Let Attackers Program Ideal Models: Modularity and Composability for Adaptive Compromise

Joseph Jaeger[0000−0002−4934−3405]

School of Cybersecurity and Privacy
Georgia Institute of Technology, Atlanta, USA
josephjaeger@gatech.edu
https://cc.gatech.edu/~josephjaeger/

Abstract. We show that the adaptive compromise security definitions of Jaeger and Tyagi (Crypto '20) cannot be applied in several natural use-cases. These include proving multi-user security from single-user security, the security of the cascade PRF, and the security of schemes sharing the same ideal primitive. We provide new variants of the definitions and show that they resolve these issues with composition. Extending these definitions to the asymmetric settings, we establish the security of the modular KEM/DEM and Fujisaki-Okamoto approaches to public key encryption in the full adaptive compromise setting. This allows instantiations which are more efficient and standard than prior constructions.

Keywords: Adaptive security · Ideal models · Selective-opening attacks.

1 Introduction

Definitions lie at the heart of modern cryptography. They allow us to mathematically specify what should be achieved by a scheme in practice and give modular, proof-based analyses to ensure these properties are achieved. Studying and understanding definitions is fundamental to the field of cryptography.

There are multiple desiderata to consider when giving a security definition for a primitive including: (i.) Is it philosophically sound? Does it meaningfully model the uses and goals of a primitive in the real world? (ii.) Is it sufficiently strong? Can we prove that this security notion will imply security of higher-level protocols constructed from the primitive? (iii.) Is it sufficiently weak? Can we prove that schemes which "should be" secure satisfy the definition?[1]

In this work, we consider a set of definitions recently introduced by Jaeger and Tyagi [23] for the security of encryption schemes and pseudorandom functions in the "adaptive compromise" setting. They gave several examples of schemes

[1] More nuanced versions of (ii.) and (iii.) ask not just whether these proofs are possible, but also how easy they are to write. Definitions which are difficult to work with can result in proof errors or cryptographers only loosely sketching their proofs (potentially hiding errors).

achieving their definitions as well as higher-level protocols which can be proven secure based on sub-primitives achieving their definitions, thereby evidencing that their definitions achieve desiderata (ii.) and (iii.). We provide counter-evidence. There are natural goals and constructions for which their definitions fail with respect to (ii.) and (iii.).[2]

As an example, it does not seem to be possible to prove that single-user restrictions of their definitions implies the full multi-user versions. Across a wide variety of definitions, the notion of multi-user security that is considered "correct" follows from single-user security by a straightforward hybrid argument. Thus, whether this holds for a definition might be considered a sort of litmus test. A definition for which this is not possible should be examined carefully to understand why. Having done so, we propose new variants of Jaeger and Tyagi's definitions and show that they resolve these shortcomings, while preserving the positive qualities of the original definitions.

1.1 Adaptive Compromise and SIM-AC security

Before discussing our contributions, let us first briefly recall the adaptive compromise setting broadly and the specific SIM-AC definitions of Jaeger and Tyagi (simulation security under adaptive compromise). Roughly speaking, the adaptive compromise setting captures times when there are multiple users of a system, each of whom have their own secrets. An attacker then interacts with these users and based on these interactions may adaptively decide to steal some of the secrets. In applications of these definitions, this description may be somewhat metaphorical. For example, in the searchable encryption scheme of CJJJKR [14] the "users" are keywords, each of which are assigned a secret key. The "stealing" of keys occurs because to perform a search for a particular keyword, the protocol shares the keyword's secret key. The adaptive compromise setting is widely studied in cryptography and is associated with a variety of terms including (but not limited to) adaptive corruption/compromise/security [23,26], non-committing encryption [13,25,10,12], and selective-opening attacks [6,9,7,18,19,21].

Jaeger and Tyagi's work was motivated by various papers that ran into adaptive compromise issues for symmetric encryption or PRFs and had addressed the issues by fixing particular uses of random oracles acting like PRFs. They observed that these works all technically required the same detail-intensive random oracle analysis (which was usually omitted or incorrect). To address this, they introduced their SIM-AC definitions which allow one to abstract away this detail-intensive analysis as something that need only be done once at the lowest levels of analysis. They showed that these notions were achieved by standard efficient schemes in appropriate ideal models, and sufficed for proving the security of their motivating higher-level applications. Broadly their definitions were online-simulator based definitions in which the attacker tries to distinguish between a

[2] The examples for which (iii.) fails are "intermediate-level" proofs where both the assumption and desired result use their definitions. Arguably then, it is only with respect to (ii.) that these definitions have issues.

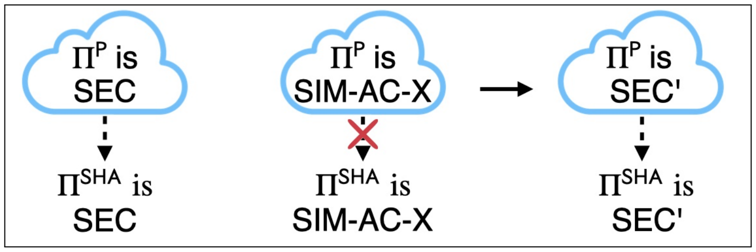

Fig. 1. Left: Typically one proves a scheme Π achieves a security notion with random oracle P, then heuristically assumes it is SEC secure with a particular hash function (e.g. SHA-384). **Middle and Right:** A scheme Π *cannot* be SIM-AC-X secure with any standard model hash function [23,25]. Instead, one uses SIM-AC-X security of Π^{P} as an intermediate step to showing that Π^{P} achieves some security notion SEC$'$. Then one heuristically assumes Π^{SHA} is SEC$'$ secure.

real world where they interact with the honest algorithms of the scheme and an ideal world where the simulator provides responses for every oracle query (including ideal primitive queries). Security requires that for every adversary there is a simulator whose responses it cannot distinguish from the real world.

Notably these definitions were defined explicitly for use only with ideal primitives because techniques of Nielsen [25] show that such definitions cannot be achieved in the standard model. Arguably this causes issues with desiderata (i.). Consider a scheme Π^{H} which expects access to a hash function H. In practice, the might be deployed with the hash function SHA-384 (giving Π^{SHA}) under the hope that it achieves some security notion SEC. Towards justifying this the scheme may be analyzed when the hash function is replaced with a random oracle P (giving Π^{P}). If Π^{P} is shown to be SEC secure, this may be taken as heuristic evidence that Π^{SHA} will be SEC secure. However, this clearly cannot be the case for SEC=SIM-AC-X from the aforementioned result that SIM-AC notions cannot be achieved in the standard model.

From our perspective, the "correct" interpretation is that the SIM-AC definitions are intentionally chosen to be overly strong so that (in ideal models) they imply any other security property SEC$'$ one desires. Suppose SEC$'$ is plausibly achievable in the standard model and one proves that SIM-AC-X security implies SEC$'$ security. Then a proof that Π^{P} is SIM-AC-X secure can be viewed as part of a longer ideal model proof that it achieves SEC$'$. Then the proof can act as as heuristic evidence that Π^{SHA} is a standard model scheme achieving SEC$'$. We represent this pictorially in Fig. 1.

A similar viewpoint can be taken to proving that a particular hash function construction is indifferentiable from a random oracle. It is trivial to show that no standard model hash function can achieve this. However, analyzing indifferentiability in ideal models still serves as a convenient intermediate notion for heuristically justifying the use of the hash function in some contexts.

1.2 Our results

Shortcomings of SIM-AC. After introducing notations and other preliminaries, we start in Section 3 by recalling the original SIM-AC definitions of Jaeger and Tyagi. In their definitions, an attacker interacts with either a real world (where oracles are instantiated honestly) or an ideal world (where oracles are all simulated by a simulator given only some leakage about the queries being made). The definitions are multi-user and allow the attack to ask that a particular users secrets be revealed at any time. Then, in the ideal world, the simulator is given all of the suppressed information about prior queries and must produce a consistent key, lest it be discovered. In the ideal world, the simulator completely controls the responses of the ideal primitive.

We evidence some shortcomings of these definitions, in that they are seemingly unable to prove some very natural results.[3] One example, which came up in their own work, is that their definitions cannot be used for proofs wherein the ideal object is used multiple times within a protocol (whether by multiple different sub-primitives or repeated use of the same sub-primitive). For example, in the searchable encryption construction of CJJJKR [14] the same random oracle was shared across encryption and a PRF, but for the analysis done by Jaeger and Tyagi they were forced to use different primitives for the two uses. One can generically solve this problem via oracle cloning [5], but we find this unsatisfactory. A good definitional framework should allow us to capture when uses of ideal primitives don't require domain separation techniques. Furthermore, while domain separation is relatively fast and efficient for random oracles, we are generally interested in the use of a variety of ideal primitives and it is much less clear how to do oracle cloning efficiently with something like an ideal cipher.

Similar and even more subtle issues arise in some "standard" results that one would expect to hold with a "good" definition. One would expect that it should be possible to prove secure the cascade construction of a PRF [4,17] which iteratively applies a smaller PRF, as well as to prove that for most security notions single-user security implies multi-user security. The cascade construction underlies several other construction PRFs including AMAC, HMAC, and NMAC [2,3,1]. These (and other) issues all stem from a common cause. In SIM-AC, the simulator completely controls and replaces the ideal primitive. As such the definition is not robust to proofs which require multiple different applications of security with respect to the same ideal primitive.

New definition, SIM*-AC. Motivated by these shortcomings, in Section 4 we propose new variants of these definitions, which we term SIM*-AC. Our new definitions match the prior SIM-AC definitions, but make three crucial modifications. The first is that rather having complete control of the ideal primitive, we give the simulator access to an oracle for querying the primitive and which

[3] We use "seemingly" here and similar phrasing elsewhere because, while we have deeply considered these problems and do not see how SIM-AC could be used to prove these results, we do not have any explicit counterexamples showing it is impossible.

additionally provides the special power of being able to give an input-output pair which the primitive will program itself to be consistent with, if possible. This modification means that applications of SIM*-AC in a proof will leave the ideal primitive around for use in further proof steps. However, these future steps can run into issues where the simulator is supposed to have programmed the ideal primitive, but a reduction attacker who wants to run the simulator internally has no way of forcing other parties to use a programmed ideal primitive. This issue is resolved by our second modification which *gives the adversary the ability to program the ideal primitive*. The final modification is aimed at proofs which require a polynomial number of hybrids and, as such, the reduction adversary needs to depend on the simulator so that it can properly simulate internal hybrids. We simply reverse the order of quantification so that a universal simulator is quantified before a specific attacker.

After the introduction of the new definitions we show by example that the modifications suffice to write the proofs we identified as seemingly not possible with the original SIM-AC definitions. Namely, we prove that for all of our SIM*-AC definitions (with one exception) single-user security implies multi-user security[4] and that the cascade construction of a large-domain PRF from a small-domain PRF is secure. Both proofs are hybrid arguments which conceptually resemble such proofs for most standard indistinguishability-based security notions. For going from single-user to multi-user the hybrid is over how many of the users will be honestly run versus emulated by a copy of the single-user simulator. For the cascade construction (which is a generalization of the GGM construction of a PRF from a PRG), we think of there being an underlying tree structure imposed on the internal values of the computation. The proof performs a hybrid over how many layers of the tree are honestly run versus emulated by a multi-user simulator for the underlying PRF. Using multi-user security allows us to hybrid one layer at a time, rather than having hybrid over each node individually.

Asymmetric encryption. The SIM-AC definitions focus on symmetric primitives (encryption and PRFs) because this is what was required by their applications. However, adaptive compromise has been studied in detail for public-key encryption, so it is natural to ask how a SIM*-AC notion for public key encryption would work. We do so in Section 5, providing a definition that captures the compromise of receiver secret decryption keys and sender randomness. The resulting definition roughly matches the SIM-FULL definition of Camensich, Lehmann, Neven, and Samelin [12].[5] In their work, they showed that SIM-FULL was stronger than various prior adaptive compromise definitions [11,18] and equivalent to a new universal composability definition they introduce.

Casting this definition in SIM*-AC language provides benefits. Where CLNS constructed one particular secure encryption scheme from one-way trapdoor permutations, the broader context of SIM*-AC style definitions allows us to fol-

[4] The exception is key-private security which is meaningless with only a single user.
[5] Their definition is basically a SIM-AC-CCA (not SIM*-AC) definition with labels and using a random oracle.

low the example of Jaeger and Tyagi by giving modular analysis. In particular, we introduce SIM*-AC definitions for key-encapsulation mechanisms (KEM), then show the KEM/DEM approach [15] allows one to combine a KEM with a symmetric encryption scheme to construct public-key encryption. We consider one version of the Fujisaki-Okamoto transformation [16] (as modularized by Hofheinz, Hövelmanns, and Kiltz [20]) to show that it can lift a KEM satisfying a one-wayness security notion to a KEM satisfying our full SIM*-AC-CCA notion. Thereby we have a more general collection of different options how to construct a public-key encryption scheme secure against adaptive compromise. We can instantiate this with well-studied and standardized schemes, improving efficiency because our analysis allows the use of block-cipher based symmetric encryption for the DEM.

An interesting comparison point for our KEM/DEM analysis is the work of Heuer and Poettering [19] who also looked at the KEM/DEM construction. They proved a weaker offline-simulation notion of security for public key encryption by making a particular concrete assumption about the DEM being constructed from a blockcipher and having to have a particular simulatable form.

New definition, old results. Jaeger and Tyagi showed a number positive results in their original work. These include that random oracles and ideal ciphers make SIM-AC-PRF secure function families, that various constructions of symmetric encryption achieve SIM-AC security when their underlying function families are SIM-AC-PRF secure, and that higher-level protocols can be proven secure assuming the SIM-AC security of their constituent elements. It would be rather disappointing if our switch to SIM*-AC security required us to re-prove all of these results from scratch.

In Section 6, we dedicate the end of our paper to showing that these results hold with SIM*-AC security. We roughly divide these pre-existing results into three categories: low-level results (constructing basic SIM-AC primitives directly from ideal primitives), intermediate-level results (using one notion of SIM-AC to achieve another), and high-level results (proving secure some non-SIM-AC protocol). For each we discuss how the existing result can be seen, possibly with minor modification to the proof, to hold for SIM*-AC security. In some cases we can get minor improvements along the way, such as allowing the proof to handle when a single ideal primitive is shared between multiple schemes.

2 Preliminaries

Pseudocode notation. We define security notions using pseudocode-based games. The pseudocode "Require bool" is shorthand for "If not bool then return $\bot$". If S is a set, then $x \leftarrow\$ S$ sets x equal to a uniformly random element of S. The notation $x_{(\cdot)} \leftarrow\$ S$ means that each x_u will be sampled according to $x_u \leftarrow\$ S$ the first time it is accessed.

The notation $y \leftarrow\$ A(x_1, x_2, \cdots : \sigma)$ denotes the (randomized) execution of A with state σ. Deterministic execution uses $\leftarrow$. The state σ is passed by reference, so changes that A makes to σ are maintained after A's execution. All other inputs are passed by value. 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.

The symbol $\perp$ is used to indicate rejection or uninitialized variables. The symbol $\diamond$ is used as a return value by functions that do not need to return anything. Unless specified otherwise, these values are assumed not to be contained in sets. Algorithms and oracles will typically assume their input is from a particular domain (e.g. the message space of an encryption scheme). We implicitly assume adversaries never provide them with input not in these domains.

A list T of length $n \in \mathbb{N}$ specifies an ordered sequence of elements $T[1]$, $T[2]$, $\ldots$, $T[n]$. The operation $T.\mathsf{add}(x)$ appends x to this list by setting $T[n+1] \leftarrow x$, so T is now of length $n+1$. We let $|T|$ denote the length of 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. For example, $x \in T$ is true if $x = T[i]$ for any $1 \leq i \leq |T|$. The loop "For $x \in T$" is defined to be looping "For $i = 1, \ldots, |T|$" and defining $x \leftarrow T[i]$ in each iteration.

If T is a list of tuples (x, y) then we index into T like a table where $T\langle x\rangle$ is the y value of the last tuple in the list with first component x (or is $\perp$ if no such tuple exists). By $T.\mathsf{add}(x, y)$ we mean $T.\mathsf{add}((x, y))$.

We use an asymptotic formalism with security parameter λ. A function f is negligible if for all polynomials p there exists a $\lambda_p \in \mathbb{N}$ such that $f(\lambda) \leq 1/p(\lambda)$ for all $\lambda \geq \lambda_p$. We say it is super-polynomial if $1/f$ is negligible and super-logarithmic if 2^f is super-polynomial.

Suppose $\mathrm{G}^{\mathsf{sec}}_x$ is a game that samples a uniformly random bit b, runs an adversary which guesses bit b', and then returns the boolean $(b = b')$. Then for $d \in \{0, 1\}$, we let $\mathrm{G}^{\mathsf{sec}}_{x,d}$ be the game with b hardcoded to have value d and which outputs the boolean $(b' = 1)$. Standard conditional probability calculations give that $2\Pr[\mathrm{G}^{\mathsf{sec}}_x] - 1 = \Pr[\mathrm{G}^{\mathsf{sec}}_{x,1}] - \Pr[\mathrm{G}^{\mathsf{sec}}_{x,0}]$.

Ideal primitives. Most of the definitions we consider are dependent on ideal primitives such as random oracles or ideal ciphers, so we require a careful formalization of them. An ideal primitive P specifies (for each $\lambda \in \mathbb{N}$) a distibution $\mathcal{P}_\lambda$ over functions $f : \mathcal{K}_\lambda \times \mathcal{D}_\lambda \to \mathcal{R}_\lambda$. When needed to avoid ambiguity we write $\mathsf{P}.\mathcal{P}_\lambda$, $\mathsf{P}.\mathcal{K}_\lambda$, $\mathsf{P}.\mathcal{D}_\lambda$, and $\mathsf{P}.\mathcal{R}_\lambda$. In the P ideal model, $f \leftarrow\$\ \mathcal{P}_\lambda$ is sampled at the beginning of any security game and algorithms are given oracle access to f.

It is often important that oracle access to an ideal primitive can be efficiently simulated despite the fact that each $f \in \mathcal{P}_\lambda$ is typically exponential in size. This is referred to as lazy sampling, which we notate using an algorithm $\mathsf{P.Ls}$. We will think of f as being (partially) specified by a table σ_{P} indexed by $\mathcal{K}_\lambda \times \mathcal{D}_\lambda$. Then the evaluation algorithm has syntax $y \leftarrow\$\ \mathsf{P.Ls}(1^\lambda, k, x : \sigma_{\mathsf{P}})$. If $\sigma_{\mathsf{P}}[k, x] = \perp$, it samples $\sigma_{\mathsf{P}}[k, x]$ according to the appropriate distribution conditioned on the current value of σ_{P}.[6] Then it outputs $\sigma_{\mathsf{P}}[k, x]$. We sometimes use A^{P} as shorthand for giving algorithm A oracle access to $\mathsf{P.Ls}(1^\lambda, \cdot, \cdot : \sigma_{\mathsf{P}})$.

[6] Concretely, this is the distribution induced by sampling $f \leftarrow\$\ \mathcal{P}_\lambda$ subject to $f(k', x') = \sigma[k', x']$ wherever the latter is not $\perp$ and assigning $\sigma_{\mathsf{P}}[k, x] \leftarrow f(k, x)$.

The standard model is captured by the primitive P_{sm} for which $\mathcal{P}_\lambda$ always returns the function f defined exactly by $f(\varepsilon,\varepsilon)=\varepsilon$. A random oracle $\mathsf{P}_{\mathsf{rom}}$ is captured by $\mathcal{P}_\lambda$'s output being uniform over the set of all functions $f:\mathcal{K}_\lambda\times\mathcal{D}_\lambda\to\mathcal{R}_\lambda$. An ideal injection $\mathsf{P}_{\mathsf{inj}}$ is captured by letting $\mathcal{K}_\lambda$ consist of tuples $(\circ,k)$ for $\circ\in\{+,-\}$. Then $\mathcal{P}_\lambda$ returns a uniform f for which $f((+,k),\cdot)$ is an injection with inverse $f((-,k),\cdot)$ (we define inverse functions to output $\diamond$ on input a value not in the image of the original function). An ideal cipher $\mathsf{P}_{\mathsf{icm}}$ is an ideal injection for which $f((+,k),\cdot)$ is a bijection on the finite set $\mathcal{D}_\lambda=\mathcal{R}_\lambda$. Standard techniques allow Ls to be efficiently evaluated for such functions.

Cryptographic schemes may be constructed from multiple underlying cryptographic schemes, each expecting its own ideal primitive. Let P' and P'' be ideal primitives. We define $\mathsf{P}=\mathsf{P}'\times\mathsf{P}''$ via the following algorithms.

```
$\underline{\mathsf{P.Init}(1^\lambda)}$
$\sigma_{\mathsf{P}}' \leftarrow\$ \mathsf{P}'.\mathsf{Init}(1^\lambda)$
$\sigma_{\mathsf{P}}'' \leftarrow\$ \mathsf{P}''.\mathsf{Init}(1^\lambda)$
Return $(\sigma_{\mathsf{P}}',\sigma_{\mathsf{P}}'')$

$\underline{\mathsf{P.Ls}(1^\lambda,k,x:\sigma_{\mathsf{P}})}$
$(\sigma_{\mathsf{P}}',\sigma_{\mathsf{P}}'') \leftarrow \sigma_{\mathsf{P}}$
$(d,k) \leftarrow k$
If $d=1$ then $y \leftarrow\$ \mathsf{P}'.\mathsf{Ls}(1^\lambda,k,x:\sigma_{\mathsf{P}}')$
If $d=2$ then $y \leftarrow\$ \mathsf{P}''.\mathsf{Ls}(1^\lambda,k,x:\sigma_{\mathsf{P}}'')$
$\sigma_{\mathsf{P}} \leftarrow (\sigma_{\mathsf{P}}',\sigma_{\mathsf{P}}'')$
Return $y$
```

In other words, $\mathsf{P}.\mathcal{P}_\lambda$ samples $f'\leftarrow\$\,\mathsf{P}'.\mathcal{P}_\lambda$ and $f''\leftarrow\$\,\mathsf{P}''.\mathcal{P}_\lambda$, then defines f by $f((1,k),x)=f'(k,x)$ and $f((2,k),x)=f''(k,x)$.

Programming ideal primitives. For our new security notions we need to make explicit a notion of "programming" an ideal model. By this we mean allowing some third party to define the output of ideal model on inputs that have not previously been queried. Let σ_{P} be a table indexed by $\mathcal{K}_\lambda\times\mathcal{D}_\lambda$ and let $(k,x,y)\in\mathcal{K}_\lambda\times\mathcal{D}_\lambda\times\mathcal{R}_\lambda$. We say that σ_{P} is compatible with (k,x,y), denoted $\sigma_{\mathsf{P}}\heartsuit(k,x,y)$ if there exists $f\in\mathcal{P}_\lambda$ such that (i) $\sigma_{\mathsf{P}}[k',x']=f(k',x')$ wherever $\sigma_{\mathsf{P}}[k',x']\neq\bot$ and (ii) $f(k,x)=y$. Then we allow programming of an ideal model P using the algorithm $\mathsf{P.Prog}$ defined as follows.

```
$\underline{\mathsf{P.Prog}(1^\lambda,k,x,y:\sigma_{\mathsf{P}})}$
If $\sigma_{\mathsf{P}}\heartsuit(k,x,y)$ then $\sigma_{\mathsf{P}}[k,x] \leftarrow y$
Return $\diamond$
```

This ensures that P cannot be redefined on an input where it was already defined and that an ideal injection cannot be made to have inconsistent inverses.

Our careful formalizing of ideal primitives in terms of functions, particularly in requiring that $\mathsf{P.Prog}$ maintain consistency, is important for avoiding subtle issues in later proofs. This formalization ensures that a deterministic algorithm with oracle access to P always gives consistent outputs even if P is programmed between executions. Correctness of a scheme with access to P (e.g. that decryption inverts encryption) is maintained even if P is programmed between executions of different algorithms. Without these properties it would be difficult to avoid erroneous proofs that implicitly assumed them during typically "straightforward" proof steps.

This is not without cost. The requirement for consistency in programming has the potential to introduce subtle errors elsewhere in proofs by implicitly assuming an attempt to program an oracle worked, when in fact it failed because of inconsistency. Additionally, the act of honestly querying the ideal primitive can be detected by a programming adversary who attempts to program at that point and then checks if they succeed in this programming. We believe this cost to be worthwhile because in the analyses we have considered, the places that could cause such proof errors would anyway need to be analyzed carefully to avoid other errors if we were using a more permission notion of programming.

For generality, we allow the use of non-programmable ideal primitives in games that allow programming. This is captured by defining $\mathsf{P.Prog}$ to immediately return $\diamond$. When we quantify over an arbitrary ideal primitive, we allow it to be programmable or non-programmable (or the combination of multiple ideal primitives – some programmable, some not). When we discuss a specific ideal primitive, we mean the programmable version unless specified otherwise.

Syntax for cryptographic primitives. We assume familiarity with (randomized) symmetric encryption, asymmetric encryption, function families (e.g. PRFs), and key encapsulation mechanisms. We use the following syntax.

Symmetric encryption	Asymmetric encryption
$k \leftarrow_{\$} \mathsf{SE.Kg}(1^\lambda)$	$(ek, dk) \leftarrow_{\$} \mathsf{PKE.Kg}(1^\lambda)$
$c \leftarrow_{\$} \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, k, m)$	$c \leftarrow_{\$} \mathsf{PKE.Enc}^{\mathsf{P}}(1^\lambda, ek, m)$
$m \leftarrow \mathsf{SE.Dec}^{\mathsf{P}}(1^\lambda, k, c)$	$m \leftarrow \mathsf{PKE.Dec}^{\mathsf{P}}(1^\lambda, dk, c)$

Function Family	Key Encapsulation Mechanism
$k \leftarrow_{\$} \mathsf{F.Kg}^{\mathsf{P}}(1^\lambda)$	$(ek, dk) \leftarrow_{\$} \mathsf{KEM.Kg}(1^\lambda)$
$y \leftarrow \mathsf{F.Ev}^{\mathsf{P}}(1^\lambda, k, x)$	$(c, k) \leftarrow_{\$} \mathsf{KEM.Encaps}^{\mathsf{P}}(1^\lambda, ek)$
$x \leftarrow \mathsf{F.Inv}^{\mathsf{P}}(1^\lambda, k, y)$	$k \leftarrow \mathsf{KEM.Decaps}^{\mathsf{P}}(1^\lambda, dk, c)$

A family of functions F only has inverse algorithm $\mathsf{F.Inv}$ if it is a blockcipher. For simplicity, we assume perfect correctness which holds for all $f \in \mathsf{P}.\mathcal{P}_\lambda$. We will make careful note of where proofs make use of this correctness. To use notions of imperfect correctness in these proofs, one must choose an imperfect correctness notion that is "robust" to the ideal primitive being programmable.

We additionally will sometimes assume a notion we call *query consistency* which requires that if c is produced by encryption/encapsulation, then decrypting/decapsulating c with the correct key only makes ideal primitive queries that were also made by encryption/encapsulation. This ensures that any querying of the ideal primitive while decrypting/decapsulating an honest ciphertext cannot be detected by a programming adversary.

3 SIM-AC Definitions and Their Shortcomings

We start by recalling the definitions that Jaeger and Tyagi [23] introduced for the simulation security of symmetric encryption or pseudorandom functions under adaptive compromise. Jaeger and Tyagi showed that these definition were achieved by very natural encryption/PRF constructions in the random oracle or ideal cipher model and that they moreover sufficed for proving the security of higher-level constructions (e.g. searchable encryption schemes, asymmetric password-authenticated key exchange, and self-revocable encrypted cloud storage). In this section, we will identify ways in which these definitions fall short. Namely, that there are other natural encryption/PRF constructions and high-level construction which cannot be proven secure using these definitions.[7]

3.1 SIM-AC Definitions

All of the SIM-AC definitions have a common structure; they measure the ability of an adversary to distinguish between a “real” and a “simulated” world. In the real world, the adversary interacts with multiple “users” that honestly execute the algorithms of scheme. The adversary has access to an exposure oracle which it can query to be given the secret keys of any users it chooses. Finally, the adversary has oracle access to the ideal primitive algorithm $\mathsf{P.Ls}$. In the ideal world, the output of *all* of these oracles is provided instead by a simulator S. For the definition to be meaningful, the behavior of the simulator when responding to queries for “unexposed” users is restricted in some manner. (For example, the simulator may be required to return a uniformly random string or may only be given partial information about what the query was.)

Pseudorandom function security. We start with the notion of SIM-AC-PRF security for a function family F. It is captured by the game $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}$ shown in Fig. 2. The variable X is used to track which users have been exposed, so X_u is true when the user has been exposed. The game hardcodes that random values are returned for evaluation queries to unexposed users in the simulated world. Inputs and outputs to evaluation are stored in the table T_u which is given to S when u is exposed.

We define $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda)] - 1$ and say that 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. Intuitively, this definition captures that the outputs of F_k look random to an adversary until they expose k.

Encryption definitions. Next we recall the SIM-AC security notions for a symmetric encryption scheme SE. Consider the game $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda)$ shown in Fig. 2. During encryption queries for unexposed users, the simulator is only told

[7] Technically, we do not show that these proofs are impossible. We show why the “natural” proofs fail and informally argue why it seems difficult to find other proofs.

```
Game $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda)$
    $k_{(\cdot)} \leftarrow\!\$\ \mathsf{F.Kg}(1^\lambda)$
    $\sigma_{\mathsf{P}} \leftarrow\!\$\ \mathsf{P.Init}(1^\lambda)$
    $\sigma \leftarrow\!\$\ \mathsf{S.Init}(1^\lambda)$
    $b \leftarrow\!\$\ \{0,1\}$
    $b' \leftarrow\!\$\ \mathcal{A}_{\mathsf{prf}}^{\textsc{Ev},\textsc{Exp},\textsc{Prim}}(1^\lambda)$
    Return $(b = b')$

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

$\textsc{Ev}(u, x)$
    If $T_u[x] \neq \bot$ then return $T_u[x]$
    $y_1 \leftarrow \mathsf{F.Ev}^{\mathsf{P}}(1^\lambda, k_u, x)$
    If $X_u$ then $y_0 \leftarrow\!\$\ \mathsf{S.Ev}(1^\lambda, u, x : \sigma)$
    Else $y_0 \leftarrow\!\$\ \mathsf{F.Out}(\lambda)$
    $T_u[x] \leftarrow y_b$
    Return $y_b$

$\textsc{Exp}(u)$
    $k'_1 \leftarrow k_u$
    $k'_0 \leftarrow\!\$\ \mathsf{S.Exp}(1^\lambda, u, T_u : \sigma)$
    $X_u \leftarrow \mathsf{true}$
    Return $k'_b$
```

```
Game $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda)$
    $k_{(\cdot)} \leftarrow\!\$\ \mathsf{SE.Kg}(1^\lambda)$
    $\sigma_{\mathsf{P}} \leftarrow\!\$\ \mathsf{P.Init}(1^\lambda)$
    $\sigma \leftarrow\!\$\ \mathsf{S.Init}(1^\lambda)$
    $b \leftarrow\!\$\ \{0,1\}$
    $b' \leftarrow\!\$\ \mathcal{A}_{\mathsf{cca}}^{\textsc{Enc},\textsc{Dec},\textsc{Exp},\textsc{Prim}}(1^\lambda)$
    Return $(b = b')$

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

$\textsc{Enc}(u, m)$
    If not $X_u$ then $\ell \leftarrow |m|$ else $\ell \leftarrow m$
    $c_1 \leftarrow\!\$\ \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, k_u, m)$
    $c_0 \leftarrow\!\$\ \mathsf{S.Enc}(1^\lambda, u, \ell : \sigma)$
    $M_u.\mathsf{add}(c_b, m)$; Return $c_b$

$\textsc{Dec}(u, c)$
    If $M_u\langle c\rangle \neq \bot$ then return $M_u\langle c\rangle$
    $m_1 \leftarrow \mathsf{SE.Dec}^{\mathsf{P}}(1^\lambda, k_u, c)$
    $m_0 \leftarrow\!\$\ \mathsf{S.Dec}(1^\lambda, u, c : \sigma)$
    Return $m_b$

$\textsc{Exp}(u)$
    $k'_1 \leftarrow k_u$; $k'_0 \leftarrow\!\$\ \mathsf{S.Exp}(1^\lambda, u, M_u : \sigma)$
    $X_u \leftarrow \mathsf{true}$; Return $k'_b$
```

Fig. 2. Games defining SIM-AC-PRF security of F and SIM-AC-CCA security of SE.

the length of the message m. The list M_u stores the messages queried to user u and ciphertexts returned. It is given to the simulator when that user is exposed. If the attacker forwards challenge ciphertexts from encryption to decryption, this list is used to respond appropriately.

We define $\mathsf{Adv}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda)] - 1$ and 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. Intuitively, this definition captures that an adversary learns nothing (other than the length) about a message m encrypted with a key k until they expose k. For chosen-plaintext security we restrict attention to attackers that never query decryption. We then write the superscript sim-ac-cpa.

Stronger notions of security are captured by requiring that S be chosen from some restricted set. Key-private security (SIM-AC-KP) requires that the CPA simulator respond to encryption queries for un-exposed users using an algorithm $\mathsf{S.Enc}_1(1^\lambda, \ell : \sigma)$ which *is not* given u as input. Indistinguishable from random security (SIM-AC-\$) requires that the CPA simulator respond to encryption queries for un-exposed users by sampling c from a set $\mathsf{S.Out}(\lambda, \ell)$. Authenticated encryption security (SIM-AC-AE) requires that the CCA simulator respond to encryption queries as in SIM-AC-\$ security and to decryption queries for unexposed users with $\perp$.

Simplifying assumptions. Jaeger and Tyagi observed the following simplifying assumptions (copied almost verbatim from [23]) for their SIM-AC definitions.

- If an oracle is deterministic in the real world 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 $\mathsf{S.Ls}$).
- We can assume the adversary always queries with $u \in [u_\lambda] = \{1, 2, \ldots, u_\lambda\}$ 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 will be efficiently computable given the transcripts of queries the adversary has made.)

Looking ahead, we will be able to make the analogous assumptions for the new definitions introduced in this paper. These assumptions are convenient for proving that a scheme satisfies a given SIM-AC definition of security. The fact that these assumptions are not hardcoded into the security game is convenient when proving the security of a higher-level construction assuming that constituent schemes satisfy some SIM-AC security notion.

3.2 Shortcomings of SIM-AC

Now that we have introduced SIM-AC security notions we can discuss ways that they fall short of being able to establish the results we would like.

Multiple schemes with the same P. Suppose a higher-level protocol is constructed from multiple underlying schemes satisfying SIM-AC security notions. We generally will not be able to prove the security of the protocol if the underlying schemes make use of the same P.[8] Performing a SIM-AC reduction with the first scheme will replace the entirety of P with some $\mathsf{S.Ls}$. With P being gone, the security of the second scheme with respect to P is of no use.

As a toy example, we might consider function families F_0 and F_1. Even assuming they are both SIM-AC-PRF secure with P, it seems impossible to prove F is SIM-AC-PRF secure where $\mathsf{F.Ev}^{\mathsf{P}}(1^\lambda, (k_0, k_1), (b, x)) = \mathsf{F}_b.\mathsf{Ev}^{\mathsf{P}}(1^\lambda, k_b, x)$. Several of Jaeger and Tyagi's proofs were restricted by this and had to assume underlying schemes used distinct ideal primitives.

[8] Note this is the more general result, as we could let $\mathsf{P} = \mathsf{P}_1 \times \mathsf{P}_2 \times \ldots$ and have the i-th scheme using P only actually query P_i.

Multiple uses of the same scheme. Suppose a higher-level protocol is constructed from an underlying scheme satisfying a SIM-AC security notion and that this scheme is used in several distinct ways in the protocol.

If it's not possible to write a careful reduction that covers all of the uses of the scheme at once, then we run into a similar issue as the above. The first application of the scheme's SIM-AC security will replace its ideal primitive with a simulator, preventing us from applying its security again.

As a toy example, we might consider a function family F. Even assuming F is SIM-AC-PRF secure with P, it seems impossible to prove that F' is SIM-AC-PRF secure where $\mathsf{F}'.\mathsf{Ev}^{\mathsf{P}}(1^\lambda, k, (x_0, x_1)) = \mathsf{F}.\mathsf{Ev}^{\mathsf{P}}(1^\lambda, \mathsf{F}.\mathsf{Ev}^{\mathsf{P}}(1^\lambda, k, x_0), x_1)$.

One of Jaeger and Tyagi's proofs (for their Theorem D.1) almost ran into issue with this. However, they seemingly got "lucky" in that for that particular proof they were able to use just plain PRF security for the first use of the underlying function family.

Single-user security implies multi-user security. With most "standard" security notions (e.g. PRF, IND-CPA, IND-CCA) single-user security implies multi-user security. These results are proven by a "hybrid proof" wherein the single-user attacker picks a user u at random. It externally simulates u with its own oracle, internally simulates all "prior" users as in the $b = 0$ world, and internally simulates all "later" users as in the $b = 1$ world.

We run into issue if we try to write an analogous proof for SIM-AC definitions. Note that simulating the $b = 0$ world for some users requires the attacker to run the given single-user simulator. This creates a circular dependency as in SIM-AC the simulator is allowed to depend on the adversary.

Even if we changed the order of quantification, we would still run into issues. Each instance of the single-user simulator expects to already have complete control of the ideal primitive. This makes it unclear what ideal primitive oracle the single-user adversary should provide the multi-user adversary it runs internally. Because of these issues, Jaeger and Tyagi directly consider multi-user SIM-AC definitions and do not discuss single-user variants thereof.

It may seem strange to consider "adaptive compromise" in a single-user setting. Do expose queries make sense where there is only one user to be exposed? It is useful to first observe that multi-user SIM-AC notions would be unchanged if we required that the attacker expose all users before halting. Crucially, these definitions use "online" simulators that are forced to commit to simulated ciphertexts (without knowledge of the encrypted message) for users that will later be exposed (at which time the simulator is told the messages).

4 SIM*-AC Security

We saw in the previous section some ways in which SIM-AC security definitions cannot be used for proving results which intuitively "should" be possible to prove with a "good" security defintion. In this section, we will introduce a related class of security definitions which we notate by SIM*-AC. These new definition will strengthen the power of the attacker and weaken the power of the simulator. This allows proving the results that were a challenge for the prior definitions, while still maintaining the value of the prior definitions. In particular, the results previously shown by Jaeger and Tyagi with SIM-AC can be shown to hold with SIM*-AC, while requiring minimal modifications to the proofs. We discuss the details of this in Section 6.

Motivating the new definition. The starting place for our new definitions partially goes back to the original explicit proposal of random oracles by Bellare and Rogaway [8]. Therein, their definition of zero knowledge in the random oracle model requires that the (offline) simulator's final outputs includes the list of points at which it would like the random oracle to have given values. At all other points, the oracle is sampled at random. Wee [28] built on this, considering different levels of how the simulator controls the random oracle and showing that zero-knowledge proofs are closed under sequential composition when the random oracle is explicitly programmable (or non-programmable). Sequential composition fails in the "fully programmable" model as applying the simulator for the first round of execution replaces the random oracle completely, at which point we cannot use it to reason about further rounds.

There is a second subtle detail allowing sequential composition proof to go though with polynomially many rounds. It is important that (part of) the adversary was quantified *after* the simulator. The proof followed a hybrid argument wherein rounds of zero knowledge are switched from real to simulated, one at a time. To apply security for a particular round, the attacker must simulate the other (real and simulated) rounds. For a constant number of rounds, we could fix the attacker for the first round, be given its simulator, use the simulator in the attacker for the second round, be given its simulator, and so on. When the number of rounds is polynomial, we cannot fix an attacker for each round. Instead a single attacker must work for all rounds, which requires knowing the simulator ahead of time so it can properly emulate simulated rounds.

To resolve the issues identified with composition and hybrid arguments for SIM-AC we will restrict the simulator to explicitly program the ideal primitive and require a universal simulator that works for all attackers. However, this still is not enough! The zero knowledge composition discussed above is importantly "sequential" in an "offline simulation" setting. The simulator runs once in isolation, then provides its output to the attacker which runs in isolation. The attacker has complete control over all code executing with it, so can perfectly emulate the programmed random oracle. In an "online simulation" setting like SIM-AC, the attacker runs in parallel with the honest scheme algorithms or the simulator. Our proofs would run into issues when attackers internally run copies of the simulator which wants to program the random oracle, but the attacker is then unable to force the honest scheme algorithms or simulator it does not control to use this modified random oracle. We resolve this issue by expanding the power of the adversary and giving it the capacity to program the ideal primitive.[9] We use the prefix SIM*-AC for the definitions we write in this style.

Summarizing, in our SIM*-AC definitions simulators and adversaries can access an oracle PPrim which allows them to evaluate *or explicitly program* the ideal primitive. Schemes are still restricted to not program the ideal primitive. This is a restriction on the simulator and strengthening of the attacker. Because of the programmability of P we must write the code so that S is only run in the ideal world and SE is only run in the real world.

Comparisons to prior definitions. Through this sequence of ideas we have reached the same general structure of random oracle modeling proposed by Camenisch, Drijvers, Gagliardoni, Lehmann, and Neven [10]. Their work is in the universal composability (UC) setting where they consider several models for global random oracles. In one, simulators and adversaries can explicitly program the random oracle. They show it allows security proofs that very efficient and natural random oracle-based constructions of several primitives satisfy the desired security. Our work generalizes this any ideal primitive (not just random oracles) and considers its application outside the universal composability framework. That UC and SIM-AC work well with a similar programability notion is, in hindsight, natural as they both consider *online* simulation.

Our SIM*-AC definitions as not strictly better for cryptographers than the SIM-AC definitions of Jaeger and Tyagi [23]. One benefit of their work was the ease with which existing results could be ported to the SIM-AC setting (e.g. replacing IND-CPA in a proof with SIM-AC-CPA). This holds to some extent with the new SIM*-AC definitions as well, but proofs do occasionally run into additional difficulties because of fragilities caused by the programming of the oracle. Overall we believe that this cost is worth the benefits provided by our new definitions being able to show natural and desirable results that are seemingly out of reach of plain SIM-AC.

High-level remarks There is value in incorporating this explicit programming capacity for adversaries even into non-simulation definitions. Consider the construction of some high-level system making use of multiple underlying schemes that use the same ideal primitive, some for SIM*-AC security and some for non-simulation security notions. (See, e.g., the searchable encryption proof in [23] that involved the standard notion of PRF security in addition to SIM-AC-PRF/KPA security). If the proof requires use of the non-simulation security notion *after* a SIM*-AC notion has already been applied, this will only be possible if the attacker can program the ideal primitive in the non-simulation notion.

Allowing the adversary to program the ideal primitive is *strange*. It does not seem to capture anything about reality, despite the fact that we allow the adversary to do this programming even in the “real world”. However, this ability

[9] Wee would have run into similar issues had their hybrid tried to switch rounds to simulated from first to last, rather than the last to first approach they took.

```
Game $\mathrm{G}^{\mathsf{sim^*\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}(\lambda)$
$k_{(\cdot)} \leftarrow\!\$\ \mathsf{F.Kg}(1^\lambda)$
$\sigma_{\mathsf{P}} \leftarrow\!\$\ \mathsf{P.Init}(1^\lambda)$
$\sigma \leftarrow\!\$\ \mathsf{S.Init}(1^\lambda)$
$b \leftarrow\!\$\ \{0,1\}$
$b' \leftarrow\!\$\ \mathcal{A}_{\mathsf{prf}}^{\textsc{Ev},\textsc{Exp},\textsc{PPrim}}(1^\lambda)$
Return $(b = b')$

$\textsc{PPrim}(\mathsf{Op}, k, x, y)$
Require $\mathsf{Op} \in \{\mathsf{Ls}, \mathsf{Prog}\}$
$y \leftarrow\!\$\ \mathsf{P.Op}(1^\lambda, k, x, y : \sigma_{\mathsf{P}})$
Return $y$

$\textsc{Ev}(u, x)$
If $T_u[x] \neq \bot$ then return $T_u[x]$
If $b = 1$ then $y \leftarrow \mathsf{F.Ev}^{\mathsf{P}}(1^\lambda, k_u, x)$
If $b = 0$ then
    If $X_u$ then $y \leftarrow \mathsf{S.Ev}^{\textsc{PPrim}}(1^\lambda, u, x : \sigma)$
    Else $y \leftarrow\!\$\ \mathsf{F.Out}(\lambda)$
$T_u[x] \leftarrow y$
Return $y$

$\textsc{Exp}(u)$
If $b = 1$ then $k' \leftarrow k_u$
If $b = 0$ then $k' \leftarrow\!\$\ \mathsf{S.Exp}^{\textsc{PPrim}}(1^\lambda, u, T_u : \sigma)$
$X_u \leftarrow \mathsf{true}$; Return $k'$
```

```
Game $\mathrm{G}^{\mathsf{sim^*\text{-}ac\text{-}cca}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda)$
$k_{(\cdot)} \leftarrow\!\$\ \mathsf{SE.Kg}(1^\lambda)$
$\sigma_{\mathsf{P}} \leftarrow\!\$\ \mathsf{P.Init}(1^\lambda)$
$\sigma \leftarrow\!\$\ \mathsf{S.Init}(1^\lambda)$
$b \leftarrow\!\$\ \{0,1\}$
$b' \leftarrow\!\$\ \mathcal{A}_{\mathsf{cca}}^{\textsc{Enc},\textsc{Dec},\textsc{Exp},\textsc{PPrim}}(1^\lambda)$
Return $(b = b')$

$\textsc{PPrim}(\mathsf{Op}, k, x, y)$
Require $\mathsf{Op} \in \{\mathsf{Ls}, \mathsf{Prog}\}$
$y \leftarrow\!\$\ \mathsf{P.Op}(1^\lambda, k, x, y : \sigma_{\mathsf{P}})$
Return $y$

$\textsc{Enc}(u, m)$
If not $X_u$ then $\ell \leftarrow |m|$ else $\ell \leftarrow m$
If $b = 1$ then $c \leftarrow\!\$\ \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, k_u, m)$
If $b = 0$ then $c \leftarrow\!\$\ \mathsf{S.Enc}^{\textsc{PPrim}}(1^\lambda, u, \ell : \sigma)$
$M_u.\mathsf{add}(c, m)$; Return $c$

$\textsc{Dec}(u, c)$
If $M_u\langle c\rangle \neq \bot$ then return $M_u\langle c\rangle$
If $b = 1$ then $m \leftarrow \mathsf{SE.Dec}^{\mathsf{P}}(1^\lambda, k_u, c)$
If $b = 0$ then $m \leftarrow\!\$\ \mathsf{S.Dec}^{\textsc{PPrim}}(1^\lambda, u, c : \sigma)$
Return $m$

$\textsc{Exp}(u)$
If $b = 1$ then $k' \leftarrow k_u$
If $b = 0$ then $k' \leftarrow\!\$\ \mathsf{S.Exp}^{\textsc{PPrim}}(1^\lambda, u, M_u : \sigma)$
$X_u \leftarrow \mathsf{true}$; Return $k'$
```

Fig. 3. Games defining SIM*-AC-PRF security of F and SIM*-AC-CCA security of SE. We use highlighting to indicate where the definitions differ from SIM-AC versions.

will be crucial to how we can use this new definition to prove the results that we were unable to with the original SIM-AC definitions. We can view this in the same paradigm we discussed for SIM-AC-style definitions in general; there is value in studying very strong definitions which exploit ideal primitives beyond how they can reasonably be thought to capture something about reality because these notions can then serve as intermediate steps for proving (in the ideal model) that the scheme satisfies other more "reasonable" security notions.

4.1 SIM*-AC Definitions

Pseudorandom function security. We start with PRF security for a function family F. Our new definition is captured by the game $\mathrm{G}^{\mathsf{sim^*\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}$ shown in Fig. 3. It differs from $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{prf}}}$ as described above; namely, $\mathcal{A}_{\mathsf{prf}}$ is given oracle PPrim which uses P in both the real and simulated world.[10] In the simulated world, S is also given PPRIM to query and program P. Note that the scheme algorithm $\mathsf{F.Ev}$ is still given access only to $\mathsf{P.Ls}$ and not to $\mathsf{P.Prog}$.

We define $\mathsf{Adv}^{\mathsf{sim}^*\text{-}\mathsf{ac}\text{-}\mathsf{prf}}_{\mathsf{F,S,P},\mathcal{A}_{\mathsf{prf}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{sim}^*\text{-}\mathsf{ac}\text{-}\mathsf{prf}}_{\mathsf{F,S,P},\mathcal{A}_{\mathsf{prf}}}(\lambda)] - 1$ and say that F is SIM*-AC-PRF secure with P if there exists a PPT S such that for all PPT $\mathcal{A}_{\mathsf{prf}}$, the advantage function $\mathsf{Adv}^{\mathsf{sim}^*\text{-}\mathsf{ac}\text{-}\mathsf{prf}}_{\mathsf{F,S,P},\mathcal{A}_{\mathsf{prf}}}(\cdot)$ is negligible. Note here that we quantified the simulator before the adversary, unlike in SIM-AC-PRF security where the simulator is allowed to depend on the adversary. This strengthens the definition and is necessary for some of our positive results, but for some of our results the weaker quantification will suffice. We say F is wSIM*-AC-PRF secure with P if for all PPT $\mathcal{A}_{\mathsf{prf}}$ there exists a PPT S such that $\mathsf{Adv}^{\mathsf{sim}^*\text{-}\mathsf{ac}\text{-}\mathsf{prf}}_{\mathsf{F,S,P},\mathcal{A}_{\mathsf{prf}}}(\cdot)$ is negligible.

Encryption definitions. The SIM*-AC-CCA security of an encryption scheme SE is similarly captured by the game $\mathrm{G}^{\mathsf{sim}^*\text{-}\mathsf{ac}\text{-}\mathsf{cca}}$ defined in Fig. 3 which modifies the SIM-AC game to have the attacker and simulator both use PPRIM. We define $\mathsf{Adv}^{\mathsf{sim}^*\text{-}\mathsf{ac}\text{-}\mathsf{cca}}_{\mathsf{SE,S,P},\mathcal{A}_{\mathsf{cca}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{sim}^*\text{-}\mathsf{ac}\text{-}\mathsf{cca}}_{\mathsf{SE,S,P},\mathcal{A}_{\mathsf{cca}}}(\lambda)] - 1$ and say SE is SIM*-AC-CCA secure with P if there exists a PPT S such that for all PPT $\mathcal{A}_{\mathsf{cca}}$, the advantage function $\mathsf{Adv}^{\mathsf{sim}^*\text{-}\mathsf{ac}\text{-}\mathsf{cca}}_{\mathsf{SE,S,P},\mathcal{A}_{\mathsf{cca}}}(\cdot)$ is negligible. wSIM*-AC-CCA is captured by quantifying the simulator after the adversary.

Chosen-plaintext security is captured by restricting attention to attackers that do not query decryption. We then write $\mathsf{sim}^*\text{-}\mathsf{ac}\text{-}\mathsf{cpa}$ in superscripts. SIM*-AC-X and wSIM*-AC-X security for $\mathrm{X} \in \{\mathrm{KPA}, \$, \mathrm{AE}\}$ security are defined by restricting the behavior of the simulator appropriately.

4.2 Single-user security implies multi-user security

As with SIM-AC security, we can capture single-user SIM*-AC security by requiring that all of the attacker's oracle queries use the same value of u. The following theorem captures that single-user SIM*-AC-CPA security implies multi-user security. The result would also hold with SIM*-AC-X security for any $\mathrm{X} \in \{\mathrm{PRF}, \mathrm{CCA}, \$, \mathrm{AC}\}$, via the same proof technique. If *does not* hold for $\mathrm{X} = \mathrm{KP}$. We will discuss why in more detail after the proof.

Theorem 1. *Single-user SIM*-AC-CPA security implies multi-user SIM*-AC-CPA security.*

This proof follows using the ideas from a fairly standard single-user to multi-user proof via a hybrid argument. Given a single-user simulator S_1 and multi-user adversary $\mathcal{A}$, we define single-user $\mathcal{A}_1$ to pick a random t and respond to queries with $u < t$ by encrypting honestly, with $u = t$ using its own encryption oracle, and with $u > t$ using a copy of S_1 specific for that user. The multi-user simulator we construct runs multiple independent copies of the single-user simulator – one

[10] Here we are using a notational convention that an algorithm given more inputs than it expects will ignore any extra inputs, so $\mathsf{P.Ls}(1^\lambda, k, x, y : \sigma_{\mathsf{P}})$ is equivalent to $\mathsf{P.Ls}(1^\lambda, k, x : \sigma_{\mathsf{P}})$.

for each user. Note that this proof critically requires all three of the changes we used to derive SIM*-AC from SIM-AC: (i) the simulator needs to be quantified before the adversary so that $\mathcal{A}_1$ can run S_1, (ii) the simulator must not have full control of the ideal primitives output so there is no ambiguity in which "copy" of the simulator run by $\mathcal{A}_1$ should get to respond to primitive queries, and (iii) the adversary must be able to program the ideal primitive so that $\mathcal{A}_1$ is able to correctly control the primitive when running copies of S_1.

Proof. Let SE be single-user SIM*-AC-CPA secure with P and S_1 be the simulator that is guaranteed to exist. We show that SE is SIM*-AC-CPA secure with P via the following simulator which runs independent copies of S_1 for each user.

```
S.Init(1^λ)
    $\sigma_{(\cdot)} \leftarrow\!\!\$\ \mathsf{S}_1.\mathsf{Init}(1^\lambda)$
    Return $\sigma_{(\cdot)}$

S.Enc^PPrim(1^λ, u, ℓ : σ_(·))
    $c \leftarrow\!\!\$\ \mathsf{S}_1.\mathsf{Enc}^{\mathrm{PPRIM}}(1^\lambda, u, \ell : \sigma_u)$
    Return $c$

S.Exp^PPrim(1^λ, u, M_u : σ_(·))
    $k \leftarrow\!\!\$\ \mathsf{S}_1.\mathsf{Exp}^{\mathrm{PPRIM}}(1^\lambda, u, M_u : \sigma_u)$
    Return $k$
```

Let $\mathcal{A}$ be a SIM*-AC-CPA adversary. It will be notationally convenient to assume that it only queries users with identifiers $u \in [u_\lambda] = \{1, \ldots, u_\lambda\}$ where $u_{(\cdot)}$ is a polynomial. This assumption is without loss of generality.

```
Hybrid $\mathrm{H}_i(\lambda)$, $0 \le i \le u_\lambda$
    For $u \in [u_\lambda]$ do
        $k_u \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
        $\sigma_u \leftarrow\!\!\$\ \mathsf{S}_1.\mathsf{Init}(1^\lambda)$
    $\sigma_{\mathsf{P}} \leftarrow\!\!\$\ \mathsf{P.Init}(1^\lambda)$
    $b' \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{ENC},\mathrm{EXP},\mathrm{PPRIM}}(1^\lambda)$
    Return $(b' = 1)$

Enc$(u, m)$
    If $u \le i$ then $d \leftarrow 0$
    Else $d \leftarrow 1$
    $c \leftarrow \mathrm{ENC}_d(u, m)$
    Return $c$

Exp$(u)$
    If $u \le i$ then $d \leftarrow 0$
    Else $d \leftarrow 1$
    $k \leftarrow \mathrm{EXP}_d(u)$
    Return $k$

Enc$_d(u, m)$
    If not $X_u$ then $\ell \leftarrow |m|$ else $\ell \leftarrow m$
    If $d = 1$ then $c \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, k_u, m)$
    Else $c \leftarrow\!\!\$\ \mathsf{S}_1.\mathsf{Enc}^{\mathrm{PPRIM}}(1^\lambda, u, \ell : \sigma_u)$
    $M_u.\mathsf{add}(c, m)$
    Return $c$

Exp$_d(u)$
    If $d = 1$ then $k \leftarrow k_u$
    Else $k \leftarrow\!\!\$\ \mathsf{S}_1.\mathsf{Exp}^{\mathrm{PPRIM}}(1^\lambda, u, M_u : \sigma_u)$
    $X_u \leftarrow \mathsf{true}$
    Return $k$
```

```
Adversary $\mathcal{A}_1^{\mathrm{ENC},\mathrm{EXP},\mathrm{PPRIM}}(\lambda)$
    For $u \in [u_\lambda]$ do
        $k_u \leftarrow\!\!\$\ \mathsf{SE.Kg}(1^\lambda)$
        $\sigma_u \leftarrow\!\!\$\ \mathsf{S}_1.\mathsf{Init}(1^\lambda)$
    $t \leftarrow\!\!\$\ \{1, \ldots, u_\lambda\}$
    $b' \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{ENCSIM},\mathrm{EXPSIM},\mathrm{PPRIM}}(1^\lambda)$
    Return $b'$

Enc$_d(u, m)$, Exp$_d(u)$
    //Unchanged from above

EncSim$(u, m)$
    If $u < t$ then $c \leftarrow \mathrm{ENC}_0(u, m)$
    Else if $u = t$ then $c \leftarrow \mathrm{ENC}(u, m)$
    Else $c \leftarrow \mathrm{ENC}_1(u, m)$
    Return $c$

ExpSim$(u)$
    If $u < t$ then $k \leftarrow \mathrm{EXP}_0(u)$
    Else if $u = t$ then $k \leftarrow \mathrm{EXP}(u)$
    Else $k \leftarrow \mathrm{EXP}_1(u)$
    Return $k$
```

Fig. 4. Hybrids and adversary showing single-user security implies multi-user.

Now, consider the hybrid games H_i for $i = 0, \dots, u_\lambda$ defined in Fig. 4. For $u \le i$, the game uses $\textsc{Enc}_0$ and $\textsc{Exp}_0$ to respond to encryption and exposure queries as in the $b = 0$ simulated world of $\mathrm{G}^{\mathsf{sim^*\text{-}ac\text{-}cpa}}$ using S. Otherwise, it uses $\textsc{Enc}_1$ and $\textsc{Exp}_1$ to respond as in the $b = 1$ real world. Each hybrid game returns true whenever $\mathcal{A}$ outputs 1. When $i = u_\lambda$, it always holds that $u \le i$ so this game is identical to the $b = 0$ simulated world (except that the output boolean is flipped). In the other extreme, when $i = 0$, it never holds that $u \le i$ so this game is identical to the $b = 1$ real world. Then (by standard conditional probability calculation) we have

$$\mathsf{Adv}^{\mathsf{sim^*\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}}(\lambda) = \Pr[\mathrm{H}_0] - \Pr[\mathrm{H}_{u_\lambda}] = \sum_{i=1}^{u_\lambda} \Pr[\mathrm{H}_{i-1}] - \Pr[\mathrm{H}_i].$$

We construct a single-user adversary $\mathcal{A}_1$ that obtains advantage $1/u_\lambda$ times the above. It samples an index $t \in \{1, \dots, u_\lambda\}$ at random. Then it runs $\mathcal{A}$, simulating their oracle queries. When $u < t$, it responds as in the simulated world of $\mathrm{G}^{\mathsf{ep\text{-}sim\text{-}ac\text{-}cpa}}$ using S_1. When $u = t$ it forwards the query to its own oracle. Otherwise, it responds to $\textsc{Enc}$ and $\textsc{Exp}$ queries as in the real world. Let b denote the bit in the game $\mathcal{A}_1$ is being run in and t be the random value picked by $\mathcal{A}_1$. Then in the view of $\mathcal{A}$, the oracles for the first $t - b$ users are simulated and the rest are real – this is identical to its view in the hybrid game H_{t-b}.

Then the following calculations complete the proof.

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{sim^*\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S}_1,\mathsf{P},\mathcal{A}_1}(\lambda) &= \mathbf{E}_t[\Pr[\mathrm{H}_{t-1}]] - \mathbf{E}_t[\Pr[\mathrm{H}_{t-0}]] \\
&= (1/u_\lambda) \sum_{t=1}^{u_\lambda} \Pr[\mathrm{H}_{t-1}] - (1/u_\lambda) \sum_{t=1}^{u_\lambda} \Pr[\mathrm{H}_t] \\
&= (1/u_\lambda) \sum_{i=1}^{u_\lambda} \Pr[\mathrm{H}_{i-1}] - \Pr[\mathrm{H}_i] \\
&= (1/u_\lambda) \mathsf{Adv}^{\mathsf{sim^*\text{-}ac\text{-}cpa}}_{\mathsf{SE},\mathsf{S},\mathsf{P},\mathcal{A}}(\lambda).
\end{aligned}$$

Here $\mathbf{E}_t$ denotes expectation over $t \leftarrow_\$ \{1, \dots, u_\lambda\}$. □

We can note in the above proof that for $\mathcal{A}_1$ to be able to correctly run $\textsc{Enc}_0$ and $\textsc{Exp}_0$ it needed to run S_1. This means that we needed the stronger quantification where the adversary can depend on the simulator and that the adversary needed to have the ability to program the random oracle.

Key-private security. Among the various SIM*-AC security notions we consider here, the only variant for which single-user security does not imply multi-user security is SIM*-AC-KPA security. Here, the simulator may not make use of its input u when replying to encryption queries for un-exposed users (beyond checking if they are exposed). Note that in the hybrid argument above, the multi-user simulator S uses the user identifier u to decide which state σ_u to use. Hence this is incompatible with SIM*-AC-KPA security. Taking a step back, we can notice that this issue with the proof is unsurprising and inherent. The issue is that that single-user SIM*-AC-KPA does not meaningfully capture any notion of key-privacy because the restriction on the simulator's behavior is trivially achievable when the attacker will only every query a single user. This is nicely captured by the following result.

Theorem 2. *Single-user SIM*-AC-KPA security is equivalent to SIM*-AC-CPA security, which is weaker than SIM*-AC-KPA security.*

Proof (Sketch). Note that single-user SIM*-AC-KPA security implies single-user SIM*-AC-CPA security trivially. Then, by Theorem 1 this implies SIM*-AC-CPA security. In the other direction, we can create a single-user SIM*-AC-KPA simulator from a SIM*-AC-CPA simulator by always running the latter on, say, $u = 1$. Hence the first claim of the theorem holds.

We can see that SIM*-AC-CPA security is weaker than SIM*-AC-KPA security by constructing a contrived scheme. Given some scheme SE, we define a new scheme which adds a random bit d to its keys and then appends d to every ciphertext produced. It is straightforward to show this new scheme is SIM*-AC-CPA secure if SE was, but that is is not SIM*-AC-KPA secure. □

4.3 Cascade Construction

If $\mathsf{F} : \mathsf{F.K} \times \mathsf{F.Inp} \to \mathsf{F.K}$ is a function family and n is a polynomial, then the n-cascade construction $\mathsf{F}^n : \mathsf{F.K} \times \mathsf{F.Inp}^n \to \mathsf{F.K}$ is defined by the evaluation algorithm $\mathsf{F^n.Ev}(1^\lambda, k_0, \boldsymbol{x})$ which computes $k_i \leftarrow \mathsf{F.Ev}(1^\lambda, k_{i-1}, \boldsymbol{x}_i)$ for $i = 1, \ldots, n(\lambda)$ and then outputs $k_{n(\lambda)}$. Here $\boldsymbol{x}_i$ denotes the i-th entry of vector $\boldsymbol{x}$. This is a "domain extension" technique for building a PRF with a large domain from one with a small domain. It was originally defined and analyzed in [4].[11] F^n generalizes the GGM construction of a PRF from a PRG [17]. It underlies several other constructions of PRFs including AMAC, HMAC, and NMAC [2,3,1].

Theorem 3. *If F is SIM*-AC-PRF secure with P, then F^n is as well.*

The proof of this result is given in the full version. Intuitively, we can think of the possible keys generated by F^n existing in a tree structure. Our proof does a hybrid argument over the layers of the tree where we one at a time switch the layers to being simulated. The simulator for a given layer treats all of the keys at its layer as being multiple F "users". This proof requires the "strong" quantification, the simulator to not completely replacing the ideal primitive, and the adversary having the ability to program the ideal primitive so that it can internally run the simulator for layers that have been switched already.

Jaeger and Tyagi [23, ePrint, p.22-23] said, "It is often useful to construct a PRF H with large input domains from a PRF F with smaller input domains [. . .]

[11] Technically, they considered a more general construction where the number of iterations was not a priori fixed and so the adversary was restricted to make only prefix-free queries. Our proof would extend to this setting as well.

one can often [use our techniques] to lift a PRF security proof for H to a SIM-AC-PRF security proof for H whenever F is SIM-AC-PRF secure." The cascade construction is one choice of H for which this is *not* possible with SIM-AC, but becomes possible with SIM*-AC.

5 Asymmetric Encryption

In this section, we provide our treatment for the security of asymmetric cryptographic primitives against adaptive compromise. We start by providing our security definitions for public-key encryption (PKEs) and key-encapsulation mechanisms (KEMs). Then we discuss how our definitions compare to prior definitions, in particular those of Camensich, Lehmann, Neven, and Samelin [12]. We show that the KEM/DEM approach to constructing a PKE scheme works with these definitions and that standard ways of constructing CPA/CCA secure KEMs from one-way secure primitives and a random oracle are secure.

5.1 Definitions

Public-key encryption. The SIM*-AC-CCA security of a public-key encryption scheme PKE is captured by the game $\mathrm{G}^{\mathsf{sim^*\text{-}ac\text{-}cca}}$ shown in Fig. 5. It differs from the SIM*-AC-CCA definition for symmetric encryption (Fig. 2) in that it introduces an encryption key oracle (Ek) that the adversary can call to learn the public encryption key for a user and it has oracles for two different kinds of exposure. The receiver exposure oracle (RExp) is like the exposure oracles from prior games, returning a user's secret decryption key. The sender exposure oracle (SExp) allows the attacker to ask for the randomness underlying the ciphertexts that were returned by encryption.

We define $\mathsf{Adv}^{\mathsf{sim^*\text{-}ac\text{-}cca}}_{\mathsf{PKE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda) = 2\Pr[\mathrm{G}^{\mathsf{sim^*\text{-}ac\text{-}cca}}_{\mathsf{PKE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda)]-1$ and say PKE is SIM*-AC-CCA secure with P if there exists a PPT S such that for all PPT $\mathcal{A}_{\mathsf{cca}}$, the advantage function $\mathsf{Adv}^{\mathsf{sim^*\text{-}ac\text{-}cca}}_{\mathsf{PKE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\cdot)$ is negligible. wSIM*-AC-CCA is captured by quantifying the simulator after the adversary. We capture xSIM*-AC-CPA by ignoring the decryption oracle. Security considering only compromise of the receiver/sender can be captured by ignoring the appropriate oracle. Then we write SIM*-rAC or SIM*-sAC.

Key encapsulation mechanism. We also give definitions for key encapsulation mechanisms (KEM). Our SIM*-AC definitions are highly analogous to the corresponding public-key encryption definition. They are formally specified by the game $\mathrm{G}^{\mathsf{sim\text{-}ac\text{-}cca}}$ shown in Fig. 5. Therein, the Enc and Dec oracles have been replaced with Encaps and Decaps oracles. The encapsulation oracle returns a ciphertext along with the corresponding encapsulated key. In the ideal world, the simulator provides the ciphertext and the encapsulated key is chosen at random from the key space by the game for unexposed users.

We define $\mathsf{Adv}^{\mathsf{sim^*\text{-}ac\text{-}cca}}_{\mathsf{KEM},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cpa}}}(\lambda)$ and the notions xSIM*-yAC-X for $\mathrm{x} \in \{\varepsilon, \mathrm{w}\}$, $\mathrm{y} \in \{\varepsilon, \mathrm{r}, \mathrm{s}\}$, and $\mathrm{X} \in \{\mathrm{CCA}, \mathrm{CPA}\}$ as for PKE.

```
Games $\mathrm{G}^{\mathsf{sim^*\text{-}ac\text{-}cca}}_{\mathsf{PKE},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda)$
    $(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\$ \mathsf{PKE.Kg}(1^\lambda)$
    $\sigma_{\mathsf{P}} \leftarrow\$ \mathsf{P.Init}(1^\lambda)$
    $\sigma \leftarrow\$ \mathsf{S.Init}(1^\lambda)$
    $b \leftarrow\$ \{0,1\}$
    $b' \leftarrow\$ \mathcal{A}_{\mathsf{cca}}^{\textsc{Ek},\textsc{Enc},\textsc{Dec},\textsc{SExp},\textsc{RExp},\textsc{PPrim}}(1^\lambda)$
    Return $(b = b')$

Game $\mathrm{G}^{\mathsf{sim^*\text{-}ac\text{-}cca}}_{\mathsf{KEM},\mathsf{S},\mathsf{P},\mathcal{A}_{\mathsf{cca}}}(\lambda)$
    $(ek_{(\cdot)}, dk_{(\cdot)}) \leftarrow\$ \mathsf{KEM.Kg}(1^\lambda)$
    $\sigma_{\mathsf{P}} \leftarrow\$ \mathsf{P.Init}(1^\lambda)$
    $\sigma \leftarrow\$ \mathsf{S.Init}(1^\lambda)$
    $b \leftarrow\$ \{0,1\}$
    $b' \leftarrow\$ \mathcal{A}_{\mathsf{cca}}^{\textsc{Ek},\textsc{Encaps},\textsc{Decaps},\textsc{SExp},\textsc{RExp},\textsc{PPrim}}(1^\lambda)$
    Return $(b = b')$

$\textsc{Ek}(u)$
    $\boxed{ek' \leftarrow ek_u}$
    $\dbox{ek' \leftarrow\$ \mathsf{S.Ek}^{\textsc{PPrim}}(1^\lambda, u : \sigma)}$
    Return $ek'$

$\textsc{PPrim}(\mathsf{Op}, k, x, y)$
    Require $\mathsf{Op} \in \{\mathsf{Ls}, \mathsf{Prog}\}$
    $y \leftarrow\$ \mathsf{P.Op}(1^\lambda, k, x, y : \sigma_{\mathsf{P}})$
    Return $y$

$\textsc{Enc}(u, m)$
    If not $X_u$ then $\ell \leftarrow |m|$ else $\ell \leftarrow m$
    $r \leftarrow\$ \mathsf{PKE.Rand}(\lambda)$
    $\boxed{c \leftarrow \mathsf{PKE.Enc}^{\mathsf{P}}(1^\lambda, ek_u, m; r)}$
    $\dbox{c \leftarrow\$ \mathsf{S.Enc}^{\textsc{PPrim}}(1^\lambda, u, \ell : \sigma)}$
    $M_u.\mathsf{add}(c, m)$; $R_u.\mathsf{add}(r)$
    Return $c$

$\textsc{Dec}(u, c)$
    If $M_u\langle c\rangle \neq \perp$ then return $M_u\langle c\rangle$
    $\boxed{m \leftarrow \mathsf{PKE.Dec}^{\mathsf{P}}(1^\lambda, dk_u, c)}$
    $\dbox{m \leftarrow\$ \mathsf{S.Dec}^{\textsc{PPrim}}(1^\lambda, u, c : \sigma)}$
    Return $m$

$\textsc{SExp}(u, i)$
    $\boxed{r \leftarrow R_u[i]}$
    $\dbox{r \leftarrow\$ \mathsf{S.SExp}^{\textsc{PPrim}}(1^\lambda, u, i, M_u[i] : \sigma)}$
    Return $r$

$\textsc{RExp}(u)$
    $\boxed{dk' \leftarrow dk_u}$
    $\dbox{dk' \leftarrow\$ \mathsf{S.RExp}^{\textsc{PPrim}}(1^\lambda, u, M_u : \sigma)}$
    $X_u \leftarrow \mathsf{true}$
    Return $dk'$

$\textsc{Encaps}(u)$
    $r \leftarrow\$ \mathsf{KEM.Rand}(\lambda)$
    $\boxed{(c, k) \leftarrow \mathsf{KEM.Encaps}^{\mathsf{P}}(1^\lambda, ek_u; r)}$
    $\dbox{(c, k) \leftarrow\$ \mathsf{S.Encaps}^{\textsc{PPrim}}(1^\lambda, u : \sigma)}$
    $\dbox{\text{If not } X_u \text{ then } k \leftarrow\$ \mathsf{KEM.K}(\lambda)}$
    $M_u.\mathsf{add}(c, k)$; $R_u.\mathsf{add}(r)$
    Return $(c, k)$

$\textsc{Decaps}(u, c)$
    If $M_u\langle c\rangle \neq \perp$ then return $M_u\langle c\rangle$
    $\boxed{k \leftarrow \mathsf{KEM.Decaps}^{\mathsf{P}}(1^\lambda, dk_u, c)}$
    $\dbox{k \leftarrow\$ \mathsf{S.Decaps}^{\textsc{PPrim}}(1^\lambda, u, c : \sigma)}$
    Return $k$
```

Fig. 5. Games defining the SIM*-AC-CCA security of PKE and KEM. Solid-boxed code is only executed if $b = 1$. Dash-boxed code is only executed if $b = 0$.

5.2 Comparison to SIM-FULL Definition

The definition we have arrived at is similar to the FULL-SIM security definition for PKE introduced by Camensich, Lehmann, Neven, and Samelin (CLNS) [12]. We quickly summarize the differences. There are two dimensions in which their definition is strong than ours. First, their definition considers PKE with labels, while we have decided not consider labels. Labels can easily be added. Likely, the best way to incorporate labels in constructions would be to use a symmetric encryption scheme that accepts associated data as part of the KEM/DEM transform (discussed momentarily). Second, in FULL-SIM the randomness used by key generation is revealed rather than the decryption key. SIM-AC* can be used to reason over this case by simply modify the scheme to use said randomness as its decryption key (and recompute the actual decryption key during decryption).

Our definition strengthens theirs in several dimension. Theirs is more closely analogous to SIM-AC than SIM*-AC as the simulator is given complete control of the random oracle, the adversary is not able to modify it, and the "weak" quantification is used. Resultantly, their single-user definition is seemingly unable to prove that a corresponding multi-user definition holds.

We are not restrictive in the type of ideal primitive considered. CLNS considered only one specific construction in which they basically used a trapdoor permutation generator as a KEM and then hand-crafted a symmetric encryption scheme using a random oracle.[12] We will momentarily show that the task of building SIM*-AC secure PKE can be broken down into constructing KEMs and symmetric encryption. This is modular, allowing numerous instantiation and in particular, allowing the symmetric encryption to be instantiated by well-studied and standardized schemes based on blockciphers rather than using less efficient hash functions throughout.

CLNS showed that SIM-FULL implied a variety of prior definitions considering compromise scenarios for PKE. These implications will carry over to our definition as well. We explore the relationship between SIM*-AC-CCA, SIM-FULL, and these other definitions more formally in the full version. Further, CLNS considered a UC secure notion and proved it to be essentially equivalent to SIM-FULL. Camenisch, Drijvers, Gagliardoni, Lehmann, and Nevin [10] considered this in the UC programmable random oracle model, proving the same construction secure. Likely SIM*-AC-CCA is equivalent to this notion and so our result will give modular, standard, efficient instantiations of UC secure public key encryption secure under adaptive compromise.

5.3 KEM/DEM Hybrid Encryption.

A common technique for building public key encryption is KEM/DEM hybrid encryption in which a key encapsulation mechanism produces a key which is

[12] Speaking loosely, they basically use a random input to the trapdoor permutation as a "symmetric key" with which they perform counter mode encryption, using the random oracle as a pseudorandom function and then perform a MAC over all of the relevant variables, again using the random oracle as a pseudorandom function.

then used to encrypt the message with a symmetric encryption scheme (i.e. "data encapsulation mechanism"). This was proven secure by Cramer and Shoup [15].

Let KEM be a key encapsulation mechanism and SE be a symmetric encryption scheme (i.e. data encapsulation mechanism) where $\mathsf{SE.Kg}$ samples uniformly from $\mathsf{KEM.K}$. We denote the KEM/DEM scheme as $\mathsf{KD}[\mathsf{KEM},\mathsf{SE}]$ and provide the algorithms $\mathsf{KD.Enc}$ and $\mathsf{KD.Dec}$ below, where we assume KEM and SE expect access to ideal primitive P. Then KD expects access to P. It key generation algorithm is defined by $\mathsf{KD}[\mathsf{KEM},\mathsf{SE}].\mathsf{Kg} = \mathsf{KEM.Kg}$.

```
$\underline{\mathsf{KD}[\mathsf{KEM},\mathsf{SE}].\mathsf{Enc}^{\mathsf{P}}(1^\lambda, ek, m)}$
$(c_{\mathsf{KEM}}, k) \leftarrow\!\!\$\ \mathsf{KEM.Encaps}^{\mathsf{P}}(1^\lambda, ek)$
$c_{\mathsf{SE}} \leftarrow\!\!\$\ \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, k, m)$
$c \leftarrow (c_{\mathsf{KEM}}, c_{\mathsf{SE}})$
Return $c$

$\underline{\mathsf{KD}[\mathsf{KEM},\mathsf{SE}].\mathsf{Dec}^{\mathsf{P}}(1^\lambda, dk, c)}$
$(c_{\mathsf{KEM}}, c_{\mathsf{SE}}) \leftarrow c$
$k \leftarrow \mathsf{KEM.Decaps}^{\mathsf{P}}(1^\lambda, dk, c_{\mathsf{KEM}})$
$m \leftarrow \mathsf{SE.Dec}^{\mathsf{P}}(1^\lambda, k, c_{\mathsf{SE}})$
Return $m$
```

It is assumed that $\mathsf{SE.Dec}$ immediately halts and returns $\bot$ if $k = \bot$. Next, we show that given the appropriate adaptive compromise security for the underlying KEM scheme and encryption scheme, the composed KEM/DEM scheme is also secure against adaptive compromise.

Exposure of encryption randomness is not captured by our definitions for symmetric encryption. Rather than introduce a new security definition, in these cases we restrict attention to *coin extractable* schemes for which there exists an algorithm $\mathsf{SE.CExt}$ which always satisfies $\mathsf{SE.CExt}^{\mathsf{P}}(1^\lambda, k, \mathsf{SE.Enc}^{\mathsf{P}}(1^\lambda, k, m; r)) = r$. We are not aware of any practically deployed schemes which do not satisfy this. For technical reasons, we assume that $\mathsf{SE.CExt}$ is query consistent by which we mean that it does not make any ideal primitive queries that were not made by the execution of $\mathsf{SE.Enc}$ that produced its input.

Theorem 4. *Let $x \in \{\varepsilon, w\}$, $y \in \{\varepsilon, r, s\}$, and $X \in \{CPA, CCA\}$. If* KEM *is xSIM*-yAC-X secure with* P *and* SE *is xSIM*-AC-X secure with* P *(and coin extractable if $y \in \{\varepsilon, s\}$), then* $\mathsf{KD}[\mathsf{KEM},\mathsf{SE}]$ *is xSIM*-yAC-X secure with* P.

In fact, for the DEM we need only "single-challenge" security wherein the attacker makes at most one encryption query per user. This allows the use of deterministic DEMs. The proof of this result is given in the full version. The general flow of the proof is what one would expect, first we replace honest use of the KEM with simulated use that outputs uniformly random keys. We think of the i-th key generated for user u as correspond to a DEM user (u, i) and replace the DEM with simulation.

5.4 Hashed KEM

We consider a simple, standard way to construct a CPA secure KEM from a one-way secure KEM and a random oracle. Conceptually, this construction follows from the CPA secure PKE scheme considered in [8]. Let KEM be a key encapsulation mechanism. Then the hashed KEM scheme which outputs the hash of a key generated by KEM is denoted as $\mathsf{HKEM}[\mathsf{KEM}]$. Its algorithms are defined as follows. Its key generation algorithm is defined by $\mathsf{HKEM}[\mathsf{KEM}].\mathsf{Kg} = \mathsf{KEM.Kg}$.

```
$\underline{\mathsf{HKEM}[\mathsf{KEM}].\mathsf{Encaps}^{\mathsf{P}\times\mathsf{P}_{\mathsf{rom}}}(1^\lambda, ek)}$
$(c, k_{\mathsf{KEM}}) \leftarrow\!\!\$\ \mathsf{KEM}.\mathsf{Encaps}^{\mathsf{P}}(1^\lambda, ek)$
$k \leftarrow \mathsf{P}_{\mathsf{rom}}(k_{\mathsf{KEM}}, \varepsilon)$; Return $(c, k)$

$\underline{\mathsf{HKEM}[\mathsf{KEM}].\mathsf{Decaps}^{\mathsf{P}\times\mathsf{P}_{\mathsf{rom}}}(1^\lambda, dk, c)}$
$k_{\mathsf{KEM}} \leftarrow \mathsf{KEM}.\mathsf{Decaps}^{\mathsf{P}}(1^\lambda, dk, c)$
$k \leftarrow \mathsf{P}_{\mathsf{rom}}(k_{\mathsf{KEM}}, \varepsilon)$; Return $k$
```

If the KEM expects access to P, then HKEM expects access to $\mathsf{P} \times \mathsf{P}_{\mathsf{rom}}$. Note that the random oracle must be "new" and cannot be queried by KEM. This is necessary as the KEM could otherwise query the random oracle on the key it will output and include that as part of the ciphertext. Note that one can use oracle cloning [5], to create multiple random oracles from a single random oracle.

Intuitively, CPA security is achieved if the attacker cannot predict k_{KEM} and query it to the random oracle, i.e., as long as the KEM is one-way secure.

Theorem 5. *If* KEM *is OW* secure with* P*, then* $\mathsf{HKEM}[\mathsf{KEM}]$ *is SIM*-AC-CPA secure with respect to* $\mathsf{P} \times \mathsf{P}_{\mathsf{rom}}$.

The full proof (and the formal definition of OW*) are given in the full version. The proof works as one would expect. The simulator produces ciphertexts by using KEM honestly. On exposures, it returns the keys/randomness it used and attempts to reprogram the random oracle to map keys encapsulated by KEM to the keys that were randomly sampled by the encapsulation oracle.

5.5 Fujisaki-Okamoto Transform

Finally, we consider a way to construct a CCA secure KEM from a one-way secure KEM. In particular, we look at part of one version of the Fujisaki-Okamoto transformation [16]. We work from the modular treatment of Hofheinz, Hövelmanns, and Kiltz [20] (HHK), in particular showing that the transformation which they refer to as $\mathsf{U}^{\not\perp}$ achieves SIM*-AC-CCA security. This should extend to the other variants as well, but have focused on one for simplicity. Slightly corrected versions of HHK's proofs can be found in [22, Sec. 2.1-2.2].

Let KEM be a key encapsulation mechanism and F be a function family. Then we consider the scheme $\mathsf{U}^{\not\perp}[\mathsf{KEM}, \mathsf{F}]$ defined as follows. The key generation algorithm $\mathsf{U}^{\not\perp}[\mathsf{KEM}, \mathsf{F}].\mathsf{Kg}$ generates keys $(ek, dk) \leftarrow\!\!\$\ \mathsf{KEM}.\mathsf{Kg}(1^\lambda)$ and $fk \leftarrow\!\!\$\ \mathsf{F}.\mathsf{Kg}(1^\lambda)$, then outputs $(ek, (dk, fk))$.

```
$\underline{\mathsf{U}^{\not\perp}[\mathsf{KEM},\mathsf{F}].\mathsf{Encaps}^{\mathsf{P}\times\mathsf{P}_{\mathsf{rom}}}(1^\lambda, ek)}$
$(c, k_{\mathsf{KEM}}) \leftarrow\!\!\$\ \mathsf{KEM}.\mathsf{Encaps}^{\mathsf{P}}(1^\lambda, ek)$
$k \leftarrow \mathsf{P}_{\mathsf{rom}}(k_{\mathsf{KEM}}, c)$; Return $(c, k)$

$\underline{\mathsf{U}^{\not\perp}[\mathsf{KEM},\mathsf{F}].\mathsf{Decaps}^{\mathsf{P}\times\mathsf{P}_{\mathsf{rom}}}(1^\lambda, (dk, fk), c)}$
$k_{\mathsf{KEM}} \leftarrow \mathsf{KEM}.\mathsf{Decaps}^{\mathsf{P}}(1^\lambda, dk, c)$
If $k_{\mathsf{KEM}} \neq \perp$ then $k \leftarrow \mathsf{P}_{\mathsf{rom}}(k_{\mathsf{KEM}}, c)$
Else $k \leftarrow \mathsf{F}.\mathsf{Ev}^{\mathsf{P}\times\mathsf{P}_{\mathsf{rom}}}(1^\lambda, fk, c)$
Return $k$
```

Here $\mathsf{U}^{\not\perp}[\mathsf{KEM}, \mathsf{F}].\mathsf{K}(\lambda) = \mathsf{F}.\mathsf{Out}(\lambda) = \mathsf{P}_{\mathsf{rom}}.\mathcal{R}_\lambda$. Note that if KEM expects access to P, then $\mathsf{U}^{\not\perp}[\mathsf{KEM}, \mathsf{F}]$ expects access to $\mathsf{P} \times \mathsf{P}_{\mathsf{rom}}$. We allow F to have access to $\mathsf{P} \times \mathsf{P}_{\mathsf{rom}}$. It is important that KEM not have access to the random oracle used by the transform (otherwise it could, for example, ensure that it always produces output for which the first bit of $\mathsf{P}_{\mathsf{rom}}(k_{\mathsf{KEM}}, c)$ is 0 and thus distinguishable from random).

However, our results show there is no issue with F having access to the same random oracle used by $\mathsf{U}^{\not\perp}$. Indeed, HHK actually used the specific construction

$\mathsf{F}.\mathsf{Ev}(1^\lambda, \mathit{fk}, c) = \mathsf{P}_{\mathsf{rom}}(\mathit{fk}, c)$. Considering an arbitrary F is more general. We emphasize the proof with this generality only works because we are using our new SIM*-AC security definitions. Moreover, given that caveat, this supports Jaeger and Tyagi's motivation for introducing SIM-AC definitions because this modularity allows our proof to avoid the details of the random oracle analysis required to prove that $\mathsf{P}_{\mathsf{rom}}(k_{\mathsf{KEM}}, c)$ is secure.

Additionally HHK use a public-key encryption scheme applied to a random message in place of KEM. Again, $\mathsf{U}^{\not\perp}[\mathsf{KEM}, \mathsf{F}]$ is a generalization of this as the security they assume of the encryption scheme implies that the KEM obtained by encrypting a randome message satisfies the security we require.

They showed that the construction is IND-CCA secure as long as the underlying scheme achieves a variant of one-way security which provides to the attacker a plaintext checking oracle which decrypts a given ciphertext and returns a boolean indicating whether the result is the same as a given message. We show the same for our security definition.

Theorem 6. *If* KEM *is OW*-PCA secure with* P *and* F *is SIM*-AC-PRF secure with* $\mathsf{P} \times \mathsf{P}_{\mathsf{rom}}$*, then* $\mathsf{U}^{\not\perp}[\mathsf{KEM}, \mathsf{F}]$ *is SIM*-AC-CCA secure with* $\mathsf{P} \times \mathsf{P}_{\mathsf{rom}}$*.*

HHK gave a transform T which transforms a OW secure PKE scheme into a OW-PCA secure PKE scheme. Interpreting this as a KEM in the natural manner gives a OW-PCA secure KEM.

6 Recovering Prior Results

Finally, we conclude by showing that the positive results Jaeger and Tyagi [23] established regarding various notions of SIM-AC security also hold with respect to our analogous SIM*-AC notions. For this, we divide the results of Jaeger and Tyagi into three general categories. This first category covers results where (non-SIM-AC) security of some "high-level" construction is shown assuming its constituent elements satisfy SIM-AC security. The second category covers results where SIM-AC security of some "intermediate-level" construction is shown assuming its constituent elements satisfy SIM-AC security. The final category covers results where SIM-AC security of some "low-level" primitive is shown by direct ideal model analysis.

6.1 High-level Proofs

The first category is the easiest in which to replace SIM-AC with SIM*-AC. In particular Jaeger and Tyagi showed: (1) SIM-AC-CPA secure encryption suffices for a version of the OPAQUE password-authenticated key exchange protocol of Jarecki, et al. [24] (because the latter was proven secure assuming "equivocable encryption" which is a weaker notion than SIM-AC-CPA security), (2) SIM-AC-PRF secure PRFs and SIM-AC-KP secure encryption suffice for a searchable symmetric encryption scheme of Cash, et al. [14], and (3) SIM-AC-CPA secure encryption suffices for the self-revocable cloud storage scheme of Tyagi, et al. [27].

We can recover these results with wSIM*-AC in place of SIM-AC by noting that our new notion is strictly stronger.

Lemma 1. *For* $\mathrm{X} \in \{\mathrm{PRF}, \mathrm{CPA}, \mathrm{KPA}, \$, \mathrm{CCA}, \mathrm{AE}\}$*, wSIM*-AC-X security implies SIM-AC-X security. The converse does not hold.*

This result follows from the fact that wSIM*-AC security strengthens adversaries (by allowing them to program the ideal primitive) and weakens simulators (by restricting them to explicitly program the ideal primitive rather than having complete control of it). For the converse, note that a SIM*-AC adversary can, e.g., break the one-way function or collision-resistance security of a random oracle by programming it appropriately. Hence, one can modify a SIM-AC secure scheme to be trivially insecure (e.g. reveal its secret key) when a collision in the random oracle is known. SIM-AC security will be maintained, but the modified scheme will not be SIM*-AC secure.

In each of the searchable symmetric encryption and BurnBox proof, Jaeger and Tyagi had to assume that the constituent elements each used separate ideal primitives. Using SIM*-AC definitions we could reproduce these results without the assumption of separate ideal primitives using the proof modifications we discussion for intermediate-level proofs.

6.2 Intermediate-level Proofs

In the second category, Jaeger and Tyagi gave security results for several encryption schemes. There is no general way to prove that these result carry over from SIM-AC to SIM*-AC security notions.[13] However, by examining the details of the proofs used for each of these result we can see that we are in luck. In each, the ideal primitive was used as a black-box. Constructed SIM-AC reduction adversaries provided the given SIM-AC adversaries with direct access to their own PRIM oracle. The $\mathsf{S.Ls}$ algorithm of any constructed SIM-AC simulators S just ran the corresponding algorithms of the given SIM-AC simulators.

As such, modifying these proofs for SIM*-AC (or wSIM*-AC) requires only syntactic change to treat the ideal primitive as a black-box. Reduction adversaries provide their given adversaries with direct access to PPRIM. Rather than having a $\mathsf{S.Ls}$ algorithm, SIM*-AC simulators will provide their given underlying simulators with direct access to PPRIM. Otherwise the analysis follows as given.

In fact, in places where multiple SIM-AC primitive had to use separate ideal primitives, this black-box use of the primitives allow them to share the same primitive for SIM*-AC security without any extra effort.

Moreover, the only way in which constructed simulators depended on adversaries was through dependance on given simulators for the constituent algorithms (which were allowed to depend on the adversary per SIM-AC security). As such, there is no issue when using the order of quantification required for SIM-AC rather than wSIM*-AC security. Hence the following results hold.

[13] This follows from the counter-example described above where we construct a scheme which is trivially insecure if a collision in the random oracle is known.

Lemma 2. *Let $x \in \{\varepsilon, w\}$. Then the following hold.*

- *If* SE *is xSIM*-AC-CPA and INT-CTXT* secure with* P*, then* SE *is xSIM*-AC-CCA secure with* P*.*
- *If* SE *is xSIM*-AC-CPA secure with* P *and* F *is UF-CMA* secure with* P*, then* $(\mathsf{SE}, \mathsf{F})$ *encrypt-then-mac is xSIM*-AC-CCA secure with* P*.*
- *If* $\mathsf{SE}[\cdot]$ *is IND-AC-EXT secure and* F *is xSIM*-AC-PRF secure with* P*, then* $\mathsf{SE}[\mathsf{F}]$ *is xSIM*-AC-\$ secure with* P*.*

This last result covers modes of operation such as counter (CTR), cipher-block chaining (CBC), cipher feedback (CFB), and output feedback (OFB) mode.

The asterisks added to INT-CTXT and UF-CMA indicate that we need these security notions to hold *even for adversaries who are able to program the ideal primitive*. We note, for example, that UF-CMA* security is implied by SIM*-AC-PRF security. We similarly expect that schemes which are known to achieve INT-CTXT security when constructed from a PRF secure function family can be shown by essentially the same proof to achieve INT-CTXT* security when using a SIM*-AC-PRF secure function family.

6.3 Low-level Proofs

For the third category, Jaeger and Tyagi used information theoretic analysis to show that random oracles are SIM-AC-PRF secure, ideal ciphers are SIM-AC-PRF secure, and the ideal encryption model [27] is SIM-AC-AE secure.[14]

To re-establish these results one technically would have to re-write the proofs. We will sketch how to modify the SIM-AC proofs for the first two of these.

Lemma 3. *Random oracles are SIM*-AC-PRF secure (assuming $|\mathcal{K}_\lambda|$ is super-polynomial) and ideal ciphers are SIM*-AC-PRF secure (assuming $|\mathcal{K}_\lambda|$ and $|\mathcal{D}_\lambda|$ are super-polynomial).*

The simulators given for both work by honestly simulating the ideal primitive except whenever a new users is exposed they sample the key at random and then program the primitive to be consistent with the random values returned by earlier evaluation queries. This can done with the more restricted SIM*-AC syntax for simulators. These simulators do not depend on the chosen adversary.

If $\mathsf{F} : \mathcal{K}_\lambda \times \mathcal{D}_\lambda \to \mathcal{R}_\lambda$, their analysis showed that

$$\mathsf{Adv}^{\mathsf{sim^*\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S}_{\mathsf{prf}},\mathsf{P}_{\mathsf{rom}},\mathcal{A}}(\lambda) \le \frac{u_\lambda^2}{2|\mathcal{K}_\lambda|} + \frac{u_\lambda p_\lambda}{2|\mathcal{K}_\lambda|} \text{ and}$$

$$\mathsf{Adv}^{\mathsf{sim^*\text{-}ac\text{-}prf}}_{\mathsf{F},\mathsf{S}_{\mathsf{prf}},\mathsf{P}_{\mathsf{icm}},\mathcal{A}}(\lambda) \le \frac{u_\lambda^2}{2|\mathcal{K}_\lambda|} + \frac{u_\lambda p_\lambda}{2|\mathcal{K}_\lambda|} + \frac{q_\lambda^2}{2|\mathcal{D}_\lambda|}.$$

Here u is the number of distinct users $\mathcal{A}$ interacts with, p is the number of ideal primitive queries it makes, and q is the number of evaluation queries it makes.

[14] The last of these is a slight "cheat" as the ideal encryption model does not satisfy their (or our) definitions of what an ideal primitive is.

Each summand represents a bound of the probability that a bad event occurs which could let an adversary distinguish the real and simulated worlds. The first corresponds to distinct users choosing the same random key. The second corresponds to the attacker making an ideal primitive query with an unexposed user's key. The third corresponds to random outputs of Ev colliding. A useful proof flow for this would introduce a notion of SIM*-AC-PRP security then prove that it is achieved by an ideal cipher and equivalent to SIM*-AC-PRF security up to the birthday bound. We sketch this in the full version.

For SIM*-AC-PRF/PRP security of these constructions the only additional bad event we could have to analyze is the probability that the attacker happening to make an ideal model *programming* query using an unexposed user's key. If p'_λ denotes the number of programming queries the attacker makes, this just adds an additional term of $0.5u_\lambda p'_\lambda/|\mathcal{K}_\lambda|$ to either bound. Alternatively, we could leave the bound unchanged and redefine p_λ to include programming queries as well.

Acknowledgements We thank Nirvan Tyagi for their collaboration on [23], early drafts of which considered asymmetric encryption. Some definitions and results in Section 5 grew out of discussions about those drafts.

References

1. Bellare, M.: New proofs for NMAC and HMAC: Security without collision-resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2006). https://doi.org/10.1007/11818175_36
2. Bellare, M., Bernstein, D.J., Tessaro, S.: Hash-function based PRFs: AMAC and its multi-user security. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 566–595. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016). https://doi.org/10.1007/978-3-662-49890-3_22
3. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 1996). https://doi.org/10.1007/3-540-68697-5_1
4. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: The cascade construction and its concrete security. In: 37th FOCS. pp. 514–523. IEEE Computer Society Press, Burlington, Vermont (Oct 14–16, 1996). https://doi.org/10.1109/SFCS.1996.548510
5. Bellare, M., Davis, H., Günther, F.: Separate your domains: NIST PQC KEMs, oracle cloning and read-only indifferentiability. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 3–32. Springer, Heidelberg, Germany, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45724-2_1
6. Bellare, M., Dowsley, R., Waters, B., Yilek, S.: Standard security does not imply security against selective-opening. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 645–662. Springer, Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012). https://doi.org/10.1007/978-3-642-29011-4_38

7. Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg, Germany, Cologne, Germany (Apr 26–30, 2009). https://doi.org/10.1007/978-3-642-01001-9_1
8. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press, Fairfax, Virginia, USA (Nov 3–5, 1993). https://doi.org/10.1145/168588.168596
9. Böhl, F., Hofheinz, D., Kraschewski, D.: On definitions of selective opening security. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 522–539. Springer, Heidelberg, Germany, Darmstadt, Germany (May 21–23, 2012). https://doi.org/10.1007/978-3-642-30057-8_31
10. Camenisch, J., Drijvers, M., Gagliardoni, T., Lehmann, A., Neven, G.: The wonderful world of global random oracles. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 280–312. Springer, Heidelberg, Germany, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78381-9_11
11. Camenisch, J., Lehmann, A., Neven, G., Samelin, K.: Virtual smart cards: How to sign with a password and a server. In: Zikas, V., De Prisco, R. (eds.) SCN 16. LNCS, vol. 9841, pp. 353–371. Springer, Heidelberg, Germany, Amalfi, Italy (Aug 31 – Sep 2, 2016). https://doi.org/10.1007/978-3-319-44618-9_19
12. Camenisch, J., Lehmann, A., Neven, G., Samelin, K.: UC-secure non-interactive public-key encryption. In: Köpf, B., Chong, S. (eds.) CSF 2017 Computer Security Foundations Symposium. pp. 217–233. IEEE Computer Society Press, Santa Barbara, CA, USA (Aug 21–25, 2017). https://doi.org/10.1109/CSF.2017.14
13. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: 28th ACM STOC. pp. 639–648. ACM Press, Philadephia, PA, USA (May 22–24, 1996). https://doi.org/10.1145/237814.238015
14. Cash, D., Jaeger, J., Jarecki, S., Jutla, C.S., Krawczyk, H., Rosu, M.C., Steiner, M.: Dynamic searchable encryption in very-large databases: Data structures and implementation. In: NDSS 2014. The Internet Society, San Diego, CA, USA (Feb 23–26, 2014)
15. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing **33**(1), 167–226 (2003)
16. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of Cryptology **26**(1), 80–101 (Jan 2013). https://doi.org/10.1007/s00145-011-9114-1
17. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM **33**(4), 792–807 (Oct 1986)
18. Hazay, C., Patra, A., Warinschi, B.: Selective opening security for receivers. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 443–469. Springer, Heidelberg, Germany, Auckland, New Zealand (Nov 30 – Dec 3, 2015). https://doi.org/10.1007/978-3-662-48797-6_19
19. Heuer, F., Poettering, B.: Selective opening security from simulatable data encapsulation. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 248–277. Springer, Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016). https://doi.org/10.1007/978-3-662-53890-6_9
20. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS,

vol. 10677, pp. 341–371. Springer, Heidelberg, Germany, Baltimore, MD, USA (Nov 12–15, 2017). https://doi.org/10.1007/978-3-319-70500-2_12
21. Hofheinz, D., Rao, V., Wichs, D.: Standard security does not imply indistinguishability under selective opening. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 121–145. Springer, Heidelberg, Germany, Beijing, China (Oct 31 – Nov 3, 2016). https://doi.org/10.1007/978-3-662-53644-5_5
22. Hövelmanns, K.: Generic constructions of quantum-resistant cryptosystems. Ph.D. thesis, Dissertation, Bochum, Ruhr-Universität Bochum, 2020 (2021), https://doi.org/10.13154/294-7758
23. Jaeger, J., Tyagi, N.: Handling adaptive compromise for practical encryption schemes. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 3–32. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56784-2_1
24. Jarecki, S., Krawczyk, H., Xu, J.: OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 456–486. Springer, Heidelberg, Germany, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78372-7_15
25. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2002). https://doi.org/10.1007/3-540-45708-9_8
26. Panjwani, S.: Tackling adaptive corruptions in multicast encryption protocols. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 21–40. Springer, Heidelberg, Germany, Amsterdam, The Netherlands (Feb 21–24, 2007). https://doi.org/10.1007/978-3-540-70936-7_2
27. Tyagi, N., Mughees, M.H., Ristenpart, T., Miers, I.: BurnBox: Self-revocable encryption in a world of compelled access. In: Enck, W., Felt, A.P. (eds.) USENIX Security 2018. pp. 445–461. USENIX Association, Baltimore, MD, USA (Aug 15–17, 2018)
28. Wee, H.: Zero knowledge in the random oracle model, revisited. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 417–434. Springer, Heidelberg, Germany, Tokyo, Japan (Dec 6–10, 2009). https://doi.org/10.1007/978-3-642-10366-7_25