORGE}(m^*, \sigma^*)$
	Return (σ, r)	Halt execution

Fig. 5. Adversary $\mathcal{A}_{\mathsf{u}}$ used in proof of Theorem 1.

$0.5|\mathsf{D.R}|$. *Let* $\mathcal{A}_{\mathsf{u}}$ *be the* $\mathsf{Inj}^{\pm}(\mathsf{DS.M}, [q_{\text{SIGN}}], \mathsf{D.R})$*-oracle adversary shown in Fig. 5. Then,*

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{ufrma}}_{\mathsf{DS},\mathsf{D}}(\mathcal{A}_{\mathsf{r}}) &\leqslant \mathsf{Adv}^{\mathsf{ufcma}}_{\mathsf{DS}}(\mathcal{A}_{\mathsf{u}}) + (0.5 \cdot q_{\text{SIGN}}^2 + 2 \cdot q_{\text{SIGN}} \cdot q_{\text{FORGE}})/|\mathsf{D.R}| \\
\mathbf{Query}(\mathcal{A}_{\mathsf{u}}) &= (q_{\text{SIGN}}, 1, q_{\mathrm{h}} + q_{\text{FORGE}} \cdot \mathbf{Query}(\mathsf{DS})) \\
\mathbf{Time}^*(\mathcal{A}_{\mathsf{u}}) &= \mathbf{Time}(\mathcal{A}_{\mathsf{r}}) + q_{\text{SIGN}} \cdot \mathbf{Time}(\mathsf{D}) + q_{\text{FORGE}}(\mathbf{Time}(\mathsf{D}) + \mathbf{Time}(\mathsf{DS})) \\
\mathbf{Mem}^*(\mathcal{A}_{\mathsf{u}}) &= \mathbf{Mem}(\mathcal{A}_{\mathsf{r}}) + \mathbf{Mem}(\mathsf{D}) + \mathbf{Mem}(\mathsf{DS}) + \lg(q_{\text{SIGN}}).
\end{aligned}$$

This is time-tight because $\mathbf{Time}(\mathcal{A}_{\mathsf{r}}) \in \Omega(q_{\mathsf{Sign}} + q_{\text{FORGE}})$ must hold and $\mathbf{Time}(\mathsf{D})$ and $\mathbf{Time}(\mathsf{DS})$ will be small. This is memory-tight because $\mathbf{Mem}(\mathsf{D})$, $\mathbf{Mem}(\mathsf{DS})$, and $\lg(q_{\text{SIGN}})$ will be small.

The main idea of $\mathcal{A}_{\mathsf{u}}$ is using the output of an invertible random injection f on the message and a counter as coins instead of sampling them uniformly at random when answering RSIGN queries. Since D is fully extractable, during a FORGE query on m^*, we can extract $(m, r) \leftarrow \mathsf{D.X}(m^*)$ and use the fact that f is invertible to compute $f^{-1}(m, r)$ and check if the index is in $[q_{\text{SIGN}}]$. This is used to avoid remembering $\mathcal{S}$. If $m^* \in \mathcal{S}$, and $(m, r) \leftarrow \mathsf{D.X}(m^*)$, then there exists $j \in [q_{\text{SIGN}}]$ such that $r = f(m, j)$ – so the check passes. We can argue that if $m^* \notin \mathcal{S}$, our check is unlikely to pass. We give the formal proof of this theorem in Section 4.3. It applies the switching lemma to argue the use of f cannot be distinguished from honestly sampling r with advantage better than $0.5 \cdot q_{\text{SIGN}}^2/|\mathsf{D.R}|$ and shows that the probability of falsely making the check pass is bounded by $2q_{\text{SIGN}}q_{\text{FORGE}}/|\mathsf{D.R}|$.

We would not be able to use the technique in this proof to prove mxSUFRMA from 1SUFCMA in a memory-tight way. In particular, since the coins r of the message distribution are chosen before σ is known, our trick of using r to signal freshness of a forgery query does not work for a message-signature pair.

4.3 Proof of Theorem 1 (1UFCMA⇒mUFRMA)

Proof. We consider a sequence of hybrids H_0 through H_4 defined in Fig. 6. When examining these hybrids recall our conventions regarding "$//\mathsf{H}_{[i,j)}$" comments described in Sec. 2.1. Of these hybrids we will make the following claims, which establish the upper bound on the advantage of $\mathcal{A}_{\mathsf{r}}$ claimed in the proof.

1. $\Pr[\mathsf{G}^{\mathsf{ufrma}}_{\mathsf{DS},\mathsf{D}}(\mathcal{A}_{\mathsf{r}})] = \Pr[\mathsf{H}_0] = \Pr[\mathsf{H}_1]$
2. $\Pr[\mathsf{H}_1] \leqslant \Pr[\mathsf{H}_2] + 0.5 \cdot q_{\text{SIGN}}^2/|\mathsf{D.R}|$
3. $\Pr[\mathsf{H}_2] = \Pr[\mathsf{H}_3]$
4. $\Pr[\mathsf{H}_3] \leqslant \Pr[\mathsf{H}_4] + 2q_{\text{SIGN}}q_{\text{FORGE}}/|\mathsf{D.R}|$
5. $\Pr[\mathsf{H}_4] = \mathsf{Adv}^{\mathsf{ufcma}}_{\mathsf{DS}}(\mathcal{A}_{\mathsf{u}})$

TRANSITION H_0 TO H_1. The hybrid H_0 is simply a copy of the game $\mathsf{G}^{\mathsf{ufrma}}$. (We also added code to initialize variables i and $I[\cdot]$ that will be used in later hybrids.) Hence $\Pr[\mathsf{G}^{\mathsf{ufrma}}(\mathcal{A}_{\mathsf{r}})] = \Pr[\mathsf{H}_0]$. In hybrid H_1, we replace the random sampling of r for D in RSIGN with the output of a random function f applied to m, using a counter i to provide domain separation between different queries. This method of choosing r is equivalent, so $\Pr[\mathsf{H}_0] = \Pr[\mathsf{H}_1]$.

TRANSITION H_1 TO H_2. In hybrid H_2 we replace the random function with a random injection. This modifies the behavior of the game only in that values of r are guaranteed not to repeat across different signing queries that used the same message. There are at most q_{SIGN} invocations of f, so the switching lemma (Lemma 1) tells us that $\Pr[\mathsf{H}_1] \leqslant \Pr[\mathsf{H}_2] + 0.5 \cdot q_{\text{SIGN}}^2/|\mathsf{D.R}|$.

$\underline{\text{Games } \mathsf{H}_h \text{ for } 0 \leqslant h \leqslant 4}$
$\mathrm{h} \leftarrow\!\!\$\ \mathsf{DS.I}$
$(vk, sk) \leftarrow\!\!\$\ \mathsf{DS.K};\ \mathcal{S} \leftarrow \varnothing$
$\mathsf{win} \leftarrow \mathsf{false}$
$i \leftarrow 0;\ I[\cdot] \leftarrow \varnothing$
$f \leftarrow\!\!\$\ \mathsf{Fcs}(\mathsf{DS.M}, [q_{\text{SIGN}}], \mathsf{D.R})$ $/\!/\mathsf{H}_{[1,2)}$
$f \leftarrow\!\!\$\ \mathsf{Inj}^{\pm}(\mathsf{DS.M}, [q_{\text{SIGN}}], \mathsf{D.R})$ $/\!/\mathsf{H}_{[2,\infty)}$
Run $\mathcal{A}_{\mathsf{r}}^{\text{RSIGN},\text{FORGE},\mathrm{h}}(vk)$
Return win

$\underline{\text{RSIGN}(m)}$
$r \leftarrow\!\!\$\ \mathsf{D.R}$ $/\!/\mathsf{H}_{[0,1)}$
$i \leftarrow i + 1$ $/\!/\mathsf{H}_{[1,\infty)}$
$r \leftarrow f(m, i)$ $/\!/\mathsf{H}_{[1,\infty)}$
$I[m] \leftarrow I[m] \cup \{i\}$ $/\!/\mathsf{H}_{[3,4)}$
$m' \leftarrow \mathsf{D.S}(m; r)$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{m'\}$ $/\!/\mathsf{H}_{[0,3)}$
$\sigma \leftarrow\!\!\$\ \mathsf{DS.Sign}^{\mathrm{h}}(sk, m')$
Return (σ, r)

$\underline{\text{FORGE}(m^*, \sigma^*)}$
$(m, r) \leftarrow \mathsf{D.X}(m^*)$
If $m^* \notin \mathcal{S}$: $/\!/\mathsf{H}_{[0,3)}$
If $f^{-1}(m, r) \notin I[m]$: $/\!/\mathsf{H}_{[3,4)}$
If $f^{-1}(m, r) \notin [q_{\text{SIGN}}]$: $/\!/\mathsf{H}_{[4,\infty)}$
  If $\mathsf{DS.Ver}^{\mathrm{h}}(vk, m^*, \sigma^*)$:
    $\mathsf{win} \leftarrow \mathsf{true}$

Fig. 6. Hybrid games used in proof of Theorem 1.

TRANSITION H_2 TO H_3. In hybrid H_3, we replace the check whether $m^* \notin \mathcal{S}$ in oracle FORGE with a check if $f^{-1}(m, r) \notin I[m]$ where $(m, r) = \mathsf{D.X}(m^*)$. Here $I[\cdot]$ is a new table introduced into the game. In RSIGN, code was added which uses $I[m]$ to store each of the counter values for which $\mathcal{A}_{\mathsf{r}}$ made a signing query for m. Hence $f^{-1}(m, r)$ will be in $I[m]$ iff m^* is in $\mathcal{S}$ and so $\Pr[\mathsf{H}_2] = \Pr[\mathsf{H}_3]$.

TRANSITION H_3 TO H_4. In the final transition to hybrid H_4 we replace the FORGE check $f^{-1}(m, r) \notin I[m]$ with $f^{-1}(m, r) \notin [q_{\text{SIGN}}]$. This *does* change behavior if $\mathcal{A}_{\mathsf{r}}$ ever makes a successful forgery query for $m^* = \mathsf{D.S}(m; f(m, i))$ without its i-th signing query having used the message m. This would require guessing $f(m, i)$ for some $i \in [q_{\text{SIGN}}] \backslash I[m]$. We can bound the probability of this ever occurring by a union bound over the FORGE queries made by $\mathcal{A}_{\mathsf{r}}$. Consider the set $f(m, [q_{\text{SIGN}}] \backslash I[m]) = \{f(m, i) : i \in [q_{\text{SIGN}}] \backslash I[m]\}$. It has size at most q_{SIGN}. Because f is a random injection it is uniform subset of the set $\mathsf{D.R} \backslash f(m, I[m])$ (which has size at least $|\mathsf{D.R}| - q_{\text{SIGN}}$). Hence the probability of any particular query triggering this different behavior is at most $q_{\text{SIGN}}/(|\mathsf{D.R}| - q_{\text{SIGN}}) \leqslant 2q_{\text{SIGN}}/|\mathsf{D.R}|$. Applying the union bound gives us $\Pr[\mathsf{H}_3] \leqslant \Pr[\mathsf{H}_4] + 2q_{\text{SIGN}} \cdot q_{\text{FORGE}}/|\mathsf{D.R}|$.

REDUCTION TO UFCMA. We complete the proof using adversary $\mathcal{A}_{\mathsf{u}}$ from Fig. 5 which simulates hybrid H_4 and succeeds whenever $\mathcal{A}_{\mathsf{r}}$ would. The adversary $\mathcal{A}_{\mathsf{u}}$ samples f at random from $\mathsf{Inj}(\mathsf{DS.M}, [q_{\text{SIGN}}], \mathsf{D.R})$. When run on input vk, it runs $\mathcal{A}_{\mathsf{r}}$ on the same input. It gives $\mathcal{A}_{\mathsf{r}}$ direct access to h. To simulate a query $\text{RSIGN}(m)$, it computes $m' \leftarrow \mathsf{D.S}(m; f(m, i))$, increments i, and queries $\text{SIGN}(m')$, returning the result to $\mathcal{A}_{\mathsf{r}}$. On a query $\text{FORGE}(m^*, \sigma^*)$, it computes $(m, r) \leftarrow \mathsf{D.X}(m^*)$. If $f^{-1}(r) \notin [q_{\text{SIGN}}]$ and $\mathsf{DS.Ver}(vk, m^*, \sigma^*) = \mathsf{true}$ then it queries its own oracle with (m^*, σ^*) and halts. Otherwise it ignores the query.

If adversary $\mathcal{A}_{\mathsf{u}}$ ever makes a FORGE query, it will succeed. It ensured that (m^*, σ^*) is verified correctly and $f^{-1}(r) \notin [q_{\text{SIGN}}]$ ensures that it is has not previously made a SIGN query for m^*. If $\mathcal{A}_{\mathsf{r}}$ would have succeeded in hybrid H_4, its winning query will cause $\mathcal{A}_{\mathsf{u}}$ to make a FORGE query. Hence, we have $\Pr[\mathsf{H}_4] = \mathsf{Adv}^{\mathsf{ufcma}}_{\mathsf{DS}}(\mathcal{A}_{\mathsf{u}})$.

The claimed complexity of $\mathcal{A}_{\mathsf{u}}$ is straightforward. Clearly it makes q_{SIGN} queries to its signing oracle and 1 query to its forgery oracle. It forwards all of $\mathcal{A}_{\mathsf{r}}$'s queries to h and additionally may make queries when running $\mathsf{DS.Ver}$ in SIMFORGE giving $q_{\mathrm{h}} + q_{\text{FORGE}} \cdot \mathbf{Query}(\mathsf{DS.Ver})$ queries total. The time complexity of $\mathcal{A}_{\mathsf{u}}$ includes that of $\mathcal{A}_{\mathsf{r}}$, plus the time to execute $\mathsf{D.S}$ for each signing query, and the time to run $\mathsf{D.X}$ and $\mathsf{DS.Ver}$ for each forgery attempt. The memory complexity of $\mathcal{A}_{\mathsf{u}}$ includes that of $\mathcal{A}_{\mathsf{r}}$, plus the amount of memory required to run the algorithms $\mathsf{D.S}$, $\mathsf{D.X}$, and $\mathsf{DS.Ver}$ and to store the counter i. □

4.4 Applications and Weakly Extractable Variant

We discuss some applications of Theorem 1. This includes scenarios where extractable message distributions are used and proving security of digital signature schemes when their messages are padded with randomness. Additionally, we give a variant of the theorem when the underlying message distribution is only weakly extractable. The resulting reduction is memory- but not time-tight.

Example extractable distributions. The simplest extractable distribution does not accept parameters as input and simply outputs its randomness as the message. Security with respect to this is the standard notion of security against random message attacks which was originally introduced by Even, Goldreich, and Micali [15].

Extractable distributions arise naturally when the messages being signed include random values. For example, protocols often include random nonces in messages that are signed. In TLS 1.3, for example, when the server is responding to the Client Hello Message it signs a transcript of the conversation up until that point which includes a 256-bit nonce just chosen by the server. We could think of the security for this setting being captured by an extractable distribution D_{tls} that takes as input message parameter m that specifies all of the transcript other than the nonce and sets the nonce to its randomness $r \in \{0,1\}^{256}$.

Padding schemes with randomness. Using Theorem 1, we can see that augmenting any digital signature scheme by appending auxiliary randomness will give us a memory-tight reduction from the mUFCMA security of the augmented scheme to the 1UFCMA security of the original scheme.

Let DS be a digital signature scheme and R be a set. We define $\mathsf{RDS}[\mathsf{DS},\mathsf{R}]$ by having $\mathsf{RDS}[\mathsf{DS},\mathsf{R}].\mathsf{Sign}(sk,m)$ do "$r \leftarrow\!\!\$\ \mathsf{R}$; Return $\mathsf{DS}.\mathsf{Sign}(sk, m \,\|\, r) \,\|\, r$" and having $\mathsf{RDS}[\mathsf{DS},\mathsf{R}].\mathsf{Ver}(vk,m,\sigma')$ do "$\sigma \,\|\, r \leftarrow \sigma'$; Return $\mathsf{DS}.\mathsf{Ver}(vk, m \,\|\, r, \sigma)$." We also define a related message distribution $\mathsf{RD}[\mathsf{R}]$ by $\mathsf{RD}[\mathsf{R}].\mathsf{R} = \mathsf{R}$ and $\mathsf{RD}[\mathsf{R}].\mathsf{S}(m;r) = m \,\|\, r$. Clearly it is extractable.

The following reduces the mUFCMA security of RDS to the mUFRMA security of DS. Theorem 1 can in turn be used to reduce this to the 1UFCMA security of DS. It also relates the mSUFCMA security of RDS to the mSUFRMA security of DS. We note this because if DS has unique signatures, then its mSUFRMA and mUFRMA security are identical and hence UFCMA security of DS implies mSUFCMA security of RDS in a memory-tight way.

Theorem 2. *Let* DS *be a digital signature scheme and* R *be a set. Then for any* $\mathcal{A}_{\mathsf{u}}$ *we can construct* $\mathcal{A}_{\mathsf{r}}$ *such that* $\mathsf{Adv}^{\mathsf{ufcma}}_{\mathsf{RDS}[\mathsf{DS},\mathsf{R}]}(\mathcal{A}_{\mathsf{u}}) = \mathsf{Adv}^{\mathsf{ufrma}}_{\mathsf{DS},\mathsf{RD}[\mathsf{R}]}(\mathcal{A}_{\mathsf{r}})$. *It additionally holds that* $\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{RDS}[\mathsf{DS},\mathsf{R}]}(\mathcal{A}_{\mathsf{u}}) = \mathsf{Adv}^{\mathsf{sufrma}}_{\mathsf{DS},\mathsf{RD}[\mathsf{R}]}(\mathcal{A}_{\mathsf{r}})$. *Adversary* $\mathcal{A}_{\mathsf{r}}$ *has essentially the same complexity as* $\mathcal{A}_{\mathsf{u}}$.

Proof (Sketch). The proof of this is straightforward. If $\mathcal{A}_{\mathsf{u}}$ queries $\textsc{Sign}(m)$, then $\mathcal{A}_{\mathsf{r}}$ queries $\textsc{Sign}(m)$ and receives (σ, r) and returns$\sigma \,\|\, r$ to $\mathcal{A}_{\mathsf{u}}$. If $\mathcal{A}_{\mathsf{u}}$ queries $\textsc{Forge}(m^*, \sigma^* \,\|\, r^*)$, then $\mathcal{A}_{\mathsf{r}}$ queries $\textsc{Forge}(m^* \,\|\, r^*, \sigma^*)$. Note that $\mathcal{A}_{\mathsf{r}}$ wins whenever $\mathcal{A}_{\mathsf{u}}$ would. ◻

In independent and concurrent work, Diemert, Gellert, Jager, and Lyu [12] also considered RDS, proving that if DS can be proven SUFCMA1 secure (in this notion the game records its responses to signature queries and repeats them if the adversary repeats a query) with a restricted class of "canonical" memory-tight reductions, then there is a memory-tight reduction for the mSUFCMA security of RDS. This complements our results as they use a more restrictive assumption to prove mSUFCMA while we use a generic assumption to prove mUFCMA.

Weakly extractable message distribution. If D is only weakly extractable (but still has high entropy), then we can prove a variant of Theorem 1 with a less efficient reduction. (In particular, the running time of the reduction has an additional term of $q_{\text{Forge}} \cdot q_{\text{Sign}} \cdot \mathbf{Time}(\mathsf{D.S})$.) This difference arises because rather than extracting r and computing $j \leftarrow f^{-1}(m,r)$ in $\textsc{Forge}$ we instead need to iterate over the possible values of $f(m,j)$ to check for consistency. Thus the proof is an instance of our inefficient tagging technique.

Theorem 3 (1UFCMA ⇒ mwUFRMA). *Let* DS *be a digital signature scheme and* D *be a weakly extractable message distribution. Let* $\mathcal{A}_{\mathsf{r}}$ *be an adversary with* $(q_{\text{Sign}}, q_{\text{Forge}}, q_{\text{h}}) = \mathbf{Query}(\mathcal{A}_{\mathsf{r}})$. *Define the*

Adversary $\mathcal{A}_{\mathsf{u}}^{\text{Sign},\text{Forge},\mathrm{h}}(vk)$	$\text{SimRSign}(m)$	$\text{SimForge}(m^*,\sigma^*)$
$i \leftarrow 0$	$i \leftarrow i+1$	$(m,r) \leftarrow \mathsf{D.X}(m^*)$
$f \leftarrow\!\!\$\ \mathsf{Fcs}(\mathsf{DS.M},[q_{\text{Sign}}],\mathsf{D.R})$	$r \leftarrow f(m,i)$	If $\forall j \in [q_{\text{Sign}}], \mathsf{D.S}(m; f(m,j)) \neq m^*$:
Run $\mathcal{A}_{\mathsf{r}}^{\text{SimRSign},\text{SimForge},\mathrm{h}}(vk)$	$m' \leftarrow \mathsf{D.S}(m;r)$	If $\mathsf{DS.Ver}^{\mathrm{h}}(vk,m^*,\sigma^*)$:
	$\sigma \leftarrow \text{Sign}(m')$	Query $\text{Forge}(m^*,\sigma^*)$
	Return (σ,r)	Halt execution

Fig. 7. Reduction adversary used in proof of Theorem 3.

Games H_h for $0 \leqslant h \leqslant 4$	$\text{RSign}(m)$
$\mathrm{h} \leftarrow\!\!\$\ \mathsf{DS.I}$	$r \leftarrow\!\!\$\ \mathsf{D.R}$ //$\mathsf{H}_{[0,1)}$
$(vk,sk) \leftarrow\!\!\$\ \mathsf{DS.K}$; $\mathcal{S} \leftarrow \varnothing$	$i \leftarrow i+1$ //$\mathsf{H}_{[1,\infty)}$
$\mathsf{win} \leftarrow \mathsf{false}$	$r \leftarrow f(m,i)$ //$\mathsf{H}_{[1,\infty)}$
$i \leftarrow 0$; $I[\cdot] \leftarrow \varnothing$	$I[m] \leftarrow I[m] \cup \{i\}$ //$\mathsf{H}_{[3,4)}$
$f \leftarrow\!\!\$\ \mathsf{Fcs}(\mathsf{DS.M},[q_{\text{Sign}}],\mathsf{D.R})$ //$\mathsf{H}_{[1,\infty)}$	$m' \leftarrow \mathsf{D.S}(m;r)$
Run $\mathcal{A}_{\mathsf{r}}^{\text{RSign},\text{Forge},\mathrm{h}}(vk)$	$\mathcal{S} \leftarrow \mathcal{S} \cup \{m'\}$ //$\mathsf{H}_{[0,3)}$
Return win	$\sigma \leftarrow\!\!\$\ \mathsf{DS.Sign}^{\mathrm{h}}(sk,m')$
	Return (σ,r)
	$\text{Forge}(m^*,\sigma^*)$
	$m \leftarrow \mathsf{D.X}(m^*)$
	If $m^* \notin \mathcal{S}$: //$\mathsf{H}_{[0,3)}$
	If $\forall j \in I[m], \mathsf{D.S}(m; f(m,j)) \neq m^*$: //$\mathsf{H}_{[3,4)}$
	If $\forall j \in [q_{\text{Sign}}], \mathsf{D.S}(m; f(m,j)) \neq m^*$: //$\mathsf{H}_{[4,\infty)}$
	If $\mathsf{DS.Ver}(vk,m^*,\sigma^*)$:
	$\mathsf{win} \leftarrow \mathsf{true}$

Fig. 8. Hybrid games used in proof of Theorem 3.

$\mathsf{Fcs}(\mathsf{DS.M},[q_{\text{Sign}}],\mathsf{D.R})$-*oracle adversary* $\mathcal{A}_{\mathsf{u}}$ *as shown in Fig. 7. Then.*

$$\mathsf{Adv}^{\mathsf{ufrma}}_{\mathsf{DS},\mathsf{D}}(\mathcal{A}_{\mathsf{r}}) \leqslant \mathsf{Adv}^{\mathsf{ufcma}}_{\mathsf{DS}}(\mathcal{A}_{\mathsf{u}}) + q_{\text{Sign}} \cdot q_{\text{Forge}} \cdot 2^{-\mathsf{D.H}_\infty}$$
$$\mathbf{Query}(\mathcal{A}_{\mathsf{u}}) = (q_{\text{Sign}}, 1, q_{\mathrm{h}} + q_{\text{Forge}} \cdot \mathbf{Query}(\mathsf{DS}))$$
$$\mathbf{Time}^*(\mathcal{A}_{\mathsf{u}}) = \mathbf{Time}(\mathcal{A}_{\mathsf{r}}) + q_{\text{Sign}} \cdot \mathbf{Time}(\mathsf{D}) + q_{\text{Forge}}(q_{\text{Sign}}\mathbf{Time}(\mathsf{D}) + \mathbf{Time}(\mathsf{DS}))$$
$$\mathbf{Mem}^*(\mathcal{A}_{\mathsf{u}}) = \mathbf{Mem}(\mathcal{A}_{\mathsf{r}}) + \mathbf{Mem}(\mathsf{D}) + \mathbf{Mem}(\mathsf{DS}) + \lg(q_{\text{Sign}}).$$

This running time is not time-tight because $\mathbf{Time}(\mathcal{A}_{\mathsf{r}}) \in O(q_{\mathsf{Sign}} + q_{\text{Forge}})$ may hold, while $\mathbf{Time}^*(\mathcal{A}_{\mathsf{u}}) \in \Omega(q_{\text{Forge}} \cdot q_{\text{Sign}})$. This is memory-tight because we expect $\mathbf{Mem}(\mathsf{D}) + \mathbf{Mem}(\mathsf{DS}) + \lg(q_{\text{Sign}})$ to be small. Recall that, as discussed in the introduction, such non-time-tight reductions may be useful when the best attack for the underlying problem with low memory requires significantly more running time than the best attack with high memory.

This theorem is useful when the messages being signed include values derived from randomness which are hard to invert to recover the underlying randomness. Examples of this include the signing of public keys, as is done for certificates or some key exchange protocols, and the signing of ciphertexts, as is done for signcryption. We could capture such settings with appropriate choices of D.

The main idea behind adversary $\mathcal{A}_{\mathsf{u}}$ is very similar to the idea behind the adversary in Theorem 1. Because we now only assume weak extractability, we extract m and then iterate over each choice of j to check whether $\mathsf{D.S}(m; f(m,j)) = m^*$. Moreover, since we no longer need to invert f here, it suffices for f to be a random function. We give formal proof of Theorem 3.

Proof. We consider a sequence of hybrids H_0 through H_4 defined in Fig. 8. Of these hybrids we will make the following claims, which establish the upper bound on the advantage of $\mathcal{A}_\mathsf{r}$ claimed in the proof.

1. $\Pr[\mathsf{G}^{\mathsf{ufrma}}_{\mathsf{DS},\mathsf{D}}(\mathcal{A}_\mathsf{r})] = \Pr[\mathsf{H}_0] = \Pr[\mathsf{H}_1]$
2. $\Pr[\mathsf{H}_1] = \Pr[\mathsf{H}_2]$
3. $\Pr[\mathsf{H}_2] = \Pr[\mathsf{H}_3]$
4. $\Pr[\mathsf{H}_3] \leqslant \Pr[\mathsf{H}_4] + q_{\text{SIGN}} \cdot q_{\text{FORGE}} \cdot 2^{-\mathsf{D}.\mathsf{H}_\infty}$
5. $\Pr[\mathsf{H}_4] = \mathsf{Adv}^{\mathsf{ufcma}}_{\mathsf{DS}}(\mathcal{A}_\mathsf{u})$

TRANSITION TO H_0. The hybrid H_0 is simply a copy of the game $\mathsf{G}^{\mathsf{ufrma}}$. (We also added code to initialize variables i and $I[\cdot]$ that will be used in later hybrids.) Hence $\Pr[\mathsf{G}^{\mathsf{ufrma}}(\mathcal{A}_\mathsf{r})] = \Pr[\mathsf{H}_0]$.

TRANSITION H_0 TO H_1. In hybrid H_1, we replace the random sampling of r for D in RSIGN with the output of a random function f applied to m, using a counter i to provide domain separation between different queries. This method of choosing r is equivalent, so $\Pr[\mathsf{H}_0] = \Pr[\mathsf{H}_1]$.

TRANSITION H_1 TO H_2. Hybrid H_2 is identical to hybrid H_1. So $\Pr[\mathsf{H}_1] = \Pr[\mathsf{H}_2]$. We include this redundant hybrid to maintain consistency of the hybrid numbers with the proof of Theorem 3.

TRANSITION H_2 TO H_3. In hybrid H_3, we replace the check whether $m^* \notin \mathcal{S}$ in oracle FORGE with a check if $\forall j \in I[m], \mathsf{D.S}(m; f(m,j)) \neq m^*$. Here $I[\cdot]$ is a new table introduced into the game. In RSIGN, code was added which uses $I[m]$ to store each of the counter values for which $\mathcal{A}_\mathsf{r}$ made a signing query for m. It is easy to see that if for all $j \in I[m]$, $\mathsf{D.S}(m; f(m,j)) \neq m^*$, then $m^* \notin \mathcal{S}$. Also if there exists $j \in I[m]$ such that $\mathsf{D.S}(m; f(m,j)) = m^*$, the j-th signing query was on m and hence $m^* \in \mathcal{S}$. Therefore the new check is equivalent to the replaced one and so $\Pr[\mathsf{H}_2] = \Pr[\mathsf{H}_3]$.

TRANSITION H_3 TO H_4. In the final transition to hybrid H_4 we replace the FORGE check $\forall j \in I[m]$, $\mathsf{D.S}(m; f(m,j)) \neq m^*$ with $\forall j \in [q_{\text{SIGN}}], \mathsf{D.S}(m; f(m,j)) \neq m^*$. This *does* change behavior if $\mathcal{A}_\mathsf{r}$ ever makes a successful forgery attempt for $m^* = \mathsf{D.S}(m; f(m,j))$ without its j-th signing query having used the message m. Note that the view of the adversary would be independent of $f(m,j)$ in this case and hence $f(m,j)$ can be thought of as a value chosen uniformly at random from $\mathsf{D.R}$. Thus for every FORGE query and $j \in [q_{\text{SIGN}}]$ the probability that $m^* = \mathsf{D.S}(m; f(m,j))$ is at most $2^{-\mathsf{D}.\mathsf{H}_\infty}$. By an union bound over all values of j it follows that for every FORGE this happens with probability at most $q_{\text{FORGE}} \cdot 2^{-\mathsf{D}.\mathsf{H}_\infty}$. By a union bound over all FORGE queries we get that $\Pr[\mathsf{H}_3] \leqslant \Pr[\mathsf{H}_4] + q_{\text{SIGN}} \cdot q_{\text{FORGE}} \cdot 2^{-\mathsf{D}.\mathsf{H}_\infty}$.

REDUCTION TO UFCMA. We complete the proof by designing an adversary $\mathcal{A}_\mathsf{u}$ (see Fig. 7)). It is easy to see that adversary $\mathcal{A}_\mathsf{u}$ simulates hybrid H_4 to $\mathcal{A}_\mathsf{r}$ and succeeds whenever $\mathcal{A}_\mathsf{r}$ would. It follows that, $\Pr[\mathsf{H}_4] = \mathsf{Adv}^{\mathsf{ufcma}}_{\mathsf{DS}}(\mathcal{A}_\mathsf{u})$.

The claimed complexity of $\mathcal{A}_\mathsf{u}$ is straightforward. Clearly it makes q_{SIGN} queries to its signing oracle and 1 query to its forgery oracle. It forwards all of $\mathcal{A}_\mathsf{r}$'s queries to h and additionally may make queries when running $\mathsf{DS.Ver}$ in SIMFORGE giving $q_\mathrm{h} + q_{\text{FORGE}} \cdot \mathbf{Query}(\mathsf{DS.Ver})$ queries total. The time complexity of $\mathcal{A}_\mathsf{u}$ includes that of $\mathcal{A}_\mathsf{r}$, plus the time to execute $\mathsf{D.S}$ for each signing query, and the time to run $\mathsf{D.X}$ at most q_{SIGN} times and $\mathsf{DS.Ver}$ once for each forgery attempt. The memory complexity of $\mathcal{A}_\mathsf{u}$ includes that of $\mathcal{A}_\mathsf{r}$, plus the memory required to run the algorithms $\mathsf{D.S}$, $\mathsf{D.X}$, and $\mathsf{DS.Ver}$ and to store the counter i. ◻

4.5 mSUFCMA Security of RSA-PFDH

We saw that augmenting any digital signature scheme by including extra randomness gives us a memory-tight reduction for the mUFCMA security of the augmented scheme from the 1UFCMA security of the original scheme in Theorem 2. Now we will consider a particular signature scheme, RSA-based Probabilistic Full-Domain Hash (RSA-PFDH) which was originally introduced by Coron [10]. This is a digital signature scheme obtained by including extra randomness in the standard RSA-based Full Domain Hash (RSA-FDH) scheme. Theorem 2 gives us a memory-tight reduction from the mUFCMA security of RSA-PFDH to the 1UFCMA security of RSA-FDH, but the UFCMA security reduction of the latter to hardness of inverting the RSA permutation is not tight in terms of advantage. In this section, we use the efficient tagging trick to give a direct reduction from the mSUFCMA security of RSA-PFDH to the hardness of RSA which is tight in terms of both memory and advantage.

$\underline{\mathsf{RSA.K}}$	$\underline{\mathsf{RSA.Sign}^{\mathrm{h}}(sk, m)}$	$\underline{\mathsf{RSA.Ver}^{\mathrm{h}}(vk, m^*, \sigma^*)}$
$(N, e, d) \leftarrow\$ \mathsf{R.Gen}$	$(N, d) \leftarrow sk$	$(N, e) \leftarrow vk$
$sk \leftarrow (N, d)$	$r \leftarrow\$ \{0,1\}^{\mathsf{rl}}$	$z \,\|\, r \leftarrow \sigma^*$
$vk \leftarrow (N, e)$	$w \leftarrow \mathrm{h}(N, m \,\|\, r)$	$w \leftarrow z^e \bmod N$
Return (sk, vk)	$z \leftarrow w^d \bmod N$	Return $(w = \mathrm{h}(N, m^* \,\|\, r))$
	Return $z \,\|\, r$	

Fig. 10. Digital signature scheme $\mathsf{RSA} = \mathsf{RSA\text{-}PFDH}[\mathsf{R}, \mathsf{rl}]$.

RSA TRAPDOOR PERMUTATION. The RSA function defines a trapdoor permutation that is plausibly one-way. It is based on the observation that given modulus $N \in \mathbb{N}$ and an integer $e \geqslant 2$ relatively prime to $\phi(N)$ (where ϕ is Euler's totient function), exponentiation to the e-th power modulo N is a permutation on $\mathbb{Z}_N^*$. An RSA generator R specifies a generation algorithm $\mathsf{R.Gen}$ such that $\mathsf{R.Gen}$ returns (N, e, d) where N is an integer such that e is co-prime to N and $d = e^{-1} \bmod \phi(N)$. We assume N is always of a fixed bit-length $\mathsf{R.k}$. Typically $N = pq$ for distinct $(\mathsf{R.k}/2)$-bit primes p and q. The one-wayness of an RSA generator R is defined by the game $\mathsf{G}^{\mathsf{ow\text{-}rsa}}_{\mathsf{R}}$ in Fig. 9. The game runs $\mathsf{R.Gen}$ to obtain (N, e, d), and then samples x uniformly at random from $\mathbb{Z}_N^*$. It computes $y = x^e \bmod N$ and runs the adversary on input (N, e, y). The adversary wins if it returns $x' = x$. The advantage of an adversary $\mathcal{A}$ against RSA generator R is $\mathsf{Adv}^{\mathsf{ow\text{-}rsa}}_{\mathsf{R}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{ow\text{-}rsa}}_{\mathsf{R}}(\mathcal{A})]$.

We let $\mathbf{Time}(\mathsf{R})$ denote the time required by $\mathsf{R.Gen}$ plus an upper bound on the time to compute multiplication or exponentiation by e in $\mathbb{Z}_N^*$ for any (N, e, d) output but R. We define $\mathbf{Mem}(\mathsf{R})$ analogously.

$\underline{\text{Game } \mathsf{G}^{\mathsf{ow\text{-}rsa}}_{\mathsf{R}}(\mathcal{A})}$
$(N, e, d) \leftarrow\$ \mathsf{R.Gen}$
$x \leftarrow\$ \mathbb{Z}_N^*$
$y \leftarrow x^e \bmod N$
$x' \leftarrow \mathcal{A}(N, e, y)$
Return $x = x'$

Fig. 9. RSA one-wayness security game.

FULL DOMAIN HASH. Full Domain Hash (FDH) [5] is a digital signature scheme where the message m is first hashed using a hash function h onto the domain of a one-way trapdoor permutation f. Then the signature is $f^{-1}(\mathrm{h}(m))$. When instantiated with the RSA trapdoor permutation, it is known as RSA-FDH.

Assuming h is a random oracle, it can be proven [5,9] that for every adversary $\mathcal{A}$ that makes q_{SIGN} queries to its signing oracle and achieves advantage ϵ against the UFCMA security of RSA-FDH, we can construct an adversary $\mathcal{B}$ that breaks RSA with advantage $\epsilon' \approx q_{\text{SIGN}} \cdot \epsilon$. Auerbach et al. [1] showed how to make this reduction memory-tight.

In order to overcome the advantage loss factor of q_{SIGN}, Coron [10] introduced Probabilistic FDH (PFDH) where a random salt r is hashed with the message m and the signature is $f^{-1}(\mathrm{h}(m \,\|\, r)) \,\|\, r$. Using Theorem 2, and the result of Auerbach et al. we can give a memory-tight reduction for the mUFCMA security of RSA-PFDH, but the reduction is not-tight in terms of advantage. Via a more direct proof we will obtain a reduction that is tight in all metrics.

We let $\mathsf{RSA\text{-}PFDH}[\mathsf{R}, \mathsf{rl}]$ denote the instantiation of RSA-PFDH with a given RSA generator R and randomness length $\mathsf{rl} \in \mathbb{N}$. For compactness we typically define $\mathsf{RSA} = \mathsf{RSA\text{-}PFDH}[\mathsf{R}, \mathsf{rl}]$. Its algorithms are given in Fig. 10. They expect access to a random oracle $\mathrm{h} \in \mathsf{RSA\text{-}PFDH}[\mathsf{R}, \mathsf{rl}].\mathsf{I} = \mathsf{Fcs}(\{0,1\}^{\mathsf{R.k}}, \{0,1\}^*, \mathbb{Z}^*_{(\cdot)})$. Typical analysis of FDH constructions (e.g., [5,9]) uses a single non-tweakable hash function with range Z_N^*. Thus h depends on N which is determined by the verification key used. This dependence does not make sense with our notation conventions for treating ideal models, so we instead represent h as a hash function tweaked by N. Here the $\mathbb{Z}^*_{(\cdot)}$ indicates that for $h \in \mathsf{RSA\text{-}PFDH}[\mathsf{R}, \mathsf{rl}].\mathsf{I}$, the function $h(N, \cdot)$ has a range of $\mathbb{Z}_N^*$. Thus $\mathsf{RSA\text{-}PFDH}[\mathsf{R}, \mathsf{rl}]$ is the same as the scheme considered by Coron, merely with different notational conventions.

SECURITY RESULT. The following result gives a memory-tight and advantage-tight reduction for the mSUFCMA security of RSA-PFDH.

Adversary $\mathcal{A}_{\mathsf{RSA}}(N, e, y)$

$f_1 \leftarrow_{\$} \mathsf{Fcs}(\{0,1\}^{\mathsf{R.k}}, \{0,1\}^*, \mathbb{Z}^*_{(\cdot)})$
$f_2 \leftarrow_{\$} \mathsf{Inj}^{\pm}(\{0,1\}^*, [q_{\text{SIGN}}], \{0,1\}^{\mathsf{rl}})$
$i \leftarrow 0$
Run $\mathcal{A}_{\mathsf{m}}^{\text{SIGN},\text{FORGE},\text{h}}((N, e))$

$\text{h}(N', m \,\|\, r)$

If $N \neq N'$:
  Return $f_1(N', m \,\|\, r)$
If $f_2^{-1}(m, r) \notin [q_{\text{SIGN}}]$:
  Return $y \cdot f_1(N, m \,\|\, r)^e \bmod N$
Return $f_1(N, m \,\|\, r)^e \bmod N$

$\text{SIGN}(m)$

$i \leftarrow i + 1$
$r \leftarrow f_2(m, i)$
$z \leftarrow f_1(N, m \,\|\, r)$
$\sigma \leftarrow z \,\|\, r$
Return σ

Oracle $\text{FORGE}(m^*, \sigma^*)$

$z \,\|\, r \leftarrow \sigma^*$
If $f_2^{-1}(m, r) \notin [q_{\text{SIGN}}]$:
  $w \leftarrow z^e \bmod N$
  If $w = \text{h}(N, m \,\|\, r)$:
    $\text{Halt}(z \cdot f_1(N, m \,\|\, r)^{-1} \bmod N)$

Fig. 11. Adversary for Theorem 4. Highlighting shows where it halts with the specified output.

Theorem 4 (mSUFCMA security of RSA-PFDH). *Let* R *be an RSA generator. Let* $\mathsf{rl} \in \mathbb{N}$ *and assume* $\mathsf{rl} < \mathsf{R.k}$. *Let* $\mathsf{RSA} = \mathsf{RSA\text{-}PFDH}[\mathsf{R}, \mathsf{rl}]$. *Let* $\mathcal{A}_{\mathsf{m}}$ *be an adversary with* $(q_{\text{SIGN}}, q_{\text{FORGE}}, q_{\text{h}}) = \mathbf{Query}(\mathcal{A}_{\mathsf{m}})$ *and assume* $q_{\text{SIGN}} \leqslant 2^{\mathsf{rl}-1}$. *Let* $\mathcal{A}_{\mathsf{RSA}}$ *be the adversary defined in Fig. 11. Then,*

$$\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{RSA}}(\mathcal{A}_{\mathsf{m}}) \leqslant \mathsf{Adv}^{\mathsf{ow\text{-}rsa}}_{R}(\mathcal{A}_{\mathsf{RSA}}) + (0.5 \cdot q^2_{\text{SIGN}} + 2 \cdot q_{\text{SIGN}} \cdot q_{\text{FORGE}})/2^{\mathsf{rl}}$$
$$\mathbf{Time}^*(\mathcal{A}_{\mathsf{RSA}}) = O(\mathbf{Time}(\mathcal{A}_{\mathsf{m}})) + O((q_{\text{h}} + q_{\text{FORGE}})\mathbf{Time}(\mathsf{R}))$$
$$\mathbf{Mem}^*(\mathcal{A}_{\mathsf{RSA}}) = O(\mathbf{Mem}(\mathcal{A}_{\mathsf{m}})) + O(\mathbf{Mem}(\mathsf{R})) + \lg(q_{\text{SIGN}}).$$

Adversary $\mathcal{A}_{\mathsf{RSA}}$ *is an* $\mathcal{F}$*-oracle adversary for*

$$\mathcal{F} = \mathsf{Fcs}(\{0,1\}^{\mathsf{R.k}}, \{0,1\}^*, \mathbb{Z}^*_{(\cdot)}) \times \mathsf{Inj}(\{0,1\}^*, [q_{\text{SIGN}}], \{0,1\}^{\mathsf{rl}}).$$

The advantage of the adversary $\mathcal{A}_{\mathsf{RSA}}$ is nearly the same as the advantage of $\mathcal{A}_{\mathsf{m}}$ if rl is chosen so that $0.5 \cdot q^2_{\text{SIGN}} + 2 \cdot q_{\text{SIGN}} \cdot q_{\text{FORGE}} \ll 2^{\mathsf{rl}}$ will hold. The reduced complexity of $\mathcal{A}_{\mathsf{RSA}}$ is nearly the same as the running time and memory of $\mathcal{A}_{\mathsf{m}}$. Therefore, the reduction is tight with respect to advantage, time, and memory.

Its main idea of $\mathcal{A}_{\mathsf{RSA}}$ is based around its simulation of h using the random function f_1. We have $\text{h}(N', m \,\|\, r)$ simply return $f_1(N', m \,\|\, r)$ whenever $N' \neq N$ because these queries are not relevant for security. We want to return $f_1(N, m \,\|\, r)^e \bmod N$ for any $m \,\|\, r$ that will be signed in SIGN. This allows us to return $f_1(N, m \,\|\, r) \,\|\, r$ as the signature because then $\text{h}(N, m \,\|\, r)^d \bmod N = f_1(N, m \,\|\, r)$. Finally we want to embed our challenge y into all other h queries by returning $y \cdot f_1(N, m \,\|\, r)^e \bmod N$. Then given a forgery with respect to such a value (that is, given z such that $z^e \bmod N = \text{h}(N, m \,\|\, r)$) we can see that $y^d \bmod N = z \cdot f_1(N, m \,\|\, r)^{-1} \bmod N$, solving the RSA game.

If memory was not a concern we could sample r at random in SIGN and remember each $m \,\|\, r$ used to respond appropriately to them in h and FORGE. To make things memory-tight we instead chose r as $f_2(m, i)$ for a random function f_2 and counter i. Then whenever we see $m \,\|\, r$ such that $f_2^{-1}(m, r) \in [q_{\text{SIGN}}]$ we assume this must have been used in a signing query and respond appropriately. In concurrent work, Diemert, Gellert, Jager, and Lyu [12] also give a time, memory, and advantage tight reduction for RSA-PFDH via a different proof.

We give the formal proof of Theorem 4.

Proof. We consider a sequence of hybrids H_0 through H_3 and L_0 through L_3 defined in Fig. 12 and 13. Of these hybrids we will make the following claims, which establish the upper bound on the advantage of $\mathcal{A}_{\mathsf{m}}$ claimed in the proof.

1. $\Pr[\mathsf{G}^{\mathsf{sufcma}}_{\mathsf{RSA}}(\mathcal{A}_{\mathsf{m}})] = \Pr[\mathsf{H}_0] = \Pr[\mathsf{H}_1] = \Pr[\mathsf{H}_2]$
2. $\Pr[\mathsf{H}_2] \leqslant \Pr[\mathsf{H}_3] + 0.5 \cdot q^2_{\text{SIGN}}/2^{\mathsf{rl}}$
3. $\Pr[\mathsf{H}_3] = \Pr[\mathsf{L}_0]$
4. $\Pr[\mathsf{L}_0] \leqslant \Pr[\mathsf{L}_1] + 2q_{\text{SIGN}} \cdot q_{\text{FORGE}}/2^{\mathsf{rl}}$
5. $\Pr[\mathsf{L}_1] = \Pr[\mathsf{L}_2]$
6. $\Pr[\mathsf{L}_2] = \mathsf{Adv}^{\mathsf{ow\text{-}rsa}}_{\mathsf{R}}(\mathcal{A}_{\mathsf{RSA}})$

Transition to H_0. Hybrid H_0 was obtained by embedding the code of RSA into $\mathsf{G}^{\mathsf{sufcma}}$ and rewriting h as an oracle that simply evaluates the random function f_1 chosen in the same way that h was sampled. Hence, $\Pr[\mathsf{G}^{\mathsf{sufcma}}_{\mathsf{RSA}}(\mathcal{A}_{\mathsf{m}})] = \Pr[\mathsf{H}_0]$.

Transition from H_0 to H_1. In hybrid H_1, we assign $f_1(N, m \| r)^e \bmod N$ to $\mathrm{h}(N, m \| r)$ instead of $f_1(N, m \| r)$. Because exponentiation by e us a permutation this is still a uniformly random element. Then for a Sign query, we directly assign $f_1(N, m \| r)$ to z since $(f_1(N, m \| r)^e)^d \bmod N = f_1(N, m \| r)$. This does not change the distribution of the hash values or the signatures, so $\Pr[\mathsf{H}_0] = \Pr[\mathsf{H}_1]$.

Transition from H_1 to H_2. In hybrid H_2, we replace the random sampling of r during signing with the output of a random function f_2, using a counter i to provide domain separation between different Sign queries. This method of choosing r is equivalent, so $\Pr[\mathsf{H}_1] = \Pr[\mathsf{H}_2]$.

Transition from H_2 to H_3. In hybrid H_3 we replace the random function f_2 with a random injection (tweaked by the message m). This modifies the behavior of the game only in that values of r are guaranteed not to repeat across different signing queries that used the same message. There are at most q_{Sign} invocations of f_2, so the switching lemma (Lemma 1) tells us that $\Pr[\mathsf{H}_2] \leqslant \Pr[\mathsf{H}_3] + 0.5 \cdot q_{\text{Sign}}^2 / 2^{\mathsf{rl}}$.

Transition from H_3 to L_0. Now consider L_0 shown in Fig. 13. We've used grey highlighting to indicate places where code was changed from H_3 to L_0. In hybrid L_0's oracle Forge, we replace the check whether $(m^*, \sigma^*) \notin \mathcal{S}$ with a check whether $f^{-1}(m, r) \notin I[m]$. Here $I[\cdot]$ is a new table; code was added to Sign which in $I[m]$ stores each of the counter values for which $\mathcal{A}_{\mathsf{m}}$ made a signing query for m. Note that $f^{-1}(m, r)$ is in $I[m]$ iff the $f^{-1}(m, r)$-th signing query was for m and returned $z' \| r$ for some z'. Hence, if $(m^*, \sigma^*) \in \mathcal{S}$, then $f^{-1}(m, r) \in I[m]$. If $(m^*, \sigma^*) \notin \mathcal{S}$ either $f^{-1}(m, r) \notin I[m]$ or $f^{-1}(m, r) \in I[m]$, but in the latter case it must be that $\sigma^* = z \| r$ for some z not equal to the z' returned by the signing query and so $z^e \neq \mathrm{h}(N, m^* \| r)$. Hence these games are equivalent, giving $\Pr[\mathsf{H}_3] = \Pr[\mathsf{L}_0]$.

Transition L_0 to L_1. Next, in hybrid L_1 we replace the Forge check $f^{-1}(m, r) \notin I[m]$ with $f^{-1}(m, r) \notin [q_{\text{Sign}}]$. Detecting this change requires guessing $f_2(m^*, i)$ for some $i \in [q_{\text{Sign}}] \backslash I[m]$. We can bound the probability of this ever occurring in L_0 by a union bound over the Forge queries made by the adversary. Consider the set $f(m^*, [q_{\text{Sign}}] \backslash I[m^*]) = \{f(m^*, i) : i \in [q_{\text{Sign}}] \backslash I[m^*]\}$. It has size at most q_{Sign}. Because f_2 is a random injection, this is uniform subset of the set $\{0,1\}^{\mathsf{rl}} \backslash f_2(m^*, I[m^*])$ (which has size at least $2^{\mathsf{rl}} - q_{\text{Sign}}$). Hence the probability of any particular query being the first to trigger this different behavior is at most $q_{\text{Sign}} / (2^{\mathsf{rl}} - q_{\text{Sign}}) \leqslant 2 q_{\text{Sign}} / 2^{\mathsf{rl}}$. Applying the union bound gives us $\Pr[\mathsf{L}_0] \leqslant \Pr[\mathsf{L}_1] + 2 q_{\text{Sign}} \cdot q_{\text{Forge}} / 2^{\mathsf{rl}}$.

Transition from L_1 to L_2. In hybrid L_2 we now begin the game by sampling x at random from $\mathbb{Z}_N^*$ and setting $y \leftarrow x^e \bmod N$. Our goal is to "embed" y into the responses to random oracle queries so that a successful forgery allows x to be recovered. In particular, we change the output of h whenever $f^{-1}(m, r) \notin [q_{\text{Sign}}]$, now returning $y \cdot f_2(m \| r)^e \bmod N$. Note that multiplying the fixed element $y \in \mathbb{Z}_N^*$ by a uniformly random element still gives a uniformly random element. Because we only perform this modification for $f^{-1}(m, r) \notin [q_{\text{Sign}}]$, it will not cause any inconsistency with Sign where $f^{-1}(m, r) \in [q_{\text{Sign}}]$ always holds. Hence the view of the adversary is unchanged and so $\Pr[\mathsf{L}_1] = \Pr[\mathsf{L}_2]$.

Reduction to RSA. We complete the proof by arguing that $\mathcal{A}_{\mathsf{RSA}}$ perfectly simulates hybrid L_2 and succeeds whenever $\mathcal{A}_{\mathsf{m}}$ would. It is formally defined in Fig. 11. Examining its code, we can see that the code is basically identical to that of L_2, except it is given (N, e, y) as input rather than generating them locally. The grey highlighted code in Forge shows where $\mathcal{A}_{\mathsf{RSA}}$ will halt early whenever $\mathcal{A}_{\mathsf{m}}$ would win. The check immediately before this ensures that $z^e \bmod N = y \cdot f_3(m \| r)^e \bmod N$. Hence $y = (z \cdot (f_3(m \| r))^{-1})^e \bmod N$, meaning that $z \cdot (f_3(m \| r))^{-1}$ is indeed the correct response. Hence $\Pr[\mathsf{L}_2] = \mathsf{Adv}^{\mathsf{ow\text{-}rsa}}_{R}(\mathcal{A}_{\mathsf{RSA}})$.

The complexity of $\mathcal{A}_{\mathsf{RSA}}$ follows from its description. The counter i is the additional $\lg(q_{\text{Sign}})$ storage. The **Time**(R) and **Mem**(R) terms come from performing operations in $\mathbb{Z}_N^*$ for h and Forge queries. □

5 AE Security of Encrypt-then-PRF

For nonce-based secret-key encryption schemes, we often want Authenticated Encryption (AE) security which simultaneously asks for confidentiality and ciphertext integrity. The common approach to prove AE

Hybrids H_h for $0 \leqslant h \leqslant 3$

$(N, e, d) \leftarrow\$ \mathsf{R.Gen}$
$\mathcal{S} \leftarrow \varnothing$
$f_1 \leftarrow\$ \mathsf{Fcs}(\{0,1\}^{\mathsf{R.k}}, \{0,1\}^*, \mathbb{Z}^*_{(\cdot)})$
$f_2 \leftarrow\$ \mathsf{Fcs}(\{0,1\}^*, [q_{\text{SIGN}}], \{0,1\}^{\mathsf{rl}})$ $//\mathsf{H}_{[2,3)}$
$f_2 \leftarrow\$ \mathsf{Inj}(\{0,1\}^*, [q_{\text{SIGN}}], \{0,1\}^{\mathsf{rl}})$ $//\mathsf{H}_{[3,\infty)}$
$i \leftarrow 0$ $//\mathsf{H}_{[2,\infty)}$
$\mathsf{win} \leftarrow \mathsf{false}$
Run $\mathcal{A}_{\mathsf{m}}^{\text{SIGN},\text{FORGE},\mathrm{h}}((N, e))$
Return win

$\mathrm{h}(N', m \,\|\, r)$

Return $f_1(N', m \,\|\, r)$ $//\mathsf{H}_{[0,1)}$
If $N \neq N'$: $//\mathsf{H}_{[1,\infty)}$
  Return $f_1(N', m \,\|\, r)$ $//\mathsf{H}_{[1,\infty)}$
Return $f_1(N, m \,\|\, r)^e \bmod N$ $//\mathsf{H}_{[1,\infty)}$

SIGN(m)

$i \leftarrow i + 1$ $//\mathsf{H}_{[2,\infty)}$
$r \leftarrow\$ \{0,1\}^{\mathsf{rl}}$ $//\mathsf{H}_{[0,2)}$
$r \leftarrow f_2(m, i)$ $//\mathsf{H}_{[2,\infty)}$
$w \leftarrow \mathrm{h}(N, m \,\|\, r)$ $//\mathsf{H}_{[0,1)}$
$z \leftarrow w^d \bmod N$ $//\mathsf{H}_{[0,1)}$
$z \leftarrow f_1(N, m \,\|\, r)$ $//\mathsf{H}_{[1,\infty)}$
$\sigma \leftarrow z \,\|\, r$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{m, \sigma\}$
Return σ

FORGE(m^*, σ^*)

$z \,\|\, r \leftarrow \sigma^*$
If $(m^*, \sigma^*) \notin \mathcal{S}$:
  $w \leftarrow z^e \bmod N$
  If $w = \mathrm{h}(N, m^* \,\|\, r)$:
    $\mathsf{win} \leftarrow \mathsf{true}$

Fig. 12. Hybrid games H_0 through H_3 used in proof of Theorem 4.

Hybrids L_h for $0 \leqslant h \leqslant 2$

$(N, e, d) \leftarrow\$ \mathsf{R.Gen}$
$x \leftarrow\$ \mathbb{Z}^*_N$; $y \leftarrow x^e \bmod N$ $//\mathsf{L}_{[2,\infty)}$
$I[\cdot] \leftarrow \varnothing$
$f_1 \leftarrow\$ \mathsf{Fcs}(\{0,1\}^{\mathsf{R.k}}, \{0,1\}^*, \mathbb{Z}^*_{(\cdot)})$
$f_2 \leftarrow\$ \mathsf{Inj}^{\pm}(\{0,1\}^*, [q_{\text{SIGN}}], \{0,1\}^{\mathsf{rl}})$
$i \leftarrow 0$
$\mathsf{win} \leftarrow \mathsf{false}$
Run $\mathcal{A}_{\mathsf{m}}^{\text{SIGN},\text{FORGE},\mathrm{h}}((N, e))$
Return win

$\mathrm{h}(N', m \,\|\, r)$

If $N \neq N'$:
  Return $f_1(N', m \,\|\, r)$
If $f_2^{-1}(m, r) \notin [q_{\text{SIGN}}]$: $//\mathsf{L}_{[2,\infty)}$
  Return $y \cdot f_1(N, m \,\|\, r)^e \bmod N$ $//\mathsf{L}_{[2,\infty)}$
Return $f_1(N, m \,\|\, r)^e \bmod N$

SIGN(m)

$i \leftarrow i + 1$
$I[m] \leftarrow I[m] \cup \{i\}$
$r \leftarrow f_2(m, i)$
$z \leftarrow f_1(N, m \,\|\, r)$
$\sigma \leftarrow z \,\|\, r$
Return σ

FORGE(m^*, σ^*)

$z \,\|\, r \leftarrow \sigma^*$
If $f_2^{-1}(m^*, r) \notin I[m^*]$: $//\mathsf{L}_{[0,1)}$
If $f_2^{-1}(m^*, r) \notin [q_{\text{SIGN}}]$: $//\mathsf{L}_{[1,\infty)}$
  $w \leftarrow z^e \bmod N$
  If $w = \mathrm{h}(N, m^* \,\|\, r)$:
    $\mathsf{win} \leftarrow \mathsf{true}$

Fig. 13. Hybrid games L_0 through L_2 used in proof of Theorem 4. Grey highlighting indicates where L_0 differs from H_3.

---

security of a nonce-based encryption scheme is to give separate reductions to the indistinguishability of its ciphertexts from truly random ones (INDR security) and its ciphertext integrity. Ghoshal et al. [17] proved an impossibility result showing that a (certain form of black-box) reduction from AE security to INDR security and ciphertext integrity cannot be memory-tight. Making the INDR part memory-tight is of particular interest because of results which establish tight time-memory trade-offs for INDR security [26,21,14,11,25].

In this section we look at a particular scheme which we refer to as Encrypt-then-PRF. Given a nonce-based encryption scheme NE that only has INDR security, one generic way to construct a new encryption scheme NE′ which also achieves ciphertext integrity is to use a PRF and let the ciphertext of NE′ be the concatenation of the ciphertext of NE and a tag which is the evaluation of the PRF on the ciphertext and the nonce.

Game $\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},b}(\mathcal{A})$	$\text{Enc}_b(n,m)$	$\text{Dec}^w_b(n,c)$
$K \leftarrow\$ \mathsf{NE.K}$	$c_1 \leftarrow \mathsf{NE.E}(K,n,m)$	If $M[n,c] \neq \bot$:
$b' \leftarrow \mathcal{A}^{\text{Enc}_b}$	$c_0 \leftarrow\$ \{0,1\}^{\mathsf{NE.cl}(\lvert m \rvert)}$	Return $M[n,c]$ if $w = \mathtt{m}$
Return $b' = 1$	$M[n,c_b] \leftarrow m$	Return $\diamond$ if $w = \diamond$
Game $\mathsf{G}^{\mathsf{ae}\text{-}w}_{\mathsf{NE},b}(\mathcal{A})$	Return c_b	Return $\bot$ if $w = \bot$
$K \leftarrow\$ \mathsf{NE.K}$		$m_1 \leftarrow \mathsf{NE.D}(k,n,c)$
$b' \leftarrow \mathcal{A}^{\text{Enc}_b,\text{Dec}^w_b}$		$m_0 \leftarrow \bot$
Return $b' = 1$		Return m_b

Fig. 14. Games defining INDR and AE-w security of NE for $w \in \{\mathtt{m}, \diamond, \bot\}$.

We show that in the context of Encrypt-then-PRF, for two of the notions of AE security introduced in [17], we can give a memory-tight reduction to the INDR security of the underlying encryption scheme and a non-memory-tight reduction to the security of the PRF. This shows that we can bypass the generic impossibility result of [17] if we consider specific constructions of nonce-based authenticated encryption schemes. In more detail, the impossibility result of [17] rules out lifting the INDR security of a scheme to full AE security in a memory tight way, when additionally assuming ciphertext integrity *for a generic scheme*. Here, we show that for the *specific* case of Encrypt-then-PRF schemes, lifting the INDR security of the encryption scheme to full AE security of Encrypt-then-PRF is possible in a memory-tight way, assuming security of the PRF.

5.1 Syntax and Security Definitions

Nonce-Based Encryption. A nonce-based (secret-key) encryption scheme NE specifies algorithms $\mathsf{NE.K}$, $\mathsf{NE.E}$, and $\mathsf{NE.D}$. It specifies message space $\mathsf{NE.M}$ and nonce space $\mathsf{NE.N}$. The syntax of the algorithms is shown in Fig. 15. The secret key is denoted by K, the message is m, the nonce is n, and the ciphertext is c. The decryption algorithm may return $m = \bot$ to indicate rejection of the ciphertext. Correctness requires for all $K \in [\mathsf{NE.K}]$, $n \in \mathsf{NE.N}$, and $m \in \mathsf{NE.M}$ that $\mathsf{NE.D}(K,n,\mathsf{NE.E}(K,n,m)) = m$. We assume there is a ciphertext-length function $\mathsf{NE.cl} : \mathbb{N} \to \mathbb{N}$ such that for all $K \in [\mathsf{NE.K}]$, $n \in \mathsf{NE.N}$, and $m \in \mathsf{NE.M}$ we have $|c| = \mathsf{NE.cl}(|m|)$ where $c \leftarrow \mathsf{NE.E}(K,n,m)$. We define $\mathsf{NE.C} = \bigcup_{m \in \mathsf{NE.M}} \{0,1\}^{\mathsf{NE.cl}(|m|)}$. Typically, a nonce-based encryption scheme also takes associated data as input which is authenticated during encryption. This does not meaningfully affect our proof, so we omit it for simplicity.

NE Syntax
$K \leftarrow\$ \mathsf{NE.K}$
$c \leftarrow \mathsf{NE.E}(K,n,m)$
$m \leftarrow \mathsf{NE.D}(K,n,c)$

Fig. 15. Syntax of (nonce-based) secret-key encryption scheme.

Encrypt-then-PRF. In this section we consider the Encrypt-then-PRF construction of a nonce-based encryption scheme, due to Rogaway [24]. Namprempre et al. [23] gave a more extensive exploration of the many ways to construct an AEAD encryption scheme via generic composition. Given nonce-based encryption scheme NE and function family F, we define $\mathsf{EtP}[\mathsf{NE},\mathsf{F}]$ by the following algorithms. We refer to the t component of the ciphertext returned by $\mathsf{EtP}[\mathsf{NE},\mathsf{F}].\mathsf{E}$ as the "tag" below. When including associated data, it would be input to F.

$\mathsf{EtP}[\mathsf{NE},\mathsf{F}].\mathsf{K}$	$\mathsf{EtP}[\mathsf{NE},\mathsf{F}].\mathsf{E}(K,n,m)$	$\mathsf{EtP}[\mathsf{NE},\mathsf{F}].\mathsf{D}(K,n,c)$
$K \leftarrow\$ \mathsf{NE.K}$	$(K,K') \leftarrow K$	$(K,K') \leftarrow K$; $(c',t) \leftarrow c$
$K' \leftarrow\$ \mathsf{F.K}$	$c' \leftarrow \mathsf{NE.E}(K,n,m)$	If $t = \mathsf{F}_{K'}(n,c')$:
Return (K,K')	$t \leftarrow \mathsf{F}_{K'}(n,c')$	Return $\mathsf{NE.D}(K,n,m)$
	Return (c',t)	Return $\bot$

Our security result will analyze the authenticated security of EtP assuming NE has ciphertexts indistinguishable from random ciphertexts and F is pseudorandom. Let us recall these security notions.

Indistinguishability From Random (INDR) Security. This security notion requires that ciphertexts output by the encryption scheme cannot be distinguished from random strings. Consider the game $\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},b}$ defined in Fig. 14. Here an adversary $\mathcal{A}$ is given access to an encryption oracle $\textsc{Enc}_b$ to which it can query a pair (n, m) and receive an honest encryption of message m with nonce n if $b = 1$ or a random string of the appropriate length if $b = 0$. We restrict attention to "valid" adversaries that never repeat the nonce n across different encryption queries. We define $\mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},0}(\mathcal{A})]$.

Authenticated Encryption (AE) security. AE security simultaneously asks for integrity and confidentiality. Consider the games $\mathsf{G}^{\mathsf{ae}\text{-}w}_{\mathsf{NE},b}$ which defines three variants of authenticated encryption security parameterized by $w \in \{\mathtt{m}, \diamond, \bot\}$ shown in Fig. 14. In this game, the adversary is given access to an encryption oracle and a decryption oracle. Its goal is to distinguish between a "real" and "ideal" world. In the real world $(b = 1)$ the oracles use NE to encrypt messages and decrypt ciphertexts. In the ideal world $(b = 0)$ encryption returns random messages of the appropriate length and decryption returns $\bot$. For simplicity, we will again restrict attention nonce-respecting adversaries which do not repeat nonces across encryption queries. (Note that there is no restriction placed on nonces used for decryption queries.)

The decryption oracle is parameterized by the value $w \in \{\mathtt{m}, \diamond, \bot\}$ corresponding to three different security notions. In all three, we use a table $M[\cdot, \cdot]$ to detect when the adversary forwards encryption queries on to its decryption oracle. When $w = \mathtt{m}$, the decryption oracle returns $M[n, c]$. When $w = \diamond$, it returns a special symbol $\diamond$. When $w = \bot$, it returns the symbol $\bot$ which is also used by the encryption scheme to represent rejection. For $w \in \{\mathtt{m}, \diamond, \bot\}$ we define the advantage of an adversary $\mathcal{A}$ by $\mathsf{Adv}^{\mathsf{ae}\text{-}w}_{\mathsf{NE}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{ae}\text{-}w}_{\mathsf{NE},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\mathsf{ae}\text{-}w}_{\mathsf{NE},0}(\mathcal{A})]$.

Discussion of variants. This choice of considering three variants of the definition follows the same choice made by Ghoshal et al. [17]. First off, we note that *if there are no restrictions on the memory of the adversary, all the three definitions are tightly equivalent.* An adversary can simply remember its past encryption queries and answers, and without loss of generality never make a decryption query on the answer of an encryption query. In the memory restricted setting these definitions *no longer appear to be equivalent.* The only known implication is that $w = \diamond$ security tightly implies $w = \bot$ security. Other implications seem to require remembering all encryption queries to properly simulate the decryption oracle. In Sec. 6 we parameterize public-key encryption CCA definitions similarly. This discussion applies to those definitions as well.

Ghoshal et al. argued that $w = \mathtt{m}$ is the "correct" definition. They argue that chosen ciphertext security is intended to capture the power of an adversary that can observe the behavior of a decrypting party. Both the $w = \bot$ and $w = \diamond$ definitions restrict what the adversary learns about this behavior when honestly generated ciphertexts are forwarded, which does not seem to model anything about real use of encryption. The $w = \mathtt{m}$ definition avoids this unnatural restriction.

We provide some technical context for this philosophical argument. In Appendix B we give memory-tight proofs for the security of encryption schemes constructed with the KEM/DEM paradigm with $w = \mathtt{m}$ and noting this does not seem possible for the other choices of w. In this section and Sec. 6 we prove the AE/CCA-w security of encryption schemes for differing choices of w. We view this as a general exploration of what results are possible with memory-tight proofs. A proof which works for some w, but not others helps build some understanding of how these notions related.

5.2 Security Result

Now we give a proof of the AE-$\diamond$ security of $\mathsf{EtP}[\mathsf{NE}, \mathsf{F}]$. In particular we provide a memory-tight reduction to the INDR security of NE and a non-memory-tight reduction to the security of F. Such a result is useful if a time-memory tradeoff is known for NE and F is sufficiently secure even against high-memory attackers.

Theorem 5 (Security of EtP). *Let* NE *be a nonce-based encryption scheme and* F *be a family of function with* $\mathsf{F.F} = \mathsf{Fcs}(\mathsf{NE.N}, \mathsf{NE.C}, \{0,1\}^\tau)$ *for* $\tau \in \mathbb{N}$. *Let* $\mathcal{A}_a$ *be an* AE-$\diamond$ *adversary with* $(q_{\text{Enc}}, q_{\text{Dec}}) = \mathbf{Query}(\mathcal{A}_a)$.

Adversary $\mathcal{A}_{\mathsf{p}}^{\text{Ev}}$	Adversary $\mathcal{A}_r^{\text{Enc}}$
$K \leftarrow\$ \mathsf{NE.K}$	$f \leftarrow\$ \mathsf{Fcs}(\mathsf{NE.N}, \mathsf{NE.M}, \{0,1\}^\tau)$
$b' \leftarrow \mathcal{A}_a^{\text{SimEnc},\text{SimDec}}$	$b' \leftarrow \mathcal{A}_a^{\text{SimEnc},\text{SimDec}}$
Return b'	Return b'
$\text{SimEnc}(n, m)$	$\text{SimEnc}(n, m)$
$c' \leftarrow \mathsf{NE.E}(K, n, m)$	$c' \leftarrow \text{Enc}(n, m)$
$t \leftarrow \text{Ev}(n, c')$	$t \leftarrow f(n, c')$
$c \leftarrow (t, c')$	$c \leftarrow (t, c')$
$M[n, c] \leftarrow m$	Return c
Return c	$\text{SimDec}(n, c)$
$\text{SimDec}(n, c)$	$(t, c') \leftarrow c$
If $M[n, c] \neq \bot$: Return $\diamond$	If $t = f(n, c')$: Return $\diamond$
$(t, c') \leftarrow c$	Return $\bot$
If $t = \text{Ev}(n, c')$: Return $\mathsf{NE.D}(k, n, c')$	
Return $\bot$	

Fig. 16. Adversaries used for proof of Theorem 5.

Define adversaries $\mathcal{A}_{\mathsf{p}}$ and $\mathcal{A}_r$ as shown in Fig. 16. Then,

$$\mathsf{Adv}^{\mathsf{ae}\text{-}\diamond}_{\mathsf{EtP[NE,F]}}(\mathcal{A}_a) \leqslant \mathsf{Adv}^{\mathsf{pr}}_{\mathsf{F}}(\mathcal{A}_{\mathsf{p}}) + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{A}_r) + 2q_{\text{Dec}}/2^\tau$$

$$\mathbf{Query}(\mathcal{A}_{\mathsf{p}}) = q_{\text{Enc}} + q_{\text{Dec}} \qquad \mathbf{Query}(\mathcal{A}_r) = q_{\text{Enc}}$$
$$\mathbf{Time}(\mathcal{A}_{\mathsf{p}}) = \mathbf{Time}(\mathsf{G}^{\mathsf{ae}\text{-}\diamond}_{\mathsf{EtP[NE,F]}}(\mathcal{A}_a)) \qquad \mathbf{Time}^*(\mathcal{A}_r) = \mathbf{Time}(\mathcal{A}_a)$$
$$\mathbf{Mem}(\mathcal{A}_{\mathsf{p}}) = \mathbf{Mem}(\mathsf{G}^{\mathsf{ae}\text{-}\diamond}_{\mathsf{EtP[NE,F]}}(\mathcal{A}_a)) \qquad \mathbf{Mem}^*(\mathcal{A}_r) = \mathbf{Mem}(\mathcal{A}_a).$$

Adversary $\mathcal{A}_r$ is an $\mathsf{F.F}$-oracle adversary.

The standard (not memory-tight) proof of the security of EtP begins identically to our proof; we start in $\mathsf{G}^{\mathsf{ae}\text{-}\diamond}_{\mathsf{EtP[NE,F]},1}$ replace the use of F with a truly random function (using $\mathcal{A}_{\mathsf{p}}$) and then information theoretically argue that the attacker shall be incapable of creating any forgeries. In the standard proof we would transition to a game where the decryption oracle is exactly that of $\text{Dec}_0^\diamond$, i.e. it always returns $\bot$ when $M[n, c] = \bot$. Then we reduce to the security of NE to replace the generated ciphertexts with random. However this standard reduction will not be memory-tight because the attacker must store the table $M[\cdot, \cdot]$ to know whether it should return $\diamond$ or $\bot$ when simulating decryption queries.[7] Instead we first transition to a world where F has been replaced by the random function f and Dec always returns $\diamond$ when given a ciphertext with a correct tag. (Which we can do because either $M[n, c] \neq \bot$ held or the attacker managed to guess a random tag, which is unlikely.) Now we can make our INDR reduction memory-tight. It forwards encryption queries to its encryption oracle and then uses its own function f to create the tag. For decryption queries it checks $f(n, c') = t$, returning $\diamond$ if so and $\bot$ otherwise. Then we can finally conclude by switching to the decryption oracle $\text{Dec}_0^\diamond$ by arguing that noticing this change requires guessing a random tag.

It does not seem possible to extend this proof technique to AE-$\mathtt{m}$ security because the tag would be too short to embed values of m we need to remember.

We give the formal proof of Theorem 5.

Proof. We consider a sequence of hybrids H_0 through H_4 defined in Fig. 17. Of these hybrids we will make the following claims, which establish the upper bound on the advantage of $\mathcal{A}_a$ claimed in the proof.

[7] Note this *would* be memory-tight for AE-$\bot$ security.

Games H_h for $0 \leqslant h \leqslant 4$	$\textsc{Enc}(n, m)$	$\textsc{Dec}(n, c)$
$K \leftarrow\!\!\$\ \mathsf{NE.K}$	$c' \leftarrow \mathsf{NE.E}(K, n, m)$ $/\!/\mathsf{H}_{[0,3)}$	If $M[n, c] \neq \bot$: Return $\diamond$
$K' \leftarrow\!\!\$\ \mathsf{F.K}$ $/\!/\mathsf{H}_{[0,1)}$	$c' \leftarrow\!\!\$\ \{0,1\}^{\mathsf{NE.cl}(\lvert m \rvert)}$ $/\!/\mathsf{H}_{[3,\infty)}$	$(t, c') \leftarrow c$
$f \leftarrow\!\!\$\ \mathsf{Fcs}(\mathsf{NE.N}, \mathsf{NE.C}, \{0,1\}^\tau)$ $/\!/\mathsf{H}_{[1,\infty)}$	$t \leftarrow \mathsf{F}_{K'}(n, c')$ $/\!/\mathsf{H}_{[0,1)}$	If $t = F_{K'}(n, c')$: $/\!/\mathsf{H}_{[0,1)}$
$b' \leftarrow \mathcal{A}_a^{\textsc{Enc},\textsc{Dec}}$	$t \leftarrow f(n, c')$ $/\!/\mathsf{H}_{[1,\infty)}$	If $t = f(n, c')$: $/\!/\mathsf{H}_{[1,\infty)}$
Return $b' = 1$	$c \leftarrow (t, c')$	$\mathsf{bad} \leftarrow \mathsf{true}$
	$M[n, c] \leftarrow m$	Return $\mathsf{NE.D}(K, n, c')$ $/\!/\mathsf{H}_{[0,2)}$
	Return c	Return $\diamond$ $/\!/\mathsf{H}_{[2,4)}$
		Return $\bot$ $/\!/\mathsf{H}_{[4,\infty)}$
		Return $\bot$

Fig. 17. Hybrid games for proof of Theorem 5.

1. $\Pr[\mathsf{G}^{\mathsf{ae}\text{-}\diamond}_{\mathsf{EtP[NE,F]},1}(\mathcal{A}_a)] = \Pr[\mathsf{H}_0]$
2. $\Pr[\mathsf{H}_0] \leqslant \Pr[\mathsf{H}_1] + \mathsf{Adv}^{\mathsf{pr}}_{\mathsf{F}}(\mathcal{A}_{\mathsf{p}})$
3. $\Pr[\mathsf{H}_1] \leqslant \Pr[\mathsf{H}_2] + q_{\textsc{Dec}}/2^\tau$
4. $\Pr[\mathsf{H}_2] \leqslant \Pr[\mathsf{H}_3] + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{A}_r)$
5. $\Pr[\mathsf{H}_3] \leqslant \Pr[\mathsf{H}_4] + q_{\textsc{Dec}}/2^\tau$
6. $\Pr[\mathsf{H}_4] = \Pr[\mathsf{G}^{\mathsf{ae}\text{-}\diamond}_{\mathsf{EtP[NE,F]},0}(\mathcal{A}_a)]$

The claims regarding the complexities of the adversaries considered are clear from their code.

Transition to H_0. The hybrid H_0 was obtained by plugging the code of $\mathsf{EtP[NE,F]}$ into $\mathsf{G}^{\mathsf{ae}\text{-}\diamond}_{\mathsf{EtP[NE,F]},1}(\mathcal{A}_a)$, so the first claim is clear.

Transition H_0 to H_1. In H_1 we replace each use of F with a random f sampled from $\mathsf{F.F}$. The reduction to the PR security of F is given by $\mathcal{A}_{\mathsf{p}}$ in Fig. 16. It simply simulates these hybrids for $\mathcal{A}_a$, using its $\textsc{Ev}$ oracle in place of F or f. The claimed bound follows (and is in fact an equality).

Transition H_1 to H_2. In H_2, we change the behavior of $\textsc{Dec}$. Now the oracle returns $\diamond$ when $M[n, c] = \bot$ and $t = f(n, c')$. Note that this is the only case in which the hybrids we are considering differ. In particular, they are identical-until-bad and so the Fundamental Lemma of Game Playing [6] gives $\Pr[\mathsf{H}_1] \leqslant \Pr[\mathsf{H}_2] + \Pr[\mathsf{H}_2 \text{ sets } \mathsf{bad}]$. Setting bad requires guessing a value $f(n, c')$ which is a uniform value in $\{0,1\}^\tau$. Hence by a union bound $\Pr[\mathsf{H}_2 \text{ sets } \mathsf{bad}] \leqslant q_{\textsc{Dec}}/2^\tau$, giving the claim.

Transition H_2 to H_3. In H_3, we replace the real encryption of m in $\textsc{Enc}$ with a uniformly random c'. Consider the adversary $\mathcal{A}_r$ given in Fig. 16. It perfectly simulates hybrid H_2 to $\mathcal{A}_a$ when interacting with $\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},1}$ and hybrid H_3 to $\mathcal{A}_a$ when interacting with $\mathsf{G}^{\mathsf{indr}}_{\mathsf{NE},0}$. Note that $\mathcal{A}_r$ avoids storing the table M – this is possible because observe that $M[n, c] \neq \bot$ holds if and only if $f(n, c') = t$ also holds in $\mathsf{H}_2, \mathsf{H}_3$. Since the check $M[n, c] \neq \bot$ is the sole place in the code of hybrids $\mathsf{H}_2, \mathsf{H}_3$ where M affects execution, $\mathcal{A}_r$ replaces it with the equivalent check $f(n, c') = t$ and avoids storing M. So it follows that $\Pr[\mathsf{H}_2] \leqslant \Pr[\mathsf{H}_3] + \mathsf{Adv}^{\mathsf{indr}}_{\mathsf{NE}}(\mathcal{A}_r)$.

Transition H_3 to H_4. In H_4, we change the behavior of $\textsc{Dec}$. Now the oracle returns $\bot$ when $M[n, c] = \bot$ and $t = f(n, c')$. Using an identical-until-bad argument analogous to when we transitioned between H_1 and H_2 we get $\Pr[\mathsf{H}_1] \leqslant \Pr[\mathsf{H}_2] + q_{\textsc{Dec}}/2^\tau$ as desired.

Final transition. Finally we claim that the view of $\mathcal{A}_a$ in H_4 is identical to its view in $\mathsf{G}^{\mathsf{ae}\text{-}\diamond}_{\mathsf{EtP[NE,F]},0}$, giving $\Pr[\mathsf{H}_4] = \Pr[\mathsf{G}^{\mathsf{ae}\text{-}\diamond}_{\mathsf{EtP[NE,F]},0}(\mathcal{A}_a)]$. If $M[n, c] = \bot$ in $\textsc{Dec}$, then the oracle it always returns $\bot$ so we can ignore its use of f. This means the only use of f is in $\textsc{Enc}$ and these uses never repeat inputs because the nonces do not repeat. Hence setting $t \leftarrow f(n, c)$ is equivalent to sampling t uniformly from $\{0,1\}^\tau$ which gives us exactly the view expected in $\mathsf{G}^{\mathsf{ae}\text{-}\diamond}_{\mathsf{EtP[NE,F]},0}$. □

6 Chosen Ciphertext Security of Public Key Encryption

Now we apply our techniques to give memory-tight reductions between single- and multi-challenge notions of chosen-ciphertext security. The standard reduction bounds the advantage of an adversary making $q_{\textsc{Enc}}$ encryption queries by $q_{\textsc{Enc}}$ times the advantage of an adversary making 1 query. The reduction requires

Game $\mathsf{G}^{\mathsf{cca}\text{-}w}_{\mathsf{PKE},b}(\mathcal{A})$	$\text{ENC}_b(m_0, m_1)$	$\text{DEC}^w(c)$

$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.K}$
$b' \leftarrow \mathcal{A}^{\text{ENC}_b,\text{DEC}}(ek)$
Return $b' = 1$
	$// \ \lvert m_0\rvert = \lvert m_1\rvert$
	$c_0 \leftarrow\!\!\$\ \mathsf{PKE.E}(ek, m_0)$
	$c_1 \leftarrow\!\!\$\ \mathsf{PKE.E}(ek, m_1)$
	$M[c_b] \leftarrow m_1$
	Return c_b
		If $M[c] \neq \bot$:
		  Return $M[c]$ if $w = \mathtt{m}$
		  Return $\diamond$ if $w = \diamond$
		  Return $\bot$ if $w = \bot$
		$m \leftarrow \mathsf{PKE.D}(dk, c)$
		Return m

Fig. 19. Game defining CCA-w security of PKE for $w \in \{\mathtt{m}, \diamond, \bot\}$.

memory linear in q_{ENC} and so is not memory-tight.[8] In Section 6.1, we consider the most common "left-or-right" definition of CCA security and introduce three different variants (mirroring the three notions for AE security in Section 5). We give a memory-tight reduction between single- and multi-challenge security for two of the three variants ($\diamond$ and $\bot$), but the reduction is not time-tight. In Section 6.2, we look at the CCA security variant that requires ciphertexts be indistinguishable from random. We give a memory-tight and time-tight reduction between single- and multi-challenge security for all three variants of this notion.

PKE Syntax
$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.K}$ $c \leftarrow\!\!\$\ \mathsf{PKE.E}(ek, m)$ $m \leftarrow \mathsf{PKE.D}(dk, c)$

Fig. 18. Syntax of a public key encryption scheme PKE.

Public key encryption. A public key encryption scheme PKE specifies algorithms PKE.K, PKE.E, and PKE.D. The syntax of these algorithms is shown in Fig. 18. The key generation algorithm PKE.K returns encryption key ek and decryption key dk. The encryption algorithm PKE.E encrypts message m with ek to produce a ciphertext c. We write $\mathsf{PKE.E}(ek, m; r)$ when making random coins $r \in \mathsf{PKE.R}$ explicit. The decryption algorithm decrypts c with dk to produce m. The decryption algorithm may output $m = \bot$ to indicate rejection.

Correctness requires that $\mathsf{PKE.D}(dk, c) = m$ for all $(ek, dk) \in [\mathsf{PKE.K}]$, all m, and all $c \in [\mathsf{PKE.E}(ek, m)]$. We define the min-entropy of PKE as

$$\mathsf{PKE.H}_\infty = -\lg \max_{m,ek,c} \Pr[r \leftarrow\!\!\$\ \mathsf{PKE.R} : \mathsf{PKE.E}(ek, m; r) = c] \ .$$

6.1 Left-or-right CCA Security of PKE

Left-or-right CCA security. In this section, we consider the left-or-right definition of CCA-security most commonly used in the literature. For $w \in \{\mathtt{m}, \diamond, \bot\}$ we denote this as CCA-w[9] and the corresponding security game $\mathsf{G}^{\mathsf{cca}\text{-}w}_{\mathsf{PKE},b}$ is defined in Fig. 19. The adversary gets the encryption key ek and has access to an encryption and a decryption oracle. The encryption oracle takes in messages m_0 and m_1 and encrypts m_b where b is the secret bit. The decryption oracle returns the decryption of a ciphertext, unless the ciphertext was previously returned by an encryption query. This is tracked by table M. When $w = \mathtt{m}$, the decryption oracle returns $M[c]$ which is m_1 from the earlier encryption query. When $w = \diamond$, it returns $\diamond$. When $w = \bot$, it returns $\bot$ which is also used by the encryption scheme to represent rejection. The advantage of an adversary $\mathcal{A}$ against the CCA-w security of PKE is defined as $\mathsf{Adv}^{\mathsf{cca}\text{-}w}_{\mathsf{PKE}}(\mathcal{A}) = \Pr[\mathsf{G}^{\mathsf{cca}\text{-}w}_{\mathsf{PKE},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\mathsf{cca}\text{-}w}_{\mathsf{PKE},0}(\mathcal{A})]$.

The goal of this section is to relate the advantage of attacks making only a single encryption query and those making many such queries. When wanting to make the distinction explicit we may use the adjectives "many" and "single" or prefix the abbreviation of a security notion with an 'm' or '1'.

[8] Auerbach et al. [1] stated that this reduction is memory-tight for both CPA and CCA security. While the former is correct, the latter depends on the definition of CCA. In personally communication with Auerbach et al. [2], they concurred that their claim was incorrect for their intended definition of CCA security ($w = \diamond$) but pointed out that it does work for an "exclusion" variant, $w = E$, which we will mention briefly.

[9] The discussion in Section 5 about the choice to have three variants of the definitions is applicable here as well.

Adversary $\mathcal{A}_1^{\text{Enc},\text{Dec}}(ek)$

$k \leftarrow_\$ [q_{\text{Enc}}]$
$i \leftarrow 0$
$D_{(\cdot)} \leftarrow \{0,1\}^{(\cdot)} \times [q_{\text{Enc}}]$
$f \leftarrow_\$ \mathsf{Fcs}(\mathbb{N}, D, \mathsf{PKE.R})$
$b' \leftarrow \mathcal{A}_{\mathsf{m}}^{\text{SimEnc},\text{SimDec}}(ek)$
Return b'

$\text{SimEnc}(m_0, m_1)$

$i \leftarrow i + 1$
For $d \in \{0,1\}$:
  $r_d \leftarrow f(|m_d|, (m_d, i))$
  $c_d \leftarrow \mathsf{PKE.E}(ek, m_d; r_d)$
If $i < k$: $c \leftarrow c_1$
If $i = k$:
  $c \leftarrow \text{Enc}(m_0, m_1)$
  $c^* \leftarrow c$
  $(m_0^*, m_1^*) \leftarrow (m_0, m_1)$
If $i > k$: $c \leftarrow c_0$
Return c

$\text{SimDec}(c)$

If $c = c^*$: Return $\diamond$
$m \leftarrow \text{Dec}(c)$
If $m = \bot$: Return $\bot$
For $j \in [i]$ do:
  If $m \in \{m_0^*, m_1^*\}$ and $j = k$:
    Skip to next j
  $r \leftarrow f(|m|, (m, j))$
  If $\mathsf{PKE.E}(ek, m; r) = c$:
    Return $\diamond$
Return m

Fig. 20. Adversary $\mathcal{A}_1$ for Theorem 6.

1CCA-$\diamond$ IMPLIES mCCA-$\diamond$. The following theorem gives a memory-tight reduction establishing that CCA-$\diamond$ security against adversaries making one encryption query implies security for an arbitrary number of queries. The proof makes use of our inefficient tagging technique. The reduction performs a hybrid over the encryption queries of the original adversary and is thus not advantage-tight.

Theorem 6 (1CCA-$\diamond$ $\Rightarrow$ mCCA-$\diamond$)**.** *Let* PKE *be a public key encryption scheme. Let* $\mathcal{A}_{\mathsf{m}}$ *be an adversary with* $(q_{\text{Enc}}, q_{\text{Dec}}) = \textbf{Query}(\mathcal{A}_{\mathsf{m}})$*. Define* $D_{(\cdot)}$ *by* $D_n = \{0,1\}^n \times [q_{\text{Enc}}]$*. Let* $\mathcal{A}_1$ *be the* $\mathsf{Fcs}(\mathbb{N}, D, \mathsf{PKE.R})$*-oracle adversary shown in Fig. 20. Then,*

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{cca}\text{-}\diamond}_{\mathsf{PKE}}(\mathcal{A}_{\mathsf{m}}) \leqslant q_{\text{Enc}} \cdot \mathsf{Adv}^{\mathsf{cca}\text{-}\diamond}_{\mathsf{PKE}}(\mathcal{A}_1) + 4 \cdot q_{\text{Enc}} \cdot q_{\text{Dec}} / 2^{\mathsf{PKE.H}_\infty} \\
&\textbf{Query}(\mathcal{A}_1) = (1, q_{\text{Dec}}) \\
&\textbf{Time}^*(\mathcal{A}_1) = O(\textbf{Time}(\mathcal{A}_{\mathsf{m}})) + q_{\text{Enc}}(q_{\text{Dec}} + 1)\textbf{Time}(\mathsf{PKE}) \\
&\textbf{Mem}^*(\mathcal{A}_1) = O(\textbf{Mem}(\mathcal{A}_{\mathsf{m}})) + \textbf{Mem}(\mathsf{PKE}) + \lg q_{\text{Enc}}.
\end{aligned}$$

The standard (non-memory-tight) reduction against 1CCA security picks an index $k \in [q_{\text{Enc}}]$ where q_{Enc} is the number of encryption queries made by $\mathcal{A}_{\mathsf{m}}$. It runs $\mathcal{A}_{\mathsf{m}}$, simulating encryption queries as follows. For the first $k-1$ encryption queries, it answers with an encryption of m_1, for the k-th encryption query it forwards the query to its own encryption oracle, and the rest of the queries it answers with an encryption of m_0. To answer the decryption queries, the reduction returns $\diamond$ if it was ever queried the ciphertext for a previous encryption query. Otherwise, it forwards the query to its own decryption oracle. Finally, the reduction adversary outputs whatever $\mathcal{A}_{\mathsf{m}}$ outputs. Standard hybrid analysis shows that if the advantage of $\mathcal{A}_{\mathsf{m}}$ is ϵ, then the advantage of the reduction adversary is $\epsilon / q_{\text{Enc}}$. Simulating decryption queries required remembering all prior encryption queries and hence the reduction is not memory-tight.

We give an adversary $\mathcal{A}_1$ in Fig. 20 that is very similar to the reduction just described, but avoids remembering prior encryption queries. The main idea is that it picks the coins when encrypting m_0 or m_1 locally as the output of a random function f applied to the message and a counter. This allows $\mathcal{A}_1$ to detect whether a ciphertext c queried to the decryption oracle is one it answered to an earlier encryption query as follows: it first asks for the decryption of c from its own decryption oracle and receives m. Then it iterates over all counter values for which encryption queries have been made so far and checks if c was the encryption of m using the output of f on m and the counter as coins. If any of these checks succeed it returns $\diamond$, otherwise it returns m. If c was the answer of an encryption query $\mathcal{A}_1$ detects it successfully. The probability that $\mathcal{A}_1$ returns $\diamond$ for a decryption query when it should not is small.

Notice that the additional memory overhead for $\mathcal{A}_1$ is just that required to store a counter, run $\mathsf{PKE.E}$, and store (c^*, m_0^*, m_1^*). However, there is an *increase* in runtime by $q_{\text{Enc}} \cdot q_{\text{Dec}} \cdot \textbf{Time}(\mathsf{PKE})$ because of the iteration over the counters to answer decryption queries. As discussed in the introduction, such reductions may be useful when the best attack for the underlying problem with low memory requires significantly more running time than the best attack with high memory.

Extension to CCA-$\perp$. We can prove the same result for CCA-$\perp$, using a very similar proof flow. Alternatively, Theorem 6 directly implies the same result for CCA-$\perp$. First off, 1CCA-$\perp$ implies 1CCA-$\diamond$ in a memory-tight way because an adversary with access to $\text{Dec}^{\perp}$ can simulate $\text{Dec}^{\diamond}$ by just remembering the ciphertext c^* returned for the single Enc query, returning $\diamond$ if c^* is queried to Dec, and otherwise forwarding the response of its own decryption oracle. We also noted above that mCCA-$\diamond$ implies mCCA-$\perp$ in a memory-tight way. Putting it together with Theorem 6 gives the desired result.

However, it does not seem possible to extend this proof technique to CCA-$\mathtt{m}$ security because if the adversary queries the decryption oracle on a ciphertext c which was an answer to a previous query for (m_0, m_1) the oracle needs to return m_1 even if c is an encryption of m_0. This seems to require memory to simulate.

Other variants (exclusion and penalty). In personal communication with Auerbach et al. [2] they pointed out two other variants of CCA security which were given in [3]. For the sake of concreteness we will described them as based on CCA-$\perp$, but the way they are defined means we could just as well have started from CCA-$\mathtt{m}$ or CCA-$\diamond$. The first, which we will refer to as CCA-E (where 'E' stands for "exclusion") is defined the same as CCA-$\perp$ except we require security only for adversaries that will *never* make a decryption query $\text{Dec}(c)$ if c was ever returned by a prior encryption query. The standard hybrid argument is memory-tight for CCA-E. This follows immediately because we do not have to simulate decryption for forwarded ciphertexts.

For the second, which we will refer to as CCA-P (where 'P' stands for "penalty"), we can think of the adversary as being penalized at the end of the game if it even makes a decryption query $\text{Dec}(c)$ if c was ever returned by a prior encryption query. Let $\mathsf{G}^{\text{cca-P}}_{\mathsf{PKE},1}$ be identical to $\mathsf{G}^{\text{cca-}\perp}_{\mathsf{PKE},1}$, except the game returns false no matter what b' is if the adversary ever made such a query. Similarly, let $\mathsf{G}^{\text{cca-P}}_{\mathsf{PKE},0}$ be identical to $\mathsf{G}^{\text{cca-}\perp}_{\mathsf{PKE},0}$, except the game returns true no matter what b' is if the adversary ever made such a query. Then we define $\mathsf{Adv}^{\text{cca-P}}_{\mathsf{PKE}}(\mathcal{A}) = \Pr[\mathsf{G}^{\text{cca-P}}_{\mathsf{PKE},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\text{cca-P}}_{\mathsf{PKE},0}(\mathcal{A})]$. It is not clear how to write a memory-tight hybrid argument for CCA-P.

The philosophical and technical arguments from Sec. 5 for why $w = \mathtt{m}$ may be "correct" apply similarly to argue in favor of it over $w = \mathrm{E}$ and $w = \mathrm{P}$. Additionally, $w = \mathrm{E}$ seems particularly "weak" because it seems overly restrictive. Consider a low-memory attacker that has made a large number of encryption queries so far. It will be incredibly restricted in what decryption queries it can make because it is required to absolutely avoid any query that has a non-zero chance of being a ciphertext returned by a prior encryption given its current state. Note that even very naive adversaries are excluded, for example one that asks a random string of appropriate format to the decryption oracle after seeing some challenge ciphertexts.

Time-tightness if messages do not repeat. If we require that $\mathcal{A}_{\mathsf{m}}$ never repeats messages queried to Enc then we can make Theorem 6 time-tight as well. In that case, $\mathcal{A}_1$ would not need to use the counter i to ensure domain separation for f and so it would not have to use the loop inside SimDec. One setting where this suffices is if PKE is being used as a key-encapsulation mechanism. Then we can think of m_0 and m_1 being picked uniformly at random. Using the switching lemma we can switch to m_0 and m_1 being sampled without replacement, meaning the encryption queries do not repeat.[10] We next give the formal proof of Theorem 6.

Proof. We start by considering the hybrid games H^b_h for $0 \leqslant h \leqslant 2$ defined in Fig. 21. Of these we make the following claims for $b \in \{0, 1\}$.

1. $\Pr[\mathsf{G}^{\text{cca-}\diamond}_{\mathsf{PKE},b}] = \Pr[\mathsf{H}^b_0]$
2. $\Pr[\mathsf{H}^b_0] = \Pr[\mathsf{H}^b_1]$
3. $|\Pr[\mathsf{H}^b_1] - \Pr[\mathsf{H}^b_2]| \leqslant q_{\text{Enc}} \cdot q_{\text{Dec}} \cdot 2^{-\mathsf{PKE.H}_\infty}$

Combining the above claims, we get that

$$\begin{aligned}\mathsf{Adv}^{\text{cca-}\diamond}_{\mathsf{PKE}}(\mathcal{A}_{\mathsf{m}}) &= \Pr[\mathsf{G}^{\text{cca-}\diamond}_{\mathsf{PKE},1}] - \Pr[\mathsf{G}^{\text{cca-}\diamond}_{\mathsf{PKE},0}] = \Pr[\mathsf{H}^1_0] - \Pr[\mathsf{H}^0_0] \\ &= \Pr[\mathsf{H}^1_1] - \Pr[\mathsf{H}^1_0] \leqslant \Pr[\mathsf{H}^1_2] - \Pr[\mathsf{H}^0_2] + 2 \cdot q_{\text{Enc}} \cdot q_{\text{Dec}} \cdot 2^{-\mathsf{PKE.H}_\infty}. \end{aligned} \quad (1)$$

[10] Here the list of m_0 and m_1 to be queried can be specified by an oracle given to the adversary.

Games H^b_h for $0 \leqslant h \leqslant 2$	$\text{ENC}_b(m_0, m_1)$	$\text{DEC}(c)$
$(ek, dk) \leftarrow\$ \mathsf{PKE.K}$	$i \leftarrow i + 1$	If $M[c] \neq \bot$ then $//\mathsf{H}^b_{[0,2)}$
$i \leftarrow 0$	For $d \in \{0, 1\}$:	Return $\diamond$ $//\mathsf{H}^b_{[0,2)}$
$D_{(\cdot)} \leftarrow \{0,1\}^{(\cdot)} \times [q_{\text{ENC}}]$ $//\mathsf{H}^b_{[1,\infty)}$	$r_d \leftarrow\$ \mathsf{PKE.R}$ $//\mathsf{H}^b_{[0,1)}$	$m \leftarrow \mathsf{PKE.D}(dk, c)$
$f \leftarrow\$ \mathsf{Fcs}(\mathbb{N}, D, \mathsf{PKE.R})$ $//\mathsf{H}^b_{[1,\infty)}$	$r_d \leftarrow f(\lvert m_d \rvert, (m_d, i))$ $//\mathsf{H}^b_{[1,\infty)}$	For $j \in [i]$: $//\mathsf{H}^b_{[2,\infty)}$
$b' \leftarrow \mathcal{A}_{\mathsf{m}}^{\text{ENC}_b, \text{DEC}}(ek)$	$c_d \leftarrow \mathsf{PKE.E}(ek, m_d; r_d)$	$r \leftarrow f(\lvert m \rvert, (m, j))$ $//\mathsf{H}^b_{[2,\infty)}$
Return $b' = 1$	$M[c_b] \leftarrow m_b$ $//\mathsf{H}^b_{[0,2)}$	If $c = \mathsf{PKE.E}(ek, m; r)$: $//\mathsf{H}^b_{[2,\infty)}$
	Return c_b	Return $\diamond$ $//\mathsf{H}^b_{[2,\infty)}$
		Return m

Fig. 21. First set of hybrids H^b_h used in the proof of Theorem 6.

Hybrids $\mathsf{H}^{k,b}$	$\text{ENC}_b(m_0, m_1)$	$\text{DEC}(c)$
$//(k, b) \in [q_{\text{ENC}}] \times \{0, 1\}$	$i \leftarrow i + 1$	$m \leftarrow \mathsf{PKE.D}(dk, c)$
$(ek, dk) \leftarrow\$ \mathsf{PKE.K}$	For $d \in \{0, 1\}$:	For $j \in [i]$:
$i \leftarrow 0$	$r_d \leftarrow f(\lvert m_d \rvert, (m_d, i))$	$r \leftarrow f(\lvert m \rvert, (m, j))$
$D_{(\cdot)} \leftarrow \{0,1\}^{(\cdot)} \times [q_{\text{ENC}}]$	$c_d \leftarrow \mathsf{PKE.E}(ek, m_d; r_d)$	If $c = \mathsf{PKE.E}(ek, m; r)$:
$f \leftarrow\$ \mathsf{Fcs}(\mathbb{N}, D, \mathsf{PKE.R})$	If $i < k$: $c \leftarrow c_1$	Return $\diamond$
$b' \leftarrow \mathcal{A}_{\mathsf{m}}^{\text{ENC}_b, \text{DEC}}(ek)$	If $i = k$: $c \leftarrow c_b$	Return m
Return $b' = 1$	If $i > k$: $c \leftarrow c_0$	
	Return c	

Fig. 22. Second set of hybrids used in the proof of Theorem 6.

Next, we prove the claims.

TRANSITION TO H^b_0. The game H^b_0 was copied from the game $\mathsf{G}^{\mathsf{cca}\text{-}\diamond}_{\mathsf{PKE},b}$. We added variable i that counts the number of ENC queries and will be used for future hybrids and unrolled the encryption to make the sampling of coins explicit. It follows that $\Pr[\mathsf{G}^{\mathsf{cca}\text{-}\diamond}_{\mathsf{PKE},b}] = \Pr[\mathsf{H}^b_0]$.

TRANSITION FROM H^b_0 TO H^b_1. In game H^b_1, we replace the random sampling of r_0 and r_1 with the output of a random function f, using a counter i to provide domain separation between different queries. This method of choosing r is equivalent, so $\Pr[\mathsf{H}^b_0] = \Pr[\mathsf{H}^b_1]$.

TRANSITION FROM H^b_1 TO H^b_2. In game H^b_2, we stop using $M[\cdot]$ to keep track ciphertexts that were returned by ENC. Instead we first decrypt c to m and then iterating over $j \in [i]$ to check whether m encrypted with ek using randomness $r = f(|m|, (m, j))$ is c. Note that if $M[c] \neq \bot$ holds in H^b_1 then there will necessarily be such a j (in particular j being the value i held at the time of the query that set $M[c]$).

So H^b_1 and H^b_2 are identical unless the following bad event happens: there is a DEC query on c and for some $j \in [j]$, it holds that $c = \mathsf{PKE.E}(ek, m; f(|m|, (m, j)))$ and despite m was not m_b for the j-th query to ENC. We can analyze the probability of this in H^b_1. Since m was not the j-th query to ENC, the view of the adversary was independent of $f(|m|, (m, j))$ at this time. By the min-entropy of PKE, the probability of this occurring for a given decryption query and j is at most $1/2^{\mathsf{PKE.H}_\infty}$. Taking a union bound over all $j \in [q_{\text{ENC}}]$ and decryption queries the overall probability is at most $q_{\text{ENC}} \cdot q_{\text{DEC}}/2^{\mathsf{PKE.H}_\infty}$. We could formalize this via the Fundamental Lemma of Game Playing [6], to get $|\Pr[\mathsf{H}^b_1] - \Pr[\mathsf{H}^b_2]| \leqslant q_{\text{ENC}} \cdot q_{\text{DEC}}/2^{\mathsf{PKE.H}_\infty}$ as desired.

TRANSITION TO $\mathsf{H}^{k,b}$ HYBRIDS. We next consider the hybrid games $\mathsf{H}^{k,b}$ for $(k, b) \in [q_{\text{ENC}}] \times \{0, 1\}$ defined in Fig. 22. In these hybrids, ciphertexts for m_1 are returned when $i < k$, ciphertexts for m_0 are returned when $i > k$, and a ciphertext for m_b is returned when $i = k$. Note that at the extremes ($k = 1, b = 0$ and $k = q_{\text{ENC}}, b = 1$), $\mathcal{A}_{\mathsf{m}}$ will receive either all ciphertexts of m_0 or all ciphertexts of m_1. The decryption queries are answered as in H^b_2. So $\mathsf{H}^{1,0}$ perfectly matches H^0_2 and $\mathsf{H}^{q_{\text{ENC}},1}$ perfectly matches H^1_2. Hence combining with

Hybrids H_h for $0 \leqslant h \leqslant 2$
$(k,b) \leftarrow\!\!\$\ [q_{\text{Enc}}] \times \{0,1\}$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.K}$
$i \leftarrow 0$
$D_{(\cdot)} \leftarrow \{0,1\}^{(\cdot)} \times [q_{\text{Enc}}]$
$f \leftarrow\!\!\$\ \mathsf{Fcs}(\mathbb{N}, D, \mathsf{PKE.R})$
$b' \leftarrow \mathcal{A}_{\mathsf{m}}^{\text{Enc}_b, \text{Dec}}(ek)$
Return $b' = b$

$\text{Dec}(c)$
If $c = c^*$: Return $\diamond$ $/\!/\mathsf{H}_{[1,\infty)}$
$m \leftarrow \mathsf{PKE.D}(dk, c)$
For $j \in [i]$ do:
  If $m \in \{m_0^*, m_1^*\}$ and $j = k$: $/\!/\mathsf{H}_{[1,\infty)}$
    Skip to next j $/\!/\mathsf{H}_{[1,\infty)}$
  $r \leftarrow f(|m|, (m, j))$
  If $\mathsf{PKE.E}(ek, m; r) = c$:
    Return $\diamond$
Return m

$\text{Enc}_b(m_0, m_1)$
$i \leftarrow i + 1$
For $d \in \{0,1\}$:
  $r_d \leftarrow f(|m_d|, (m_d, i))$
  $c_d \leftarrow \mathsf{PKE.E}(ek, m_d; r_d)$
If $i < k$:
  $c \leftarrow c_1$
If $i = k$:
  $c \leftarrow c_b$ $/\!/\mathsf{H}_{[0,1)}$
  $r \leftarrow f(|m_b|, (m_b, i))$ $/\!/\mathsf{H}_{[1,2)}$
  $r \leftarrow\!\!\$\ \mathsf{PKE.R}$ $/\!/\mathsf{H}_{[2,\infty)}$
  $c \leftarrow \mathsf{PKE.E}(ek, m_b; r)$ $/\!/\mathsf{H}_{[1,\infty)}$
  $c^* \leftarrow c$ $/\!/\mathsf{H}_{[1,\infty)}$
  $(m_0^*, m_1^*) \leftarrow (m_0, m_1)$ $/\!/\mathsf{H}_{[1,\infty)}$
If $i > k$:
  $c \leftarrow c_0$
Return c

Fig. 23. Final set of hybrids for the proof of Theorem 6.

---

(1), we have

$$\mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{cca}\text{-}\diamond}(\mathcal{A}_{\mathsf{m}}) \leqslant \Pr[\mathsf{H}^{q_{\text{Enc}},1}] - \Pr[\mathsf{H}^{1,0}] + 2 \cdot q_{\text{Enc}} \cdot q_{\text{Dec}} \cdot 2^{-\mathsf{PKE.H}_\infty}\ . \tag{2}$$

In general, in $\mathsf{H}^{k,b}$ the first $k+b-1$ encryption queries use m_1 and the rest use m_0; so $\Pr[\mathsf{H}^{k,1}] = \Pr[\mathsf{H}^{k+1,0}]$ holds. Hence

$$\begin{aligned}\Pr[\mathsf{H}^{q_{\text{Enc}},1}] - \Pr[\mathsf{H}^{1,0}] &= \Pr[\mathsf{H}^{q_{\text{Enc}},1}] - \Pr[\mathsf{H}^{1,0}] + \sum_{k \in [q_{\text{Enc}}-1]} \Pr[\mathsf{H}^{k,1}] - \Pr[\mathsf{H}^{k+1,0}] \\ &= \sum_{k \in [q_{\text{Enc}}]} \Pr[\mathsf{H}^{k,1}] - \Pr[\mathsf{H}^{k,0}]. \end{aligned} \tag{3}$$

Transition to H_h hybrids. Next, consider the games shown in Fig. 28. Of these we make the following claims.

1. $2\Pr[\mathsf{H}_0] - 1 = (1/q_{\text{Enc}}) \sum_{k \in [q_{\text{Enc}}]} \Pr[\mathsf{H}^{k,1}] - \Pr[\mathsf{H}^{k,0}]$
2. $\Pr[\mathsf{H}_0] \leqslant \Pr[\mathsf{H}_1] + q_{\text{Dec}} \cdot 2^{-\mathsf{PKE.H}_\infty}$
3. $\Pr[\mathsf{H}_1] = \Pr[\mathsf{H}_2]$
4. $2\Pr[\mathsf{H}_2] - 1 = \mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{cca}\text{-}\diamond}(\mathcal{A}_1)$

In the rest of the proof we address these one at a time. Putting them together along with (2) and (3) gives the bound claimed in the theorem statement via,

$$\begin{aligned}\mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{cca}\text{-}\diamond}(\mathcal{A}_{\mathsf{m}}) &= 2 \cdot q_{\text{Enc}} \cdot q_{\text{Dec}} \cdot 2^{-\mathsf{PKE.H}_\infty} + \sum_{k \in [q_{\text{Enc}}]} \Pr[\mathsf{H}^{k,1}] - \Pr[\mathsf{H}^{k,0}] \\ &= 2 \cdot q_{\text{Enc}} \cdot q_{\text{Dec}} \cdot 2^{-\mathsf{PKE.H}_\infty} + q_{\text{Enc}}(2\Pr[\mathsf{H}_0] - 1) \\ &\leqslant 2 \cdot q_{\text{Enc}} \cdot q_{\text{Dec}} \cdot 2^{-\mathsf{PKE.H}_\infty} + q_{\text{Enc}}(2\Pr[\mathsf{H}_2] + 2q_{\text{Dec}} \cdot 2^{-\mathsf{PKE.H}_\infty} - 1) \\ &= \mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{cca}\text{-}\diamond}(\mathcal{A}_1) + 4q_{\text{Enc}} \cdot q_{\text{Dec}} \cdot 2^{-\mathsf{PKE.H}_\infty}.\end{aligned}$$

Transition to H_0. Game H_0 is identical to $\mathsf{H}^{k,b}$ with (k,b) chosen at random and with the game returning true if $\mathcal{A}_{\mathsf{m}}$ correctly guessed b. Standard calculations by conditioning on all the possible values of (k,b) gives the claim.

Game $\mathsf{G}^{\$\mathsf{cca}\text{-}w}_{\mathsf{PKE},b}(\mathcal{A})$	$\textsc{Enc}_b(m)$	$\textsc{Dec}^w(c)$

$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.K}$
$b' \leftarrow \mathcal{A}^{\textsc{Enc}_b, \textsc{Dec}^w}(ek)$
Return $b' = 1$
	$c_1 \leftarrow\!\!\$\ \mathsf{PKE.E}(ek, m)$
	$c_0 \leftarrow\!\!\$\ \mathsf{PKE.C}(ek, \lvert m\rvert)$
	$M[c_b] \leftarrow m$
	Return c_b
		If $M[c] \neq \bot$:
		  Return $M[c]$ if $w = \mathtt{m}$
		  Return $\diamond$ if $w = \diamond$
		  Return $\bot$ if $w = \bot$
		$m \leftarrow \mathsf{PKE.D}(dk, c)$
		Return m

Fig. 24. Game defining \$CCA-w security of PKE for $w \in \{\mathtt{m}, \diamond, \bot\}$.

Transition from H_0 to H_1. We make three changes to transition to H_1. First, when $i = k$ in Enc we store the two messages queried as m_0^* and m_1^* along with the ciphertext returned as c^*. Then on a Dec query, $\diamond$ is returned immediately if $c = c^*$. Finally in Dec we skip over the iteration of $j = k$ if $m \in \{m_0^*, m_1^*\}$. When $m = m_b^*$ this does not change anything because the $c = c^*$ check will have covered that case.

So this only changes the behavior of the oracle when $m = m_{1-b}^* \neq m_b^*$ and $c = \mathsf{PKE.E}(ek, m; f(|m|, (m, k)))$, in H_0 it would return $\diamond$ while in H_1 it may return m. Note then that in H_1 the view of the adversary is completely independent of this $f(|m|, (m, k))$. So we can think of the adversary having Dec attempts to guess a ciphertext generated with uniformly random coins. Hence the probability of this sort of query in H_1 is at most $q_{\textsc{Dec}} \cdot 2^{-\mathsf{PKE.H}_\infty}$. This gives $\Pr[\mathsf{H}_0] \leqslant \Pr[\mathsf{H}_1] + q_{\textsc{Dec}} \cdot 2^{-\mathsf{PKE.H}_\infty}$.

Transition from H_1 to H_2. The only change in H_2 is that in Enc for $i = k$, the randomness is sampled uniformly at random instead of by evaluating f. Because $f(|m_b|, (m_b^*, k))$ is used nowhere else, this does not change the behavior of the game. It follows that $\Pr[\mathsf{H}_1] = \Pr[\mathsf{H}_2]$.

Adversary $\mathcal{A}_1$. Finally we can see that our adversary $\mathcal{A}_1$ (defined in Fig. 20) perfectly simulates the view of $\mathcal{A}_\mathsf{m}$ in H_2. It was obtained by copying the code of H_2 and then modifying it to query its Enc and Dec oracle as appropriate. It follows that $\Pr[\mathsf{H}_2] = 0.5 \Pr[\mathsf{G}^{\mathsf{cca}\text{-}\diamond}_{\mathsf{PKE},1}(\mathcal{A}_1)] + 0.5(1 - \Pr[\mathsf{G}^{\mathsf{cca}\text{-}\diamond}_{\mathsf{PKE},0}(\mathcal{A}_1)])$. Hence, $2\Pr[\mathsf{H}_2] - 1 = \mathsf{Adv}^{\mathsf{cca}\text{-}\diamond}_{\mathsf{PKE}}(\mathcal{A}_1)$.

Adversary $\mathcal{A}_1$'s extra running time comes from using $\mathsf{PKE.E}$ in SimEnc and in the loop in SimDec. Its extra memory is that required for running $\mathsf{PKE.E}$, for storing i, and for storing (c^*, m_0^*, m_1^*). ▫

6.2 Indistinguishable from Random CCA Security of PKE

We saw in the previous section that we could have a memory-tight reduction from mCCA-$\diamond$ to 1CCA-$\diamond$; however, the reduction is not tight with respect to running time. In this section, we show that for a different formalization of CCA security, we can indeed have a memory-tight and time-tight reduction between many- and single-challenge variants.

Ciphertext and encryption key space. Before describing the indistinguishable from random formalization of CCA security, we need to make some assumptions on PKE. We define the encryption keyspace by $\mathsf{PKE.Ek} = \{ek : (ek, dk) \in \mathsf{PKE.K}\}$. We assume for each $ek \in \mathsf{PKE.Ek}$ and allowed message length $n \in \mathbb{N}$ there is a set $\mathsf{PKE.C}(ek, n)$ such that $\mathsf{PKE.E}(ek, m; r) \in \mathsf{PKE.C}(ek, |m|)$ always holds. Let $\mathsf{PKE.C}^{-1}(ek, c)$ returns n such that $c \in \mathsf{PKE.C}(ek, n)$. Correctness implies that $\mathsf{PKE.C}(ek, n)$ and $\mathsf{PKE.C}(ek, n')$ are disjoint for $n \neq n'$.

Indistinguishable from random ciphertext CCA security. The security notion we will consider in this section is captured by the game $\mathsf{G}^{\$\mathsf{cca}\text{-}w}$ shown in Fig. 24. It requires that ciphertexts output by the encryption scheme cannot be distinguished from ciphertexts chosen at random even given access to a decryption oracle. The adversary gets the encryption key ek and has access to an encryption oracle Enc and a decryption oracle Dec. The adversary needs to distinguish the following real and ideal worlds: in the real world, a query to Enc with a message m returns an encryption of m under ek, while in the ideal world, the same query returns a uniformly random element of $\mathsf{PKE.C}(ek, |m|)$. The decryption oracle $\textsc{Dec}^w$ acts exactly

Adversary $\mathcal{A}_1^{\text{Enc},\text{Dec}}(ek)$	$\text{SimEnc}(m)$	$\text{SimDec}(c)$
$//0 \leqslant h \leqslant 2$	$i \leftarrow i+1$	If $c = c^*$: Return m^*
$k \leftarrow\$ [q_{\text{Enc}}]$	$c_1 \leftarrow\$ \mathsf{PKE.E}(ek, m)$	$n \leftarrow \mathsf{PKE.C}^{-1}(ek, c)$
$i \leftarrow 0$	$c_0 \leftarrow f((\lvert m\rvert, ek), (m, i))$	$(m, j) \leftarrow f^{-1}((n, ek), c)$
$f \leftarrow\$ \mathsf{Inj}^{\pm}(T, D, R)$	If $i < k$: $c \leftarrow c_1$	If $m \neq \bot$ and $k \leqslant j \leqslant i$:
$b' \leftarrow \mathcal{A}_{\mathsf{m}}^{\text{SimEnc},\text{SimDec}}(ek)$	If $i = k$:	If $(m, j) = (m^*, k)$:
Return b'	$c \leftarrow \text{Enc}(m)$	Skip next line
	$(c^*, m^*) \leftarrow (c, m)$	Return m
	If $i > k$: $c \leftarrow c_0$	$m \leftarrow \text{Dec}(c)$
	Return c	Return m

Fig. 25. Adversary $\mathcal{A}_1$ for Theorem 7.

as the corresponding oracle in $\mathsf{G}^{\mathsf{cca}\text{-}w}$.[11] The advantage of an adversary $\mathcal{A}$ against the \$CCA-w security of PKE is defined as $\mathsf{Adv}^{\$\mathsf{cca}\text{-}w}_{\mathsf{PKE}}(\mathcal{A}) = \Pr[\mathsf{G}^{\$\mathsf{cca}\text{-}w}_{\mathsf{PKE},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\$\mathsf{cca}\text{-}w}_{\mathsf{PKE},0}(\mathcal{A})]$. If $\mathsf{PKE.E}(ek, m; \cdot)$ is injective, then this is exactly identical to the standard CCA notion for $\mathsf{PKE.C}(ek, n) = \{\mathsf{PKE.E}(ek, m; r) : \lvert m\rvert = n, r \in \mathsf{PKE.R}\}$.

1\$CCA-m IMPLIES m\$CCA-m. The following theorem captures a memory-tight reduction establishing that 1\$CCA-m security implies m\$CCA-m security. The proof makes use of our message encoding technique.

Theorem 7 (1\$CCA-m $\Rightarrow$ m\$CCA-m). *Let* PKE *be a public key encryption scheme. Let τ satisfy $\lvert\mathsf{PKE.C}(ek, n)\rvert \geqslant 2^n \cdot 2^\tau$ for all n, ek. Let $\mathcal{A}_{\mathsf{m}}$ be an adversary with $(q_{\text{Enc}}, q_{\text{Dec}}, q_{\text{h}}) = \mathbf{Query}(\mathcal{A}_{\mathsf{m}})$ and assume $q_{\text{Enc}} + q_{\text{Dec}} \leqslant 0.5 \cdot 2^\tau$. Let $\mathcal{F} = \mathsf{Inj}^{\pm}(T, D, R)$ where T, D, and R are defined by $T = \mathbb{N} \times \mathsf{PKE.Ek}$, $D_{n,ek} = \{0,1\}^n \times [q_{\text{Enc}}]$ and $R_{n,ek} = \mathsf{PKE.C}(ek, n)$. Let $\mathcal{A}_1$ be the $\mathcal{F}$-oracle adversary defined in Fig. 25. Then,*

$$\begin{aligned}
&\mathsf{Adv}^{\$\mathsf{cca}\text{-}m}_{\mathsf{PKE}}(\mathcal{A}_{\mathsf{m}}) \leqslant q_{\text{Enc}} \cdot \mathsf{Adv}^{\$\mathsf{cca}\text{-}m}_{\mathsf{PKE}}(\mathcal{A}_1) + 8q_{\text{Enc}}q_{\text{Dec}}/2^\tau + 5q_{\text{Enc}}^2/2^\tau\\
&\mathbf{Query}(\mathcal{A}_1) = (1, q_{\text{Dec}}, q_{\text{h}})\\
&\mathbf{Time}^*(\mathcal{A}_1) = O(\mathbf{Time}(\mathcal{A}_{\mathsf{m}})) + q_{\text{Enc}}\mathbf{Time}(\mathsf{PKE})\\
&\mathbf{Mem}^*(\mathcal{A}_1) = O(\mathbf{Mem}(\mathcal{A}_{\mathsf{m}})) + \mathbf{Mem}(\mathsf{PKE}) \lg q_{\text{Enc}}.
\end{aligned}$$

The standard (non-memory-tight) reduction against 1\$CCA security that runs an m\$CCA adversary $\mathcal{A}_{\mathsf{m}}$ works in a similar manner as the standard reduction from an 1CCA adversary and an mCCA adversary that we described in Section 6.1. Again here, simulating decryption queries requires remembering all the answers of the encryption queries, and hence the reduction is not memory-tight.

We give an adversary $\mathcal{A}_1$ in Fig. 25 that is very similar to the standard reduction, but avoids remembering all the answers of the encryption queries. The main idea here is picking the ciphertext c_0 as the output of a random injective function f evaluated on the message and a counter, instead of sampling it uniformly at random. This way of picking the c_0 allows $\mathcal{A}_1$ detect whether a ciphertext c queried to the decryption oracle was the answer to an earlier encryption query as follows: it first checks if the inverse of f on the ciphertext is defined (i.e., not $\bot$), it returns the message part of the inverse. Otherwise it asks for the decryption of the ciphertext to its own decryption oracle and returns the answer. Using our assumption on the size of $\mathsf{PKE.C}(ek, n)$, we can argue that except with small probability, $\mathcal{A}_1$ simulates the decryption oracle correctly. The additional memory overhead for $\mathcal{A}_1$ is only a counter. Moreover, there is no increase in the running time of $\mathcal{A}_1$ unlike the adversary in Theorem 6.

[11] As mentioned, the discussion in Section 5 about the three variants definitions is applicable here as well. In Appendix B we give an example where we can prove CCA security of a KEM/DEM scheme in the memory restricted setting, but only if we use the $w = \mathtt{m}$ definition.

Games $\mathsf{H}_h^b(\mathcal{A})$ for $0 \leqslant h \leqslant 3$	$\text{Enc}_b(m)$	$\text{Dec}^{\mathtt{m}}(c)$
$(ek, dk) \leftarrow\$ \mathsf{PKE.K}$	$i \leftarrow i + 1$	If $M[c] \neq \bot$ then $//\mathsf{H}^b_{[0,2)}$
$i \leftarrow 0$	$c_1 \leftarrow\$ \mathsf{PKE.E}(ek, m)$	Return $M[c]$ $//\mathsf{H}^b_{[0,2)}$
$f \leftarrow\$ \mathsf{Inj}^{\pm}(T, D, R)$	$c_0 \leftarrow\$ \mathsf{PKE.C}(ek, \lvert m\rvert)$ $//\mathsf{H}^b_{[0,1)}$	If $b = 0$ and $i \geqslant 1$:
$b' \leftarrow \mathcal{A}^{\text{Enc}_b, \text{Dec}^{\mathtt{m}}}(ek)$	$c_0 \leftarrow f((\lvert m\rvert, ek), (m, i))$ $//\mathsf{H}^b_{[1,\infty)}$	$n \leftarrow \mathsf{PKE.C}^{-1}(ek, c)$
Return $b' = 1$	$M[c_b] \leftarrow m$	$(m, j) \leftarrow f^{-1}((n, ek), c)$
	Return c_b	If $m \neq \bot$ and $j \leqslant i$:
		If $M[c] = \bot$ then:
		$\mathsf{bad} \leftarrow \mathsf{true}$
		Return m $//\mathsf{H}^b_{[3,\infty)}$
		Else
		Return m
		$m \leftarrow \mathsf{PKE.D}(dk, c)$
		Return m

Fig. 26. First set of hybrids used for proof of Theorem 7. Highlighting indicates modifications in H_0^b from $\mathsf{G}^{\$\mathsf{cca}\text{-}\mathtt{m}}_{\mathsf{PKE},b}$.

Extension to \$CCA-$\diamond$, \$CCA-$\bot$. We can prove the same result for \$CCA-$\diamond$, \$CCA-$\bot$ but the adversary would not be tight with respect to running time. The adversary in these cases would pick the coins for encrypting m (to compute c_1) like the adversary in Theorem 6. This would require iterating over counters to answer decryption queries and hence lead to looseness with respect to running time. We omit the theorems for these notions because they would not involve any new ideas beyond those presented in Theorems 6 and 7.

We give the formal proof of Theorem 7.

Proof. We start by considering the hybrid games H_h^b defined in Fig. 26. In this and future games we define T, D, and R by $T = \mathbb{N} \times \mathsf{PKE.Ek}$, $D_{n,ek} = \{0,1\}^n \times [q_{\text{Enc}}]$ and $R_{n,ek} = \mathsf{PKE.C}(n, ek)$ Of these we make the following claims for $b \in \{0, 1\}$.

1. $\Pr[\mathsf{G}^{\$\mathsf{cca}\text{-}\mathtt{m}}_{\mathsf{PKE},b}] = \Pr[\mathsf{H}_0^b]$
2. $\lvert \Pr[\mathsf{H}_0^b] - \Pr[\mathsf{H}_1^b] \rvert \leqslant 0.5 \cdot q_{\text{Enc}}^2 / 2^\tau$
3. $\Pr[\mathsf{H}_1^b] = \Pr[\mathsf{H}_2^b]$
4. $\lvert \Pr[\mathsf{H}_2^b] - \Pr[\mathsf{H}_3^b] \rvert \leqslant 2 q_{\text{Dec}} q_{\text{Enc}} / 2^\tau$

Combining the above claims, we get that

$$\begin{aligned} \mathsf{Adv}^{\$\mathsf{cca}\text{-}\mathtt{m}}_{\mathsf{PKE}}(\mathcal{A}_{\mathsf{m}}) &= \Pr[\mathsf{G}^{\$\mathsf{cca}\text{-}\mathtt{m}}_{\mathsf{PKE},1}] - \Pr[\mathsf{G}^{\$\mathsf{cca}\text{-}\mathtt{m}}_{\mathsf{PKE},0}] = \Pr[\mathsf{H}_0^1] - \Pr[\mathsf{H}_0^0] \\ &\leqslant \Pr[\mathsf{H}_3^1] - \Pr[\mathsf{H}_3^0] + q_{\text{Enc}}^2/2^\tau + 4 q_{\text{Enc}} q_{\text{Dec}}/2^\tau. \end{aligned} \quad (4)$$

Next, we prove the claims.

Transition to H_0^b. The game H_0^b was copied from the game $\mathsf{G}^{\$\mathsf{cca}\text{-}\mathtt{m}}_{\mathsf{PKE},b}$ with some code added that has been highlighted (note that the entire code inside the highlighted if statement is new). We added a variable i that counts the number of Enc queries and sample a random injective function f that we will be used for future hybrids. We add an if statement in Dec that checks if $b = 0$ and $i \geqslant 1$ and if true it computes $(m, j) \leftarrow f^{-1}((n, ek), c)$ where $n \leftarrow \mathsf{PKE.C}^{-1}(ek, c)$ and then if the check $m \neq \bot$ and $j \leqslant i$ succeeds, it sets a flag bad, otherwise returns m. We note that in H_0^b, the return statement never occurs because if $M[c] \neq \bot$, we would have returned before the execution of this if statement. Hence the code inside the highlighted if statement can be ignored in H_0^b. Therefore, there is no change in behavior in H_0^b compared to $\mathsf{G}^{\$\mathsf{cca}\text{-}\mathtt{m}}_{\mathsf{PKE},b}$. It follows that $\Pr[\mathsf{G}^{\$\mathsf{cca}\text{-}\mathtt{m}}_{\mathsf{PKE},b}] = \Pr[\mathsf{H}_0^b]$.

Hybrids $\mathsf{H}^{k,b}$	$\text{Enc}(m)$	$\text{Dec}(c)$
$//(k,b) \in [q_{\text{Enc}}] \times \{0,1\}$	$i \leftarrow i+1$	If $i \geqslant k+b$:
$i \leftarrow 0$	$c_1 \leftarrow\!\!\$\ \mathsf{PKE.E}(ek, m)$	$n \leftarrow \mathsf{PKE.C}^{-1}(ek, c)$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.K}$	$c_0 \leftarrow f((\lvert m\rvert, ek), (m, i))$	$(m, j) \leftarrow f^{-1}((n, ek), c)$
$f \leftarrow\!\!\$\ \mathsf{Inj}(T, D, R)$	If $i < k$: $c \leftarrow c_1$	If $m \neq \perp$ and $k+b \leqslant j \leqslant i$:
$b' \leftarrow \mathcal{A}_{\mathsf{m}}^{\text{Enc},\text{Dec}}(ek)$	If $i = k$: $c \leftarrow c_b$	Return m
Return $b' = 1$	If $i > k$: $c \leftarrow c_0$	$m \leftarrow \mathsf{PKE.D}(dk, c)$
	Return c	Return m

Fig. 27. Second set of hybrids used for proof of Theorem 7. Highlighting indicates modifications from H_3^b.

Transition from H_0^b to H_1^b. In game H_1^b, we replace the random sampling of c_0 with the output of the random injective function f, using the counter i to provide domain separation between different queries. We again note that in H_1^b, we would never return anything from inside the highlighted if statement, and hence the code inside it can be ignored. In particular that means the behavior of Dec is independent of f. So this modification in how we compute c_0 changes behavior only in Enc since the values of c_0 will never repeat in H_1^b unlike H_0^b. Hence, the switching lemma (Lemma 1) gives us $|\Pr[\mathsf{H}_0^b] - \Pr[\mathsf{H}_1^b]| \leqslant 0.5 \cdot q_{\text{Enc}}^2/2^\tau$ (since the image of f always has size at least 2^τ).

Transition from H_1^b to H_2^b. In Dec of game H_2^b, we stop returning $M[c]$ if it is not $\perp$ at the beginning. If $b = 1$ the behavior remains the same as the highlighted if statement fails and we return $m = \mathsf{PKE.D}(dk, c)$ which we would have returned in H_1^1 (not if $M[c] \neq \perp$ then $M[c] = \mathsf{PKE.D}(dk, c)$ holds in this game). When $b = 0$ and $M[c] \neq \perp$, then both H_1^0 and H_2^0 return $M[c] = f^{-1}((n, ek), c)$. If $b = 0$ and $M[c] = \perp$, then both H_1^b and H_2^b return $m = \mathsf{PKE.D}(dk, c)$. Hence the behavior of Dec is identical in H_1^b and H_2^b. So, $\Pr[\mathsf{H}_1^b] = \Pr[\mathsf{H}_2^b]$.

Transition from H_2^b to H_3^b. In game H_3^b, we return m if the bad flag gets set. Note that for a Dec query on c, bad is set in H_2 only if $M[c] = \perp$ and $f^{-1}((n, ek), c) \neq \perp$. This has no effect when $b = 1$. The probability that in H_2^0 that a given Dec query has a c such that $M[c] = \perp$ and $f^{-1}((n, ek), c) \neq \perp$ is at most $q_{\text{Enc}} 2^n/(2^{n+\tau} - q_{\text{Enc}}) \leqslant 2q_{\text{Enc}}/2^\tau$. This follows because there are $q_{\text{Enc}} \cdot 2^n$ values in the domain (and hence image) of f and the view of the adversary in H_2^b is dependent only on q_{Enc} points of f which are mapped to c' satisfying $M[c'] \neq \perp$ (i.e. those returned by Enc). Taking a union bound over all Dec queries, we get that bad is set with probability at most $2q_{\text{Enc}} \cdot q_{\text{Dec}}/2^\tau$. Since H_2^b and H_3^b are identical-until-bad, using the Fundamental Lemma of Game Playing [6], we get, for $b \in \{0,1\}$, $|\Pr[\mathsf{H}_2^b] - \Pr[\mathsf{H}_3^b]| \leqslant 2q_{\text{Enc}} \cdot q_{\text{Dec}}/2^\tau$.

Transition to $\mathsf{H}^{k,b}$ hybrids. We next consider the hybrid games $\mathsf{H}^{k,b}$ for $(k,b) \in [q_{\text{Enc}}] \times \{0,1\}$ defined in Fig. 27 which have been derived by cleaning up (removing M and bad) and modifying the code of H_3^b. The modified code has been highlighted in Fig. 27. In Enc of $\mathsf{H}^{k,b}$, ciphertexts c_0 and c_1 are computed as in H_3^b, but $\mathsf{H}^{k,b}$ returns c_1 when $i < k$, c_0 when $i > k$, and c_b when $i = k$. Note that at the extremes ($k = 1, b = 0$ and $k = q_{\text{Enc}}, b = 1$) $\mathcal{A}_{\mathsf{m}}$ will either always receive c_0 or always receive c_1. The decryption queries are answered as in H_3^b with some modifications – the $b = 0$ and $i \geqslant 1$ check is modified to $i \geqslant k + b$ and the check $j \leqslant i$ is modified to $k + b \leqslant j \leqslant i$.[12] Observe that Dec queries in $\mathsf{H}^{1,0}$ will be answered identically as in H_3^0 because the condition $i \geqslant k + b$ in $\mathsf{H}^{1,0}$ is $i \geqslant 1$ and the condition $b = 0 \wedge i \geqslant 1$ is equivalent to $i \geqslant 1$ in H_3^0, and $1 \leqslant j \leqslant i$ is equivalent to $j \leqslant i$ since $j \geqslant 1$. Similarly the Dec queries in $\mathsf{H}^{q_{\text{Enc}},1}$ will be answered identically as in H_3^1 because the condition $i \geqslant q_{\text{Enc}} + 1$ in $\mathsf{H}^{q_{\text{Enc}},0}$ is always false and the condition $b = 0 \wedge i \geqslant 1$ is always false in H_3^1, and the check $k + b \leqslant j \leqslant i$ is never executed in $\mathsf{H}^{q_{\text{Enc}},1}$ just like the check $j \leqslant i$ in H_3^1. So $\mathsf{H}^{1,0}$ perfectly matches H_3^0 and $\mathsf{H}^{q_{\text{Enc}},1}$ perfectly matches H_3^1. Hence combining with (4), we have

$$\mathsf{Adv}_{\mathsf{PKE}}^{\$\mathsf{cca\text{-}m}}(\mathcal{A}_{\mathsf{m}}) \leqslant \Pr[\mathsf{H}^{1,1}] - \Pr[\mathsf{H}^{q_{\text{Enc}},0}] + (q_{\text{Enc}}^2 + 4 \cdot q_{\text{Enc}} \cdot q_{\text{Dec}})/2^\tau \ . \tag{5}$$

[12] The latter check already implies former, so in future games we remove the former.

Hybrids H_h	$\textsc{Enc}(m)$	$\textsc{Dec}(c)$
$//0 \leqslant h \leqslant 2$	$i \leftarrow i+1$	If $c = c^*$: Return m^* $//\mathsf{H}_{[1,\infty)}$
$(k,b) \leftarrow\!\!\$\ [q_{\text{Enc}}] \times \{0,1\}$	$c_1 \leftarrow\!\!\$\ \mathsf{PKE.E}(ek, m)$	$n \leftarrow \mathsf{PKE.C}^{-1}(ek, c)$
$i \leftarrow 0$	$c_0 \leftarrow f((\lvert m\rvert, ek), (m, i))$	$(m, j) \leftarrow f^{-1}((n, ek), c)$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.K}$	If $i < k$:	If $m \neq \bot$ and $k{+}b \leqslant j \leqslant i$: $//\mathsf{H}_{[0,1)}$
$f \leftarrow\!\!\$\ \mathsf{Inj}^{\pm}(T, D, R)$	$\quad c \leftarrow c_1$	If $m \neq \bot$ and $k \leqslant j \leqslant i$: $//\mathsf{H}_{[1,\infty)}$
$b' \leftarrow \mathcal{A}_{\mathsf{m}}^{\textsc{Enc},\textsc{Dec}}(ek)$	If $i = k$:	$\quad$ If $(m, j) = (m^*, k)$: $//\mathsf{H}_{[1,\infty)}$
Return $b' = b$	$\quad c_0 \leftarrow\!\!\$\ \mathsf{PKE.C}(ek, \lvert m\rvert)$ $//\mathsf{H}_{[2,\infty)}$	$\quad\quad$ Skip next line $//\mathsf{H}_{[1,\infty)}$
	$\quad c \leftarrow c_b$	$\quad$ Return m
	$\quad (c^*, m^*) \leftarrow (c, m)$	$m \leftarrow \mathsf{PKE.D}(dk, c)$
	If $i > k$:	Return m
	$\quad c \leftarrow c_0$	
	Return c	

Fig. 28. Final set of hybrids for the proof of Theorem 7. The highlighted line is new with respect to the code of $\mathsf{H}^{k,b}$.

In general, in $\mathsf{H}^{k,b}$ the first $k + b - 1$ encryption queries use c_1 and the rest use c_0; also in $\textsc{Dec}$, the checks involve $k + b$, making them identical in $\mathsf{H}^{k,1}$ and $\mathsf{H}^{k+1,0}$, so $\Pr[\mathsf{H}^{k,1}] = \Pr[\mathsf{H}^{k+1,0}]$ holds. Hence

$$\Pr[\mathsf{H}^{q_{\text{Enc}},1}] - \Pr[\mathsf{H}^{1,0}] = \Pr[\mathsf{H}^{q_{\text{Enc}},1}] - \Pr[\mathsf{H}^{1,0}] + \sum_{k\in[q_{\text{Enc}}-1]} \Pr[\mathsf{H}^{k,1}] - \Pr[\mathsf{H}^{k+1,0}]$$
$$= \sum_{k\in[q_{\text{Enc}}]} \Pr[\mathsf{H}^{k,1}] - \Pr[\mathsf{H}^{k,0}]. \tag{6}$$

Transition to H_h hybrids. Next, consider the games shown in Fig. 28. Of these we make the following claims.

1. $2\Pr[\mathsf{H}_0] - 1 = (1/q_{\text{Enc}}) \sum_{k\in[q_{\text{Enc}}]} \Pr[\mathsf{H}^{k,1}] - \Pr[\mathsf{H}^{k,0}]$
2. $\Pr[\mathsf{H}_0] \leqslant \Pr[\mathsf{H}_1] + (q_{\text{Enc}} + 2q_{\text{Dec}})/2^\tau$
3. $\Pr[\mathsf{H}_1] \leqslant \Pr[\mathsf{H}_2] + q_{\text{Enc}}/2^\tau$
4. $2\Pr[\mathsf{H}_2] - 1 = \mathsf{Adv}^{\$\mathsf{cca\text{-}m}}_{\mathsf{PKE}}(\mathcal{A}_1)$

In the rest of the proof we address these one at a time. Putting them together along with (5) and (3) gives the bound claimed in the theorem statement via the following calculation, where we let $\epsilon = (q_{\text{Enc}}^2 + 4q_{\text{Enc}}q_{\text{Dec}})/2^\tau$,

$$\mathsf{Adv}^{\$\mathsf{cca\text{-}m}}_{\mathsf{PKE}}(\mathcal{A}_{\mathsf{m}}) = \epsilon + \sum_{k\in[q_{\text{Enc}}]} \Pr[\mathsf{H}^{k,1}] - \Pr[\mathsf{H}^{k,0}]$$
$$= \epsilon + q_{\text{Enc}}(2\Pr[\mathsf{H}_0] - 1)$$
$$\leqslant \epsilon + q_{\text{Enc}}(2\Pr[\mathsf{H}_1] + (2q_{\text{Enc}} + 4q_{\text{Dec}})/2^\tau - 1)$$
$$\leqslant \epsilon + q_{\text{Enc}}(2\Pr[\mathsf{H}_2] + (2q_{\text{Enc}} + 4q_{\text{Dec}})/2^\tau + 2q_{\text{Enc}}/2^\tau - 1)$$
$$= \mathsf{Adv}^{\$\mathsf{cca\text{-}m}}_{\mathsf{PKE}}(\mathcal{A}_1) + \epsilon + 4q_{\text{Enc}}q_{\text{Dec}}/2^\tau + 4q_{\text{Enc}}^2/2^\tau$$
$$= \mathsf{Adv}^{\$\mathsf{cca\text{-}m}}_{\mathsf{PKE}}(\mathcal{A}_1) + 8q_{\text{Enc}}q_{\text{Dec}}/2^\tau + 5q_{\text{Enc}}^2/2^\tau$$

Transition to H_0. Game H_0 is identical to $\mathsf{H}^{k,b}$ with (k, b) chosen at random and with the game returning true if $\mathcal{A}_{\mathsf{m}}$ correctly guessed b. We added highlighted code to $\textsc{Enc}$ which stores the message and ciphertext from when $i = k$ in preparation for the next hybrid. We removed the $i \geqslant k + b$ check in $\textsc{Dec}$ because it is anyway implied by the $k + b \leqslant j \leqslant i$ check. Standard calculations by conditioning on all the possible values of (k, b) gives the claim.

Transition from H_0 to H_1. We make three changes to Dec to transition to H_1. First, it immediately returns m^* if given $c = c^*$. Secondly, we modify the check $k + b \leqslant j \leqslant i$ to $k \leqslant j \leqslant i$. Thirdly, we add an if statement inside the if statement checking $m \neq \perp$ and $k \leqslant j \leqslant i$, which checks if $(m, j) = (m^*, k)$ and skips returning m if that is the case.

Note that if $b = 0$, a Dec query on $c = c^*$ would never exhibit different behavior in H_1 compared to H_0 because it returns m^* in both cases. A Dec query on $c \neq c^*$ would never exhibit different behavior in H_1 compared to H_0 because in H_1 the condition to skip a line never gets triggered since $f^{-1}((n, ek), c) \neq f^{-1}((n, ek), c^*) = (m^*, k)$ (because f is injective). Since $b = 0$, the change from $k + b \leqslant j$ to $k \leqslant j$ has no effect.

For $b = 1$, a Dec query on $c = c^*$ would exhibit different behavior in H_0 than in H_1 only if $f^{-1}((n, ek), c^*) \neq \perp$ in H_1 (recall that for $b = 1$, c^* is an actual encryption of m^*). Note that, up until c^* is defined in H_0, the view of the adversary is independent of f. So, the probability that $f^{-1}((n, ek), c^*) \neq \perp$ is at most $q_{\text{Enc}} 2^n / 2^{n+\tau} \leqslant q_{\text{Enc}}/2^\tau$.

For $b = 1$, a Dec query on $c \neq c^*$ would exhibit different behavior in H_1 compared to H_0 only if $f^{-1}((n, ek), c) = (m, j)$ and $m \neq \perp$ and $k = j$ holds in H_1. Because otherwise if $m = \perp$ or $k < j$, the outer if statements would not be true in either game or if $k > j$ then the skip condition (the only difference inside the outer if statement for H_0 and H_1) would never be triggered for H_1.

For every Dec(c) query, the probability that it was first query where c satisfies $f^{-1}((n, ek), c) = (m, k)$ is at most $2^n/(2^{n+\tau} - q_{\text{Enc}} - q_{\text{Dec}}) \leqslant 2/2^\tau$. This follows because there are 2^n values in the domain of f^{-1} that map to (m, k) for some m and because it is the first query where c satisfies $f^{-1}((n, ek), c) = (m, k)$, the view of the adversary is dependent only on points of f^{-1} which are mapped to $\perp$ or (m', k') where $k' \neq k$. Taking a union bound over all Dec queries, this event happens with probability at most $2q_{\text{Dec}}/2^\tau$ (where we use the fact that $q_{\text{Dec}} + q_{\text{Enc}} \leqslant 0.5 \cdot \tau$). This gives us, $\Pr[\mathsf{H}_0] \leqslant \Pr[\mathsf{H}_1] + (q_{\text{Enc}} + 2q_{\text{Dec}})/2^\tau$.

Transition from H_1 to H_2. The only change in H_2 is that in Enc for $i = k$, c_0 sampled uniformly at random instead of evaluating f. Note that this changes behavior in Enc of H_2 only if the c_0 sampled is the same as c_0 for some previous Enc query – since the size of $\mathsf{PKE.C}$ is always at least 2^τ, this happens with probability at most $q_{\text{Enc}}/2^\tau$. For Dec queries on $c = c^*$, the behavior is identical in H_1 and H_2 because of the check at the beginning. For a Dec query on $c \neq c^*$, the answer of the query in no way would depend on c_0 sampled for $i = k$. So the behavior of Dec is unchanged. It follows that $\Pr[\mathsf{H}_1] \leqslant \Pr[\mathsf{H}_2] + q_{\text{Enc}}/2^\tau$.

Adversary $\mathcal{A}_1$. Finally we can see that our adversary $\mathcal{A}_1$ (defined in Fig. 25) perfectly simulates the view of $\mathcal{A}_\mathsf{m}$ in H_3. It was obtained by copying the code of H_3 and then modifying it to query its Enc and Dec oracle as appropriate. It follows that $\Pr[\mathsf{H}_2] = 0.5 \Pr[\mathsf{G}^{\$\text{cca-m}}_{\mathsf{PKE},1}] + 0.5(1 - \Pr[\mathsf{G}^{\$\text{cca-m}}_{\mathsf{PKE},0}])$. Hence, $2\Pr[\mathsf{H}_2] - 1 = \mathsf{Adv}^{\$\text{cca-m}}_{\mathsf{PKE}}(\mathcal{A}_1)$.

Adversary $\mathcal{A}_1$'s extra running time comes from using $\mathsf{PKE.E}$ in SimEnc. Its extra memory is that required for running $\mathsf{PKE.E}$, for storing i, and for storing (c^*, m^*). ▭

Acknowledgements

Ashrujit Ghoshal, Joseph Jaeger, and Stefano Tessaro were partially supported by NSF grants CNS-1930117 (CAREER), CNS-1926324, CNS-2026774, a Sloan Research Fellowship, and a JP Morgan Faculty Award. Joseph Jaeger's work was done while at the University of Washington.

References

1. 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. 1, 2, 3, 4, 5, 6, 8, 9, 10, 16, 24
2. Benedikt Auerbach, David Cash, Manuel Fersch, and Eike Kiltz. Personal communication, 2021. 24, 26
3. Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? *Journal of Cryptology*, 28(1):29–48, January 2015. 26
4. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology*, 21(4):469–491, October 2008. 4

5. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. 4, 16
6. 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. 6, 23, 27, 32
7. Daniel J Bernstein. Extending the salsa20 nonce. In *Workshop record of Symmetric Key Encryption Workshop*, volume 2011. Citeseer, 2011. 3
8. 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. 1, 8
9. Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Heidelberg, August 2000. 16
10. Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 272–287. Springer, Heidelberg, April / May 2002. 4, 9, 15, 16
11. 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. 1, 4, 19
12. 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, Cham., December 2021. 1, 4, 8, 13, 17
13. 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. 1, 2, 7, 8
14. 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. 1, 2, 4, 19
15. Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, March 1996. 13
16. Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 98–129. Springer, Heidelberg, August 2017. 37
17. 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. 1, 2, 4, 5, 19, 20, 21
18. 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. 1, 2, 5
19. Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 482–499. Springer, Heidelberg, August 2003. 36
20. Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust authenticated-encryption AEZ and the problem that it solves. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 15–44. Springer, Heidelberg, April 2015. 36
21. 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. 1, 2, 4, 7, 8, 19
22. Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988. 36
23. Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 257–274. Springer, Heidelberg, May 2014. 4, 20
24. Phillip Rogaway. Nonce-based symmetric encryption. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 348–359. Springer, Heidelberg, February 2004. 4, 20
25. 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. 1, 2, 4, 7, 8, 19

26. 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. 1, 4, 19
27. 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. 1, 2, 5, 8

A Extended Discussion of Oracle Adversaries

Following our convention from Section 3, the reduction adversaries we provide in a variety of our results are $\mathcal{F}$-oracle adversaries for different choices of $\mathcal{F}$. This is summarized by the table in Fig. 29. One justification for why this is acceptable is that this oracle can always be pseudorandomly instantiated, if needed (as captured by Theorem 2). In this section we will give examples to show that appropriate pseudorandom objects exist.

Result	Security Notion	Required Oracle
Thm. 1	1UFCMA	$\mathsf{Inj}^{\pm}(\mathsf{DS.M}, [q_{\text{SIGN}}], \mathsf{D.R})$
Thm. 3	1UFCMA	$\mathsf{Fcs}(\mathsf{DS.M}, [q_{\text{SIGN}}], \mathsf{D.R})$
Thm. 4	OW-RSA	$\mathsf{Fcs}(\{0,1\}^{\mathsf{R.k}}, \{0,1\}^*, \mathbb{Z}^*_{(\cdot)}) \times \mathsf{Inj}(\{0,1\}^*, [q_{\text{SIGN}}], \{0,1\}^{\mathsf{rl}})$
Thm. 5	INDR	$\mathsf{Fcs}(\mathsf{NE.N}, \mathsf{NE.C}, \{0,1\}^{\tau})$
Thm. 6	1CCA-$\diamond$	$\mathsf{Fcs}(\mathbb{N}, D, \mathsf{PKE.R})$
Thm. 7	1\$CCA-`m`	$\mathsf{Inj}^{\pm}(\mathbb{N} \times \mathsf{PKE.Ek}, \{0,1\}^n \times [q_{\text{ENC}}], \mathsf{PKE.C}(ek, n))$

Fig. 29. Summary of the $\mathcal{F}$-oracle adversaries obtained by the results in out paper.

However we emphasize that our perspective is that, in general, fixing specific concrete choices for these instantiations is a secondary concern. Suppose, for example, that an efficient, low-memory adversary $\mathcal{A}$ is shown to exist against the security of a digital signature scheme DS which was proven secure with an $\mathcal{F}$-oracle reduction $\mathcal{R}[]$ to cryptographic assumption Π. A motivated cryptographer could easily put together a list $\mathsf{F}_1, \mathsf{F}_2, \ldots$ or many candidate $\mathcal{F}$-pseudorandom functions. Either instantiating $\mathcal{R}[\mathcal{A}]$'s oracle with one of the functions on this list would give an efficient, low-memory attack against Π or $\mathcal{R}[\mathcal{A}]$ would be successfully attacking the pseudorandomness of *every* F_i on our list (which was chosen after $\mathcal{R}$ and $\mathcal{A}$ were fixed).

Consequently, our priority in this section is to highlight the existence of appropriate pseudorandom objects, not in making optimal choices for them. We optimize our choices for ease of explanation. For example, all the required tweakable random functions (Theorems 3, 4, 5, and 6) can implemented using a hash function (say SHA2 or SHA3) which is commonly modeled as a random oracle. For $\mathsf{Fcs}(T, D, R)$ given a hash function $H : \{0,1\}^* \to R$ we simply define $\mathsf{F}_K(t, d) = H(K \,\|\, \langle t, d \rangle)$ where is $\langle \cdot, \cdot \rangle$ encoding of tuples from $T \times D$. If $R = \{0,1\}^{\tau}$, finding such an H is straightforward. For other R of interest we expect to be able to use standard techniques to create such a H from H' with range $\{0,1\}^{\tau}$ (e.g. using rejection sampling, so $H(x) = H'(i, x)$ for the first $i \in [n]$ such that $H'(i, x) \in R$).

Tweakable injections, $\mathsf{Inj}^{\pm}(T, D, R)$, are needed for Theorems 1, 4, and 7. We first note that a tweakable pseudorandom injection with efficient inversion can be instantiated from blockciphers using the CMC enciphering scheme [19]. Since blockciphers can in turn be constructed from PRFs [22], the assumption of existence of tweakable pseudorandom injection with efficient inversions is essentially same as the assumption of existence of PRFs. A large, structured tweak set T can always be handled by using a collision-resistant hash function to map it to a more standard tweak set. For examples of interest, the choices of R for Theorems 1 and 4 are likely to be $\{0,1\}^{128}$ or $\{0,1\}^{256}$. Thus we can implement them by taking tweakable blockciphers obtained from standard blockciphers and restricting their domain to match D. Should Theorem 1 require a larger R we can use a large-block blockcipher such as those designed by Hoang, Krovetz, and Rogaway [20].

For Theorem 7, given tweak (n, ek) we need an injection from elements of $\{0,1\}^n \times [q_{\text{Enc}}]$ to $\mathsf{PKE.C}(ek, n)$ the ciphertext space of PKE using key ek for messages of length n. How this is achieved, will of course depend on the choice of PKE. One common possibility is $\mathsf{PKE.C}(ek, n) = \mathcal{C}(ek) \times \{0,1\}^{n+\tau}$ where $\mathcal{C}$ is some structured set dependent on ek (e.g. $\mathbb{Z}_N$ for RSA) and τ is a constant (e.g. 128). If $\lceil \log q_{\text{Enc}} \rceil$ bits of a random element of $c \in \mathcal{C}(ek)$ look uniformly random (when not conditioned on the rest of c) then we can use a misuse-resistant authenticated encryption scheme with ciphertexts τ bits longer than the input message to encrypt the message $(m, i) \in \{0,1\}^n \times [q_{\text{Enc}}]$ using (n, ek) as a nonce to obtain a ciphertext of length $\lceil \log q_{\text{Enc}} \rceil + n + \tau$. Treating the first $\lceil \log q_{\text{Enc}} \rceil$ bits of the ciphertext as part of some $c \in \mathcal{C}(ek)$ we can use deterministic rejection sampling (using some PRF) to sample the rest of the bits of c.

B KEM/DEM (Application Requiring AE/\$CCA-m)

In this section we exhibit reductions which are memory-tight when we use the $w = \mathtt{m}$ variant of definitions, but for which this memory-tightness does not appear possible if we use $w = \diamond$ or $\bot$. The reductions are for proving the security encryption schemes based on the KEM/DEM paradigm. In the paradigm we construct an encryption scheme KD given two encryption schemes KEM and DEM. To encrypt a message m we first sample a random key K for DEM which we encrypt using KEM. Then using DEM with key K we encrypt m. Decryption proceeds by using KEM to recover K and then using it with DEM to recover m.

We consider the cases when KEM (and hence KD) is a secret-key or public-key encryption scheme.[13] In either case we use a secret-key encryption scheme for DEM. For now let us consider the secret-key case; the public-key case is similar. Our goal is to show that if KEM is AE-w secure and DEM is AE-w secure (against multi-user attacks making one encryption query per user), then KD is AE-w secure. Our goal is for these reductions to be memory-tight.

In particular, for motivating the usefulness of -m style definition, we are most interested in the reduction to the security of KEM. In fact, the standard reduction to the security of KEM works and is memory-tight when $w = \mathtt{m}$. (The reduction to the security of DEM will require non-standard steps making use of our efficient tagging technique.) It seems unlikely the reduction to KEM's security can be made memory-tight if $w \in \{\bot, \diamond\}$ instead.

To understand the issue at hand, let us discuss how the proof works at a high level. Broadly, we use a reduction to the security of KEM to switch ciphertexts encrypting K to be random so that K itself is random from the perspective of the adversary. With these random K we can then apply the security of DEM. Our focus is on the memory-tightness of the first step. Given adversary $\mathcal{A}_{\mathsf{kd}}$ against KD we build $\mathcal{A}_{\mathsf{k}}$ against security of KEM as following. On an encryption query for m, adversary $\mathcal{A}_{\mathsf{k}}$ will sample a random K and then ask its own encryption oracle for an encryption c_{k} of K and locally use K to encrypt m into the ciphertext c_{d}. It returns $(c_{\mathsf{k}}, c_{\mathsf{d}})$ to $\mathcal{A}_{\mathsf{kd}}$. The interesting challenge is in simulating decryption queries for some $(c_{\mathsf{k}}, c_{\mathsf{d}})$. When $w = \mathtt{m}$, adversary $\mathcal{A}_{\mathsf{k}}$ can query c_{k} to its own decryption oracle to receive some K and use that to decrypt c_{d}. For $w \in \{\bot, \diamond\}$, this works fine unless c_{k} was previously returned by an encryption query in which case $\mathcal{A}_{\mathsf{k}}$'s oracle returns $\bot$ or $\diamond$ (as appropriate) which does not enable $\mathcal{A}_{\mathsf{k}}$ to simulate decryption properly if c_{d} was not also returned by that encryption query. As such, it is not clear how $\mathcal{A}_{\mathsf{k}}$ could respond correctly without storing all prior encryption queries.

The step described above is main motivation of this section, showing how $w = \mathtt{m}$ can be useful for security proofs. The second step of the proof also required some care to be memory-tight. The natural reduction to the multi-user security of DEM works as follows. For an encryption query it samples a random ciphertext for the KEM, samples key K at random, then queries its oracles to create a new user for DEM and have that user encrypt K. On a decryption query $(c_{\mathsf{k}}, c_{\mathsf{d}})$, if c_{k} was a ciphertext it picked randomly for a prior encryption query, it wants to query its decryption oracle to get c_{d} decrypted. This requires remembering the name of the new user created during the encryption query that returned c_{k} and would thus not be memory-tight. We use our message encoding technique to make this memory-tight. In particular, to respond to encryption

[13] The public-key case is a standard "textbook" construction of an encryption scheme. The secret-key case also arises in practice where a master key is used to encryption subkeys which are used to encryption the message. See, for example, some key-rotation schemes [16].

$\mathsf{KD}[\mathsf{NE}_\mathsf{k},\mathsf{NE}_\mathsf{d}].\mathsf{K}$	$\mathsf{KD}[\mathsf{NE}_\mathsf{k},\mathsf{NE}_\mathsf{d}].\mathsf{E}(K_\mathsf{k},n,m)$	$\mathsf{KD}[\mathsf{NE}_\mathsf{k},\mathsf{NE}_\mathsf{d}].\mathsf{D}(K_\mathsf{k},n,c)$
$K_\mathsf{k} \leftarrow\!\$\ \mathsf{NE}_\mathsf{k}.\mathsf{K}$ Return K	$K \leftarrow\!\$\ \mathsf{NE}_\mathsf{d}.\mathsf{K}$ $c_\mathsf{k} \leftarrow \mathsf{NE}_\mathsf{k}.\mathsf{E}(K_\mathsf{k},n,K)$ $c_\mathsf{d} \leftarrow \mathsf{NE}_\mathsf{d}.\mathsf{E}(K,0,m)$ Return $(c_\mathsf{k},c_\mathsf{d})$	$(c_\mathsf{k},c_\mathsf{d}) \leftarrow c$ $K \leftarrow \mathsf{NE}_\mathsf{k}.\mathsf{D}(K_\mathsf{k},n,c_\mathsf{k})$ If $K \neq \perp$: Return $\mathsf{NE}_\mathsf{d}.\mathsf{D}(K,0,c_\mathsf{d})$ Return $\perp$

Fig. 30. Nonce-based AE scheme $\mathsf{KD}[\mathsf{NE}_\mathsf{k},\mathsf{NE}_\mathsf{d}]$ constructed from nonce-based AE schemes NE_k and NE_d via the KEM/DEM paradigm.

Game $\mathsf{G}^{\text{mu-ae-}w}_{\mathsf{NE},b}(\mathcal{A})$	$\textsc{Enc}_b(i,n,m)$	$\textsc{Dec}^w_b(i,n,c)$
$u \leftarrow 0$ $b' \leftarrow \mathcal{A}^{\textsc{New},\textsc{Enc}_b,\textsc{Dec}^w_b}$ Return $b' = 1$ $\textsc{New}()$ $u \leftarrow u+1$ $K_u \leftarrow\!\$\ \mathsf{NE}.\mathsf{K}$	$c_1 \leftarrow \mathsf{NE}.\mathsf{E}(K_i,n,m)$ $c_0 \leftarrow\!\$\ \{0,1\}^{\mathsf{NE}.\mathsf{cl}(\lvert m \rvert)}$ $M[i,n,c_b] \leftarrow m$ Return c_b	If $M[i,n,c] \neq \perp$ Return $M[i,n,c]$ if $w = \mathtt{m}$ Return $\diamond$ if $w = \diamond$ Return $\perp$ if $w = \perp$ $m_1 \leftarrow \mathsf{NE}.\mathsf{D}(K_i,n,c)$ $m_0 \leftarrow \perp$ Return m_b

Fig. 31. Game defining multi-user AE security

queries we create c_k as the output of an injection applied to the name of the new user being created which allows us to later recover this when responding to decryption queries.

In Section B.1 we formalize the proof when KEM is a secret-key, nonce-based scheme and in Section B.2 we do the same for when it is a public-key scheme.

B.1 Secret-key KEM/DEM

KEM/DEM Scheme. Let NE_k and NE_d be nonce-based encryption schemes. Then we define the KEM/DEM nonce-based encryption scheme $\mathsf{KD}[\mathsf{NE}_\mathsf{k},\mathsf{NE}_\mathsf{d}]$ as shown in Fig. 30. Technically KD does not meet our syntax for nonce-based encryption from Section 5, due to being randomized. Requiring nonce-based encryption be deterministic is not actually important for our purposes, so we ignore this technicality.

Multi-user AE security. For our proof we will require that NE_d provide multi-user security against attacks making one encryption query. So we require an extension of the notion of AE security to the multi-user setting. The multi-user setting allows the adversary to make encryption and decryption queries for multiple keys. We use the games in Fig. 31 to define the multi-user AE (muAE) security. For $w \in \{\mathtt{m},\diamond,\perp\}$, the advantage against multi-user AE security of a scheme NE is defined as $\mathsf{Adv}^{\text{mu-ae-}w}_{\mathsf{NE}}(\mathcal{A}) = \Pr[\mathsf{G}^{\text{mu-ae-}w}_{\mathsf{NE},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\text{mu-ae-}w}_{\mathsf{NE},0}(\mathcal{A})]$.

Security result. The following theorem captures our result, that $\mathsf{KD}[\mathsf{NE}_\mathsf{k},\mathsf{NE}_\mathsf{d}]$ can be proven AE-$\mathtt{m}$ secure from the AE-$\mathtt{m}$ security of NE_k and the muAE-$\mathtt{m}$ security of NE_d, where both reductions are memory-tight.

Theorem 8. *Let NE_k, NE_d be nonce-based encryption schemes and let $\tau = \mathsf{NE}_\mathsf{k}.\mathsf{cl}(\mathsf{NE}_\mathsf{d}.\mathsf{kl}) > 0$. Let $\mathcal{A}_a$ be an AE-m adversary with $\mathbf{Query}(\mathcal{A}_a) = (q_{\textsc{Enc}}, q_{\textsc{Dec}})$. Define $(T,D,R) = (\mathsf{NE}_\mathsf{k}.\mathsf{N}, [q_{\textsc{Enc}}], \{0,1\}^\tau)$. Let $\mathcal{B}_a$ be as defined in Fig. 33 and $\mathcal{C}_a$ be the $\mathsf{Inj}^{\pm}(T,D,R)$-oracle adversary defined in Fig. 35. Then,*

$$\mathsf{Adv}^{\mathsf{ae}\text{-}m}_{\mathsf{KD}[\mathsf{NE}_\mathsf{k},\mathsf{NE}_\mathsf{d}]}(\mathcal{A}_a) \leqslant \mathsf{Adv}^{\mathsf{ae}\text{-}m}_{\mathsf{NE}_\mathsf{k}}(\mathcal{B}_a) + \mathsf{Adv}^{\text{mu-ae-}m}_{\mathsf{NE}_\mathsf{d}}(\mathcal{C}_a) + q_{\textsc{Enc}}(q_{\textsc{Enc}} + 4q_{\textsc{Dec}})/2^\tau \ .$$

$$\mathbf{Query}(\mathcal{B}_a) = (q_{\textsc{Enc}}, q_{\textsc{Dec}}) \qquad \mathbf{Query}(\mathcal{C}_a) = (q_{\textsc{Enc}}, q_{\textsc{Dec}})$$
$$\mathbf{Time}(\mathcal{B}_a) = \mathbf{Time}(\mathcal{A}_a) + (q_{\textsc{Enc}} + q_{\textsc{Dec}})\mathbf{Time}(\mathsf{NE}_\mathsf{d}) \qquad \mathbf{Time}^*(\mathcal{C}_a) = \mathbf{Time}(\mathcal{A}_a)$$
$$\mathbf{Mem}(\mathcal{B}_a) = \mathbf{Mem}(\mathcal{A}_a) + \mathbf{Mem}(\mathsf{NE}_\mathsf{d}). \qquad \mathbf{Mem}^*(\mathcal{C}_a) = \mathbf{Mem}(\mathcal{A}_a).$$

Hybrids H_h for $0 \leqslant h \leqslant 3$	$\textsc{Enc}(n,m)$	$\textsc{Dec}(n,c)$

$f \leftarrow\$ \mathsf{Inj}(T,D,R)$ $//\mathsf{H}_{[2,\infty)}$
$l \leftarrow \mathbb{I}$
$u \leftarrow 0$ $//\mathsf{H}_{[2,\infty)}$
$K_{\mathsf{k}} \leftarrow\$ \mathsf{NE_k.K}$
$b' \leftarrow \mathcal{A}_a^{\textsc{Enc},\textsc{Dec}}$
Return $b' = 1$
	$u \leftarrow u + 1$ $//\mathsf{H}_{[2,\infty)}$
	$K \leftarrow\$ \mathsf{NE_d.K}$
	$c_{\mathsf{k}} \leftarrow \mathsf{NE_k.E}(K_{\mathsf{k}}, n, K)$ $//\mathsf{H}_{[0,1)}$
	$c_{\mathsf{k}} \leftarrow\$ \{0,1\}^{\mathsf{NE_k.cl}(\lvert K \rvert)}$ $//\mathsf{H}_{[1,2)}$
	$c_{\mathsf{k}} \leftarrow f(n,u)$ $//\mathsf{H}_{[2,\infty)}$
	$c_{\mathsf{d}} \leftarrow \mathsf{NE_d.E}(K, 0, m)$
	$Y[l(n, c_{\mathsf{k}})] \leftarrow K$
	$M[l(n, c_{\mathsf{k}}), c_{\mathsf{d}}] \leftarrow m$ $//\mathsf{H}_{[3,\infty)}$
	Return $(c_{\mathsf{k}}, c_{\mathsf{d}})$
		$(c_{\mathsf{k}}, c_{\mathsf{d}}) \leftarrow c$
		If $Y[l(n, c_{\mathsf{k}})] \neq \bot$:
		  If $M[l(n, c_{\mathsf{k}}), c_{\mathsf{d}}] \neq \bot$: $//\mathsf{H}_{[3,\infty)}$
		    Return $M[l(n, c_{\mathsf{k}}), c_{\mathsf{d}}]$ $//\mathsf{H}_{[3,\infty)}$
		  $K \leftarrow Y[l(n, c_{\mathsf{k}})]$
		  Return $\mathsf{NE_d.D}(K, 0, c_{\mathsf{d}})$
		Else:
		  $K \leftarrow \mathsf{NE_k.D}(K_{\mathsf{k}}, n, c_{\mathsf{k}})$ $//\mathsf{H}_{[0,1)}$
		  $K \leftarrow \bot$ $//\mathsf{H}_{[1,\infty)}$
		  If $K \neq \bot$:
		    Return $\mathsf{NE_d.D}(K, 0, c_{\mathsf{d}})$
		Return $\bot$

Fig. 32. First set of hybrids used for proof of Theorem 8. $\mathbb{I}$ denotes the identity function.

Adversary $\mathcal{B}_a^{\textsc{Enc},\textsc{Dec}}$	$\textsc{SimEnc}(n,m)$	$\textsc{SimDec}(n,c)$
$b' \leftarrow \mathcal{A}_a^{\textsc{SimEnc},\textsc{SimDec}}$ Return b'	$K \leftarrow\$ \mathsf{NE_d.K}$ $c_{\mathsf{k}} \leftarrow \textsc{Enc}(n, K)$ $c_{\mathsf{d}} \leftarrow \mathsf{NE_d.E}(K, 0, m)$ Return $(c_{\mathsf{k}}, c_{\mathsf{d}})$	$(c_{\mathsf{k}}, c_{\mathsf{d}}) \leftarrow c$ $K \leftarrow \textsc{Dec}(n, c_{\mathsf{k}})$ If $K \neq \bot$: Return $\mathsf{NE_d.D}(K, 0, c_{\mathsf{d}})$ Return $\bot$

Fig. 33. Adversary $\mathcal{B}_a$ for Theorem 8.

Proof. We consider the hybrids H_0 through H_3 and L_0 through L_4 defined in Figs. 32 and 34. Of these hybrids we will make the following claims, which establish the claimed upper bound on the advantage of $\mathcal{A}_a$.

1. $\Pr[\mathsf{G}^{\mathsf{ae\text{-}m}}_{\mathsf{KD}[\mathsf{NE_k},\mathsf{NE_d}],1}(\mathcal{A}_a)] = \Pr[\mathsf{H}_0]$
2. $\Pr[\mathsf{H}_0] \leqslant \Pr[\mathsf{H}_1] + \mathsf{Adv}^{\mathsf{ae\text{-}m}}_{\mathsf{NE_k}}(\mathcal{B}_a)$
3. $\Pr[\mathsf{H}_1] \leqslant \Pr[\mathsf{H}_2] + 0.5 \cdot q^2_{\textsc{Enc}}/2^{\tau}$
4. $\Pr[\mathsf{H}_2] = \Pr[\mathsf{H}_3]$
5. $\Pr[\mathsf{H}_3] \leqslant \Pr[\mathsf{L}_0] + 2q_{\textsc{Enc}}q_{\textsc{Dec}}/2^{\tau}$
6. $\Pr[\mathsf{L}_0] \leqslant \Pr[\mathsf{L}_1] + \mathsf{Adv}^{\mathsf{mu\text{-}ae\text{-}m}}_{\mathsf{NE_d}}(\mathcal{C}_a)$
7. $\Pr[\mathsf{L}_1] \leqslant \Pr[\mathsf{L}_2] + 2q_{\textsc{Enc}}q_{\textsc{Dec}}/2^{\tau}$
8. $\Pr[\mathsf{L}_2] \leqslant \Pr[\mathsf{L}_3] + 0.5 \cdot q^2_{\textsc{Enc}}/2^{\tau}$
9. $\Pr[\mathsf{L}_3] = \Pr[\mathsf{G}^{\mathsf{ae\text{-}m}}_{\mathsf{KD}[\mathsf{NE_k},\mathsf{NE_d}],0}(\mathcal{A}_a)]$

Transition to H_0. We claim that H_0 is identical to $\mathsf{G}^{\mathsf{ae\text{-}m}}_{\mathsf{KD}[\mathsf{NE_k},\mathsf{NE_d}],1}$. Note that, in the latter, Enc produces honest encryptions using KD and Dec produces honest decryptions. It is immediately clear that the same holds for Enc in H_0 and the else branch of Dec in H_0. Consider the first branch in Dec. We have used grey highlighting to indicate the relevant code. Note that l is the identity function (its presence will be notationally convenient for future game transitions). Y is a table indexed by n, c_{k} pairs which stores the key in encrypted c_{k}. The use of Y in Dec simply recovers this K without decrypting c_{k}. By the correctness of $\mathsf{NE_k}$ this is identical to having done the decryption, so $\Pr[\mathsf{G}^{\mathsf{ae\text{-}m}}_{\mathsf{KD}[\mathsf{NE_k},\mathsf{NE_d}],1}(\mathcal{A}_a)] = \Pr[\mathsf{H}_0]$ follows.

Transition H_0 to H_1. In hybrid H_1, the ciphertext c_{k} is now sampled uniformly at random. Also in Dec, the key K is assigned the value $\bot$ if the $Y[n, c_{\mathsf{k}}] = \bot$. These difference correspond to what we expect from the security of $\mathsf{NE_k}$. We define $\mathcal{B}_a$ in Fig. 33 from hybrid H_1 by replacing encryption and decryption of K with appropriate queries to its oracle. This adversary is nonce-respecting because $\mathcal{A}_a$ is. Note that the use of table Y in the hybrids matches the table from $\mathsf{G}^{\mathsf{ae\text{-}m}}$. We can see that when interacting with $\mathsf{G}^{\mathsf{ae\text{-}m}}_{\mathsf{NE_k},1}$, $\mathcal{B}_a$ simulates H_0 to $\mathcal{A}_a$ and when interacting with $\mathsf{G}^{\mathsf{ae\text{-}m}}_{\mathsf{NE_k},0}$, $\mathcal{B}_a$ simulates H_1 to $\mathcal{A}_a$. Hence $\Pr[\mathsf{H}_0] = \Pr[\mathsf{H}_1] + \mathsf{Adv}^{\mathsf{ae\text{-}m}}_{\mathsf{NE_k}}(\mathcal{B}_a)$.

After this transition the else branch in Dec is dead code. We remove it later when transitioning to L_0.

Hybrids L_ℓ for $0 \leqslant \ell \leqslant 3$	$\text{Enc}(n, m)$	$\text{Dec}(n, c)$
$f \leftarrow\!\!\$\ \mathsf{Inj}^{\pm}(T, D, R)$ $//\mathsf{L}_{[0,3)}$	$u \leftarrow u + 1$ $//\mathsf{L}_{[0,3)}$	$(c_\mathsf{k}, c_\mathsf{d}) \leftarrow c$
$l \leftarrow f^{-1}$ $//\mathsf{L}_{[0,2)}$	$K \leftarrow\!\!\$\ \mathsf{NE_d.K}$	If $1 \leqslant f^{-1}(n, c_\mathsf{k}) \leqslant u$: $//\mathsf{L}_{[0,2)}$
$l \leftarrow \mathbb{I}$ $//\mathsf{L}_{[2,\infty)}$	$c_\mathsf{k} \leftarrow f(n, u)$ $//\mathsf{L}_{[0,3)}$	If $Y[l(n, c_\mathsf{k})] \neq \bot$: $//\mathsf{L}_{[2,\infty)}$
$u \leftarrow 0$ $//\mathsf{L}_{[0,3)}$	$c_\mathsf{k} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{NE_k.cl}(\lvert K \rvert)}$ $//\mathsf{L}_{[3,\infty)}$	If $M[l(n, c_\mathsf{k}), c_\mathsf{d}] \neq \bot$:
$K_\mathsf{k} \leftarrow\!\!\$\ \mathsf{NE_k.K}$	$c_\mathsf{d} \leftarrow \mathsf{NE_d.E}(K, 0, m)$ $//\mathsf{L}_{[0,1)}$	Return $M[l(n, c_\mathsf{k}), c_\mathsf{d}]$
$b' \leftarrow \mathcal{A}_a^{\text{Enc},\text{Dec}}$	$c_\mathsf{d} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{NE_d.cl}(\lvert m \rvert)}$ $//\mathsf{L}_{[1,\infty)}$	$K \leftarrow Y[l(n, c_\mathsf{k})]$ $//\mathsf{L}_{[0,1)}$
Return $b' = 1$	$Y[l(n, c_\mathsf{k})] \leftarrow K$	Return $\mathsf{NE_d.D}(K, 0, c_\mathsf{d})$ $//\mathsf{L}_{[0,1)}$
	$M[l(n, c_\mathsf{k}), c_\mathsf{d}] \leftarrow m$	Return $\bot$
	Return $(c_\mathsf{k}, c_\mathsf{d})$	

Fig. 34. Second set of hybrids used for proof of Theorem 8.

Adversary $\mathcal{C}_a^{\text{New},\text{Enc},\text{Dec}}$	$\text{SimEnc}(n, m)$	$\text{SimDec}(n, c)$
$f \leftarrow\!\!\$\ \mathsf{Inj}(T, D, R)$	$u \leftarrow u + 1$	$(c_\mathsf{k}, c_\mathsf{d}) \leftarrow c$
$u \leftarrow 0$	$\text{New}()$	$i \leftarrow f^{-1}(n, c_\mathsf{k})$
$b' \leftarrow \mathcal{A}_a^{\text{SimEnc},\text{SimDec}}$	$c_\mathsf{k} \leftarrow f(n, u)$	If $1 \leqslant i \leqslant u$:
Return b'	$c_\mathsf{d} \leftarrow \text{Enc}(u, 0, m)$	Return $\text{Dec}(i, 0, c_\mathsf{d})$
	Return $(c_\mathsf{k}, c_\mathsf{d})$	Return $\bot$

Fig. 35. Adversary $\mathcal{C}_a$ for Theorem 8.

TRANSITION H_1 TO H_2. In hybrid H_2, instead of sampling c_k at random, we assign it the output of a random injective function f applied to nonce n and a counter u (we will later make u correspond to users in $\mathsf{G}^{\mathsf{mu\text{-}ae\text{-}}w}$). The switching lemma tells us that $\Pr[\mathsf{H}_2] \leqslant \Pr[\mathsf{H}_1] + 0.5 \cdot q_{\text{Enc}}^2/2^\tau$.

TRANSITION H_2 TO H_3. In hybrid H_3, we introduce a table M which is indexed by n, c_k, c_d (the first two via l) and stores the value of m whose encryption under $K = Y[l(n, c_\mathsf{k})]$ is c_d. This table is used in Dec to skip the step of decrypting c_d. By the correctness of $\mathsf{NE_d}$ this does not change functionality, so $\Pr[\mathsf{H}_2] = \Pr[\mathsf{H}_3]$.

TRANSITION H_3 TO L_0. Next we transition to hybrid L_0 shown in Fig. 34. We have highlighted all ways that this differs from hybrid H_3 (other than the elimination of the aforementioned dead code in the else branch of Dec). Our changes were twofold, consisting of switching the function l indexing into our tables to f^{-1} and switching the if condition in Dec. Considering the latter first, note that the checks $Y[l(n, c_\mathsf{k})] \neq \bot$ in H_3 and $1 \leqslant f^{-1}(n, c_\mathsf{k}) \leqslant u$ in L_0 can only differ if the second is true and the first is false. This requires the adversary to guess something in the image of $f(n, \cdot)$ other than the (at most) one example it can obtain from Enc. Switching to using $l = f^{-1}$ similarly can only change behavior if a $\text{Dec}(n, (c_\mathsf{k}, c_\mathsf{d}))$ query is made where c_k is in the image of $f(n, \cdot)$, but c_k was not returned in a prior Enc query with n. Note that the domain (and hence the image) of $f(n, \cdot)$ has size q_{Enc} and its range has size 2^τ. So for a given decryption query this bad event happens in H_3 with probability at most $q_{\text{Enc}}/(2^\tau - 1) \leqslant 2q_{\text{Enc}}/2^\tau$. Taking a union bound over all Dec queries gives that $\Pr[\mathsf{H}_3] \leqslant \Pr[\mathsf{L}_0] + 2q_{\text{Enc}}q_{\text{Dec}}/2^\tau$.

TRANSITION L_0 TO L_1. In hybrid L_1, the ciphertext c_d is now sampled uniformly at random. Also in Dec, if the $M[l(n, c_\mathsf{k}), c_\mathsf{d}] = \bot$ then the oracle always returns $\bot$. These difference correspond to what we expect from the multi-user security of $\mathsf{NE_d}$. We define $\mathcal{C}_a$ in Fig. 35 from hybrid L_1 by replacing encryption and decryption of c_d with appropriate queries to its oracle. We claim that when interacting with $\mathsf{G}_{\mathsf{NE},1}^{\mathsf{mu\text{-}ae\text{-}m}}$, $\mathcal{C}_a$ simulates L_0 to $\mathcal{A}_a$ and when interacting with $\mathsf{G}_{\mathsf{NE},0}^{\mathsf{mu\text{-}ae\text{-}m}}$, $\mathcal{C}_a$ simulates L_1 to $\mathcal{A}_a$. To see this, note that the variable u in these hybrids matches the same variable in $\mathsf{G}^{\mathsf{mu\text{-}ae\text{-}m}}$. Additionally, if $i = l(n, c_\mathsf{k})$ then the values $Y[i]$ and $M[i, c_\mathsf{d}]$ in these hybrids always match K_i and $M[i, 0, c_\mathsf{d}]$ in $\mathsf{G}^{\mathsf{mu\text{-}ae\text{-}m}}$. (The claims can be

Game $\mathsf{G}^{\text{mu-\$cca-}w}_{\mathsf{NE},b}(\mathcal{A})$	$\text{Enc}_b(i,n,m)$	$\text{Dec}^w_b(i,n,c)$
$u \leftarrow 0$	$c_1 \leftarrow \mathsf{NE.E}(K_i,n,m)$	If $M[i,n,c] \neq \bot$
$b' \leftarrow \mathcal{A}^{\text{New},\text{Enc}_b,\text{Dec}^w_b}$	$c_0 \leftarrow\$ \{0,1\}^{\mathsf{NE.cl}(\lvert m \rvert)}$	Return $M[i,n,c]$ if $w = \mathtt{m}$
Return $b' = 1$	$M[i,n,c_b] \leftarrow m$	Return $\diamond$ if $w = \diamond$
$\text{New}()$	Return c_b	Return $\bot$ if $w = \bot$
$u \leftarrow u+1$		$m \leftarrow \mathsf{NE.D}(K_i,n,c)$
$K_u \leftarrow\$ \mathsf{NE.K}$		Return m

Fig. 36. Game defining multi-user \$CCA security of nonce-based encryption.

rigorously verified by plugging the code of $\mathsf{G}^{\text{mu-ae-m}}_{\mathsf{NE},b}$ into $\mathcal{C}_a$ and comparing side-by-side with L_{1-b}.) This gives $\Pr[\mathsf{L}_1] \leqslant \Pr[\mathsf{L}_0] + \mathsf{Adv}^{\text{mu-ae-m}}_{\mathsf{NE}}(\mathcal{C}_a)$.

Transition L_1 to L_2. In hybrid L_2, we undo the code changes used to transition from H_3 to L_0. Namely, l is set back to $\mathbb{I}$ and the if statement in Dec is reverted. By the same logic as that prior transition, $\Pr[\mathsf{L}_2] \leqslant \Pr[\mathsf{L}_1] + 2q_{\text{Enc}}q_{\text{Dec}}/2^\tau$.

Transition L_2 to L_3. In hybrid L_3, we undo the code changes used to transition from H_1 to H_2. Namely, we get rid of the injection f and sample c_{k} uniformly. By the same logic as that prior transition, $\Pr[\mathsf{L}_3] \leqslant \Pr[\mathsf{L}_2] + 0.5 \cdot q^2_{\text{Enc}}/2^\tau$.

Final Transition. Finally, we claim that L_3 and $\mathsf{G}^{\text{ae-m}}_{\mathsf{KD}[\mathsf{NE_k},\mathsf{NE_d}],0}$ are identical. Recall that in the latter, Enc always returns random ciphertexts. The same holds in L_3. Similarly, the two games have the same behavior in only responding with non-$\bot$ values to Dec queries for ciphertexts that were trivially forwarded from earlier Enc queries. In particular, we can see this by noting that $M[l(n,c_{\mathsf{k}}),c_{\mathsf{d}}]$ in L_3 always has the same value as $M[n,(c_{\mathsf{k}},c_{\mathsf{d}})]$ in $\mathsf{G}^{\text{ae-m}}_{\mathsf{KD}[\mathsf{NE_k},\mathsf{NE_d}],0}$ and that $M[l(n,c_{\mathsf{k}}),c_{\mathsf{d}}] \neq \bot$ implies $Y[l(n,c_{\mathsf{k}})] \neq \bot$. □

B.2 Public-key KEM/DEM

KEM/DEM scheme. Let NE be a nonce-based encryption scheme. Let PKE be a public-key encryption scheme. Then we define the KEM/DEM public-key encryption scheme $\mathsf{KD}[\mathsf{PKE},\mathsf{NE}]$ as shown in Fig. 37.

Multi-user \$CCA security of NE. For our proof we will again require that NE provide multi-user security against attacks making one encryption query. For this proof we will use a \$CCA-style definition rather than AE. We use the games in Fig. 36 to define the multi-user \$CCA (mu\$CCA) security of NE. If differs from muAE only in that the decryption oracle always returns honest decryptions of ciphertexts that were not trivially forwarded. For $w \in \{\mathtt{m},\diamond,\bot\}$, the advantage against multi-user AE security of a scheme NE is defined as $\mathsf{Adv}^{\text{mu-\$cca-}w}_{\mathsf{NE}}(\mathcal{A}) = \Pr[\mathsf{G}^{\text{mu-\$cca-}w}_{\mathsf{NE},1}(\mathcal{A})] - \Pr[\mathsf{G}^{\text{mu-ae-}w}_{\mathsf{NE},0}(\mathcal{A})]$.

Uniformity of PKE. For our proof we will additionally need to make the statistical assumption that the distribution of sampling ciphertexts randomly from $\mathsf{PKE.C}(ek,\mathsf{NE.kl})$ is close (in statistical distance) to the distribution obtained by encrypting a random K. Formally, we say that PKE is (κ,ϵ)-*uniform* if for all $(ek,dk) \in \mathsf{PKE.K}$ and all (not necessarily efficient) $\mathcal{A}$ it holds that

$$\Pr[\mathcal{A}(c) = 1 : c \leftarrow\$ \mathsf{PKE.C}(ek,\kappa)] - \Pr[\mathcal{A}(c) = 1 : K \leftarrow\$ \{0,1\}^\kappa; c \leftarrow\$ \mathsf{PKE.E}(ek,K)] \leqslant \epsilon.$$

As one example, this will hold with $\epsilon = 0$ for schemes in which $\mathsf{PKE.E}(ek,K;\cdot)$ is injective for all $K \in \{0,1\}^\kappa$ and $\mathsf{PKE.C}(ek,\kappa) = \{\mathsf{PKE.E}(ek,K;r)\}_{K,r}$.

Security result. The following theorem captures that $\mathsf{KD}[\mathsf{PKE},\mathsf{NE}]$ can be proven \$CCA-$\mathtt{m}$ secure from the security of PKE and NE, where all reductions are memory-tight.

Theorem 9. *Let* NE *be a nonce-based encryption scheme with* $\mathsf{NE.cl}(n) \geqslant n + \mathsf{NE.xl}$ *for all* n *and let* PKE *be a* $(\mathsf{NE.kl},\epsilon)$*-uniform public key encryption scheme. Let* τ *satisfy* $|\mathsf{PKE.C}(ek,\mathsf{NE.kl})| \geqslant 2^\tau$ *for all* ek *and* $\mathsf{NE.xl} \geqslant \tau$. *Let* $\mathcal{A}_c$ *be an* \$CCA-$\mathtt{m}$ *adversary with* $\mathbf{Query}(\mathcal{A}_c) = (q_{\text{Enc}},q_{\text{Dec}})$ *and assume* $q_{\text{Enc}} \leqslant 0.5 \cdot 2^\tau$.

$\mathsf{KD[PKE,NE].K}$	$\mathsf{KD[PKE,NE].E}(ek, m)$	$\mathsf{KD[PKE,NE].D}(dk, c)$
$(ek, dk) \leftarrow\$ \mathsf{PKE.K}$ Return (ek, dk)	$K \leftarrow\$ \mathsf{NE.K}$ $c_\mathsf{k} \leftarrow\$ \mathsf{PKE.E}(ek, K)$ $c_\mathsf{d} \leftarrow \mathsf{NE.E}(K, 0, m)$ Return $(c_\mathsf{k}, c_\mathsf{d})$	$(c_\mathsf{k}, c_\mathsf{d}) \leftarrow c$ $K \leftarrow \mathsf{PKE.D}(dk, c_\mathsf{k})$ If $K \neq \bot$: Return $\mathsf{NE.D}(K, 0, c_\mathsf{d})$ Return $\bot$

Fig. 37. A public-key scheme $\mathsf{KD[PKE,NE]}$ constructed from a public key encryption scheme PKE and a nonce-based encryption scheme NE.

Hybrids H_h for $0 \leqslant h \leqslant 3$	$\text{ENC}(m)$	$\text{DEC}(c)$

$f \leftarrow\$ \mathsf{Inj}(T, D, R)$ //$\mathsf{H}_{[2,\infty)}$
$l \leftarrow \mathbb{I}$
$u \leftarrow 0$ //$\mathsf{H}_{[2,\infty)}$
$(ek, dk) \leftarrow\$ \mathsf{PKE.K}$
$b' \leftarrow \mathcal{A}_c^{\text{ENC},\text{DEC}}(ek)$
Return $b' = 1$
	$u \leftarrow u + 1$ //$\mathsf{H}_{[2,\infty)}$
	$K \leftarrow\$ \mathsf{NE.K}$
	$c_\mathsf{k} \leftarrow\$ \mathsf{PKE.E}(ek, K)$ //$\mathsf{H}_{[0,1)}$
	$c_\mathsf{k} \leftarrow\$ \mathsf{PKE.C}(ek, \lvert K\rvert)$ //$\mathsf{H}_{[1,2)}$
	$c_\mathsf{k} \leftarrow f(ek, u)$ //$\mathsf{H}_{[2,\infty)}$
	$c_\mathsf{d} \leftarrow \mathsf{NE.E}(K, 0, m)$
	$Y[l(ek, c_\mathsf{k})] \leftarrow K$
	$M[l(ek, c_\mathsf{k}), c_\mathsf{d}] \leftarrow m$ //$\mathsf{H}_{[3,\infty)}$
	Return $(c_\mathsf{k}, c_\mathsf{d})$
		$(c_\mathsf{k}, c_\mathsf{d}) \leftarrow c$
		If $Y[l(ek, c_\mathsf{k})] \neq \bot$:
		  If $M[l(ek, c_\mathsf{k}), c_\mathsf{d}] \neq \bot$: //$\mathsf{H}_{[3,\infty)}$
		    Return $M[l(ek, c_\mathsf{k}), c_\mathsf{d}]$ //$\mathsf{H}_{[3,\infty)}$
		  $K \leftarrow Y[l(ek, c_\mathsf{k})]$
		  Return $\mathsf{NE.D}(K, 0, c_\mathsf{d})$
		Else:
		  $K \leftarrow \mathsf{PKE.D}(dk, c_\mathsf{k})$
		  If $K \neq \bot$:
		    Return $\mathsf{NE.D}(K, 0, c_\mathsf{d})$
		Return $\bot$

Fig. 38. First set of hybrids used for proof of Theorem 9. $\mathbb{I}$ denotes the identity function.

Define $T = \mathsf{PKE.Ek}$ *and* $(D_{ek}, R_{ek}) = ([q_{\text{ENC}}], \mathsf{PKE.C}(ek, \mathsf{NE.kl}))$ *for* $ek \in T$. *Define* $T' = \{0,1\}^* \times \mathbb{N}$, $D'_{(c_\mathsf{k},l)} = \{0,1\}^l$, *and* $R'_{(c_\mathsf{k},l)} = \{0,1\}^{\mathsf{NE.cl}(l)}$. *Let* $\mathcal{B}_c$ *be as defined in Fig. 39, let* $\mathcal{C}_a$ *be the* $\mathsf{Inj}^{\pm}(T, D, R)$*-oracle adversary defined in Fig. 41, and let* $\mathcal{E}_c$ *be the* $\mathsf{Inj}^{\pm}(T', D', R')$*-oracle adversary defined in Fig. 43. Then,*

$$\mathsf{Adv}^{\$\mathsf{cca}\text{-}m}_{\mathsf{KD[PKE,NE]}}(\mathcal{A}_c) \leqslant \mathsf{Adv}^{\$\mathsf{cca}\text{-}m}_{\mathsf{PKE}}(\mathcal{B}_c) + \mathsf{Adv}^{\mathsf{mu}\text{-}\$\mathsf{cca}\text{-}m}_{\mathsf{NE}}(\mathcal{C}_a) + \mathsf{Adv}^{\mathsf{cca}\text{-}m}_{\mathsf{PKE}}(\mathcal{E}_c) + 2q_{\text{ENC}} \cdot \epsilon + (2.5 \cdot q_{\text{ENC}}^2 + 4q_{\text{DEC}})/2^{\tau}$$

$$\begin{aligned} \mathbf{Query}(\mathcal{B}_c) &= (q_{\text{ENC}}, q_{\text{DEC}}) \\ \mathbf{Time}(\mathcal{B}_c) &= \mathbf{Time}(\mathcal{A}_c) + (q_{\text{ENC}} + q_{\text{DEC}})\mathbf{Time}(\mathsf{NE}) \\ \mathbf{Mem}(\mathcal{B}_c) &= \mathbf{Mem}(\mathcal{A}_c) + \mathbf{Mem}(\mathsf{NE}). \end{aligned}$$

$$\begin{aligned} \mathbf{Query}(\mathcal{C}_a) &= (q_{\text{ENC}}, q_{\text{ENC}}, q_{\text{DEC}}) \\ \mathbf{Time}^*(\mathcal{C}_a) &= \mathbf{Time}(\mathcal{A}_c) \\ \mathbf{Mem}^*(\mathcal{C}_a) &= \mathbf{Mem}(\mathcal{A}_c). \end{aligned}$$

$$\begin{aligned} \mathbf{Query}(\mathcal{E}_c) &= q_{\text{ENC}} + q_{\text{DEC}} \\ \mathbf{Time}(\mathcal{E}_c) &= \mathbf{Time}(\mathcal{A}_c) + (q_{\text{ENC}} + q_{\text{DEC}})\mathbf{Time}(\mathsf{NE}) \\ \mathbf{Mem}(\mathcal{E}_c) &= \mathbf{Mem}(\mathcal{A}_c) + \mathbf{Mem}(\mathsf{NE}). \end{aligned}$$

Proof. We consider the hybrids H_0 through H_3 and L_0 through L_3 defined in Figs. 38 and 40. Of these hybrids we the following claims establish the upper bound on the advantage of $\mathcal{A}_c$ claimed in the proof.

Adversary $\mathcal{B}_c^{\text{Enc},\text{Dec}}(ek)$	SimEnc(m)	SimDec(c)
$b' \leftarrow \mathcal{A}_c^{\text{SimEnc},\text{SimDec}}(ek)$ Return b'	$K \leftarrow\$ \mathsf{NE.K}$ $c_\mathsf{k} \leftarrow \text{Enc}(K)$ $c_\mathsf{d} \leftarrow \mathsf{NE.E}(K, 0, m)$ Return $(c_\mathsf{k}, c_\mathsf{d})$	$(c_\mathsf{k}, c_\mathsf{d}) \leftarrow c$ $K \leftarrow \text{Dec}(c_\mathsf{k})$ If $K \neq \bot$: Return $\mathsf{NE.D}(K, 0, c_\mathsf{d})$ Return $\bot$

Fig. 39. Adversary $\mathcal{B}_c$ for Theorem 9.

1. $\Pr[\mathsf{G}^{\text{\$cca-m}}_{\mathsf{KD[PKE,NE]},1}(\mathcal{A}_c)] = \Pr[\mathsf{H}_0]$
2. $\Pr[\mathsf{H}_0] \leqslant \Pr[\mathsf{H}_1] + \mathsf{Adv}^{\text{\$cca-m}}_{\mathsf{PKE}}(\mathcal{B}_c)$
3. $\Pr[\mathsf{H}_1] \leqslant \Pr[\mathsf{H}_2] + 0.5 \cdot q_{\text{Enc}}^2/2^\tau$
4. $\Pr[\mathsf{H}_2] = \Pr[\mathsf{H}_3] = \Pr[\mathsf{L}_0]$
5. $\Pr[\mathsf{L}_0] \leqslant \Pr[\mathsf{L}_1] + \mathsf{Adv}^{\text{mu-\$cca-m}}_{\mathsf{NE}}(\mathcal{C}_a)$
6. $\Pr[\mathsf{L}_1] = \Pr[\mathsf{L}_2] = \Pr[\mathsf{L}_3]$
7. $\Pr[\mathsf{L}_3] \leqslant \Pr[\mathsf{L}_4] + 0.5 \cdot q_{\text{Enc}}^2/2^\tau$
8. $\Pr[\mathsf{L}_4] = \Pr[\mathsf{M}_0]$
9. $\Pr[\mathsf{M}_0] \leqslant \Pr[\mathsf{M}_1] + 0.5 \cdot q_{\text{Enc}}^2/2^\tau$
10. $\Pr[\mathsf{M}_1] \leqslant \Pr[\mathsf{M}_2] + 2q_{\text{Dec}}/2^\tau$
11. $\Pr[\mathsf{M}_2] \leqslant \Pr[\mathsf{M}_3] + q_{\text{Enc}} \cdot \epsilon$
12. $\Pr[\mathsf{M}_3] \leqslant \Pr[\mathsf{M}_4] + \mathsf{Adv}^{\text{cca-m}}_{\mathsf{PKE}}(\mathcal{E}_c)$
13. $\Pr[\mathsf{M}_4] \leqslant \Pr[\mathsf{M}_5] + q_{\text{Enc}} \cdot \epsilon$
14. $\Pr[\mathsf{M}_5] \leqslant \Pr[\mathsf{M}_6] + 2q_{\text{Dec}}/2^\tau$
15. $\Pr[\mathsf{M}_6] \leqslant \Pr[\mathsf{M}_7] + 0.5 \cdot q_{\text{Enc}}^2/2^\tau$
16. $\Pr[\mathsf{M}_7] \leqslant \Pr[\mathsf{M}_8] + 0.5 \cdot q_{\text{Enc}}^2/2^\tau$
17. $\Pr[\mathsf{M}_8] = \Pr[\mathsf{G}^{\text{\$cca-m}}_{\mathsf{KD[PKE,NE]},0}(\mathcal{A}_c)]$

Transition to H_0. We claim that H_0 is identical to $\mathsf{G}^{\text{\$cca-m}}_{\mathsf{KD[PKE,NE]},1}$. Note that, in the latter, Enc produces honest encryptions using KD and Dec produces honest decryptions. It is immediately clear that the same holds for Enc in H_0 and the else branch of Dec in H_0. Consider the first branch in Dec. We have used grey highlighting to indicate the relevant code. Note that l is the identity function $\mathbb{I}$ (its presence will be notationally convenient for future game transitions). Y is a table indexed by ek, c_k pairs which stores the key encrypted in c_k. The use of Y in Dec simply recovers this K without decrypting c_k. By the correctness of PKE this is identical to having done the decryption, so $\Pr[\mathsf{G}^{\text{\$cca-m}}_{\mathsf{KD[PKE,NE]},1}(\mathcal{A}_c)] = \Pr[\mathsf{H}_0]$ follows.

Transition H_0 to H_1. In hybrid H_1, the ciphertext c_k is now sampled uniformly at random. This difference corresponds to what we expect from the \$CCA-m security of PKE. We define $\mathcal{B}_c$ in Fig. 39 from hybrid H_1 by replacing encryption and decryption of K with appropriate queries to its oracle. Note that the use of table Y in the hybrids is similar to the table M from $\mathsf{G}^{\text{\$cca-m}}$ (in Y all entries are additionally indexed by the same ek). We can see that when interacting with $\mathsf{G}^{\text{\$cca-m}}_{\mathsf{PKE},1}$, $\mathcal{B}_c$ simulates H_0 to $\mathcal{A}_c$ and when interacting with $\mathsf{G}^{\text{\$cca-m}}_{\mathsf{PKE},0}$, $\mathcal{B}_c$ simulates H_1 to $\mathcal{A}_c$. Hence $\Pr[\mathsf{H}_0] \leqslant \Pr[\mathsf{H}_1] + \mathsf{Adv}^{\text{\$cca-m}}_{\mathsf{PKE}}(\mathcal{B}_c)$.

Transition H_1 to H_2. In hybrid H_2, instead of sampling c_k at random, we assign it the output of a random injective function f applied to the encryption key ek as the tweak and a counter u (we will later make u correspond to users in $\mathsf{G}^{\text{mu-ae-m}}$). The switching lemma tells us that $\Pr[\mathsf{H}_1] \leqslant \Pr[\mathsf{H}_2] + 0.5 \cdot q_{\text{Enc}}^2/2^\tau$.

Transition H_2 to H_3. In hybrid H_3, we introduce a table M which is indexed by ek, c_k, c_d (the first two via l) and stores the value of m whose encryption under $K = Y[l(ek, c_\mathsf{k})]$ is c_d. This table is used in Dec to skip the step of decrypting c_d. By the correctness of NE this does not change functionality, so $\Pr[\mathsf{H}_2] = \Pr[\mathsf{H}_3]$.

Transition H_3 to L_0. Next we transition to hybrid L_0 shown in Fig. 40. We have highlighted all ways that this differs from hybrid H_3. Our changes were twofold, consisting of switching the function l indexing into our tables to f^{-1} and switching the if condition in Dec. Considering the latter first, note that the checks $Y[l(ek, c_\mathsf{k})] \neq \bot$ in H_3 and $1 \leqslant f^{-1}(ek, c_\mathsf{k}) \leqslant u$ in L_0 are identical. Note that $Y[l(ek, c_\mathsf{k})] \neq \bot$ iff c_k was returned by the u'-th Enc query for some $1 \leqslant u' \leqslant u$ which holds iff $f^{-1}(ek, c_\mathsf{k}) = u'$. Switching to using $l = f^{-1}$ cannot change behavior since $(ek, c_\mathsf{k}) = (ek, c'_\mathsf{k})$ iff $f^{-1}(ek, c_\mathsf{k}) = f^{-1}(ek, c'_\mathsf{k}) \neq \bot$. Thus $\Pr[\mathsf{H}_3] = \Pr[\mathsf{L}_0]$.

Transition L_0 to L_1. In hybrid L_1, the ciphertext c_d is now sampled uniformly at random. We define $\mathcal{C}_a$ in Fig. 41 from hybrid L_1 by replacing encryption and decryption of c_d with appropriate queries to its oracle. We claim that when interacting with $\mathsf{G}^{\text{mu-\$cca-m}}_{\mathsf{NE},1}$, $\mathcal{C}_a$ simulates L_0 to $\mathcal{A}_c$ and when interacting with $\mathsf{G}^{\text{mu-\$cca-m}}_{\mathsf{NE},0}$,

Hybrids L_ℓ for $0 \leqslant \ell \leqslant 4$	$\textsc{Enc}(m)$	$\textsc{Dec}(c)$
$f \leftarrow\$ \mathsf{Inj}^{\pm}(T, D, R)$ // $\mathsf{L}_{[0,4)}$	$u \leftarrow u + 1$ // $\mathsf{L}_{[0,4)}$	$(c_\mathsf{k}, c_\mathsf{d}) \leftarrow c$
$g \leftarrow\$ \mathsf{Fcs}(T', D', R')$ // $\mathsf{L}_{[2,\infty)}$	$K \leftarrow\$ \mathsf{NE.K}$	If $1 \leqslant f^{-1}(ek, c_\mathsf{k}) \leqslant u$: // $\mathsf{L}_{[0,3)}$
$l \leftarrow f^{-1}$ // $\mathsf{L}_{[0,3)}$	$c_\mathsf{k} \leftarrow f(ek, u)$ // $\mathsf{L}_{[0,4)}$	If $Y[l(ek, c_\mathsf{k})] \neq \bot$: // $\mathsf{L}_{[3,\infty)}$
$l \leftarrow \mathbb{I}$ // $\mathsf{L}_{[3,\infty)}$	$c_\mathsf{k} \leftarrow\$ \mathsf{PKE.C}(ek, \lvert K \rvert)$ // $\mathsf{L}_{[4,\infty)}$	If $M[l(ek, c_\mathsf{k}), c_\mathsf{d}] \neq \bot$:
$u \leftarrow 0$ // $\mathsf{L}_{[0,4)}$	$c_\mathsf{d} \leftarrow \mathsf{NE.E}(K, 0, m)$ // $\mathsf{L}_{[0,1)}$	Return $M[l(ek, c_\mathsf{k}), c_\mathsf{d}]$
$(ek, dk) \leftarrow\$ \mathsf{PKE.K}$	$c_\mathsf{d} \leftarrow\$ \{0,1\}^{\mathsf{NE.cl}(\lvert m \rvert)}$ // $\mathsf{L}_{[1,2)}$	$K \leftarrow Y[l(ek, c_\mathsf{k})]$
$b' \leftarrow \mathcal{A}_a^{\textsc{Enc},\textsc{Dec}}(ek)$	$c_\mathsf{d} \leftarrow g((c_\mathsf{k}, \lvert m \rvert), m)$ // $\mathsf{L}_{[2,\infty)}$	Return $\mathsf{NE.D}(K, 0, c_\mathsf{d})$
Return $b' = 1$	$Y[l(ek, c_\mathsf{k})] \leftarrow K$	Else:
	$M[l(ek, c_\mathsf{k}), c_\mathsf{d}] \leftarrow m$	$K \leftarrow \mathsf{PKE.D}(dk, c_\mathsf{k})$
	Return $(c_\mathsf{k}, c_\mathsf{d})$	If $K \neq \bot$:
		Return $\mathsf{NE.D}(K, 0, c_\mathsf{d})$
		Return $\bot$

Fig. 40. Second set of hybrids used for proof of Theorem 9.

Adversary $\mathcal{C}_a^{\textsc{New},\textsc{Enc},\textsc{Dec}}$	$\textsc{SimEnc}(m)$	$\textsc{SimDec}(c)$
$f \leftarrow\$ \mathsf{Inj}^{\pm}(T, D, R)$	$u \leftarrow u + 1$	$(c_\mathsf{k}, c_\mathsf{d}) \leftarrow c$
$u \leftarrow 0$	$\textsc{New}()$	$i \leftarrow f^{-1}(ek, c_\mathsf{k})$
$(ek, dk) \leftarrow\$ \mathsf{PKE.K}$	$c_\mathsf{k} \leftarrow f(ek, u)$	If $1 \leqslant i \leqslant u$:
$b' \leftarrow \mathcal{A}_c^{\textsc{SimEnc},\textsc{SimDec}}(ek)$	$c_\mathsf{d} \leftarrow \textsc{Enc}(u, 0, m)$	Return $\textsc{Dec}(i, 0, c_\mathsf{d})$
Return b'	Return $(c_\mathsf{k}, c_\mathsf{d})$	Else:
		$K \leftarrow \mathsf{PKE.D}(dk, c_\mathsf{k})$
		If $K \neq \bot$:
		Return $\mathsf{NE.D}(K, 0, c_\mathsf{d})$
		Return $\bot$

Fig. 41. Adversary $\mathcal{C}_a$ for Theorem 9.

$\mathcal{C}_a$ simulates L_1 to $\mathcal{A}_c$. To see this, note that the variable u in these hybrids matches the same variable in $\mathsf{G}^{\mathsf{mu\text{-}\$cca\text{-}m}}$. Additionally, if $i = l(ek, c_\mathsf{k})$ then the values $Y[i]$ and $M[i, c_\mathsf{d}]$ in these hybrids always match K_i and $M[i, 0, c_\mathsf{d}]$ in $\mathsf{G}^{\mathsf{mu\text{-}\$cca\text{-}m}}$. (The claims can be rigorously verified by plugging the code of $\mathsf{G}^{\mathsf{mu\text{-}\$cca\text{-}m}}_{\mathsf{NE},b}$ into $\mathcal{C}_a$ and comparing side-by-side with L_{1-b}.) This gives $\Pr[\mathsf{L}_0] \leqslant \Pr[\mathsf{L}_1] + \mathsf{Adv}^{\mathsf{mu\text{-}\$cca\text{-}m}}_{\mathsf{NE}}(\mathcal{C}_a)$.

Transition L_1 to L_2. In hybrid L_2, the ciphertext c_d is now the output of a random function g from $\mathsf{Fcs}(T', D', R')$, rather than being sampled uniformly. (Recall $T' = \{0,1\}^* \times \mathbb{N}$, $D'_{(c_\mathsf{k}, l)} = \{0,1\}^l$, and $R'_{(c_\mathsf{k}, l)} = \{0,1\}^{\mathsf{NE.cl}(l)}$.) The inputs to g never repeat (because c_k never repeats), so this does not modify the behavior of the game, giving $\Pr[\mathsf{L}_1] = \Pr[\mathsf{L}_2]$.

Transition L_2 to L_3. In hybrid L_3, we undo the code changes used to transition from H_3 to L_0. Namely, l is set back to $\mathbb{I}$ and the if statement in Dec is reverted. By the same logic as that prior transition, $\Pr[\mathsf{L}_2] = \Pr[\mathsf{L}_3]$.

Transition L_3 to L_4. In hybrid L_4, we undo the code changes used to transition from H_1 to H_2. Namely, we get rid of the injection f and sample c_k uniformly. By the same logic as that prior transition, $\Pr[\mathsf{L}_3] \leqslant \Pr[\mathsf{L}_4] + 0.5 \cdot q_{\textsc{Enc}}^2 / 2^{\tau}$.

Transition L_4 to M_0. Next we transition to the game M_0 shown in Fig 42. In this game we made simplifications to L_4 to prepare us for our final transitions. In particular, we removed the labelling function l and instead index into tables Y and M directly. We additionally drop the use of ek in this indexing. Then the code in Dec was rewritten for ease of comparison to the final game we are trying to reach (M_8, which we will show is equivalent to $\mathsf{G}^{\mathsf{\$cca\text{-}m}}_{\mathsf{KD[PKE,NE]},0}(\mathcal{A}_c)$). Because $M[ek, c_\mathsf{k}, c_\mathsf{d}] \neq \bot$ implies $Y[ek, c_\mathsf{k}] \neq \bot$, we separated

Hybrids M_ℓ for $0 \leqslant \ell \leqslant 8$	$\textsc{Enc}(m)$	$\textsc{Dec}(c)$
$(ek, dk) \leftarrow\$ \mathsf{PKE.K}$	$K \leftarrow\$ \mathsf{NE.K}$	$(c_\mathsf{k}, c_\mathsf{d}) \leftarrow c$
$g \leftarrow\$ \mathsf{Fcs}(T', D', R')$ $//\mathsf{M}_{[0,1)}, \mathsf{M}_{[7,8)}$	$c_\mathsf{k} \leftarrow\$ \mathsf{PKE.C}(ek, \lvert K\rvert)$ $//\mathsf{M}_{[0,3)}, \mathsf{M}_{[5,\infty)}$	If $M[c_\mathsf{k}, c_\mathsf{d}] \neq \bot$: $//\mathsf{M}_{[0,2)}, \mathsf{M}_{[6,\infty)}$
$g \leftarrow\$ \mathsf{Inj}(T', D', R')$ $//\mathsf{M}_{[1,7)}$	$K' \leftarrow\$ \mathsf{NE.K}$ $//\mathsf{M}_{[3,5)}$	Return $M[c_\mathsf{k}, c_\mathsf{d}]$ $//\mathsf{M}_{[0,2)}, \mathsf{M}_{[6,\infty)}$
$b' \leftarrow \mathcal{A}_c^{\textsc{Enc},\textsc{Dec}}(ek)$	$c_\mathsf{k} \leftarrow\$ \mathsf{PKE.E}(ek, K')$ $//\mathsf{M}_{[3,5)}$	$m \leftarrow g^{-1}((c_\mathsf{k}, \lvert c_\mathsf{d}\rvert - \mathsf{NE.xl}), c_\mathsf{d})$ $//\mathsf{M}_{[2,6)}$
Return $b' = 1$	$c_\mathsf{d} \leftarrow g((c_\mathsf{k}, \lvert m\rvert), m)$ $//\mathsf{M}_{[0,8)}$	If $m \neq \bot$: Return m $//\mathsf{M}_{[2,6)}$
	$c_\mathsf{d} \leftarrow\$ \{0,1\}^{\mathsf{NE.cl}(\lvert m\rvert)}$ $//\mathsf{M}_{[8,\infty)}$	If $Y[c_\mathsf{k}] \neq \bot$:
	$Y[c_\mathsf{k}] \leftarrow K$	$K \leftarrow Y[c_\mathsf{k}]$ $//\mathsf{M}_{[0,4)}$
	$M[c_\mathsf{k}, c_\mathsf{d}] \leftarrow m$	$K \leftarrow \mathsf{PKE.D}(dk, c_\mathsf{k})$ $//\mathsf{M}_{[4,\infty)}$
	Return $(c_\mathsf{k}, c_\mathsf{d})$	Else:
		$K \leftarrow \mathsf{PKE.D}(dk, c_\mathsf{k})$
		If $K \neq \bot$:
		Return $\mathsf{NE.D}(K, 0, c_\mathsf{d})$
		Return $\bot$

Fig. 42. Third set of hybrids used for proof of Theorem 9. Hybrids after M_4 are mostly undoing prior transitions.

out the if statement which checks M, rather than leaving it nested inside of the check for Y. Rather than repeating the code which runs $\mathsf{NE.D}$ in two separate branches we consolidated to be run at the end of the oracle. None of these changes modify the behavior of the game, so $\Pr[\mathsf{L}_4] = \Pr[\mathsf{M}_0]$.

Note that the final game M_8 we are moving toward differs from this game primarily in that, for decryption queries when $M[ek, c_\mathsf{k}, c_\mathsf{d}] = \bot$ but $Y[ek, c_\mathsf{k}] \neq \bot$ game M_0 uses the key stored in Y (which was chosen at random) to decrypt c_d while game M_8 will used whatever key is encrypted in c_k. These extra transition were not needed in our secret-key KEM/DEM proof because the final game we were transitioning to in that proof returned $\bot$ for *any* decryption query not directly forwarded from encryption.

Transition M_0 to M_1. To transition to M_1 we sample g as an injection, rather than a function. The switching lemma tells us that $\Pr[\mathsf{M}_0] \leqslant \Pr[\mathsf{M}_1] + 0.5 \cdot q_{\textsc{Enc}}^2/2^\tau$.

Transition M_1 to M_2. In M_2 we replace the use of the table M with g^{-1} in $\textsc{Dec}$. These games will differ only if the adversary makes a decryption query for $(c_\mathsf{k}, c_\mathsf{d})$ where $M[c_\mathsf{k}, c_\mathsf{d}] = \bot$ and $g^{-1}((c_\mathsf{k}, |c_\mathsf{d}| - \mathsf{NE.xl}), c_\mathsf{d}) \neq \bot$. We can bound the probability of such a query in M_1. To make such a query, the adversary must be "guessing" a new point in the image of $g^{-1}((c_\mathsf{k}, |c_\mathsf{d}| - \mathsf{NE.xl}), \cdot)$ other than the at most $q_{\textsc{Enc}}$ examples it may have been given from $\textsc{Enc}$. Note that in M_1, these examples from $\textsc{Enc}$ are the only way that g affects the behavior of the game. Note that the range of $g^{-1}((c_\mathsf{k}, |c_\mathsf{d}| - \mathsf{NE.xl}), \cdot)$ has size $2^{|c_\mathsf{d}|}$ and its image has size $2^{|c_\mathsf{d}| - \mathsf{NE.xl}}$. Thus (using that $q_{\textsc{Enc}} \leqslant 0.5 \cdot 2^\tau$) any particular decryption query has probability at most $2^{|c_\mathsf{d}| - \mathsf{NE.xl}} / \left(2^{|c_\mathsf{d}|} - q_{\textsc{Enc}}\right) \leqslant 2/2^\tau$. Applying a union bound gives $\Pr[\mathsf{M}_1] \leqslant \Pr[\mathsf{M}_2] + 2q_{\textsc{Dec}}/2^\tau$.

Transition M_2 to M_3. Now in M_3 we sample c_k as the encryption of a random key K' rather than choosing it uniformly at random. The key K' will thus be the key obtained if c_k is decrypted in $\textsc{Dec}$. A standard hybrid argument using the $(\mathsf{NE.kl}, \epsilon)$-uniformity of PKE gives that $\Pr[\mathsf{M}_2] \leqslant \Pr[\mathsf{M}_3] + q_{\textsc{Enc}} \cdot \epsilon$.

Transition M_3 to M_4. Next we change the behavior of the decryption oracle when $Y[c_\mathsf{k}] \neq \bot$ and $g^{-1}((c_\mathsf{k}, |c_\mathsf{d}| - \mathsf{NE.xl}), c_\mathsf{d}) = \bot$. In M_3, we used the key $Y[c_\mathsf{k}]$ to decrypt c_d. In M_4, we use the decryption of c_k (which was called K' when originally sampled in $\textsc{Enc}$). This difference actually corresponds to the CCA security of PKE.

Consider the adversary $\mathcal{E}_c$ shown in Fig. 43. It was obtained by modifying the code of hybrids M_3 and M_4 to query its own encryption oracle to obtain c_k when responding to encryption queries and to use its own decryption oracle to obtain K when responding to and query for which it cannot obtain its response using g^{-1}. We claim that the view of $\mathcal{A}_c$ in M_{3+b} perfectly matches its view when simulated by $\mathcal{E}_c$ in $\mathsf{G}^{\mathsf{cca\text{-}m}}_{\mathsf{PKE},b}$. In particular, K sampled during encryption in M_{3+b} "matches" the K_{1-b} sampled for encryption in $\mathsf{G}^{\mathsf{cca\text{-}m}}_{\mathsf{PKE},b}(\mathcal{E}_c)$ and K' matches K_b. Note that K_b/K' is always the key encrypted in c_k. When $b = 0$, a later decryption query using c_k will use the key stored in $Y[c_\mathsf{k}]$ which is K_1/K. When $b = 1$, a later decryption query using

Adversary $\mathcal{E}_c^{\text{Enc},\text{Dec}}(ek)$	SimEnc(m)	SimDec(c)
$g \leftarrow\$ \mathsf{Inj}(T', D', R')$ $b' \leftarrow \mathcal{A}_c^{\text{SimEnc},\text{SimDec}}(ek)$ Return $1 - b'$	$K_0 \leftarrow\$ \mathsf{NE.K}$ $K_1 \leftarrow\$ \mathsf{NE.K}$ $c_\mathsf{k} \leftarrow \text{Enc}(K_0, K_1)$ $c_\mathsf{d} \leftarrow g((c_\mathsf{k}, \lvert m\rvert), m)$ Return $(c_\mathsf{k}, c_\mathsf{d})$	$(c_\mathsf{k}, c_\mathsf{d}) \leftarrow c$ $m \leftarrow g^{-1}((c_\mathsf{k}, \lvert c_\mathsf{d}\rvert - \mathsf{NE.xl}), c_\mathsf{d})$ If $m \neq \bot$: Return m $K \leftarrow \text{Dec}(c_\mathsf{k})$ If $K \neq \bot$: Return $\mathsf{NE.D}(K, 0, c_\mathsf{d})$ Return $\bot$

Fig. 43. Adversary $\mathcal{E}_c$ for Theorem 9.

c_k will use the key actually encrypted by c_k which is K_1/K' (in $\mathsf{G}_{\mathsf{PKE},1}$ this value is stored in Y, in M_4 this is obtained by encryption).

Now, noting that $\mathcal{E}_c$ returns the opposite bit of what $\mathcal{A}_c$ returned we have that $\Pr[\mathsf{M}_{3+b}] = 1 - \Pr[\mathsf{G}_{\mathsf{PKE},b}(\mathcal{E}_c)]$ and so $\Pr[\mathsf{M}_3] \leqslant \Pr[\mathsf{M}_4] + \mathsf{Adv}^{\mathsf{cca\text{-}m}}_{\mathsf{PKE}}(\mathcal{E}_c)$.

Transitions M_4 through M_8. The next couple transition all serve to undo prior transitions. In M_5, we switch back to c_k being sampled uniformly. By the same logic as our transition to M_3 we have $\Pr[\mathsf{M}_4] \leqslant \Pr[\mathsf{M}_5] + q_{\text{Enc}} \cdot \epsilon$. In M_6, we switch back to using the table M rather than g^{-1}. By analogous reasoning to our transition to M_2 we have $\Pr[\mathsf{M}_5] \leqslant \Pr[\mathsf{M}_6] + 2q_{\text{Dec}}/2^\tau$. In M_7, we switch from g being a random injection to a random function. By the switching lemma, $\Pr[\mathsf{M}_6] \leqslant \Pr[\mathsf{M}_7] + 0.5 \cdot q_{\text{Enc}}^2/2^\tau$. In M_8, we switch back to c_d being sampled uniformly rather than using g. This give differing behavior only if c_k ever repeats, giving $\Pr[\mathsf{M}_7] \leqslant \Pr[\mathsf{M}_8] + 0.5 \cdot q_{\text{Enc}}^2/2^\tau$.

Final Transition. Finally, we conclude by observing that M_8 is identical to $\mathsf{G}^{\$\mathsf{cca\text{-}m}}_{\mathsf{KD}[\mathsf{PKE},\mathsf{NE}],0}(\mathcal{A}_c)$. In both Enc returns a uniformly random ciphertext. For a decryption query $\text{Dec}(c)$, if c was returned by an earlier encryption query, then the table M is used to return the message from that query. Otherwise, c is honestly decrypted as specified by $\mathsf{KD}[\mathsf{PKE},\mathsf{NE}]$. Hence, $\Pr[\mathsf{M}_8] = \Pr[\mathsf{G}^{\$\mathsf{cca\text{-}m}}_{\mathsf{KD}[\mathsf{PKE},\mathsf{NE}],0}(\mathcal{A}_c)]$. $\square$