A preliminary version of this paper appears in CRYPTO 2020. This is the full version.

Handling Adaptive Compromise for Practical Encryption Schemes

Joseph Jaeger[1] Nirvan Tyagi[2]

June 2020

Abstract

We provide a new definitional framework capturing the multi-user security of encryption schemes and pseudorandom functions in the face of adversaries that can adaptively compromise users' keys. We provide a sequence of results establishing the security of practical symmetric encryption schemes under adaptive compromise in the random oracle or ideal cipher model. The bulk of analysis complexity for adaptive compromise security is relegated to the analysis of lower-level primitives such as pseudorandom functions.

We apply our framework to give proofs of security for the BurnBox system for privacy in the face of border searches and the in-use searchable symmetric encryption scheme due to Cash et al. In both cases, prior analyses had bugs that our framework helps avoid.

[1] Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, USA. Email: jsjaeger@cs.washington.edu. URL: https://homes.cs.washington.edu/~jsjaeger/. Supported in part by NSF grant CNS-1717640, NSF grant CNS-1719146, and a Sloan Research Fellowship.

[2] Cornell Tech, New York, USA. Email: tyagi@cs.cornell.edu. URL: https://www.cs.cornell.edu/~tyagi/. Supported in part by NSF grant CNS-1704296.

Contents

1 Introduction

A classic question in cryptography has been dealing with adversaries that adaptively compromise particular parties, thereby learning their secrets. Consider a setting where parties use keys $k_1, \ldots, k_n$ to encrypt messages $m_1, \ldots, m_n$ to derive ciphertexts $\mathsf{Enc}(k_1, m_1), \ldots, \mathsf{Enc}(k_n, m_n)$. An adversary obtains the ciphertexts and compromises a chosen subset of the parties to learn their keys. What can we say about the security of the messages encrypted by the keys that remain secret? Surprisingly, traditional approaches to formal security analysis, such as using encryption schemes that provide semantic security [22], fail to suffice for proving these messages' confidentiality. This problem was first discussed in the context of secure multiparty computation [12], and it arises in a variety of important cryptographic applications, as we explain below.

In this work, we introduce a new framework for formal analyses when security in the face of adaptive compromise is desired. Our approach provides a modular route towards analysis using idealized primitives (such as random oracles or ideal ciphers) for practical and in-use schemes. This modularity helps us sidestep the pitfalls of prior ideal-model analyses that either invented new (less satisfying) ideal primitives, omitted proofs, or gave detailed but incorrect proofs. We exercise our framework across applications including searchable symmetric encryption (SSE), revocable cloud storage, and asymmetric password-authenticated key exchange (aPAKE). In particular, we provide full, correct proofs of security against adaptive adversaries for the Cash et al. [14] searchable symmetric encryption scheme that is used often in practice and the BurnBox system [36] for dealing with compelled-access searches. We show that our new definitions imply the notion of equivocable encryption introduced to prove security of the OPAQUE [26] asymmetric password-authenticated key exchange protocol. More broadly, our framework can be applied to a wide variety of constructions [2, 3, 11, 15, 19, 23, 24, 27–31, 37].

Current approaches to the "commitment problem". Our motivating applications have at their core an adaptive simulation-based model of security. Roughly speaking, they ask that no computationally bound adversary can distinguish between two worlds. In the first world, the adversary interacts with the scheme whose security is being measured. In the second world, the "ideal" world, the adversary's queries are instead handled by a simulator that must make do with only limited information which represents allowable "leakage" about the queries the adversary has made so far. The common unifying factor between varying applications we consider is that the adversary can make queries resulting in being given a ciphertexts encrypting messages of its choosing, then with future queries adaptively choose to expose the secret keys underlying some of the ciphertexts. The leakage given to the simulator will not include the messages encrypted unless a query has been made to expose the corresponding key.

Proving security in this model, however, does not work based on standard assumptions of the underlying encryption scheme. The problem is that the simulator must commit to ciphertexts, revealing them to the adversary, before knowing the messages associated to them. Hence the commitment problem. Several prior approaches for proving positive security results exist.

One natural approach attempts to build special non-committing encryption schemes [12] that can be proven (in the standard model) to allow opening some a priori fixed ciphertext to a message. But these schemes are not practical, as they require key material at least as long as the underlying message. Another unsatisfying approach considers only non-adaptive security in which an attacker specifies all of its queries at the beginning of the game. This is one of the two approaches that were simultaneously taken by Cash et al. [14]. Here the simulator is given the leakage for all of these queries at once and generates a transcript of all of its response. This is unsatisfying because more is lost when switching from adaptive to non-adaptive security than just avoiding the commitment

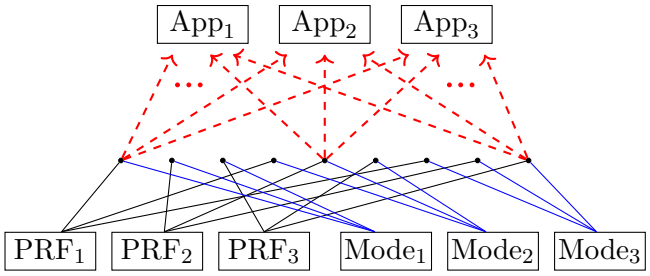

Figure 1: Old state of affairs. Red dashed lines correspond to implications proved through programming in an ideal model proof. A different programming proof is needed to prove an application secure for each pair of PRF and symmetric encryption mode.

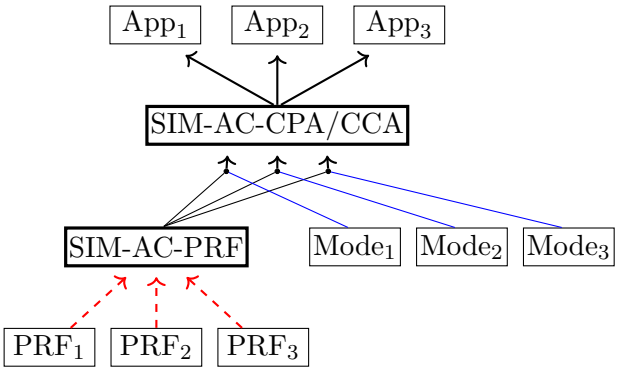

Figure 2: New state of affairs. Red dashed lines correspond to implications proved through programming in an ideal model proof. New definitions are in bold boxes. Programming proofs are only needed to show each low level PRF construction meets SIM-AC-PRF.

problem. It is an easy exercise to construct encryption schemes which are secure when all queries to it must be chosen ahead of time but are not secure even against key-recovery attacks when an adversary may adaptively choose its queries.

The primary approach used to avoid this is to use idealized models, which we can again split into two versions. The first is to use an idealized model of encryption. Examples of this include indifferentiable authenticated encryption [4] (IAE) or the ideal encryption model (IEM) of Tyagi et al [36]. Security analyses in these models might not say much when one uses real encryption schemes, even when one is willing to use more established idealized models such as the ideal cipher model (ICM) or the random oracle model (ROM). One hope would be to use approaches such as indifferentiability [32] to modularly show that symmetric schemes sufficiently "behave like" ideal encryption, but this approach is unlikely to work for most encryption schemes used in practice [4].

The final approach, which is by far the most common in searchable symmetric encryption [2, 3, 11, 14, 15, 19, 23, 24, 27–31, 37], is to fix a *particular encryption scheme* and prove security with respect to it in the ICM or ROM. Typically encryption schemes are built as modes of operations of an underlying pseudorandom function (PRF) and this function (or its constituent parts) is what is modeled as an ideal function. The downside of this is represented in Fig. 1. On the top, we have the applications one would like to prove secure, and on the bottom, we have the different modes of operation and PRFs that one might use. Using this approach means that for each application, we have to provide a separate ideal model proof for each different choice of a mode of operation and a PRF (represented by dotted red arrows in Fig. 1). If there are A applications, P PRFs, and

M modes of operation one might consider using, then this requires $A \cdot P \cdot M$ ideal model proofs in total, an unsatisfying state of affairs.

Moreover, the required ideal analysis can be tedious[1] and error-prone. This is presumably why only a few of the papers we found actually attempt to provide the full details of the ROM proof. We have identified bugs in all of the proofs that did provide the details. The lack of a full, valid proof among the fifteen papers we considered indicates the need for a more modular framework to capture this use of the random oracle. Our work provides such a framework, allowing the random oracle details to be abstracted away as a proof that only needs to be provided once. This framework provides definitions for use by other cryptographers that are simple to use, apply to practical encryption schemes, and allow showing adaptive security in well-studied models.

Examples of the "commitment problem". We proceed by discussing the example applications where we will apply our framework.

Revocable cloud storage and the compelled access setting. We start with the recently introduced compelled access setting (CAS) [36]. Here one wants encryption tools that provide privacy in the face of an authority searching digital devices, e.g., government searches of mobile phones or laptops at border crossings. To protect against compelled access searches, the BurnBox tool [36] uses what they call revocable encryption. At its core, this has the system encrypt a user's files $m_1, \ldots, m_n$ with independent keys $k_1, \ldots, k_n$. Ciphertexts are stored on (adversarially visible) cloud storage. Before a search occurs, the user instructs the application to delete the keys corresponding to files that the user wishes to hide from the authority, thereby revoking their own access to them. The other keys and file contents are disclosed to the authority.

The formal security definition introduced by Tyagi et al. captures confidentiality for revoked files even in the face of adversarial choice of which files to revoke, meaning they want security in the face of adaptive compromises. This very naturally results in the commitment problem because the simulator can be forced to provide ciphertexts for files, but only later learn the contents of these files at the time of key revelation. At which point, it is supposed to give keys which properly decrypt these ciphertexts.

To address the commitment problem they introduced the IEM. This models symmetric encryption as perfect: every encryption query is associated to a freshly chosen random string as ciphertext, and decryption is only allowed on previously returned ciphertexts. Analyses in the IEM can commit to ciphertexts (when the adversary doesn't know the key) and later open them to arbitrary messages. In their implementation, they used AES-GCM for encryption which cannot be thought of as indifferentiable from the IEM. Hence their proof can ultimately only provide heuristic evidence for the security of their implemenation.

Symmetric searchable encryption. Our second motivating setting is symmetric searchable encryption (SSE), which has similar issues as that discussed above for BurnBox, but with added complexity. SSE handles the following problem: a client wants to offload storage of a database of documents to an untrusted server while maintaining the ability to perform keyword searches on the database. The keyword searches should not reveal the contents of the documents to the server. To enable efficient solutions, we allow queries to leak some partial information about documents. Security is formalized via a simulation-based definition [17], in which a simulator given only the allowed leakage must trick an adversary into believing it is interacting with the actual SSE construction. An adaptive adversary can submit keyword searches as a function of prior returned results. Proving security here establishes that the scheme only leaks what is allowed and nothing more. While the leakage itself has been shown to be damaging in various contexts [13, 25], our focus here is on the

[1] Even more-so because SSE protocols often *also* run into the commitment problem with a PRF and need to model that using a random oracle as well.

formal analyses showing that leakage-abuse attacks are the best possible ones.

A common approach for SSE can be summarized at a high level as follows. The client generates a sequence of key pairs $(k_1, k'_1), \ldots, (k_n, k'_n)$ for keywords $w \in \{1, \ldots, n\}$ represented as integers for simplicity. The first key k_w in each pair is used to encrypt the identifiers of documents containing w. The latter key k'_w is used as a pseudorandom function (PRF) key to derive pseudorandom locations to store the encryption of the document identifiers. When the client later wants to search for documents containing w it sends the associated (k_w, k'_w) keys to the server. The server then uses k'_w to re-derive the pseudorandom locations of the ciphertexts and uses k_w to decrypt them.

To prove adaptive security, the simulator for such a protocol runs into the commitment problem because it must commit to ciphertexts of the document identifiers before knowing what the identifiers are. Perhaps less obviously, a simulator also runs into a commitment issues with the PRF. To ensure security the simulated locations of ciphertexts must be random, but then when responding to a search query the simulator is on the hook to find a key for the PRF that "explains" the simulated locations. Papers on SSE typically address these issue by modeling the PRF as a random oracle and fixing a specific construction of an encryption scheme based on a random oracle. As noted earlier, this has resulted in a need for many tedious and error-prone proofs.

Asymmetric password-authenticated key exchange and equivocable encryption. In independent and concurrent work, Jarecki et al. updated the ePrint version of [26] to introduce the notion of equivocable encryption and use it to prove security of their asymmetric password-authenticated key exchange protocol OPAQUE. The definition of equivocable encryption is essentially a weakened version of our confidentiality definition, considering only single-user security and allowing only a single encryption query; whereas we consider multi-user security and arbitrarily many adaptively chosen queries. Since their definition is implied by ours, our results will make rigorous their claim that "common encryption modes are equivocable under some idealized assumption".

A new approach. We introduce a new framework for analyzing security in adaptive compromise scenarios. Our framework has a simple, but powerful recipe: augment traditional simulation-based, property-based security definitions to empower adversaries with the ability to perform adaptive compromise of secret keys. For symmetric encryption, for example, we convert the standard simulation-based, multi-user indistinguishability under chosen plaintext attack (mu-IND-CPA) [5] to a game that includes the same adversarial powers, but adds an additional oracle for adaptively compromising individual user secret keys. Critical to our approach is (1) the use of simulators, which allows handling corruptions gracefully, and (2) incorporating handling of idealized models (e.g., the ROM or ICM). The latter is requisite for analyzing practical constructions.

We offer new definitions for multi-user CPA and CCA security of symmetric encryption, called SIM-AC-CPA (simulation-based security under adaptive corruption, chosen plaintext attack) and SIM-AC-CCA (chosen ciphertext attack). By restricting the classes of allowed simulators we can obtain stronger definitions (e.g., SIM-AC-$ which requires that ciphertexts look like random strings).

Symmetric encryption under adaptive compromise. We then begin exercising our framework by first answering the question: Are practical, in-use symmetric encryption schemes secure in the face of adaptive compromises? We give positive results here, in idealized models. Taking an encrypt-then-MAC scheme such as AES in counter mode combined with HMAC [6] as an example, we could directly show SIM-AC-CCA security while modeling AES as an ideal cipher and HMAC as a random oracle (c.f., [18]). But this would lead to a rather complex proof, and we'd have to do similarly complex proofs for other encryption schemes.

Instead, we provide simple, modular proofs by lifting the underlying assumptions made about primitives (AES and HMAC) to hold in adaptive compromise scenarios. Specifically, we introduce a new security notion for pseudorandom functions under adaptive compromise attacks (SIM-AC-

PRF). This adapts the standard multi-user PRF notion to also give adversaries the ability to adaptively compromise particular keys. Then we prove that AES and HMAC each achieve this notion in the ICM and ROM, respectively. The benefit is that these proofs encapsulate the complexity of ideal model programming proofs in the simpler context of SIM-AC-PRF (as opposed to SIM-AC-CCA).

The workflow when using our framework is represented by Fig. 2. Here PRFs are individually shown to achieve SIM-AC-PRF security in an ideal model. Then modes of operation are proven secure under the assumption that they use a SIM-AC-PRF secure PRF. Then each application is proven secure under the appropriate assumption of the encryption scheme used. This decreases the total number of proofs required to $A + P + M$, significantly fewer than the $A \cdot P \cdot M$ required previously. Moreover, the complicated ideal model programming analysis (represented by red dashed arrows) is restricted to only appearing in the the simplest of these proofs (analyzing of PRFs); it can then simply be "passed along" to the higher level proofs.

We can then show that for most CPA modes of operation (e.g., CBC mode or CTR mode), one can prove SIM-AC-CPA security assuming the underlying block cipher is SIM-AC-PRF. The core requirement is that the mode of operation enjoys a property that we call extraction security. This is a technical condition capturing the key security properties needed to prove that a mode of operation is SIM-AC-CPA assuming the underlying block cipher is SIM-AC-PRF. Moreover, we show that most existing (standard) proofs of IND-CPA security show, implicitly, the extraction security of the mode. Thus, we can easily establish adaptive compromise proofs given existing (standard) ones.

The above addresses only confidentiality. Luckily, integrity is inherited essentially for free from existing analysis. We generically show that SIM-AC-CPA security combined with the standard notion of ciphertext integrity implies SIM-AC-CCA security. Thus, one can prove encrypt-then-MAC is SIM-AC-CCA secure assuming the SIM-AC-CPA security of the encryption and the standard unforgeability security of the MAC. This is an easy adaptation of the standard proof [9] of encrypt-then-MAC.

Applying the framework to high-level protocols. Equipped with our new SIM-AC-CCA and SIM-AC-PRF security notions, we can return to our motivating task: providing positive security analyses of BurnBox and the Cash et al. SSE scheme.

We give a proof of BurnBox's CAS security assuming the SIM-AC-CPA security of the underlying symmetric encryption scheme. Our proof is significantly simpler than the original analysis, avoiding specifically the nuanced programming logic that led to the bug in the original analysis. For the Cash et al. scheme we apply our SIM-AC-PRF definition and a key-private version of our SIM-AC-CPA definition. Their adaptive security claim was accompanied only by a brief proof sketch which fails to identify an important detail that need to be considered in the ROM analysis (see Appendix F). Our proof handles this detail cleanly while being ultimately of comparable complexity to their non-adaptive security proof.

Unfortunately, these settings and constructions are inherently complicated. So even with the simplification provided by our analysis techniques there is not space to fit their analysis in the body of our paper; it has instead been relegated to the appendices of this work. We choose this organization because our main contribution is the definition abstraction which we believe will be of use for future work, rather than the particular applications we chose to exhibit its use.

Treatment of symmetric encryption. In this work, we focus on randomized encryption, over more modern nonce-based variants because this was the form of encryption used by the applications we identified. In Appendix C, we extend our definitions to nonce-based encryption. The techniques we introduce for analyzing randomized symmetric encryption schemes should extend to nonce-based encryption schemes.

Related works. A related line of work is that of selective-opening attacks [8] which studies the security of asymmetric encryption schemes against compromises in a multi-sender setting (where coins underlying encryption may be compromised) or multi-receiver setting (where secret decryption keys may be compromised). Selective-opening definitions are typically formulated to aim for standard model (or non-programmable random oracle model) achievability and hence do not suffice for the applications we consider in this work.

2 Preliminaries

A list T of length $n \in \mathbb{N}$ specifies an ordered sequence of elements $T[1]$, $T[2]$, ..., $T[n]$. The operation $T.\mathsf{add}(x)$ appends x to this list by setting $T[n+1] \leftarrow x$. This making T a list of length $n+1$. We let $|T|$ denote the length of T. The operation $x \leftarrow T.\mathsf{dq}()$ sets x equal to the last element of T and removes this element from T. In pseudocode lists are assumed to be initialized empty (i.e. have length 0). An empty list or table is denoted by $[\cdot]$. We sometimes use set notation with a list, e.g. $x \in T$ is true if $x = T[i]$ for any $1 \le i \le |T|$.

Let S and S' be two sets with $|S| \le |S'|$. Then $\mathsf{Inj}(S, S')$ is the set of all injections from S to S'. We will sometimes abuse terminology and refer to functions with co-domain $\{ S : S \subseteq \{0,1\}^* \}$ as sets. For $n \in \mathbb{N}$ we define $[n] = \{1, \ldots, n\}$.

The notation $y \leftarrow\!\!\$\ A(x_1, x_2, \cdots : \sigma)$ denotes the (randomized) execution of A with state σ. Changes that A makes to its input variable σ are maintained after A's execution. For given $x_1, x_2, \ldots$ and σ we let $[A(x_1, x_2, \cdots : \sigma)]$ denote the set of possible outputs of A given these inputs.

We define security notions using pseudocode-based games. The pseudocode "Require bool" is shorthand for "If not bool then return $\bot$". We will sometimes use infinite loops defining variable x_u for all $u \in \{0,1\}^*$. Such code is executed lazily; the code is initially skipped, then later if a variable x_u would be used, the necessary code to define it is first run. The pseudocode "$\exists x \in X$ s.t. $p(x)$" for some predicate p evaluates to the boolean value $\bigvee_{x \in X} p(x)$. If this is true, the variable x is set equal to the lexicographically first $x \in X$ for which $p(x)$ is true.

We use an asymptotic formalism. The security parameter is denoted λ. Our work is generally written in a way to allow concrete security bounds to be extracted easily. In security proofs we typically explicitly state how we will bound the advantage of an adversary by the advantages of reduction adversaries we build (and possibly other terms). Reduction adversaries and simulators are explicitly given in code (from which concrete statements about their efficiency can be obtained by observation).

Let $f : \mathbb{N} \to \mathbb{N}$. We say f is negligible if for all polynomials p there exists a $\lambda_p \in \mathbb{N}$ such that $f(\lambda) \le 1/p(\lambda)$ for all $\lambda \ge \lambda_p$. We say f is super-polynomial if $1/f$ is negligible. We say f is super-logarithmic if 2^f is super-polynomial.

Ideal primitives. We will make liberal use of ideal primitives such as random oracles or ideal ciphers. An ideal primitive P specifies algorithms $\mathsf{P.Init}$ and $\mathsf{P.Prim}$. The initialization algorithm has syntax $\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$. The stateful evaluation algorithm has syntax $y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$. We sometimes us A^{P} as shorthand for giving algorithm A oracle access to $\mathsf{P.Prim}(1^\lambda, \cdot : \sigma_{\mathsf{P}})$. Adversaries are often given access to P via an oracle PRIM.

Ideal primitives should be *stateless*. By this we mean that after σ_{P} is output by $\mathsf{P.Init}$, it is never modified by $\mathsf{P.Prim}$ (so we could have used the syntax $y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x, \sigma_{\mathsf{P}})$). However, when written this way, ideal primitives are typically inefficient, e.g., for the random oracle model σ_{P} would store a huge random table. Our security results will necessitate that P be efficiently instantiated so we have adopted the stateful syntax to allow ideal primitives to be written in their efficient "lazily sampled" form. Despite this notational convenience, we will assume that any ideal

primitive we reference is *essentially stateless*. By this, we mean that it could have been equivalently written to be stateless (if inefficient).[2]

The standard model is captured by the primitive P_{sm} for which $\mathsf{P}_{\mathsf{sm}}.\mathsf{Init}(1^\lambda)$ and $\mathsf{P}_{\mathsf{sm}}.\mathsf{Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$ always returns the empty string ε.

We define a random oracle that takes arbitrary input and produce variable length outputs. It is captured by the primitive $\mathsf{P}_{\mathsf{rom}}$ defined as follows.

$\underline{\mathsf{P}_{\mathsf{rom}}.\mathsf{Init}(1^\lambda)}$	$\underline{\mathsf{P}_{\mathsf{rom}}.\mathsf{Prim}(1^\lambda, x : T)}$
Return $[\cdot]$	$(x, l) \leftarrow x$
	If $T[x, l] = \bot$ then $T[x, l] \leftarrow\!\!\$\ \{0,1\}^l$
	Return $T[x, l]$

The ideal-cipher model is parameterized by a block-length $n : \mathbb{N} \to \mathbb{N}$ and captured by $\mathsf{P}^n_{\mathsf{icm}}$ defined as follows.[3]

$\underline{\mathsf{P}^{\mathsf{n}}_{\mathsf{icm}}.\mathsf{Init}(1^\lambda)}$	$\underline{\mathsf{P}^{\mathsf{n}}_{\mathsf{icm}}.\mathsf{Prim}(1^\lambda, x : (E, D))}$
Return $([\cdot], [\cdot])$	$(\mathrm{op}, K, y) \leftarrow x$
	If $\mathrm{op} = +$ then
	If $E[K, y] = \bot$ then
	$z \leftarrow\!\!\$\ \{0,1\}^{n(\lambda)} \setminus \{\ E[K, a]\ :\ a \in \{0,1\}^{n(\lambda)}\ \}$
	$E[K, y] \leftarrow z\ ;\ D[K, z] \leftarrow y$
	Return $E[K, y]$
	Else
	If $D[K, y] = \bot$ then
	$z \leftarrow\!\!\$\ \{0,1\}^{n(\lambda)} \setminus \{\ D[K, a]\ :\ a \in \{0,1\}^{n(\lambda)}\ \}$
	$D[K, y] \leftarrow z\ ;\ E[K, z] \leftarrow y$
	Return $D[K, y]$

It stores tables E and D which it uses to lazily sample a random permutation for each K, with $E[K, \cdot]$ representing the forward evaluation and $D[K, \cdot]$ its inverse. It parses its input as a tuple (op, K, y) where $\mathrm{op} \in \{+, -\}$ specifies the direction of evaluation and $K \in \{0,1\}^*$ and $y \in \{0,1\}^{n(\lambda)}$ specify the input.

Sometimes we construct a cryptographic primitive from multiple underlying cryptographic primitives which expect different ideal primitives. To capture this it will be useful to have a notion of combining ideal primitives. Let P' and P'' be ideal primitives. We define their cross product $\mathsf{P} = \mathsf{P}' \times \mathsf{P}''$ as follows.

$\underline{\mathsf{P}.\mathsf{Init}(1^\lambda)}$	$\underline{\mathsf{P}.\mathsf{Prim}(1^\lambda, x : \sigma_{\mathsf{P}})}$
$\sigma'_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P}'.\mathsf{Init}(1^\lambda)$	$(\sigma'_{\mathsf{P}}, \sigma''_{\mathsf{P}}) \leftarrow \sigma_{\mathsf{P}}$
$\sigma''_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P}''.\mathsf{Init}(1^\lambda)$	$(d, x) \leftarrow x$
Return $(\sigma'_{\mathsf{P}}, \sigma''_{\mathsf{P}})$	If $d = 1$ then $y \leftarrow\!\!\$\ \mathsf{P}'.\mathsf{Prim}(1^\lambda, x : \sigma'_{\mathsf{P}})$
	Else $y \leftarrow\!\!\$\ \mathsf{P}''.\mathsf{Prim}(1^\lambda, x : \sigma''_{\mathsf{P}})$
	$\sigma_{\mathsf{P}} \leftarrow (\sigma'_{\mathsf{P}}, \sigma''_{\mathsf{P}})$
	Return y

By our earlier convention $A^{\mathsf{P}' \times \mathsf{P}''}$ is shorthand for giving algorithm A oracle access to $\mathsf{P}.\mathsf{Prim}(1^\lambda, \cdot : \sigma_{\mathsf{P}})$. In A's code, $B^{\mathsf{P}'}$ denotes giving B oracle access to $\mathsf{P}.\mathsf{Prim}(1^\lambda, (1, \cdot) : \sigma_{\mathsf{P}})$ and $B^{\mathsf{P}''}$ to denote giving B oracle access to $\mathsf{P}.\mathsf{Prim}(1^\lambda, (2, \cdot) : \sigma_{\mathsf{P}})$.

[2]Without this restrictions an ideal primitive could behave in undesirable, contrived ways (e.g., on some special input outputting all prior inputs it has received).

[3]We will implicitly assume $n(\lambda)$ can be computed in time polynomial in λ. We make similar implicit assumptions for other functions that parameterize ideal primitives or constructions of cryptographic primitives.

2.1 Standard Cryptographic Definitions

We recall standard cryptographic syntax and security notions.

Symmetric encryption syntax. A symmetric encryption scheme SE specifies algorithms $\mathsf{SE.Kg}$, $\mathsf{SE.Enc}$, and $\mathsf{SE.Dec}$ as well as sets $\mathsf{SE.M}$, $\mathsf{SE.Out}$, and $\mathsf{SE.K}$ representing the message, ciphertext, and key space respectively. The key generation algorithm has syntax $K \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$. The encryption algorithm has syntax $c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, K, m)$, where $c \in \mathsf{SE.Out}(\lambda, |m|)$ is required. The deterministic decryption algorithm and has syntax $m \leftarrow \mathsf{SE.Dec}^{\mathsf{P}}(1^\lambda, K, c)$. Rejection of c is represented by returning $m = \bot$. Informally, correctness requires that encryptions of messages in $\mathsf{SE.M}(\lambda)$ decrypt properly. We assume the boolean $(m \in \mathsf{SE.M}(\lambda))$ can be efficiently computed.

Integrity of ciphertexts. Integrity of ciphertext security is defined by the game $\mathrm{G}^{\mathsf{int\text{-}ctxt}}_{\mathsf{SE},\mathcal{A}_{\mathsf{ctxt}}}$ shown in Fig. 3. In the game, the attacker interacts with one of two "worlds" (determined by the bit b) via its oracles Enc, Prim, Exp, and Dec. The attacker's goal is to determine which world it is interacting with.

$\underline{\text{Game } \mathrm{G}^{\mathsf{int\text{-}ctxt}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{ctxt}}}(\lambda)}$	$\underline{\textsc{Enc}(u, m)}$	$\underline{\textsc{Dec}(u, c)}$
For $u \in \{0,1\}^*$ do	Require $m \in \mathsf{SE.M}(\lambda)$	Require $u \notin X$
$\quad K_u \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, K_u, m)$	Require $c \notin C_u$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$	$C_u.\mathsf{add}(c)$	$m_1 \leftarrow \mathsf{SE.Dec}^{\mathsf{P}}(1^\lambda, K_u, c)$
$b \leftarrow\!\!\$\ \{0,1\}$	Return c	$m_0 \leftarrow \bot$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Enc},\textsc{Dec},\textsc{Exp},\textsc{Prim}}_{\mathsf{ctxt}}(1^\lambda)$	$\underline{\textsc{Exp}(u)}$	Return m_b
Return $(b = b')$	$X.\mathsf{add}(u)$	
$\underline{\textsc{Prim}(x)}$	Return K_u	
$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$		
Return y		

Figure 3: Game defining multi-user CTXT security of SE in the face of exposures.

The Prim oracle gives the attacker access to the ideal primitive P. The encryption oracle Enc takes as input a user u and message m, then returns the encryption of that message using the key of that user, K_u. Recall that by our convention each K_u is not sampled until needed. The exposure oracle Exp takes in u and then returns K_u to the attacker. The decryption oracle Dec is the only oracle whose behavior depends on the bit b. It takes as input a user u and ciphertext c. When $b = 1$, it will return the decryption of c using K_u while when $b = 0$ it will always return $\bot$. Thus, the goal of the attacker it to forge a ciphertext which will decrypt to a non-$\bot$ value.

To prevent trivial attacker, we disallow querying a ciphertext to $\textsc{Dec}(u, \cdot)$ if it came from $\textsc{Enc}(u, \cdot)$ or if u was already exposed. This is captured by the "Require" statements in Dec using lists C_u and X (which store the ciphertexts returned by $\textsc{Enc}(u, \cdot)$ and the users that have been exposed, respectively).

We define the advantage function $\mathsf{Adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{ctxt}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{int\text{-}ctxt}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{ctxt}}}(\lambda)] - 1$. We say SE is INT-CTXT secure with P if for all PPT $\mathcal{A}_{\mathsf{ctxt}}$, the advantage $\mathsf{Adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{ctxt}}}(\cdot)$ is negligible. INT-CTXT security is typically defined to only consider a single user and no exposures. Using a hybrid argument one can show that our definition of INT-CTXT security is implied by the more standard definition.

Function family. A family of functions F specifies algorithms $\mathsf{F.Kg}$ and $\mathsf{F.Ev}$ together with sets $\mathsf{F.Inp}$ and $\mathsf{F.Out}$. The key generation algorithm has syntax $K \leftarrow\!\!\$\ \mathsf{F.Kg}^{\mathsf{P}}(1^\lambda)$. The evaluation algorithm is deterministic and has the syntax $y \leftarrow \mathsf{F.Ev}(1^\lambda, K, x)$. It is required that for all $\lambda \in \mathbb{N}$ and $K \in [\mathsf{F.Kg}(1^\lambda)]$ that $\mathsf{F.Ev}(1^\lambda, K, x) \in \mathsf{F.Out}(\lambda)$ whenever $x \in \mathsf{F.Inp}(\lambda)$. It is assumed that random

elements of $\mathsf{F.Out}(\lambda)$ can be efficiently sampled.

Game $\mathrm{G}^{\mathsf{ow}}_{\mathsf{F},\mathsf{P},\mathcal{A}}(\lambda)$	$\textsc{Prim}(x)$
$K \leftarrow\!\!\$\ \mathsf{F.Kg}(1^\lambda)\ ;\ \sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$	$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$
$x \leftarrow\!\!\$\ \mathsf{F.Inp}(\lambda)\ ;\ y \leftarrow \mathsf{F.Ev}^{\mathsf{P}}(\lambda, K, x)$	Return y
$x' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Prim}}(1^\lambda, K, y)$	
Return $(\mathsf{F.Ev}^{\mathsf{P}}(1^\lambda, K, x') = y)$	

Figure 4: Game defining one-wayness of F.

One-wayness. The one-wayness of a family of functions F is given by the game G^{ow} shown in Fig. 4. The adversary is given a key K to F and the image y of a random point x in the domain. Its goal is to find a point with the same image. We define the advantage function $\mathsf{Adv}^{\mathsf{ow}}_{\mathsf{F},\mathsf{P},\mathcal{A}}(\lambda) = \Pr[\mathrm{G}^{\mathsf{ow}}_{\mathsf{F},\mathsf{P},\mathcal{A}}(\lambda)]$ and say F is OW secure with P if $\mathsf{Adv}^{\mathsf{ow}}_{\mathsf{F},\mathsf{P},\mathcal{A}}(\cdot)$ is negligible for all PPT $\mathcal{A}$.

Security definitions. In the body of this paper we sometimes informally reference other security notions for symmetric encryption schemes (IND-CPA, IND-CCA, IND-KP, IND-\$) and function families (PRF, UF-CMA). These definitions are recalled in Appendix A.

3 New Security Definitions for Symmetric Primitives

In this section we provide our definitions for the security of symmetric cryptographic primitives (namely randomized encryption and pseudorandom functions) against attackers able to adaptively compromise users' keys.

3.1 Randomized Symmetric Encryption

We describe our security definitions for randomized symmetric encryption. We refer to them as SIM-AC-CPA and SIM-AC-CCA security. The definition of SIM-AC-CPA (resp. SIM-AC-CCA) security is a generalization of IND-CPA (IND-CCA) security to a multi-user setting in which some users' keys may be compromised by an attacker.

Consider game $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cpa}}$ shown in Fig. 5. It is parameterized by a symmetric encryption scheme SE, simulator S, ideal primitive P, and attacker $\mathcal{A}_{\mathsf{cpa}}$. The attacker interacts with one of two "worlds" via its oracles Enc, Exp, and Prim. The attacker's goal is to determine which world it is interacting with.

In the real world ($b = 1$) the encryption oracle Enc takes (u, m) as input and returns an encryption of m using u's key K_u. Oracle Prim returns the output of the ideal primitive on input x. Oracle Exp returns u's key K_u to the attacker.

In the ideal world ($b = 0$), the return values of each of these oracles are instead chosen by a simulator S. In Prim it is given the input provided to the oracle. In Enc it is given the name of the current user u and some leakage ℓ about the message m. If u has not yet been exposed ($u \notin X$) this leakage is just the length of the message. Otherwise the leakage is the message itself. The inputs and outputs of this oracle for a user u are stored in the lists M_u and C_u so they can be leaked to the simulator when $\textsc{Exp}(u)$ is called.

We define $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda)] - 1$. We say SE is SIM-AC-CPA secure with P if for all PPT $\mathcal{A}_{\mathsf{cpa}}$ there exists a PPT S such that $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\cdot)$ is negligible. Intuitively, this definition captures that ciphertexts reveal nothing about the messages encrypted other than their length unless the encryption key is known to the attacker. In Appendix B, we show that

SIM-AC-CPA security is impossible in the standard model. The proof is a simple application of the ideas of Nielsen [33].

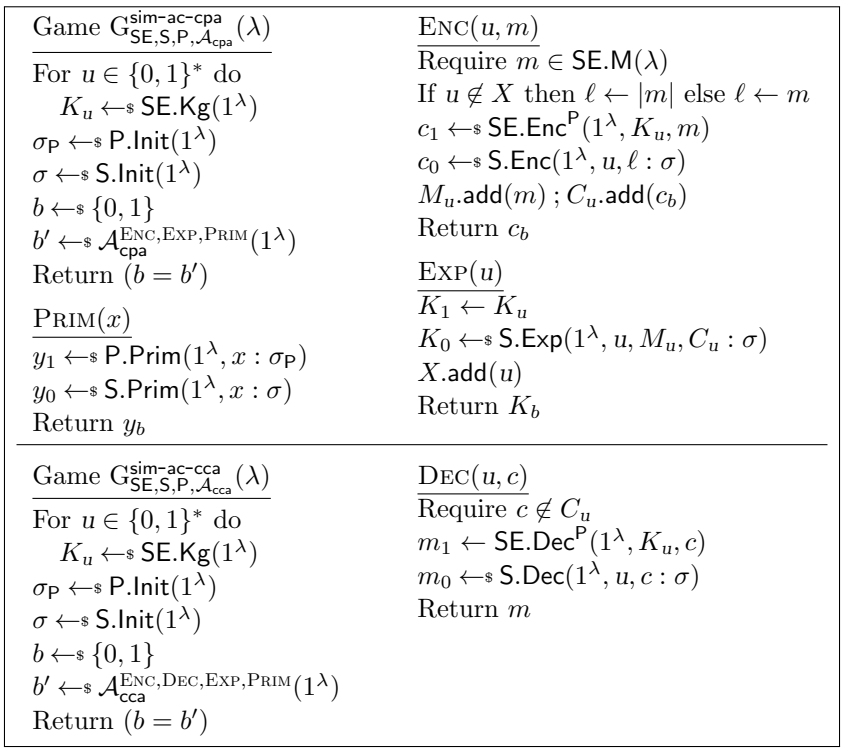

Figure 5: Games defining SIM-AC-CPA and SIM-AC-CCA security of SE.

SIM-AC-CCA security extends SIM-AC-CPA security by giving $\mathcal{A}_{\mathsf{cca}}$ access to a decryption oracle which takes as input (u, c). In the real world, it returns the decryption of c using K_u. In the ideal world, the simulator simulates this. To prevent trivial attacks, the attacker is disallowed from querying (u, c) if c was returned from an earlier query $\textsc{Enc}(u, m)$. We define $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda) = 2\Pr[G^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda)] - 1$. We say SE is SIM-AC-CCA secure with P if for all PPT $\mathcal{A}_{\mathsf{cca}}$ there exists a PPT S such that $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\cdot)$ is negligible.

Simplifications. It will be useful to keep in mind simplifications we can make to restrict the behavior of the adversary or simulator without loss of generally. They are applicable to all SIM-AC-style definitions we provide in this paper.

- If an oracle is deterministic in the real world, then we can assume that the adversary never repeats a query to this oracle or that the simulator always provides the same output to repeated queries.
- We can assume the adversary never makes a query to a user it has already exposed or that for such queries the simulator just runs the code of the real world (replacing calls to P with calls to $\mathsf{S.Prim}$).
- We can assume the adversary always queries with $u \in [u_\lambda]$ for some polynomial $u_{(\cdot)}$ or that the simulator is agnostic to the particular strings used to reference users.
- We can assume that adversaries never make queries that fail "Require" statements. (All

requirements of oracles we provide will be efficiently computable given the transcripts of queries the adversary has made.)

Proving these are slightly more subtle to establish than analogous simplifications would be in non-simulation-based games because of the order that algorithms are quantified in our security definitions. They all follow the same pattern though, so we sketch the second of these.

Suppose SE is SIM-AC-CPA secure for all adversaries that never make a call $\textsc{Enc}(u, m)$ after having made a call $\textsc{Exp}(u)$, then we claim SE is SIM-AC-CPA secure. Let $\mathcal{A}$ be an arbitrary adversary. Then we build a wrapper adversary $\mathcal{A}'$ that simply forwards all of $\mathcal{A}$'s queries except for encryption queries made for a user that has already been exposed. In these cases $\mathcal{B}$ responds with the output of $\mathsf{SE.Enc}^{\textsc{Prim}(\cdot)}(1^\lambda, K_u, m)$ (or $\perp$ if $m \notin \mathsf{SE.M}(\lambda)$), where K_u is the key last returned from $\textsc{Exp}(u)$. Let S' be a simulator for $\mathcal{A}'$. Then we construct S for $\mathcal{A}$ which responds exactly as S' would except in response to encryption queries made for a user that has already been exposed. In these cases S' responds with the output of $\mathsf{SE.Enc}^{\mathsf{S}'.\mathsf{Prim}(1^\lambda,\cdot:\sigma)}(1^\lambda, K_u, m)$, where K_u is the key it last returned for $\textsc{Exp}(u)$. It is clear that $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S}',\mathsf{P},\mathcal{A}'}(\lambda)$ because the view of $\mathcal{A}$ is identical in the corresponding games.

Stronger security notions. It is common in the study of symmetric encryption primitives to study stronger security definitions than IND-CPA security. Most schemes instead aim directly for their output to be indistinguishable from random bits (IND-\$). This implies IND-CPA security and additional nice properties such as forms of key-privacy.

We can capture such notions by placing restrictions on the behavior of the simulator. Let S be a simulator (for which we think of $\mathsf{S.Enc}$ as being undefined) which additionally defines algorithms $\mathsf{S.Enc}_1$ and $\mathsf{S.Enc}_2$ as well as set $\mathsf{S.Out}$. Then we define simulators $\mathsf{S}_\mathsf{k}[\mathsf{S}]$ and $\mathsf{S}_\$[\mathsf{S}]$ to be identical to S except for the following encryption simulation algorithms.

$\underline{\mathsf{S}_\mathsf{k}[\mathsf{S}].\mathsf{Enc}(1^\lambda, u, \ell : \sigma)}$	$\underline{\mathsf{S}_\$[\mathsf{S}].\mathsf{Enc}(1^\lambda, u, \ell : \sigma)}$
If $\ell \in \mathbb{N}$ then $c \leftarrow\!\!\$\ \mathsf{S.Enc}_1(1^\lambda, \ell : \sigma)$	If $\ell \in \mathbb{N}$ then $c \leftarrow\!\!\$\ \mathsf{S.Out}(\lambda, \ell)$
Else $c \leftarrow\!\!\$\ \mathsf{S.Enc}_2(1^\lambda, u, \ell : \sigma)$	Else $c \leftarrow\!\!\$\ \mathsf{S.Enc}_2(1^\lambda, u, \ell : \sigma)$
Return c	Return c

Checking $\ell \in \mathbb{N}$ acts as a convenient way of verifying if the user being queried has been exposed yet. Because $\mathsf{S.Enc}_1(1^\lambda, \ell : \sigma)$ is not given u in S_k, the output of S_k is distributed identically for any unexposed users. The class of key-anonymous simulators $\mathcal{S}_k$ is the set of all $\mathsf{S}_\mathsf{k}[\mathsf{S}]$ for some S. Similarly, $\mathsf{S}_\$$ always outputs a random bitstring as the ciphertext for any unexposed user. The class of random-ciphertext simulators $\mathcal{S}_\$$ is the set of all $\mathsf{S}_\$[\mathsf{S}]$ for some S. Note that $\mathcal{S}_\$ \subset \mathcal{S}_k$.

We say SE is SIM-AC-KP secure with P if for all PPT $\mathcal{A}_{\mathsf{cpa}}$ there exists a PPT $\mathsf{S} \in \mathcal{S}_k$ such that $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\cdot)$ is negligible. We say that SE is SIM-AC-\$ secure with P if for all PPT $\mathcal{A}_{\mathsf{cpa}}$ there exists a PPT $\mathsf{S} \in \mathcal{S}_\$$ such that $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\cdot)$ is negligible. It is straightforward to see that SIM-AC-\$ security implies SIM-AC-KP security which in turn implies SIM-AC-CPA security. Standard counter-examples will show that these implications do not hold in the other direction.

It is sometimes useful to define security in an all-in-one style, introduced by Rogaway and Shrimpton [35], which simultaneously requires IND-\$ security and INT-CTXT security. In our framework we can define $\mathcal{S}_\perp$ as the class of IND-CCA simulators which always return $\perp$ for decryption queries to unexposed users. Then we say SE is SIM-AC-AE secure with P if for all PPT $\mathcal{A}_{\mathsf{cca}}$ there exists a PPT $\mathsf{S} \in \mathcal{S}_\$ \cap \mathcal{S}_\perp$ such that $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\cdot)$ is negligible.

Nonce-based Encryption. In Appendix C, we provide the analogous SIM-AC-style definitions for nonce-based encryption.

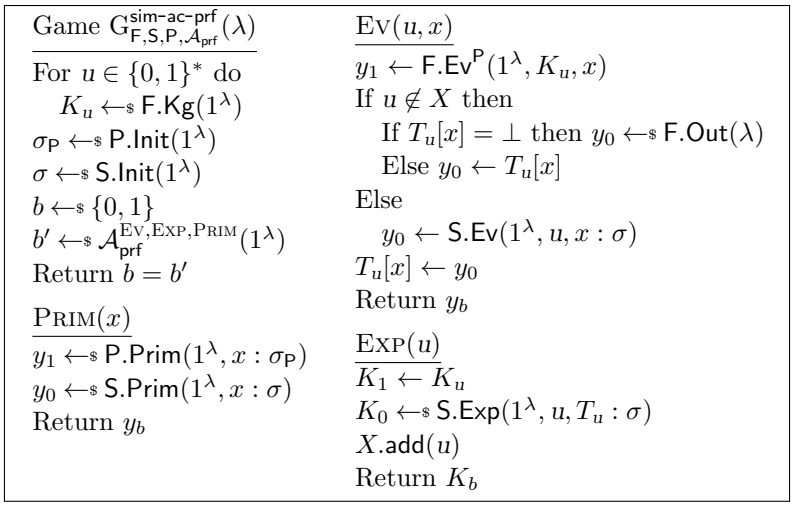

Figure 6: Game defining multi-user PRF security of F in the face of exposures.

3.2 Pseudorandom Functions

Typically a symmetric encryption scheme will use a PRF as one of their basic building blocks. For modularity, it will be useful to provide a simulation-based security definition for PRFs in the face of active compromises. In Section 6, we show our PRF definition can be applied to construct a SIM-AC secure symmetric encryption scheme. Additionally, in Appendix D, we show that our definition is of independent use by using it to prove the adaptive security of a searchable symmetric encryption scheme introduced by Cash et al. [14].

The game $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}$ is shown in Fig. 6. In the real world, Ev gives adversary $\mathcal{A}_{\mathsf{prf}}$ the real output of F. In the ideal world, Ev's output is chosen at random (and stored in the table T_u), unless u has already been exposed in which case simulator S chooses the output. The table T_u is given to S when an exposure of u happens so it can output a key consistent with prior Ev queries; we assume it is easy to iterate over all $(x, T_u[x])$ pairs for which $T_u[x]$ is not $\perp$. We define $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda)] - 1$. We say F is SIM-AC-PRF secure with P if for all PPT $\mathcal{A}_{\mathsf{prf}}$ there exists a PPT S such that $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\cdot)$ is negligible.

4 Applications

The value of our definitions stems from their usability in proving the security of protocols constructed from symmetric encryption and pseudorandom functions. In this section, we discuss the application our definitions to simplify and modularize existing security results of Cash et al. [14] and Tyagi et al. [36], and how they imply the notion of equivocable encryption introduced by Jarecki et al. [26].

4.1 Asymmetric Password-Authenticated Key Exchange: OPAQUE

Password-authenticated key exchange (PAKE) protocols allow a client and a server with a shared password to establish a shared key resistant to offline guessing attacks. *Asymmetric* PAKE (aPAKE) further considers security in the case of server compromise, meaning that the server must store some secure representation of the password, rather than the password itself.

OPAQUE [26] is an aPAKE protocol currently being considered for standardization by the IETF. At a high level, OPAQUE is constructed from an oblivious pseudorandom function (OPRF) and an authenticated key exchange protocol (AKE). User key material for the AKE protocol is stored encrypted under an password-derived key from an OPRF. Key exchange proceeds in two steps: (1) the user rederives the encryption key by running the OPRF protocol with the server on their password, then (2) retrieves and decrypts the AKE keys from the server-held ciphertext and proceeds with the AKE protocol. The "commitment problem" arises when an adversary comprises the server state and then later compromises a user password.

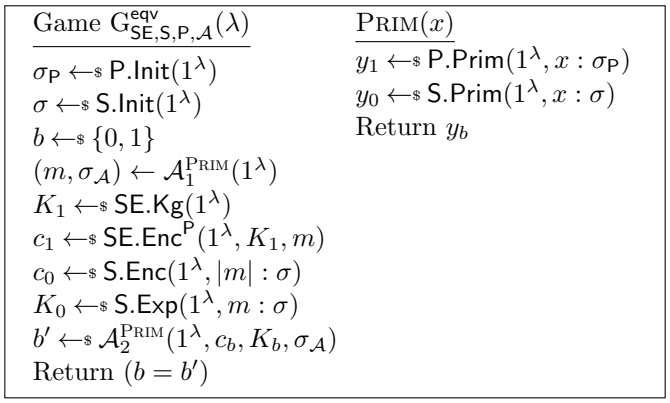

Figure 7: Game defining EQV security of SE.

Comparison to equivocable encryption. To prove security of their scheme, Jarecki et al. independently propose a weaker version of SIM-AC-CPA that they call equivocable encryption (EQV). Consider game $\mathrm{G}^{\mathsf{eqv}}$ defined in Fig. 7. An encryption scheme SE is *equivocable* if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a simulator S, such that the advantage function $\mathsf{Adv}^{\mathsf{eqv}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{eqv}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}}(\lambda)] - 1$ is negligible. The [26] definition does not specify how to incorporate the ideal model, so we make a reasonable assumption.

Note that EQV is a weaker version of SIM-AC-CPA in that it allows for only one user and only one encryption query. Showing SIM-AC-CPA implies EQV can be done with a simple wrapper reduction in which the output of $\mathcal{A}_1$ from EQV is forwarded to the encryption oracle of SIM-AC-CPA. Since EQV allows for only one encryption query, we can further show that EQV does not imply SIM-AC-CPA. Consider a scheme that uses a key $K = (K_1, K_2)$ and constructs ciphertexts as $(K_1, \mathsf{Enc}_{K_2}(m))$ unless $m = K_1$, in which case it is formed as $(K_2, \mathsf{Enc}_{K_2}(m))$. Such a scheme could be secure with respect to EQV but will not be secure in a game that allows multiple encryption queries. Interestingly, showing that our multi-user SIM-AC-CPA notion is implied by its single-user version through a hybrid argument is not straightforward due to managing inconsistencies in simulator state between hybrid steps. We have not been able to prove this result and leave it open for future work. Thus, even if EQV was extended to allow multiple encryption queries, it still may not be widely applicable to situations that require multiple users.

Ultimately, our work fills in the claim of Jarecki, et al. that "common encryption modes are equivocable under some idealized assumption".

4.2 Searchable Symmetric Encryption

In Appendix D, we show that our symmetric encryption and PRF security definitions are useful for proving the security of searchable searchable symmetric encryption (SSE) schemes. An SSE

scheme allows a client with a database of documents to store them in encrypted form on a server while still being able to perform keyword searches on these documents.

As a concrete example, we consider Cash et al. [14] which proved non-adaptive security of an SSE scheme when using a PRF and an IND-$ secure encryption scheme and claimed adaptive security when the PRF is replaced with a random oracle and the encryption scheme is replaced with a specific random-oracle-based scheme. We will prove their adaptive result, this time assuming the family of functions is SIM-AC-PRF secure and the encryption scheme is SIM-AC-KP secure. This makes the result more modular because one is no longer restricted to use their specific choices of a PRF and encryption scheme constructed from a random oracle. As a concrete benefit of this, their choice of encryption scheme does not provide INT-CTXT security. To replace the scheme with one that does would require a separate proof while our proof allows the user to choice their favorite INT-CTXT secure scheme without requiring any additional proofs (assuming that scheme is SIM-AC-CPA secure).

Our proof is roughly as complex as their non-adaptive proof; it consists of three similar reductions to the security of the underlying primitives. Without our definitions, a full adaptive proof would have been a technically detailed (though "standard" and not conceptually difficult) proof because it would have to deal with programming the random oracle. Perhaps because of this, the authors of [14] only provided a sketch of the result, arguing that it follows from the same ideas as their non-adaptive proof plus programming the random oracle to maintain consistency. They claim, "[t]he only defects in the simulation occur when an adversary manages to query the random oracle with a key before it is revealed". This is technically insufficient; a defect also occurs if the same key is sampled multiple times by the simulator (analogously to parts of our proofs for Theorem 5.3 and Theorem 5.4). In our SSE proof, we need not address these details because programming the ideal primitive is handled by the assumed simulation security of the underlying primitives.

A large number of other works on SSE have used analogous techniques of constructing a PRF and/or encryption scheme from a random oracle to achieve adaptive security [2, 3, 11, 14, 15, 19, 23, 24, 27–31, 37]. As we discuss in Appendix F, these papers all similarly elided the details of the random oracle programming proof and/or made mistakes in writing these details. The mistakes are individually small and not difficult to fix, but their prevalence indicates the value our definitions can provide to modularize and simplify the proofs in these works. We chose to analyze the Cash et al. scheme to highlight the application of our definitions because it was the simplest construction requiring both SIM-AC-PRF and SIM-AC-KP secure and because their thorough non-adaptive proof served as a useful starting point from which to build our proof.

4.3 Self-Revocable Encrypted Cloud Storage: BurnBox

In Appendix E.1, we consider the BurnBox construction of a self-revocable cloud storage scheme proposed by Tyagi et al. [36]. Its goal is to help provide privacy in the face of an authority searching digital devices, e.g., searches of mobile phones or laptops at border crossings. In their proposed scheme a user stores encrypted version of their files on cloud storage. At any point in time they are able to temporarily revoke their own access to these files. Thereby an authority searching their device is unable to learn the content of these files despite their possession of all the secrets stored on the user's device.

Proving security of their scheme in their security model necessitates solving the "commitment problem." A simulator is forced to simulate the attacker's view by providing ciphertexts for files that it does not know the contents of, then later produce a plausible looking key which decrypts the files properly when told the contents. To resolve this issue in their security they modeled the symmetric encryption scheme in the ideal encryption model (which they introduced for this

purpose). We are able to recover their result assuming the SIM-AC-CPA security of the encryption scheme. This provides rigorous justification for the use of practically-used encryption schemes which cannot necessarily be thought of as well modeled by the ideal encryption model (e.g. AES-GCM which they used in their prototype implementation). Moreover, the proof we obtain is simpler than the original proof of Tyagi et al. because we do not have to reason about the programming of the ideal encryption model. The original proof has a bug in this programming which we discuss in Appendix F.

5 Symmetric Encryption Security Results

In this section, we show that important existing results about the security of symmetric encryption schemes "carry over" to our new definitions. These results (together with our results in the next section) form the foundation of our claim that encryption schemes used in practice can be considered to achieve SIM-AC-AE security when their underlying components are properly idealized. First, we show that SIM-AC-CPA and INT-CTXT security imply SIM-AC-CCA security. Then we show that the classic Encrypt-then-MAC scheme achieves SIM-AC-CCA security. Each of these results are, conceptually, a straightforward extension of their standard proof. Finally, we show that random oracles and ideal ciphers are SIM-AC-PRF secure and ideal encryption [36] is SIM-AC-AE secure.

CPA and CTXT imply CCA. The following theorem captures that SIM-AC-CPA and INT-CTXT security imply SIM-AC-CCA security. Bellare and Namprempre [9] showed the analogous result for IND-CPA and IND-CCA security.

Theorem 5.1 *If* SE *is SIM-AC-CPA and INT-CTXT secure with* P*, then* SE *is SIM-AC-CCA secure with* P*.*

Proof (Sketch): Here we sketch the main ideas of the proof. The full details are provided in Appendix G.1.

The SIM-AC-CCA simulator we provide is parameterized by a SIM-AC-CPA simulator $\mathsf{S}_{\mathsf{cpa}}$. As state it stores σ of $\mathsf{S}_{\mathsf{cpa}}$ and keeps each K_u that is has returned to exposure queries. For Prim, Enc, and Exp queries it simply runs $\mathsf{S}_{\mathsf{cpa}}$. For Dec queries it does one of two things. If u has already been exposed it uses the key it previously returned to run the actual decryption algorithm (with oracle access to $\mathsf{S}_{\mathsf{cpa}}$'s emulation of P) and returns the result. Otherwise it assumes the adversary has failed at producing a forgery and simply returns $\perp$. (Note this means we have SIM-AC-AE security if SE is SIM-AC-\$ secure.)

The SIM-AC-CPA security of SE ensures that the adversary cannot differentiate between the real and ideal world queries to Prim, Enc, and Exp. The INT-CTXT security of SE does the same for the Dec queries. ∎

Encrypt-then-MAC. Let SE be an encryption scheme. Let F be a family of functions for which $\mathsf{F.Inp}(\lambda) = \{0,1\}^*$. Then the Encrypt-then-MAC encryption scheme using SE and F is denoted $\mathsf{EtM}[\mathsf{SE},\mathsf{F}]$. Its message space is defined as $\mathsf{EtM}[\mathsf{SE},\mathsf{F}].\mathsf{M}(\lambda) = \mathsf{SE.M}(\lambda)$. If SE expects access to ideal primitive P_1 and F expects access to ideal primitive P_2 then $\mathsf{EtM}[\mathsf{SE},\mathsf{F}]$ expects access to $\mathsf{P}_1 \times \mathsf{P}_2$. The key-generation algorithm $\mathsf{EtM}[\mathsf{SE},\mathsf{F}].\mathsf{Kg}$ returns $K = (K_{\mathsf{SE}}, K_{\mathsf{F}})$ where K_{SE} was sampled with $\mathsf{SE.Kg}(1^\lambda)$ and K_{F} was sampled with $\mathsf{F.Kg}(1^\lambda)$. Algorithms $\mathsf{EtM}[\mathsf{SE},\mathsf{F}].\mathsf{Enc}$, and $\mathsf{EtM}[\mathsf{SE},\mathsf{F}].\mathsf{Dec}$ are defined as follows.

$$\begin{array}{l|l}
\underline{\mathsf{EtM}[\mathsf{SE},\mathsf{F}].\mathsf{Enc}^{\mathsf{P}_1\times\mathsf{P}_2}(1^\lambda, K, m)} & \underline{\mathsf{EtM}[\mathsf{SE},\mathsf{F}].\mathsf{Dec}^{\mathsf{P}_1\times\mathsf{P}_2}(1^\lambda, K, (c_{\mathsf{SE}}, \tau))} \\
(K_{\mathsf{SE}}, K_{\mathsf{F}}) \leftarrow K & (K_{\mathsf{SE}}, K_{\mathsf{F}}) \leftarrow K \\
c_{\mathsf{SE}} \leftarrow\!\!\$\ \mathsf{SE}.\mathsf{Enc}^{\mathsf{P}_1}(1^\lambda, K_{\mathsf{SE}}, m) & \text{If } \tau \neq \mathsf{F}.\mathsf{Ev}^{\mathsf{P}_2}(1^\lambda, K_{\mathsf{F}}, c_{\mathsf{SE}}) \text{ then return } \perp \\
\tau \leftarrow \mathsf{F}.\mathsf{Ev}^{\mathsf{P}_2}(1^\lambda, K_{\mathsf{F}}, c_{\mathsf{SE}}) & m \leftarrow \mathsf{SE}.\mathsf{Dec}^{\mathsf{P}_1}(1^\lambda, K_{\mathsf{SE}}, c_{\mathsf{SE}}) \\
\text{Return } (c_{\mathsf{SE}}, \tau) & \text{Return } m
\end{array}$$

The following theorem establishes that the generic composition result of Bellare and Namprempre [9] holds with our simulation-based definitions of security. We sketch its straightforward proof in Appendix G.2.

Theorem 5.2 *Let* SE *be an encryption scheme. Let* F *be a family of functions for which* $\mathsf{F.Inp}(\lambda) = \{0,1\}^*$. *If* SE *is SIM-AC-CPA secure with* P_1 *and* F *is UF-CMA secure with* P_2, *then* $\mathsf{EtM}[\mathsf{SE},\mathsf{F}]$ *is SIM-AC-CCA secure with* $\mathsf{P}_1 \times \mathsf{P}_2$.

Random oracles are good PRFs. We show that a SIM-AC-PRF secure family of functions can be constructed simply in the random oracle model. Consider R defined as follows. It is parameterized by a key-length function $\mathsf{R.kl} : \mathbb{N} \to \mathbb{N}$ and output length function $\mathsf{R.ol} : \mathbb{N} \to \mathbb{N}$. It has input set $\mathsf{R.Inp}(\lambda) = \{0,1\}^*$ and output set $\mathsf{R.Out}(\lambda) = \{0,1\}^{\mathsf{R.ol}(\lambda)}$.

$$\begin{array}{l|l}
\underline{\mathsf{R.Kg}(1^\lambda)} & \underline{\mathsf{R.Ev}^{\mathsf{P}}(1^\lambda, K, x)} \\
K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{R.kl}(\lambda)} & y \leftarrow \mathsf{P}((K \,\|\, x, \mathsf{R.ol}(\lambda))) \\
\text{Return } K & \text{Return } y
\end{array}$$

Theorem 5.3 R *is SIM-AC-PRF secure with* $\mathsf{P}_{\mathsf{rom}}$ *if* $\mathsf{R.kl}$ *is super-logarithmic.*

Concretely, in our proof we provide a simulator $\mathsf{S}_{\mathsf{prf}}$ for which we show that,

$$\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{R},\mathsf{S}_{\mathsf{prf}},\mathsf{P}_{\mathsf{rom}},\mathcal{A}_{\mathsf{prf}}}(\lambda) \leq \frac{u_\lambda^2 + p_\lambda u_\lambda}{2^{\mathsf{R.kl}(\lambda)}}$$

where u_λ is an upper bound on the number of users that $\mathcal{A}_{\mathsf{prf}}$ queries to and p_λ is an upper bound on the number of PRIM queries that $\mathcal{A}_{\mathsf{prf}}$ makes.

This theorem captures the random oracle programming implicit in the adaptive security claims of the numerous SSE papers we have identified that used a random oracle like a PRF to achieve adaptive security [2,3,11,14,15,19,23,24,27–31,37]. Of these works, most chose to elide the details of establishing that the adversary cannot detect the random oracle programming, likely considering them simple and/or standard. Despite this, we have identified bugs in all of the proofs that did provide more details. We discuss these bugs in more detail in Appendix F.

To be clear, we do not claim that any of the SSE schemes studied in these works are insecure. The prevalence of this issue speaks to the difficulty of properly accounting for the details in an ideal model programming proof. Our SIM-AC-PRF notion provides a convenient intermediate definition via which these higher-level protocols could have been proved secure without having to deal with the tedious details of a random oracle programming proof.

Proof (Sketch): Here we sketch the main ideas of the proof. The full details are provided in Appendix H.1. The SIM-AC-PRF simulator works are follows. For PRIM queries it just emulates $\mathsf{P}_{\mathsf{rom}}$ using a table T. For EV queries, it just runs $\mathsf{R.Ev}$ honestly with the key it previously returned for the given user. For EXP queries (on an unexposed user) it picks a random key for this user and sets T to be consistent with values in the table T_u it is given. This simulation is only detectable by an attacker that makes a query to the random oracle with some key that is later chosen by the

simulator in response to an exposure or if the simulator happened to chose the same key for two different users.[4] These events happen with negligible probability.▮

Ideal ciphers are good PRFs. One of the most commonly used PRFs is AES so it would be useful to think of it as being SIM-AC-PRF secure; however, due to its invertible nature we cannot realistically model it as a random oracle and refer to the above theorem. Instead, AES is often modeled as an ideal cipher. Let $\mathsf{B.kl} : \mathbb{N} \to \mathbb{N}$ be given and consider B defined as follows. It has input set $\mathsf{B.Inp}(\lambda) = \{0,1\}^{n(\lambda)}$ and output set $\mathsf{B.Out}(\lambda) = \{0,1\}^{n(\lambda)}$.

$\underline{\mathsf{B.Kg}(1^\lambda)}$	$\underline{\mathsf{B.Ev}^{\mathsf{P}}(1^\lambda, K, x)}$
$K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{B.kl}(\lambda)}$	$y \leftarrow \mathsf{P}((+, K, x))$
Return K	Return y

The following establishes that an ideal cipher is SIM-AC-PRF secure.

Theorem 5.4 B *is SIM-AC-PRF secure with* $\mathsf{P}^n_{\mathsf{icm}}$ *if* $\mathsf{B.kl}$*,* n *are super-logarithmic.*

Concretely, in our proof we provide a simulator $\mathsf{S}_{\mathsf{prf}}$ for which we show that,

$$\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{B},\mathsf{S}_{\mathsf{prf}},\mathsf{P}^n_{\mathsf{icm}},\mathcal{A}_{\mathsf{prf}}}(\lambda) \le \frac{u_\lambda^2 + p_\lambda u_\lambda}{2^{\mathsf{B.kl}(\lambda)}} + \frac{q_\lambda^2}{2^{n(\lambda)+1}}$$

where u_λ is an upper bound on the number of users that $\mathcal{A}_{\mathsf{prf}}$ queries to, p_λ is an upper bound on the number of Prim queries that $\mathcal{A}_{\mathsf{prf}}$ makes, and q_λ is an upper bound on the number of Ev queries that $\mathcal{A}_{\mathsf{prf}}$ makes.

The proof of this theorem follows the same general pattern as the proof that a random oracle is SIM-AC-PRF secure (Theorem 5.3). It only needs to extend the ideas of this prior result slightly to apply a birthday bound so that we can treat the values of $\mathsf{P}^n_{\mathsf{icm}}$ as being sampled with replacement. It works best to process this step last so we do not have to consider the order in which queries are made. The proof is given in Appendix H.2.

Ideal encryption model. In Appendix I, we recall the ideal encryption model used in the analysis of Tyagi et al. [36] and show that it gives a SIM-AC-AE secure encryption scheme. While doing so, we identify and show how to fix a bug in their proof which used this model.

6 Security of Modes of Operation

In the previous section, we showed that existing analysis of the integrity of a symmetric encryption scheme carries over to our simulation setting to lift SIM-AC-CPA security to SIM-AC-CCA security. It would be convenient to be able to similarly prove that existing IND-CPA security of an encryption scheme suffices to imply SIM-AC-CPA security. Unfortunately, we cannot possibly hope for this to be the case. We know that IND-CPA security can be achieved in the standard model (assuming one-way functions exist), but SIM-AC-CPA security necessarily requires the use of ideal models.

For any typical encryption scheme we could figure out the appropriate way to idealize its underlying components and then write a programming proof to establish security. This would likely be detail intensive and prone to mistakes. We can improve on this by noting that typical symmetric encryption schemes are built as modes of operation using an underlying PRF. We can aim to prove security more modularly by assuming the SIM-AC-PRF security of the underlying

[4]The latter of these points is the subtle issue that does not have appear to have been identified in *any* of the SSE papers that were (implicitly) using a random oracle as a SIM-AC-PRF.

family of functions. This alleviates the detail-intensiveness of the proof because the ideal model programming has already been handled in the assumption of SIM-AC-PRF security; it can simply be "passed" along to the new analysis.

In this section, we will show that we can do *even better* than that. We will restrict attention to modes of operation which are IND-$ secure when built from a PRF and satisfy a special extractability property we define in Section 6.1 (which standard examples of models of operation do). Then, in Section 6.2, we establish a generic proof framework to elevate an existing IND-$ security proof to a SIM-AC-$ security proof, by showing that existing proofs of IND-$ security security tend to (implicitly) prove that the scheme satisfies our extractability property. Finally, in Section 6.3 we discuss how the techniques of this section can be extended to other constructions not captured by our formalism, but also note the existence of a (contrived) mode of operation which is IND-$ secure with any secure PRF, but is never SIM-AC-$ secure.

6.1 Modes of Operation and Extractability

We first need to have a formalism capturing what a mode of operation is. Our formalism does not capture all possible modes of operation, but does seem to capture most constructions that are of practical interest and would not be hard to modify to capture other constructions.

A mode of operation SE specifies efficient algorithms $\mathsf{SE.Kg}$, $\mathsf{SE.Enc}$, and $\mathsf{SE.Dec}$ as well as sets $\mathsf{SE.M}$, $\mathsf{SE.Out}$, $\mathsf{SE.FInp}$, and $\mathsf{SE.FOut}$. For any family of functions F with $\mathsf{F.Inp} = \mathsf{SE.FInp}$ and $\mathsf{F.Out} = \mathsf{SE.FOut}$, it defines a symmetric encryption scheme $\mathsf{SE}[\mathsf{F}]$ as follows.

$$\begin{array}{l|l|l}
\underline{\mathsf{SE}[\mathsf{F}].\mathsf{Kg}(1^\lambda)} & \underline{\mathsf{SE}[\mathsf{F}].\mathsf{Enc}^{\mathsf{P}}(1^\lambda, K, m)} & \underline{\mathsf{SE}[\mathsf{F}].\mathsf{Dec}^{\mathsf{P}}(1^\lambda, K, c)} \\
K_{\mathsf{F}} \leftarrow\!\!\$\ \mathsf{F.Kg}(1^\lambda) & (K_{\mathsf{SE}}, K_{\mathsf{F}}) \leftarrow K & (K_{\mathsf{SE}}, K_{\mathsf{F}}) \leftarrow K \\
K_{\mathsf{SE}} \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda) & c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{F}^{\mathsf{P}}_{K_{\mathsf{F}}}}(1^\lambda, K_{\mathsf{SE}}, m) & m \leftarrow \mathsf{SE.Dec}^{\mathsf{F}^{\mathsf{P}}_{K_{\mathsf{F}}}}(1^\lambda, K_{\mathsf{SE}}, c) \\
\text{Return } (K_{\mathsf{SE}}, K_{\mathsf{F}}) & \text{Return } c & \text{Return } m
\end{array}$$

The superscript $\mathsf{F}^{\mathsf{P}}_{K_{\mathsf{F}}}$ is shorthand for oracle access to $\mathsf{F.Ev}^{\mathsf{P}}(1^\lambda, K_{\mathsf{F}}, \cdot)$. It is required that $\mathsf{SE}[\mathsf{F}].\mathsf{M} = \mathsf{SE.M}$. Moreover, for a given $\lambda \in \mathbb{N}$ the encryption of a message $m \in \mathsf{SE.M}(\lambda)$ must always be in $\mathsf{SE.Out}(\lambda, |m|)$.

Suppose we want to prove that SE is SIM-AC-$ whenever F is SIM-AC-PRF. The natural way to do so is to build our simulator S from the encryption scheme from the given simulator $\mathsf{S_F}$ for F. In PRIM we can simply have $\mathsf{S.Prim}$ run $\mathsf{S_F.Prim}$. In ENC the ciphertext is chosen at random if the user has not been exposed, otherwise we can simply run $\mathsf{SE.Enc}$ but use $\mathsf{S_F.Ev}$ in place of $\mathsf{F}_{K_{\mathsf{F}}}$. This just leaves EXP, here we are given a list of ciphertexts for the user and need to output a key to "explain" them. A natural approach is to randomly pick our own K_{SE} and use $\mathsf{S_F.Exp}$ to chose K_{F}. Doing so requires giving $\mathsf{S_F}$ a list of input and outputs to the family of function. Intuitively, it seems we want to be able to "extract" a list of input-outputs pairs for F that explain our ciphertexts.

Extractability. A mode of operation is extractable if it additionally specifies an efficient extraction algorithm $\mathsf{SE.Ext}$ satisfying a correctness and uniformity property we now define. The extraction algorithm $\mathsf{SE.Ext}$ has syntax $(\vec{y}, r) \leftarrow\!\!\$\ \mathsf{SE.Ext}(1^\lambda, K_{\mathsf{SE}}, c, m)$. The goal of this algorithm is to "extract" a sequence of responses $\vec{y}$ by F and a string of randomness r that explains how message m could be encrypted to ciphertext c when using key K_{SE}. We formally define correctness by the following game. It is assumed that $\mathsf{SE.Ext}$ provides outputs of the appropriate lengths to make this code well-defined. Extraction correctness of SE requires that $\Pr[\mathrm{G}^{\mathsf{corr}}_{\mathsf{SE},m}(1^\lambda)] = 1$ for all $\lambda \in \mathbb{N}$ and $m \in \mathsf{SE.M}(\lambda)$.

Game $\mathrm{G}^{\mathsf{corr}}_{\mathsf{SE},m}(1^\lambda)$	Distribution 1
$K_{\mathsf{SE}} \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	$c \leftarrow\!\!\$\ \mathsf{SE.Out}(\lambda, \lvert m\rvert)$
$c \leftarrow\!\!\$\ \mathsf{SE.Out}(\lambda, \lvert m\rvert)$	$(\vec{y}, r) \leftarrow\!\!\$\ \mathsf{SE.Ext}(1^\lambda, K_{\mathsf{SE}}, c, m)$
$(\vec{y}, r) \leftarrow\!\!\$\ \mathsf{SE.Ext}(1^\lambda, K_{\mathsf{SE}}, c, m)$	Return $(\vec{y}, r)$
$i \leftarrow 0$	Distribution 2
$c' \leftarrow \mathsf{SE.Enc}^{\mathrm{RF}}(1^\lambda, K_{\mathsf{SE}}, m; r)$	For $i = 1, \ldots, q(\lambda, \lvert m\rvert)$ do
Return $c = c'$	$\quad \vec{y}[i] \leftarrow\!\!\$\ \mathsf{SE.Out}(\lambda)$
$\mathrm{RF}(x)$	$r \leftarrow\!\!\$\ \{0,1\}^{l(\lambda, \lvert m\rvert)}$
$i \leftarrow i + 1$	Return $(\vec{y}, r)$
Return $\vec{y}[i]$	

We will also require a uniformity property of $\mathsf{SE.Ext}$. Specifically we require that its output be uniformly random whenever c is. Formally, there must exist $q, l : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that the two distributions on the right above are equivalent for all $\lambda \in \mathbb{N}$, $m \in \mathsf{SE.M}(\lambda)$, and $K_{\mathsf{SE}} \in [\mathsf{SE.Kg}(1^\lambda)]$.[5]

Extraction security. A core step in our proof will require an additional property of SE which we will now define. Roughly, the desired property is that if $\mathsf{SE.Ext}$ is repeatedly used to explain randomly chosen ciphertexts an adversary cannot notice if it causes inconsistent values to be returned to $\mathsf{SE.Enc}$.

Game $\mathrm{G}^{\mathsf{ind\text{-}ac\text{-}ext}}_{\mathsf{SE},\mathcal{A}}(\lambda)$	$\mathrm{ENC}(u, m)$	$\mathrm{RF}(u, x)$ *//private*
For $u \in \{0,1\}^*$ do	Require $m \in \mathsf{SE.M}(\lambda)$	$i \leftarrow i + 1$
$\quad K_{\mathsf{SE},u} \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	Require $u \notin X$	If $T_u[x] \neq \perp$ then
$b \leftarrow\!\!\$\ \{0,1\}$	$c \leftarrow\!\!\$\ \mathsf{SE.Out}(\lambda, \lvert m\rvert)$	$\quad$ If $b = 1$ then
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{ENC},\mathrm{EXP}}(1^\lambda)$	$(\vec{y}, r) \leftarrow\!\!\$\ \mathsf{SE.Ext}(1^\lambda, K_{\mathsf{SE},u}, c, m)$	$\quad\quad \vec{y}[i] \leftarrow T_u[x]$
Return $(b = b')$	$i \leftarrow 0$	$T_u[x] \leftarrow \vec{y}[i]$
$\mathrm{EXP}(u)$	$c \leftarrow \mathsf{SE.Enc}^{\mathrm{RF}(u,\cdot)}(1^\lambda, K_{\mathsf{SE},u}, m; r)$	Return $T_u[x]$
$X.\mathsf{add}(u)$	Return c	
Return $(K_{\mathsf{SE},u}, T_u)$		

Figure 8: Game defining IND-AC-EXT security of SE. Note that the adversary is not given oracle access to the "private" oracle RF.

Formally, consider the game $\mathrm{G}^{\mathsf{ind\text{-}ac\text{-}ext}}$ shown in Fig. 8. In it, a key is chosen for each user and then the adversary is given access to an encryption oracle. In this oracle a random ciphertext is sampled. Then $\mathsf{SE.Ext}$ is run to provide vector $\vec{y}$ and coins r which explain this ciphertext with respect to the queried message. Finally, $\mathsf{SE.Enc}$ is run with coins r and access to an oracle RF whose behavior depends on the chosen $\vec{y}$. The ciphertext it outputs is returned to the adversary.

When $b = 0$, this oracle simply returns the entries of $\vec{y}$, one at a time. The value returned for an input x is stored as $T_u[x]$. The behavior when $b = 1$ is similar except that if an input x to RF is ever repeated for a user u, then the value stored in $T_u[x]$ is used instead of the corresponding entry of $\vec{y}$. The attacker's goal is to distinguish between these two cases.

The adversary may choose to expose any user u, learning $K_{\mathsf{SE},u}$ and T_u. After doing so it is no longer able to make ENC queries to that user (as captured by the second "Require" statement in ENC). Note that by the uniformity of $\mathsf{SE.Ext}$ we could instead think of $\vec{y}$ and r as simply being

[5] Computational relaxations of our uniformity and correctness property would suffice for our results, but seem to be unnecessary for any "natural" modes of operation.

picked at random without $\mathsf{SE.Ext}$ being run, but we believe the current framing is conceptually more clear.

We define $\mathsf{Adv}^{\mathsf{ind\text{-}ac\text{-}ext}}_{\mathsf{SE},\mathcal{A}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{ind\text{-}ac\text{-}ext}}_{\mathsf{SE},\mathcal{A}}] - 1$ and say that SE is IND-AC-EXT secure if $\mathsf{Adv}^{\mathsf{ind\text{-}ac\text{-}ext}}_{\mathsf{SE},\mathcal{A}}(\cdot)$ is negligible for all PPT $\mathcal{A}$. This notion will be used for an important step of the coming security proof. Of the properties required from an extraction algorithm it is typically the most difficult to verify.

Example Modes. As a simple example, we can consider counter-mode encryption. For it, we let $\mathsf{CTR.ol}, \mathsf{CTR.il} : \mathbb{N} \to \mathbb{N}$ be fixed and the latter be super-logarithmic. Then CTR is defined as follows. Its key generation algorithm, $\mathsf{CTR.Kg}$, always returns ε. Its sets are defined by

$$\mathsf{CTR.M}(\lambda) = (\{0,1\}^{\mathsf{CTR.ol}(\lambda)})^*, \;\; \mathsf{CTR.Out}(\lambda, l) = \{0,1\}^{l+\mathsf{CTR.il}(\lambda)}$$
$$\mathsf{CTR.FInp}(\lambda) = \{0,1\}^{\mathsf{CTR.il}(\lambda)}, \;\; \mathsf{CTR.FOut}(\lambda) = \{0,1\}^{\mathsf{CTR.ol}(\lambda)}.$$

Algorithms $\mathsf{CTR.Enc}$, $\mathsf{CTR.Dec}$, and $\mathsf{CTR.Ext}$ are defined below where $+$ is addition modulo $2^{\mathsf{CTR.il}(\lambda)}$ with elements of $\{0,1\}^{\mathsf{CTR.il}(\lambda)}$ interpreted as integers.

$\underline{\mathsf{CTR.Enc}^O(1^\lambda, K_{\mathsf{SE}}, m)}$	$\underline{\mathsf{CTR.Dec}^O(1^\lambda, K_{\mathsf{SE}}, c)}$	$\underline{\mathsf{CTR.Ext}(1^\lambda, K, c, m)}$
$c_0 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{CTR.il}(\lambda)}$	$c_0 \,\|\, c' \leftarrow c$	$r \leftarrow c_0$
For $i = 1, \ldots, \lvert m\rvert_{\mathsf{CTR.ol}(\lambda)}$	For $i = 1, \ldots, \lvert c'\rvert_{\mathsf{CTR.ol}(\lambda)}$	For $i = 1, \ldots, \lvert m\rvert_{\mathsf{F.ol}(\lambda)}$
$\quad c_i \leftarrow m_i \oplus O(c_0 + i)$	$\quad m_i \leftarrow c_i \oplus O(c_0 + i)$	$\quad \vec{y}[i] \leftarrow m_i \oplus c_i$
Return c	Return m	Return $(\vec{y}, r)$

It is clear that $\mathsf{CTR.Ext}$ is correct and that its outputs are distributed uniformly when c is picked at random. The IND-AC-EXT security of CTR follows from the probabilistic analysis done in existing proofs of security for CTR, such as the proof of Bellare, Desai, Jokipii, and Rogaway [7]. The standard analysis simply bounds the probability that any of the values $r_1 + 1, \ldots, r_1 + l_1, r_2 + 1, \ldots, r_2 + l_2, \ldots, r_q + 1, \ldots, r_q + l_q$ collide when the r_i's are picked uniformly and the l_i's are adaptively chosen (before the corresponding r_i is chosen).[6]

Other IND-AC-EXT secure modes of operation include cipher-block chaining (CBC), cipher feedback (CFB), and output feedback (OFB).

6.2 Extractability Implies SIM-AC-$ Security

Finally, we can state the main result of this section, that IND-AC-EXT security of an extractable mode of operation implies SIM-AC-$ security.

Theorem 6.1 *Let* SE *be an extractable mode of operation which is IND-AC-EXT secure. Then* $\mathsf{SE}[\mathsf{F}]$ *is SIM-AC-$ secure with* P *whenever* F *is SIM-AC-PRF secure with* P *and satisfies* $\mathsf{F.Inp} = \mathsf{SE.FInp}$ *and* $\mathsf{F.Out} = \mathsf{SE.FOut}$.

The full proof is given in Appendix J. It considers a sequence of games which transition from the real world of $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cpa}}$ to the ideal world (using a simulator we specify). In the first transition we use the security of F to replace SE's oracle access to it with oracle access to a lazily-sampled random function (or simulation by a given simulator $\mathsf{S}_{\mathsf{prf}}$ if the corresponding user has been exposed). Next we modify the game so that (for unexposed users) ciphertexts are chosen at random and then explained by $\mathsf{SE.Ext}$. Then $\mathsf{SE.Enc}$ is run with the chosen random coins and oracle access to this explanation (except for whenever a repeat query is made) to produce a modified ciphertext which

[6] This corresponds exactly to a bound on the BAD-EXT security (defined in Appendix K) of CTR.

is returned. The uniformity of $\mathsf{SE.Ext}$ ensures this game is identical to the prior game. Then we apply the IND-AC-EXT security of SE so that the oracle given to $\mathsf{SE.Enc}$ is not kept consistent on repeated queries. The correctness of $\mathsf{SE.Ext}$ gives that the output of $\mathsf{SE.Enc}$ is equal to the c that was sampled at random. We provide simulator $\mathcal{S}_\$$ that simulates this game perfectly. It runs $\mathsf{S}_{\mathsf{prf}}$ whenever the game would. On an exposure it generate the table T_u for $\mathsf{S}_{\mathsf{prf}}$ by running $\mathsf{SE.Ext}$ on ciphertexts to obtain explanatory outputs of the PRF.

Concretely, in the proof we construct adversaries $\mathcal{A}_{\mathsf{prf}}$ and $\mathcal{A}_{\mathsf{ext}}$ along with simulator $\mathsf{S}_{\mathsf{cpa}}$ for which we show

$$\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE[F]},\mathsf{S}_\$[\mathsf{S}_{\mathsf{cpa}}],\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S}_{\mathsf{prf}},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda) + \mathsf{Adv}^{\mathsf{ind\text{-}ac\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{ext}}}(\lambda).$$

In Appendix K, we show that a variant of IND-AC-EXT security without exposures (which we call IND-EXT) necessarily holds if $\mathsf{SE[F]}$ is single-user IND-\$ secure for all single-user PRF secure F's. Moreover, we identify that the typical way that IND-EXT security is shown in security proofs for SE is by proving a slightly stronger property which *will* suffice to imply IND-AC-EXT security. Thereby, one can obtain a SIM-AC-\$ security proof from a IND-\$ security proof by using the information theoretic core of the existing proof.

6.3 Extensions and a Counter-example Construction

Simple extensions. For encryption schemes not covered by our formalism, it will often be easy to extend the underlying ideas to cover the scheme. Suppose SE uses two distinct function families as PRFs, one could extend our mode of operation syntax to cover this by giving two separate PRF oracles to the encryption and decryption oracles. Then security would follow if there is an extraction algorithm satisfies analogous properties which explains outputs for both of the oracles. The proof would just require an additional step in which the second SIM-AC-PRF is replaced with simulation, as in our transition between games G_0 and G_1.

One can analogously prove the SIM-AC-\$ security of the Encrypt-then-PRF construction, where instead of a second SIM-AC-PRF function family we have a SIM-AC-\$ encryption scheme. From random ciphertexts it is straightforward to extract the required output of the function family and encryption scheme.

We can also extend the analysis to cover GCM when its nonces chosen uniformly at random. It is not captured by our current syntax because the encryption algorithm always applies the PRF to the all-zero string to derive a sub-key for a hash function. It is straightforward to extend our extraction ideas to allow consistency on this PRF query while maintaining our general proof technique.

Non-extractable counterexample. We showed our general security result for extractable modes of operations and described how to extend it for some simple variants. One might optimistically hope that SIM-AC-\$ security would hold for any IND-\$ secure mode of operation (when a SIM-AC-PRF secure function family is used). Unfortunately, we can show that this is not the case. We can provide an example mode of operation which is IND-\$ secure when using a PRF, but not SIM-AC-CPA secure for any choice of function family. It will be clear that this mode of operation is not extractable, as required by our earlier theorem.

Fix $n : \mathbb{N} \to \mathbb{N}$. Let G be a function family that is OW secure with P_{sm} and for which $\mathsf{G.Kg}(1^\lambda)$ always returns ε and $\mathsf{G.Ev}(1^\lambda, \varepsilon, \cdot)$ is always a permutation on $\{0,1\}^{n(\lambda)}$. Such a G us a one-way permutation on n-bits. From G we construct our counterexample CX. It has sets $\mathsf{CX.Out}(\lambda, l) = \{0,1\}^{l+n(\lambda)}$ and $\mathsf{CX.M}(\lambda) = \mathsf{CX.FInp}(\lambda) = \mathsf{CX.FOut}(\lambda) = \{0,1\}^{n(\lambda)}$. Key generation is given by $\mathsf{CX.Kg} = \mathsf{G.Kg}$. Encryption and decryption are given as follows.

$$\begin{array}{l|l}
\underline{\mathsf{CX}.\mathsf{Enc}^{O}(1^\lambda, K_{\mathsf{SE}}, m)} & \underline{\mathsf{CX}.\mathsf{Dec}^{O}(1^\lambda, K_{\mathsf{SE}}, c)} \\
c_0 \leftarrow\!\!\$\ \{0,1\}^{n(\lambda)} & c_0 \,\|\, c_1 \leftarrow c \\
y \leftarrow \mathsf{G}.\mathsf{Ev}^{\varepsilon}(1^\lambda, \varepsilon, O(c_0)) & y \leftarrow \mathsf{G}.\mathsf{Ev}^{\varepsilon}(1^\lambda, \varepsilon, O(c_0)) \\
c_1 \leftarrow y \oplus m & m \leftarrow y \oplus c_1 \\
\text{Return } c & \text{Return } m
\end{array}$$

Above, the superscript ε is used as shorthand for the oracle that always returns ε. Note that this is exactly the behavior of G's expected ideal primitive P_{sm}. This counterexample uses the ideas originally introduced by Fischlin et al. [21] to construct non-programmable random oracles by exploiting a one-way permutation. The construction is not extractable because doing so would require being able to invert the one-way permutation. The following theorem formally establishes that this is a counterexample.

Theorem 6.2 *Fix $n : \mathbb{N} \to \mathbb{N}$. Let G be a one-way permutation on n-bits. Let F be a family of functions with $\mathsf{F}.\mathsf{Out}(\lambda) = \mathsf{F}.\mathsf{Inp}(\lambda) = \{0,1\}^{n(\lambda)}$ and P be an ideal primitive. Then $\mathsf{CX}[\mathsf{F}]$ is IND-\$ secure with P if F is PRF secure with P. However, $\mathsf{CX}[\mathsf{F}]$ is not SIM-AC-CPA secure with P.*

Proof (Sketch): That $\mathsf{CX}[\mathsf{F}]$ is IND-\$ secure when F is PRF secure follows from, e.g., the standard security proof for CTR plus the observation that a permutation applied to a PRF is still a PRF. For the negative result, let S be any simulator and consider the following SIM-AC-CPA adversary $\mathcal{A}_{\mathsf{cpa}}$ and OW adversary $\mathcal{A}$.

$$\begin{array}{l|l}
\underline{\mathcal{A}_{\mathsf{cpa}}^{\textsc{Enc},\textsc{Exp},\textsc{Prim}}(1^\lambda)} & \underline{\mathcal{A}^{\textsc{Prim}}(1^\lambda, K, y)} \\
m \leftarrow\!\!\$\ \{0,1\}^{n(\lambda)} & \sigma \leftarrow \mathsf{S}.\mathsf{Init}(1^\lambda) \\
c_0 \,\|\, c_1 \leftarrow \textsc{Enc}(1, m) & c_0 \,\|\, c_1 \leftarrow\!\!\$\ \mathsf{S}.\mathsf{Enc}(1^\lambda, 1, n(\lambda) : \sigma) \\
y \leftarrow c_1 \oplus m & m \leftarrow c_1 \oplus y \\
(K_{\mathsf{SE}}, K_{\mathsf{F}}) \leftarrow \textsc{Exp}(1) & M.\mathsf{add}(m)\,;\, C.\mathsf{add}(c_0 \,\|\, c_1) \\
x \leftarrow \mathsf{F}.\mathsf{Ev}^{\textsc{Prim}}(1^\lambda, K_{\mathsf{F}}, c_0) & (K_{\mathsf{SE}}, K_{\mathsf{F}}) \leftarrow\!\!\$\ \mathsf{S}.\mathsf{Exp}(1^\lambda, 1, M, C : \sigma) \\
\text{If } \mathsf{G}.\mathsf{Ev}^{\varepsilon}(1^\lambda, \varepsilon, x) = y \text{ then return } 1 & x \leftarrow \mathsf{F}.\mathsf{Ev}^{\mathsf{S}.\mathsf{Prim}(1^\lambda, \cdot : \sigma)}(1^\lambda, K_{\mathsf{F}}, c_0) \\
\text{Return } 0 & \text{Return } x
\end{array}$$

Adversary $\mathcal{A}_{\mathsf{cpa}}$ queries for the encryption of a random message. Then it exposes the corresponding users and uses the given key to calculate the input-output pair this claims for G. If indeed, this is a valid pair it returns 1, otherwise it returns 0. When $b = 1$, note that $\mathcal{A}_{\mathsf{cpa}}$ will always return 1. Intuitively, when $b = 0$, adversary $\mathcal{A}_{\mathsf{cpa}}$ should almost never return 1 because from the perspective of the simulator S it looks like y was chosen at random, so finding a pre-image for it requires breaking the security of G.

This intuition is captured by the adversary $\mathcal{A}$. It simulates the view S would see when run for $\mathcal{A}$, except instead of picking m at random it waits until after running $\mathsf{S}.\mathsf{Enc}$ and sets $m \leftarrow c_1 \oplus y$ where y is the G image it was given as input. Note that y is a uniformly random string because G is a permutation and S is only given the length of the message at this point. Thus, this re-ordering of the calculation of m does not change the view of S. By asking S for the appropriate key and running $\mathsf{F}.\mathsf{Ev}$, the adversary obtains a potential pre-image for y.

Simple calculations give $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda) = 1 - \mathsf{Adv}^{\mathsf{ow}}_{\mathsf{G},\mathsf{P},\mathcal{A}}(\lambda)$. The latter advantage is negligible from the security of G, so the former is non-negligible. ∎

Extensions to PRFs. It is often useful to construct a PRF H with large input domains from a PRF F with smaller input domains. The smaller PRF F is often thought of as being reasonably

modeled by a random oracle or ideal cipher. If the larger construction H is an indifferentiable construction of a random oracle [16, 32], then we can apply Theorem 5.3 to obtain the SIM-AC-PRF security of H.

In the case that H is not indifferentiable, one can often use techniques similar to the above to lift a PRF security proof for H to a SIM-AC-PRF security proof for H whenever F is SIM-AC-PRF secure. Implicit in the existing security proof there will often be a way of "explaining" a random output of H with random outputs by F. On exposure queries, the simulator for H would extract these explanations and feed them to the existing simulator for F to obtain the key to output. For primitive queries, it would just run the F simulator and for evaluation queries after exposure it would just run H using the F simulator in place of F.

Acknowledgments

We thank Thomas Ristenpart for insightful discussions and helpful contributions in the earlier stage of this project.

References

[1] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, Mar. 2002. 27

[2] G. Asharov, M. Naor, G. Segev, and I. Shahaf. Searchable symmetric encryption: optimal locality in linear space via two-dimensional balanced allocations. In D. Wichs and Y. Mansour, editors, *48th ACM STOC*, pages 1101–1114, Cambridge, MA, USA, June 18–21, 2016. ACM Press. 2, 3, 15, 17, 45

[3] G. Asharov, G. Segev, and I. Shahaf. Tight tradeoffs in searchable symmetric encryption. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 407–436, Santa Barbara, CA, USA, Aug. 19–23, 2018. Springer, Heidelberg, Germany. 2, 3, 15, 17, 45

[4] M. Barbosa and P. Farshim. Indifferentiable authenticated encryption. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 187–220, Santa Barbara, CA, USA, Aug. 19–23, 2018. Springer, Heidelberg, Germany. 3

[5] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. 5

[6] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 1–15, Santa Barbara, CA, USA, Aug. 18–22, 1996. Springer, Heidelberg, Germany. 5

[7] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403, Miami Beach, Florida, Oct. 19–22, 1997. IEEE Computer Society Press. 21

[8] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35, Cologne, Germany, Apr. 26–30, 2009. Springer, Heidelberg, Germany. 7

[9] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545, Kyoto, Japan, Dec. 3–7, 2000. Springer, Heidelberg, Germany. 6, 16, 17, 49

[10] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. 52, 54, 60, 67

[11] R. Bost. $\Sigma o\phi o\varsigma$: Forward secure searchable encryption. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 1143–1154, Vienna, Austria, Oct. 24–28, 2016. ACM Press. 2, 3, 15, 17, 45

[12] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648, Philadephia, PA, USA, May 22–24, 1996. ACM Press. 2

[13] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-abuse attacks against searchable encryption. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015*, pages 668–679, Denver, CO, USA, Oct. 12–16, 2015. ACM Press. 4

[14] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *NDSS 2014*, San Diego, CA, USA, Feb. 23–26, 2014. The Internet Society. 2, 3, 13, 15, 17, 30, 31, 32, 45, 46

[15] D. Cash and S. Tessaro. The locality of searchable symmetric encryption. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 351–368, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. 2, 3, 15, 17, 45

[16] J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448, Santa Barbara, CA, USA, Aug. 14–18, 2005. Springer, Heidelberg, Germany. 24

[17] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 79–88, Alexandria, Virginia, USA, Oct. 30 – Nov. 3, 2006. ACM Press. 4, 31

[18] Y. Dodis, T. Ristenpart, J. P. Steinberger, and S. Tessaro. To hash or not to hash again? (In)differentiability results for H^2 and HMAC. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 348–366, Santa Barbara, CA, USA, Aug. 19–23, 2012. Springer, Heidelberg, Germany. 5

[19] M. Etemad, A. Küpçü, C. Papamanthou, and D. Evans. Efficient dynamic searchable encryption with forward privacy. *PoPETs*, 2018(1):5–20, Jan. 2018. 2, 3, 15, 17, 45

[20] M. Fischlin. Pseudorandom function tribe ensembles based on one-way permutations: Improvements and applications. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 432–445, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. 27

[21] M. Fischlin, A. Lehmann, T. Ristenpart, T. Shrimpton, M. Stam, and S. Tessaro. Random oracles with(out) programmability. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 303–320, Singapore, Dec. 5–9, 2010. Springer, Heidelberg, Germany. 23

[22] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 2

[23] F. Hahn and F. Kerschbaum. Searchable encryption with secure and efficient updates. In G.-J. Ahn, M. Yung, and N. Li, editors, *ACM CCS 2014*, pages 310–320, Scottsdale, AZ, USA, Nov. 3–7, 2014. ACM Press. 2, 3, 15, 17, 45

[24] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren. Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 792–800, April 2018. 2, 3, 15, 17, 45

[25] M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS 2012*, San Diego, CA, USA, Feb. 5–8, 2012. The Internet Society. 4

[26] S. Jarecki, H. Krawczyk, and J. Xu. OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 456–486, Tel Aviv, Israel, Apr. 29 – May 3, 2018. Springer, Heidelberg, Germany. 2, 5, 13, 14

[27] S. Kamara and C. Papamanthou. Parallel and dynamic searchable symmetric encryption. In A.-R. Sadeghi, editor, *FC 2013*, volume 7859 of *LNCS*, pages 258–274, Okinawa, Japan, Apr. 1–5, 2013. Springer, Heidelberg, Germany. 2, 3, 15, 17, 45

[28] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic searchable symmetric encryption. In T. Yu, G. Danezis, and V. D. Gligor, editors, *ACM CCS 2012*, pages 965–976, Raleigh, NC, USA, Oct. 16–18, 2012. ACM Press. 2, 3, 15, 17, 45

[29] K. S. Kim, M. Kim, D. Lee, J. H. Park, and W.-H. Kim. Forward secure dynamic searchable symmetric encryption with efficient updates. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017*, pages 1449–1463, Dallas, TX, USA, Oct. 31 – Nov. 2, 2017. ACM Press. 2, 3, 15, 17, 45, 46

[30] J. Li, Y. Huang, Y. Wei, S. Lv, Z. Liu, C. Dong, and W. Lou. Searchable symmetric encryption with forward search privacy. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2019. 2, 3, 15, 17, 45

[31] Q. Liu, Y. Tian, J. Wu, T. Peng, and G. Wang. Enabling verifiable and dynamic ranked search over outsourced data. *IEEE Transactions on Services Computing*, pages 1–1, 2019. 2, 3, 15, 17, 45

[32] U. M. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39, Cambridge, MA, USA, Feb. 19–21, 2004. Springer, Heidelberg, Germany. 3, 24

[33] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126, Santa Barbara, CA, USA, Aug. 18–22, 2002. Springer, Heidelberg, Germany. 11, 28

[34] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In M. K. Reiter and P. Samarati, editors, *ACM CCS 2001*, pages 196–205, Philadelphia, PA, USA, Nov. 5–8, 2001. ACM Press. 30

[35] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. 12, 30

[36] N. Tyagi, M. H. Mughees, T. Ristenpart, and I. Miers. BurnBox: Self-revocable encryption in a world of compelled access. In W. Enck and A. P. Felt, editors, *USENIX Security 2018*, pages 445–461, Baltimore, MD, USA, Aug. 15–17, 2018. USENIX Association. 2, 3, 4, 13, 15, 16, 18, 38, 39, 40, 45, 46, 55, 56, 58

[37] C. Zuo, S. Sun, J. K. Liu, J. Shao, and J. Pieprzyk. Dynamic searchable symmetric encryption schemes supporting range queries with forward (and backward) security. In J. López, J. Zhou, and M. Soriano, editors, *ESORICS 2018, Part II*, volume 11099 of *LNCS*, pages 228–246, Barcelona, Spain, Sept. 3–7, 2018. Springer, Heidelberg, Germany. 2, 3, 15, 17, 45

Supplementary Appendices

A Standard Definitions

In this section we recall standard security definitions for encryption schemes and families of functions. We give multi-user definitions for each of our definitions, the more commonly used single-user version are captured by restricting attention to adversaries that only make queries for a single user. Standard hybrid arguments show that multi-user security is implied by single-user security.

Symmetric encryption. The (multi-user) IND-CPA and IND-CCA security of a symmetric encryption scheme are defined via the games shown in Fig. 9. In it the adversary is given access to a left-or-right oracle LR that returned the encryption of one of two equilength messages depending on a secret bit b. The goal of the adversary is to determine that secret bit. We define the advantage

functions $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda)] - 1$ and $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{ind\text{-}cca}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda)] - 1$. We say SE is IND-CPA secure with P if $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\cdot)$ is negligible for all PPT $\mathcal{A}_{\mathsf{cpa}}$. We say SE is IND-CCA secure with P if $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\cdot)$ is negligible for all PPT $\mathcal{A}_{\mathsf{cca}}$.

Game $\mathrm{G}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda)$	Game $\mathrm{G}^{\mathsf{ind\text{-}cca}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda)$	$\textsc{Prim}(x)$	$\textsc{Lr}(u, m_0, m_1)$
For $u \in \{0,1\}^*$ do	For $u \in \{0,1\}^*$ do	$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$	Require $m_0, m_1 \in \mathsf{SE.M}(\lambda)$
$\quad K_u \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	$\quad K_u \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	Return y	Require $\lvert m_0\rvert = \lvert m_1\rvert$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$	$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$	$\textsc{Dec}(u, c)$	$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, K_u, m_b)$
$b \leftarrow\!\!\$\ \{0,1\}$	$b \leftarrow\!\!\$\ \{0,1\}$	Require $c \notin C_u$	$C_u.\mathsf{add}(c)$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Lr},\textsc{Prim}}_{\mathsf{cpa}}(1^\lambda)$	$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Lr},\textsc{Dec},\textsc{Prim}}_{\mathsf{cca}}(1^\lambda)$	$m_1 \leftarrow \mathsf{SE.Dec}^{\mathsf{P}}(1^\lambda, K_u, c)$	Return c
Return $(b = b')$	Return $(b = b')$	$m_0 \leftarrow \bot$	
		Return m_b	

Figure 9: Games defining multi-user IND-CPA and IND-CCA security of SE.

Game $\mathrm{G}^{\mathsf{ind\text{-}kp}}_{\mathsf{SE},\mathsf{P},\mathcal{A}}(\lambda)$	$\textsc{Prim}(x)$	$\textsc{Lr}(u, m_0, m_1)$
For $u \in \{0,1\}^*$ do	$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$	Require $m_0, m_1 \in \mathsf{SE.M}(\lambda)$
$\quad K^1_u \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	Return y	Require $\lvert m_0\rvert = \lvert m_1\rvert$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$		$K^0_u \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$		$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, K^b_u, m_b)$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Lr},\textsc{Prim}}(1^\lambda)$		Return c
Return $(b = b')$		

Game $\mathrm{G}^{\mathsf{ind\text{-}\$}}_{\mathsf{SE},\mathsf{P},\mathcal{A}}(\lambda)$	$\textsc{Prim}(x)$	$\textsc{Ror}(u, m)$
For $u \in \{0,1\}^*$ do	$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$	Require $m \in \mathsf{SE.M}(\lambda)$
$\quad K_u \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	Return y	If $b = 1$ then $c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, K_u, m)$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$		Else $c \leftarrow\!\!\$\ \mathsf{SE.Out}(\lambda, \lvert m\rvert)$
$b \leftarrow\!\!\$\ \{0,1\}$		Return c
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Ror},\textsc{Prim}}(1^\lambda)$		
Return $(b = b')$		

Figure 10: Games defining multi-user IND-KP and IND-$ security of SE.

The (multi-user) IND-KP security of SE is defined by the game $\mathrm{G}^{\mathsf{ind\text{-}kp}}$ shown in Fig. 10. In it the adversary $\mathcal{A}$ is given access to a left-or-right oracle Lr oracle that returns the encryption of one of two equilength messages depending on the bit b. When $b = 1$ each user has a key K^1_u which is used for every encryption while when $b = 0$ each encryption uses a freshly chosen K^0_u. This game simultaneously captures key and message privacy. We define the advantage function $\mathsf{Adv}^{\mathsf{ind\text{-}kp}}_{\mathsf{SE},\mathsf{P},\mathcal{A}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{ind\text{-}kp}}_{\mathsf{SE},\mathsf{P},\mathcal{A}}(\lambda)] - 1$. We say SE is IND-KP secure with P if for all PPT $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{ind\text{-}kp}}_{\mathsf{SE},\mathsf{P},\mathcal{A}}(\cdot)$ is negligible. Definitions of key-privacy were first given by Fischlin [20] and Abadi and Rogaway [1]. The definition we've given is equivalent to that of Abadi and Rogaway and equivalent to that of Fischlin together with IND-CPA security (up to polynomial multiplicative factors in both cases from hybrid arguments).

The (multi-user) IND-$ security of SE is defined by the game $\mathrm{G}^{\mathsf{ind\text{-}\$}}$ shown in Fig. 10. In it the adversary $\mathcal{A}$ is given access to real-or-random oracle Ror which either returns encryptions of messages of its choice of random strings of the appropriate length. The goal of the adversary is to

distinguish which is the case. We define the advantage functions $\mathsf{Adv}^{\mathsf{ind\text{-}\$}}_{\mathsf{SE},\mathsf{P},\mathcal{A}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{ind\text{-}\$}}_{\mathsf{SE},\mathsf{P},\mathcal{A}}(\lambda)] - 1$. We say SE is IND-\$ secure with P if for all PPT $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{ind\text{-}\$}}_{\mathsf{SE},\mathsf{P},\mathcal{A}}(\cdot)$ is negligible.

We say SE is IND-AE secure with P if it is IND-\$ and INT-CTXT secure with P.

Pseudorandom function security. The (multi-user) security of a family of functions F as a pseudorandom function is given by the game $\mathrm{G}^{\mathsf{prf}}_{\mathsf{F},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda)$ shown in Fig. 11. In it the adversary tries to distinguish whether the oracle Ev is returning random values or outputs of F. For technical reasons because our formulation of ideal primitives is stateful it is important that the code of F is not run when $b = 0$.

Game $\mathrm{G}^{\mathsf{prf}}_{\mathsf{F},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda)$	$\underline{\textsc{Prim}(x)}$	$\underline{\textsc{Ev}(u,x)}$
For $u \in \{0,1\}^*$ do	$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$	If $b = 1$ then $y_1 \leftarrow \mathsf{F.Ev}^{\mathsf{P}}(1^\lambda, K_u, x)$
$\quad K_u \leftarrow\!\!\$\ \mathsf{F.Kg}(1^\lambda)$	Return y	If $T_u[x] = \perp$ then $y_0 \leftarrow\!\!\$\ \mathsf{F.Out}(\lambda)$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$		Else $y_0 \leftarrow T_u[x]$
$b \leftarrow\!\!\$\ \{0,1\}$		$T_u[x] \leftarrow y_0$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Ev},\textsc{Prim}}_{\mathsf{prf}}(1^\lambda)$		Return y_b
Return $b = b'$		

Figure 11: Game defining multi-user PRF security of F.

We define the advantage function $\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{prf}}_{\mathsf{F},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda)] - 1$. We say F is PRF secure with ideal primitive P if for all PPT $\mathcal{A}_{\mathsf{prf}}$, the advantage $\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\cdot)$ is negligible.

Unforgeability. The (multi-user) unforgeability security of a family of functions F is given by the game $\mathrm{G}^{\mathsf{uf\text{-}cma}}$ shown in Fig. 12. In it the adversary is given access to an evaluation oracle, Ev, via which it can obtain F applied to inputs of its choice. Then it has a verification oracle, Vrfy, to which it can specify attempted forgeries. A set S_u is used to prevent it trivially winning by forging with values it learned from Ev. When $b = 1$ it is told whether its forgery was successful, in the ideal world it is always told that its forgery failed.

Game $\mathrm{G}^{\mathsf{uf\text{-}cma}}_{\mathsf{F},\mathsf{P},\mathcal{A}}(\lambda)$	$\underline{\textsc{Prim}(x)}$	$\underline{\textsc{Ev}(u,x)}$	$\underline{\textsc{Vrfy}(u,x,y)}$
For $u \in \{0,1\}^*$ do	$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$	$y \leftarrow \mathsf{F.Ev}^{\mathsf{P}}(1^\lambda, K_u, x)$	Require $x \notin S_u$
$\quad K_u \leftarrow\!\!\$\ \mathsf{F.Kg}(1^\lambda)$	Return y	$S_u \leftarrow S_u \cup \{x\}$	If $b = 1$ then return $(\mathsf{F.Ev}^{\mathsf{P}}(x) = y)$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$		Return y	Else return false
$b \leftarrow\!\!\$\ \{0,1\}$			
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Ev},\textsc{Vrfy},\textsc{Prim}}(1^\lambda)$			
Return $b = b'$			

Figure 12: Game defining unforgeability of F.

We define the advantage function $\mathsf{Adv}^{\mathsf{uf\text{-}cma}}_{\mathsf{F},\mathsf{P},\mathcal{A}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{uf\text{-}cma}}_{\mathsf{F},\mathsf{P},\mathcal{A}}(\lambda)] - 1$. We say F is UF-CMA secure with P if for all PPT $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{uf\text{-}cma}}_{\mathsf{F},\mathsf{P},\mathcal{A}}(\cdot)$ is negligible.

B Standard Model Impossibility

In this section we show that our adaptive compromise simulation definitions are impossible to achieve in the standard model. The techniques used are not novel; we are just applying the ideas of Nielsen [33] to our definition. We will prove the result only for SIM-AC-CPA security. The impossibility of our other definitions follows analogously.

Theorem B.1 *Let* SE *be a perfectly correct symmetric encryption scheme. Then* SE *is not SIM-AC-CPA secure with* $\mathsf{P_{sm}}$.

Proof: Assume, without loss of generality, that for all $\lambda \in \mathbb{N}$ we have $\{0,1\} \subseteq \mathsf{SE.M}(\lambda)$ and $[\mathsf{SE.Kg}(\lambda)] \subseteq \{0,1\}^\lambda$. Then consider the following attack.

$\underline{\mathcal{A}^{\textsc{Enc},\textsc{Ev},\textsc{Prim}}(\lambda)}$
For $i = 1, \ldots, 2\lambda$ do $m_i \leftarrow\!\!\$\ \{0,1\}$; $c_i \leftarrow \textsc{Enc}(1, m_i)$
$K \leftarrow \textsc{Exp}(1)$
For $i = 1, \ldots, 2\lambda$ do $m'_i \leftarrow \mathsf{SE.Dec}^{\mathsf{P_{sm}}}(1^\lambda, K, c_i)$
If $|K| \neq \lambda$ or $\exists i$ s.t. $m_i \neq m'_i$ then return 0
Return 1

We claim that for any choice of S, this adversary has advantage $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S},\mathsf{P_{sm}},\mathcal{A}}(\lambda) \geq 1 - 2^{-\lambda}$. Note that $\mathcal{A}$ will always return 1 when $b = 1$ from the correctness of SE. In the ideal world note that S's choice of each c_i is independent of m_i because the leakage it was given on each query was $\ell = |m_i| = 1$. Each potential $K \in \{0,1\}^\lambda$ that S could return decrypts the sequence of c_i's it returned to a sequence of m_i^K's. The probability that $\mathcal{A}$ outputs 1 in the ideal world is bounded by the probability that the real m_i sequence happens to be one of these potential m_i^K sequences. There are $2^{2\lambda}$ potential real sequences and 2^λ potential m_i^K sequences, giving the stated bound. ∎

C Security Definitions for Nonce-based Encryption

In this section we provide our security definitions of nonce-based encryption. Our definitions are highly analogous to those for randomized encryption.

Nonce-based encryption syntax. A nonce-based symmetric encryption scheme NE specifies algorithms $\mathsf{NE.Kg}$, $\mathsf{NE.Enc}$, and $\mathsf{NE.Dec}$ as well as the set $\mathsf{NE.M}$. The key generation algorithm has syntax $K \leftarrow\!\!\$\ \mathsf{NE.Kg}^{\mathsf{P}}(1^\lambda)$. The encryption algorithm is deterministic and has syntax $c \leftarrow \mathsf{NE.Enc}^{\mathsf{P}}(1^\lambda, K, n, m, a)$. The decryption algorithm is deterministic and has syntax $m \leftarrow \mathsf{NE.Dec}^{\mathsf{P}}(1^\lambda, K, n, c, a)$. Rejection of a ciphertext is represented by returning $m = \perp$.

Informally, correctness requires that encryptions of messages in $\mathsf{NE.M}(\lambda)$ decrypt properly when the same nonce n and associated data a are used in encryption and decryption. We assume the boolean $(m \in \mathsf{NE.M}(\lambda))$ can be efficiently computed.

Security. SIM-AC-CPA security is captured by game $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cpa}}$ shown in Fig. 13. It differs from the SIM-AC-CPA definition for randomized encryption (Fig. 5) only in that $\textsc{Enc}$ additionally takes as input a nonce n and associated data a. These are provided to the simulator. The adversary is disallowed from repeating nonces across queries to single user u via the set N_u.[7]

We define $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{NE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{NE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda)] - 1$. We say NE is SIM-AC-CPA secure with P if for all PPT $\mathcal{A}_{\mathsf{cpa}}$ there exists a PPT S such that $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{NE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\cdot)$ is negligible. Intuitively, this definition captures that ciphertexts reveal nothing about the messages encrypted other than their length unless the encryption key is known to the attacker. Note, however, that this definition does not require anything be hidden about the nonce or associated data.

[7] Our definitions could be extended to capture notions of resilience against nonce-misuse.

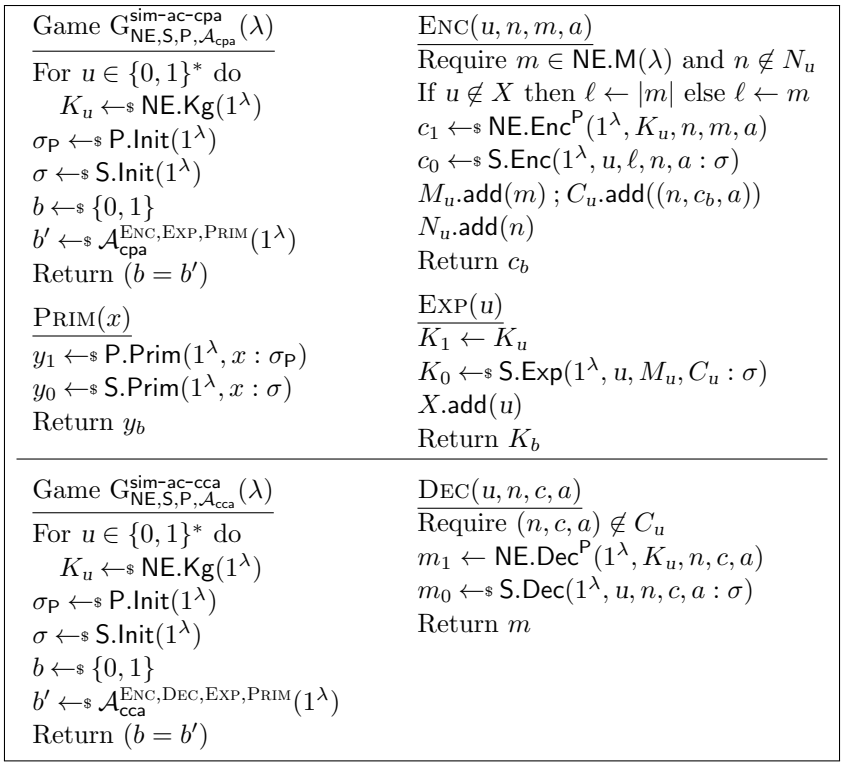

Figure 13: Games defining SIM-AC-CPA and SIM-AC-CCA security of NE.

SIM-AC-CCA security is defined by game $G^{\mathsf{sim\text{-}ac\text{-}cca}}$ defined in Fig. 13. It adds a decryption oracle to $G^{\mathsf{sim\text{-}ac\text{-}cpa}}$ which returns real or simulated decryptions. Via C_u, the adversary is disallowed from making queries to a user u which correspond to ciphertexts previously returned by an encryption query to u. We define $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{NE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda) = 2\Pr[G^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{NE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda)] - 1$ and NE is SIM-AC-CCA secure with P if for all PPT $\mathcal{A}_{\mathsf{cca}}$ there exists a PPT S such that $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{NE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\cdot)$ is negligible.

Stronger security notions. In analysis of nonce-based encryption schemes it is common to target security beyond just CPA and CCA security. Following Rogaway et al. [34], indistinguishable from random bits security is often used in place of CPA security. Following Rogaway and Shrimpton [35], "all-in-one" security notions are often used in place of CCA security. These notions simultaneously capture indistinguishable from random bits security and integrity of ciphertexts. As with randomized encryption, these notions can be captured by considering restricted classes of simulators.

Security conjecture. In Section 5 and Section 6, we saw techniques for extending the analysis of most standard randomized symmetric encryption schemes to imply security according to our new definitions (when underlying components are appropriately idealized). We believe that the same ideas should be applicable for nonce-based encryption schemes.

D Searchable Symmetric Encryption Application

In this section we formally prove the security of a searchable symmetric encryption (SSE) scheme originally proposed by Cash et al. [14]. We start by providing the syntax of what an SSE scheme is and what security is expected of them. Then we proceed to a formal specification of their scheme and our security proof assuming the SIM-AC-PRF security of a function family and the SIM-AC-KP

security of a symmetric encryption scheme.

Syntax. A searchable symmetric encryption scheme SSE specifies an algorithm $\mathsf{SSE.Setup}$ and a protocol $\mathsf{SSE.Srch}$. The setup algorithm has syntax $(K, \mathsf{EDB}) \leftarrow\!\!\$\ \mathsf{SSE.Setup}^{\mathsf{P}}(1^\lambda, \mathsf{DB})$. Given a database DB, it produces an encrypted database EDB to be stored by a server, and a key K to be stored by the client. A database is, for some $d \in \mathbb{N}$, a d-tuple $(id_i, W_i)_{i=1}^d$. Here $id_i \in \{0,1\}^\lambda$ is an identifier and $W \subseteq \{0,1\}^*$ is a keyword-set. In general, the search protocol $\mathsf{SSE.Srch}$ may consist of an arbitrary number of message exchanges between a client and a server. For our purposes we only need to consider a two-message protocol consisting of algorithms $\mathsf{SSE.CSrch}$ and $\mathsf{SSE.SSrch}$. The client search algorithm has syntax $\tau \leftarrow\!\!\$\ \mathsf{SSE.CSrch}^{\mathsf{P}}(1^\lambda, K, w)$. The deterministic server search algorithm has syntax $V \leftarrow \mathsf{SSE.SSrch}^{\mathsf{P}}(1^\lambda, \mathsf{EDB}, \tau)$. Here the client is using its key to produce a search token τ which is sent to the server which then uses it to produce a list V of identifiers associated with that keyword.

For a database $\mathsf{DB} = (id_i, W_i)_{i=1}^d$ and keyword w we let $\mathsf{DB}(w) = \{\ id_i\ :\ w \in W_i\ \}$ and $W(\mathsf{DB}) = \bigcup_{i=1}^d W_i$. Informally speaking, for a correct scheme when the client searches with w the server should produce $V = \mathsf{DB}(w)$. For a formal definition of correctness we refer the reader to [14].

Security. Cash et al. [14] considered a simulation-based definition of security for an SSE scheme (following Curtmola et al. [17]). Consider the game $\mathrm{G}^{\mathsf{sse}}$ shown in Fig. 14. It is parameterized by an SSE scheme SSE, simulator S, ideal primitive P, and adversary $\mathcal{A}_{\mathsf{sse}}$. The goal of the adversary is to determine whether it is in a real ($b = 1$) or ideal ($b = 0$) world. In the real world, the attacker chooses (via Init) a database to be encrypted and then (via Srch) asks for search queries to be made on the encrypted database. We think of the attacker as being an "honest-but-curious" server so it is sees the encrypted database EDB and all search tokens τ that the client sends to the server. The database chosen by the adversary is stored as DB^*. We check whether this has been initialized yet to prevent the adversary from making search queries before initializing the database or from initializing the database multiple times.

In the ideal world, these returned values are chosen by the simulator given only some leakage about the queries made by the adversary. SSE security definitions are typically parameterized by a leakage function which determines the value of ℓ based on all of the queries made by the adversary so far. For concreteness we have hard-coded the leakage of the particular scheme we are considering. On setup we leak the size of the database, i.e. the number of keyword/identifier matches in the database. On a search query for keyword w we leak the correct search results $\mathsf{DB}^*(w)$ and the search pattern sp which is the history of searches that have been made for this keyword.

Game $\mathrm{G}^{\mathsf{sse}}_{\mathsf{SSE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{sse}}}(\lambda)$	$\textsc{Setup}(\mathsf{DB})$	$\textsc{Srch}(w)$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$	Require $\mathsf{DB}^* = \bot$	Require $\mathsf{DB}^* \neq \bot$
$\sigma \leftarrow\!\!\$\ \mathsf{S.Init}(1^\lambda)$	$\mathsf{DB}^* \leftarrow (id_i, W_i)_{i=1}^d \leftarrow \mathsf{DB}$	$j \leftarrow j + 1$
$b \leftarrow\!\!\$\ \{0,1\}$	$\ell \leftarrow \sum_{i=1}^d \lvert W_i \rvert$	$Q_{\mathrm{srch}} \leftarrow Q_{\mathrm{srch}} \cup \{(j, w)\}$
$b' \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{sse}}^{\textsc{Init},\textsc{Srch},\textsc{Prim}}(1^\lambda)$	$(K, \mathsf{EDB}_1) \leftarrow\!\!\$\ \mathsf{SSE.Setup}^{\mathsf{P}}(1^\lambda, \mathsf{DB})$	$\mathsf{sp} \leftarrow \{\ i\ :\ (i, w) \in Q_{\mathrm{srch}}\ \}$
Return $(b = b')$	$\mathsf{EDB}_0 \leftarrow\!\!\$\ \mathsf{S.Setup}(1^\lambda, \ell : \sigma)$	$\ell \leftarrow (\mathsf{DB}^*(w), \mathsf{sp})$
$\textsc{Prim}(x)$	Return EDB_b	$\tau_1 \leftarrow\!\!\$\ \mathsf{SSE.CSrch}^{\mathsf{P}}(1^\lambda, K, w)$
$y_1 \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$		$\tau_0 \leftarrow\!\!\$\ \mathsf{S.CSrch}(1^\lambda, \ell : \sigma)$
$y_0 \leftarrow\!\!\$\ \mathsf{S.Prim}(1^\lambda, x : \sigma)$		Return τ_b
Return y_b		

Figure 14: Game defining security of searchable symmetric encryption scheme SSE.

We define $\mathsf{Adv}^{\mathsf{sse}}_{\mathsf{SSE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{sse}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{sse}}_{\mathsf{SSE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{sse}}}(\lambda)] - 1$. We say SSE is secure with P if for all

PPT $\mathcal{A}_{\mathsf{sse}}$ there exists a PPT S such that $\mathsf{Adv}^{\mathsf{sse}}_{\mathsf{SSE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{sse}}}(\cdot)$ is negligible.

Basic Scheme. We will analyze the scheme Π_{bas} which was designed by Cash et al. [14]. This scheme makes use of a PRF F and a symmetric encryption scheme SE. We require that $\mathsf{F.Inp}(\lambda) = \{0,1\}^*$, that $\mathsf{F.Out}(\lambda) = \{0,1\}^{\mathsf{F.ol}}(\lambda)$ for some $\mathsf{F.ol} : \mathbb{N} \to \mathbb{N}$, and that $\mathsf{SE.Kg}$ and $\mathsf{F.Kg}$ return uniform elements from $\mathsf{F.Out}$. If F expects access to ideal primitive P_1 and SE expects access to ideal primitive P_2 then $\mathsf{EtM}[\mathsf{SE},\mathsf{F}]$ expects access to $\mathsf{P}_1 \times \mathsf{P}_2$. Pseudocode for Π_{bas} is given below.

The scheme additionally makes use of a (static) dictionary. This consists of an algorithm Dict which has syntax $T \leftarrow\!\!\$\ \mathsf{Dict}(L)$. It takes as input a list L where each $L[i]$ is a tuple (l_i, c_i) and outputs a table T. Correctness requires that if $T[l] = \perp$ then $l \neq l_i$ for all i and if $T[l] \neq \perp$ then there exists some i such that $l = l_i$ and $T[l] = l_i$. We say Dict is history independent if the output of $\mathsf{Dict}(L)$ does not depend on the order of entries in L.

$\underline{\Pi_{\mathsf{bas}}.\mathsf{Setup}^{\mathsf{P}_1\times\mathsf{P}_2}(1^\lambda, \mathsf{DB})}$
$K \leftarrow\!\!\$\ \mathsf{F.Kg}$
For $w \in W(\mathsf{DB})$ do
  $K_1 \leftarrow \mathsf{F.Ev}^{\mathsf{P}_1}(1^\lambda, K, 1 \,\|\, w)$
  $K_2 \leftarrow \mathsf{F.Ev}^{\mathsf{P}_1}(1^\lambda, K, 2 \,\|\, w)$
  $i \leftarrow 0$
  For $id \in \mathsf{DB}(w)$
    $l \leftarrow \mathsf{F.Ev}^{\mathsf{P}_1}(1^\lambda, K_1, i)$
    $c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}_2}(1^\lambda, K_2, id)$
    $L.\mathsf{add}((l,c))$
    $i \leftarrow i + 1$
$\mathsf{EDB} \leftarrow \mathsf{Dict}(L)$
Return (K, EDB)

$\underline{\Pi_{\mathsf{bas}}.\mathsf{CSrch}^{\mathsf{P}_1\times\mathsf{P}_2}(1^\lambda, K, w)}$
$K_1 \leftarrow \mathsf{F.Ev}^{\mathsf{P}_1}(1^\lambda, K, 1 \,\|\, w)$
$K_2 \leftarrow \mathsf{F.Ev}^{\mathsf{P}_1}(1^\lambda, K, 2 \,\|\, w)$
Return (K_1, K_2)

$\underline{\Pi_{\mathsf{bas}}.\mathsf{SSrch}^{\mathsf{P}_1\times\mathsf{P}_2}(1^\lambda, \mathsf{EDB}, \tau)}$
$i \leftarrow 0$
$V \leftarrow \emptyset$
$(K_1, K_2) \leftarrow \tau$
Loop
  $l \leftarrow \mathsf{F.Ev}^{\mathsf{P}_1}(1^\lambda, K_1, i)$
  $c \leftarrow \mathsf{EDB}[l]$
  If $c = \perp$ then exit loop
  $id \leftarrow \mathsf{SE.Dec}^{\mathsf{P}_2}(1^\lambda, K_2, c)$
  $V \leftarrow V \cup \{id\}$
  $i \leftarrow i + 1$
Return V

To construct the initial database, $\Pi_{\mathsf{bas}}.\mathsf{Setup}$ first samples a key for F that will be used to derive per-keyword keys for F and SE. Then for each keyword w in $W(\mathsf{DB})$ it loops through each identifier id in $\mathsf{DB}(w)$ while iterating a counter i. It applies F to each i (with the per-keyword key K_1) to obtain a label l and uses SE to encrypt id (with the per-keyword key K_2) to obtain a ciphertext c. These (l, c) pairs are added to a list L which is then used to create the dictionary EDB.

To search for a keyword w, the client uses its key to rederive the per-keyword keys and send them to the server. The server can then rederive the labels and use them to index into EDB to obtain the corresponding ciphertexts. Decrypting them give the search results.

Theorem D.1 *If* F *is PRF secure with* P_1*,* F *is SIM-AC-PRF secure with* P_1*,* SE *is SIM-AC-KP secure with* P_2*, and* Dict *is history independent then* Π_{bas} *is secure with* $\mathsf{P}_1 \times \mathsf{P}_2$*.*

The assumption that F is PRF secure is technically superfluous because it can be shown to be implied by the SIM-AC-PRF security of F.

Proof: Let $\mathcal{A}_{\mathsf{sse}}$ be an adversary against the security of Π_{bas}. We will construct adversary $\mathcal{B}_{\mathsf{prf}}$ against the PRF security of F, $\mathcal{A}_{\mathsf{prf}}$ against the SIM-AC-PRF security of F, and adversary $\mathcal{A}_{\mathsf{cpa}}$ against the SIM-AC-CPA security of SE. Then, given PPT simulator $\mathsf{S}_{\mathsf{prf}}$ and PPT simulator $\mathsf{S}_{\mathsf{cpa}} \in \mathcal{S}_k$, we construct an SSE simulator $\mathsf{S}_{\mathsf{sse}}$ such that the following holds.

$$\mathsf{Adv}^{\mathsf{sse}}_{\Pi_{\mathsf{bas}},\mathsf{S}_{\mathsf{sse}},\mathsf{P}_1\times\mathsf{P}_2,\mathcal{A}_{\mathsf{sse}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F},\mathsf{P}_1,\mathcal{B}_{\mathsf{prf}}}(\lambda) + \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S}_{\mathsf{prf}},\mathsf{P}_1,\mathcal{A}_{\mathsf{prf}}}(\lambda) + \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S}_{\mathsf{cpa}},\mathsf{P}_2,\mathcal{A}_{\mathsf{cpa}}}(\lambda).$$

The new algorithms we introduce are PPT, so the theorem holds by letting $\mathsf{S}_{\mathsf{prf}}$ and $\mathsf{S}_{\mathsf{cpa}}$ be the simulators which makes the respective advantage functions negligible.

The proof considers games G_0 through G_5. These games gradually transform the view of the adversary from being generated by Π_{bas} (in G_0) to being generated in a way that can be emulated perfectly by a simulator (in G_5). The inequality above follows from simple calculations based on the following claims which we will justify.

1. $\mathsf{Adv}^{\mathsf{sse}}_{\Pi_{\mathsf{bas}},\mathsf{S}_{\mathsf{sse}},\mathsf{P}_1\times\mathsf{P}_2,\mathcal{A}_{\mathsf{sse}}}(\lambda) = \Pr[G_0(\lambda)] - \Pr[G_5(\lambda)]$

2. $\Pr[G_0(\lambda)] - \Pr[G_1(\lambda)] = \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F},\mathsf{P}_1,\mathcal{B}_{\mathsf{prf}}}(\lambda)$

3. $\Pr[G_1(\lambda)] - \Pr[G_2(\lambda)] = 0$

4. $\Pr[G_2(\lambda)] - \Pr[G_3(\lambda)] = \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S}_{\mathsf{prf}},\mathsf{P}_1,\mathcal{A}_{\mathsf{prf}}}(\lambda)$

5. $\Pr[G_3(\lambda)] - \Pr[G_4(\lambda)] = 0$

6. $\Pr[G_4(\lambda)] - \Pr[G_5(\lambda)] = \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S}_{\mathsf{cpa}},\mathsf{P}_2,\mathcal{A}_{\mathsf{cpa}}}(\lambda)$

Games $G_0(\lambda)$, $\boxed{G_1(\lambda)}$

$\sigma'_{\mathsf{P}} \leftarrow\$ \mathsf{P}_1.\mathsf{Init}(1^\lambda)$
$\sigma''_{\mathsf{P}} \leftarrow\$ \mathsf{P}_2.\mathsf{Init}(1^\lambda)$
$b' \leftarrow\$ \mathcal{A}^{\text{INIT,SRCH,PRIM}}_{\mathsf{sse}}(1^\lambda)$
Return $(b' = 1)$

PRIM(x)
$(d, x) \leftarrow x$
If $d = 1$ then
  $y \leftarrow\$ \mathsf{P}_1.\mathsf{Prim}(x : \sigma'_{\mathsf{P}})$
Else
  $y \leftarrow\$ \mathsf{P}_2.\mathsf{Prim}(x : \sigma''_{\mathsf{P}})$
Return y

SRCH(w)
Require $\mathsf{DB}^* \neq \perp$
If $w \notin W(\mathsf{DB})$ and $K_{1,w} = \perp$ then
  $K_{1,w} \leftarrow \mathsf{F.Ev}^{\mathsf{P}_1}(1^\lambda, K, 1 \,\|\, w)$
  $K_{2,w} \leftarrow \mathsf{F.Ev}^{\mathsf{P}_1}(1^\lambda, K, 2 \,\|\, w)$
  $\boxed{K_{1,w} \leftarrow\$ \mathsf{F.Out}(\lambda)}$
  $\boxed{K_{2,w} \leftarrow\$ \mathsf{F.Out}(\lambda)}$
Return $(K_{1,w}, K_{2,w})$

SETUP(DB)
Require $\mathsf{DB}^* = \perp$
$\mathsf{DB}^* \leftarrow \mathsf{DB}$
$K \leftarrow\$ \mathsf{F.Kg}$
For $w \in W(\mathsf{DB})$ do
  $K_{1,w} \leftarrow \mathsf{F.Ev}^{\mathsf{P}_1}(1^\lambda, K, 1 \,\|\, w)$
  $K_{2,w} \leftarrow \mathsf{F.Ev}^{\mathsf{P}_1}(1^\lambda, K, 2 \,\|\, w)$
  $\boxed{K_{1,w} \leftarrow\$ \mathsf{F.Out}(\lambda)}$
  $\boxed{K_{2,w} \leftarrow\$ \mathsf{F.Out}(\lambda)}$
  $i \leftarrow 0$
  For $id \in \mathsf{DB}(w)$
    $l_w[i] \leftarrow \mathsf{F.Ev}^{\mathsf{P}_1}(1^\lambda, K_{1,w}, i)$
    $c \leftarrow\$ \mathsf{SE.Enc}^{\mathsf{P}_2}(1^\lambda, K_{2,w}, id)$
    $L.\mathsf{add}((l_w[i], c))$
    $i \leftarrow i + 1$
$\mathsf{EDB} \leftarrow \mathsf{Dict}(L)$
Return EDB

Figure 15: Games G_0 and G_1 used in proof of Theorem D.1. Highlighted code is only included in G_0. Boxed code is only included in game G_1.

Claim 2. We start with the second claim. The games G_0 and G_1 are shown in Fig. 15. Highlighted code is only included in G_0. Boxed code is only included in game G_1. Game G_0 is identical to G^{sse} when $b = 1$. It was obtained by plugging in the code of Π_{bas} and $\mathsf{P}_1 \times \mathsf{P}_2$ and making some simplification. The most important changes was remembering the per-keyword keys $K_{1,w}$ and $K_{2,w}$ instead of recomputing them in SRCH. In the case that a w not in the database is queried its key is still computed in SRCH.

These games differ only in the computation of keys $K_{1,w}$ and $K_{2,w}$ in SETUP. In G_0, these are the output of F while in G_1 they are chosen uniformly at random. Consequently, distinguishing between these two games corresponds quite naturally to breaking the PRF security of F. The adversary $\mathcal{B}_{\mathsf{prf}}$ establishing this is shown in Fig. 16. It runs $\mathcal{A}_{\mathsf{sse}}$ as a subroutine, as if it were in these two games except it calls its own PRIM oracle whenever P_1 would be run and the computation of keys $K_{1,w}$ and $K_{2,w}$ is done via the oracle EV which is either executing F (as in G_0) or picking them at random (as in G_1). For these EV queries it arbitrarily uses $u = 0$. The view of $\mathcal{A}_{\mathsf{sse}}$ thus perfectly matches its view in G_0 when $b = 1$ in G^{prf} and perfectly matches its view in G_1 when $b = 0$. Hence, standard probability analysis gives claim 2.

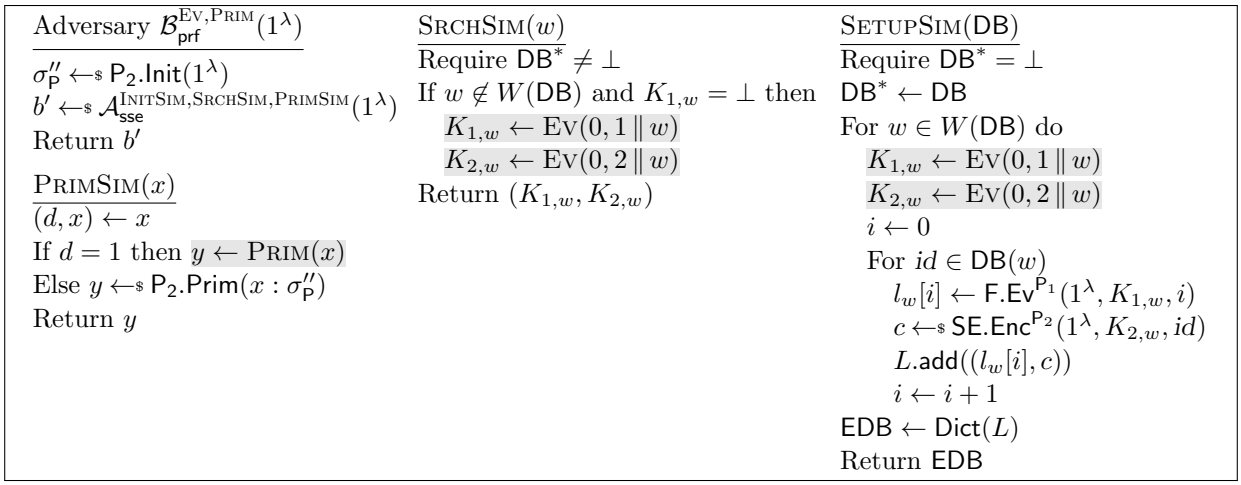

Figure 16: Adversary $\mathcal{B}_{\mathsf{prf}}$ used in proof of Theorem D.1. Highlighting indicates the "interesting" lines of code.

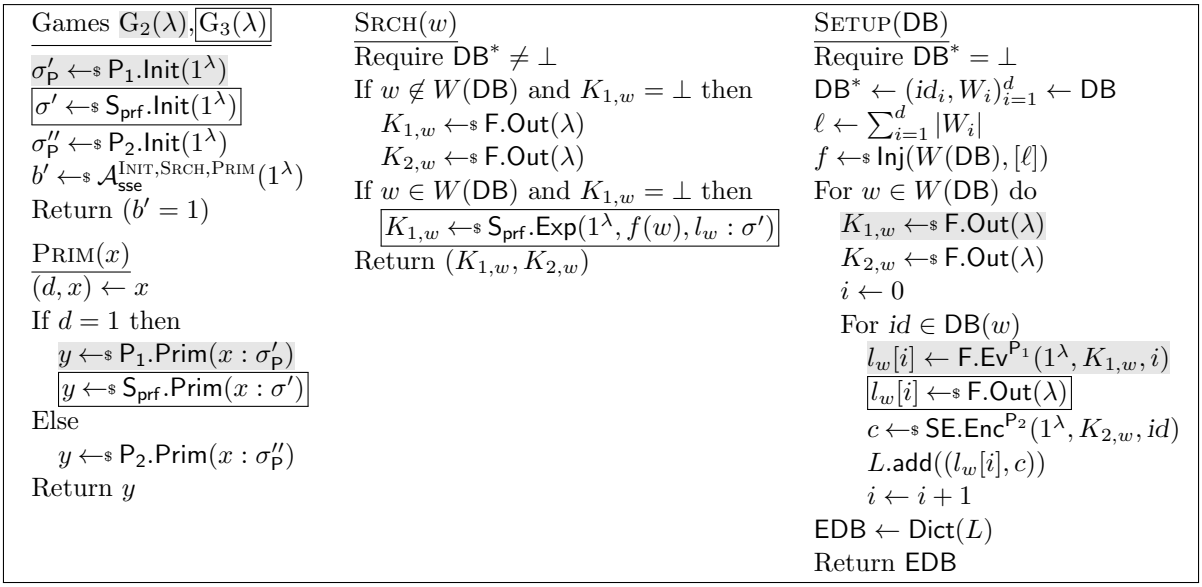

Figure 17: Games G_2 and G_3 used in proof of Theorem D.1. Highlighted code is only included in G_2. Boxed code is only included in game G_3.

Claim 3. For the third claim, consider game G_2 shown in Fig. 17. It contains the highlighted code while game G_3 contains the boxed code. It is an exact copy of G_1, except for small modification to Setup which does no affect its behavior. The line sampling K (which is not used anywhere) in Setup has been removed. Additionally, code has been added to compute the size of the database ℓ and create an injection from the keywords in the database to $[\ell]$. This injection is unused in G_2. The claim follows trivially from this observation.

Claim 4. Now consider game G_3 shown in the same figure. It differs from G_2 in that the use of F is being replaced with simulation. The keys $K_{1,w}$ are chosen in Srch by $\mathsf{S}_{\mathsf{prf}}$ instead of randomly in Setup. Here, the simulator is given $f(w)$ as the name of the "user" for which it is producing a

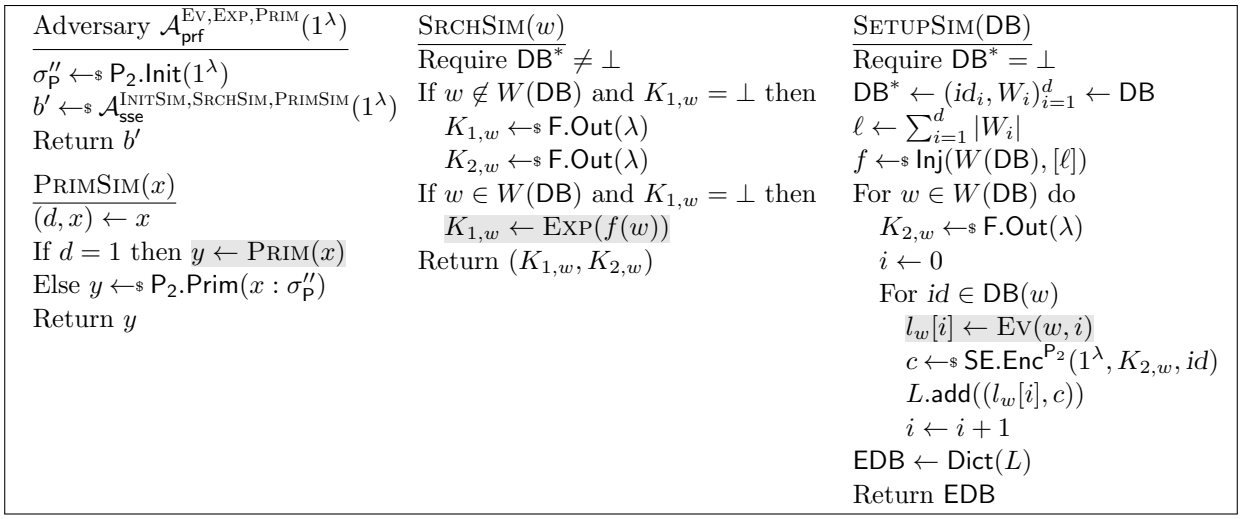

Figure 18: Adversary $\mathcal{A}_{\mathsf{prf}}$ used in proof of Theorem D.1. Highlighting indicates the "interesting" lines of code.

key.[8] The ideal primitive P_1 has been replace with simulation. Most importantly, the output of F in Setup is instead being sampled uniformly at random.

Distinguishing these differences reduces to breaking the SIM-AC-PRF security of F. The adversary $\mathcal{A}_{\mathsf{prf}}$ establishing this is shown in Fig. 18. It runs $\mathcal{A}_{\mathsf{sse}}$ as in the games, replacing the code which differs between the games with calls to its own oracles. These replacements are indicated by highlighting. Note that the "Require" statements in SrchSim and SetupSim enforce that no Exp(w, i) queries are made after a Exp(w) query has already been made. Consequently, when $b = 0$ in $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}prf}}$ the output of Ev is always chosen uniformly at random. Consequently, the view of $\mathcal{A}_{\mathsf{sse}}$ will perfectly match its view in G_2 when $b = 1$ and perfectly match its view in G_3 when $b = 0$. Hence, standard probability analysis gives claim 4.

Claim 5. For the fifth claim, consider game G_4 shown in Fig. 19. It contains the highlighted code while game G_5 contains the boxed code. It is an exact copy of G_4 so the claim follows trivially.

Claim 6. Now consider game G_5 shown in the same figure. It differs from G_4 in that the use of SE is being replaced with simulation. The keys $K_{2,w}$ are chosen in Srch by $\mathsf{S}_{\mathsf{cpa}}$ instead of randomly in Setup. The ideal primitive P_2 has been replace with simulation. Most importantly, the output of SE in Setup is instead being emulated by $\mathsf{S}_{\mathsf{cpa}}.\mathsf{Enc}_1$.

Distinguishing these differences reduces to breaking the SIM-AC-KP security of SSE. The adversary establishing this is shown in Fig. 20. It runs $\mathcal{A}_{\mathsf{sse}}$ as in these games, replacing the code which differs between the games with calls to its own oracles. These replacements are indicated by highlighting. The "Require" statements in SrchSim and SetupSim enforce that no Exp(w, id) queries are made after a Exp(w) query has already been made. Consequently, when $b = 0$ in $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cpa}}$ the output of $\mathsf{S}_{\mathsf{cpa}}$ in Enc is always chosen by $\mathsf{S}_{\mathsf{cpa}}.\mathsf{Enc}_1$ instead of $\mathsf{S}_{\mathsf{cpa}}.\mathsf{Enc}_2$. Consequently, the view of $\mathcal{A}_{\mathsf{sse}}$ perfectly matches its view in G_4 when $b = 1$ and perfectly matches its view in G_5 when $b = 0$. Standard probability analysis gives claim 6.

Claim 1. The justify claim 1 we need to argue that the view of $\mathcal{A}_{\mathsf{sse}}$ in G_0 perfectly matches its

[8] We use $f(w)$ instead of w because our eventual simulator must emulate this name without knowing w.

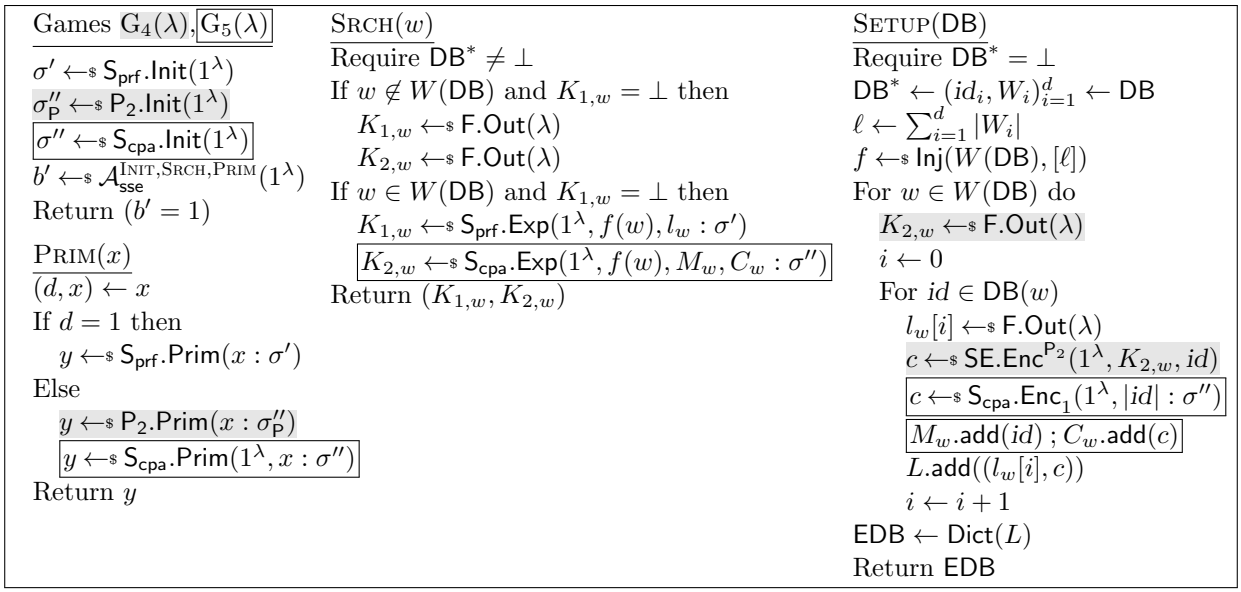

Figure 19: Games G_4 and G_5 used in proof of Theorem D.1. Highlighted code is only included in G_4. Boxed code is only included in game G_5.

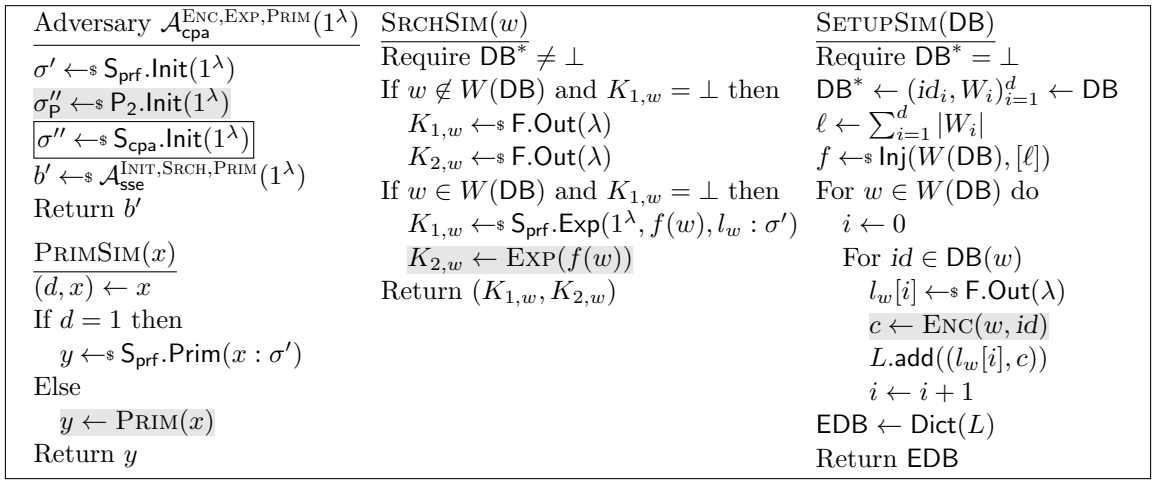

Figure 20: Adversary $\mathcal{A}_{\mathsf{cpa}}$ used in proof of Theorem D.1. Highlighting indicates the "interesting" lines of code.

view in $\mathrm{G}^{\mathsf{sse}}$ when $b = 1$ and that its view in G_5 perfectly matches its view in $\mathrm{G}^{\mathsf{sse}}$ when $b = 0$ (when using a simulator we will define). The claim then follows from standard conditional probability calculations. The former argument was already made in our discussion of claim 2. For the latter, consider the following simulator $\mathsf{S}_{\mathsf{sse}}$.

$\underline{\mathsf{S}_{\mathsf{sse}}.\mathsf{Init}(1^\lambda)}$
$\sigma' \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Init}(1^\lambda)$
$\sigma'' \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{cpa}}.\mathsf{Init}(1^\lambda)$
Return $(\sigma', \sigma'', \emptyset, \bot, \bot)$

$\underline{\mathsf{S}_{\mathsf{sse}}.\mathsf{Prim}(1^\lambda, x : (\sigma', \sigma'', U, L, K_{(\cdot)}))}$
$(d, x) \leftarrow x$
If $d = 1$ then $y \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Prim}(x : \sigma')$
Else $y \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{cpa}}.\mathsf{Prim}(1^\lambda, x : \sigma'')$
Return y

$\underline{\mathsf{S}_{\mathsf{sse}}.\mathsf{Setup}(1^\lambda, \ell : (\sigma', \sigma'', U, L, K_{(\cdot)}))}$
$U \leftarrow [\ell]$
For $i = 1, \ldots, \ell$ do
  $l \leftarrow\!\!\$\ \mathsf{F}.\mathsf{Out}(\lambda)$
  $c \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{cpa}}.\mathsf{Enc}_1(1^\lambda, \lambda : \sigma'')$
  $L.\mathsf{add}((l, c))$
$\mathsf{EDB} \leftarrow \mathsf{Dict}(L)$
Return EDB

$\underline{\mathsf{S}_{\mathsf{sse}}.\mathsf{CSrch}(1^\lambda, \ell : (\sigma', \sigma'', U, L, K_{(\cdot)}))}$
$(S, \mathsf{sp}) \leftarrow \ell$
$j \leftarrow \min \mathsf{sp}$
If $|\mathsf{sp}| = 1$ then
  If $S = \emptyset$ then
    $K_{1,j} \leftarrow\!\!\$\ \mathsf{F}.\mathsf{Out}(\lambda)$
    $K_{2,j} \leftarrow\!\!\$\ \mathsf{F}.\mathsf{Out}(\lambda)$
  Else
    For $id \in S$ do
      $(l, c) \leftarrow L.\mathsf{dq}()$
      $l.\mathsf{add}(l)\ ; M.\mathsf{add}(id)\ ; C.\mathsf{add}(c)$
    $u \leftarrow\!\!\$\ U\ ; U \leftarrow U \setminus \{u\}$
    $K_{1,j} \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Exp}(1^\lambda, u, l : \sigma')$
    $K_{2,j} \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{cpa}}.\mathsf{Exp}(1^\lambda, u, M, C : \sigma'')$
Return $(K_{1,j}, K_{2,j})$

This simulator runs copies of $\mathsf{S}_{\mathsf{prf}}$ and $\mathsf{S}_{\mathsf{cpa}}$. In addition to their states (σ' and σ''), it stores set U, list L, and table K. Primitive queries are simulated by running the appropriate $\mathsf{S}_{\mathsf{prf}}$ or $\mathsf{S}_{\mathsf{cpa}}$ algorithm.

Now consider the behavior of SETUP in G_5. For each (w, id)-pair in the database, a label-ciphertext pair is added to the list L. This pair was created by picking the label at random and picking the ciphertext according to $\mathsf{S}_{\mathsf{cpa}}.\mathsf{Enc}_1$. Because each id has the same length, λ, these label-ciphertext pairs are independent of the (w, id)-pair for which they were created. Consequently, SSE is able to simulate EDB properly by generating the appropriate number of label-ciphertext pairs. (Recall that the number of such pairs is the leakage it is given as input.)

The crux of the simulation lies in the simulation of the keys returned by SRCH. In game G_5 these are picked at random if $w \notin W(\mathsf{DB})$ and otherwise are chosen by $\mathsf{S}_{\mathsf{prf}}$ and $\mathsf{S}_{\mathsf{cpa}}$ given the appropriate lists l_w, M_w, and C_w along with the name, $f(w)$, for the "user" associated with that keyword. If a search query is repeated, the same keys are the prior query are returned.

We can consider the simulation of $\mathsf{S}_{\mathsf{sse}}$ is similar cases. It is given as leakage the results of this query $\mathsf{DB}(w)$ which is calls S and the search patter for this keyword sp. If $|\mathsf{sp}| \neq 1$, then this keyword has been searched for before and the simulator will return the same keys it did last time. The value $\min \mathsf{sp}$ will be invariant across all searches for a keyword (and not repeat for different keywords) so it is used to index into the table $K_{(\cdot)}$ which $\mathsf{S}_{\mathsf{sse}}$ uses to store all keys that it has previously returned.

Now suppose it is the first time a keyword has been queries (i.e. $|\mathsf{sp}| = 1$). If $S = \emptyset$, then this keyword is not in the database so the simulator can simply pick the keys to return at random. If $S \neq \emptyset$, then the keyword is in the database. Properly simulating this boils down to properly simulating l_w, M_w, C_w, and $f(w)$ which can then be used to run $\mathsf{S}_{\mathsf{prf}}$ and $\mathsf{S}_{\mathsf{cpa}}$. To simulate the tables, $\mathsf{S}_{\mathsf{sse}}$ exploits the fact the history independence of Dict. This means that EDB (and hence the view of $\mathcal{A}_{\mathsf{sse}}$) is independent of the order that values are stored in L. So $\mathsf{S}_{\mathsf{sse}}$ can simply remove the last $|S|$ entries of L and, pretending they were the entries corresponding to the keyword being searched, use them to compute the tables for $\mathsf{S}_{\mathsf{prf}}$ and $\mathsf{S}_{\mathsf{cpa}}$. Since f in G_5 is a random injection,

$\mathsf{S}_{\mathsf{sse}}$ emulates $f(w)$ by sampling u from the appropriate set without replacement.

The above justifies the claim that the view of $\mathcal{A}_{\mathsf{sse}}$ in G_5 perfectly matches its view in $\mathrm{G}^{\mathsf{sse}}$ when $b = 0$. This completes our justification of claim 1 and hence completes the proof. ∎

E Self-revocable Encrypted Cloud Storage Application

In this section, we look at compelled access security for encrypted cloud storage. This setting considers the security of documents stored on the cloud, even in the event that keys are compromised on the client device, as in a compelled access search, e.g., at a border crossing. We first recall the compelled access security definitions and construction called Burnbox proposed by Tyagi et al. [36]. Then we show that our adaptive security definitions for symmetric encryption can replace the ideal encryption model used in the original proof.

Before we proceed, let us recall standard public key encryption definitions.

Public Key Encryption Syntax. A public key encryption scheme PKE specifies algorithms $\mathsf{PKE.Kg}$, $\mathsf{PKE.Enc}$, and $\mathsf{PKE.Dec}$ as well as the set $\mathsf{PKE.M}$ representing the message space. The key generation algorithm has syntax $(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda)$ producing a public encryption key ek and a secret decryption key dk. The encryption algorithm has syntax $c \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\mathsf{P}}(1^\lambda, ek, m)$. The decryption algorithm is deterministic and has syntax $m \leftarrow \mathsf{PKE.Dec}^{\mathsf{P}}(1^\lambda, dk, c)$.

Informally, correctness requires that encryptions of messages in $\mathsf{PKE.M}(\lambda)$ decrypt properly. We assume the boolean $(m \in \mathsf{PKE.M}(\lambda))$ can be efficiently computed.

Game $\mathrm{G}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathsf{P},\mathcal{A}}(\lambda)$	$\textsc{Lr}(m_0, m_1)$	$\textsc{Prim}(x)$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda)$	Require $m_0, m_1 \in \mathsf{PKE.M}(\lambda)$	$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$	Require $\lvert m_0\rvert = \lvert m_1\rvert$	Return y
$b \leftarrow\!\!\$\ \{0,1\}$	$c_b \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\mathsf{P}}(1^\lambda, ek, m_b)$	
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Lr},\textsc{Prim}}(1^\lambda, ek)$	$C_u.\mathsf{add}(c_b)$	
Return $(b = b')$	Return c_b	

Figure 21: Game defining IND-CPA security of public key encryption scheme PKE.

Public Key Encryption Security. The IND-CPA security of a public key encryption scheme is defined via the game shown in Fig. 21. In this game, the adversary is tasked with determining the decryption of challenge ciphertexts output from $\textsc{Lr}$ to one of two distinct self-chosen plaintexts. We define the advantage function $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathsf{P},\mathcal{A}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathsf{P},\mathcal{A}}(\lambda)] - 1$. We say PKE is IND-CPA secure with P if for all PPT $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathsf{P},\mathcal{A}}(\cdot)$ is negligible.

E.1 Proof of Compelled Access Security for BurnBox

Syntax. A revocable cloud storage scheme RCS specifies algorithms $\mathsf{RCS.Init}$, $\mathsf{RCS.Add}$, $\mathsf{RCS.Access}$, $\mathsf{RCS.Delete}$, $\mathsf{RCS.Revoke}$, $\mathsf{RCS.Restore}$. The algorithms interface with a remote server through a key-value abstraction making $\mathsf{Put}(K, V)$ and $\mathsf{Get}(K)$ requests. Both Put and Get are available as oracles to all RCS algorithms, though we omit their explicit mention from notation for simplicity. The initialization algorithm has syntax $(\sigma, k) \leftarrow\!\!\$\ \mathsf{RCS.Init}(1^\lambda)$ which takes in a security parameter and outputs a state variable (implicitly used by all remaining algorithms) and a restoration key. The file add algorithm has syntax $\mathsf{RCS.Add}(lbl, m)$ and takes in file contents m along with a label string

lbl of length $\mathsf{RCS.lbll}(\lambda)$ to be used to handle the file. RCS.Access, RCS.Delete, and RCS.Revoke each take in the file label and, respectively, return the file contents, delete access to a file permanently, and revoke access to a file temporarily. Lastly, RCS.Restore can be called with the restoration key created during initialization to restore access to files that were temporarily revoked via RCS.Revoke.

Security. Tyagi et al. [36] consider a simulation-based definition of RCS in which the adversary's view consists of the transcript of put/get operations made to the key-value store. The original presentation the security game, called compelled access security (CA), is parameterized by an RCS scheme, an ideal primitive P, a simulator S, a leakage regime, and an adversary; it differs slightly from ours ($\mathrm{G}^{\mathsf{ca\text{-}poh}}$ in Figure 22). In both presentations, the task of the adversary is to determine whether they are in the real ($b = 1$) or the ideal ($b = 0$) world.

In the real world, the adversary interacts with oracles to add, access, delete, revoke, and restore files, each of these oracles executes the appropriate algorithm of the RCS scheme. The RCS algorithms are implicitly wrapped to replace their return value with the transcript of key-value operations made during execution of the algorithm, denoted with special tokens: $(\mathrm{P}, (K, V))$ for a Put and (G, K) for a Get. Additionally, a one-time use compromise oracle is provided to return the client state variable, representing compelled access to the client device.

In the ideal world, these returned values are chosen by the simulator given only some leakage about the adversary's queries as determined by the leakage regime. We will only be working with one leakage regime: the pseudonymous operation history leakage regime used by the Burnbox construction. To simplify our presentation, $\mathrm{G}^{\mathsf{ca\text{-}poh}}$ includes the pseudocode for compelled access security for pseudonymous operation history leakage (CA-POH). Intuitively, this corresponds to leaking to the simulator that a file has been added or accessed by referring to a pseudonym identifier, and leaking the file contents associated with pseudonyms of active files during a compromise (see [36] for more details). The bookkeeping in the pseudocode takes the form of two tables, A for tracking active files and R for tracking revoked files that are not active, but not permanently deleted.

We define the advantage $\mathsf{Adv}^{\mathsf{ca\text{-}poh}}_{\mathsf{RCS,S,P},\mathcal{A}_{\mathsf{ca}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{ca\text{-}poh}}_{\mathsf{RCS,S,P},\mathcal{A}_{\mathsf{ca}}}(\lambda)] - 1$ and say an RCS scheme is secure with P if for all PPT $\mathcal{A}_{\mathsf{ca}}$ there exists a PPT S such that $\mathsf{Adv}^{\mathsf{ca\text{-}poh}}_{\mathsf{RCS,S,P},\mathcal{A}_{\mathsf{ca}}}(\cdot)$ is negligible.

Burnbox Scheme. We will consider the scheme $\mathsf{BB}[\mathsf{SE}, \mathsf{PKE}]$ given in [36] which is parameterized by a symmetric encryption scheme SE and a public key encryption scheme PKE. For each added file, a random encryption key and random identifier string is sampled. The file is encrypted using SE and stored on the key-value store under the identifier string. A lookup table between file labels to identifier strings and encryption keys is maintained in client state to allow file accesses. To remove access to a file, the corresponding row of the lookup is deleted. However, to support temporary revocation of a file, the file label, identifier string, and encryption key for all files are encrypted under a restoration public key using PKE and stored in a backup table. Restoring access to a file is done by decrypting the rows in the backup table with the restoration secret key and re-adding the relevant information to the lookup table of active files. This means to permanently delete a file, in addition to deleting the row from the lookup table, the relevant row in the backup table is also deleted. The pseudocode for $\mathsf{BB}[\mathsf{SE}, \mathsf{PKE}]$ is given in Figure 23. In this code we have written Add and Access to output the transcripts from their key-value operations and we have omitted any code that does not effect these transcripts. (See [36] for a more complete description of the scheme.)

Theorem E.1 *If* SE *is SIM-AC-CPA secure with* P_1, PKE *is IND-CPA secure with* P_2, *and* BB.idl *is super-logarithmic then* $\mathsf{BB}[\mathsf{SE}, \mathsf{PKE}]$ *is CA-POH secure with respect to* $\mathsf{P}_1 \times \mathsf{P}_2$.

Proof: Let $\mathcal{A}_{\mathsf{ca}}$ be an adversary against the CA-POH security of $\mathsf{BB}[\mathsf{SE}, \mathsf{PKE}]$. We will construct adversary $\mathcal{A}_{\mathsf{cpa}}$ against the SIM-AC-CPA security of SE and $\mathcal{B}_{\mathsf{cpa}}$ against the IND-CPA security of

$\underline{\text{Game } \mathrm{G}^{\mathsf{ca\text{-}poh}}_{\mathsf{RCS},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{ca}}}(\lambda)}$
$i \leftarrow 0$
$(\sigma, k) \leftarrow\!\!\$\ \mathsf{RCS.Init}(1^\lambda)$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$
$\sigma_{\mathsf{S}} \leftarrow\!\!\$\ \mathsf{S.Init}(1^\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}}_{\mathsf{ca}}(1^\lambda)$
Return $b = b'$

$\underline{\textsc{Prim}(x)}$
$y_1 \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$
$y_0 \leftarrow\!\!\$\ \mathsf{S.Prim}(1^\lambda, x : \sigma_{\mathsf{S}})$
Return y_b

$\underline{\textsc{Add}(lbl, m)}$
Require $(A[lbl] = \bot) \wedge (R[lbl] = \bot)$
$i \leftarrow i + 1$
$A[lbl] \leftarrow (i, m)$
$\tau_1 \leftarrow\!\!\$\ \mathsf{RCS.Add}(1^\lambda, lbl, m : \sigma)$
$\tau_0 \leftarrow\!\!\$\ \mathsf{S.Add}(1^\lambda, i, |m| : \sigma_{\mathsf{S}})$
Return τ_b

$\underline{\textsc{Access}(lbl)}$
Require $A[lbl] \neq \bot$
$(i, m) \leftarrow A[lbl]$
$\tau_1 \leftarrow\!\!\$\ \mathsf{RCS.Access}(1^\lambda, lbl : \sigma)$
$\tau_0 \leftarrow\!\!\$\ \mathsf{S.Access}(1^\lambda, i : \sigma_{\mathsf{S}})$
Return τ_b

$\underline{\textsc{Compromise}}$
$\sigma_1 \leftarrow \sigma$
$\sigma_0 \leftarrow\!\!\$\ \mathsf{S.Compromise}(1^\lambda, A : \sigma_{\mathsf{S}})$
Return σ_b

$\underline{\textsc{Delete}(lbl)}$
Require $A[lbl] \neq \bot$
$A[lbl] \leftarrow \bot$
$\mathsf{RCS.Delete}(1^\lambda, lbl : \sigma)$

$\underline{\textsc{Revoke}(lbl)}$
Require $A[lbl] \neq \bot$
$R[lbl] \leftarrow A[lbl]$
$A[lbl] \leftarrow \bot$
$\mathsf{RCS.Revoke}(1^\lambda, lbl : \sigma)$

$\underline{\textsc{Restore}}$
For $lbl \in R$ do
  $A[lbl] \leftarrow R[lbl]$
  $R[lbl] \leftarrow \bot$
$\mathsf{RCS.Restore}(1^\lambda, k : \sigma)$

Figure 22: Game defining compelled access security for revocable cloud storage schemes under the pseudonymous operation history leakage regime [36]. The adversary has access to oracles $\mathcal{O} = \{\textsc{Prim}, \textsc{Add}, \textsc{Access}, \textsc{Delete}, \textsc{Revoke}, \textsc{Restore}, \textsc{Compromise}\}$.

$\underline{\mathsf{BB}[\mathsf{SE},\mathsf{PKE}].\mathsf{Init}(1^\lambda)}$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda)$
$\sigma \leftarrow ([\cdot], [\cdot], ek)$
Return (σ, dk)

$\underline{\mathsf{BB}[\mathsf{SE},\mathsf{PKE}].\mathsf{Add}^{\mathsf{P}_1 \times \mathsf{P}_2}(1^\lambda, lbl, m : (T, B, ek))}$
$id \leftarrow\!\!\$\ \{0,1\}^{\mathsf{BB.idl}(\lambda)}$
$K \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
$T[lbl] \leftarrow (id, K)$
$B[id] \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\mathsf{P}_2}(ek, (1, lbl, K))$
$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}_1}(K, m)$
Return $(\mathrm{P}, (id, c))$

$\underline{\mathsf{BB}[\mathsf{SE},\mathsf{PKE}].\mathsf{Access}^{\mathsf{P}_1 \times \mathsf{P}_2}(1^\lambda, lbl : (T, B, ek))}$
$(id, K) \leftarrow T[lbl]$
Return (G, id)

$\underline{\mathsf{BB}[\mathsf{SE},\mathsf{PKE}].\mathsf{Delete}^{\mathsf{P}_1 \times \mathsf{P}_2}(1^\lambda, lbl : (T, B, ek))}$
$(id, K) \leftarrow T[lbl]$
$B[id] \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\mathsf{P}_2}(ek, 0^{\mathsf{BB.lbll}(\lambda)+\mathsf{SE.kl}(\lambda)+1})$
$T[lbl] \leftarrow \bot$

$\underline{\mathsf{BB}[\mathsf{SE},\mathsf{PKE}].\mathsf{Revoke}^{\mathsf{P}_1 \times \mathsf{P}_2}(1^\lambda, lbl : (T, B, ek))}$
$T[lbl] \leftarrow \bot$

$\underline{\mathsf{BB}[\mathsf{SE},\mathsf{PKE}].\mathsf{Restore}^{\mathsf{P}_1 \times \mathsf{P}_2}(1^\lambda, dk : (T, B, ek))}$
For $id \in B$ do
  $(b_R, lbl, K) \leftarrow \mathsf{PKE.Dec}^{\mathsf{P}_2}(dk, B[id])$
  If $b_R \neq 0$ then $T[lbl] \leftarrow (id, K)$

Figure 23: Burnbox construction of a revocable cloud storage scheme given in [36]. The presentation is modified here to return the key-value operation transcript as relevant for the security proof.

PKE. Then given any SIM-AC-CPA simulator $\mathsf{S}_{\mathsf{cpa}}$, we construct a CA-POH simulator S_{ca} (shown in Figure 24) such that the following holds:

$$\mathsf{Adv}^{\mathsf{ca\text{-}poh}}_{\mathsf{BB}[\mathsf{SE},\mathsf{PKE}],\mathsf{S}_{\mathsf{ca}},\mathsf{P}_1\times\mathsf{P}_2,\mathcal{A}_{\mathsf{ca}}}(\lambda) \leq a_\lambda^2/2^{\mathsf{BB.idl}(\lambda))+1} + \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S}_{\mathsf{cpa}},\mathsf{P}_1,\mathcal{A}_{\mathsf{cpa}}}(\lambda) + \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathsf{P}_2,\mathcal{B}_{\mathsf{cpa}}}(\lambda)\,.$$

Here a_λ is the number of oracle queries that $\mathcal{A}_{\mathsf{ca}}$ makes to $\textsc{Add}$. Our adversary and simulator constructions are PPT, so the theorem holds by letting $\mathsf{S}_{\mathsf{cpa}}$ be the simulator which makes the advantage function for SE negligible. Roughly, S_{ca} simulates the key-value transcript by using

$\mathsf{S}_{\mathsf{cpa}}.\mathsf{Enc}$ to simulate the encrypted files. Later during compromise, when the file contents of active files are leaked, S_{ca} uses $\mathsf{S}_{\mathsf{cpa}}.\mathsf{Exp}$ to populate the client state with keys consistent with the previously chosen ciphertexts. Lastly, S_{ca} stores dummy encryptions of zero strings in the backup table.

$\underline{\mathsf{S}_{\mathsf{ca}}.\mathsf{Init}(1^\lambda)}$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda)$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P}_2.\mathsf{Init}(1^\lambda)$
$\sigma_{\mathsf{S}} \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{cpa}}.\mathsf{Init}(1^\lambda)$
$\mathcal{I} \leftarrow \{0,1\}^{\mathsf{BB.idl}(\lambda)}$
Return $([\cdot], [\cdot], ek, \sigma_{\mathsf{S}}, \sigma_{\mathsf{P}})$

$\underline{\mathsf{S}_{\mathsf{ca}}.\mathsf{Add}(1^\lambda, i, mlen : (T_{\mathsf{S}}, B_{\mathsf{S}}, ek, \sigma_{\mathsf{S}}, \sigma_{\mathsf{P}}))}$
$id \leftarrow\!\!\$\ \mathcal{I}\,; \mathcal{I} \leftarrow \mathcal{I} \setminus \{id\}$
$c \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{cpa}}.\mathsf{Enc}(1^\lambda, i, mlen : \sigma_{\mathsf{S}})$
$T_{\mathsf{S}}[i] \leftarrow (id, c)$
$B_{\mathsf{S}}[id] \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\mathsf{P}_2.\mathsf{Prim}(1^\lambda, \cdot : \sigma_{\mathsf{P}})}(ek, 0^{\mathsf{BB.lbll}(\lambda)+\mathsf{SE.kl}(\lambda)+1})$
Return $(\mathrm{P}, (id, c))$

$\underline{\mathsf{S}_{\mathsf{ca}}.\mathsf{Access}(1^\lambda, i : (T_{\mathsf{S}}, B_{\mathsf{S}}, ek, \sigma_{\mathsf{S}}, \sigma_{\mathsf{P}}))}$
$(id, c) \leftarrow T_{\mathsf{S}}[i]\,;$ Return (G, id)

$\underline{\mathsf{S}_{\mathsf{ca}}.\mathsf{Prim}(1^\lambda, x : (T_{\mathsf{S}}, B_{\mathsf{S}}, ek, \sigma_{\mathsf{S}}, \sigma_{\mathsf{P}}))}$
$(d, x) \leftarrow x$
If $d = 1$ then $y \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{cpa}}.\mathsf{Prim}(1^\lambda, x : \sigma_{\mathsf{S}})$
Else $y \leftarrow\!\!\$\ \mathsf{P}_2.\mathsf{Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$
Return y

$\underline{\mathsf{S}_{\mathsf{ca}}.\mathsf{Compromise}(1^\lambda, A : (T_{\mathsf{S}}, B_{\mathsf{S}}, ek, \sigma_{\mathsf{S}}, \sigma_{\mathsf{P}}))}$
$T \leftarrow [\cdot]$
For $lbl \in A$ do
  $(i, m) \leftarrow A[lbl]$
  $(id, c) \leftarrow T_{\mathsf{S}}[i]$
  $K \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{cpa}}.\mathsf{Exp}(1^\lambda, i, [m], [c] : \sigma_{\mathsf{S}})$
  $T[lbl] \leftarrow (id, K)$
Return $(T, B_{\mathsf{S}}[\cdot], ek)$

Figure 24: Simulator S_{ca} used in proof of Theorem E.1.

We bound the advantage of $\mathcal{A}_{\mathsf{ca}}$ by bounding the advantage of each of a series of game hops. We define the first game G_{-1} to be identical to the real world, and the last game G_6 is identical to the ideal world (except it returns true when $b = 1$). Hence $\mathsf{Adv}^{\mathsf{ca\text{-}poh}}_{\mathsf{BB}[\mathsf{SE},\mathsf{PKE}],\mathsf{S}_{\mathsf{ca}},\mathsf{P}_1\times\mathsf{P}_2,\mathcal{A}_{\mathsf{ca}}}(\lambda) = \Pr[\mathrm{G}_{-1}] - \Pr[\mathrm{G}_6]$ by standard conditional probability computation.

Games G_0, $\boxed{\mathrm{G}_1}$
$i \leftarrow 0$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda)$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$
$\mathcal{I} \leftarrow \{0,1\}^{\mathsf{BB.idl}(\lambda)}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}}_{\mathsf{ca}}(1^\lambda)$
Return $b' = 1$

PRIM(x)
$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$
Return y

ADD(lbl, m)
Require $(A[lbl] = \bot) \wedge (R[lbl] = \bot)$
$i \leftarrow i + 1$
$A[lbl] \leftarrow (i, m)$
$id \leftarrow\!\!\$\ \mathcal{I}\,; \mathcal{I} \leftarrow \mathcal{I} \setminus \{id\}$
$K \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
$T[lbl] \leftarrow (id, K)$
$B[id] \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\mathsf{P}_2}(ek, (1, lbl, K))$
$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}_1}(K, m)$
Return $(\mathrm{P}, (id, c))$

ACCESS(lbl)
Require $A[lbl] \neq \bot$
$(i, m) \leftarrow A[lbl]$
$(id, K) \leftarrow T[lbl]$
Return (G, id)

COMPROMISE
Return (T, B, ek)

DELETE(lbl)
Require $A[lbl] \neq \bot$
$A[lbl] \leftarrow \bot$
$(id, K) \leftarrow T[lbl]$
$B[id] \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\mathsf{P}_2}(ek, 0^{\mathsf{BB.lbll}(\lambda)+\mathsf{SE.kl}(\lambda)+1})$
$T[lbl] \leftarrow \bot$

REVOKE(lbl)
Require $A[lbl] \neq \bot$
$R[lbl] \leftarrow A[lbl]\,; \boxed{R'[lbl] \leftarrow T[lbl]}$
$A[lbl] \leftarrow \bot\,; T[lbl] \leftarrow \bot$

RESTORE
For $lbl \in R$ do
  $A[lbl] \leftarrow R[lbl]\,; \boxed{T[lbl] \leftarrow R'[lbl]}$
  $R[lbl] \leftarrow \bot\,; \boxed{R'[lbl] \leftarrow \bot}$
For $id \in B$ do
  $(b_R, lbl, K) \leftarrow \mathsf{PKE.Dec}^{\mathsf{P}_2}(dk, B[id])$
  If $b_R \neq 0$ then $T[lbl] \leftarrow (id, K)$

Figure 25: Games G_0 and G_1 used in the proof of Theorem E.1. Highlighted code is only included in G_0 and boxed code is only included in G_1.

The first set of games to consider is G_0 and G_1 shown in Figure 25. Game G_0 is identical to the

real world with the code of $\mathsf{BB}[\mathsf{SE}, \mathsf{PKE}]$ except that a set $\mathcal{I}$ is used in ADD so that id is sampled without replacement. Thus $\Pr[\mathrm{G}_{-1}(\lambda)] - \Pr[\mathrm{G}_0(\lambda)] \leq \binom{a_\lambda}{2}/2^{\mathsf{BB.idl}(\lambda))}$. Game G_1 dispenses with $\mathsf{PKE.Dec}$, instead introducing a table R' to store the restoration plaintext information of revoked files. This does not change the functionality of the game as only legitimate ciphertexts generated in ADD are ever decrypted and we assume a correct PKE scheme. Thus, $\Pr[\mathrm{G}_0(\lambda)] = \Pr[\mathrm{G}_1(\lambda)]$.

Games G_1, $\boxed{\mathrm{G}_2}$, G_3

$i \leftarrow 0$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda)$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$
$\mathcal{I} \leftarrow \{0,1\}^{\mathsf{BB.idl}(\lambda)}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}}_{\mathsf{ca}}(1^\lambda)$
Return $b' = 1$

PRIM(x)
$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$
Return y

ACCESS(lbl)
Require $A[lbl] \neq \bot$
$(i, m) \leftarrow A[lbl]$
$(id, K) \leftarrow T[lbl]$
Return (G, id)

ADD(lbl, m)
Require $(A[lbl] = \bot) \wedge (R[lbl] = \bot)$
$i \leftarrow i + 1$
$A[lbl] \leftarrow (i, m)$
$id \leftarrow\!\!\$\ \mathcal{I}\,; \mathcal{I} \leftarrow \mathcal{I} \setminus \{id\}$
$K \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
$T[lbl] \leftarrow (id, K)$
$B[id] \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\mathsf{P}_2}(ek, (1, lbl, K))$
$\boxed{B[id] \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\mathsf{P}_2}(ek, 0^{\mathsf{BB.lbll}(\lambda)+\mathsf{SE.kl}(\lambda)+1})}$
$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}_1}(K, m)$
Return $(\mathrm{P}, (id, c))$

DELETE(lbl)
Require $A[lbl] \neq \bot$
$A[lbl] \leftarrow \bot$
$\boxed{(id, K) \leftarrow T[lbl]}$
$\boxed{B[id] \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\mathsf{P}_2}(ek, 0^{\mathsf{BB.lbll}(\lambda)+\mathsf{SE.kl}(\lambda)+1})}$
$T[lbl] \leftarrow \bot$

REVOKE(lbl)
Require $A[lbl] \neq \bot$
$R[lbl] \leftarrow A[lbl]$
$R'[lbl] \leftarrow T[lbl]$
$A[lbl] \leftarrow \bot$
$T[lbl] \leftarrow \bot$

RESTORE
For $lbl \in R$ do
  $A[lbl] \leftarrow R[lbl]$
  $T[lbl] \leftarrow R'[lbl]$
  $R[lbl] \leftarrow \bot$
  $R'[lbl] \leftarrow \bot$

COMPROMISE
Return (T, B, ek)

Figure 26: Games used for proof of Theorem E.1. Highlighted code is included in G_1, boxed code is included in G_2, green highlighted code is included in G_3. Other code is common to all games.

The second set of game hops from G_1 to G_3 shown in Figure 26 deals with the use of $\mathsf{PKE.Enc}$. Game G_2 replaces the call of $\mathsf{PKE.Enc}$ in ADD to encrypt a dummy plaintext of all-zeros instead of the restoration plaintext to populate the backup table. Game G_3 removes the rewriting of the backup table in DELETE so the only one call of $\mathsf{PKE.Enc}$ occurs per file.

We apply the security of PKE to bound the advantage of the first hop from G_1 to G_2. More precisely, the IND-CPA adversary $\mathcal{B}_{\mathsf{cpa}}$ given in Figure 27 simulates $\mathcal{A}_{\mathsf{ca}}$'s environment as from G_1 except it replaces calls to $\mathsf{PKE.Enc}$ with a query to its left-or-right LR oracle. In ADD it sets m_1 to the label and key of the restoration plaintext as in G_1 and m_2 as the dummy all-zero string as in G_2. In DELETE, both messages are set to the dummy all-zero string. Thus, the advantage of $\mathcal{B}_{\mathsf{cpa}}$ is exactly the same as the advantage in distinguishing between G_1 and G_2: $\Pr[\mathrm{G}_1(\lambda)] - \Pr[\mathrm{G}_2(\lambda)] = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathsf{P}_2,\mathcal{B}_{\mathsf{cpa}}}(\lambda)$.

Now in game G_2, a public key encryption of a dummy all-zero string is written to the backup table in both ADD and DELETE. Game G_3 removes the update to the backup table in DELETE. The only way for the adversary to observe the contents of the backup table is through COMPROMISE. Since COMPROMISE can only be called once, not replacing a row with a fresh encryption of the same all-zero string does not change the adversary's view; they still only see an encryption of an all-zero string. This gives, $\Pr[\mathrm{G}_2(\lambda)] = \Pr[\mathrm{G}_3(\lambda)]$.

The next set of games hops from G_3 to G_5 (shown in Figure 28) switches from updating the state table T on every oracle call to lazily constructing the state table only when needed during COMPROMISE. Game G_4 introduces T_{S} to store the information needed to populate T in

Adversary $\mathcal{B}_{\mathsf{cpa}}^{\text{LR},\text{PRIM}}(1^\lambda)$
$i \leftarrow 0$
$\sigma'_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P}_1.\mathsf{Init}(1^\lambda)$
$\mathcal{I} \leftarrow \{0,1\}^{\mathsf{BB.idl}(\lambda)}$
$b' \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{ca}}^{\text{ADDSIM},\text{DELETESIM},\text{PRIMSIM},\mathcal{O}}(1^\lambda)$
Return b'

ADDSIM(lbl, m)
Require $(A[lbl] = \bot) \wedge (R[lbl] = \bot)$
$i \leftarrow i + 1$
$A[lbl] \leftarrow (i, m)$
$id \leftarrow\!\!\$\ \mathcal{I}\,; \mathcal{I} \leftarrow \mathcal{I} \setminus \{id\}$
$K \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
$T[lbl] \leftarrow (id, K)$
$m_1 \leftarrow (1, lbl, K)$
$m_0 \leftarrow 0^{\mathsf{BB.lbll}(\lambda)+\mathsf{SE.kl}(\lambda)+1}$
$B[id] \leftarrow \text{LR}(m_0, m_1)$
$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}_1}(K, m)$
Return $(\mathrm{P}, (id, c))$

PRIMSIM(x)
$(d, x) \leftarrow x$
If $d = 1$ then
  $y \leftarrow\!\!\$\ \mathsf{P}_1.\mathsf{Prim}(1^\lambda, x : \sigma'_{\mathsf{P}})$
Else $y \leftarrow\!\!\$\ \text{PRIM}(x)$
Return y

DELETESIM(lbl)
Require $A[lbl] \neq \bot$
$A[lbl] \leftarrow \bot$
$(id, K) \leftarrow T[lbl]$
$m \leftarrow 0^{\mathsf{BB.lbll}(\lambda)+\mathsf{SE.kl}(\lambda)+1}$
$B[id] \leftarrow \text{LR}(m, m)$
$T[lbl] \leftarrow \bot$

Figure 27: Adversary $\mathcal{B}_{\mathsf{cpa}}$ for proof of Theorem E.1. When $\mathcal{B}_{\mathsf{cpa}}$ runs $\mathcal{A}_{\mathsf{ca}}$, oracle $\mathcal{O}$ represents access to oracles $\{\text{ACCESS}, \text{REVOKE}, \text{RESTORE}, \text{COMPROMISE}\}$ defined as by G_1 in Fig. 26.

Games G_3, $\boxed{\mathrm{G}_4}$, G_5
$i \leftarrow 0$
$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda)$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$
$\mathcal{I} \leftarrow \{0,1\}^{\mathsf{BB.idl}(\lambda)}$
$b' \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{ca}}^{\mathcal{O}}(1^\lambda)$
Return $b' = 1$

PRIM(x)
$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$
Return y

ACCESS(lbl)
Require $A[lbl] \neq \bot$
$(i, m) \leftarrow A[lbl]$
$(id, K) \leftarrow T[lbl]$
$\boxed{(id, c, K) \leftarrow T_{\mathsf{S}}[i]}$
Return (G, id)

ADD(lbl, m)
Require $(A[lbl] = \bot) \wedge (R[lbl] = \bot)$
$i \leftarrow i + 1$
$A[lbl] \leftarrow (i, m)$
$id \leftarrow\!\!\$\ \mathcal{I}\,; \mathcal{I} \leftarrow \mathcal{I} \setminus \{id\}$
$K \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
$\boxed{T[lbl] \leftarrow (id, K)}$
$B[id] \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\mathsf{P}_2}(ek, 0^{\mathsf{BB.lbll}(\lambda)+\mathsf{SE.kl}(\lambda)+1})$
$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}_1}(K, m)$
$\boxed{T_{\mathsf{S}}[i] \leftarrow (id, c, K)}$
Return $(\mathrm{P}, (id, c))$

COMPROMISE
For $lbl \in A$ do
  $(i, m) \leftarrow A[lbl]$
  $(id, c, K) \leftarrow T_{\mathsf{S}}[i]$
  $T[lbl] \leftarrow (id, K)$
Return (T, B, ek)

DELETE(lbl)
Require $A[lbl] \neq \bot$
$A[lbl] \leftarrow \bot$
$\boxed{T[lbl] \leftarrow \bot}$

REVOKE(lbl)
Require $A[lbl] \neq \bot$
$R[lbl] \leftarrow A[lbl]$
$\boxed{R'[lbl] \leftarrow T[lbl]}$
$A[lbl] \leftarrow \bot$
$\boxed{T[lbl] \leftarrow \bot}$

RESTORE
For $lbl \in R$ do
  $A[lbl] \leftarrow R[lbl]$
  $\boxed{T[lbl] \leftarrow R'[lbl]}$
  $R[lbl] \leftarrow \bot$
  $\boxed{R'[lbl] \leftarrow \bot}$

Figure 28: Games used for proof of Theorem E.1. Highlighted code is included in G_3, boxed code is included in G_4, green highlighted code is included in G_5. Other code is common to all games.

COMPROMISE and respond to ACCESS queries. Game G_5 removes updates of T from ADD, ACCESS, DELETE, REVOKE, and RESTORE, and adds creation of T in COMPROMISE. The files that should be present in the state table T on compromise are the currently active files in the system, which are tracked by A. One can observe in Figure 28, the removed updates to T were redundant alongside updates to A. Then in COMPROMISE, G_5 populates a row in T with the contents from T_{S} if a corresponding row exists in A. The change from ongoing to lazy evaluation of T does not change the view of the adversary, so $\Pr[\mathrm{G}_3(\lambda)] = \Pr[\mathrm{G}_4(\lambda)] = \Pr[\mathrm{G}_5(\lambda)]$.

The last set of game hops from G_5 to G_6 (shown in Fig. 29) move from sampling keys and encrypting file contents in ADD to simulating the encrypted files using $\mathsf{S}_{\mathsf{cpa}}.\mathsf{Enc}$. When the file contents of active files are learned in COMPROMISE, $\mathsf{S}_{\mathsf{cpa}}.\mathsf{Exp}$ is used to retrieve a consistent key to populate

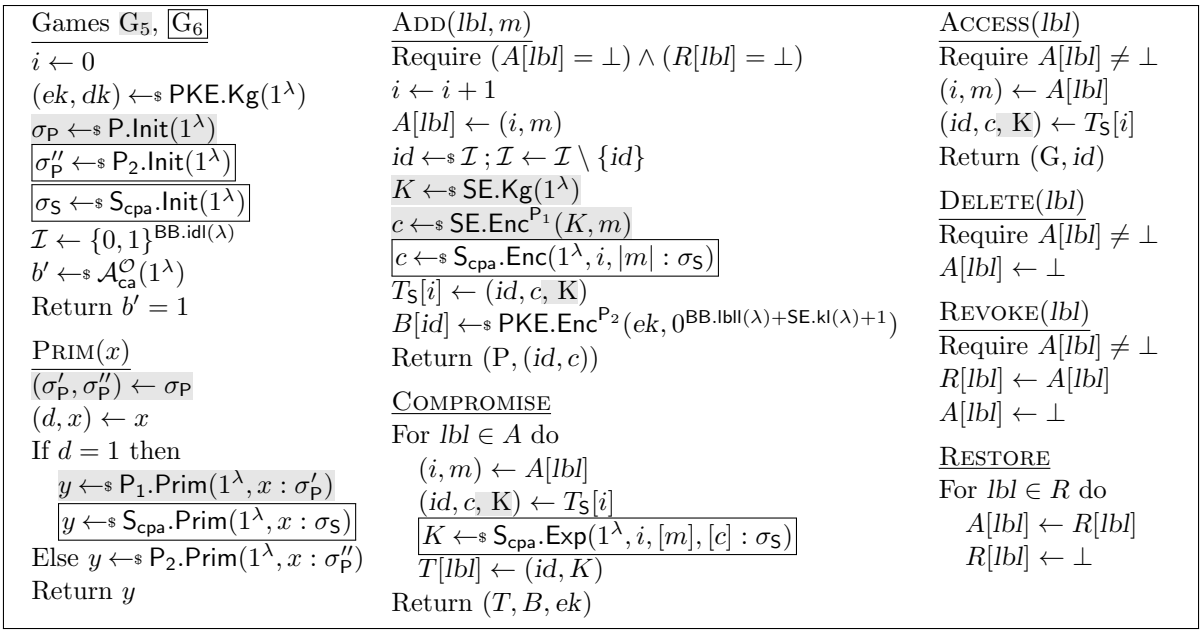

Figure 29: Games introducing $\mathsf{S}_{\mathsf{cpa}}$ used in the proof of Theorem E.1. Highlighted code is only included in G_5 and boxed codes is only included in G_6.

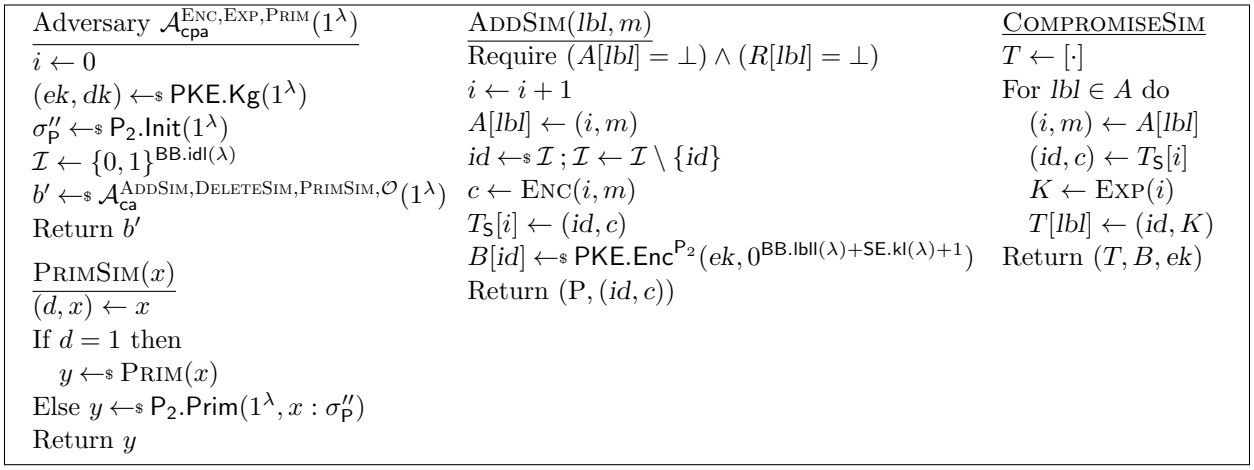

Figure 30: Adversary $\mathcal{A}_{\mathsf{cpa}}$ used in the proof of Theorem E.1. When running adversary $\mathcal{A}_{\mathsf{ca}}$ the oracle $\mathcal{O}$ denotes access to oracles $\{\mathrm{ACCESS}, \mathrm{REVOKE}, \mathrm{RESTORE}, \mathrm{COMPROMISE}\}$ as defined G_6.

the state table T.

We apply the SIM-AC-CPA security of SE to bound the advantage of distinguishing between G_5 and G_6. More precisely, adversary $\mathcal{A}_{\mathsf{cpa}}$ given in Fig. 30 simulates $\mathcal{A}_{\mathsf{ca}}$'s environment as from G_5 except it replaces calls to $\mathsf{SE.Kg}$ and $\mathsf{SE.Enc}$ in ADD with queries to its ENC oracle on index i, and replaces the symmetric key lookup from T_S in COMPROMISE with a query to EXP on index i of the active file. The message set and ciphertext set passed into EXP consist of the single ciphertext created during ADD and the corresponding message leaked during compromise; only one message is ever passed to ENC for any given i. Thus, the SIM-AC-CPA advantage of $\mathcal{A}_{\mathsf{cpa}}$ in distinguishing between SE and $\mathsf{S}_{\mathsf{cpa}}$ is exactly the same advantage as distinguishing between G_5

and G_6: $\Pr[G_5(\lambda)] - \Pr[G_6(\lambda)] = \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S}_{\mathsf{cpa}},\mathsf{P}_1,\mathcal{A}_{\mathsf{cpa}}}(\lambda)$.

The originally stated advantage inequality follows from standard probability calculation. ∎

F Bugs In Prior Proofs

We have identified fifteen papers where our definitional framework can be applied [2, 3, 11, 14, 15, 19, 23, 24, 27–31, 36, 37] to modularize the ideal model analysis required to prove adaptive security. The majority of these examples omitted a full analysis of the ideal model programming (likely considering it standard and conceptually straightforward, yet detail intensive). We have identified bugs stemming from subtle details in all of the examples that attempted to provide the full details of the ideal model analysis [11, 30, 36, 37] and in some which only sketched the details [14, 29]. We emphasize that once identified, these proof bugs are straightforward to fix (if tedious to do so in thorough detail). This difficulty in writing a thorough analysis of the ideal model programming shows the benefit of our definitional framework which allows this analysis to be done once for basic primitives and then "passed along" to more complicated protocols. We emphasize that the bugs we identified have no bearing on the achieved security of the protocols; they indicate only that their current proofs are technically not valid.

A core component of the sorts of random oracle proofs we are interested in (proofs for other ideal models are similar) can be thought of as splitting the random oracle into the real random oracle and a separate random oracle for each "user". Here a user might be a literal human user of a system or more abstractly any sort of entity that is associated with its own key used to seed the random oracle. Bounding an adversary's attack success then boils down to enumerating the possibly ways that the adversary can cause inconsistencies in the separated random oracles that would not be possible when a single random oracle is being shared. The proof writer specifies these "bad" events and then bounds the probability that each of them occur.

The errors we have found tend to be of the form of failing to identify all of these "bad" events and consequently arguing consistency of tables which are not kept consistent. We will now walk through the specific examples we have identified.

The proof of Theorem 1 in [30] considers a sequence of games G_0 through G_3 (not all of which are give explicitly). In each transition from one game to the next they are replacing honestly generated output of a random oracle with (somewhat underspecified) programming. These games are claimed to be equivalent, which cannot be the case since an adversary which happens to make a particular random oracle query before it is programmed could detect the programming.

The proofs of Theorem 1 in [11] and Theorem 1 in [37] are highly similar. The latter of these considers a sequence of games $G_{1,0}$ through $G_{1,4}$. The core random oracle steps in the analysis are the transitions from $G_{1,1}$ to $G_{1,2}$ and from $G_{1,2}$ to $G_{1,3}$. For these steps they cite [11] as having shown how to bound the difference between these games by the advantage of a one-way adversary. The proof in [11] does provide such a claim to transition from their G_1 to G_2 and from G_2 to G_3. Let us focus on the first of these, for which they first introduce an intermediate $\widetilde{G_2}$ which they claim to be equivalent to G_1. This claim is false. In G_1, a random oracle is being honestly executed. In $\widetilde{G_2}$ the random oracle is distributed between table $\mathtt{H}_1$ and a table $\mathtt{UT}[w, \cdot]$ for each keyword w. The entry $\mathtt{UT}[w, c]$ corresponds to the table entry $\mathtt{H}_1(K_w, ST_c)$. The particular meaning of the variables K_w and ST_c is not important beyond noting that *they can repeat across different choices of* w. Game $\widetilde{G_2}$ has code for maintaining consistency between $\mathtt{H}_1$ and each individual $\mathtt{UT}[w, \cdot]$, but not for maintaining consistency among $\mathtt{UT}[w, \cdot]$ for differing w. Hence the claimed of equivalence is false.

Analogous issues can be found in [29] and [14]. Neither of these provide a thorough game-

based argument bounding the adversary's probability of detecting inconsistency in their view. Instead each briefly describes how the adversary could notice inconsistencies in the random oracle programming and why these events are unlikely. However both only identify the possibility of an adversary detecting a simulation if it makes a random oracle query using a key which is unknown to it. For example, the latter [14] says, "The only defects in the simulation occur when an adversary manages to query the random oracle with a key before it is revealed, which can be shown to happen with negligible in λ probability." in the proof sketch for their Theorem 7. In the proof of their Theorem 4.1, the former [29] says "On the other hand, it is possible that the adversary $\mathcal{A}$ has already made random oracles queries H_2 or H_3 with inputs $(\mathsf{ukey}^{(w)}, \mathsf{ucnt}^{(w)})$ and got responses $\mathsf{out}_2 = H_2(\mathsf{ukey}^{(w)}, \mathsf{ucnt}^{(w)})$ or $\mathsf{out}_3 = H_3(\mathsf{ukey}^{(w)}, \mathsf{ucnt}^{(w)})$. In this case, if $\mathsf{out}_2 \neq \mathsf{label}^{(2)}$ or $\mathsf{out}_3 \neq \mathsf{mask}$, the simulation fails." Both are failing to notice the possibility of inconsistencies without the adversary making random oracle queries that would be caused by there happening to be collisions in the keys that are sampled.

The bug in a proof of Tyagi, et al. [36] stems from a unique attribute of the ideal encryption model. The output of its decryption procedure can *depend on the order that encryption calls were made* if the the ciphertexts generated happen to collide. In particular, the issue arises in the proof of their Theorem 1 when transitioning between games G_5 and G_6. In game G_5, the IEM is being executed honestly. In game G_6, the IEM decryption table is stored as two separate tables D' and D. Whenever an inconsistency is found D' is used to overwrite D, which misses the possibility that D may have been set later than D' and thus should *not* be overwritten. We overcome this technical hurdle in our own IEM proof (Appendix I.1) by introducing an auxiliary table Q which tracks when IEM queries were made. We believe this auxiliary table idea may be useful for any future proofs written in the IEM.

G Omitted Proofs for Classic Results

G.1 Proof of Theorem 5.1 (SIM-CPA + INT-CTXT ⇒ SIM-CCA)

We start with a proof of Theorem 5.1 from Section 5. This proof establishes that SIM-AC-CPA security and INT-CTXT security imply SIM-AC-CCA security.

Proof (of Theorem 5.1): Let $\mathcal{A}_{\mathsf{cca}}$ be a PPT adversary against the SIM-AC-CCA security of SE. Then we construct adversaries $\mathcal{A}_{\mathsf{ctxt}}$ and $\mathcal{A}_{\mathsf{cpa}}$ against the INT-CTXT and SIM-AC-CPA security of SE. Then, given a PPT SIM-AC-CPA simulator $\mathsf{S}_{\mathsf{cpa}}$, we will construct a SIM-AC-CCA simulator $\mathsf{S}_{\mathsf{cca}}$ such that the following holds,

$$\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S}_{\mathsf{cca}},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{ctxt}}}(\lambda) + \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S}_{\mathsf{cpa}},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda).$$

It will be clear from examination of the code that the new algorithms we introduce are also PPT. Thus the theorem follows by letting $\mathsf{S}_{\mathsf{cpa}}$ be the simulator guaranteed to exist for which $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S}_{\mathsf{cpa}},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}$ is negligible.

The proof considers the games G_0, G_1, G_2, and G_3. The inequality above follows from simple calculations based on the following claims which we will justify.

1. $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S}_{\mathsf{cca}},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda) = \Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_3(\lambda)]$
2. $\Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_1(\lambda)] = \mathsf{Adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{SE},\mathsf{P},\mathcal{A}_{\mathsf{ctxt}}}(\lambda)$
3. $\Pr[\mathrm{G}_1(\lambda)] - \Pr[\mathrm{G}_2(\lambda)] = 0$
4. $\Pr[\mathrm{G}_2(\lambda)] - \Pr[\mathrm{G}_3(\lambda)] = \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S}_{\mathsf{cpa}},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda)$

Claim 2. We start with the second claim. The game G_0 and G_1 are shown in Fig. 31. Highlighted code is only included in G_0. Boxed code is only included in game G_1. Game G_0 is identical to $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cca}}$ when $b = 1$ and he two games differ only in DEC when the user has not been exposed (i.e. $u \notin X$). Then G_0 returns the actual decryption of the ciphertext c and G_1 returns $\perp$.

Games $\mathrm{G}_0(\lambda)$,$\boxed{\mathrm{G}_1(\lambda)}$	$\underline{\textsc{Enc}(u, m)}$	$\underline{\textsc{Dec}(u, c)}$
For $u \in \{0,1\}^*$ do	Require $m \in \mathsf{SE.M}(\lambda)$	Require $c \notin C_u$
$\quad K_u \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, K_u, m)$	If $u \in X$ do
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$	$C_u.\mathsf{add}(c)$	$\quad m \leftarrow \mathsf{SE.Dec}^{\mathsf{P}}(1^\lambda, K_u, c)$
$b' \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{cca}}^{\textsc{Enc},\textsc{Dec},\textsc{Exp},\textsc{Prim}}(1^\lambda)$	Return c	Else
Return $(b' = 1)$	$\underline{\textsc{Exp}(u)}$	$\quad m \leftarrow \mathsf{SE.Dec}^{\mathsf{P}}(1^\lambda, K_u, c)$
$\underline{\textsc{Prim}(x)}$	$K \leftarrow K_u$	$\quad \boxed{m \leftarrow \perp}$
$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$	$X.\mathsf{add}(u)$	Return m
Return y	Return K	

Figure 31: Games G_0 and G_1 used in proof of Theorem 5.1. Highlighted code is only included in G_0. Boxed code is only included in game G_1.

Thus, distinguishing between these two games very naturally reduces to breaking the INT-CTXT security of SE. The adversary $\mathcal{A}_{\mathsf{ctxt}}$ establishing this is shown in Fig. 32. It essentially just gives $\mathcal{A}_{\mathsf{cca}}$ access to its own oracles and then returns whatever bit $\mathcal{A}_{\mathsf{cca}}$ does. Adversary $\mathcal{A}_{\mathsf{cca}}$ expects its decryption oracle to return the correct decryption whenever $u \in X$, while the oracle in $\mathrm{G}^{\mathsf{int\text{-}ctxt}}$ would return $\perp$ in this case (due to its "Require" statement). So $\mathcal{A}_{\mathsf{ctxt}}$ keeps track of C_u, K_u, and X when ENC and EXP queries are made so that it can properly simulate this case. It uses its own DEC oracle to simulate when $u \notin X$. Thus it properly simulates either G_0 or G_1 depending on the bit underlying $\mathrm{G}^{\mathsf{int\text{-}ctxt}}$. From standard conditional probability analysis we then get claim 2.

$\underline{\text{Adversary } \mathcal{A}_{\mathsf{ctxt}}^{\textsc{Enc},\textsc{Dec},\textsc{Exp},\textsc{Prim}}(1^\lambda)}$	$\underline{\textsc{EncSim}(u, m)}$	$\underline{\textsc{ExpSim}(u)}$	$\underline{\textsc{DecSim}(u, c)}$
$b' \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{cca}}^{\textsc{EncSim},\textsc{DecSim},\textsc{ExpSim},\textsc{Prim}}(1^\lambda)$	$c \leftarrow \textsc{Enc}(u, m)$	$K_u \leftarrow \textsc{Exp}(u)$	Require $c \notin C_u$
Return b'	$C_u.\mathsf{add}(c)$	$X.\mathsf{add}(u)$	If $u \in X$ do
	Return c	Return K_u	$\quad m \leftarrow \mathsf{SE.Dec}^{\textsc{Prim}}(1^\lambda, K_u, m)$
			Else
			$\quad m \leftarrow \textsc{Dec}(u, c)$
			Return m

Figure 32: Adversary $\mathcal{A}_{\mathsf{ctxt}}$ used in proof of Theorem 5.1.

Claim 3. For the third claim, consider game G_2 shown in Fig. 33. It includes the highlighted code, but does not include the boxed code. It was obtained by making some minor modifications to G_1 which do not change its behavior. So we immediately get claim 3.

Claim 4. For the fourth claim, consider game G_3 – also shown in Fig. 33. It includes the boxed code, but not the highlighted code. Is uses the simulator $\mathsf{S}_{\mathsf{cpa}}$ and differs from G_2 in that this simulator is used to choose the output of PRIM, ENC, and EXP. They share the same DEC.

Thus, distinguishing between these two games reduces to the IND-CPA security of SE. The adversary $\mathcal{A}_{\mathsf{cpa}}$ used for this is shown in Fig. 34. It runs adversary $\mathcal{A}_{\mathsf{cca}}$, simulating encryption, exposure, and ideal primitive queries by forwarding them on to its corresponding oracles. While doing so it stores the variables C_u, K_u, and X. Using these it simply runs the code of DEC locally and returns

Games $\mathrm{G}_2(\lambda)$,$\boxed{\mathrm{G}_3(\lambda)}$

For $u \in \{0,1\}^*$ do
  $K_u \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$
$\sigma \leftarrow\!\!\$\ \mathsf{S_{cpa}.Init}(1^\lambda)$
$b' \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{cca}}^{\textsc{Enc},\textsc{Dec},\textsc{Exp},\textsc{Prim}}(1^\lambda)$
Return $(b' = 1)$

$\textsc{Prim}(x)$
$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$
$\boxed{y \leftarrow\!\!\$\ \mathsf{S_{cpa}.Prim}(1^\lambda, x : \sigma)}$
Return y

$\textsc{Enc}(u, m)$
Require $m \in \mathsf{SE.M}(\lambda)$
If $u \notin X$ then $\ell \leftarrow |m|$ else $\ell \leftarrow m$
$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, K_u, m)$
$\boxed{c \leftarrow\!\!\$\ \mathsf{S_{cpa}.Enc}(1^\lambda, u, \ell : \sigma)}$
$M_u.\mathsf{add}(m)$; $C_u.\mathsf{add}(c)$
Return c

$\textsc{Exp}(u)$
$K \leftarrow K_u$
$\boxed{K \leftarrow\!\!\$\ \mathsf{S_{cpa}.Exp}(1^\lambda, u, M_u, C_u : \sigma)}$
$X.\mathsf{add}(u)$
Return K

$\textsc{Dec}(u, c)$
Require $c \notin C_u$
If $u \in X$ do $m \leftarrow \mathsf{SE.Dec}^{\textsc{Prim}}(1^\lambda, K_u, c)$
Else $m \leftarrow \bot$
Return m

Figure 33: Games G_2 and G_3 used in proof of Theorem 5.1. Highlighted code is only included in G_2. Boxed code is only included in game G_3.

Adversary $\mathcal{A}_{\mathsf{cpa}}^{\textsc{Enc},\textsc{Exp},\textsc{Prim}}(1^\lambda)$
$b' \leftarrow\!\!\$\ \mathcal{A}_{\mathsf{cca}}^{\textsc{EncSim},\textsc{DecSim},\textsc{ExpSim},\textsc{Prim}}(1^\lambda)$
Return b'

$\textsc{EncSim}(u, m)$
$c \leftarrow \textsc{Enc}(u, m)$
$C_u.\mathsf{add}(c)$
Return c

$\textsc{ExpSim}(u)$
$K_u \leftarrow \textsc{Exp}(u)$
$X.\mathsf{add}(u)$
Return K_u

$\textsc{DecSim}(u, c)$
Require $c \notin C_u$
If $u \in X$ do
  $m \leftarrow \mathsf{SE.Dec}^{\textsc{Prim}}(1^\lambda, K_u, c)$
Else $m \leftarrow \bot$
Return m

Figure 34: Adversary $\mathcal{A}_{\mathsf{cpa}}$ used in proof of Theorem 5.1.

the result to $\mathcal{A}_{\mathsf{cca}}$. Thus is properly simulates either G_2 or G_3 depending on the bit underlying $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cpa}}$. From standard conditional probability analysis we then get claim 4.

$\mathsf{S_{cca}.Init}(1^\lambda)$
$\sigma \leftarrow\!\!\$\ \mathsf{S_{cpa}.Init}(1^\lambda)$
Return $(\sigma, [\cdot])$

$\mathsf{S_{cca}.Prim}(1^\lambda, x : (\sigma, K_*))$
$y \leftarrow\!\!\$\ \mathsf{S_{cpa}.Prim}(1^\lambda, x : \sigma)$
Return y

$\mathsf{S_{cca}.Enc}(1^\lambda, u, \ell : (\sigma, K_*))$
$c \leftarrow\!\!\$\ \mathsf{S_{cpa}.Enc}(1^\lambda, u, \ell : \sigma)$
Return c

$\mathsf{S_{cca}.Exp}(1^\lambda, u, M_u, C_u : (\sigma, K_*))$
$K_u \leftarrow\!\!\$\ \mathsf{S_{cpa}.Exp}(1^\lambda, u, M_u, C_u : \sigma)$
Return K_u

$\mathsf{S_{cca}.Dec}(1^\lambda, u, c : (\sigma, K_*))$
If $K_u \neq \bot$ do
  $m \leftarrow \mathsf{SE.Dec}^{\mathsf{S_{cpa}.Prim}(1^\lambda, \cdot : \sigma)}(1^\lambda, K_u, c)$
Else $m \leftarrow \bot$
Return m

Figure 35: Simulator $\mathsf{S_{cca}}$ used in proof of Theorem 5.1.

Claim 1. Finally we proceed to justifying the first claim. Consider the simulator $\mathsf{S_{cca}}$ defined in Fig. 35. For simulating encryption, exposure, and ideal primitive queries it just runs $\mathsf{S_{cpa}}$. While doing so it remembers each K_u it returned to exposures. It uses this to mirror the code of $\textsc{Dec}$ in game G_3. The oracle access to $\textsc{Prim}$ given to $\mathsf{SE.Dec}$ has been replaced with oracle access to $\mathsf{S_{cpa}.Prim}$ and the check $u \in X$ has been replaced with checking if $K_u \neq \bot$ holds.

We can then verify that the view of $\mathcal{A}_{\mathsf{cca}}$ in G_3 is identical to its view in the ideal world of $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S_{cca}},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}$. We previously noted that game G_0 is identical to the real world of $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S_{cca}},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}$. These observations give us the following equalities.

$$\Pr[\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S_{cca}},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda) | b = 1] = \Pr[\mathrm{G}_0(\lambda)]$$

$$\Pr[\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S}_{\mathsf{cca}},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda)|b=0] = 1 - \Pr[\mathrm{G}_3(\lambda)]$$

The first claim then follows from standard probability calculations. ∎

G.2 Proof Sketch for Theorem 5.2 (Encrypt-then-MAC is SIM-AC-CCA)

Proof (Sketch): We provide a sketch of the proof. First, from Bellare and Namprempre [9, Theorem 4.4] we have that $\mathsf{EtM}[\mathsf{SE},\mathsf{F}]$ is INT-CTXT secure. Hence, by Theorem 5.1, we need only show that $\mathsf{EtM}[\mathsf{SE},\mathsf{F}]$ is SIM-AC-CPA secure. Let S_{SE} be a SIM-AC-CPA simulator for SE. Then we would construct a simulator $\mathsf{S}_{\mathsf{cpa}}$ as follows. Note that $\mathsf{S}_{\mathsf{cpa}}$ is PPT if S_{SE} and P_2 are PPT.

$\underline{\mathsf{S}_{\mathsf{cpa}}.\mathsf{Init}(1^\lambda)}$	$\underline{\mathsf{S}_{\mathsf{cpa}}.\mathsf{Enc}(1^\lambda, u, \ell : (\sigma, L_{(\cdot)}, \sigma_{\mathsf{P}}))}$	$\underline{\mathsf{S}_{\mathsf{cpa}}.\mathsf{Exp}(1^\lambda, u, M_u, C_u : (\sigma, L_{(\cdot)}, \sigma_{\mathsf{P}}))}$
$\sigma \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{SE}}.\mathsf{Init}(1^\lambda)$	If $L_u = \perp$ then	If $L_u = \perp$ then $L_u \leftarrow\!\!\$\ \mathsf{F}.\mathsf{Kg}(1^\lambda)$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P}_2.\mathsf{Init}(1^\lambda)$	$\quad L_u \leftarrow\!\!\$\ \mathsf{F}.\mathsf{Kg}(1^\lambda)$	For $i = 1, \ldots, \lvert C_u \rvert$ do
Return $(\sigma, \perp, \sigma_{\mathsf{P}})$	$c_{\mathsf{SE}} \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{SE}}.\mathsf{Enc}(1^\lambda, u, \ell : \sigma)$	$\quad (c_{\mathsf{SE}}, \tau) \leftarrow C_u[i]\,;\, C'_u.\mathsf{add}(c_{\mathsf{SE}})$
$\underline{\mathsf{S}_{\mathsf{cpa}}.\mathsf{Prim}(1^\lambda, x : (\sigma, L_{(\cdot)}, \sigma_{\mathsf{P}}))}$	$\tau \leftarrow \mathsf{F}.\mathsf{Ev}^{\mathsf{P}_2}(1^\lambda, L_u, c_{\mathsf{SE}})$	$K_{\mathsf{SE}} \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{cpa}}.\mathsf{Exp}(1^\lambda, u, M_u, C'_u : \sigma)$
$(d, x) \leftarrow x$	Return (c_{SE}, τ)	Return (K_{SE}, L_u)
If $d = 1$ then		
$\quad y \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{SE}}.\mathsf{Prim}(1^\lambda, x : \sigma)$		
Else		
$\quad y \leftarrow\!\!\$\ \mathsf{P}_2.\mathsf{Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$		
Return y		

Given a PPT adversary $\mathcal{A}_{\mathsf{cpa}}$ against the SIM-AC-CPA security of $\mathsf{EtM}[\mathsf{SE},\mathsf{F}]$ it is straightforward to construct a PPT adversary $\mathcal{A}$ against the SIM-AC-CPA security of SE such that

$$\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{EtM}[\mathsf{SE},\mathsf{F}],\mathsf{S}_{\mathsf{cpa}},\mathsf{P}_1\times\mathsf{P}_2,\mathcal{A}_{\mathsf{cpa}}}(\lambda) = \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S}_{\mathsf{SE}},\mathsf{P}_1,\mathcal{A}}(\lambda).$$

From this the claim follows by letting S_{SE} be a simulator (guaranteed to exist by the SIM-AC-CPA security of SE) which makes the latter term negligible. ∎

H Random Oracles and Ideal Ciphers are SIM-AC-PRF Secure

In this section we prove our claims that random oracles and ideal ciphers make good SIM-AC-PRFs.

H.1 Proof of Theorem 5.3 (ROM is SIM-AC-PRF)

Proof: Let $\mathcal{A}_{\mathsf{prf}}$ be any efficient adversary. We will claim $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{R},\mathsf{S}_{\mathsf{prf}},\mathsf{P}_{\mathsf{rom}},\mathcal{A}_{\mathsf{prf}}}$ is negligible where simulator $\mathsf{S}_{\mathsf{prf}}$ is defined as follows.

$\underline{\mathsf{S}_{\mathsf{prf}}.\mathsf{Init}(1^\lambda)}$	$\underline{\mathsf{S}_{\mathsf{prf}}.\mathsf{Ev}(1^\lambda, u, x : (T, K_*))}$	$\underline{\mathsf{S}_{\mathsf{prf}}.\mathsf{Exp}(1^\lambda, u, T_u : (T, K_*))}$
Return $([\cdot], [\cdot])$	$x \leftarrow (K_u \,\|\, x, \mathsf{R}.\mathsf{ol}(\lambda))$	If $K_u = \perp$ then
$\underline{\mathsf{S}_{\mathsf{prf}}.\mathsf{Prim}(1^\lambda, x : (T, K_*))}$	$y \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Prim}(1^\lambda, x : (T, K_*))$	$\quad K_u \leftarrow\!\!\$\ \mathsf{R}.\mathsf{Kg}(\lambda)$
$(x, l) \leftarrow x$	Return y	$\quad$ For $x \in T_u$ do
If $T[x, l] = \perp$ then		$\quad\quad T[K_u \,\|\, x, \mathsf{R}.\mathsf{ol}(\lambda)] \leftarrow T_u[x]$
$\quad T[x, l] \leftarrow\!\!\$\ \{0,1\}^l$		Return K_u
Return $T[x, l]$		

It stores a table T to simulate the random oracle and a list of the keys K_* it has sampled for users so far. The ideal primitive algorithm $\mathsf{S_{prf}.Prim}$ exactly acts as a random oracle. The evaluation algorithm $\mathsf{S_{prf}.Ev}$ simply calls $\mathsf{S_{prf}.Prim}$ in the same way that $\mathsf{R.Ev}$ calls its ideal primitive. The most interesting algorithm is $\mathsf{S_{prf}.Exp}$ which, if K_u has not yet been generated, picks it uniformly at random. Then it programs the random oracle table to be consistent with this having been the key used by that user. It remembers the key it chose and will use it during future evaluations or exposures of that user.

An attacker will only be able to detect the simulation if the retroactive programming would cause an inconsistency in the random oracle. In particular we will show that,

$$\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{R},\mathsf{S_{prf}},\mathsf{P_{rom}},\mathcal{A}_{\mathsf{prf}}}(\lambda) \leq \frac{u_\lambda^2 + p_\lambda u_\lambda}{2^{\mathsf{R.kl}(\lambda)}}$$

where u_λ is an upper bound on the number of users that $\mathcal{A}_{\mathsf{prf}}$ queries to and p_λ is an upper bound on the number of PRIM queries that $\mathcal{A}_{\mathsf{prf}}$ makes. This bound is negligible because u_λ and p_λ are polynomially bounded and $\mathsf{R.kl}(\lambda)$ is super-logarithmic. Because our simulator is agnostic to the labels chosen for users we can assume without loss of generality that it always queries with $u \in [u_\lambda]$.

In the proof we consider a sequence of games G_0 through G_3 which transform the real world of $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{R},\mathsf{S_{prf}},\mathsf{P_{rom}},\mathcal{A}_{\mathsf{prf}}}$ into the ideal world. The inequality above follows from straightforward calculations based on the following claims which we will justify.

1. $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{R},\mathsf{S_{prf}},\mathsf{P_{rom}},\mathcal{A}_{\mathsf{prf}}} = \Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_3(\lambda)]$
2. $\Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_1(\lambda)] \leq \binom{u_\lambda}{2} / 2^{\mathsf{R.kl}(\lambda)}$
3. $\Pr[\mathrm{G}_1(\lambda)] - \Pr[\mathrm{G}_2(\lambda)] \leq \binom{u_\lambda}{2} / 2^{\mathsf{R.kl}(\lambda)}$
4. $\Pr[\mathrm{G}_2(\lambda)] - \Pr[\mathrm{G}_3(\lambda)] \leq p_\lambda u_\lambda / 2^{\mathsf{R.kl}(\lambda)}$

Claim 2. We start by comparing G_0 (which we define to be identical to $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}prf}}$ with b hardcoded to 1) to the game G_1 (which is shown in Fig. 36 along with G_2 and G_3). Game G_1 includes both the highlighted code and the boxed code. We will argue that their behavior differs only in that game G_1 samples users' keys without replacement.[9] Claim 1 is then obtained as a bound on the statistical distance between these two ways of sampling keys.

In game G_1 the values of the random oracle are distributed across the random oracle table T and the per-user tables T_u. If the user is unexposed in EV, then rather than calling PRIM with the appropriate input the output is instead chosen locally using T_u (and picked at random, then remembered if $T_u[x] = \bot$). If the user is unexposed, PRIM is simply called with the appropriate input. The value stored as $T_u[x]$ corresponds to the random oracle entry $T[K_u \,\|\, x, \mathsf{R.ol}(\lambda)]$. The first is set by a $\textsc{Ev}(u, x)$ query while the second is set by a query $\textsc{Prim}(K_u \,\|\, x, \mathsf{R.ol}(\lambda))$.

As described so far there would be inconsistency in the game if the adversary was able to make such queries. To resolve this, code has been added in various places to maintain consistency of the tables. Note that since the K_u are sampled without replacement we do not need to worry about maintaining consistency between T_u for differing users. The consistency maintenance occurs in three place. The first time a user is exposed $T[K_u \,\|\, x, \mathsf{R.ol}(\lambda)]$ is set equal to $T_u[x]$ for all non-$\bot$ entries of T_u. When $\textsc{Ev}(u, x)$ is called for an unexposed user, $T_u[x]$ will be set to $T[K_u \,\|\, x, \mathsf{R.ol}(\lambda)]$ (if the latter is already defined and the latter is not). Similarly, if $\textsc{Prim}(K_u \,\|\, x, \mathsf{R.ol}(\lambda))$ is called while u is unexposed then $T[K_u \,\|\, x, \mathsf{R.ol}(\lambda)]$ will be set to $T_u[x]$.

[9] The corresponding code in G_1 is ill-defined if $u_\lambda > 2^{\mathsf{R.kl}}$. We can ignore this issue since this value is super-polynomial and we only care about PPT attackers.

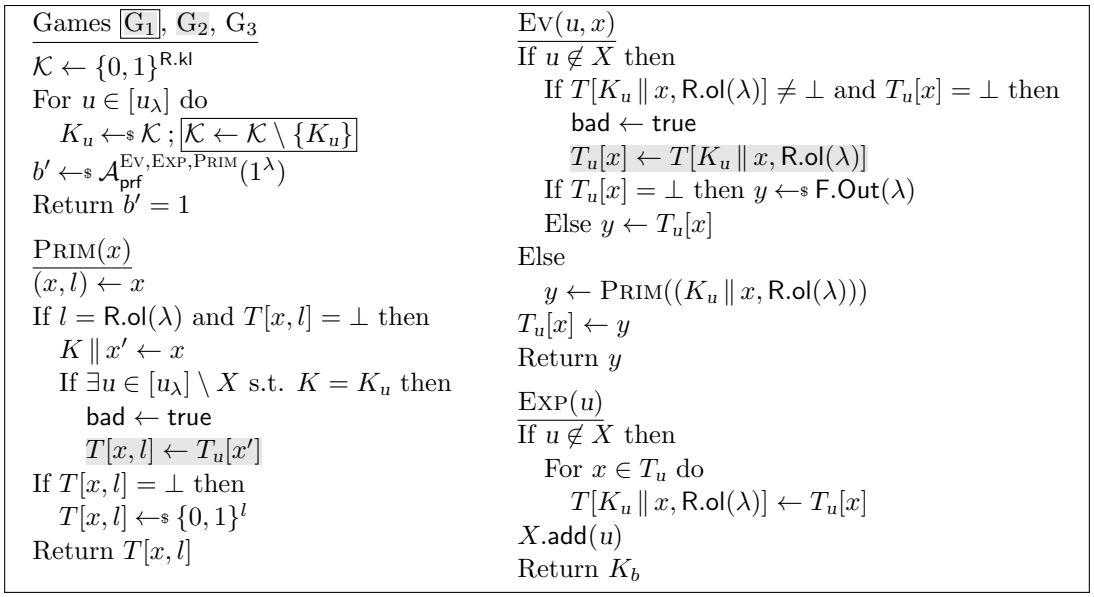

Figure 36: Games used in the proof of Theorem 5.3. Boxed code is only included in game G_1. Higlighted code is not included in game G_3.

Formally, we claim that for an execution of the game by an adversary, all x, and u it holds that each query of the form $\text{Ev}(u, x)$ or $\text{Prim}(K_u \,\|\, x, \mathsf{R.ol}(\lambda))$ returned the same value, which was chosen uniformly at random. We will argue in cases depending on which type of query was made first that $T[K_u \,\|\, x, \mathsf{R.ol}(\lambda)]$ and $T_u[x]$ will each only ever store at most one non-$\perp$ value, which was chosen uniformly. The above property then follows by observing that the oracles only ever return values from these tables and never do so when the tables are storing $\perp$.

Suppose such a Prim query was made first. At that time $T[K_u \,\|\, x, \mathsf{R.ol}(\lambda)]$ will first be set equal to $T_u[x]$ (which is $\perp$) and then set again to equal be a uniformly random value, which we will call z. We claim the following invariants will hold for the rest of the execution: $T[K_u \,\|\, x, \mathsf{R.ol}(\lambda)] = z$ and $T_u[x] \in \{z, \perp\}$. This invariant cannot be changed by Prim because it will only modify entries of T that equal $\perp$. This invariant cannot be changed by Exp because it will only set entries of T to equal non-$\perp$ entries of T_u. This invariant cannot be changed by Ev when $u \in X$ because T_u is set to equal the output of Prim. This invariant cannot be changed by Ev when $u \notin X$ because if $T_u[x] \neq \perp$ then it will not be modified and if $T_u[x] = \perp$ then it will be set to equal $T[K_u \,\|\, x, \mathsf{R.ol}(\lambda)]$.

Suppose such a Ev query was made first. If $u \in X$ then this and all future such queries will be forwarded to Prim from which the desired statement holds. Otherwise (since $T[K_u \,\|\, x, \mathsf{R.ol}(\lambda)] = \perp$), $T_u[x]$ will be set to a value which was just sampled uniformly at random. Call this value z. We claim the following invariants will hold for the rest of the execution: $T_u[x] = z$ and $T[K_u \,\|\, x, \mathsf{R.ol}(\lambda)] \in \{z, \perp\}$. This invariant cannot be changed by Exp because it sets entries of T equal to the corresponding entries of T_u. This invariant cannot be changed by Prim when $u \in X$ because $T[K_u \,\|\, x, \mathsf{R.ol}(\lambda)] \in \{y, \perp\}$ will have been set to z in Exp and Prim will not change non-$\perp$ entries of T. This invariant cannot be changed by Prim when $u \notin X$ because if $T[K_u \,\|\, x, \mathsf{R.ol}(\lambda)] = \perp$ then the highlighted code will set it equal to z and otherwise it will be unmodified. This invariant cannot be changed by Ev because if $u \in X$ then $T_u[x]$ will be set to the output of Prim (which will be z)and if $u \notin X$ then it can only be set to z.

This concludes our justification that G_1 differs from G_0 only in the sampling of the users' keys.

Claim 3. Now consider G_2 which differs from the G_1 only in that the boxed code has been removed. In other words, the keys of users are again being sampled with replacement. The given bound again follows from the statistical distance between the two ways the keys are being sampled.

Claim 4. Now consider G_3 which differs from the G_2 only in that the highlighted code (which is executed only after bad is set) has been removed, so by the fundamental lemma of game playing [10],

$$\Pr[G_2(\lambda)] - \Pr[G_3(\lambda)] \leq \Pr[G_3(\lambda) \text{ sets } \mathsf{bad}].$$

The flag can only be set if the adversary makes a query to PRIM of the form $(K \,\|\, x', \mathsf{R.ol}(\lambda))$ where K equals the key of an unexposed users. Note that other than setting bad, which doesn't affect the view of the adversary, a user's key K_u is only used after that user has already been exposed (and hence bad can no longer be set based on it). Before then, the adversaries view is independent of K_u. The claim then follows as a simple union bound over the probability that any particular PRIM query is the first query using a user's key in the specified way.

Claim 1. We can conclude by justifying the first claim. By definition G_0 is identical to the real world of $G^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{R},\mathsf{S}_{\mathsf{prf}},\mathsf{P}_{\mathsf{rom}},\mathcal{A}_{\mathsf{prf}}}$. Hence, we need only note that the view of $\mathcal{A}_{\mathsf{prf}}$ in G_3 is identical to its view in the ideal world of $G^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{R},\mathsf{S}_{\mathsf{prf}},\mathsf{P}_{\mathsf{rom}},\mathcal{A}_{\mathsf{prf}}}$. This can be verified by comparing the code of $\mathsf{S}_{\mathsf{prf}}$ to the corresponding oracles in G_3. Standard conditional probability calculations give the claim. ∎

H.2 Proof of Theorem 5.4 (ICM is SIM-AC-PRF)

Proof: Let $\mathcal{A}_{\mathsf{prf}}$ be any efficient adversary. We will claim $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{B},\mathsf{S}_{\mathsf{prf}},\mathsf{P}_{\mathsf{icm}},\mathcal{A}_{\mathsf{prf}}}$ is negligible where simulator $\mathsf{S}_{\mathsf{prf}}$ is defined as follows.

$\underline{\mathsf{S}_{\mathsf{prf}}.\mathsf{Init}(1^\lambda)}$
Return $([\cdot],[\cdot],[\cdot])$
$\underline{\mathsf{S}_{\mathsf{prf}}.\mathsf{Prim}(1^\lambda, x : (E, D, K_*))}$
$y \leftarrow\!\!\$\ \mathsf{P}^{\mathsf{n}}_{\mathsf{icm}}.\mathsf{Prim}(1^\lambda, x : (E, D))$
Return y

$\underline{\mathsf{S}_{\mathsf{prf}}.\mathsf{Ev}(1^\lambda, u, x : (E, D, K_*))}$
$x \leftarrow (+, K_u, x)$
$y \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Prim}(1^\lambda, x : (E, D, K_*))$
Return y

$\underline{\mathsf{S}_{\mathsf{prf}}.\mathsf{Exp}(1^\lambda, u, T_u : (E, D, K_*))}$
If $K_u = \perp$ then
  $K_u \leftarrow\!\!\$\ \mathsf{B.Kg}(1^\lambda)$
  For $x \in T_u$ do
    $E[K_u, x] \leftarrow T_u[x]$
    $D[K_u, T_u[x]] \leftarrow x$
$X.\mathsf{add}(u)$
Return K_u

It stores a tables E and D to simulate the ideal cipher and a list of the keys K_* it has sampled for users so far. The ideal primitive algorithm $\mathsf{S}_{\mathsf{prf}}.\mathsf{Prim}$ simply runs $\mathsf{P}^{\mathsf{n}}_{\mathsf{icm}}.\mathsf{Prim}$. The evaluation algorithm $\mathsf{S}_{\mathsf{prf}}.\mathsf{Ev}$ simply calls $\mathsf{S}_{\mathsf{prf}}.\mathsf{Prim}$ in the same way that $\mathsf{B.Ev}$ calls its ideal primitive. The most interesting algorithm is $\mathsf{S}_{\mathsf{prf}}.\mathsf{Exp}$ which, if K_u has not yet been generated, picks it uniformly at random. Then it programs the ideal cipher tables to be consistent with this having been the used by that user. It remembers the key it chose and will use it during future evaluations or exposures of that user.

We will prove the following bound on the advantage trying to distinguish this simulation from the real execution of B.

$$\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{B},\mathsf{S}_{\mathsf{prf}},\mathsf{P}^n_{\mathsf{icm}},\mathcal{A}_{\mathsf{prf}}}(\lambda) \leq \frac{u_\lambda^2 + p_\lambda u_\lambda}{2^{\mathsf{B.kl}(\lambda)}} + \frac{q_\lambda^2}{2^{n(\lambda)+1}}$$

where u_λ is an upper bound on the number of users that $\mathcal{A}_{\mathsf{prf}}$ queries to, p_λ is an upper bound on the number of PRIM queries that $\mathcal{A}_{\mathsf{prf}}$ makes, and q_λ is an upper bound on the number of EV queries that $\mathcal{A}_{\mathsf{prf}}$ makes. This bound is negligible because u_λ, p_λ, and q_λ are polynomially bounded and $\mathsf{B.kl}(\lambda)$ and $n(\lambda)$ are super-logarithmic.

In the proof we consider a sequence of games G_0 through G_5 which transform the real world of $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{B},\mathsf{S}_{\mathsf{prf}},\mathsf{P}^n_{\mathsf{icm}},\mathcal{A}_{\mathsf{prf}}}$ into the ideal world. The inequality above follows from straightforward calculations based on the following claims which we will justify.

1. $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{B},\mathsf{S}_{\mathsf{prf}},\mathsf{P}^n_{\mathsf{icm}},\mathcal{A}_{\mathsf{prf}}} = \Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_5(\lambda)]$

2. $\Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_1(\lambda)] \leq \binom{u_\lambda}{2}/2^{\mathsf{B.kl}(\lambda)}$

3. $\Pr[\mathrm{G}_1(\lambda)] - \Pr[\mathrm{G}_2(\lambda)] \leq \binom{u_\lambda}{2}/2^{\mathsf{B.kl}(\lambda)}$

4. $\Pr[\mathrm{G}_2(\lambda)] - \Pr[\mathrm{G}_3(\lambda)] \leq p_\lambda u_\lambda/2^{\mathsf{B.kl}(\lambda)}$

5. $\Pr[\mathrm{G}_3(\lambda)] - \Pr[\mathrm{G}_4(\lambda)] = 0$

6. $\Pr[\mathrm{G}_4(\lambda)] - \Pr[\mathrm{G}_5(\lambda)] \leq \binom{q_\lambda}{2}/2^{n(\lambda)}$

Because our simulator is agnostic to the labels chosen for users we can assume without loss of generality that it always queries with $u \in [u_\lambda]$.

Claim 2. We start by comparing G_0 (which we define to be identical to $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}prf}}$ with b hard-coded to 1) to the game G_1 (which is shown in Fig. 37 along with G_2 and G_3). Game G_1 includes both the highlighted code and the boxed code. We will argue that their behavior differs only in that game G_1 samples users' keys without replacement.[10] Claim 1 is then obtained as a bound on the statistical distance between these two ways of sampling keys.

In these games we have introduced some pseudocode shorthands. The variables $\mathcal{U}$, $\mathcal{E}_K$, $\mathcal{D}_K$, and $\mathcal{E}^u$ are all defined to be the specified sets. Additionally we introduce the algorithm PROG. Calling it via $(E, D) \leftarrow \textsc{Prog}(E, D, E')$ makes E and D consistent with all non-$\perp$ entries of E'. Otherwise E and D are unmodified.

In game G_1 the values of the ideal cipher are distributed across the ideal cipher tables E and D and the per-user tables E_u and D_u. If the user is unexposed in EV, then rather calling PRIM with the appropriate input the output is instead chosen locally using E_u and D_u (and picked at random, then remembered if $E_u[x] = \perp$). If the user is unexposed, PRIM is simply called with the appropriate input. The value stored as $E_u[x]$ corresponds to the ideal cipher entry $D[K_u, x]$. The first is set by a $\textsc{Ev}(u, x)$ query while the second is set by a query $\textsc{Prim}((+, K_u, x))$ or a query $\textsc{Prim}((-, K_u, y))$ in which x happens to be sampled to be inserted into $D[K_u, y]$.

As described so far there would be inconsistency in the game if the adversary was able to make such queries. To resolve this, code has been added in various places to maintain consistency of the tables. Note that since the K_u are sampled without replacement we do not need to worry about maintaining consistency between E_u for differing users. The consistency maintenance occurs in three place. The first time a user is exposed $E[K_u, \cdot]$ and $D[K_u, \cdot]$ are set consistent with all non-$\perp$ entries of $E_u[\cdot]$. Henceforth E_u will be unused. When $\textsc{Ev}(u, x)$ is called for an unexposed user, $E_u[x]$ will be set consistent with non-$\perp$ entries of $E[K_u$ if there are any such non-$\perp$ entries. Similarly, if $\textsc{Prim}((\cdot, K_u, \cdot))$ is called while u is unexposed then $E[K_u, \cdot]$ and $D[K_u, \cdot]$ are set consistent with the non-$\perp$ entries of E_u.

[10] The corresponding code in G_1 is ill-defined if $u_\lambda > 2^{\mathsf{B.kl}}$. We can ignore this issue since the latter value is super-polynomial and we only care about PPT attackers.

Games $\boxed{\mathrm{G}_1}$, G_2, G_3

$\mathcal{K} \leftarrow \{0,1\}^{\mathsf{B.kl}}$
For $u \in [u_\lambda]$ do
  $K_u \leftarrow\!\!\$\ \mathcal{K}$; $\boxed{\mathcal{K} \leftarrow \mathcal{K} \setminus \{K_u\}}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{Ev},\mathrm{Exp},\mathrm{Prim}}_{\mathsf{prf}}(1^\lambda)$
Return $b = b'$

Definitions:
$\mathcal{U} = \{0,1\}^{n(\lambda)}$
$\mathcal{E}_K = \{\ E[K,a] \neq \bot\ :\ a \in \mathcal{U}\ \}$
$\mathcal{D}_K = \{\ D[K,a] \neq \bot\ :\ a \in \mathcal{U}\ \}$
$\mathcal{E}^u = \{\ E_u[a] \neq \bot\ :\ a \in \mathcal{U}\ \}$

$\textsc{Prog}(E, D, E')$
For $x \in E'$ do
  $y \leftarrow E'[x]$
  $E[x] \leftarrow y$; $D[y] \leftarrow x$
Return (E, D)

$\textsc{Ev}(u, x)$
If $u \notin X$ then
  If $\mathcal{E}_{K_u} \neq \emptyset$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$
    $(E_u, D_u) \leftarrow \textsc{Prog}(E_u, D_u, E[K_u, \cdot])$
  If $E_u[x] = \bot$ then
    $z \leftarrow\!\!\$\ \{0,1\}^{n(\lambda)} \setminus \mathcal{E}^u$; $E_u[x] \leftarrow z$; $D_u[z] \leftarrow x$
  $y \leftarrow E_u[x]$
Else
  $y \leftarrow \textsc{Prim}((+, K_u, x))$
Return y

$\textsc{Exp}(u)$
If $u \notin X$ then
  $(E[K_u, \cdot], D[K_u, \cdot]) \leftarrow \textsc{Prog}(E[K_u, \cdot], D[K_u, \cdot], E_u)$
$X.\mathsf{add}(u)$
Return K_u

$\textsc{Prim}(x)$
$(\mathrm{op}, K, y) \leftarrow x$
If $\exists u \in [u_\lambda]$ s.t. $K = K_u$ then
  $\mathsf{bad} \leftarrow \mathsf{true}$
  $(E[K, \cdot], D[K, \cdot]) \leftarrow \textsc{Prog}(E[K, \cdot], D[K, \cdot], E_u)$
If $\mathrm{op} = +$ then
  If $E[K, y] = \bot$ then
    $z \leftarrow\!\!\$\ \{0,1\}^{n(\lambda)} \setminus \mathcal{E}_K$; $E[K, y] \leftarrow z$; $D[K, z] \leftarrow y$
  Return $E[K, y]$
Else
  If $D[K, y] = \bot$ then
    $z \leftarrow\!\!\$\ \{0,1\}^{n(\lambda)} \setminus \mathcal{D}_K$; $D[K, y] \leftarrow z$; $E[K, z] \leftarrow y$
  Return $D[K, y]$

Figure 37: Games used in the proof of Theorem 5.4. Boxed code is only included in game G_1. Highlighted code is not included in game G_3. The games use some additional pseudocode definitions and the algorithm Prog for compactness.

The claim then follows because before one of the tables is used it is always set consistent with the corresponding entries of the other table.

Claim 3. Now consider G_2 which differs from the G_1 only in that the boxed code has been removed. In other words, the keys of users are again being sampled with replacement. The given bound again follows from the statistical distance between the two ways the keys are being sampled.

Claim 4. Now consider G_3 which differs from the G_2 only in that the highlighted code has been removed. This code is executed only after bad is set, so by the fundamental lemma of game playing [10],

$$\Pr[\mathrm{G}_2(\lambda)] - \Pr[\mathrm{G}_3(\lambda)] \leq \Pr[\mathrm{G}_3(\lambda) \text{ sets } \mathsf{bad}].$$

The flag can only be set if the adversary makes a query to Prim of the form $(\cdot, K, \cdot)$ where K equals the key of an unexposed users. Note that other than setting bad, which doesn't affect the view of the adversary, a user's key K_u is only used after that user has already been exposed (and hence bad can no longer be set based on it). Before then, the adversaries view is independent of K_u. The

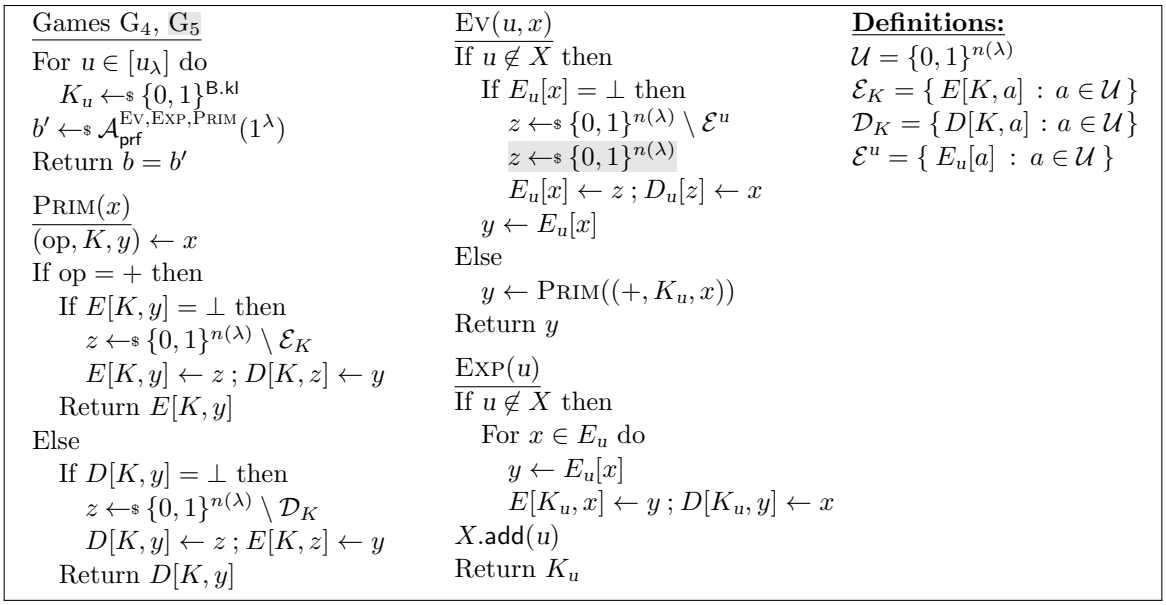

Figure 38: Games used in the proof of Theorem 5.4. Highlighted code is only included in game G_5.

claim then follows as a simple union bound over the probability that any particular Prim query is the first query using a user's key in the specified way.

Claim 5. Now consider game G_4 defined in Fig. 38 (along with G_5). It does not include the highlighted code. It was obtained by simplifying the code of G_3. The initial sampling of keys with replacement was simplified. Code dealing with the setting of bad was dead code in G_3 so could be removed. These are the only differences, so the claim follows.

Claim 6. Note that G_5 differs from G_4 only in how that the values of z are sampled with replacement instead of without. The claim follows from a bound on the statistical distance between these two ways of sampling z.

Claim 1. We conclude by justifying the first claim. By definition G_0 is identical to the real world of $G^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{B},\mathsf{S}_{\mathsf{prf}},\mathsf{P}^n_{\mathsf{icm}},\mathcal{A}_{\mathsf{prf}}}$. Hence, we need only note that the view of $\mathcal{A}_{\mathsf{prf}}$ in G_5 is identical to its view in the ideal world of $G^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{R},\mathsf{S}_{\mathsf{prf}},\mathsf{P}_{\mathsf{rom}},\mathcal{A}_{\mathsf{prf}}}$. Note that the table U_u in G_5 acts exactly like T_u in $G^{\mathsf{sim\text{-}ac\text{-}prf}}$. Thus the correctness of simulation can be verified by comparing the code of $\mathsf{S}_{\mathsf{prf}}$ to the corresponding oracles in G_5. The claim then follows from standard conditional probability calculations. ∎

I Ideal Encryption is SIM-AC-AE Secure

In this section we recall the ideal encryption model (IEM) used in the analysis of Tyagi et al. [36] and show that it gives a SIM-AC-AE secure encryption scheme. While doing so, we identify and show how to fix a bug in their proof which used the ideal encryption model.

Ideal encryption model. The IEM is parameterized by a ciphertext length function $\mathsf{clen} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and captured by $\mathsf{P}^{\mathsf{clen}}_{\mathsf{iem}}$ defined by the following algorithms.

$$\begin{array}{l|l}
\dfrac{\mathsf{P}^{\mathsf{clen}}_{\mathsf{iem}}.\mathsf{Init}(1^\lambda)}{\text{Return } [\cdot]} & \dfrac{\mathsf{P}^{\mathsf{clen}}_{\mathsf{iem}}.\mathsf{Prim}(1^\lambda, x : D)}{(\text{op}, K, y) \leftarrow x} \\
 & \text{If op} = \text{E then} \\
 & \quad c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{clen}(\lambda,|y|)} \\
 & \quad D[K, c] \leftarrow y \\
 & \quad \text{Return } c \\
 & \text{Return } D[K, y]
\end{array}$$

Its state consists of a table D. Queries to the primitive consist of tuples (op, K, y) where y is interpreted as a message if $\text{op} = \text{E}$ and a ciphertext if $\text{op} = \text{D}$ (we assume x is parsed such that one of these cases always holds). Encryption ($\text{op} = \text{E}$) picks a random ciphertext of length $\mathsf{clen}(\lambda, |y|)$ and using D to remember its decryption choice. Decryption ($\text{op} = \text{D}$) simply returns $D[K, y]$.

Technically, $\mathsf{P}^{\mathsf{clen}}_{\mathsf{iem}}$ is not essentially stateless because its response to a decryption query can change over time as more encryption queries are made. Assuming clen is sufficiently expanding, this is not a big issue; we could construct an essentially stateless ideal primitive which is statistically indistinguishable from $\mathsf{P}^{\mathsf{clen}}_{\mathsf{iem}}$ to any algorithm making as most polynomially many queries.

Secure IEM encryption scheme. We can easily construct a SIM-AC-CCA secure encryption scheme in the ideal encryption model by simply querying the ideal object for all functions. This gives the symmetric encryption scheme IEM, defined as follows. It is (implicitly) parameterized by a key-length function $\mathsf{IEM.kl} : \mathbb{N} \to \mathbb{N}$.

$$\begin{array}{l|l|l}
\dfrac{\mathsf{IEM.Kg}(1^\lambda)}{K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{IEM.kl}(\lambda)}} & \dfrac{\mathsf{IEM.Enc}^{\mathsf{P}}(1^\lambda, K, m)}{c \leftarrow \mathsf{P}((\text{E}, K, m))} & \dfrac{\mathsf{IEM.Dec}^{\mathsf{P}}(1^\lambda, K, c)}{\text{Return } \mathsf{P}((\text{D}, K, c))} \\
\text{Return } K & \text{Return } c &
\end{array}$$

Theorem I.1 *Let* $\mathsf{IEM.kl} : \mathbb{N} \to \mathbb{N}$ *and* $\mathsf{clen} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *be fixed. If* $\mathsf{IEM.kl}$ *is super-logarithmic, then* IEM *is SIM-AC-AE secure with* $\mathsf{P}^{\mathsf{clen}}_{\mathsf{iem}}$.

This theorem captures in isolation the IEM programming that was part of the analysis of the main proof of Tyagi et al. [36]. We have identified a bug in their proof, but do not claim that their scheme is insecure. (Indeed, Theorem I.1 combined with Theorem E.1 re-establishes the security of their scheme in the IEM.) Proofs in the IEM are even heavily detail intensive because one has to carefully account for the possibility of a ciphertext being sampled more than once, overwriting an entry in the table D. As we show in Appendix E, our new security definition is a convenient intermediate notion via which the higher-level construction of Tyagi et al. can be proven secure without having to deal with the tedious details of an IEM proof. Moreover, it allows us to concretely show that encryption schemes used in practice (which cannot necessarily be thought of as well modeled by the IEM) can be used to instantiate their protocol.

Proof (Sketch): Here we sketch the main ideas of the proof. The full details are provided in Appendix I.1. In many ways the proof is analogous to the ROM proof of Theorem 5.3, with some subtle extra details that need to be accounted for because of the unique behavior of the IEM. In particular, the current values stored in the table D depend on the *order* that queries were made (in the case that there are repetitions in the c produced). A correct proof must account for this possibility.[11] We do so by introducing an auxiliary table Q which tracks when queries occurred. The SIM-AC-CCA simulator we provide works as follows. Before exposures have occurred, encryption queries are responded to with random strings of the appropriate length and decryption queries are responded to with $\perp$. (Thus the simulator is in $\mathcal{S}_{\$} \cap \mathcal{S}_{\perp}$.) On an exposure, the simulator returns

[11] The bug we identified in the proof of Tyagi et al. [36] stems from not properly accounting for this.

a random key and then programs the decryption table of its $\mathsf{P}_{\mathsf{iem}}$ simulation as if the previous encryption responses for that user had been produced by ideal encryptions using that key. Then all future queries for that user are responded to honestly. We show that this simulation is only detectable by an attacker if it makes an ideal encryption query with some K that is later chosen by the simulator in response to an exposure or if the simulator chooses the same K for two different users. These events happen with negligible probability.▮

I.1 Proof of Theorem I.1 (IEM is SIM-AC-AE)

Proof: One could prove this result by separately proving that IEM is SIM-AC-$ and INT-CTXT secure then referring to Theorem 5.2. However, there is sufficient overlap between those separate proofs that we instead choose to prove the SIM-AC-AE security of IEM directly.

Let $\mathcal{A}_{\mathsf{cca}}$ be any efficient adversary. We claim that $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{IEM},\mathsf{S},\mathsf{P}^{\mathsf{clen}}_{\mathsf{iem}},\mathcal{A}_{\mathsf{cca}}}(\cdot)$ is negligible when the simulator S is defined as follows.

$\underline{\mathsf{S}_{\mathsf{cca}}.\mathsf{Init}(1^\lambda)}$
Return $([\cdot],[\cdot])$

$\underline{\mathsf{S}_{\mathsf{cca}}.\mathsf{Prim}(1^\lambda, x : (D, K_*))}$
$(\mathrm{op}, K, y) \leftarrow x$
If op = E then
  $c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{clen}(\lambda,|y|)}$
  $D[K, c] \leftarrow y$
  Return c
Return $D[K, y]$

$\underline{\mathsf{S}_{\mathsf{cca}}.\mathsf{Enc}(1^\lambda, u, \ell : (D, K_*))}$
If $K_u = \bot$ then
  $c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{clen}(\lambda,\ell)}$
Else
  $c \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{cca}}.\mathsf{Prim}(1^\lambda, (\mathrm{E}, K_u, \ell) : (D, K_*))$
Return c

$\underline{\mathsf{S}_{\mathsf{cca}}.\mathsf{Exp}(1^\lambda, u, M_u, C_u : (D, K_*))}$
If $K_u = \bot$ then
  $K_u \leftarrow\!\!\$\ \mathsf{IEM}.\mathsf{Kg}(\lambda)$
  For $(m, c) \in (M_u, C_u)$ do
    $D_u[c] \leftarrow m$
  For $c \in D_u$ do
    $D[K_u, c] \leftarrow D_u[c]$
Return K_u

$\underline{\mathsf{S}_{\mathsf{cca}}.\mathsf{Dec}(1^\lambda, u, c : (D, K_*))}$
If $K_u = \bot$ then
  Return $\bot$
Else
  $m \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{cca}}.\mathsf{Prim}(1^\lambda, (\mathrm{D}, K_u, c) : (D, K_*))$
Return m

It stores a table D to simulate the ideal encryption model and a list of keys K_* it has sampled for users so far. The ideal primitive algorithm $\mathsf{S}_{\mathsf{cca}}.\mathsf{Prim}$ exactly acts as the ideal encryption model. When a user is unexposed the encryption simulator $\mathsf{S}_{\mathsf{cca}}.\mathsf{Enc}$ returns a random string of the appropriate length and decryption simulator $\mathsf{S}_{\mathsf{cca}}.\mathsf{Dec}$ returns $\bot$, as require for AE security. If the user is already exposed, instead they both call $\mathsf{S}_{\mathsf{cca}}.\mathsf{Prim}$ in the manner that IEM would call its primitive oracle. The most interesting algorithm is $\mathsf{S}_{\mathsf{cca}}.\mathsf{Exp}$ which, if this is the first time u is being exposed, picks K_u uniformly at random. Then it programs the ideal decryption table to correctly decrypt previous encryptions from that user. The key sampled is then remembered for future encryptions, decryptions, or exposures of that user.

An attacker will only be able to detect the simulation if the retroactive programming would cause an inconsistency in the table D. In particular we will show that,

$$\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{IEM},\mathsf{S}_{\mathsf{cca}},\mathsf{P}_{\mathsf{iem}},\mathcal{A}_{\mathsf{cca}}}(\lambda) \leq \frac{u_\lambda^2 + p_\lambda u_\lambda}{2^{\mathsf{IEM}.\mathsf{kl}(\lambda)}}$$

where u_λ is an upper bound on the number of users that $\mathcal{A}_{\mathsf{cca}}$ queries to and p_λ is an upper bound on the number of PRIM queries that $\mathcal{A}_{\mathsf{cca}}$ makes. This bound is negligible because u_λ and p_λ are polynomially bounded and $\mathsf{IEM.kl}(\lambda)$ is super-logarithmic.

In the proof we consider a sequence of games G_0 through G_3 which transform the real world of $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cca}}$ to the ideal world. The inequality above follows from straightforward calculations based on the following claims which we will justify.

1. $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{IEM},\mathsf{S}_{\mathsf{cca}},\mathsf{P}_{\mathsf{iem}},\mathcal{A}_{\mathsf{cca}}}(\lambda) = \Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_3(\lambda)]$
2. $\Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_1(\lambda)] \leq \binom{u_\lambda}{2}/2^{\mathsf{IEM.kl}(\lambda)}$
3. $\Pr[\mathrm{G}_1(\lambda)] - \Pr[\mathrm{G}_2(\lambda)] \leq \binom{u_\lambda}{2}/2^{\mathsf{IEM.kl}(\lambda)}$
4. $\Pr[\mathrm{G}_2(\lambda)] - \Pr[\mathrm{G}_3(\lambda)] \leq p_\lambda u_\lambda/2^{\mathsf{IEM.kl}(\lambda)}$

The general flow of this proof follows that of the proof for Theorem 5.3. We temporarily switch to the keys of users being sampled without replacement so that we do not have to account for them causing inconsistencies in D. Then we argue that inconsistencies between a user and PRIM can only occur it the adversary makes a PRIM queries using a K_u that is at that point independent of the view of the attacker.

Games $\boxed{\mathrm{G}_1(\lambda)}$,$\mathrm{G}_2(\lambda)$,$\mathrm{G}_3(\lambda)$

$\mathcal{K} \leftarrow \{0,1\}^{\mathsf{R.kl}(\lambda)}$
For $u \in [u_\lambda]$ do
  $K_u \leftarrow\!\!\$\ \mathcal{K}$; $\boxed{\mathcal{K} \leftarrow \mathcal{K} \setminus \{K_u\}}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Enc},\textsc{Dec},\textsc{Exp},\textsc{Prim}}_{\mathsf{cca}}(1^\lambda)$
Return $(b' = 1)$

PRIM(x)
$(\mathrm{op}, K, y) \leftarrow x$
If $\mathrm{op} = \mathrm{E}$ then
  $c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{clen}(\lambda,|y|)}$
  $D[K,c] \leftarrow y$; $Q[K,c] \leftarrow t$
  $t \leftarrow t+1$
  Return c
If $\exists u \in [u_\lambda] \setminus X$ s.t. $K = K_u$
  bad $\leftarrow$ true
  If $Q[K,y] < Q_u[y]$ then
    $D[K,y] \leftarrow D_u[y]$
Return $D[K,y]$

ENC(u,m)
Require $m \in \mathsf{SE.M}(\lambda)$
If $u \notin X$ then
  $c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{clen}(\lambda,|m|)}$
  $D_u[c] \leftarrow m$; $Q_u[c] \leftarrow t$
  $t \leftarrow t+1$
Else
  $c \leftarrow$ PRIM$((\mathrm{E}, K_u, m))$
$M_u.\mathsf{add}(m)$; $C_u.\mathsf{add}(c)$
Return c

EXP(u)
If $u \notin X$ then
  For $c \in D_u$ do
    If $D[K_u,c] \neq \perp$ then
      bad $\leftarrow$ true
      If $Q_u[c] < Q[K_u,c]$ then
        $D_u[c] \leftarrow D[K_u,c]$
    $D[K_u,c] \leftarrow D_u[c]$
$X.\mathsf{add}(u)$
Return K_u

DEC(u,c)
If $u \notin X$ then
  If $D[K_u,c] \neq \perp$ then
    bad $\leftarrow$ true
    If $Q_u[c] < Q[K_u,c]$ then
      $D_u[c] \leftarrow D[K_u,c]$
  $m \leftarrow D_u[c]$
Else
  $m \leftarrow$ PRIM$((\mathrm{D}, K_u, c))$
Require $c \notin C_u$
Return m

Figure 39: Games used for proof of Theorem I.1. Boxed code is only included in game G_1. Highlighted code is not included in game G_3.

A unique property of the ideal encryption model is that it is possible for the same ciphertext c to be sampled more than once for a given key K, causing the value of $D[K,c]$ to be overwritten. Hence the current output of the primitive *depends on the order in which prior queries were made.* When rewriting games we need to make sure to account for this. The bug in the proof of Tyagi et al. [36] stems from missing this subtlety. To address this, we introduce the technique of using an auxiliary table Q which tracks when queries were made so that collisions in D can be resolved correctly.

In the coming games we adopt the convention that for $x \in \mathbb{N}$ the expression $x < \perp$ evaluates to false and the expression $\perp < x$ evaluates to true (equivalently, $\perp$ can be thought of as having the value -1 for these comparisons). Because our simulator is agnostic to the labels chosen for users we will assume without loss of generality that the adversary $\mathcal{A}_{\mathsf{cca}}$ always queries with $u \in [u_\lambda]$. This allows us to easily sample all of the required K_u values at the beginning of the games.

Claim 2. We start by comparing game G_0 (which we define to be identical to $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cca}}$ with b hardcoded to 1) to the game G_1 (which is shown in Fig. 39 along with games G_2 and G_3). Game G_1 includes both the highlighted and boxed code. We will argue that their behavior differs only in that game G_1 samples users' keys without replacement.[12] Claim 1 is then obtained as a bound on the statistical distance between these two ways of sampling keys.

In game G_1 the values stored by ideal encryption model are distributed across the table D and the per-user tables D_u. If the user is unexposed in ENC then rather than calling PRIM with the appropriate input, the output is chosen at random locally and stored in D_u. The value stored in $D_u[c]$ corresponds to the entry $D[K_u, c]$. The former is set when c is sampled in a $\textsc{Enc}(u, \cdot)$ query while the latter is set when c is sampled in a $\textsc{Prim}((\mathrm{E}, K_u, \cdot))$ query. To remember when this assignment occured we track a variable t which is incremented whenever an entry would be stored in D. At that time we store t in $Q_u[c]$ or $Q[K, c]$ as appropriate.

There would be potential inconsistency in the game if the two tables were used independently. To resolve this, code has been added in various places to maintain consistency of the tables. Note that since the K_u are sampled without replacement in G_1 we do not need to worry about maintaining consistency between D_u for differing users. The consistency maintenance occurs in three places. In each of these places we compare the appropriate entries of Q and Q_u to verify if anything needs to be done. When $\textsc{Dec}(u, c)$ is called for an unexposed user, $D_u[c]$ will be set to $D[K_u, c]$ (if the latter was defined more recently than the latter). Similarly, if $\textsc{Prim}((\mathrm{D}, K_u, c))$ is called while u is unexposed then $D[K_u, c]$ will be set to $D_u[c]$ (if the latter was defined more recently than the latter). Finally, the first time a user u is exposed $D[K_u, c]$ will be set to $D_u[c]$ where appropriate. Henceforth D_u will be unused because all queries will be forwarded through PRIM.

Formally, we claim that for any execution of the game by an adversary, all c, and all u it holds that each query of the form $\textsc{Dec}(u, c)$ or $\textsc{Prim}((\mathrm{D}, K_u, c))$ the string m returned was the most recent input to a $\textsc{Enc}(u, \cdot)$ or $\textsc{Prim}((\mathrm{E}, K_u, \cdot))$ query that returned c (or $m = \perp$ if no such queries exist).

Let $m_{\textsc{Enc}}$ denote the most recent input to a $\textsc{Enc}(u, \cdot)$ query that returned c and $m_{\textsc{Prim}}$ denote the most recent input to a $\textsc{Prim}((\mathrm{E}, K_u, \cdot))$ query that returned c. Let m^* be whichever of $m_{\textsc{Enc}}$ or $m_{\textsc{Prim}}$ was defined most recently (i.e. the one that should be returned by $\textsc{Dec}(u, c)$ or $\textsc{Prim}((\mathrm{D}, K_u, c))$ queries). We claim the following invariants always hold after the execution of an oracle query. If $m^* = m_{\textsc{Enc}} = m_{\textsc{Dec}}$, then $D_u[c] = D[K_u, c] = m^*$. If $m^* = m_{\textsc{Enc}} \neq m_{\textsc{Prim}}$, then $D_u[c] = m^*$ and $Q_u[c] > Q[K_u, c]$. If $m^* = m_{\textsc{Prim}} \neq m_{\textsc{Enc}}$, then $D[K_u, c] = m^*$ and $Q[K_u, c] > Q_u[c]$.

By analyzing PRIM and DEC we can see that this invariant suffices to imply the desired claim. To prove that it is invariant we first note that it trivially holds at the beginning of execution and then individually note that each oracle is incapable of changing it. We defer the details of this (straightforward) analysis to the end of the proof. That will complete our argument that games G_0 and G_1 differ only in that game G_1 samples users' keys without replacement which gives claim 2.

[12] The corresponding code in G_1 is ill-defined if $u_\lambda > 2^{\mathsf{R.kl}}$. We can ignore this issue since this value is super-polynomial and we only care about PPT attackers.

Claim 3. Now consider G_2 which differs from G_1 only in that the boxed code has been removed. In other words, the keys of users are again being sampled with replacement. The given bound again follows from the statistical distance between the two ways that the keys are being sampled.

Claim 4. How consider game G_3 which differs from G_2 in that the highlighted code has been removed. Note that this code is only executed after bad has been set, so by the fundamental lemma of game playing [10],

$$\Pr[G_2(\lambda)] - \Pr[G_3(\lambda)] \leq \Pr[G_3(\lambda) \text{ sets } \mathsf{bad}].$$

The flag can only be set if the adversary makes a query to PRIM of the form $(\cdot, K, \cdot)$ where K equals the key of an unexposed users. Note that other than setting bad (which doesn't affect the view of the adversary) a user's key K_u is only used after that user has already been exposed (and hence bad can no longer be set based on it). The claim then follows as a union bound over the probability that any particular PRIM query is the first query using a user's key in the specified way.

Claim 1. By definition, G_0 is identical to the real world of $G^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{IEM},\mathsf{S_{cca}},\mathsf{P_{iem}},\mathcal{A}_{\mathsf{cca}}}$. Hence, we only need to note that the view of $\mathcal{A}_{\mathsf{cca}}$ in G_3 is identical to its view in the ideal world of $G^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{IEM},\mathsf{S_{cca}},\mathsf{P_{iem}},\mathcal{A}_{\mathsf{cca}}}$. This can be verified by comparing the code of $\mathsf{S_{cca}}$ to the corresponding oracles in G_3. The only point of note is that $\mathsf{S_{cca}}$ defers the computation of D_u until it is given the corresponding M_u and C_u. The claim follows from standard conditional probability calculations.

Invariant analysis. First we establish that our invariant suffices for the claim. Note that the value returned by a DEC(u, c) or PRIM$((\mathrm{D}, K_u, c))$ will equal $D_u[c]$ if $Q_u[c] > Q[K_u, c]$ and otherwise it will equal $D[K_u, c]$. From the invariant it follows this is always m^*.

At the beginning of execution $m^* = m_{\mathrm{ENC}} = m_{\mathrm{DEC}} = D_u[c] = D[K_u, c] = \perp$, so the invariant holds. A ENC query updating m^* will set $D_u[c] = m^*$ and $Q_u[c] = t$, satisfying the invariant. A PRIM query updating m^* will set $D[K_u, c] = m^*$ and $Q[K_u, c] = t$, satisfying the invariant. Now consider a decryption query to EXP, DEC, or PRIM$((\mathrm{D}, K, y))$. These queries cannot change m^*. They can set an entry of one of the tables to equal the corresponding entry of the other table, but only if the former table's corresponding Q entry was larger. This does not change the invariant.∎

J Proof of Theorem 6.1 (IND-AC-EXT implies SIM-AC-$)

Proof: Let SE be an extractable mode of operation which is IND-AC-EXT secure, F be a family of functions with $\mathsf{F.Inp} = \mathsf{SE.FInp}$ and $\mathsf{F.Out} = \mathsf{SE.FOut}$, and P be an ideal primitive. Let $\mathcal{A}_{\mathsf{cpa}}$ be an adversary against the SIM-AC-$ security of $\mathsf{SE}[\mathsf{F}]$ and $\mathsf{S_{prf}}$ be a SIM-AC-PRF simulator. We will construct adversaries $\mathcal{A}_{\mathsf{prf}}$ and $\mathcal{A}_{\mathsf{ext}}$ along with simulator $\mathsf{S_{cpa}}$ such that

$$\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE}[\mathsf{F}],\mathsf{S}_\$[\mathsf{S_{cpa}}],\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S_{prf}},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda) + \mathsf{Adv}^{\mathsf{ind\text{-}ac\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{ext}}}(\lambda).$$

It will be clear from examination that our adversaries and simulator are efficient assuming that $\mathcal{A}_{\mathsf{cpa}}$ and $\mathsf{S_{prf}}$ are. By assumption the advantage of $\mathcal{A}_{\mathsf{ext}}$ is negligible. Then, letting $\mathsf{S_{prf}}$ be the simulator guaranteed to exist by the assumed security of F gives the desired result.

The proof proceed by considering the sequence of games G_0 through G_4 which gradually transforms game $G^{\mathsf{ind\text{-}cpa}}$ with $b = 1$ to $G^{\mathsf{ind\text{-}ac\text{-}ext}}$ with $b = 0$ using the $\mathsf{S}_\$[\mathsf{S_{cpa}}]$. The inequality above follows from simple calculations based on the following claims which we will justify.

1. $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cpa}}_{\mathsf{SE}[\mathsf{F}],\mathsf{S_{cpa}},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda) = \Pr[G_0(\lambda)] - \Pr[G_4(\lambda)]$

2. $\Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_1(\lambda)] = \mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S}_{\mathsf{prf}},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda)$
3. $\Pr[\mathrm{G}_1(\lambda)] - \Pr[\mathrm{G}_2(\lambda)] = 0$
4. $\Pr[\mathrm{G}_2(\lambda)] - \Pr[\mathrm{G}_3(\lambda)] = \mathsf{Adv}^{\mathsf{ind\text{-}ac\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{ext}}}(\lambda)$
5. $\Pr[\mathrm{G}_3(\lambda)] - \Pr[\mathrm{G}_4(\lambda)] = 0$

Claim 2. We start with the second claim. The games G_0 and G_1 are defined in Fig. 40. Highlighted code is only in G_0 and boxed code is only in G_1. Game G_0 is exactly identical to $\mathrm{G}^{\mathsf{ind\text{-}ac\text{-}ext}}$ with $b = 1$ and was obtained by hardcoding SE and removing code used only for $b = 0$. In G_1, all uses of F (including queries to its ideal primitive and exposure of its key) have been replaced with simulation by $\mathsf{S}_{\mathsf{prf}}$. Note, in particular, the oracle RF which replaces $\mathsf{SE.Enc}$'s oracle access to F. It acts exactly as EV in $\mathrm{G}^{\mathsf{prf}}$ with $b = 0$.

Games G_0,$\boxed{\mathrm{G}_1}$
For $u \in \{0,1\}^*$ do
  $K_u \leftarrow\!\!\$\ \mathsf{SE[F].Kg}(1^\lambda)$
$\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$
$\boxed{\sigma \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Init}(1^\lambda)}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{ENC},\mathrm{EXP},\mathrm{PRIM}}_{\mathsf{cpa}}(1^\lambda)$
Return $(b' = 1)$

PRIM(x)
$y \leftarrow\!\!\$\ \mathsf{P.Prim}(1^\lambda, x : \sigma_{\mathsf{P}})$
$\boxed{y \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Prim}(1^\lambda, x : \sigma)}$
Return y

EXP(u)
$(K_{\mathsf{SE}}, K_{\mathsf{F}}) \leftarrow K_u$
$\boxed{K_{\mathsf{F}} \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Exp}(1^\lambda, u, T_u : \sigma)}$
$X.\mathsf{add}(u)$
Return $(K_{\mathsf{SE}}, K_{\mathsf{F}})$

ENC(u, m)
Require $m \in \mathsf{SE.M}(\lambda)$
$(K_{\mathsf{SE}}, K_{\mathsf{F}}) \leftarrow K_u$
$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{F}^{\mathsf{P}}_{K_{\mathsf{F}}}}(1^\lambda, K_{\mathsf{SE}}, m)$
$\boxed{c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathrm{RF}(u,\cdot)}(1^\lambda, K_{\mathsf{SE}}, m)}$
$M_u.\mathsf{add}(m)\,; C_u.\mathsf{add}(c)$
Return c

RF(u, x)//Only called in G_1
If $u \notin X$ then
  If $T_u[x] = \perp$ then $T_u[x] \leftarrow\!\!\$\ \mathsf{F.Out}(\lambda)$
Else
  $T_u[x] \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Ev}(1^\lambda, u, x : \sigma)$
Return $T_u[x]$

Figure 40: Games used in proof of Theorem 6.1. Note that $\mathcal{A}_{\mathsf{cpa}}$ is not given oracle access to RF. Highlighted code is only included in G_0. Boxed code is only included in G_1.

Detecting these changes corresponds to breaking the SIM-AC-PRF security of F. We capture this via the adversary $\mathcal{A}_{\mathsf{prf}}$ shown in Fig. 41. The adversary runs $\mathcal{A}_{\mathsf{cpa}}$ and then returns whatever $\mathcal{A}_{\mathsf{cpa}}$ does. To simulate EXP and ENC queries it samples and remembers its own K_{SE} for each user queried. For EXP queries it queries its own exposure oracle to obtain the K_{F} to return. To simulate ENC queries it uses its own EV oracle as the oracle that $\mathsf{SE.Enc}$ expects access to. It is straightforward to see that when $b = 1$ (resp. $b = 0$) the view of $\mathcal{A}_{\mathsf{cpa}}$ run by $\mathcal{A}_{\mathsf{prf}}$ exactly matches its view in G_0 (resp. G_1). Standard conditional probability calculations then give the claim.

Adversary $\mathcal{A}^{\mathrm{EV},\mathrm{EXP},\mathrm{PRIM}}_{\mathsf{prf}}(1^\lambda)$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{ENCSIM},\mathrm{EXPSIM},\mathrm{PRIM}}_{\mathsf{cpa}}(1^\lambda)$
Return b'

EXPSIM(u)
If $K_{\mathsf{SE},u} = \perp$ then
  $K_{\mathsf{SE},u} \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
$K_{\mathsf{F}} \leftarrow \mathrm{EXP}(u)$
$X.\mathsf{add}(u)$
Return $(K_{\mathsf{SE},u}, K_{\mathsf{F}})$

ENCSIM(u, m)
Require $m \in \mathsf{SE.M}(\lambda)$
If $K_{\mathsf{SE},u} = \perp$ then
  $K_{\mathsf{SE},u} \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathrm{EV}(u,\cdot)}(1^\lambda, K_{\mathsf{SE},u}, m)$
Return c

Figure 41: SIM-AC-PRF adversary $\mathcal{A}_{\mathsf{prf}}$ used in proof of Theorem 6.1.

Claim 3. For the next claim, consider game G_2 which is defined in Fig. 42 along with G_3 and G_4. Highlighted code is only in G_2 while boxed code is not in G_4. Note that $\mathcal{A}_{\mathsf{cpa}}$ is not given

oracle access to RF or RS. Game G_2 differs from G_1 only in the behavior of oracle ENC. When $u \in X$, both games run $\mathsf{SE.Enc}$ with oracle access to $\mathsf{S_{prf}.Ev}$ and store the values returned in T_u. It is less clear that they are equivalent when $u \notin X$. Game G_1 runs $\mathsf{SE.Enc}$ with fresh random coins and oracle access to a lazily sampled random function. Game G_2 first samples a random ciphertext c and asks $\mathsf{SE.Ext}$ to explain it. Then $\mathsf{SE.Enc}$ is then run with the explanatory coins and a lazily sampled random functions whose outputs (when not already fixed) are chosen via $\vec{y}$ instead of freshly at random. To see that these are actually equivalent, recall the required uniformity of $\mathsf{SE.Ext}$. Since the c given to $\mathsf{SE.Ext}$ is used nowhere else we can apply this uniformity to treat $\vec{y}$ and r as fresh, uniformly chosen values which gives the desired equivalence.

Games $\boxed{G_2}$,$\boxed{G_3}$,G_4	$\underline{\text{ENC}(u,m)}$	$\underline{\text{RF}(u,x)}$
//Unchanged from G_1	Require $m \in \mathsf{SE.M}(\lambda)$	$i \leftarrow i+1$
$\underline{\text{PRIM}(x)}$	$(K_{\mathsf{SE}}, K_{\mathsf{F}}) \leftarrow K_u$	If $T_u[x] \neq \perp$
//Unchanged from G_1	If $u \notin X$ then	$\mathsf{bad} \leftarrow \mathsf{true}$
$\underline{\text{EXP}(u)}$	$c \leftarrow\!\!\$\ \mathsf{SE.Out}(\lambda, \lvert m \rvert)$	$\vec{y}[i] \leftarrow T_u[x]$
//Unchanged from G_1	$(\vec{y}, r) \leftarrow\!\!\$\ \mathsf{SE.Ext}(1^\lambda, K_{\mathsf{SE}}, c, m)$	$T_u[x] \leftarrow \vec{y}[i]$
$\underline{\text{RS}(u,x)}$	$i \leftarrow 0$	Return $T_u[x]$
$T_u[x] \leftarrow\!\!\$\ \mathsf{S_{prf}.Ev}(1^\lambda, u, x : \sigma)$	$c' \leftarrow \mathsf{SE.Enc}^{\text{RF}(u,\cdot)}(1^\lambda, K_{\mathsf{SE}}, m; r)$	
Return $T_u[x]$	$\boxed{c \leftarrow c'}$	
	Else	
	$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\text{RS}(u,\cdot)}(1^\lambda, K_{\mathsf{SE}}, m)$	
	$M_u.\mathsf{add}(m)\ ;\ C_u.\mathsf{add}(c)$	
	Return c	

Figure 42: Games used in proof of Theorem 6.1. Note that $\mathcal{A}_{\mathsf{cpa}}$ is not given oracle access to RF or RS. Highlighted code is only included in G_2. Boxed code is not included in G_4.

Claim 4. For the next claim, consider how game G_3 differs from game G_2. In game G_3, the highlighted code in RF maintaining consistency of the table T_u has been removed. This switch mirrors the dependence of $G^{\mathsf{ind\text{-}ac\text{-}ext}}$ on its secret bit. This is captured by the adversary $\mathcal{A}_{\mathsf{ext}}$ shown in Fig. 43. It runs, $\mathcal{A}_{\mathsf{cpa}}$ as a subroutine using its own oracles to simulate those of $\mathcal{A}_{\mathsf{cpa}}$. Primitive queries are answered just by running $\mathsf{S_{prf}}$. Before a user is exposed, encryption queries are answered by $\mathcal{A}_{\mathsf{ext}}$'s own encryption oracle. When a user is exposed for the first time $\mathcal{A}_{\mathsf{ext}}$ queries its EXP oracle to learn $K_{\mathsf{SE},u}$ and T_u. Then it runs $\mathsf{S_{prf}}$ to determine the K_{F} to return. After a user is exposed, for encryption queries it runs $\mathsf{SE.Enc}$ with the $K_{\mathsf{SE},u}$ it learned and oracle access to $\mathsf{S_{prf}}$ (via RS, which updates T_u). When run by $\mathcal{A}_{\mathsf{ext}}$ the view of $\mathcal{A}_{\mathsf{cpa}}$ with $b = 1$ (resp. $b = 0$) is identical to its view in G_2 (resp. G_3). The claim follows.

Claim 5. For the next claim consider how game G_4 differs from G_3. In the latter ENC returns the output of $\mathsf{SE.Enc}$ for unexposed users while in the former it outputs the randomly sampled c. There are equivalent from the extraction correctness of SE so the claim follows.

Claim 1. We have already argued that the view of $\mathcal{A}_{\mathsf{cpa}}$ in G_0 is the same as its view in $G^{\mathsf{sim\text{-}ac\text{-}cpa}}$ with $b = 1$. Hence we need only to construct a simulator such that the view of $\mathcal{A}_{\mathsf{cpa}}$ with $b = 0$ is the same as its view in G_4. Consider the following simulator $\mathsf{S_{cpa}}$, giving $\mathsf{S_\$}[\mathsf{S_{cpa}}] \in \mathcal{S}_\$$.

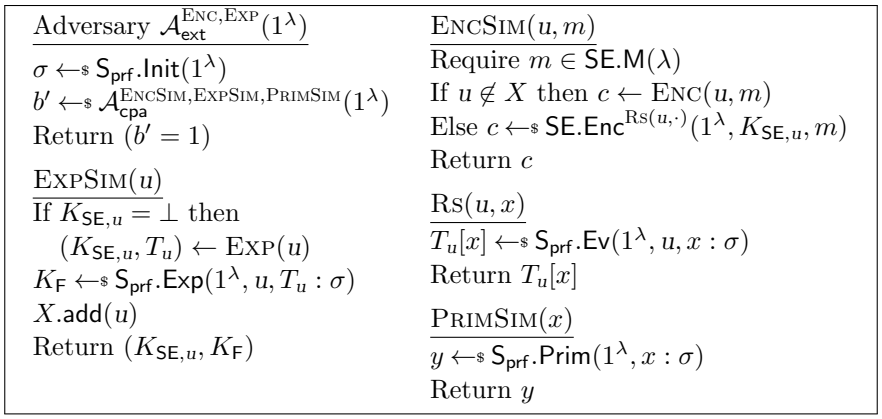

Figure 43: Adversary $\mathcal{A}_{\mathsf{ext}}$ used in proof of Theorem 6.1.

$\mathsf{S}_{\mathsf{cpa}}.\mathsf{Init}(1^\lambda)$
$\sigma \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Init}(1^\lambda)$
$T_* \leftarrow [\cdot]\,;\, K_{\mathsf{SE},*} \leftarrow \bot$
Return $(\sigma, T_*, K_{\mathsf{SE},*})$

$\mathsf{S}_{\mathsf{cpa}}.\mathsf{Prim}(1^\lambda, x : (\sigma, T_*, K_*))$
$y \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Prim}(1^\lambda, x : \sigma)$
Return y

$\mathsf{S}_{\mathsf{cpa}}.\mathsf{Enc}_2(1^\lambda, u, \ell : (\sigma, T_*, K_*))$
$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\text{Rs}(\ell,\cdot)}(1^\lambda, K_{\mathsf{SE},u}, m)$
Return c

Rs(u, x)
$T_u[x] \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Ev}(1^\lambda, u, x : \sigma)$
Return $T_u[x]$

RF(u, x)
$i \leftarrow i + 1$
$T_u[x] \leftarrow \vec{y}[i]$
Return $T_u[x]$

$\mathsf{S}_{\mathsf{cpa}}.\mathsf{Exp}(1^\lambda, u, M_u, C_u : (\sigma, T_*, K_{\mathsf{SE},*}))$
If $K_{\mathsf{SE},u} = \bot$
$K_{\mathsf{SE},u} \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
For $(m, c) \in (M_u, C_u)$ do
$(\vec{y}, r) \leftarrow\!\!\$\ \mathsf{SE.Ext}(1^\lambda, K_{\mathsf{SE},u}, c, m)$
$i \leftarrow 0$
$\mathsf{SE.Enc}^{\text{RF}(u,\cdot)}(1^\lambda, K_{\mathsf{SE},u}, m; r)$
$K_{\mathsf{F}} \leftarrow\!\!\$\ \mathsf{S}_{\mathsf{prf}}.\mathsf{Exp}(1^\lambda, u, T_u : \sigma)$
Return $(K_{\mathsf{SE},u}, K_{\mathsf{F}})$

It stores state for $\mathsf{S}_{\mathsf{prf}}$ along with a table T_u and key $K_{\mathsf{SE},u}$ for each user u. For primitive queries, it just runs $\mathsf{S}_{\mathsf{prf}}$. For post-exposure encryptions, it runs $\mathsf{SE.Enc}$ with $K_{\mathsf{SE},u}$ and oracle access to $\mathsf{S}_{\mathsf{prf}}$ (via Rs, which updates T_u). For pre-exposure encryptions, a random ciphertext was chosen (since $\mathsf{S}_{\$}[\mathsf{S}_{\mathsf{cpa}}] \in \mathcal{S}_{\$}$). The first time that user is exposed, $\mathsf{S}_{\mathsf{cpa}}$ samples its own $K_{\mathsf{SE},u}$. Using this and the tables M_u and C_u it can run the code that would have been run in G_4 to define the table T_u which is given to $\mathsf{S}_{\mathsf{prf}}$ to determine K_{F}. For future exposures, $K_{\mathsf{SE},u}$ is already known and K_{F} is obtained by again running $\mathsf{S}_{\mathsf{prf}}$ on the potentially updated T_u. From this explanation, we can see that the view of $\mathcal{A}_{\mathsf{cpa}}$ in $\text{G}^{\mathsf{sim\text{-}ac\text{-}cpa}}$ with $b = 0$ is the same as its view in G_4, completing the proof.∎

K Inferring Extraction Security from Existing Analysis

In this section we will show how IND-AC-EXT security is often implicit in existing IND-$ security proofs. We start by introducing two variants of it. The first (IND-EXT) is a strictly weaker notion which is necessarily implied by IND-$ security. The other (BAD-EXT) is often exactly what is proven in IND-$ security proofs and suffices to imply IND-AC-EXT security.

Extraction security without compromises. Recall the game $\text{G}^{\mathsf{ind\text{-}ac\text{-}ext}}$ defined in Section 6, Fig. 8. If $\mathcal{A}$ never makes any queries to Exp and only ever queries Enc with one value of u we say that $\mathcal{A}$ is a single-user, non-compromising adversary. We say that SE is IND-EXT secure if $\mathsf{Adv}^{\mathsf{ind\text{-}ac\text{-}ext}}_{\mathsf{SE},\mathcal{A}}(\cdot)$ is negligible for all PPT single-user, non-compromising $\mathcal{A}$.

Extraction security via "bad" flag. Next, we define a security definition which is a slight strengthening of IND-EXT security and is, in fact, implicit in a very natural way of proving IND-EXT security. Consider game $\mathrm{G}^{\mathsf{bad\text{-}ext}}$ shown in Fig. 44 which is parameterized by a mode of operation SE. In this game, the adversary is given access to an encryption oracle that samples a random ciphertext and then uses $\mathsf{SE.Ext}$ to explain it as in the $b = 0$ case of $\mathrm{G}^{\mathsf{ind\text{-}ac\text{-}ext}}$. Should there ever be a potential inconsistency in the explanation (i.e. $\textsc{Rf}$ is ever queried twice on the same input) then a flag bad is set. The adversary wins if this flag is ever set. We define $\mathsf{Adv}^{\mathsf{bad\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{bad}}}(\lambda) = \Pr[\mathrm{G}^{\mathsf{bad\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{bad}}}(\lambda)]$ and say that SE is BAD-EXT secure if $\mathsf{Adv}^{\mathsf{bad\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{bad}}}(\cdot)$ is negligible for all PPT $\mathcal{A}_{\mathsf{bad}}$.

Game $\mathrm{G}^{\mathsf{bad\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{bad}}}(\lambda)$	$\textsc{Enc}(m)$	$\textsc{Rf}(x)$
$K_{\mathsf{SE}} \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	Require $m \in \mathsf{SE.M}(\lambda)$	$i \leftarrow i + 1$
$\mathsf{bad} \leftarrow \mathsf{false}$	$c \leftarrow\!\!\$\ \mathsf{SE.Out}(\lambda, \lvert m \rvert)$	If $T[x] \neq \perp$ then
Run $\mathcal{A}^{\textsc{Enc}}_{\mathsf{bad}}(1^\lambda)$	$(\vec{y}, r) \leftarrow\!\!\$\ \mathsf{SE.Ext}(1^\lambda, K_{\mathsf{SE}}, c, m)$	$\quad \mathsf{bad} \leftarrow \mathsf{true}$
Return bad	$i \leftarrow 0$	$T[x] \leftarrow \vec{y}[i]$
	$c \leftarrow \mathsf{SE.Enc}^{\textsc{Rf}(\cdot)}(1^\lambda, K_{\mathsf{SE}}, m; r)$	Return $T[x]$
	Return c	

Figure 44: Game defining BAD-EXT security of SE. Note that the adversary is not given oracle access to $\textsc{Rf}$.

The value of BAD-EXT security is that it nice to analyze (in particular, is typically implicit in existing IND-$ security proofs for SE). Note that when analyzing BAD-EXT security it may be useful to think of $\vec{y}$ and r as having been picked uniformly at random using the uniformity of $\mathsf{SE.Ext}$. Moreover, BAD-EXT security suffices to imply full IND-AC-EXT security. This is captured by the following theorem whose proof is deferred to the end of this section. The proof is a somewhat straightforward combination of an identical-until-bad proof and an index-guessing proof.

Theorem K.1 *Let SE be an extractable mode of operation. If it is BAD-EXT secure, then it is IND-AC-EXT secure.*

Inferring extraction security from existing proofs. Now we proceed to giving the intuition for why BAD-EXT security is often implicitly included in existing IND-$ security proofs for SE. Towards this we start by showing that such a security proof necessarily implies the IND-EXT security of SE with the following lemma.

Lemma K.2 *Let SE be an extractable mode of operation. If $\mathsf{SE}[\mathsf{F}]$ is single-user IND-$ secure for all single-user PRF secure F with $\mathsf{F.Inp} = \mathsf{F.FInp}$ and $\mathsf{F.Out} = \mathsf{SE.FOut}$ (and there exists at least one such F), then SE is IND-EXT secure.*

The assumption that such an F exists is technically not necessary because one can always be constructed in an appropriately chosen ideal model. We make it explicit because our proof will make used of the assumed existence of such an F.

Proof: Let SE be a mode of operation satisfying the properties of the lemma, F be a family of functions with $\mathsf{F.Inp} = \mathsf{F.FInp}$ and $\mathsf{F.Out} = \mathsf{SE.FOut}$, P be an ideal primitive, and $\mathcal{A}$ be an efficient IND-EXT adversary. We will construct efficient single-user IND-$ adversary $\mathcal{A}_{\mathsf{cpa}}$ and single-user PRF adversary $\mathcal{A}_{\mathsf{prf}}$ such that the following holds.

$$\mathsf{Adv}^{\mathsf{ind\text{-}ac\text{-}ext}}_{\mathsf{SE},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ind\text{-}\$}}_{\mathsf{SE}[\mathsf{F}],\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda) + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda)$$

The theorem then follows when F and P are chosen so that F is PRF secure with P.

The proof proceed by considering the sequence of games G_0 through G_4 which gradually transforms game $\mathrm{G}^{\mathsf{ind\text{-}ac\text{-}ext}}$ with $b = 0$ to $\mathrm{G}^{\mathsf{ind\text{-}ac\text{-}ext}}$ with $b = 1$. The inequality above follows from simple calculations based on the following claims which we will justify.

1. $\mathsf{Adv}^{\mathsf{ind\text{-}ac\text{-}ext}}_{\mathsf{SE},\mathcal{A}}(\lambda) = \Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_4(\lambda)]$
2. $\Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_1(\lambda)] = 0$
3. $\Pr[\mathrm{G}_1(\lambda)] - \Pr[\mathrm{G}_2(\lambda)] = \mathsf{Adv}^{\mathsf{ind\text{-}\$}}_{\mathsf{SE[F]},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda)$
4. $\Pr[\mathrm{G}_2(\lambda)] - \Pr[\mathrm{G}_3(\lambda)] = 0$
5. $\Pr[\mathrm{G}_3(\lambda)] - \Pr[\mathrm{G}_4(\lambda)] = \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda)$

Claim 2. We start with the second claim. The games G_0 through G_2 are defined in Fig. 45. Highlighted code is only in G_0 and boxed code is only in G_2. Game G_0 is exactly identical to $\mathrm{G}^{\mathsf{ind\text{-}ac\text{-}ext}}$ with $b = 0$. We have omitted reference to the Exp oracle since it is never queried. For each Enc query a random ciphertext c is sampled, then explained by $\mathsf{SE.Ext}$, and then reproduced by running $\mathsf{SE.Enc}$ with the values output by $\mathsf{SE.Ext}$. In G_2, the code reproducing c is omitted. By the correctness of $\mathsf{SE.Ext}$ this makes no difference, giving the claim.

Games $\mathrm{G}_0(\lambda)$,$\mathrm{G}_1(\lambda)$,$\boxed{\mathrm{G}_2(\lambda)}$	$\textsc{Enc}(u, m)$	$\textsc{Rf}(u, x)$
For $u \in \{0,1\}^*$ do	Require $m \in \mathsf{SE.M}(\lambda)$	$i \leftarrow i + 1$
$\quad K_{\mathsf{SE},u} \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	$c \leftarrow\!\!\$\ \mathsf{SE.Out}(\lambda, \lvert m\rvert)$	Return $\vec{y}[i]$
$\quad \boxed{K_{\mathsf{F},u} \leftarrow\!\!\$\ \mathsf{F.Kg}(1^\lambda)}$	$\boxed{c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{F}^{\mathsf{P}}_{K_{\mathsf{F},u}}}(1^\lambda, K_{\mathsf{SE},u}, m)}$	
$\boxed{\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)}$	$(\vec{y}, r) \leftarrow\!\!\$\ \mathsf{SE.Ext}(1^\lambda, K_{\mathsf{SE},u}, c, m)$	
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Enc}}(1^\lambda)$	$i \leftarrow 0$	
Return $(b' = 0)$	$c \leftarrow \mathsf{SE.Enc}^{\textsc{Rf}(u,\cdot)}(1^\lambda, K_{\mathsf{SE},u}, m; r)$	
	Return c	

Figure 45: Games used in proof of Lemma K.2. Highlighted code is only included in G_0. Boxed code is only included in G_2. Note that Rf is only ever called in G_0.

Claim 3. For the next claim, consider how G_2 differs from G_1. In it, the randomly chosen ciphertext is overwritten by an honestly generated one. It will follow from the IND-\$ security of $\mathsf{SE[F]}$ that this change cannot be detected. Consider the IND-\$ adversary $\mathcal{A}_{\mathsf{cpa}}$ which simply runs $\mathcal{A}$ and forward on its Enc queries and final output. Note that when $b = 0$ the view of $\mathcal{A}$ perfectly matches its view in G_1 and when $b = 1$ the view of $\mathcal{A}$ perfectly matches its view in G_2. Standard conditional probability calculations then give the claim.

Claim 4. For the next claim consider G_3 which, along with G_4, is defined in Fig. 46. Game G_3 is simply a rewritten version of G_2 in which the oracle $\mathsf{F}^{\mathsf{P}}_{K_{\mathsf{F},u}}$ has been rewritten to be part of the oracle Rf. (Note that this Rf is unrelated to the similarly named oracle from Fig. 45 which could never be called in G_2.)

Claim 5. For the next claim, consider how G_4 differs from G_3. In the latter, the output of Rf is chosen at random instead of being chosen by F. These random values are kept consistent across queries by the table T_u. Unsurprisingly, distinguishing between these games reduces to breaking the PRF security of F. Consider the following adversary.

Games $\boxed{\mathrm{G}_3(\lambda)}$,$\mathrm{G}_4(\lambda)$	$\underline{\textsc{Enc}(u, m)}$	$\underline{\textsc{Rf}(u, x)}$
For $u \in \{0,1\}^*$ do	Require $m \in \mathsf{SE.M}(\lambda)$	If $T_u[x] = \bot$ then
$\quad K_{\mathsf{SE},u} \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\textsc{Rf}}(1^\lambda, K_{\mathsf{SE},u}, m)$	$\quad \boxed{T_u[x] \leftarrow \mathsf{F.Ev}^{\mathsf{P}}(1^\lambda, K_{\mathsf{F},u}, x)}$
$\quad \boxed{K_{\mathsf{F},u} \leftarrow\!\!\$\ \mathsf{F.Kg}(1^\lambda)}$	Return c	$\quad T_u[x] \leftarrow\!\!\$\ \mathsf{F.Out}(\lambda)$
$\boxed{\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)}$		Return $T_u[x]$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Enc}}(1^\lambda)$		
Return $(b' = 0)$		

Figure 46: Games used in proof of Lemma K.2. Boxed code is only included in G_3. Highlighted code is only included in G_4.

Adversary $\mathcal{A}_{\mathsf{prf}}^{\textsc{Ev},\textsc{Prim}}(\lambda)$	$\underline{\textsc{Enc}(u, m)}$
For $u \in \{0,1\}^*$ do	Require $m \in \mathsf{SE.M}(\lambda)$
$\quad K_{\mathsf{SE},u} \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	$c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\textsc{Ev}}(1^\lambda, K_{\mathsf{SE},u}, m)$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{EncSim}}(1^\lambda)$	Return c
Return $b' \oplus 1$	

It runs $\mathcal{A}_{\mathsf{prf}}$ as in both of these game, using its own Ev oracle to emulate Rf. When $\mathcal{A}_{\mathsf{prf}}$ returns a bit it flips the bit before returning it. When $b = 1$ (resp. $b = 0$) the view of $\mathcal{A}$ is identical to its view in game G_3 (resp. G_4). This observation and standard calculations give the claim.

Claim 6. The final claim follows noting that the view of $\mathcal{A}$ in G_0 is identical to its view in $\mathrm{G}^{\mathsf{ind\text{-}ac\text{-}ext}}$ when $b = 0$ and the view of $\mathcal{A}$ in G_4 is identical to its view in $\mathrm{G}^{\mathsf{ind\text{-}ac\text{-}ext}}$ when $b = 1$.∎

We have claimed that not only can IND-EXT be inferred from IND-$ security proofs, but often BAD-EXT as well. Let games G'_0 through G'_4 be defined analogously to the similarly named games in the proof above except that $\mathsf{P.Init}$ is run in all of them (instead of just G_2 and G_3) and the adversary is given oracle access to P. Then IND-$ security corresponds to arguing that G'_1 and G'_2 are indistinguishable. A common proof strategy would analyze the games in the order G'_2, G'_3, G'_4, G'_1.[13] The proof will argue that games G'_3 and G'_4 are indistinguishable due to the PRF security of F and will *directly analyze* the difference between games G'_4 and G'_1. The other relevant pairs of games are identical from the arguments we gave. The most common way of bounding the difference between G'_4 and G'_1 is exactly the identical-until-bad argument needed to prove BAD-EXT security.

We conclude the section with the proof that BAD-EXT security implies IND-AC-EXT security.

Proof (of Theorem K.1): Let $\mathcal{A}_{\mathsf{ext}}$ be an efficient adversary against the IND-AC-EXT security of SE. Assume, without loss of generality, that $\mathcal{A}_{\mathsf{ext}}$ always queries with $u \in [u_\lambda]$ where $u_{(\cdot)} : \mathbb{N} \to \mathbb{N}$ is a polynomial. We will construct an efficient adversary $\mathcal{A}_{\mathsf{bad}}$ against the BAD-EXT security of SE such that the following holds, establishing the theorem.

$$\mathsf{Adv}^{\mathsf{ind\text{-}ac\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{ext}}}(\lambda) \leq u_\lambda \cdot \mathsf{Adv}^{\mathsf{bad\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{bad}}}(\lambda).$$

First, consider the games G_0 and G_1 in Fig. 47 for which we claim $\mathsf{Adv}^{\mathsf{ind\text{-}ac\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{ext}}}(\lambda) = \Pr[\mathrm{G}_0] - \Pr[\mathrm{G}_1]$. The boxed code is included only in G_0. The games were obtained from $\mathrm{G}^{\mathsf{ind\text{-}ac\text{-}ext}}$ by: (1) removing the sampling of b, (2) changing the final return statement, and (3) modifying Rf to add bag flags and put the boxed code in a box instead of following a conditional dependent on b. Thus standard conditional probability calculations give the claim.

[13] Existing proofs are, of course, likely to write the games differently and omit some of the identical games or add extra games between G'_4 and G'_1 to aid their analysis.

Games $\boxed{\mathrm{G}_0(\lambda)}$,$\mathrm{G}_1(\lambda)$	$\textsc{Exp}(u)$	$\textsc{Enc}(u, m)$	$\textsc{Rf}(u, x)$
For $u \in \{0,1\}^*$ do	$X.\mathsf{add}(u)$	Require $m \in \mathsf{SE.M}(\lambda)$	$i \leftarrow i + 1$
$\quad K_{\mathsf{SE},u} \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	Return $(K_{\mathsf{SE},u}, T_u)$	Require $u \notin X$	If $T_u[x] \neq \perp$ then
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Enc},\textsc{Exp}}(1^\lambda)$		$c \leftarrow\!\!\$\ \mathsf{SE.Out}(\lambda, \lvert m\rvert)$	$\quad \mathsf{bad} \leftarrow \mathsf{true}$
Return $(b' = 1)$		$(\vec{y}, r) \leftarrow\!\!\$\ \mathsf{SE.Ext}(1^\lambda, K_{\mathsf{SE},u}, c, m)$	$\quad \mathsf{bad}_u \leftarrow \mathsf{true}$
		$i \leftarrow 0$	$\quad \boxed{\vec{y}[i] \leftarrow T_u[x]}$
		$c \leftarrow \mathsf{SE.Enc}^{\textsc{Rf}(u,\cdot)}(1^\lambda, K_{\mathsf{SE},u}, m; r)$	$T_u[x] \leftarrow \vec{y}[i]$
		Return c	Return $T_u[x]$

Figure 47: Game used in proof of Theorem K.1. Boxed code is only included in G_0.

Note that G_0 and G_1 are identical-until-bad, so the fundamental lemma of game playing [10] gives

$$\Pr[\mathrm{G}_0(\lambda)] - \Pr[\mathrm{G}_1(\lambda)] \leq \Pr[\mathrm{G}_1(\lambda) \text{ sets } \mathsf{bad}].$$

Now we can construct a BAD-EXT adversary which succeeds with at least $1/u_\lambda$ the probability that G_1 sets bad, which gives the result. Note that the event that $\mathrm{G}_1(\lambda)$ sets bad is the disjoint union of the events for each $u \in [u_\lambda]$ that bad_u is set and is the *first* such flag to be set.

Adversary $\mathcal{A}^{\textsc{Enc}}_{\mathsf{bad}}(\lambda)$	$\textsc{ExpSim}(u)$	$\textsc{EncSim}(u, m)$	$\textsc{Rf}(u, x)$
$u^* \leftarrow\!\!\$\ [u_\lambda]$	$X.\mathsf{add}(u)$	Require $m \in \mathsf{SE.M}(\lambda)$	$i \leftarrow i + 1$
For $u \in [u_\lambda] \setminus \{u^*\}$ do	If $u = u^*$ then	Require $u \notin X$	$T_u[x] \leftarrow \vec{y}[i]$
$\quad K_{\mathsf{SE},u} \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$	$\quad \mathbf{abort}()$	If $u = u^*$ then	Return $T_u[x]$
Run $\mathcal{A}^{\textsc{EncSim},\textsc{ExpSim}}(1^\lambda)$	Return $(K_{\mathsf{SE},u}, T_u)$	$\quad c \leftarrow \textsc{Enc}(m)$	
Return		Else	
		$\quad c \leftarrow\!\!\$\ \mathsf{SE.Out}(\lambda, \lvert m\rvert)$	
		$\quad (\vec{y}, r) \leftarrow\!\!\$\ \mathsf{SE.Ext}(1^\lambda, K_{\mathsf{SE},u}, c, m)$	
		$\quad i \leftarrow 0$	
		$\quad c \leftarrow \mathsf{SE.Enc}^{\textsc{Rf}(u,\cdot)}(1^\lambda, K_{\mathsf{SE},u}, m; r)$	
		Return c	

Figure 48: Adversary used in proof of Theorem K.1

Our adversary $\mathcal{A}_{\mathsf{bad}}$ (shown in Fig. 48) attempts to guess the u^* for which bad_u will first be set. Then it runs $\mathcal{A}_{\mathsf{ext}}$. It forwards all queries for that user to its own oracles and locally simulates the queries for all other users. Should $\textsc{Exp}$ ever be queried on u^* it is no longer possible that bad_{u^*} would be set (due to the second require statement in $\textsc{Enc}$) so $\mathcal{A}_{\mathsf{bad}}$ halts execution at this point (represented by the pseudocode command $\mathbf{abort}()$. Thus, whenever bad_{u^*} would be set in G_1 the flag bad will be set in $\mathrm{G}^{\mathsf{bad\text{-}ext}}$.

For $u \in [u_\lambda]$ let bad_u denote the event that bad_u is the first such flag set in G_1. Then we can perform the following calculations.

$$\begin{aligned}
\Pr[\mathrm{G}_1(\lambda) \text{ sets } \mathsf{bad}] &= \sum_{u \in [u_\lambda]} \Pr[\mathsf{bad}_u] \leq \sum_{u \in [u_\lambda]} \Pr[\mathrm{G}^{\mathsf{bad\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{bad}}}(\lambda) \text{ sets } \mathsf{bad} | u = u^*] \\
&= u_\lambda \cdot \sum_{u \in [u_\lambda]} (1/u_\lambda) \Pr[\mathrm{G}^{\mathsf{bad\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{bad}}}(\lambda) \text{ sets } \mathsf{bad} | u = u^*] \\
&= u_\lambda \cdot \mathsf{Adv}^{\mathsf{bad\text{-}ext}}_{\mathsf{SE},\mathcal{A}_{\mathsf{bad}}}(\lambda).
\end{aligned}$$

This complete the proof. ∎