A Comparitive Study of Achievability of Security against Related-Key Attack

Mihir Bellare[1] David Cash[2] Rachel Miller[3]

May 2011

Abstract

This paper begins with a practical contribution, namely a way to leverage the RKA security of blockciphers to provide RKA security for a suite of high-level primitives. This motivates a more general theoretical question, namely, when is it possible to transfer RKA security from a primitive P_1 to a primitive P_2? We provide both positive and negative answers. What emerges is a broad and high level picture of the way achievability of RKA security varies across primitives, showing, in particular, that some primitives resist "more" RKAs than others. A technical challenge was to achieve RKA security even for the practical classes of related-key deriving (RKD) functions underlying fault injection attacks that fail to satisfy the "claw-freeness" assumption made in previous works. We surmount this barrier for the first time based on the construction of PRGs that are not only RKA secure but satisfy a new notion of identity collision resistance.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `mihir@cs.ucsd.edu`. URL: `http://www.cs.ucsd.edu/users/mihir`. Supported in part by NSF grants CNS-0904380 and CCF-0915675.

[2] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `cdc@ucsd.edu`. URL: `http://www.cs.ucsd.edu/users/cdcash`. Supported in part by NSF grant CCF-0915675.

[3] Department of Electrical Engineering and Computer Science, MIT, 32 Vassar Street, Cambridge, Massachusetts 02139, USA. Email: `rmiller@csail.mit.edu`. URL: `http://people.csail.mit.edu/rmiller/`. Supported by a DOD NDSEG Graduate Fellowship.

1 Introduction

By fault injection [14, 7] or other means, it is possible for an attacker to induce modifications in a hardware-stored key. When it subsequently observes the outcome of the cryptographic primitive under this modified key, we have a related-key attack (RKA) [18].

The key might be a signing key of a certificate authority or SSL server; a master key for an IBE system; or someone's decryption key. Once viewed merely as a way to study the security of blockciphers [6, 25], RKAs emerge as real threats in practice and of interest for primitives beyond blockciphers.

It becomes of interest, accordingly, to achieve (provable) RKA security for popular high-level primitives. How can we do this?

Practical contributions. One approach to building RKA secure high-level primitives is to do so directly, based, say, on standard number-theoretic assumptions. This, however, is likely to yield a bunch of ad hoc results about security against classes of attacks tied to the scheme algebra that are unlikely to reflect attacks in practice.

We take a different approach. RKA security is broadly accepted in practice as a requirement for blockciphers. AES was designed with the explicit goal of resisting RKAs. We currently have blockciphers whose resistance to RKAs is backed by fifteen years of cryptanalytic and design effort. We propose to leverage this.

We will provide general and systematic ways to immunize any given instance of a high-level primitive against RKAs with the aid of an RKA-secure blockcipher, modeling the latter, for the purpose of proofs, as a RKA-secure PRF [4]. We will do this not only for "minicrypt" primitives that are "close" to PRFs like symmetric encryption, but even for public-key encryption, signatures and identity-based encryption. The methods are cheap and non-intrusive from the software perspective and able to completely transfer all the RKA security of the blockcipher, so that the high-level primitive resists attacks of the sort that arise in practice.

Theoretical contributions. The ability to transfer RKA security from PRFs to other primitives lead us to ask a broader theoretical question, namely, when is it possible to transfer RKA security from a primitive P_1 to a primitive P_2? We provide positive results across quite diverse primitives, showing, for example, that RKA secure IBE implies RKA secure IND-CCA PKE. We also provide negative results showing, for example that RKA secure signatures do not imply RKA secure PRFs.

All our results are expressed in a compact, set-based framework. For any primitive P and class Φ of related-key deriving functions —functions the adversary is allowed to apply to the target key to get a related key— we define what it means for an instance of P to be Φ-RKA secure. A transfer of RKA security from P_1 to P_2 means a construction of a Φ-RKA secure instance of P_2 given a normal-secure instance of P_2 and a Φ-RKA secure instance of P_1 and is expressed compactly as a set containment $\mathbf{RKA}[\mathsf{P}_1] \subseteq \mathbf{RKA}[\mathsf{P}_2]$. Complementing this are non-containments of the form $\mathbf{RKA}[\mathsf{P}_2] \not\subseteq \mathbf{RKA}[\mathsf{P}_1]$, which show the existence of Φ such that there exists a Φ-RKA instance of P_2 yet *no* instance of P_1 can be Φ-RKA secure, indicating, in particular, that RKA security cannot be transferred from P_2 to P_1.

As Figure 1 shows, we pick and then focus on a collection of central and representative cryptographic primitives. We then establish these containment and non-containment relations in a comprehensive and systematic way. What emerges is a broad and high level picture of the way achievability of RKA security varies across primitives, showing, in particular, that some primitives resist "more" RKAs than others.

We view these relations between $\mathbf{RKA}[\mathsf{P}]$ sets as an analog of complexity theory, where we study relations between complexity classes in order to better understand the computational complexity of particular problems. Let us now look at all this more closely.

Background. Related-key attacks were conceived in the context of blockciphers [6, 25]. The first definitions were accordingly for PRFs [4]. For $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ they consider the game that picks a random challenge bit b and random target key $K \in \mathcal{K}$, then picks, for each $L \in \mathcal{K}$, a random function $G(L, \cdot)\colon \mathcal{D} \to \mathcal{R}$, and allows the adversary multiple queries to an oracle that, given a pair (ϕ, x), where

ϕ: $\mathcal{K} \to \mathcal{K}$ and $x \in \mathcal{D}$, returns $F(\phi(K), x)$ if $b = 1$ and $G(\phi(K), x)$ if $b = 0$. They say that F is Φ-RKA secure, where Φ is a class of functions mapping $\mathcal{K}$ to $\mathcal{K}$, if the adversary has low advantage in predicting b when it is allowed, in its queries, to only use functions ϕ from Φ.

We let $\mathbf{RKA}[\mathsf{PRF}]$ be the set of all Φ for which there exists a Φ-RKA secure PRF. Which Φ are in this set? All the evidence so far is that this question has no simple answer. Bellare and Kohno [4] gave natural examples of Φ not in $\mathbf{RKA}[\mathsf{PRF}]$, showing the set is not universal. Membership of certain specific Φ in $\mathbf{RKA}[\mathsf{PRF}]$ has been shown by explicit constructions of Φ-RKA PRFs, first under novel assumptions [27] and then under standard assumptions [3]. Beyond this we must rely on cryptanalysis. Modern blockciphers including AES are designed with the stated goal of RKA security. Accordingly we are willing to assume their Φ-RKA security —meaning that $\Phi \in \mathbf{RKA}[\mathsf{PRF}]$— for whatever Φ cryptanalysts have been unable to find an attack.

Beyond PRFs. Consideration of RKAs is now expanding to primitives beyond PRFs [19, 2, 21]. This is viewed partly as a natural extension of the questions on PRFs. It is also motivated by the view of RKAs as a class of sidechannel attacks [18]. An RKA results when the attacker alters a hardware-stored key via tampering or fault injection [14, 7] and observes the result of the evaluation of the primitive on the modified key. The concern that such attacks could be mounted on a signing key of a certificate authority or SSL server, a master key for an IBE system, or decryption keys of users makes achieving RKA security interesting for a wide range of high-level primitives.

Definitions. We focus on a small but representative set of application-relevant primitives for which interesting variations in achievability of RKA security emerge. These are Sig (Signatures), $\mathsf{PKE\text{-}CCA}$ (CCA-secure public key encryption), $\mathsf{SE\text{-}CCA}$ (CCA-secure symmetric encryption), $\mathsf{SE\text{-}CPA}$ (CPA-secure symmetric encryption), IBE (identity-based encryption) and wPRF (weak PRFs [29]). We define what it means for an instance of P to the Φ-RKA secure for each $\mathsf{P} \in \{\mathsf{wPRF}, \mathsf{IBE}, \mathsf{Sig}, \mathsf{SE\text{-}CCA}, \mathsf{SE\text{-}CPA}, \mathsf{PKE\text{-}CCA}\}$. We follow the definitional paradigm of [4] but there are some delicate primitive-dependent choices that significantly affect the strength of the definitions and the challenge of achieving them (cf. Section 2). We let $\mathbf{RKA}[\mathsf{P}]$ be the set of all Φ for which there exists a Φ-RKA secure instance of P. These sets are all non-trivial.

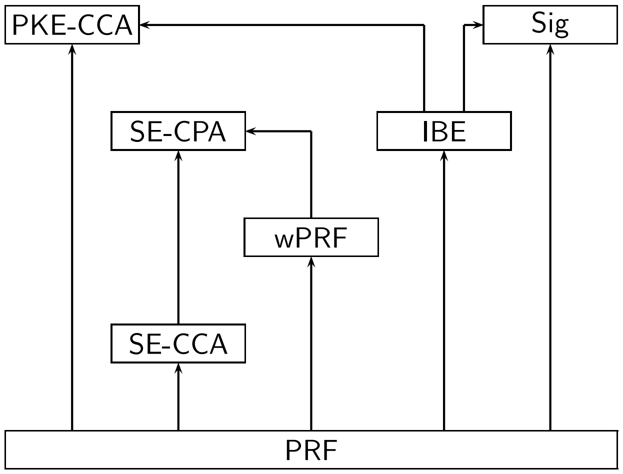
Relations. We establish two kinds of relations between sets $\mathbf{RKA}[\mathsf{P}_1]$, $\mathbf{RKA}[\mathsf{P}_2]$:

- Containment: A proof that $\mathbf{RKA}[\mathsf{P}_1] \subseteq \mathbf{RKA}[\mathsf{P}_2]$ that we establish by taking a Φ-RKA secure instance of P_1 and constructing a Φ-RKA secure instance of P_2, usually under the (minimal) additional assumption that one is given a normal-secure instance of P_2. Containments yield constructions of Φ-RKA secure instances of P_2.
- Non-containment: A proof that $\mathbf{RKA}[\mathsf{P}_2] \not\subseteq \mathbf{RKA}[\mathsf{P}_1]$. Here we exhibit a particular Φ for which we (1) construct a Φ-RKA secure instance of P_1 under some reasonable assumption, and (2) show, via attack, that *any* instance of P_2 is Φ-RKA insecure.

We show that RKA-secure PRFs are powerful enablers of RKA-security: given a Φ-RKA PRF and a normal-secure instance of P we construct a Φ-RKA secure instance of P for all $\mathsf{P} \in \{\mathsf{wPRF}, \mathsf{IBE}, \mathsf{Sig}, \mathsf{SE\text{-}CCA}, \mathsf{SE\text{-}CPA}, \mathsf{PKE\text{-}CCA}\}$. This is represented by the string of containments in the first row of the table in Figure 1. On the practical side, instantiating the PRF with a blockcipher yields a cheap way to immunize the other primitives against RKAs. On the theoretical side, instantiating the PRF with the construct of [3] yields Φ-RKA secure instances of the other primitives based on standard assumptions.

The separations shown in the first column of the table of Figure 1, however, also show that RKA-PRFs are overkill: *all* the other primitives admit Φ-RKA secure instances for a Φ for which no Φ-RKA PRF exists. This leads one to ask whether there are alternative routes to RKA-secure constructions of beyond-PRF primitives.

We show that IBE is a particularly powerful starting point. We observe that Naor's transform preserves RKA-security, allowing us to turn a Φ-RKA secure IBE scheme into a Φ-RKA secure Sig scheme. Similarly, we show that the transform of Boneh, Canetti, Halevi and Katz (BCHK) [13] turns



	PRF	wPRF	IBE	Sig	SE-CCA	SE-CPA	PKE-CCA
PRF	$\subseteq$	$\subseteq$	$\subseteq$	$\subseteq$	$\subseteq$	$\subseteq$	$\subseteq$
wPRF	$\not\subseteq$	$\subseteq$	$\not\subseteq$			$\subseteq$	$\not\subseteq$
IBE	$\not\subseteq$	$\not\subseteq$	$\subseteq$	$\subseteq$	$\not\subseteq$	$\not\subseteq$	$\subseteq$
Sig	$\not\subseteq$	$\not\subseteq$		$\subseteq$	$\not\subseteq$	$\not\subseteq$	$\not\subseteq$
SE-CCA	$\not\subseteq$				$\subseteq$	$\subseteq$	
SE-CPA	$\not\subseteq$		$\not\subseteq$		$\not\subseteq$	$\subseteq$	$\not\subseteq$
PKE-CCA	$\not\subseteq$	$\not\subseteq$			$\not\subseteq$	$\not\subseteq$	$\subseteq$

Figure 1: **Relations between $\mathbf{RKA}[\mathsf{P}]$ classes.** A containment $\mathbf{RKA}[\mathsf{P}_1] \subseteq \mathbf{RKA}[\mathsf{P}_2]$ is represented in the picture by an arrow $\mathsf{P}_1 \to \mathsf{P}_2$ and in the table by a "$\subseteq$" in the row P_1, column P_2 entry. A non-containment $\mathbf{RKA}[\mathsf{P}_1] \not\subseteq \mathbf{RKA}[\mathsf{P}_2]$ is represented in the table by a "$\not\subseteq$" in the row P_1, column P_2 entry. The picture does not show non-containments. The picture sometimes shows a redundant containment (for example the arrow $\mathsf{PRF} \to \mathsf{Sig}$ when there is already a path $\mathsf{PRF} \to \mathsf{IBE} \to \mathsf{Sig}$) because it corresponds to an interesting direct construction. A blank entry in the table means we do not know.

a Φ-RKA secure IBE scheme into a Φ-RKA secure PKE-CCA scheme. What lends these transforms well to RKA-security is that they do not change the secret key. We also show that given a Φ-RKA secure wPRF we can build a Φ-RKA secure SE-CPA scheme. (A wPRF is like a PRF except that is only required to be secure on random inputs [29].) These results motivate finding new Φ-RKA secure IBE schemes and wPRFs.

As the table of Figure 1 indicates, we show a number of other non-containments. Sig emerges as a very "RKA-resilient" primitive in the sense that it can be secure against strictly more RKAs than most other primitives. Some of the non-containments, such as $\mathbf{RKA}[\mathsf{PKE\text{-}CCA}] \not\subseteq \mathbf{RKA}[\mathsf{SE\text{-}CPA}]$ might seem odd; doesn't PKE always imply SE? What we are saying is that the trivial transformation of a PKE scheme to an SE one does not preserve RKA-security and, moreover, there are Φ for which *no transform exists* that can do this.

Claws ok. All previous constructions of Φ-RKA secure primitives [4, 27, 3, 19, 2, 21, 22] assume Φ is claw-free (distinct functions in ϕ disagree on all inputs) because it is hard to do the proofs otherwise, but the Φ underlying practical fault injection attacks are not claw-free, making it desirable to get constructions avoiding this assumption. For the first time, we are able to do this. In Section 2 we explain the technical difficulties and sketch our solution, which is based on the construction of a Φ-RKA PRG that has a novel property we call identity collision resistance (ICR), a variant of the collision-resistance property from [23].

Related work. The first theoretical treatment of RKAs was by Bellare and Kohno [4]. Being inspired by blockciphers, it was for PRFs and PRPs. They showed examples of classes not in $\mathbf{RKA}[\mathsf{PRF}]$, gave conditions on Φ for ideal ciphers to be Φ-RKA secure, and provided standard model constructs for some limited classes. Subsequently, constructions of Φ-RKA secure PRFs and PRPs for more interesting Φ were found, first under novel assumptions [27] and then under standard assumptions [3], and the results on ideal ciphers were extended in [1].

We are seeing growing interest in RKA security for other-than-PRF primitives. Goldenberg and Liskov [19] study related-secret security of lower-level primitives, namely one-way functions, hardcore bits and pseudorandom generators. Applebaum, Harnik and Ishai [2] defined RKA security for (randomized) symmetric encryption, gave several constructions achieving that definition for interesting Φ and then presented numerous applications. Connections with point obfuscation are made by Bitansky and Canetti [8].

Interest in RKA security for higher-level primitives is evidenced by Goyal, O'Neill and Rao [21, 22], who define correlated-input (CI) hash functions, show how to construct them from the q-DHI assumption based on Boneh-Boyen signatures [11, 12] and the Dodis-Yampolskiy PRF [16], and apply this to get Φ-RKA secure signatures from q-DHI for a class Φ consisting of polynomials over a field of prime order. (They indicate their approach would also work for other primitives.) Their construction is similar to ours. However, their definitions and results, unlike ours, are restricted to claw-free Φ. (As we discuss further in Section 2, many classes of practical interest are not claw-free and achieving security in this case involves additional technical challenges that we surmount.) Also, we start from Φ-RKA-PRFs and thus get in-practice security for any class Φ for which blockciphers provide them, while they start from a number-theoretic assumption and get security for a specific class Φ, related to the scheme algebra, that may not capture practical attacks. We note that their work and ours are concurrent and independent. (Ours was submitted to, and rejected from, Eurocrypt 2011, while theirs was submitted to, and accepted at, TCC 2011.)

Gennaro, Lysyanskaya, Malkin, Micali and Rabin [18] suggest that RKAs may arise by tampering. They show that one can achieve security when related keys are derived via arbitrary key modification, but assuming that an external trusted authority signs the original secret key and installs the signature on the device together with its own public key, the latter being "off limits" to the attacker. (Meaning, the related-key deriving functions.) In our case, no such authority is assumed. The off-limit quantities are confined to pre-installed public parameters. No information that is a function of the parameters and the key is installed on the chip.

Ishai, Prabhakaran, Sahai and Wagner [24] are concerned with tampering of wires in the computation of a circuit while we are concerned with tampering with hardware-stored keys. Dziembowski, Pietrzak and Wichs [17] develop an information theoretic method for preventing tampering and show that a wide class of limited, but non-trivial, Φ can be achieved (unconditionally) for any so-called "interactive stateful system."

2 Technical approach

Before providing formal definitions, constructions and proofs of our many positive and negative results, we would like to illustrate one technical issue, namely the challenges created by Φ that are not claw-free and how we resolve them. Our discussion is for concreteness restricted to the design of Φ-RKA signatures based on Φ-RKA PRFs.

The claw-freeness assumption. All known constructions of Φ-RKA-secure primitives [4, 27, 3, 19, 2, 21, 22] are restricted to Φ that are *claw-free*. This means that any two distinct functions in Φ disagree on *all* inputs. This assumption is made for technical reasons; it seems hard to do simulations and proofs without it. Yet the assumption is undesirable, for many natural and practical classes of functions are *not* claw-free. For example, fault injection might be able to set a certain bit of the key to zero, and if Φ contains the corresponding function and the identity function it is not claw-free. Accordingly it is desirable to avoid this assumption. For the first time we are able to do so, via a new technical approach.

Definitions and issues. The degree to which claw-freeness is embedded in current approaches is made manifest by the fact that the very *definition* of Φ-RKA secure signatures of [21, 22] assumes it and is unachievable without it. Let us take a closer look to see how.

The game picks secret signing key sk and associated public verification key vk. It gives the adversary a signing oracle Sign that, given ϕ, m, where ϕ is required to be in Φ, returns the signature of message m under key $\phi(sk)$. The adversary eventually outputs m, σ. Besides validity of m, σ under vk, winning requires that m be "new," meaning not "previously signed." The delicate question is, how do we define this? The choice of [21, 22] is to disallow signing query id, m, where id is the identity function. But the adversary can easily define a function ϕ that is the identity on all but a negligible fraction of its inputs. A query ϕ, m is then valid since $\phi \neq \mathsf{id}$, but almost always returns the signature σ of m under sk, so the adversary can output m, σ and win. By assuming Φ is claw-free and contains id, [21, 22] ensure that such a ϕ is not in Φ and the attack is ruled out.

Our altered definition of m being "new" is that there was no signing query ϕ, m with $\phi(sk) = sk$. This seems, indeed, the natural requirement, ruling out nothing more than that m was signed under sk.

We now have a much more general definition that is meaningful even for the non claw-free Φ that arise in practice, but it has a subtle feature that makes achieving it a challenge. Namely, *checking whether the adversary won apparently requires knowing sk* for we have to test whether or not $\phi(sk) = sk$. In the reduction proving security, however, we will be designing an adversary B attempting to distinguish "real" or "random" instances of some problem given an adversary A breaking the signature scheme, and B will see if A won, declaring "real" if so and "random" otherwise. But B will be simulating A and will not know sk, so the difficulty is how it can test that A won.

Overview of solution. We start from a Φ-RKA PRF $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ that has what we call a key fingerprint for the identity function. This is a relaxation of the notion of a key fingerprint of [3]. It consists of a vector $\mathbf{w}$ over $\mathcal{D}$ such that for all K and all $\phi \in \Phi$ with $\phi(K) \neq K$ there is some i such that $F(K, \mathbf{w}[i]) \neq F(\phi(K), \mathbf{w}[i])$. This allows us to statistically disambiguate the original key K from other keys. Such fingerprints exist for the Φ-RKA PRFs of [3] and for blockciphers and are thus a mild assumption.

We now turn F into a PRG (Pseudorandom Generator) $\mathcal{G}$ that has two properties. First, it is Φ-RKA secure, meaning the adversary has low advantage in determining the challenge bit b in the game that picks a random target key K and random function R and gives the adversary an oracle Gen that on input ϕ returns $\mathcal{G}(\phi(K))$ if $b = 1$ and $R(\phi(K))$ if $b = 0$. This is of course easily obtained from a Φ-RKA PRF. The new property is to be what we call Φ-ICR (Identity Collision Resistant). This means that it is hard for the adversary to find $\phi \in \Phi$ such that $\phi(K) \neq K$ yet $\mathcal{G}(\phi(K)) = \mathcal{G}(K)$ when K is a hidden key. At first it might seem this follows from Φ-RKA security but Proposition 5.1 shows it does not. However Proposition 5.2 shows how to build a PRG that is both Φ-RKA and Φ-ICR secure from a Φ-RKA PRF with an identity key fingerprint without assuming Φ is claw-free.

We build our Φ-RKA signature scheme from this PRG $\mathcal{G}$ and a base (normal secure) signature scheme, as follows. The secret key of our new signature scheme is a key K for the PRG. The output $\mathcal{G}(K)$ of the PRG on input K is used as coins to run the key-generation algorithm $\mathcal{K}$ of the base signature scheme to get a public key pk which becomes the public key of our scheme, the corresponding secret key being discarded. (Recall the secret key of the new scheme is the PRG key K.) To sign a message m under K, run $\mathcal{G}$ on K to get coins for $\mathcal{K}$, run the latter with these coins to get pk, sk and finally sign m under sk with the base signature scheme. Verification is just as in the base signature scheme.

For the proof we must construct an adversary B breaking the Φ-RKA security of $\mathcal{G}$ given an adversary A breaking the Φ-RKA security of our signature scheme. B thinks of the key K underlying its game as the secret key for our signature scheme and then runs A. When A makes Sign query ϕ, m, adversary B will call its Gen oracle on ϕ and use the result as coins for $\mathcal{K}$ to get a secret key under which it then signs m for A. Eventually A outputs a forgery attempt m, σ. We expect that the assumed security of the base signature scheme will make it unlikely that A's forgery is a winning one when Gen is underlain by a random function. So B would like to test if A's forgery was a winning one, outputting 1 if so and 0 otherwise, to win its game. The difficulty is that it cannot test this because, not knowing K, it cannot test whether or not A made a Sign query ϕ, m with $\phi(K) = K$. The Φ-ICR property of $\mathcal{G}$ comes to the rescue, telling us that whether or not $\phi(K) = K$ may be determined by whether or not the outputs of $\mathcal{G}$ on these two inputs, which B *does* have, are the same.

This sketch still pushes under the rug several subtle details which are dealt with in the full proof of Theorem 6.2, to be found in Appendix D.

3 Preliminaries

Notation. For sets X, Y, Z let $\mathsf{Fun}(X, Y)$ be the set of all functions mapping X to Y, and let $\mathsf{FF}(X, Y, Z) = \mathsf{Fun}(X \times Y, Z)$. The empty string is denoted ε. If $\mathbf{v}$ is a vector then $|\mathbf{v}|$ denotes the number of its coordinates and $\mathbf{v}[i]$ denotes its i-th coordinate, meaning $\mathbf{v} = (\mathbf{v}[1], \ldots, \mathbf{v}[|\mathbf{v}|])$. A (binary) string x is identified with a vector over $\{0,1\}$ so that $|x|$ is its length and $x[i]$ is its i-th bit. If $a_1, \ldots, a_n$ are strings then $a_1 \| \cdots \| a_n$ denotes their concatenation. If S is a set then $|S|$ denotes its size and $s \xleftarrow{\$} S$ the operation of picking a random element of S and calling it s.

Algorithms. Unless otherwise indicated, an algorithm is PT (Polynomial Time) and may be randomized. An adversary is an algorithm. If A is an algorithm and $\mathbf{x}$ is a vector then $A(\mathbf{x})$ denotes the vector $(A(\mathbf{x}[1]), \ldots, A(\mathbf{x}[|\mathbf{x}|]))$. By $y \leftarrow A(x_1, x_2, \ldots; r)$ we denote the operation of running A on inputs $x_1, x_2, \ldots$ and coins $r \in \{0,1\}^*$. We denote by $y \xleftarrow{\$} A(x_1, x_2, \ldots)$ the operation of picking r at random and letting $y \leftarrow A(x_1, x_2, \ldots; r)$. We denote by $[A(x_1, x_2, \ldots)]$ the set of all possible outputs of A on inputs $x_1, x_2, \ldots$. We denote by $k \in \mathbb{N}$ the security parameter and by 1^k its unary encoding. It is assumed that the length of the output of any algorithm A depends only on the lengths of its inputs. In particular we can associate to single-input algorithm A its *output length* ℓ satisfying $|A(x)| = \ell(|x|)$ for all x. If A, B are algorithms then $A \| B$ denotes the algorithm that on any input x returns $A(x) \| B(x)$.

Games. Some of our definitions and proofs are expressed via code-based games [5]. Recall that such a game consists of an Initialize procedure, procedures to respond to adversary oracle queries and a Finalize procedure. A game G is executed with an adversary A as follows. First, Initialize executes on input 1^k and its output is the input to A. Then A executes, its oracle queries being answered by the corresponding procedures of G. When A terminates, its output becomes the input to the Finalize procedure. The output of the latter, denoted G^A, is called the output of the game. We let "$\mathrm{G}^A \Rightarrow d$" denote the event that this game output takes value d. If Finalize is absent it is understood to be the identity function, so the game output is the adversary output. Boolean flags are assumed initialized to false.

4 Classes of RKDFs and RKA-PRFs

Classes of RKDFs. In [4], a class Φ of related-key deriving functions (RKDFs) is a finite set of functions, all with the same domain and range. Our more general, asymptotic treatment requires extending this, in particular to allow the functions to depend on public parameters of the scheme. For us a *class* $\Phi = (\mathcal{P}, \mathcal{Q})$ of RKDFs, also called a RKA specification, is a pair of altoritthms, the second deterministic. On input 1^k, parameter generation algorithm $\mathcal{P}$ produces parameters π. On input π, a key K and a description ϕ of an RKD function, the evaluation algorithm $\mathcal{Q}$ returns either a modified key or $\perp$. We require that for all ϕ, π, either $\mathcal{Q}(\pi, K, \phi) = \perp$ for all K or for no K. We let $\Phi_{\pi,\phi}(\cdot) = \mathcal{Q}(\pi, \cdot, \phi)$. We require that Φ always includes the identity function. (Formally, there is a special symbol id such that $\Phi_{\pi,\mathsf{id}}(K) = K$ for all K, π. This is to ensure that Φ-RKA security always implies normal security.) We let ID be the class consisting of only the identity function, so that ID-RKA security will be normal security.

A scheme (regardless of the primitive) is a tuple $(\overline{\mathcal{P}}, \cdots)$ of algorithms, the first of which is a parameter generation algorithm that on input 1^k returns a string. If ℓ is the output length of $\mathcal{P}$, we say that $\Phi = (\mathcal{P}, \mathcal{Q})$ is *compatible* with the scheme if the string formed by the first $\ell(k)$ bits of the output of $\overline{\mathcal{P}}(1^k)$ is distributed identically to the output of $\mathcal{P}(1^k)$ for all $k \in \mathbb{N}$. This is done so that, in constructing one Φ-RKA primitive from another, we can extend the parameters of the constructed scheme beyond those of the original one without changing the class of RKDFs.

We say that $\Phi = (\mathcal{P}, \mathcal{Q})$ is *claw-free* if $\phi \neq \phi'$ implies $\mathcal{Q}(\pi, K, \phi) \neq \mathcal{Q}(\pi, K, \phi')$ for all π, K. This property has been assumed almost ubiquitously in previous work [4, 27, 19, 3] because of the technical difficulties created by its absence, but its assumption is in fact quite restrictive since many natural

```
proc INITIALIZE // PRF
$\pi \xleftarrow{\$} \mathcal{P}(1^k)$ ; $K \xleftarrow{\$} \mathcal{K}(\pi)$
$b \xleftarrow{\$} \{0,1\}$
Return $\pi$

proc FN($\phi, x$) // PRF
$K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
If $K' = \perp$ then return $\perp$
If $b = 1$ then
    $T[K', x] \leftarrow \mathcal{F}(\pi, K', x)$
If $b = 0$ and $T[K', x] = \perp$ then
    $T[K', x] \xleftarrow{\$} \mathsf{Rng}(\pi)$
Return $T[K', x]$

proc FINALIZE($b'$) // PRF
Return $(b = b')$
```

```
proc INITIALIZE // PRFReal, PRFRand
$\pi \xleftarrow{\$} \mathcal{P}(1^k)$
$K \xleftarrow{\$} \mathcal{K}(\pi)$
Return $\pi$

proc FN($\phi, x$) // PRFReal
$K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
If $K' = \perp$ then return $\perp$
Return $\mathcal{F}(\pi, K', x)$

proc FN($\phi, x$) // PRFRand
$K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
If $K' = \perp$ then return $\perp$
If $T[K', x] = \perp$ then
    $T[K', x] \xleftarrow{\$} \mathsf{Rng}(\pi)$
Return $T[K', x]$
```

```
proc INITIALIZE // IDFP
$\pi \xleftarrow{\$} \mathcal{P}(1^k)$
$K \xleftarrow{\$} \mathcal{K}(\pi)$
$\mathbf{w} \xleftarrow{\$} \mathsf{IKfp}(\pi)$
Return $\pi, \mathbf{w}$

proc FN($\phi$) // IDFP
$K' \leftarrow \Phi_{\pi,\phi}(K)$
If $(K' = \perp)$ then return $\perp$
If $(K' \neq K)$ then
    If $(\mathcal{F}(K', \mathbf{w}) = \mathcal{F}(K, \mathbf{w}))$ then
        WIN $\leftarrow$ true
Return $\mathcal{F}(K', \mathbf{w})$

proc FINALIZE() // IDFP
Return WIN
```

Figure 2: Games defining Φ-RKA PRF security and Φ-IDFP security of function family $\mathcal{FF} = (\mathcal{P}, \mathcal{K}, \mathcal{F})$ having range $\mathsf{Rng}(\cdot)$.

classes do not have it. We are able to remove this assumption and provide constructs secure even for non-claw-free classes via new technical approaches.

The class $\Phi^{\mathsf{const}} = (\mathcal{P}, \mathcal{Q}^{\mathsf{const}})$ of constant functions associated to class $\Phi = (\mathcal{P}, \mathcal{Q})$ is defined by $\Phi^{\mathsf{const}}_{\pi,a}(K) = a$ for all $K, a \in \{0,1\}^*$ and all π. The union $\Phi^1 \cup \Phi^2 = (\mathcal{P}, \mathcal{Q})$ of classes $\Phi^1 = (\mathcal{P}, \mathcal{Q}^1)$ and $\Phi^2 = (\mathcal{P}, \mathcal{Q}^2)$ is defined by having $\mathcal{Q}(\pi, K, \phi)$ parse ϕ as $i \,\|\, \phi^*$ for $i \in \{1, 2\}$ and return $\mathcal{Q}^i(\pi, K, \phi^*)$.

DISCUSSION. In a non-asymptotic treatment, there is no formal line between "secure" and "insecure." This makes it unclear how to rigorously define the sets $\mathbf{RKA}[\mathsf{P}]$. Lead, accordingly, to pursue an asymptotic treatment, we introduce parameter dependence so that we can capture constructs in the literature [27, 3] where RKDFs are defined over a group that is now parameter-dependent rather than fixed. (We note that even in the non-asymptotic case, a treatment like ours is needed to capture constructs in [27] relying on a RSA group defined by random primes. This issue is glossed over in [27].) A dividend of our treatment is a separation between an RKDF and its encoding, the latter being what an adversary actually queries, another issue glossed over in previous work.

FUNCTION FAMILIES. A function family $\mathcal{FF} = (\mathcal{P}, \mathcal{K}, \mathcal{F})$ consists of a parameter generator, a key generator, and an evaluator, the last deterministic. For each $k \in \mathbb{N}$ and $\pi \in [\mathcal{P}(1^k)]$, the scheme also defines PT decidable and sampleable sets $\mathsf{Dom}(\pi)$ and $\mathsf{Rng}(\pi)$ such that $\mathcal{F}(\pi, K, \cdot)$ maps elements of $\mathsf{Dom}(\pi)$ to $\mathsf{Rng}(\pi)$. We assume there are polynomials d, l, called the input and output lengths, respectively, such that $\mathsf{Dom}(\pi) \subseteq \{0,1\}^{d(k)}$ and $\mathsf{Rng}(\pi) \subseteq \{0,1\}^{l(k)}$. Unless otherwise indicated we assume $\mathsf{Rng}(\pi) = \{0,1\}^{l(k)}$ and $l(k) = \omega(\log(k))$ and $|\mathsf{Dom}(\pi)| \geq 2^k$ for all $\pi \in [\mathcal{P}(1^k)]$ and all $k \in \mathbb{N}$.

RKA-PRFS. Let $\mathcal{FF} = (\mathcal{P}, \mathcal{K}, \mathcal{F})$ be a function family as above. Game PRF of Figure 2 is associated to $\mathcal{FF}$ and a RKA specification Φ that is compatible with $\mathcal{FF}$. Let $\mathbf{Adv}^{\text{prf-rka}}_{\mathcal{FF},A,\Phi}(k)$ equal $2\Pr[\text{PRF}^A \Rightarrow \mathsf{true}] - 1$ when the game has input 1^k. We say $\mathcal{FF}$ is Φ-RKA secure if this advantage function is negligible. For our proofs it is useful to also consider games PRFReal, PRFRand of Figure 2. Standard arguments imply that $\mathbf{Adv}^{\text{prf-rka}}_{\mathcal{FF},A,\Phi}(k)$ equals $\Pr\left[\text{PRFReal}^A \Rightarrow 1\right] - \Pr\left[\text{PRFRand}^A \Rightarrow 1\right]$.

IDENTITY KEY FINGERPRINTS. An identity key fingerprint function with vector length $v(\cdot)$ for $\mathcal{FF} = (\mathcal{P}, \mathcal{K}, \mathcal{F})$ is an algorithm IKfp that for every $\pi \in [\mathcal{P}(1^k)]$ and every $k \in \mathbb{N}$ returns, on input π, a $v(k)$-vector over $\mathsf{Dom}(\pi)$ all of whose coordinates are distinct. Game IDFP of Figure 2 is associated to $\mathcal{FF}$ and a RKA specification $\Phi = (\mathcal{P}, \mathcal{Q})$ that is compatible with $\mathcal{FF}$. Let $\mathbf{Adv}^{\text{idfp}}_{\mathcal{FF},A,\Phi}(k)$ equal $\Pr[\text{IDFP}^A \Rightarrow \mathsf{true}]$ when the game has input 1^k. We say $\mathcal{FF}$ is Φ-IDFP secure if this advantage function is negligible.

The key fingerprint notion of [3] can be seen as allowing statistical disambiguation of any pair of keys. They showed that the Naor-Reingold PRF NR had such a fingerprint, but in general, it does not seem common. Interestingly, their own Φ-RKA PRFs, which build on NR, are not known to have such a fingerprint. Our relaxation can be seen as asking for computational disambiguation of the original key from other keys, and ends up being much easier to achieve. In particular, such fingerprints exist for the constructs of [3]. This is a consequence of something more general, namely that any Φ-RKA secure PRF with large enough range is Φ-IDFP secure if Φ is claw-free, using *any* point in the domain functioning as the fingerprint. This is formalized by Proposition 4.1 below, with a proof in Appendix A. Φ-IDFP security for the constructs of [3] follows as the Φ they use is claw-free.

Proposition 4.1 *Suppose Φ is claw-free and $\mathcal{FF}$ is a Φ-RKA secure PRF with associated domain $\mathsf{Dom}(\cdot)$ and super-polynomial size range $\mathsf{Rng}(\cdot)$. Let IKfp be any algorithm that on input π returns a 1-vector over $\mathsf{Dom}(\pi)$. Then $\mathcal{FF}$ is Φ-IDFP secure.*

In practice Φ-IDFP security seems like a mild assumption even when Φ is not claw-free. A vector of a few, distinct domain points ought to be a suitable fingerprint for any practical blockcipher. This does not follow from a standard assumption on it such as PRF, but is consistent with properties assumed by cryptanalysts and can be proved in the ideal cipher model.

Φ-IDFP security of given Φ-RKA PRFs, even for non-claw-free Φ, will be important in the constructions underlying our containment results, and we make it a default assumption on a Φ-RKA PRF. The above shows that this is a mild and reasonable assumption.

RKA SETS. We say that an RKA specification $\Phi = (\mathcal{P}, \mathcal{Q})$ is achievable for the primitive PRF if there exists a Φ-RKA and Φ-IDFP secure PRF that is compatible with Φ. We let $\mathbf{RKA}[\mathsf{PRF}]$ be the set of all Φ that are achievable for PRF.

WHAT CAN ATTACKS MODIFY? We view the system as a whole as having the following components: algorithms (code), parameters, public keys (if any) and secret keys. Of these, our convention is that only secret keys are subject to RKAs. This is not the only possible model, nor is it necessarily the most realistic if considering tampering attacks in practice, but it is a clear and interesting one with some justification. Parameters are systemwide, meaning fixed beforehand and independent of users, and may, in an implementation, be part of the algorithm code. Public keys are accompanied by certificates under a CA public key that is in the parameters, so if parameters are safe, so are public keys. This leaves secret keys as the main target. One consequence of this is that in a public key setting the attack is only on the holder of the secret key, meaning the signer for signatures and the receiver for encryption, while in the symmetric setting, both sender and receiver are under attack, making this setting more complicated.

We could consider attacks on public keys but these are effectively attacks on parameters. Furthermore the only way for them to succeed is to modify the CA public key in the parameters in a rather special way, replacing it by some other key under which the attack produces signatures for the modified public key. "Natural" attacks caused by fault-injection are unlikely to do this, further support for our convention of confining attacks to secret keys.

5 ICR PRGs: A tool in our constructions

We will be using Φ-RKA PRFs to build Φ-RKA instances of many other primitives. An important technical difficulty will be to avoid assuming Φ is claw-free, which would restrict the result. A tool we will introduce and use for this purpose is a Φ-RKA PRG satisfying a weak form of collision resistance under RKA that we call Φ-ICR. In this section we define this primitive and show how to achieve it based on a Φ-RKA and Φ-IDFP secure PRF.

RKA PRGS. A PRG $\mathcal{PRG} = (\mathcal{P}, \mathcal{K}, \mathcal{G}, r)$ is specified by a parameter generation algorithm, a key generation algorithm, an evaluation algorithm and an output length $r(\cdot)$. Game PRG of Figure 3 is

```
proc Initialize // PRG
    $\pi \xleftarrow{\$} \mathcal{P}(1^k)$
    $K \xleftarrow{\$} \mathcal{K}(\pi)$ ; $b \xleftarrow{\$} \{0,1\}$
    Return $\pi$

proc Gen($\phi$) // PRG
    $K' \leftarrow \Phi_{\pi,\phi}(K)$
    If $K' = \perp$ then return $\perp$
    If $T[K'] = \perp$ then
        If $b = 1$ then $T[K'] \leftarrow \mathcal{G}(\pi, K')$
        Else $T[K'] \xleftarrow{\$} \{0,1\}^{r(k)}$
    Return $T[K']$

proc Finalize($b'$) // PRG
    Return $(b = b')$
```

```
proc Initialize // ICR
    $\pi \xleftarrow{\$} \mathcal{P}(1^k)$ ; $i \leftarrow 0$
    $K \xleftarrow{\$} \mathcal{K}(\pi)$ ; $T_0 \leftarrow \mathcal{G}(\pi, K)$
    Return $\pi$

proc Gen($\phi$) // ICR
    $K' \leftarrow \Phi_{\pi,\phi}(K)$
    If $K' = \perp$ then return $\perp$
    $S \leftarrow \mathcal{G}(\pi, K')$
    If $((S = T_0) \wedge (K \neq K'))$ then Win $\leftarrow$ true
    Return $S$

proc Finalize() // ICR
    Return Win
```

Figure 3: Games defining Φ-RKA security and identity-collision-resistance for PRG $\mathcal{PRG} = (\mathcal{P}, \mathcal{K}, \mathcal{G}, r)$.

associated to $\mathcal{PRG}$ and a RKA specification Φ that is compatible with $\mathcal{PRG}$. Let $\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG},A,\Phi}(k)$ equal $2\Pr[\mathrm{PRG}^A \Rightarrow \mathsf{true}] - 1$ when the game has input 1^k. We say $\mathcal{PRG}$ is Φ-RKA secure if this advantage function is negligible.

We clarify that unlike a normal PRG [9], we don't require a Φ-RKA to be length extending, meaning that outputs need not be longer than inputs. The utility of the primitive is getting RKA security, for which length extension is irrelevant. If one does want a length extending Φ-RKA PRG (we won't) one can get it by applying a normal-secure PRG to the output of a given Φ-RKA PRG.

ICR. We define and use a weak form of collision-resistance for PRGs which requires that the adversary be unable to find ϕ so that $\Phi_{\pi,\phi}(K) \neq K$ yet $\mathcal{G}(\Phi_{\pi,\phi}(K)) = \mathcal{G}(K)$. Game ICR of Figure 3 is associated to $\mathcal{PRG}$ and a RKA specification Φ that is compatible with $\mathcal{PRG}$. Let $\mathbf{Adv}^{\mathrm{icr}}_{\mathcal{PRG},C,\Phi}(k)$ equal $2\Pr[\mathrm{ICR}^C \Rightarrow \mathsf{true}] - 1$ when the game has input 1^k. We say $\mathcal{PRG}$ is Φ-ICR (Identity Collision Resistance) secure if this advantage function is negligible.

Does RKA security imply ICR security? At first glance it would seem that if a PRG $\mathcal{PRG} = (\mathcal{P}, \mathcal{K}, \mathcal{G}, r)$ is Φ-RKA secure then it is also Φ-ICR secure. Indeed, suppose an adversary has ϕ such that $\Phi_{\pi,\phi}(K) \neq K$ yet $\mathcal{G}(\Phi_{\pi,\phi}(K)) = \mathcal{G}(K)$. Let it query $R_0 \leftarrow \mathrm{Gen}(\mathsf{id})$ and $R_1 \leftarrow \mathrm{Gen}(\phi)$ and return 1 if $R_0 = R_1$ and 0 otherwise. In the real $(b = 1)$ case R_0, R_1 are equal but in the random $(b = 0)$ case they would appear very unlikely to be equal, so that that this strategy would appear to have high advantage in breaking the Φ-RKA security of $\mathcal{PRG}$. The catch is in our starting assumption, which made it appear that $\Phi_{\pi,\phi}(K) \neq K$ yet $\mathcal{G}(\Phi_{\pi,\phi}(K)) = \mathcal{G}(K)$ was an absolute fact, true both for $b = 0$ and $b = 1$. If $\Phi_{\pi,\phi}(K)$ and K are different in the real game but equal in the random game, the adversary sees an output collision in both cases and its advantage disappears. Can this actually happen? It can, and indeed the claim (that Φ-RKA security implies Φ-ICR security) is actually false:

Proposition 5.1 *Suppose there exists a normal-secure PRG $\overline{\mathcal{PRG}} = (\overline{\mathcal{P}}, \overline{\mathcal{K}}, \overline{\mathcal{G}}, r)$ with $r(\cdot) = \omega(\log(\cdot))$. Then there exists a PRG $\mathcal{PRG} = (\overline{\mathcal{P}}, \mathcal{K}, \mathcal{G}, r)$ and a class Φ such that $\mathcal{PRG}$ is Φ-RKA secure but $\mathcal{PRG}$ is not Φ-ICR secure.*

A proof is in Appendix B. Briefly, the constructed PRG $\mathcal{PRG}$ adds a redundant bit to the seed of $\overline{\mathcal{PRG}}$ so that seeds differing only in their first bits yield the same outputs, meaning create non-trivial collisions. But Φ is crafted so that that its members deviate from the identity function only in the real game, so that output collisions appear just as often in both cases but in the real game they are non-trivial while in the random game they are trivial. The difficulty is showing how to do this while retaining Φ-RKA security.

Construction. We saw above that not all Φ-RKA PRGs are Φ-ICR secure. Our containments will

```
proc Initialize // Sig
    $\pi \xleftarrow{\$} \mathcal{P}(1^k)$ ; $M \leftarrow \emptyset$
    $(vk, sk) \xleftarrow{\$} \mathcal{K}(\pi)$
    Return $(\pi, vk)$
proc Sign($\phi, m$) // Sig
    $sk' \leftarrow \Phi_{\pi,\phi}(sk)$
    If $sk' = \perp$ then return $\perp$
    If $(sk' = sk)$ then $M \leftarrow M \cup \{m\}$
    Return $\sigma \xleftarrow{\$} \mathcal{S}(\pi, sk', m)$
proc Finalize($m, \sigma$) // Sig
    Return $((\mathcal{V}(\pi, vk, m, \sigma) = 1) \wedge (m \notin M))$
```

```
proc Initialize // IBE
    $\pi \xleftarrow{\$} \mathcal{P}(1^k)$ ; $(mpk, msk) \xleftarrow{\$} \mathcal{M}(\pi)$
    $b \xleftarrow{\$} \{0,1\}$ ; $id^* \leftarrow \perp$
    Return $(\pi, mpk)$
proc KD($\phi, id$) // IBE
    $msk' \leftarrow \Phi_{\pi,\phi}(msk)$
    If $msk' = \perp$ then return $\perp$
    If $((msk' = msk) \wedge (id = id^*))$
        then return $\perp$
    Return $dk \leftarrow \mathcal{K}(\pi, mpk, msk', id)$
proc LR($id, m_0, m_1$) // IBE
    If $(|m_0| \neq |m_1|)$ then return $\perp$
    $id^* \leftarrow id$
    Return $C \xleftarrow{\$} \mathcal{E}(\pi, mpk, id, m_b)$
proc Finalize($b'$) // IBE
    Return $(b = b')$
```

Figure 4: Games defining Φ-RKA security for primitives Sig, IBE.

rely crucially on ones that are. We obtain them from Φ-RKA PRFs that have key fingerprints for the identity function. The proof is relatively straightforward and is omitted.

Proposition 5.2 *Let $\mathcal{FF} = (\mathcal{P}, \mathcal{K}, \mathcal{F})$ be a Φ-RKA PRF with output length l. Let* IKfp *be a Φ-IDFP secure identity key fingerprint function for $\mathcal{FF}$ with vector length v. Let $r = lv$ and let $\overline{\mathcal{K}}$, on input $\pi \,\|\, \mathbf{w}$, return $\mathcal{K}(\pi)$. Define PRG $\mathcal{PRG} = (\mathcal{P} \,\|\, \mathsf{IKfp}, \overline{\mathcal{K}}, \mathcal{G}, r)$ via*

$$\mathcal{G}(\pi \,\|\, \mathbf{w}, K) = \mathcal{F}(\pi, K, \mathbf{w}[1]) \,\|\, \cdots \,\|\, \mathcal{F}(\pi, K, \mathbf{w}[|\mathbf{w}|]) \ .$$

Then $\mathcal{PRG}$ is Φ-RKA secure and Φ-ICR secure.

6 Relations

We first present a containment and a non-containment related to Sig. Then, we illustrate a non-containment related to wPRFs that uses a different technique. Then we turn to IBE-related results. We then briefly indicate how other relations may be obtained via the same ideas.

Signatures. A signature scheme $\mathcal{DS} = (\mathcal{P}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is specified as usual by its parameter generation, key generation, signing and verifying algorithms. Game Sig of Figure 4 is associated to $\mathcal{DS}$ and a RKA specification Φ that is compatible with $\mathcal{DS}$. Let $\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A,\Phi}(k)$ be the probability that Sig^A returns true when the game has input 1^k. We say $\mathcal{DS}$ is Φ-RKA secure if this advantage function is negligible. Normal security of a signature scheme is recovered by considering Φ that contains only the identity function. One feature of the definition worth highlighting is the way we decide which messages are not legitimate forgeries. They are the ones signed with the real key sk, which means that oracle Sign needs to check when a related key equals the real one and record the corresponding message, which is a source of challenges in reduction-based proofs.

From Φ-RKA PRGs to Φ-RKA signatures. We will prove containments of the form $\mathbf{RKA}[\mathsf{PRF}] \subseteq \mathbf{RKA}[\mathsf{P}]$ by proving $\mathbf{RKA}[\mathsf{PRG}] \subseteq \mathbf{RKA}[\mathsf{P}]$ and exploiting the fact that $\mathbf{RKA}[\mathsf{PRF}] \subseteq \mathbf{RKA}[\mathsf{PRG}]$.

We start with a Φ-RKA PRG $\mathcal{PRG} = (\mathcal{P}, \mathcal{K}, \mathcal{G}, r)$ and a normal-secure signature scheme $\overline{\mathcal{DS}} = (\overline{\mathcal{P}}, \overline{\mathcal{K}}, \overline{\mathcal{S}}, \overline{\mathcal{V}})$ such that $r(\cdot)$ is the number of coins used by $\overline{\mathcal{K}}$. We now build another signature scheme $\mathcal{DS} = (\mathcal{P} \,\|\, \overline{\mathcal{P}}, \mathcal{K}', \mathcal{S}, \mathcal{V})$ as follows:

1. Parameters: Parameters for $\mathcal{DS}$ are the concatenation $\pi \| \overline{\pi}$ of independently generated parameters for $\mathcal{PRG}$ and $\overline{\mathcal{DS}}$.
2. Keys: Pick a random seed $K \xleftarrow{\$} \mathcal{K}(\pi)$ and let $(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(\overline{\pi}; \mathcal{G}(K))$ be the result of generating verifying and signing keys with coins $\mathcal{G}(K)$. The new signing key is K and the verifying key remains $\overline{vk}$. (Key $\overline{sk}$ is discarded.)
3. Signing: To sign message m with signing key K, recompute $(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(\overline{\pi}; \mathcal{G}(K))$ and then sign m under $\overline{\mathcal{S}}$ using $\overline{sk}$.
4. Verifying: Just as in the base scheme, verify that σ is a signature of m under $\overline{vk}$ using $\overline{\mathcal{V}}$.

Signature scheme $\mathcal{DS}$ remains compatible with Φ since the parameters of $\mathcal{PRG}$ prefix those of $\mathcal{DS}$.

We want $\mathcal{DS}$ to inherit the Φ-RKA security of $\mathcal{PRG}$. In fact we will show more, namely that $\mathcal{DS}$ is $(\Phi \cup \Phi_c)$-RKA secure where Φ_c is the class of constant RKDFs associated to Φ. The intuition is deceptively simple. A signing query ϕ, m of an adversary A attacking $\mathcal{DS}$ results in a signature of m under what is effectively a fresh signing key, since it is generated using coins $\mathcal{G}(\phi(K))$ that are computationally independent of $\mathcal{G}(K)$ due to the assumed Φ-RKA security of the PRG. These can accordingly be simulated without access to K. On the other hand, signing queries in which ϕ is a constant function may be directly simulated. The first difficulty is that the adversary attacking the Φ-RKA security of $\mathcal{PRG}$ that we must build needs to know when A succeeds, and for this it needs to know when a derived seed equals the real one, and it is unclear how to do this without knowing the real seed. The second difficulty is that a queried constant might equal the key. We take an incremental approach to showing how these difficulties are resolved, beginning by assuming Φ is claw-free, which makes the first difficulty vanish:

Theorem 6.1 *Let signature scheme $\mathcal{DS} = (\mathcal{P} \| \overline{\mathcal{P}}, \mathcal{K}', \mathcal{S}, \mathcal{V})$ be constructed as above from Φ-RKA PRG $\mathcal{PRG} = (\mathcal{P}, \mathcal{K}, \mathcal{G}, r)$ and normal-secure signature scheme $\overline{\mathcal{DS}} = (\overline{\mathcal{P}}, \overline{\mathcal{K}}, \overline{\mathcal{S}}, \overline{\mathcal{V}})$ and assume Φ is claw-free. Then $\mathcal{DS}$ is $(\Phi \cup \Phi_c)$-RKA secure.*

A proof of Theorem 6.1 is in Appendix C, and the intuition was discussed in Section 2. This result, however, is weaker than we would like, for, as we have already said, many interesting classes are not claw-free. Also, this result fails to prove **RKA**[PRF] $\subseteq$ **RKA**[Sig] since the first set may contain Φ that are not claw-free. To address this we show that the claw-freeness assumption on Φ can be replaced by the assumption that $\mathcal{PRG}$ is Φ-ICR secure:

Theorem 6.2 *Let signature scheme $\mathcal{DS} = (\mathcal{P} \| \overline{\mathcal{P}}, \mathcal{K}', \mathcal{S}, \mathcal{V})$ be constructed as above from Φ-RKA secure and Φ-ICR secure PRG $\mathcal{PRG} = (\mathcal{P}, \mathcal{K}, \mathcal{G}, r)$ and normal-secure signature scheme $\overline{\mathcal{DS}} = (\overline{\mathcal{P}}, \overline{\mathcal{K}}, \overline{\mathcal{S}}, \overline{\mathcal{V}})$. Then $\mathcal{DS}$ is $(\Phi \cup \Phi_c)$-RKA secure.*

A proof of Theorem 6.2 is in Appendix D. This result is much stronger because Proposition 5.2 says we can get the PRGs we want from Φ-RKA PRFs. In particular, Theorem 6.2 establishes the containment **RKA**[PRF] $\subseteq$ **RKA**[Sig]. (Theorem 6.1 only establishes it for the subset of classes that are claw-free.)

Our construction has the advantage that the verification process, and the form of the signatures and public key, are unchanged. This means it has minimal impact on software, making it easier to deploy than a totally new scheme. Signing in the scheme now involves evaluation of a Φ-RKA-PRG but this can be made cheap via an AES-based instantiation. However, signing also involves running the key-generation algorithm $\overline{\mathcal{K}}$ of the base scheme which might be expensive.

This construction also meets a stronger notion of Φ-RKA security where the adversary cannot even forge signature relative to the public keys associated with the derived secret keys. We elaborate on this in Appendix E.

Some base signature schemes lend themselves naturally and directly to immunization against RKAs via Φ-RKA PRFs. This is true for the binary-tree, one-time signature based scheme discussed in [20], where the secret key is already that of a PRF used in the scheme. If this PRF is Φ-RKA secure we can show the signature scheme (unmodified) is too, and moreover also meets the strong version of the definition alluded to above. See Appendix E.

Separating Φ-RKA PRFs from Φ-RKA signatures. Having just shown that $\mathbf{RKA}^*[\mathsf{PRF}] \subseteq \mathbf{RKA}^*[\mathsf{Sig}]$ it is natural to ask whether the converse is true as well, meaning whether the sets are equal. The answer is no, meaning $\mathbf{RKA}^*[\mathsf{Sig}] \not\subseteq \mathbf{RKA}^*[\mathsf{PRF}]$ and thus $\mathbf{RKA}[\mathsf{Sig}] \not\subseteq \mathbf{RKA}[\mathsf{PRF}]$. The interpretation is that there exist Φ such that there exist Φ-RKA secure signatures, but there are *no* Φ-RKA PRFs. An example is when $\Phi = \Phi_c$ is the set of constant functions. Theorem 6.1 implies that there exists a Φ_c-RKA secure signature scheme by setting $\Phi = \emptyset$ in the theorem, so that $\mathcal{PRG}$ need only be a normal-secure PRG. But attacks from [4] show that no PRF can be Φ_c-RKA secure. Thus, this separation is quite easily obtained. We will later see others which are technically more interesting. This separation motivates finding other avenues to Φ-RKA signatures. Below we will show that IBE is one such.

wPRF. We turn to wPRFs so that we can illustrate an interesting separation, namely with PRFs. Let $w\mathcal{PRF} = (\mathcal{P}, \mathcal{K}, \mathcal{F})$ be a family of functions as defined in Section 3 and $\mathsf{Dom}_{w\mathcal{PRF}}(\pi), \mathsf{Rng}_{w\mathcal{PRF}}(\pi)$ the finite domain and range sets associated to parameters π. Game wPRF of Figure 5 is associated to $w\mathcal{PRF}$ and a RKA specification $\Phi = (\pi, \mathcal{Q})$ that is compatible with $w\mathcal{PRF}$. Let $\mathbf{Adv}^{\text{wprf-rka}}_{w\mathcal{PRF},A,\Phi}(k)$ equal $2\Pr[\text{wPRF}^A \Rightarrow \mathsf{true}] - 1$ when the game has input 1^k. We say $w\mathcal{PRF}$ is Φ-RKA secure if this advantage function is negligible.

Separating Φ-RKA PRFs from Φ-RKA wPRFs. Obviously $\mathbf{RKA}[\mathsf{PRF}] \subseteq \mathbf{RKA}[\mathsf{wPRF}]$. We show that the converse is not true, namely $\mathbf{RKA}[\mathsf{wPRF}] \not\subseteq \mathbf{RKA}[\mathsf{PRF}]$. In this case constant RKDFs do *not* provide the separation since both primitives are insecure under these functions. Instead we create RKDFs which only help the attacker if the function may be invoked on the same input for different RKDFs, which is not possible in the wPRF game. Proceeding to the details, let $w\mathcal{PRF} = (\mathcal{P}, \mathcal{K}, \mathcal{F})$ be a normal-secure wPRF and for simplicity assume that keys are random $(k-1)$-bit strings. (Formally, $\mathcal{K}(\pi)$ induces the uniform distribution on $\{0,1\}^{k-1}$ for all $\pi \in [\mathcal{P}(1^k)]$ and all $k \in \mathbb{N}$.) If $K \in \{0,1\}^*$ we let K^- denote the string that is K with the first bit flipped. We construct a class $\Phi = (\mathcal{P}, \mathcal{Q})$ of RKDFs as follows. On input $\pi \in [\mathcal{P}(1^k)]$, key $K \in \{0,1\}^*$ and description ϕ_i, algorithm $\mathcal{Q}$ returns K if $K[i] = 1$ and K^- otherwise, for $1 \le i \le |K|$. Also given π, K and description flip it returns K^-.

Let $\mathcal{PRF} = (\overline{\mathcal{P}}, \overline{\mathcal{K}}, \overline{\mathcal{F}})$ be *any* PRF that is compatible with Φ. We show that it can be broken under a Φ-RKA. Let $\ell(k)$ denote the length of keys when the security parameter is k. Our adversary, given parameters $\overline{\pi}$, picks x at random from $\mathsf{Dom}_{\mathcal{PRF}}(\overline{\pi})$. It queries $y_{\mathsf{id}} \leftarrow \textsc{Fn}(\mathsf{id}, x)$. Then for each $1 \le i \le \ell(k)$ it queries $y_i \leftarrow \textsc{Fn}(\phi_i, x)$ and sets $K'[i] = 1$ if $y_i = y_{\mathsf{id}}$ and 0 otherwise. (It is crucial that all queries use the same x.) The result should be that $K' = K$, meaning A has recovered the key. Once it has the key, it can easily win by making a few queries under id at random inputs and returning 1 if the results are consistent with K' and 0 otherwise.

There is one subtlety above, however. Namely if $\overline{\mathcal{F}}(\overline{\pi}, K, x) = \overline{\mathcal{F}}(\overline{\pi}, K^-, x)$, meaning the functions under keys K and K^- agree at x, then the attack will recover the string $K' = 1^{\ell(k)}$ and not recover the key. This is in general problematic because a PRF may very well have the property that the functions defined by two different keys are the same. It is to resolve this problem that we put flip in Φ. Φ-RKA security now ensures that the functions defined by keys K, K^- look like random, independent ones. Thus, our adversary can simply return 1 if $K' = 1^{\ell(k)}$, meaning declare that the challenge bit was 1.

What remains is to show that there exists a wPRF that is Φ-RKA secure. We construct $\overline{w\mathcal{PRF}} = (\mathcal{P}, \overline{\mathcal{K}}, \overline{\mathcal{F}})$ based on the normal-secure $w\mathcal{PRF} = (\mathcal{P}, \mathcal{K}, \mathcal{F})$ whose existence we assumed at the start. Parameter generation is unchanged. Keys are random k-bit strings. Function $\overline{\mathcal{F}}(\pi, K, x)$ lets L be the last $(k-1)$-bits of K and returns $\mathcal{F}(\pi, L, x)$. The result is that functions $\overline{\mathcal{F}}(\pi, K, \cdot)$ and $\overline{\mathcal{F}}(\pi, K^-, \cdot)$ are identical. In the wPRF case this, rather than contradicting Φ-RKA security as in the PRF case, helps us prove it. We claim we can reduce the Φ-RKA security of $\overline{w\mathcal{PRF}}$ to the normal security of $w\mathcal{PRF}$. The reason is that on input K, all the RKDFs we put in Φ return either K or K^- and thus no matter what RKDFs the adversary queries the result can be simulated by access to $\overline{\mathcal{F}}(\pi, K, \cdot)$. As long as no input is repeated, the simulation will be correct, and since the domain has size at least 2^k and inputs are random rather than adversary-selected, the probability of a repeated input in q queries is at most

```
proc Initialize // PKE
$\pi \xleftarrow{\$} \mathcal{P}(1^k)$
$b \xleftarrow{\$} \{0,1\}$ ; $C^* \leftarrow \bot$
$(ek, dk) \xleftarrow{\$} \mathcal{K}(\pi)$
Return $(\pi, ek)$

proc Dec$(\phi, C)$ // PKE
$dk' \leftarrow \mathcal{Q}(\pi, dk, \phi)$
If $dk' = \bot$ then return $\bot$
If $((dk' = dk) \wedge (C = C^*))$
   then return $\bot$
Return $M \leftarrow \mathcal{D}(\pi, dk', C)$

proc LR$(m_0, m_1)$ // PKE
If $(|m_0| \neq |m_1|)$ then return $\bot$
Return $C^* \xleftarrow{\$} \mathcal{E}(\pi, ek, m_b)$

proc Finalize$(b')$ // PKE
Return $(b = b')$
```

```
proc Initialize // wPRF
$\pi \xleftarrow{\$} \mathcal{P}(1^k)$
$K \xleftarrow{\$} \mathcal{K}(\pi)$
$b \xleftarrow{\$} \{0,1\}$
Return $\pi$

proc Fn$(\phi)$ // wPRF
$K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
If $K' = \bot$ then return $\bot$
$x \xleftarrow{\$} \mathsf{Dom}_{w\mathcal{PRF}}(\pi)$
If $b = 1$ then $y \leftarrow \mathcal{F}(\pi, K', x)$
Else $y \xleftarrow{\$} \mathsf{Rng}_{w\mathcal{PRF}}(\pi)$
Return $(x, y)$

proc Finalize$(b')$ // wPRF
Return $(b = b')$
```

```
proc Initialize // SECPA, SECCA
$\pi \xleftarrow{\$} \mathcal{P}(1^k)$ ; $b \xleftarrow{\$} \{0,1\}$ ; $S \leftarrow \emptyset$
$K \xleftarrow{\$} \mathcal{K}(\pi)$
Return $\pi$

proc Dec$(\phi, C)$ // SECCA
$K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
If $K' = \bot$ then return $\bot$
If $((K', C) \in S)$ then return $\bot$
Return $M \leftarrow \mathcal{D}(\pi, K', C)$

proc LR$(\phi, m_0, m_1)$ // SECPA, SECCA
If $(|m_0| \neq |m_1|)$ then return $\bot$
$K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
If $K' = \bot$ then return $\bot$
$C^* \xleftarrow{\$} \mathcal{E}(\pi, K', m_b)$ ; $S \leftarrow S \cup \{(K', C^*)\}$
Return $C^*$

proc Finalize$(b')$ // SECPA, SECCA
Return $(b = b')$
```

Figure 5: Games defining Φ-RKA security for primitives PKE-CCA, wPRF, SE-CPA, SE-CCA.

$q^2/2^k$ which is negligible. We omit the details.

IBE. Our specification of an IBE scheme $\mathcal{IBE} = (\mathcal{P}, \mathcal{M}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ adds a parameter generation algorithm $\mathcal{P}$ that given 1^k returns parameters π on which the masterkey generation algorithm $\mathcal{M}$ runs to produce the master public key *mpk* and master secret key *msk*. The rest is as usual except that algorithms get π as an additional input. Game IBE of Figure 4 is associated to $\mathcal{IBE}$ and a RKA specification $\Phi = (\pi, \mathcal{Q})$ that is compatible with $\mathcal{IBE}$. An adversary is allowed only one query to LR. Let $\mathbf{Adv}^{\text{ibe-rka}}_{\mathcal{IBE},A,\Phi}(k)$ equal $2\Pr[\text{IBE}^A \Rightarrow \mathsf{true}] - 1$ when the game has input 1^k. We say $\mathcal{IBE}$ is Φ-RKA secure if this advantage function is negligible. Here the feature of the definition worth remarking on is that the key-derivation oracle KD refuses to act only when the identity it is given matches the challenge one *and* the derived key equals the real one, leading to a strong security requirement.

From Φ-RKA IBE to Φ-RKA signatures. We show that $\mathbf{RKA}[\mathsf{IBE}] \subseteq \mathbf{RKA}[\mathsf{Sig}]$ by proving that the standard Naor transform preserves RKA security. Given a Φ-RKA IBE $\mathcal{IBE} = (\mathcal{P}, \mathcal{M}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ we obtain signature scheme $\mathcal{DS} = (\mathcal{P}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ as follows: the signature parameters are the IBE parameters; the verifying and signing keys are the IBE master public and secret keys; messages being signed are identities; and signatures are decryption keys for these identities. In the most general case verification is performed by encrypting a random message (this is an IBE message) under the identity (the message whose signature is being verified) and seeing whether the ciphertext decrypts correctly under the decryption key (signature being verified). However with all the specific IBE schemes we know (eg. [15, 10, 31]), there is a direct, deterministic verification algorithm based on the algebra. The signature scheme $\mathcal{DS}$ remains compatible with Φ since its parameters are exactly those of $\mathcal{IBE}$. The following says $\mathcal{DS}$ inherits the Φ-RKA security of $\mathcal{IBE}$.

Theorem 6.3 *Let signature scheme $\mathcal{DS} = (\mathcal{P}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ be constructed as above from Φ-RKA IBE scheme $\mathcal{IBE} = (\mathcal{P}, \mathcal{M}, \mathcal{K}, \mathcal{E}, \mathcal{D})$. Then $\mathcal{DS}$ is Φ-RKA secure.*

When the adversary against the signature scheme makes Sign queries we can simulate them via KD queries. The delicate point is that the correctness of this simulation relies on the fact that the challenge identity has not yet been defined, which means that the two procedures fail in exactly the same cases. Once the signing adversary outputs its forgery with message *id* and signature *dk*, we pick two random k-bit messages m_0, m_1 and query LR(id, m_0, m_1) to get a challenge ciphertext C^*. We then decrypt C^*

using $\mathcal{D}$ with decryption key dk and return 1 if we get back m_1 and 0 otherwise. Details are deferred to the full paper.

This motivates finding Φ-RKA secure IBE schemes, which we leave as an interesting open problem. Another interesting open question is whether or not the converse of the above is true, namely, is $\mathbf{RKA}[\mathsf{Sig}] \subseteq \mathbf{RKA}[\mathsf{IBE}]$?

PKE. A public-key encryption scheme $\mathcal{PKE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is specified as usual by parameter generation, key generation, encryption and decryption algorithms. Game PKE of Figure 5 is associated to $\mathcal{PKE}$ and a RKA specification $\Phi = (\pi, \mathcal{Q})$ that is compatible with $\mathcal{PKE}$. An adversary is allowed only one query to LR. Let $\mathbf{Adv}^{\text{pke-cc-rka}}_{\mathcal{PKE},A,\Phi}(k)$ equal $2\Pr[\text{PKE}^A \Rightarrow \mathsf{true}] - 1$ when the game has input 1^k. We say $\mathcal{PKE}$ is Φ-RKA secure if this advantage function is negligible. The attack is on the secret key, which is the decryption key, so that we are considering a chosen-ciphertext related-key attack (CC-RKA). The decryption oracle DEC refuses to act only when the ciphertext it is given matches the challenge one *and* the derived key equals the real one.

FROM Φ-RKA IBE TO Φ-RKA PKE-CCA. We show the containment $\mathbf{RKA}[\mathsf{IBE}] \subseteq \mathbf{RKA}[\mathsf{PKE\text{-}CCA}]$ by showing that the construction of Boneh, Canneti, Halevi, and Katz [13] preserves RKA security. We start with a Φ-RKA IBE scheme $\mathcal{IBE} = (\mathcal{P}, \mathcal{M}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ and a regular-secure strongly-unforgeable signature scheme $\overline{\mathcal{DS}} = (\overline{\mathcal{P}}, \overline{\mathcal{K}}, \overline{\mathcal{S}}, \overline{\mathcal{V}})$, and construct a PKE scheme $\mathcal{PKE}$ as follows:

1. Parameters: Parameters for $\mathcal{PKE}$ are the concatenation $\pi \,\|\, \overline{\pi}$ of parameters for $\mathcal{IBE}$ and $\overline{\mathcal{DS}}$.
2. Keys: Pick $(mpk, msk) \leftarrow \mathcal{M}(\pi)$, and use mpk as the public key and msk as the secret key.
3. Encryption: To encrypt a message m under mpk, generate a signing-verification key pair $(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(\overline{\pi})$. Then encrypt m for the identity $id = \overline{vk}$ by computing $c \xleftarrow{\$} \mathcal{E}(\pi, \overline{vk}, m)$ and sign c under $\overline{\mathcal{S}}$ using $\overline{sk}$ to get a signature $\overline{\sigma}$. The ciphertext is $(c, \overline{vk}, \overline{\sigma})$.
4. Decryption: To decrypt a ciphertext $(c, \overline{vk}, \overline{\sigma})$, first verify that $\overline{\sigma}$ is a valid signature on c under $\overline{vk}$, and output $\perp$ if verification fails. Then compute the user secret key for the identity $id = \overline{vk}$, and decrypt c using that key.

By construction, we have that Φ is compatible with $\mathcal{PKE}$ whenever it is compatible with $\mathcal{IBE}$.

Theorem 6.4 *Let PKE scheme $\mathcal{PKE} = (\mathcal{P}', \mathcal{K}', \mathcal{E}', \mathcal{D}')$ be constructed as above from Φ-RKA IBE scheme $\mathcal{IBE} = (\mathcal{P}, \mathcal{M}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ and normal-secure strongly-unforgeable signature scheme $\overline{\mathcal{DS}} = (\overline{\mathcal{P}}, \overline{\mathcal{K}}, \overline{\mathcal{S}}, \overline{\mathcal{V}})$. Then $\mathcal{PKE}$ is Φ-RKA secure.*

A proof is given in Appendix F. It is an adaptation of the original proof for the non-RKA version of this theorem. The primary difference is in showing that the RKA games for IBE and PKE-CCA will cooperate, because they each have rules for disallowing certain queries (with IBE one cannot extract a key for id^*, and in PKE-CCA one cannot decrypt C^*, but both rules only hold under the original secret key). Handily, the Boneh et al. construction has a structure that allows these rules to fit together.

SE. Symmetric encryption is interesting because both sender and receiver have the secret key and the RKA can now be mounted on the encryption, not merely on the decryption. A symmetric encryption scheme $\mathcal{SE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is specified as usual by parameter generation, key generation, encryption and decryption algorithms. Games SECPA, SECCA of Figure 5 are associated to $\mathcal{SE}$ and a RKA specification $\Phi = (\pi, \mathcal{Q})$ that is compatible with $\mathcal{SE}$. An adversary is now allowed multiple queries to LR. Let $\mathbf{Adv}^{\text{se-cp-rka}}_{\mathcal{SE},A,\Phi}(k)$ equal $2\Pr[\text{SECPA}^A \Rightarrow \mathsf{true}] - 1$ when the game has input 1^k. We say $\mathcal{SE}$ is Φ-RKA-CPA secure if this advantage function is negligible. Let $\mathbf{Adv}^{\text{se-cc-rka}}_{\mathcal{SE},A,\Phi}(k)$ equal $2\Pr[\text{SECCA}^A \Rightarrow \mathsf{true}] - 1$ when the game has input 1^k. We say $\mathcal{SE}$ is Φ-RKA-CCA secure if this advantage function is negligible.

REMAINING RELATIONS. Containments of the form $\mathbf{RKA}^*[\mathsf{PRF}] \subseteq \mathbf{RKA}^*[\mathsf{P}]$ can be established for all the other primitives P using the same idea as illustrated above for $\mathsf{P} = \mathsf{Sig}$. All our non-containments emanate from two basic techniques. One is based on constants, as we illustrated with $\mathbf{RKA}[\mathsf{Sig}] \not\subseteq \mathbf{RKA}[\mathsf{PRF}]$. Another example use of this technique is to show $\mathbf{RKA}[\mathsf{PKE\text{-}CCA}] \not\subseteq \mathbf{RKA}[\mathsf{SE\text{-}CCA}]$.

This is true because constant functions are fine for the first but, due to the RKA on the encryption oracle, not for the second. However sometimes we want to show $\mathbf{RKA}[\mathsf{P}_1] \not\subseteq \mathbf{RKA}[\mathsf{P}_2]$ when constant functions are either fine for both or not fine for either. We do this based on variants of the technique used above to establish $\mathbf{RKA}[\mathsf{wPRF}] \not\subseteq \mathbf{RKA}[\mathsf{PRF}]$ where the idea can be viewed as using RKDFs that flip-flop between two keys depending on a bit of the original key in such a way that there is a Φ-RKA on P_2 yet the two keys are functionally equivalent under a Φ-RKA on P_1. This can be used for example to show that $\mathbf{RKA}[\mathsf{SE\text{-}CPA}] \not\subseteq \mathbf{RKA}[\mathsf{SE\text{-}CCA}]$. In this case the attack exploits the fact that the decryption oracle rejects if a query (ϕ, C) results in (K', C) being in S, meaning that C was a challenge ciphertext created under K', and this can be made to happen depending on a certain bit of K so that the rejection leaks information about K that eventually allows the attacker to recover K. We omit details on these and the remaining relations shown in Figure 1.

References

[1] M. Albrecht, P. Farshim, K. Paterson, and G. Watson. On cipher-dependent related-key attacks in the ideal-cipher model. Cryptology ePrint Archive, Report 2011/213, 2011. `http://eprint.iacr.org/`. (Cited on page 4.)

[2] B. Applebaum, D. Harnik, and Y. Ishai. Semantic security under related-key attacks and applications. In A. C.-C. Yao, editor, *ICS 2011*. Tsinghua University Press, 2011. (Cited on page 3, 4, 5.)

[3] M. Bellare and D. Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 666–684. Springer, Aug. 2010. (Cited on page 3, 4, 5, 6, 7, 8, 9.)

[4] M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, May 2003. (Cited on page 2, 3, 4, 5, 7, 13.)

[5] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, May / June 2006. (Cited on page 7, 19, 20.)

[6] E. Biham. New types of cryptoanalytic attacks using related keys (extended abstract). In T. Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 398–409. Springer, May 1993. (Cited on page 2.)

[7] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 513–525. Springer, Aug. 1997. (Cited on page 2, 3.)

[8] N. Bitansky and R. Canetti. On strong simulation and composable point obfuscation. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 520–537. Springer, Aug. 2010. (Cited on page 5.)

[9] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984. (Cited on page 10.)

[10] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, May 2004. (Cited on page 14.)

[11] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, May 2004. (Cited on page 5.)

[12] D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, Apr. 2008. (Cited on page 5.)

[13] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):915–942, 2006. (Cited on page 3, 15.)

[14] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 37–51. Springer, May 1997. (Cited on page 2, 3.)

[15] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Dec. 2001. (Cited on page 14.)

[16] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In S. Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Jan. 2005. (Cited on page 5.)

[17] S. Dziembowski, K. Pietrzak, and D. Wichs. Non-malleable codes. In A. C.-C. Yao, editor, *ICS 2010*. Tsinghua University Press, 2010. (Cited on page 5.)

[18] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 258–277. Springer, Feb. 2004. (Cited on page 2, 3, 5.)

[19] D. Goldenberg and M. Liskov. On related-secret pseudorandomness. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 255–272. Springer, Feb. 2010. (Cited on page 3, 4, 5, 7.)

[20] O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004. (Cited on page 12, 26.)

[21] V. Goyal, A. O'Neill, and V. Rao. Correlated-input secure hash functions. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 182–200. Springer, Mar. 2011. (Cited on page 3, 4, 5, 6, 17.)

[22] V. Goyal, A. O'Neill, and V. Rao. Correlated-input secure hash functions. Cryptology ePrint Archive, Report 2011/233, 2011. Full version of [21], `http://eprint.iacr.org/`. (Cited on page 4, 5, 6.)

[23] S. Halevi and H. Krawczyk. Security under key-dependent inputs. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM CCS 07*, pages 466–475. ACM Press, Oct. 2007. (Cited on page 4.)

[24] Y. Ishai, M. Prabhakaran, A. Sahai, and D. Wagner. Private circuits II: Keeping secrets in tamperable circuits. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 308–327. Springer, May / June 2006. (Cited on page 5.)

[25] L. R. Knudsen. Cryptanalysis of LOKI91. In J. Seberry and Y. Zheng, editors, *AUSCRYPT'92*, volume 718 of *LNCS*, pages 196–208. Springer, Dec. 1992. (Cited on page 2.)

[26] L. Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, Oct. 1979. (Cited on page 25.)

[27] S. Lucks. Ciphers secure against related-key attacks. In B. K. Roy and W. Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 359–370. Springer, Feb. 2004. (Cited on page 3, 4, 5, 7, 8.)

[28] R. C. Merkle. A digital signature based on a conventional encryption function. In C. Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Aug. 1988. (Cited on page 25, 26.)

[29] M. Naor and O. Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *J. Comput. Syst. Sci.*, 58(2):336–375, 1999. (Cited on page 3, 4.)

[30] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990. (Cited on page 25.)

[31] B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, May 2005. (Cited on page 14.)

A Proof of Proposition 4.1

Given A attacking the Φ-IDFP security of $\mathcal{FF}$ we construct B such that for all k we have

$$\mathbf{Adv}^{\mathrm{idfp}}_{\mathcal{FF},A,\Phi}(k) \;\leq\; \mathbf{Adv}^{\mathrm{prf\text{-}rka}}_{\mathcal{FF},B,\Phi}(k) + \frac{q(k)}{|\mathsf{Rng}(\pi)|} \;.$$

On input π, adversary B lets $\mathbf{w} \xleftarrow{\$} \mathsf{IKfp}(\pi)$ and $\mathbf{y}[1] \leftarrow \mathrm{F}\textsc{n}(\mathsf{id}, \mathbf{w}[1])$. It initializes set T to empty and then runs A on inputs $\pi, \mathbf{w}$. When A makes oracle query ϕ, adversary B replies via the following procedure:

```
proc INITIALIZE // G_0
$\overline{\pi} \xleftarrow{\$} \overline{\mathcal{P}}(1^k)$
$\overline{K} \xleftarrow{\$} \overline{\mathcal{K}}(\overline{\pi})$
$S_0 \leftarrow \overline{\mathcal{G}}(\pi, \overline{K})$
Return $\overline{\pi}$

proc GEN($R$) // G_0
Return $S_0$

proc FINALIZE($b'$) // G_0
Return $b'$
```

```
proc INITIALIZE // $\boxed{G_1}$, G_2
$\overline{\pi} \xleftarrow{\$} \overline{\mathcal{P}}(1^k)$
$\overline{K} \xleftarrow{\$} \overline{\mathcal{K}}(\overline{\pi})$
$S_0, S_1 \xleftarrow{\$} \{0,1\}^{r(k)}$
Return $\overline{\pi}$

proc GEN($R$) // G_1, G_2
$S \leftarrow S_0$
If $\overline{\mathcal{G}}(\overline{K}) = R$ then bad $\leftarrow$ true ; $\boxed{S \leftarrow S_1}$
Return $S$

proc FINALIZE($b'$) // G_1, G_2
Return $b'$
```

Figure 6: Games for proof of Proposition 5.1. Game G_1 includes the boxed code and game G_2 does not.

```
proc FNSIM($\phi$)
If ($\phi$ = id) then return $\mathbf{y}$
$\mathbf{z}[1] \leftarrow$ FN($\phi, \mathbf{w}[1]$) ; $T \leftarrow T \cup \{\mathbf{z}\}$
Return $\mathbf{z}$
```

When A halts, adversary B returns 1 if $\mathbf{y} \in T$ and 0 otherwise. For the analysis, assume (wlog) that A never repeats an oracle query. Then

$$\begin{aligned} \Pr[\text{PRFReal}^B \Rightarrow 1] &= \mathbf{Adv}^{\text{idfp}}_{\mathcal{FF},A,\Phi}(k) \\ \Pr[\text{PRFRand}^B \Rightarrow 1] &\leq \frac{q(k)}{|\mathsf{Rng}(\pi)|} \end{aligned}$$

where $q(\cdot)$ is the number of oracle queries made by A. The last equation is justified by the claw-freeness of Φ which ensures that $\Phi_{\pi,\phi}(K) = K$ only if $\phi = \mathsf{id}$. Thus in the random case all queries of A yield keys different from K.

B Proof of Proposition 5.1

Let $\ell(k)$ be the length of the seed returned by $\overline{\mathcal{K}}(\overline{\pi})$ when the security parameter underlying $\overline{\pi}$ is k. On input $\overline{\pi}$ let the key-generation algorithm $\mathcal{K}$ of $\mathcal{PRG} = (\overline{\mathcal{P}}, \mathcal{K}, \mathcal{G}, r)$ pick a random bit c and return $c \,\|\, \overline{\mathcal{K}}(\overline{\pi})$. Algorithm $\mathcal{G}(\overline{\pi}, K)$ lets $\overline{K}$ be the last $\ell(k)$ bits of K and returns $\overline{\mathcal{G}}(\overline{\pi}, K)$. Since $\mathcal{G}$ ignores the first bit of its input seed, any pair of seeds that differ only in their first bits result in the same output, meaning are non-trivial collisions.

Now we define Φ so that $\mathcal{PRG}$ is not Φ-ICR secure yet is Φ-RKA secure. For any $R \in \{0,1\}^{r(k)}$ we let $\Phi_{\overline{\pi},R}$ be the function that on input $K \in \{0,1\}^{\ell(k)+1}$ returns K if $\mathcal{G}(\overline{\pi}, K) \neq R$ and otherwise returns the string formed by flipping the first bit of K.

The adversary A that makes queries $X \leftarrow$ GEN(id) and $S \leftarrow$ GEN(ϕ_X) and returns 1 iff $S = T$ has icr-advantage 1, meaning $\mathcal{PRG}$ is not Φ-ICR secure. Now we show that $\mathcal{PRG}$ is Φ-RKA secure under the assumption that $\overline{\mathcal{PRG}}$ was normal (i.e. ID-RKA) secure.

The functions $\Phi_{\overline{\pi},R}$, on input $c \,\|\, \overline{K}$, where $c \in \{0,1\}$, have only two possible outputs, namely $c \,\|\, \overline{K}$ itself, or this string with its first bit c flipped. But by design $\mathcal{G}(\overline{\pi}, \cdot)$ has the same output on these two inputs, so in the real case, the response to a query ϕ_R is always $\overline{\mathcal{G}}(\overline{\pi}, \overline{K})$, regardless of R. This would seem to make it easy for an adversary B against the normal-security of $\overline{\mathcal{PRG}}$ to simulate A. The difficulty is the random case. Here the answer to the same query is one of two possible random strings that we call

S_0, S_1, but B does not know which. We resolve this by considering two cases in the random case. If A does not query $\mathcal{G}(\overline{\pi}, K)$ then the simulation is again easy since the response to all queries is the same. If A succeeds in querying $\mathcal{G}(\overline{\pi}, K)$, a different strategy is used. Let us now provide the actual proof.

Given A attacking the Φ-RKA security of $\mathcal{PRG}$ we construct B_1, B_2 such that for all k we have

$$\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG},A,\Phi}(k) \;\leq\; \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG},B_1,\mathsf{ID}}(k) + \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG},B_2,\mathsf{ID}}(k) + \frac{q(k)}{2^{r(k)}} \;.$$

where $q(\cdot)$ is the number of GEN queries made by A. The proof considers games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2$ of Figure 6. We begin by claiming that

$$\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG},A,\Phi}(k) \;=\; \Pr[\mathrm{G}_0^A \Rightarrow 1] - \Pr[\mathrm{G}_1^A \Rightarrow 1] \;.$$

This is justified by the discussion above. Now games $\mathbf{G}_1, \mathbf{G}_2$ are identical-until-bad so by the Fundamental Lemma of Game Playing [5] we have

$$\begin{aligned}\Pr[\mathrm{G}_0^A \Rightarrow 1] - \Pr[\mathrm{G}_1^A \Rightarrow 1] &= \Pr[\mathrm{G}_0^A \Rightarrow 1] - \Pr[\mathrm{G}_2^A \Rightarrow 1] + \Pr[\mathrm{G}_2^A \Rightarrow 1] - \Pr[\mathrm{G}_1^A \Rightarrow 1] \\ &\leq \Pr[\mathrm{G}_0^A \Rightarrow 1] - \Pr[\mathrm{G}_2^A \Rightarrow 1] + \Pr[\mathrm{BAD}(\mathrm{G}_2^A)] \;.\end{aligned}$$

We construct B_1, B_2 such that for all k we have

$$\begin{aligned}\Pr[\mathrm{G}_0^A \Rightarrow 1] - \Pr[\mathrm{G}_2^A \Rightarrow 1] &\leq \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG},B_1,\mathsf{ID}}(k) \\ \Pr[\mathrm{BAD}(\mathrm{G}_2^A)] - \frac{q(k)}{2^{r(k)}} &\leq \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG},B_2,\mathsf{ID}}(k) \;.\end{aligned}$$

On input $\overline{\pi}$, adversary B_1 lets $S_0 \stackrel{\$}{\leftarrow} \mathrm{GEN}(\mathsf{id})$ and runs A on input $\overline{\pi}$. When A makes a query R, adversary B_1 responds with S_0, regardless of the value of R. When A halts, B_1 halts with the same output as A.

On input $\overline{\pi}$, adversary B_2 lets $X \stackrel{\$}{\leftarrow} \mathrm{GEN}(\mathsf{id})$ and $S_0 \stackrel{\$}{\leftarrow} \{0,1\}^{r(k)}$. It initializes a set $T \leftarrow \emptyset$ and runs A on input $\overline{\pi}$. When A makes a query R, adversary B_1 responds with S_0 regardless of the value of R, and puts R in T. When A halts, B_2 returns 1 if $X \in T$ and 0 otherwise.

C Proof of Theorem 6.1

Let ID be the RKA specification consisting of just the identity function, so that an ID-RKA secure signature scheme is just a normal-secure signature scheme, the condition we assume is met by $\overline{\mathcal{DS}}$. Given an adversary A mounting a $(\Phi \cup \Phi_{\mathrm{c}})$-RKA on $\mathcal{DS}$ we construct adversaries P_0, P_1, S_0, S_1 such that for every $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A,\Phi\cup\Phi_{\mathrm{c}}}(k) \;\leq\; \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG},P_0,\Phi}(k) + \mathbf{Adv}^{\text{sig-rka}}_{\overline{\mathcal{DS}},S_0,\mathsf{ID}}(k) + \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG},P_1,\Phi}(k) + \mathbf{Adv}^{\text{sig-rka}}_{\overline{\mathcal{DS}},S_1,\mathsf{ID}}(k) \;. \tag{1}$$

This proves the theorem.

In the real game, where the secret signing key is a seed K for $\mathcal{PRG}$, the response to a query $\mathrm{SIGN}(\phi, m)$ is computed by letting $K' \leftarrow \mathcal{Q}(\pi, K, \phi)$ and $T[K'] \leftarrow \mathcal{G}(\pi, K')$ and then using the latter as coins for the key generation algorithm $\overline{\mathcal{K}}$ to get the secret key $\overline{sk}'$ under which the returned signature is generated. A natural approach to proving security is to consider a game R which replaces $T[K']$ with a random $r(k)$ bit string associated to K'. We expect to be able to prove that the assumed Φ-RKA security of $\mathcal{PRG}$ implies that A's success probability in R is about the same as in the real game and then prove that A will not succeed in R based on the assumed security of $\overline{\mathcal{DS}}$. However, we run into a subtle difficulty in the first step. In order to be able to test whether A succeeded, the adversary P attacking the Φ-RKA security of $\mathcal{PRG}$ needs to know the set M, and to do this it must be able to test when a derived seed K'

resulting from a signing query ϕ, m equals the real seed K so that it can put the corresponding message m in M. However, it does not know either seed so it is unclear how it can perform this test. To get around this problem we resort to assuming claw-freeness of Φ. This ensures that $K' = K$ only when $\phi = \mathsf{id}$ since no function different from id is allowed to map K to the same thing that id maps it to, namely K. Now the test is independent of K', K.

Let ϕ_a denote the (description of the) constant function that returns a. Simulation of signatures under constant functions will be done directly. For this purpose, both the games and the constructed adversaries will test whether or not ϕ in a signing query is ϕ_a for some a and branch accordingly. A delicate point is that for this to be correct it must be unambiguous, meaning no constant function is in Φ. The latter turns out to be a consequence of the claw-freeness and the fact that the identity function is in Φ. However, constant functions create another difficulty, namely that the adversary might query ϕ_a with $a = K$ being the secret signing key. The difficulty is not for a simulation to return the signature, but to know to put the message m from the query into M, just as above. This turns out to be more of a bother than we would have liked. One's first thought would be that a query of K hands us the seed and should lead at once to breaking the Φ-RKA security of $\mathcal{PRG}$. Letting $R = \textsc{Gen}(\mathsf{id})$, an adversary P, on seeing query ϕ_a, could test whether or not $\mathcal{G}(\pi, a) = R$, declaring itself to be in the real game if so and the random game otherwise. But for this to be correct we need to show that the probability of the test succeeding when R is random is small. We might hope to argue this unconditionally. But the only way one would appear to be able to do so is to show that for most R there does not *exist* a seed a such that $\mathcal{G}(\pi, a) = R$. But this is only true if the generator is expanding, meaning the number of possible seeds is much less than $2^{r(k)}$. But we are using PRGs to get RKA security and there is no reason that they should be, or should be required to be, expanding, so this argument does not work. What we do instead is prove the claim (that the success probability of the test is small when R is random) computationally, based on the assumed security of the base signature scheme. This is why we end up with two adversaries of each kind. We note that given any Φ-RKA PRG one can easily make one that is expanding by applying a normal PRG to the output of the given one, but it seemed to us a poor choice to change and add burden to the scheme for the sake of the proof when we could in fact prove it without this.

Proceeding now to the actual proof, consider games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3$ of Figure 7. We claim that

$$\begin{aligned}
\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A,\Phi\cup\Phi_c}(k) &= \Pr[\mathrm{G}_0^A \Rightarrow \mathsf{true}] && (2)\\
&= \Pr[\mathrm{G}_1^A \Rightarrow \mathsf{true}] + \Pr[\mathrm{G}_0^A \Rightarrow \mathsf{true}] - \Pr[\mathrm{G}_1^A \Rightarrow \mathsf{true}] \\
&\le \Pr[\mathrm{G}_1^A \Rightarrow \mathsf{true}] + \Pr[\textsc{Bad}(\mathrm{G}_1^A)] && (3)\\
&\le \Pr[\mathrm{G}_1^A \Rightarrow \mathsf{true}] + \Pr[\textsc{Bad}(\mathrm{G}_2^A)]\,. && (4)
\end{aligned}$$

In game G_0 we focus on what is difficult to simulate, namely testing when signing queries result in use of the real key and thus of addition of the message m to the set M. When this results from queries of constant functions we set a flag but, since the boxed code is included, take the appropriate action anyway. When the query involves a function $\phi \in \Phi$, rather than test whether $\phi(K) = K$, we test whether ϕ is the identity, which is equivalent because Φ is claw-free. This justifies Equation (2). Games $\mathrm{G}_0, \mathrm{G}_1$ are identical-until-bad so Equation (3) follows from the Fundamental Lemma of Game Playing [5]. When bad is set in G_1 it is also set in game G_2 (although not necessarily vice-versa). Thus $\Pr[\textsc{Bad}(\mathrm{G}_1^A)] \le \Pr[\textsc{Bad}(\mathrm{G}_2^A)]$ which justifies Equation (4). In game G_2 the test no longer depends on K and hence can be simulated. This justifies the above. Now we will design P_0, P_1, S_0, S_1 so that for

```
proc Initialize // $G_0, G_1, G_2$
$\pi \xleftarrow{\$} \mathcal{P}(1^k)$ ; $\overline{\pi} \xleftarrow{\$} \overline{\mathcal{P}}(1^k)$ ; $M \leftarrow \emptyset$
$K \xleftarrow{\$} \mathcal{K}(\pi)$ ; $T[K] \leftarrow \mathcal{G}(\pi, K)$
$(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(\overline{\pi}; T[K])$
Return $(\pi \,\|\, \overline{\pi}, \overline{vk})$

proc Sign$(\phi, m)$ // $\boxed{G_0}$, $G_1$
If $(\phi = \phi_a)$ for some $a$ then
   $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; \mathcal{G}(\pi, a))$
   $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
   If $(a = K)$ then
      $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{M \leftarrow M \cup \{m\}}$
Else
   $K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
   If $K' = \perp$ then return $\perp$
   If $T[K'] = \perp$ then
      $T[K'] \leftarrow \mathcal{G}(\pi, K')$
   $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; T[K'])$
   $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
   If $(\phi = \mathsf{id})$ then $M \leftarrow M \cup \{m\}$
Return $\sigma$

proc Finalize$(m, \sigma)$ // $G_0, G_1, G_2, G_3$
Return $((\overline{\mathcal{V}}(\overline{\pi}, \overline{vk}, m, \sigma) = 1) \wedge (m \notin M))$
```

```
proc Sign$(\phi, m)$ // $G_2$
If $(\phi = \phi_a)$ for some $a$ then
   $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; \mathcal{G}(\pi, a))$
   $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
   If $(\overline{vk}' = \overline{vk})$ then
      $\mathsf{bad} \leftarrow \mathsf{true}$
Else
   $K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
   If $K' = \perp$ then return $\perp$
   If $T[K'] = \perp$ then
      $T[K'] \leftarrow \mathcal{G}(\pi, K')$
   $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; T[K'])$
   $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
   If $(\phi = \mathsf{id})$ then
      $M \leftarrow M \cup \{m\}$
Return $\sigma$
```

```
proc Initialize // $G_3$
$\pi \xleftarrow{\$} \mathcal{P}(1^k)$ ; $\overline{\pi} \xleftarrow{\$} \overline{\mathcal{P}}(1^k)$ ; $M \leftarrow \emptyset$
$K \xleftarrow{\$} \mathcal{K}(\pi)$ ; $T[K] \xleftarrow{\$} \{0,1\}^{r(k)}$
$(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(\overline{\pi}; T[K])$
Return $(\pi \,\|\, \overline{\pi}, \overline{vk})$

proc Sign$(\phi, m)$ // $G_3$
If $(\phi = \phi_a)$ for some $a$ then
   $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; \mathcal{G}(\pi, a))$
   $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
   If $(\overline{vk}' = \overline{vk})$ then
      $\mathsf{bad} \leftarrow \mathsf{true}$
Else
   $K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
   If $K' = \perp$ then return $\perp$
   If $T[K'] = \perp$ then
      $T[K'] \xleftarrow{\$} \{0,1\}^{r(k)}$
   $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; T[K'])$
   $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
   If $(\phi = \mathsf{id})$ then
      $M \leftarrow M \cup \{m\}$
Return $\sigma$
```

Figure 7: Games for proof of Theorem 6.1. Game G_0 includes the boxed code and game G_1 does not.

every $k \in \mathbb{N}$

$$\Pr[G_1^A \Rightarrow \mathsf{true}] - \Pr[G_3^A \Rightarrow \mathsf{true}] \le \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG}, P_0, \Phi}(k) \tag{5}$$

$$\Pr[G_3^A \Rightarrow \mathsf{true}] \le \mathbf{Adv}^{\mathrm{sig\text{-}rka}}_{\mathcal{DS}, S_0, \mathsf{ID}}(k) \tag{6}$$

$$\Pr[\mathrm{Bad}(G_2^A)] - \Pr[\mathrm{Bad}(G_3^A)] \le \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG}, P_1, \Phi}(k) \tag{7}$$

$$\Pr[\mathrm{Bad}(G_3^A)] \le \mathbf{Adv}^{\mathrm{sig\text{-}rka}}_{\mathcal{DS}, S_1, \mathsf{ID}}(k)\ . \tag{8}$$

Now, Equations (5) and (6) bound the first term of Equation (4) while Equations (7) and (8) bound the second term. Let us now describe the claimed adversaries.

For $b \in \{0,1\}$, adversary P_b gets input π. It begins with the initializations

$$\overline{\pi} \xleftarrow{\$} \overline{\mathcal{P}}(1^k)\ ;\ M \leftarrow \emptyset\ ;\ R \xleftarrow{\$} \mathrm{Gen}(\mathsf{id})\ ;\ (\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(\overline{\pi}; R)$$

It now runs A on inputs $\pi \,\|\, \overline{\pi}, \overline{vk}$, responding to its oracle queries via the following procedure:

```
proc SignSim$(\phi, m)$
If $(\phi = \phi_a)$ for some $a$ then
   $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; \mathcal{G}(\pi, a))$
   $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
   If $(vk' = \overline{vk})$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Else
   $R' \xleftarrow{\$} \mathrm{Gen}(\phi)$
   If $R' = \perp$ then return $\perp$
```

```
        $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; R')$
        $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
        If $(\phi = \mathsf{id})$ then $M \leftarrow M \cup \{m\}$
    Return $\sigma$
```

When A makes query FINALIZE(m, σ), adversary P_0 returns 1 if $(\overline{\mathcal{V}}(\overline{\pi}, \overline{vk}, m, \sigma) = 1) \wedge (m \notin M)$ and 0 otherwise, while adversary P_1 returns 1 if it set bad and 0 otherwise.

For $b \in \{0, 1\}$, adversary S_b gets input $\overline{\pi}, \overline{vk}$. It begins with the initializations

$$\pi \xleftarrow{\$} \mathcal{P}(1^k) \; ; \; K \xleftarrow{\$} \mathcal{K}(\pi)$$

It now runs A on inputs $\pi \,\|\, \overline{\pi}, \overline{vk}$, responding to its oracle queries via the following procedure:

```
proc SIGNSIM$(\phi, m)$
If $(\phi = \phi_a)$ for some $a$ then
    $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; \mathcal{G}(\pi, a))$
    $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
    If $(\overline{vk}' = \overline{vk})$ then bad $\leftarrow$ true ; $\overline{sk}^* \leftarrow \overline{sk}'$
If $(\phi = \mathsf{id})$ then $M \leftarrow M \cup \{m\}$ ; $\sigma \xleftarrow{\$}$ SIGN$(\mathsf{id}, m)$
    Else
        $K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
        If $K' = \perp$ then return $\perp$
        If $T[K'] = \perp$ then $T[K'] \xleftarrow{\$} \{0,1\}^{r(k)}$
        $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; T[K'])$
        $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
Return $\sigma$
```

When A makes query FINALIZE(m, σ), adversary S_0 does too. Adversary S_1, however, sees whether it set bad. If so it picks some message $m \notin M$, lets $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}^*, m)$ and makes query FINALIZE(m, σ). Otherwise, it halts without output.

D Proof of Theorem 6.2

Again let ID be the RKA specification consisting of just the identity function, so that an ID-RKA secure signature scheme is just a normal-secure signature scheme, the condition we assume is met by $\overline{\mathcal{DS}}$. Given an adversary A mounting a $(\Phi \cup \Phi_{\mathrm{c}})$-RKA on $\mathcal{DS}$ we construct adversaries P_0, P_1, S_0, S_1, C such that for every $k \in \mathbb{N}$

$$\begin{aligned}&\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A,\Phi\cup\Phi_{\mathrm{c}}}(k)\\ &\quad\le\ \mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},P_0,\Phi}(k) + \mathbf{Adv}^{\text{sig-rka}}_{\overline{\mathcal{DS}},S_0,\mathsf{ID}}(k) + \mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},P_1,\Phi}(k) + \mathbf{Adv}^{\text{sig-rka}}_{\overline{\mathcal{DS}},S_1,\mathsf{ID}}(k) + \mathbf{Adv}^{\text{icr}}_{\mathcal{PRG},C,\Phi}(k)\ . \quad (9)\end{aligned}$$

This proves the theorem.

As we discussed in Appendix C, an important difficulty is that an adversary P against the PRG cannot test whether $K' = K$. There we solved this by assuming claw-freeness of Φ, which allowed us to instead test whether $\phi = \mathsf{id}$. Now that Φ may not be claw-free, we attempt to exploit the collision-resistance of $\mathcal{PRG}$. We test instead whether $T[K'] = T[K]$, hoping this happens only when $K' = K$. To ensure that the game is faithful to the original one, we put in a correction, setting bad and removing m from M if

```
proc INITIALIZE  // $G_0, G_1, G_2, G_3$
$\pi \xleftarrow{\$} \mathcal{P}(1^k)$ ; $\overline{\pi} \xleftarrow{\$} \overline{\mathcal{P}}(1^k)$ ; $M \leftarrow \emptyset$
$K \xleftarrow{\$} \mathcal{K}(\pi)$ ; $T[K] \leftarrow \mathcal{G}(\pi, K)$
$(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(\overline{\pi}; T[K])$
Return $(\pi \,\|\, \overline{\pi}, \overline{vk})$

proc SIGN$(\phi, m)$  // $\boxed{G_0}$, $G_1$
If $(\phi = \phi_a)$ for some $a$ then
    $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; \mathcal{G}(\pi, a))$
    $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
    If $(a = K)$ then bad $\leftarrow$ true ; $\boxed{M \leftarrow M \cup \{m\}}$
Else
    $K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
    If $K' = \bot$ then return $\bot$
    If $T[K'] = \bot$ then $T[K'] \leftarrow \mathcal{G}(\pi, K')$
    $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; T[K'])$ ; $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
    If $((T[K'] = T[K]) \wedge (m \notin M))$ then
        $M \leftarrow M \cup \{m\}$
        If $(K' \neq K)$ then bad $\leftarrow$ true ; $\boxed{M \leftarrow M \setminus \{m\}}$
Return $\sigma$

proc FINALIZE$(m, \sigma)$  // $G_0, G_1, G_2, G_3$
Return $((\overline{\mathcal{V}}(\overline{\pi}, \overline{vk}, m, \sigma) = 1) \wedge (m \notin M))$
```

```
proc SIGN$(\phi, m)$  // $G_2$
If $(\phi = \phi_a)$ for some $a$ then
    $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; \mathcal{G}(\pi, a))$
    $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
    If $(\overline{vk}' = \overline{vk})$ then bad $\leftarrow$ true
Else
    $K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
    If $K' = \bot$ then return $\bot$
    If $T[K'] = \bot$ then $T[K'] \leftarrow \mathcal{G}(\pi, K')$
    $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; T[K'])$ ; $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
    If $((T[K'] = T[K]) \wedge (m \notin M))$ then
        $M \leftarrow M \cup \{m\}$
Return $\sigma$

proc SIGN$(\phi, m)$  // $G_3$
If $(\phi = \phi_a)$ for some $a$ then
    $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; \mathcal{G}(\pi, a))$
    $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
Else
    $K' \leftarrow \mathcal{Q}(\pi, K, \phi)$
    If $K' = \bot$ then return $\bot$
    If $T[K'] = \bot$ then $T[K'] \leftarrow \mathcal{G}(\pi, K')$
    $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; T[K'])$ ; $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
    If $((T[K'] = T[K]) \wedge (m \notin M))$ then
        $M \leftarrow M \cup \{m\}$
        If $(K' \neq K)$ then bad $\leftarrow$ true
Return $\sigma$
```

Figure 8: Games for proof of Theorem 6.2. Game G_0 includes the boxed code and game G_1 does not.

$K' \neq K$. We have

$$\begin{aligned}\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A,\Phi\cup\Phi_c}(k) &= \Pr[G_0^A \Rightarrow \mathsf{true}] \\ &= \Pr[G_1^A \Rightarrow \mathsf{true}] + \Pr[G_0^A \Rightarrow \mathsf{true}] - \Pr[G_1^A \Rightarrow \mathsf{true}] \\ &\leq \Pr[G_1^A \Rightarrow \mathsf{true}] + \Pr[\textsc{Bad}(G_1^A)] \ .\end{aligned}$$

The boxed code being dropped in G_1, a simulator can ignore the corresponding tests that set bad, since they do not affect the outcome of the game. Thus, as in Appendix C, we can construct P_0, S_0 so that

$$\Pr[G_1^A \Rightarrow \mathsf{true}] \quad \leq \quad \mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},P_0,\Phi}(k) + \mathbf{Adv}^{\text{sig-rka}}_{\overline{\mathcal{DS}},S_0,\mathsf{ID}}(k) \ .$$

It remains to bound $\Pr[\textsc{Bad}(G_1^A)]$. The probability that the first occurence of bad is set in G_1 is at most the probability that it is set in G_2. The probability that second occurence of bad is set in G_1 equals the probability that it is set in G_3. Thus

$$\Pr[\textsc{Bad}(G_1^A)] \quad \leq \quad \Pr[\textsc{Bad}(G_2^A)] + \Pr[\textsc{Bad}(G_3^A)] \ .$$

As in Appendix C, we can construct P_1, S_1 so that

$$\Pr[\textsc{Bad}(G_2^A)] \quad \leq \quad \mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},P_1,\Phi}(k) + \mathbf{Adv}^{\text{sig-rka}}_{\overline{\mathcal{DS}},S_1,\mathsf{ID}}(k) \ .$$

Finally we construct C so that

$$\Pr[\textsc{Bad}(G_3^A)] \quad \leq \quad \mathbf{Adv}^{\text{icr}}_{\mathcal{PRG},C,\Phi}(k) \ .$$

Adversary C gets input π. It begins with the initializations

$$\overline{\pi} \xleftarrow{\$} \overline{\mathcal{P}}(1^k) \; ; \; R \xleftarrow{\$} \text{GEN}(\mathsf{id}) \; ; \; (\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(\overline{\pi}; R)$$

It now runs A on inputs $\pi \,\|\, \overline{\pi}, \overline{vk}$, responding to its oracle queries via the following procedure:

```
proc SIGNSIM($\phi, m$)
If ($\phi = \phi_a$) for some $a$ then
    $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; \mathcal{G}(\pi, a))$ ; $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
Else
    $R' \xleftarrow{\$}$ GEN($\phi$)
    If $R' = \bot$ then return $\bot$
    $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(\overline{\pi}; R')$ ; $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}', m)$
Return $\sigma$
```

When A makes query FINALIZE(m, σ), adversary P halts without output.

E Strong Φ-RKA Security

We suggest a stronger notion of Φ-RKA security for signature schemes and other primitives. Let us start with the former. In strong Φ-RKA security for signatures, forgery is not only hard relative to the original public key, but even for ones associated with the derived keys produced in the attack. We are not aware of this having any application-relevance but wish to highlight it because our constructions possess it and it may be of some utility in the future.

In the usual syntax of digital signatures, the public and secret keys are produced, together, by the (randomized) key generation algorithm. In most "real" schemes, however, the secret key is produced first and the public key is a deterministic function of the (parameters and) secret key. We call any signature scheme with this property separable and we call the algorithm that deterministically produces the public key from the parameters and secret key the public-key generator. Note that even if a scheme is not separable to start with, it can be made so by using the coins of the key-generation algorithm as the secret key in an obvious way.

DEFINITIONS. A signature scheme $\mathcal{DS} = (\mathcal{P}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is separable if there is a deterministic algorithm $\mathcal{T}$, called the public-key generator, such that for all $\pi \in [\mathcal{P}(1^k)]$ and all $k \in \mathbb{N}$ the output of the process

$$(vk, sk) \xleftarrow{\$} \mathcal{K}(\pi) \; ; \; vk \leftarrow \mathcal{T}(\pi, sk) \; ; \; \text{ Return } (vk, sk)$$

is distributed identically to the output of $\mathcal{K}(\pi)$. Let $\mathcal{DS} = (\mathcal{P}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ be a separable signature scheme with public-key generator $\mathcal{T}$. Game SSig of Figure 9 is associated to $(\mathcal{DS}, \mathcal{T})$ and a RKA specification $\Phi = (\pi, \mathcal{Q})$ that is compatible with $\mathcal{DS}$. Let $\mathbf{Adv}^{\text{ssig-rka}}_{\mathcal{DS},\mathcal{T},A,\Phi}(k)$ be the probability that SSig^A returns true when the game has input 1^k. We say $(\mathcal{DS}, \mathcal{T})$ is strongly Φ-RKA secure if this advantage function is negligible. A novel aspect of this definition is that the SIGN oracle returns not just the signature but the public key associated to the secret key under which the signature was created. This is necessary because the adversary may not be able to compute this public key on its own. To win, it need only forge a signature under one of the public keys derived via its SIGN queries.

RESULT. We first observe that the signature scheme $\mathcal{DS} = (\mathcal{P} \,\|\, \overline{\mathcal{P}}, \mathcal{K}', \mathcal{S}, \mathcal{V})$ constructed from Φ-RKA PRG $\mathcal{PRG} = (\mathcal{P}, \mathcal{K}, \mathcal{G}, r)$ and normal-secure signature scheme $\overline{\mathcal{DS}} = (\overline{\mathcal{P}}, \overline{\mathcal{K}}, \overline{\mathcal{S}}, \overline{\mathcal{V}})$ as described in Section 6 is separable. This is true regardless of whether or not $\overline{\mathcal{DS}}$ is separable. Now we claim that $\mathcal{DS}$ is strongly Φ-RKA secure.

Theorem E.1 *Let separable signature scheme $\mathcal{DS} = (\mathcal{P} \,\|\, \overline{\mathcal{P}}, \mathcal{K}', \mathcal{S}, \mathcal{V})$ be constructed as in Section 6 from Φ-RKA PRG $\mathcal{PRG} = (\mathcal{P}, \mathcal{K}, \mathcal{G}, r)$ and normal-secure signature scheme $\overline{\mathcal{DS}} = (\overline{\mathcal{P}}, \overline{\mathcal{K}}, \overline{\mathcal{S}}, \overline{\mathcal{V}})$ Let $\mathcal{T}$*

```
proc INITIALIZE // SSig
$\pi \xleftarrow{\$} \mathcal{P}(1^k)$ ; $M \leftarrow \emptyset$ ; $(vk, sk) \xleftarrow{\$} \mathcal{K}(\pi)$ ; $i \leftarrow 0$
Return $(\pi, vk)$

proc SIGN($\phi, m$) // SSig
$sk' \leftarrow \mathcal{Q}(\pi, sk, \phi)$
If $sk' = \perp$ then return $\perp$
$i \leftarrow i + 1$ ; $vk_i \leftarrow \mathcal{T}(\pi, sk')$
$M \leftarrow M \cup \{(vk_i, m)\}$
$\sigma \xleftarrow{\$} \mathcal{S}(\pi, sk', m)$
Return $(vk_i, \sigma)$

proc FINALIZE($j, m, \sigma$) // SSig
If $(j > i)$ then return false
Return $((\mathcal{V}(\pi, vk_j, m, \sigma) = 1) \wedge ((vk_j, m) \notin M))$
```

Figure 9: Game defining strong Φ-RKA security for $(\mathcal{DS}, \mathcal{T})$ where $\Phi = (\pi, \mathcal{Q})$.

denote its public-key generation algorithm and assume Φ is claw-free. Then $(\mathcal{DS}, \mathcal{T})$ is strongly Φ-RKA secure.

Notice that we do *not* claim $(\mathcal{DS}, \mathcal{T})$ is strongly $(\Phi \cup \Phi_c)$-RKA secure. Indeed, strongly Φ_c-RKA signature schemes do not exist. The strong security of any signature scheme breaks, since an adversary can itself compute signatures under a constant secret key and can then forge under them. The proof is similar to that of Theorem 6.1 and is omitted. An analogue of Theorem 6.2 may also be stated and proved.

OTHER PRIMITIVES. The analogous security definitions for strong Φ-RKA CCA asymmetric encryption also exists. For Φ-RKA CCA asymmetric encryption, the adversary selects not only two messages m_0 and m_1, but also a $\phi \in \Phi$, and then receives the encryption of one of the two messages under the related key derived using ϕ; the adversary is said to succeed if it can guess with non-negligible advantage which message was encrypted.

The generic construction that transforms a secure CCA asymmetric encryption scheme using a Φ-RKA-PRG to generate the secret key before each use gives this strong security definition. We omit the proof here, but it is very similar to the proof of strong security for the signature scheme. Intuitively, each distinct $\phi \in \Phi$ cause the Φ-RKA-PRG to output values indistinguishable from random. When this pseudorandomness is used to generate the secret key, it appears as if it was a new randomly generated instance of the primitive. Since the adversary can only interact with polynomially many of these, we can guess which one they will choose, and embed the standard Φ-RKA security challenge here.

A DEDICATED SCHEME. In addition to our general transformation from any signature scheme to a strongly Φ-RKA secure signature scheme, we present a specific and direct construction of a strongly Φ-RKA secure signature scheme based on a Φ-RKA PRF. We use the Merkle Tree [28] transformation from a secure one-time signature scheme to a secure many time signature scheme. By replacing the PRF in this construction with a Φ-RKA-PRF, we note that the scheme becomes a $(\phi \cup \Phi_c)$-RKA secure signature scheme. It is interesting that creating a Φ-RKA secure scheme here only requires replacing the PRF with a Φ-RKA-PRF. This means the scheme, unmodified, will be RKA secure if the underlying PRF is RKA secure. No modification to the scheme is needed.

To sign n-bit messages with security parameter k, we require a one-time signature scheme that can sign messages twice the length of its public key, and that supports a public-key generator π. We additionally require $\mathcal{PRF} = (\mathcal{P}, \mathcal{K}, \mathcal{F})$ with $\mathsf{Rng}(\pi)$ equal to the private key space of the one-time signature scheme for each value of k. Such a one-time signature scheme exists under the minimal assumption that one-way functions exist [30]. This scheme uses a hash and sign paradigm based on a universal one-way hash function (UOWHF), and the Lamport one-time signature scheme [26], which

```
proc INITIALIZE  // $G_0, G_1, G_2$
$\pi \xleftarrow{\$} \mathcal{P}(1^k)$ ; $\overline{\pi} \xleftarrow{\$} \overline{\mathcal{P}}(1^k)$
$(mpk, msk) \xleftarrow{\$} \mathcal{M}(\pi)$
$b \xleftarrow{\$} \{0,1\}$ ; $C^* \leftarrow \bot$
Return $((\pi, \overline{\pi}), mpk)$

proc LR$(m_0, m_1)$  // $G_0, G_1, G_2$
If $(|m_0| \neq |m_1|)$ then return $\bot$
$(\overline{vk}^*, \overline{sk}^*) \xleftarrow{\$} \overline{\mathcal{K}}(\overline{\pi})$
$c^* \xleftarrow{\$} \mathcal{E}(\pi, mpk, \overline{vk}^*, m_b)$
$\overline{\sigma}^* \leftarrow \overline{\mathcal{S}}(\overline{\pi}, \overline{sk}^*, c^*)$
$C^* \leftarrow (c^*, \overline{vk}^*, \overline{\sigma}^*)$
Return $C^*$

proc FINALIZE$(b')$  // $G_0, G_1, G_2$
Return $(b = b')$
```

```
proc DEC$(\phi, (c, \overline{vk}, \overline{\sigma}))$  // $G_0$
$msk' \leftarrow \Phi(msk, \phi, \pi)$
If $msk' = \bot$ then return $\bot$
If $((msk' = msk) \wedge ((c, \overline{vk}, \overline{\sigma}) = C^*))$
    then return $\bot$
If $\overline{\mathcal{V}}(\overline{\pi}, \overline{vk}, c, \overline{\sigma}) = 0$
    then return $\bot$
$dk \leftarrow \mathcal{K}(\pi, msk', \overline{vk})$
Return $M \leftarrow \mathcal{D}(\pi, dk, c)$
```

```
proc DEC$(\phi, (c, \overline{vk}, \overline{\sigma}))$  // $G_1$
$msk' \leftarrow \Phi(msk, \phi, \pi)$
If $msk' = \bot$ then return $\bot$
If $((msk' = msk) \wedge ((c, \overline{vk}, \overline{\sigma}) = C^*))$
    then return $\bot$
If $\overline{\mathcal{V}}(\overline{\pi}, \overline{vk}, c, \overline{\sigma}) = 0$
    then return $\bot$
If $(\overline{vk} = \overline{vk}^*) \wedge ((c, \overline{\sigma}) \neq (c^*, \overline{\sigma}^*))$
    then bad $\leftarrow$ true ;  return $\bot$
$dk \leftarrow \mathcal{K}(\pi, msk', \overline{vk})$
Return $M \leftarrow \mathcal{D}(\pi, dk, c)$

proc DEC$(\phi, (c, \overline{vk}, \overline{\sigma}))$  // $G_2$
$msk' \leftarrow \Phi(msk, \phi, \pi)$
If $msk' = \bot$ then return $\bot$
If $(msk' = msk) \wedge (\overline{vk} = \overline{vk}^*)$
    then return $\bot$
If $\overline{\mathcal{V}}(\overline{\pi}, \overline{vk}, c, \overline{\sigma}) = 0$
    then return $\bot$
$dk \leftarrow \mathcal{K}(\pi, msk', \overline{vk})$
Return $M \leftarrow \mathcal{D}(\pi, dk, c)$
```

Figure 10: Games for the proof of Theorem 6.4.

signs a bit by revealing one of two random OWF preimages.

A Merkle tree [28] is used to create a many-time signature scheme from a depth k binary tree of one-time signature instances. To sign a message, a path through the tree is randomly selected. Each internal signature scheme instance of the tree is used to sign a hash of the public keys of its children; the children's public keys are also included in the signature. The final signature instance, in a leaf of the tree, is used to sign the desired message.

Goldreich noted [20] that instead of storing the secret keys for each one-time signature scheme instance, the secret keys can generated as needed by a PRF. Given secret key K for the overall signature scheme, each instance's secret key is the PRF evaluated at the binary path to the instance in the tree. For example, the root node's secret key is $\mathcal{F}(\pi, K, \epsilon)$, and its two children's secret keys are $\mathcal{F}(\pi, K, 0)$ and $\mathcal{F}(\pi, K, 1)$. This allows the secret key for the entire construction to be comprised only of the key for the PRF. See [20] for a formal description of this scheme.

Replacing the PRF with a Φ-RKA-PRF, this scheme is a $C \cup \Phi$-RKA secure and strongly Φ-RKA secure signature scheme. The intuition for these proofs of security follows the proof for Theorem E.1.

F Proof of Theorem 6.4

The proof uses the sequence of games in Figure 10. Game G_0 implements PKE^A with $\mathcal{PKE}$, so we have

$$\mathbf{Adv}^{\text{pke-cca}}_{\mathcal{PKE},A,\Phi}(k) = 2\Pr[G_0^A] - 1.$$

Game G_1 has an additional check in the DEC oracle: now if $\overline{vk} = \overline{vk}^*$ and $(c, \overline{\sigma}) \neq (c^*, \overline{\sigma}^*)$, the game sets bad and responds to the query with $\bot$. Since G_1 and G_0 are identical until bad, we have

$$\Pr[G_1^A] - \Pr[G_0^A] \leq \Pr[E_1],$$

where E_1 is the event that G_1 sets bad. We now construct an adversary B that breaks the strong unforgeability of $\overline{\mathcal{DS}}$ with probability $\Pr[E_1]$. B takes as input $(\overline{\pi}, \overline{vk}^*)$, and starts by selecting $\pi \xleftarrow{\$}$

$\mathcal{P}(1^k)$ and $(mpk, msk) \xleftarrow{\$} \mathcal{K}(\pi)$. It runs $A((\pi, \overline{\pi}), mpk)$, and simulates the response to the LR query exactly as specified in G_1, using $\overline{vk}^*$ when called for and its own signing oracle to generate $\overline{\sigma}^*$ on c^*. It also simulates DEC query responses exactly as specified in G_1, noting the first query that triggers bad, if any. If there is such a query $\text{DEC}(\phi, c, \overline{vk}, \overline{\sigma})$, then B outputs $(c, \overline{\sigma})$ as its forgery. This is a valid forgery for B because this query must have passed the validity check $\overline{\mathcal{V}}(\overline{\pi}, \overline{vk}, c, \overline{\sigma})$ (otherwise the oracle would have returned before going to set bad), and we have that at $(c, \overline{\sigma}) \neq (c^*, \overline{\sigma}^*)$ by the check immediately before bad is set.

Game G_2 rearranged some of the validity checks in DEC oracle processing, but these changes don't affect oracle responses. This change is valid because the same ciphertexts end up being rejected in either game. We have

$$\Pr[G_2^A] = \Pr[G_1^A]$$

We are now in a position to show that an adversary winning G_2 with good probability can be used to break the Φ-RKA security of $\mathcal{IBE}$. We construct A' such that

$$\mathbf{Adv}^{\text{ibe-rka}}_{\mathcal{IBE},A,\Phi}(k) = 2\Pr[G_2^A] - 1.$$

A' takes input (π, mpk), selects $\overline{\pi} \xleftarrow{\$} \overline{\mathcal{P}}(1^k)$ and runs $A((\pi, \overline{\pi}), mpk)$. A' simulates the response to $\text{LR}(m_0, m_1)$ by generating $(\overline{vk}^*, \overline{sk}^*) \xleftarrow{\$} \overline{\mathcal{K}}(\overline{\pi})$ and querying its own oracle for $c^* \leftarrow \text{LR}(\overline{vk}^*, m_0, m_1)$, and then it signs c^* using $\overline{sk}^*$ are returns the ciphertext $C^* = (c^*, \overline{vk}^*, \overline{\sigma}^*)$. A' simulates responses to $\text{DEC}(\phi, (c, \overline{vk}, \overline{\sigma})$ by computing

$msk' \leftarrow \text{KD}(\phi, \overline{vk}^*)$; If $(msk' = \perp$ or $\overline{\mathcal{V}}(\overline{\pi}, \overline{vk}, c, \overline{\sigma}) = 0)$ then return $\perp$
$M \leftarrow \mathcal{D}(\pi, msk', c)$; Return $\mathsf{T}[h_j]$.

A' runs A until it halts, and it outputs whatever A outputs.

To complete the claim we need to argue that A' properly simulates G_2. The only subtlety is in how decryption queries are handled. But we observe that the KD oracle is performing *exactly* the same first two checks that G_2 performs during DEC, and the rest of DEC is properly simulated, and so A' performs as claimed.

The proof is completed by collecting the relationships between games G_0, G_1 and G_2.