New Cryptographic Protocols With Side-Channel Attack Security

by

Rachel A. Miller

B.A. in Computer Science and Physics, University of Virginia, 2009
M.A. in Mathematics, University of Virginia, 2009

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science
at the
Massachusetts Institute of Technology
June 2012

Signature of author: _____

Department of Electrical Engineering and Computer Science
May 23, 2012

Certified by: _____

Shafrira Goldwasser, RSA Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: _____

Leslie A. Kolodziejski, Professor of Electrical Engineering
Chair of the Committee on Graduate Students

New Cryptographic Protocols With Side-Channel Attack Security

by

Rachel A. Miller

Submitted to the Department of Electrical Engineering and Computer Science on May 23, 2012 in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

ABSTRACT

Cryptographic protocols implemented in real world devices are subject to tampering attacks, where adversaries can modify hardware or memory. This thesis studies the security of many different primitives in the Related-Key Attack (RKA) model, where the adversary can modify a secret key. We show how 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 P1 to a primitive P2? 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 without assuming the class of allowed tampering functions is "claw-free"; this mathematical assumption fails to describe how tampering occurs in practice, but was made for all prior constructions in the RKA model. To solve this challenge, we present a new construction of psuedorandom generators that are not only RKA secure but satisfy a new notion of identity-collision-resistance.

Thesis Supervisor: Shafrira Goldwasser
Title: RSA Professor of Electrical Engineering and Computer Science

Abstract

We show how to leverage the RKA (Related-Key Attack) 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 P1 to a primitive P2? 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.

Contents

Acknowledgments

This thesis is based on joint work with Professor Mihir Bellare and David Cash. I'd like to express gratitude for the time that we explored ideas together; you were wonderful to work with!

A very special thank you to my Advisor, Professor Shafi Goldwasser, for such caring mentorship – it has been such a pleasure to work and to learn under you for my time at MIT.

Finally, I'd like to thank my family and my friends for their awesome, incredible support.

1 Introduction

Historical notions of cryptographic security have assumed that adversaries have only black box access to cryptographic primitives; in these security games, adversaries can observe input and output behavior of the protocol, but gain no information from the physical implementation. However, this model fails to account for the fact that adversaries *do* gain additional information from physical implementations: such implementations leak data, such as the time or power it takes to compute with a secret key, and are also susceptible to physical tampering attacks, such as modification of the secret key with microprobes. Attacks that take advantage of this non-black-box nature of protocols are collectively known as *side channel attacks.* Side channel attacks actually exist in practice - many practical side channel attacks have been well documented for almost two decades.

To model the adversarial powers that allow side channel attacks, theoretical cryptographers have developed the enhanced security models of *leakage* and *tampering.* Leakage accounts for information passive adversaries may gain about the contents of secret memory in addition to the input/output behavior of the system. Tampering accounts for adversarial modification of an executing protocol, including changes to the bits in secret memory, the introduction of errors to the computation, and even physical changes to the circuit executing the protocol.

Theoretical works have studied leakage extensively over the last several years, especially in the context of signatures and encryption. In contrast, very few theoretical results address tampering attacks– efficient cryptographic primitives with robust security against tampering are still lacking. Though several constructions of higher level primitives exist, these constructions only protect against highly algebraic and scheme specific tampering attacks [GOR11, AHI11, Luc04, BC10, GL10], or else require heavy and inefficient machinery, such as NIZK proofs[KKS11].

The most commonly used theoretical framework in tampering is the Related-Key Attack (RKA) model, which allows adversarial modification of the secret key. In a related key attack, the adversary is allowed to modify the secret key and subsequently observe values from the cryptographic primitive (such as signatures or ciphertexts) computed with the *modified* secret key. Formally, for a primitive with secret key K, this occurs by allowing the adversary to select a function ϕ, and

giving the adversary values computed with $\phi(K)$ instead of with K. The RKA security definition is parameterized by a class of allowed tampering functions Φ; we restrict how the adversary is allowed to modify the secret key by requiring that the adversary's selected tampering function ϕ satisfies $\phi \in \Phi$. We say a scheme is Φ-RKA secure if security is maintained even when the adversary may perform related key attacks for $\phi \in \Phi$. A second commonly used framework is the Self-Destruct model, which is a relaxation of the RKA model allowing the primitive to destroy secret memory if tampering is detected.

The psuedo-random function (PRF) is the most studied primitive in the RKA model. A PRF is a family of functions, where individual functions are defined by evaluating the function using a fixed (secret) key. In the security game, an adversary is either given oracle access to a truly random function, or oracle access to the function defined by the PRF evaluated with a randomly generated secret key. A family of functions is a secure PRF if all polynomial time adversaries can only distinguish between the two cases with a negligible advantage over guessing.

This thesis will give a formal definition of a Φ-RKA secure PRF, but it will be helpful to give an intuitive definition here. When we consider PRFs in the RKA model, the adversary is given the power to modify the secret key, and so should receive oracle access to the PRF evaluated with many different key values, defining a group of functions instead of a single function. Accordingly, in the modified security game for a Φ-RKA PRF, the adversary tries to distinguish between oracle access to the family of functions Φ-RKA PRF and a *family* of truly random functions, both evaluated at a randomly generated secret key; in both cases, if the original secret key is K, the adversary obtains access to the oracle evaluated with $\phi(K)$ in the family of functions for $\phi \in \Phi$. We say a PRF is Φ-RKA secure if all polynomial time adversaries have a negligible advantage in distinguishing between these two cases.

It turns out that a Φ-RKA PRF is an especially powerful building block for constructing other tamper-resilient primitives. This thesis will give a construction for any Φ-RKA secure cryptographic primitive, using any Φ-RKA PRF and a normal secure instance of the primitive as components. Interestingly, it appears the security requirements for Φ-RKA secure PRFs seem to be especially difficult to satisfy – there exist classes of tampering functions Φ for which Φ-RKA secure signature

and encryption schemes exist, but for which *no* PRF can be Φ-RKA secure.

The construction of tamper-resilient primitives from Φ-RKA secure PRFs, and the apparent difficulty of constructing Φ-RKA secure PRFs, lead to further questions about the relationships between Φ-RKA secure primitives. For example, when can one Φ-RKA secure primitive be used to build a different Φ-RKA secure primitive in a construction preserving Φ? Are there Φ for which Φ-RKA secure instantiations of some primitives are known, but for which Φ-RKA security is provably impossible for another primitive?

1.1 Contributions

Prior works in the RKA model have all made the claw-free assumption[AHI11, BC10, GL10, GOR11]. A class of functions Φ is *claw-free* if for any two distinct $\phi_1, \phi_2 \in \Phi$, and for any K in the domain of Φ, $\phi_1(K) \neq \phi_2(K)$. In the context of the RKA model, the *claw-free assumption* is that the class of allowed tampering functions Φ an adversary may use, which has domain of the secret key space, is claw-free. Unfortunately, the claw-free assumption fails to capture the tampering attacks that exist in practice. This work drops the claw-free assumption in the RKA model for the first time.

This thesis gives a generic way to translate the standard, non-tampering security definition for a cryptographic primitive into a Φ-RKA security definition in the RKA model. With this, we give generic construction of Φ-RKA secure primitives from any normal secure instance of the primitive and any Φ-RKA secure PRF – in other words, we construct primitives that are resistant to the same class of tampering functions Φ as the PRF used. This construction maintains the efficiency of the original Φ-RKA PRF. Further, this construction uses the Φ-RKA PRF in a black-box way, so any future Φ-RKA PRFs will immediately give a suite of Φ-RKA primitives using this construction. We note that a similar construction was given in concurrent work[GOR11], but our construction in addition is able to drop the claw-free assumption.

Motivated by this construction of Φ-RKA secure primitives from Φ-RKA secure PRFs, we further explore when it is possible to construct Φ-RKA secure primitives from other Φ-RKA secure primitives. When is it possible to give a reduction from one primitive to another in a way that

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

Table 1: A summary of relationships we will show; gives whether the row is contained or not contained in the column.

preserves the class of allowed tampering functions Φ? We give positive results across a wide variety of primitives: PRFs, wPRFs, IBE, Signatures, SE-CCA, SE-CPA, and PKE-CCA. For example, we show that Φ-RKA secure IBE implies Φ-RKA secure IND-CCA PKE. We also give some negative results, showing, for example, that there exists a Φ for which Φ-RKA secure signatures exist but no Φ-RKA secure PRFs exist. Inspired by this fact , we give additional negative results: for example, there exists a Φ such that a Φ-RKA secure wPRF exists but for which no Φ-RKA secure PRF exists.

We view the relationships between different Φ-RKA secure primitives as analogous to complexity theory – the relations between these primitives should help us better understand the complexity of each particular problem. We will express these relationships in a set based notation. We define $\mathbf{RKA}[\mathsf{P}]$ to be the set of all Φ for which Φ-RKA secure versions of primitive P exist. In this framework, a construction of a Φ-RKA secure primitive P_2 from any Φ-RKA secure primitive P_1 will be expressed as $\mathbf{RKA}[\mathsf{P}_1] \subseteq \mathbf{RKA}[\mathsf{P}_2]$; these relationships will be shown in Section 9. Showing that there $\exists\Phi$ such that $\Phi \in \mathsf{P}_1$ but $\Phi \notin \mathsf{P}_2$ will be expressed $\mathbf{RKA}[\mathsf{P}_1] \not\subseteq \mathbf{RKA}[\mathsf{P}_2]$; these relationships will be shown in Section 10. In a few cases we will need to make additional assumptions, which we will note in the theorem.

Additionally, this thesis contains an extensive literature search of physical tampering attacks, in addition to referencing prior theoretical works. This study can guide which tampering attacks should be modeled by and addressed in theory.

2 Motivation - Tampering is Real!

Tampering is a major security concern when adversaries have physical access to the device running the cryptographic primitive, which is often the case for devices like smart-cards, pay-TV decoders, and GSM mobile phones. Successful attacks have taken a variety of forms, including chip re-writing, differential analysis, and observing memory remanence after the device is turned off[AK96][AK97]. Each of these attacks will be further described below.

2.1 Hardware Errors

Even in standard conditions, circuits are not perfect, and computations may contain errors. Though errors are typically rare, faulty hardware can make them much more likely. For example, a bug in the family of Pentium FDIV processors produced errors in floating point division[SB94]. Though errors for these faulty chips were rare for randomly generated floating point operations, affecting about 1 in 9 billion operations, they could be quite common for adversarially selected input.

2.2 Transient Faults

Transient faults are errors in computation caused by very temporary changes to conditions in the circuit executing the protocol or to the environment of the circuit. The most common attacks apply quick changes to circuitry's clock signal, power supply, or external electrical field[KK99]. Applying voltages or temperatures outside the normal range of the circuit can affect some write operations. One family of chips was susceptible to an attack that modifies the applied voltage during repeated write access to a security bit- this will often clear the bit without erasing the rest of secret memory[AK96]. Additionally, varying voltage can hamper some processes from running properly: another family of chips contained random number generator used for cryptographic processes, but the generator output a string of all 1's when a low voltage was applied, yielding very guessable secret keys[AK96].

For some processors, transient changes to the applied voltage or to the clock signal can vary the decoding and execution of individual instructions. Since different signal paths of information on a

chip pass through different numbers of gates, they each take (slightly) different amounts of time. For the whole computation to be correctly executed, every path must finish before the next clock signal. If an adversary can cause a small number of paths to fail to execute correctly, the protocol will yield values strongly related to the correct values but with some errors. By applying a precisely timed glitch to the clock signal, a limited number of signal paths can be affected. Similarly, changes to the applied voltage changes the amount of time each signal path takes to complete. Increasing time signals take can prevent some signal paths from completing. This has the potential to cause the CPU to execute a number of wrong instructions, or allow differential analysis[AK96].

Conditional jumps are commonly among the longest signal paths; if transient errors allow attackers to eliminate conditional jumps or test instructions[KK99], such errors can often allow them to bypass security conditions. Instruction glitches can also extend the runtime of loops; for example, an attacker might be able to extend a loop designed to print out an output buffer to print out additional values in secret memory[AK96]. Breaking out of a loop early can also be harmful, transforming a many round cipher into an easily breakable single round variant[AK97].

Varying the external electrical field can be accomplished by applying spikes of up to several hundred volts very locally on a chip by using two metal needles. Such a technique to be used to temporarily shift the voltages of nearby transistors, creating variations in signal paths based on locality instead of timing[KK99].

2.3 Overwrite attacks

Laser cutter microscopes can be used to overwrite single bits of ROM[AK97]. In the case of that the circuitry for a protocol is well known, such as for DES, attackers could find and overwrite specific bits to make key extraction easy, such as the bits the control the number of rounds for the cipher[AK97]. As another example, the lock bit of several devices with on-chip EPROM can be erased by focusing UV light on the security lock cell, which is located sufficiently far from the rest of memory. [AK96]

Microprobing needles can be used to set or reset a target bit stored in EEPROM. This can be useful for a number of attacks. Consider the following attack on DES: DES keys are required

to have odd parity, and DES returns an error message if this fails to hold. To attack DES, an adversary can find the key by setting one bit at a time to 0, and seeing if the error message is triggered[AK97].

Circuits are often constructed with built in "test" functionalities, such as printing out the contents of memory, that are destroyed after testing is completed. Bovenlander[Bov] describes an attack on smartcards using two microprobe needles to bridge the blown fuse for test functionalities, re-enabling a test routine that prints out secret memory.

Finally, lasers can also be used to destroy specific gates in a circuit, fixing their output to 0. This can be used to simplify the relationship between the secret key and the output, perhaps making the secret key easier to find[AK97].

2.4 Memory Remanence Attacks

Gutman[Gut96] describes the issue of "burn-in" in both static RAM (SRAM) and dynamic RAM (DRAM). When values are stored in the same RAM location for a significant period of time, physical changes to magnetic memory occur and leave a lasting change. To avoid these effects, sensitive information should not be stored in a single RAM location for more than several minutes.

In addition to the physical changes to memory that occur over the period of minutes, even the temporarily stored capacitance of information stored only momentarily in DRAM leaves a remanence. This remanence lasts for at least several seconds after power is removed, even at room temperature and even if removed from the motherboard. This gives adversaries with physical access to the device a (brief) chance to gain information about any key stored in DRAM[CPGR05, HSH+08]. SRAM also leaves significant traces for several seconds at room temperature[Sko02]. Theoretical works show that even a small amount of memory remanence is a problem; for example, an RSA key can be recovered from only 0.27 of its bits selected randomly[HS09].

2.5 Protocol Failures

Poorly designed protocols and hardware make attacks even simpler for an adversary to execute. For example, as documented in [AN95], one poorly designed satellite TV decoder had a processor

to store the secret key and to decrypt encoded video signals, and additionally a microcontroller to handle commands addressed to a customer smartcard ID. Unfortunately, these pieces were very modular and replacable. In the "Kentucky Fried Chip" attack, the microcontroller is replaced with one created to block deactivation messages with the customer's ID the satellite company sends out when a user stops paying for service.

2.6 Self-destructing Memory

There are several examples of circuits that self-destruct and destroy their memory when physical tampering of hardware is detected[Sko02, Gut96, SPW99]. However, this feature is rarely used in practice, and fails to detect some types of tampering.

Self-destructing memory is accomplished by using volatile memory inside a tamper-detecting enclosure. When tampering is detected, the volatile memory is powered down or even shorted to ground. It is important to note that this method only prevents attacks that the tamper-detecting enclosure can detect. Even more crucially, since memory takes some amount of time to decay after power is removed, this method requires that an adversary cannot read memory for some noticeable amount of time after tampering is first detected.

Tamper detection sensors often are sensitive to changes in voltage or other environmental conditions[AK96], since these can cause transient faults. However, self-destruct mechanisms can decrease the reliability of hardware. One family of smartcards was designed to self-destruct when they detected low clock frequencies, but had relatively common false positives due to the large fluctuations in clock frequency on card startup[AK96]. Additionally, since many transient conditions exist naturally in circuits, for robustness sensors must be designed in such a way that they are not sensitive to transient changes in voltage[AK96], limiting the applicability of this model.

Several examples of self-destructing software exist. The most popular is the current iPhone operating system, which can be set to clear memory after 10 failed passcode attempts. Unfortunately, a software based self-destruct model does not provide security guarantees in the case of physical tampering, which is the motivation for tampering security.

3 Models of Tampering

The most general tampering models must account for all possible adversarial modifications to a protocol being executed. This includes modification of the secret key, modification of intermediate computed values (possibly as fine grained as changes to specific wires), and even modification of the program code being executed. Indeed, all such tampering attacks have occurred in practice!

The vast majority of works on tampering only deal with the tampering of secret keys. These works divide nicely into two different theoretical frameworks. The first framework , the *Related-Key Attack (RKA)* model, was inspired by [Bih93], and gives the adversary the ability to modify the secret-key using some family of allowable tampering functions. The second framework, the *Self-Destruct model*, is a relaxation of the RKA model that allows secret memory to be destroyed tampering is detected to prevent further leakage of information.

This thesis will work in the RKA model, which only allows tampering on the secret key value. In this model, adversaries may replace the original secret key with a modified one between consecutive evaluations of the protocol. This is formalized by letting the adversary select a function, and replacing the secret key with this function evaluated on the secret key. Although some works in the Self-Destruct model allow the adversary to apply arbitrary polynomial time tampering functions, such strong results are actually impossible in the RKA model[KKS11]. Accordingly, works in the RKA model must further restrict which tampering functions are allowed to an adversary. Fortunately, such restrictions seem justified by the limited state of existing physical attacks, which modify a small number of bits in memory, or introduce randomized errors. On the other hand, it's important not to restrict the allowed attacks too far.

The Self-Destruct model is similar to the RKA model in that it allows adversaries to select and apply tampering functions to the secret key, but it relaxes the correctness requirements of the protocol. Here, the protocol to "self-destruct" and completely overwrite secret memory if tampering is detected. A second differentiation between the models is the subtle issue of how many devices with identical memory are available to the adversary. Though not necessarily inherent in the Self-Destruct model, all works to date in this model allow adversaries a logarithmic number

of bits of information about the secret key per device, and so require that the adversary only has access to a constant number of copies of devices with the same secret key. To contrast, the RKA model implicitly allows the adversary access to an unbounded number of copies, since the tampering function ϕ is applied to the secret key in every iteration. Since the Self-Destruct model gives relaxed conditions for correctness of protocols, and all works to date limit the number of copies of the device given to adversaries, the Self-Destruct model allows security against a strictly larger class of tampering functions than the RKA model[KKS11].

We note that modeling tampering of intermediate values is very difficult, since this depends on a specific instantiation of a protocol– only one theoretical work [IPSW06] allows arbitrary adversarial tampering of intermediate values in the circuit used to execute the computation. More common are works addressing random faults in computation, a body of work initiated by [BDL01] and referred to as *differential fault analysis*. No theoretical works to date consider modification of the program being executed.

3.1 The History of the RKA Model

Differential cryptanalysis was a technique developed by Biham and Shamir[BS90] that attacks block ciphers by observing encryptions of pairs of messages with known differences in their plaintexts; the differences between the resulting ciphertexts can give significant information about the secret key for many block ciphers. Differential cryptanalysis soon gave rise to the Related Key Attack (RKA) model, also defined in the context of block ciphers[Bih93]: instead of known relationships between plaintexts as in differential cryptanalysis, the RKA model uses known relationships between successive secret keys used. The key-scheduling algorithms used in block ciphers are generally public and well known, giving adversaries information about the relationship between successive keys. In some cases the adversary can even cause the system to move forward or backwards in the key-schedule, giving control over the key being used. Security against RKAs is now accepted as a requirement for blockciphers to be used in practice. In fact, AES was designed with the explicit goal of resisting RKAs [DR02].

After block ciphers, RKAs were next considered in the context of PRFs and psuedorandom

permutations, PRPs. Bellare and Kohno[BK03] provided formal theoretical definitions for both Φ-RKA secure PRFs and Φ-RKA secure PRPs. They gave important impossibility results: there exist polynomial time tampering functions for which no block-cipher can be secure. They also showed that secure block ciphers are secure against some very limited tampering attacks. Lucks[Luc04] constructed a block cipher secure against some more interesting RKAs, but used a non-standard number theoretic assumption, and also restricted the adversary to tampering a fixed half of the secret key. The later is an especially strong assumption since one can trivially construct a block cipher resilient against such tampering from any normal secure block cipher. To do this, have the new block cipher generate a secret key that is twice the length of the original block cipher, and only use the half that the adversary can't tamper.

Bellare and Cash[BC10] gave the first construction of a Φ-RKA secure PRF under standard assumptions, using the Decisional Diffie-Hellman (DDH) assumption. In their construction, the adversary may modify the secret key by either adding a fixed element in the group of secret keys, or else multiplying by a fixed element in the group of secret keys.

Goldenberg and Liskov[GL10] defined and considered Φ-RKA secure hardcore bits and one-way functions; for these primitives, the tampering function is applied to the secret input of the function instead of a secret key, and the security challenge still depends on the original un-altered input to the function. They also consider Φ-RKA psuedorandom generators, and show that Φ-RKA secure psuedorandom bits can be used to build Φ-RKA secure block ciphers. They additionally show that hard-core bits with typical proofs are *not* Φ-RKA secure pseudorandom bits, emphasizing the difficulty of constructing tamper-resilient psuedorandom primitives.

Applebaum, Harnik and Ishai[AHI11] gave a formal study of Φ-RKA secure symmetric encryption, and gave two constructions of a Φ-RKA secure encryption scheme where Φ is comprised of functions adding constants over the group of secret keys. They use standard assumptions, giving one construction based DDH, and the other based on lattice assumptions such as learning parity with noise (LPN) or learning with errors (LWE). They also give a construction for a Φ-RKA secure PRG under DDH, with the same Φ as their encryption scheme construction. They show that a Φ-RKA secure PRG can be used to generate a Φ-RKA secure encryption scheme when Φ is claw-free.

Finally, they show that Φ-RKA secure PRGs can be used for several other applications, such as correlation-robust hash functions as defined in [IKNP03].

A work by Goyal, O'Neill and Rao [GOR11], developed concurrently to work in this thesis, defines correlated-input (CI) hash functions, which are a generalization of the correlation-robust hash functions of [IKNP03]. As with RKA-security, the security of CI hash functions is parameterized by a class of allowed tampering functions Φ. They show that CI hash functions, with security parameterized by Φ, can be used Φ-RKA secure signature schemes. (They also indicate their approach can also be used to build other Φ-RKA secure primitives from CI hash functions as well.) Their construction is similar to one included in this thesis, but their very definition of CI hash functions immediately requires that Φ is claw-free, while this thesis is able to drop the claw-free assumption. For a CI hash function H and unique tampering functions $\phi_1, \ldots, \phi_{n+1} \in \Phi$, they require that no adversary can predict $H(\phi_{i+1}(r))$ from $H(\phi_1(r)), \ldots, H(\phi_n(r))$. If $\phi_{n+1}(r) = \phi_i(r)$ for some previous i, this is trivial! Further, the adversary of the CI hash function must select all tampering functions ϕ_i before even seeing the public hash-key, giving a non-adaptive form of security. [GOR11] gives a construction of a CI hash function where Φ is the class of polynomials over the input space, based on a variant of the q-Diffie Hellman Inversion assumption. Although the class of all polynomials is a very broad class, requiring the adversary to select the tampering functions non-adaptively gives that $\phi_i(r) = \phi_j(r)$ only with negligible probability, which is used crucially in their security proofs.

3.1.1 Claw-free assumption

As mentioned previously, the claw-free assumption for works in the RKA model requires that any two distinct tampering functions $\phi_1, \phi_2 \in \Phi$ must disagree for every secret key in the key space - for all $K \in \mathcal{K}$, $\phi_1(K) \neq \phi_2(K)$. Unfortunately, this assumption fails to hold for any tampering attacks used in practice and discussed in Section 2, including random bit errors, setting a subset of bits of the secret key to a fixed value, and memory remanence attacks. Additionally, the claw-free assumption is quite restrictive: it even disallows classes of tampering functions that intuitively give very little power to an adversary. For example, it disallows tampering functions that only modify

the secret key on a negligible number of secret key values.

Prior Works. Prior to this work, all constructions of Φ-RKA secure primitives relied on the claw-free assumption. This includes the PRF construction of [BK03] which holds for classes of claw-free permutations, the block cipher construction of [Luc04] which holds for classes of claw-free functions over the half of the secret key the adversary can tamper, the PRF construction of [BC10] which holds for the claw-free classes of addition or multiplication within the group of secret keys, the symmetric encryption schemes of [AHI11] which holds for the claw-free family of addition over the key space, and finally for the CI hash constructions of [GOR11] in which the very definition of security requirements implies that the family of allowed tampering functions must be claw-free.

Why is the Claw-Free Assumption so Useful? To reflect how tampering works in practice, in the security game for a Φ-RKA secure primitive the adversary should be able to choose *not* to modify the secret key. To model this, we let ϕ_{ID} be the identity function that maps any key back to itself, and we let $\mathsf{ID} = \{\phi_{\mathsf{ID}}\}$. In effect, ID-RKA security does not allow the adversary to modify the secret key, and so ID-RKA security should reduce to the standard (no tampering) security definition for any primitive. In general, we will assume that $\mathsf{ID} \subseteq \Phi$, and so a Φ-RKA secure primitive should always have standard security for the primitive. The claw-free assumption, combined with the assumption that $\mathsf{ID} \subseteq \Phi$, implies that no other tampering function $\phi \in \Phi$ can ever map a secret key onto itself – otherwise $\phi(K) = \phi_{\mathsf{ID}}(K) = K$.

The absence of the claw-free assumption creates serious technical difficulties in the security proof for a Φ-RKA secure primitive. Generally, the proof proceeds via a reduction of the Φ-RKA primitive to normal (not RKA) security of the primitive; this occurs by giving the adversary values from the normal primitive when the secret key is unmodified. However, the reduction only sees the output of the primitive, and not the hidden secret key, so it is difficult for the reduction to tell when the secret key is unmodified. The claw-free assumption avoids this difficulty since the secret key is only unmodified when the tampering function chosen is ϕ_{ID}.

3.2 Protection versus Detection: The History of the Self-destruct Model

It is impossible for many primitives to be secure in the RKA model against arbitrary polynomial time tampering functions. Consider the following example from [KKS11]. Let Φ contain functions that set a specific bit of the secret key to 0; then Φ-RKA secure signature schemes cannot exist. The adversary can simply proceed bit by bit, setting the bit to 0 and seeing if the scheme will still produce a valid signature under the original verification key; if it does, that bit of the signing key is a 0, and otherwise, it is a 1. Since there is no way to protect against this Φ in this model, and this Φ is a very small subset of all polynomial time functions, it is clear that Φ-RKA secure signatures do not exist when the adversary is allowed to apply arbitrary polynomial time tampering functions to the secret key.

Since many general security guarantees are not possible in the RKA model, many works add additional assumptions. A common additional assumption is self-destruction functionality, were the primitive can completely destroy secret memory if tampering is detected.

The first theoretical work to assume a self-destruct mechanism was the work of Ishai, Prabhakaran, Sahai, and Wagner[IPSW06]. This work gives a compiler from a general circuit to a circuit that allows setting *any* wire in the circuit to either 0 or to 1 as the computation progress. However, this result requires a blow-up of a factor of t in the size of the compiled circuit to allow the adversary to tamper t wires in between successive runs of the protocol. Another work that gives a general compiler for tampering resilient circuits is recent work by Kalai, Lewko and Waters : they give a general compiler for circuits to allow up $\frac{1}{10}$th of the gates on any path to have *short-circuit errors*, a limited form of tampering that adversarially replaces the output of a gate by either one of its inputs.

Work by Genero et. al[GLM+04] gives a compiler to allow tampering by arbitrary efficient tampering functions by storing a signature of the secret state along with the secret state. In addition to assuming self-destruct functionality, this requires the very strong assumption of having a secret signing key to be hard-wired into the circuit, and so avoids tampering on this. This work is extended in [LL10] to model both tampering and leakage of specific memory locations and wire values; they give both impossibility results and a modification of [IPSW06] to give a positive result.

However, the [LL10] model assumes a secret key space that is polynomial, not exponential, in the security parameter, giving adversaries a non-negligible advantage even without tampering; they also assume arbitrary polynomial time tampering functions, giving the tampering function the ability to depend on all potential secret keys, and so the ability to depend on the correct secret key.

Work by Dziembowski, Krzysztof and Wichs[DPW10] define the security notion of non-malleable codes (NMC). When an adversary is allowed to modify secret memory that has been encoded with a NMC, the NMC security guarantees that the resulting output of the protocol is unmodified, or else is completely unrelated to the original value of secret memory. By adding components to encode and decode into the NMC, [DPW10] gives a way to compile a circuit into one that gives non-malleable security. In particular, they give a construction of a NMC that protects against tampering that modifies each bit of secret memory independently, and they show that NMCs exist for a much broader classes of tampering functions.

The final and most recent work using the self-destruct assumption is that of Tauman-Kalai, Kanukurthi, and Sahai[KKS11], which gives the first signature and encryption schemes that are secure against both leakage *and* tampering. Though the tampering functions can be chosen arbitrarily, the adversary is only allowed a bounded number of tampering queries per time period. Their work additionally relies on the assumption of a non-tamperable common reference string (CRS) that is available to all parties, and uses non-interactive zero knowledge (NIZK) proofs.

3.3 The History of Tampering Intermediate Values

Boneh, DeMillo and Lipton[BDL01] first explored the dangers of errors in computation– even a very small number of bit flips in intermediate values in a protocol can give a full break of an otherwise secure scheme. Specifically, they showed that both a common implementation of the Fiat-Shamir identification protocol and a Chinese-Remainder-Theorem based implementation of RSA signatures are insecure when an adversary can induce a small number of random bit-flips in intermediate values. This paper inspired a body of work called *differential fault analysis* (DFA), which focuses on exploits of blockciphers caused by small bitwise errors; see [BS97] for a long list of ciphers broken by DFA.

Protecting protocols against arbitrary intermediate values is very difficult, as this depends on the specific instantiation of a protocol, and requires analysis of tampering of each intermediate value in the computation. The only works to date that allows non-random errors in intermediate values are the works by Ishai, Prabhakaran, Sahai, and Wagner [IPSW06] and by Kalai, Lewko and Waters mentioned above in the History of the Self-Destruct model 3.2.

4 Notation & Preliminaries

4.1 Notation

For a set S, let $|S|$ denote the size of S, and let $s \xleftarrow{\$} S$ denote the operation of picking a random element of S and calling it s.

For sets X and Y, let $\mathsf{Fun}(X, Y)$ be the set of all functions mapping X into Y. When Z is also a set, let $\mathsf{FF}(X, Y, Z) = \mathsf{Fun}(X \times Y, Z)$. A class of functions Φ will be associated with some domain X and range Y, and will be a subset of all functions mapping X to Y- $\Phi \subseteq \mathsf{Fun}(X, Y)$. A class of functions Φ associated with domain $\mathcal{K}$ will be said to be *claw-free* if $\forall \phi_1, \phi_2 \in \Phi$ with $\phi_1 \neq \phi_2$, $\forall K \in \mathcal{K}$, $\phi_1(K) \neq \phi_2(K)$. In the context of work on tampering, the *claw-free assumption* requires that the family of allowed tampering functions Φ is claw-free.

A function family is a tuple of algorithms: a parameter generator, a key generator, and a deterministic evaluator, $\mathcal{FF} = (\mathcal{P}, \mathcal{K}, \mathcal{F})$. For each $k \in \mathbb{N}$, the function family also defines sets $\mathsf{Dom}(\cdot)$ and $\mathsf{Rng}(\cdot)$ such that $\mathcal{FF}(K, \cdot)$ maps elements of $\mathsf{Dom}(k)$ to $\mathsf{Rng}(k)$. We will assume that each function family has associated polynomials d for the input length with $\mathsf{Dom}(k) \subseteq \{0,1\}^{d(k)}$, and output length l with $\mathsf{Rng}(k) \subseteq \{0,1\}^{l(k)}$.

For a bit b, we use $\bar{b}$ to denote the opposite of b, i.e. $\bar{0} = 1$ and $\bar{1} = 0$.

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. For $i \leq j$, $x[i,j]$ denotes the inclusive range from i-th bit through the j-th bit of x.

When $\mathbf{v}$ is a vector, $|\mathbf{v}|$ gives the number of coordinates of $\mathbf{v}$, and $\mathbf{v}[i]$ is used to denote its i-th coordinate- $\mathbf{v} = (\mathbf{v}[1], \ldots, \mathbf{v}[|\mathbf{v}|])$.

Algorithms. Unless otherwise noted, algorithms are randomized and are polynomial time in the

size of their inputs.

For algorithms A and B, we let $A \,\|\, B$ denotes the algorithm that on any input x returns $A(x) \,\|\, B(x)$.

We use notation for a fixed sequence of probabilistic functions, where each function is evaluated with fresh randomness, and the final object is returned: $\{x \xleftarrow{\$} A, y \xleftarrow{\$} B(x); C(x,y)\}$ represents events A, B and C being executed in order, with the returned value equal to the returned value of C.

4.2 Cryptographic Primitives & Security Games

A *cryptographic primitive* P is a tuple of algorithms, which include a key generation algorithm $\mathcal{K}$ that on input a security parameter k outputs a secret key sk and optionally also a public key pk, and a number of algorithms that may take the secret key as one of their inputs. We represent a cryptographic primitive as $\mathsf{P} = (\mathcal{K}, g_1(\cdot), \ldots, g_m(\cdot), f_1(sk,\cdot), \ldots, f_n(sk,\cdot))$, where each algorithm f_i depends on the secret key sk, and each algorithm g_i does not. Without loss of generality, all algorithms run in time polynomial in k.

Security Games.

Every cryptographic primitive P also has an associated security definition, which in our work will be given as a *security game* with a security parameter k. A security game defines an interaction between two algorithms: a stateful challenger algorithm, and a stateful adversary algorithm, both of which are required to run in time polynomial in k. The security game can be played with a class of potential adversary algorithms that conform to required interactions of the game.

A security game is comprised of *phases*, which are distinct periods of interaction between the adversary and the challenger. Each phase might optionally include the challenger correctly executing required computation, messages between the challenger and the adversary, and adversary access the output of some of the algorithms f_i in P that depend on the secret key.

When the adversary is allowed to view output of algorithms f_i that depend on the secret key, this is modeled as the adversary gaining access to an oracle that computes f_i , denoted as $\mathcal{O}^{f_i}$. An oracle call to $\mathcal{O}^{f_i}(\cdot)$ returns $f_i(sk,\cdot)$. We number the phases, and for each phase j we define the set

of allowed oracle access S_j as containing i if the adversary has oracle access to $\mathcal{O}^{f_i}$ during phase j.

As part of the security definition, some input combinations may be disallowed by the oracle, where the set of disallowed input may be based on values chosen during the interactive security game. (For example, the security game for an IND-CCA2 encryption scheme disallows the adversary from requesting a decryption of its challenge ciphertext.) Additionally, the disallowed inputs might differ even for oracle access to the same f_i during different phases. The notion of disallowed requests is formalized by a function $\mathsf{disallow}_{i,j}(\cdot)$ to disallow queries to $\mathcal{O}^{f_i}$ during phase j; $\mathsf{disallow}_{i,j}(x_1, \ldots, x_k, \cdot)$ returns true if $i \notin S_j$ or $x_1, \ldots, x_k$ should be disallowed based on the security game in phase j, and false otherwise. In phase j, the oracle for f_i sends the adversary either $\perp$ if $\mathsf{disallow}_{i,j}(x_1, \ldots, x_k, \cdot)$ is true, or sends $f_i(sk, x_1, \ldots, x_k)$ otherwise.

Each security game is associated with a boolean function function that has value 1 when the primitive is considered broken, and has a required bound on the probability of the break for the primitive to be considered *secure*. Here, the probability of a break is measured over the choice of randomness for both the challenger and for the adversary. The function break might depend on the message transcript between the challenger and the adversary or on the state of the challenger and the adversary. The scheme might be considered broken when the adversary can distinguish between two possible behaviors of the challenger by guessing a secret bit selected by the game, or when the adversary generates a value that satisfies a certain relationship with the secret or public keys of the scheme. In the last case, even if the submitted values satisfy the stated relationship, some values might not be considered a valid break based on prior values that have occurred in the security game. (For example, in the game defining security for a signature scheme, even a valid message signature pair is not a break if the adversary made a prior oracle query on that message.)

We use G^A to denote that a security game G should be executed with adversary A that is within the class of allowed adversaries for G. Some of our games will return a value; we will use $(\mathrm{G}^A \Rightarrow 1)$ to denote the event that G played with A returns a value of 1.

5 RKA-Model: Formal Treatment

So far our description of adversarially selected tampering functions has been loose, but we will now formalize structural requirements on them. For example, such functions must map secret keys back into the secret key space. We will call functions that meet these structural requirements a *Related-Key Deriving* function.

Definition 5.1 (Related-Key Deriving Functions – from [BK03]) *Consider a cryptographic primitive* P *with security parameter* k. *Let* $\mathcal{K}$ *be the space of secret keys for* P, *which is the range of secret keys created by key generation run on input* 1^k.

We say a function ϕ *is a* Related-Key Deriving *(RKD) function is* compatible *with* P *if it satisfies* $\phi : \mathcal{K} \to \mathcal{K}$.

We define $\mathsf{RKD} = \mathsf{Fun}(\mathcal{K}, \mathcal{K})$, *the set of all RKD functions for a key-space. A* class of RKD functions, *typically denoted by* Φ, *is a set* $\Phi \subseteq \mathsf{RKD}$. *A class* Φ *is said to be* compatible *with a primitive* P *if all* $\phi \in \Phi$ *are compatible with* P.

Example Classes of RKD Functions. The RKA model is parameterized by classes of allowed RKD functions. One class that will be used repeatedly in our proofs is the class ID, which contains only the identity function. Another class that we will reference several times is the class of all constant functions, which will be denoted by Cnst.

Definition 5.2 (ID, the class of the identity function) *Let* $\phi_{\mathsf{ID}}(x) = x$; $\mathsf{ID} = \{\phi_{\mathsf{ID}}\}$ *is the class containing the singleton* ϕ_{ID}.

Definition 5.3 (Cnst, the class of all constant functions) *For any* $c \in \mathcal{K}$, ϕ_c: $\mathcal{K} \to \mathcal{K}$ *is given by* $\phi_c(x) = c$ *for all* $x \in \mathcal{K}$, *returning* c *for all input. Then* Cnst *is defined as the set containing* ϕ_c *for all* $c \in \mathcal{K}$.

Oracle Access with Tampering. As we move to the Φ-RKA model, we will also want to give adversaries oracle access to values computed with tampered keys. We will use $\mathcal{O}^{f_i,\Phi}$ to denote oracle access to f_i with allowed tampering in Φ. An adversary with such access can call $\mathcal{O}^{f_i,\Phi}(\phi, \cdot)$;

if the query is disallowed or if $\phi \notin \Phi$ the oracle will return $\perp$, and otherwise the oracle will return $f_i(\phi(sk), \cdot)$.

RKA Model.

We will now formally define the Related Key Attack (RKA) model for an adversary of a cryptographic primitive, where the allowed tampering is parameterized by a class of Related Key Deriving (RKD) functions Φ. Recall that a cryptographic primitive P is a tuple of algorithms, some which take a secret key sk as input. In the standard security game for the primitive, an adversary receives oracle access (for a specified period of time) to some of the algorithms that are evaluated using sk. As examples: in the game defining signature schemes, the adversary receives oracle access to the signing algorithm $\mathcal{S}$, receiving $\mathcal{S}(sk, m)$ on query $\mathcal{O}^{\mathcal{S}}(m)$; in the game defining IND-CCA2 security for a symmetric key encryption scheme, the adversary receives access to both an encryption oracle $\mathcal{O}^{\mathcal{E}}(m)$ and to a decryption oracle $\mathcal{O}^{\mathcal{D}}(c)$, receiving $\mathcal{E}(sk, m)$ and $\mathcal{D}(sk, c)$ in response, except for the disallowed decryption query of the challenge ciphertext.

We define a Φ-RKA adversary against P to have all the abilities of a standard adversary against P, and in addition receives oracle access to the same algorithms as the standard adversary, except evaluated using $\phi(sk)$ for $\phi \in \Phi$ instead of sk. Returning to our examples: in the modified game for signature schemes, the adversary will now receive oracle access to $\mathcal{O}^{\mathcal{S},\Phi}$, and on query $\mathcal{O}^{\mathcal{S},\Phi}(\phi, m)$ an adversary receives a signature generated with tampered sk, $\mathcal{S}(\phi(sk), m)$ in response; for the game defining IND-CCA2 security for a symmetric encryption scheme, the adversary now receives access to encryption oracle $\mathcal{O}^{\mathcal{E},\Phi}(\phi, m)$ and decryption oracle $\mathcal{O}^{\mathcal{D},\Phi}(\phi, c)$, receiving encryptions under the tampered key $\mathcal{E}(\phi(sk), m)$ and decryptions under the tampered key $\mathcal{D}(\phi(sk), c)$ in response.

Moving to the RKA model causes some delicate issues to arise in when a primitive is considered broken by a Φ-RKA adversary. As an example, lets consider a Φ-RKA secure signature scheme, comprised of algorithms $(\mathcal{K}, \mathcal{S}, \mathcal{V})$ for key generation, signing, and verification respectively. In the standard security definition for a signature scheme, a signing key and verification key are produced as $(sk, vk) \xleftarrow{\$} \mathcal{K}$. An adversary adaptively asks the signing oracle for signatures for messages m_i as $\mathcal{O}^{\mathcal{S}}(m_i)$ and receives $\mathcal{S}(sk, m_i)$ in return; the scheme is considered broken if the adversary can produce a message m^* and signature σ^* such that $\mathcal{V}(vk, m^*, \sigma^*)$ returns true and such that the

adversary never queried for a signature of m^*.

What should be considered a break for a Φ-RKA adversary against a signature scheme? Such an adversary will adaptively make queries for signatures of m_i with tampering function $\phi_i \in \Phi$ to a signing oracle that will create signatures with tampered keys, denoted by $\mathcal{O}^{\mathcal{S},\Phi}(\phi_i, m_i)$, receiving $\mathcal{S}(\phi_i(sk), m_i)$ in return; in some cases $\phi_i(sk) = sk$, and in other cases $\phi_i(sk)$ will be a different related key. When the adversary produces m^* and σ^*, it is clear this should not be accepted as a forgery if the adversary received a signature of m^* under the original secret key– in other words, m^* and σ^* should not count as a forgery if there is an i such that $m_i = m^*$ and $\phi_i(sk) = sk$. However, a more difficult, and subtle question: should a forgery m^* and σ^* be considered a break if the adversary received a signature of m^* under some different, related secret key? Considering this a valid break for the Φ-RKA adversary makes minimal assumptions about what should be disallowed, and it turns out that such a notion is also achievable.

Definition 5.4 (Φ-RKA Security) *Recall that a cryptographic primitive* P *is a tuple of algorithms,* $\mathsf{P} = (g_1(\cdot), g_m(\cdot), f_1(sk, \cdot), \ldots, f_n(sk, \cdot))$*, where each* f_i *depends on the secret key* sk*, and each* g_i *does not. Say a standard (non-tampering) security game for* P *defines set* S_j *to be the set of* i *such that the adversary receives oracle access to* $\mathcal{O}^{f_i}$ *during phase* j*, possibly with inputs disallowed with* $\mathsf{disallow}_{i,j}(\cdot)$*. To use oracle access to* f_i *in phase* j*, the adversary queries* $\mathcal{O}^{f_i}(x_1, \ldots, x_k)$*, to which the oracle replies with either* $\perp$ *if* $\mathsf{disallow}_{i,j}(x_1, \ldots, x_k, \cdot)$ *is* true*, or* $f_i(sk, x_1, \ldots, x_k)$ *otherwise.*

To move to the Φ-RKA model, the Φ-RKA adversary *participates in the same security game as in the standard model the with some modifications. In phase* j*, if* $i \in S_j$ *so that the adversary received oracle access to* f_i*, the Φ-RKA adversary should now receive oracle access to* $\mathcal{O}^{f_i,\Phi}$ *instead of* $\mathcal{O}^{f_i}$*. When the Φ-RKA adversary makes call* $\mathcal{O}^{f_i,\Phi}(\phi, x_1, \ldots, x_k)$ *to the oracle, the oracle first computes the tampered key* $sk' = \phi(sk)$*. If* $sk' = sk$ and $\mathsf{disallow}_{i,j}(x_1, \ldots, x_k, \cdot)$ *is* true*, the oracle returns* $\perp$*; otherwise the oracles returns* $f_i(sk', x_1, \ldots, x_k)$*.*

The definition of a break of the security of the scheme is unmodified, and should explicitly state all uses of the secret key of the scheme. Additionally, the definition of what it means for P *to be secure remains unchanged. The standard security definition defines* P *to be secure if certain bounds*

apply to the probability of a break for all polynomial time adversaries; we say that P *is* Φ*-RKA secure if the same bounds apply to all polynomial time* Φ*-RKA adversaries.*

As part of this thesis, we will give concrete definitions for Φ-RKA security as it applies to a number of different primitives, including PRFs, wPRFs, IBE, Signatures, SE-CCA, SE-CPA, and PKE-CCA.

Remarks about Φ-RKA Security.

- We note that this model gives the Φ-RKA adversary access to many copies of primitive with the original secret key; for each iteration, the adversary is given access to the primitive evaluated on the original secret key, even though the tampering function ϕ applied in previous iterations might not have been invertible.

- Note that while some queries of a Φ-RKA adversary to an oracle $\mathcal{O}^{f_i,\Phi}$ might be disallowed via $\mathsf{disallow}()$, no such restriction is applied unless $\phi(sk) = sk$;this appears to be a minimal assumption.

- If the standard security definition for a primitive creates a challenge for the adversary, this challenge should be created with the original secret key sk Φ-RKA adversary, and not some tampered sk. Otherwise this definition would immediately disallow Φ that contain ϕ with low entropy output.

- We would like for security in the RKA model to always imply security in a standard security model. To do this, we will assume that ID, the class that only contains the identity function, is always in the class of allowed tampering functions. ID models the adversary's ability to choose not to tamper the secret key of a system, and so ID-RKA security always reduces to the normal security for the primitive.

The definition of a break of security for a Φ-RKA secure P is the same as for a normal secure instance of P. This is particularly interesting when the definition of a valid break depends on prior queries of the adversary. In the case of a Φ-RKA signature scheme, a break is defined when the

adversary produces a valid message, signature pair such that it never received a signature of the message generated with the *original* signing key; a forgery on a message is still considered valid even if the adversary received signatures on that message generated with modified signing keys! This is identical to the definition of a forgery in the normal secure case, where a forgery is valid if the adversary never received a signature on the message generated with the original signing key.

Signature schemes are of particular interest in considering the definition of a break of security in the RKA-model since the breaking condition depends on prior queries of the adversary. Prior work defining Φ-RKA security for signature schemes [GOR11] has instead defined a break as a valid message signature pair such that the adversary never queried for a signature on that message using the identity function, ϕ_{ID}, as the tampering function. This disallows a subset of the forgeries we disallow, since a query with RKD ϕ_{ID} will always generate the signature under the original unmodified secret key; however, this definition immediately requires that the adversary cannot find a $\phi \neq \phi_{\mathsf{ID}}$ such that $\phi(sk) = sk$, or the adversary will have an easy forgery. Thus the definition of [GOR11] disallows Φ-RKA secure signature schemes when Φ fails to be claw-free, while such constructions are possible and meaningful in our definition.

RKA sets. This thesis will address the relative strength of Φ-RKA security for many different cryptographic primitives. In order to achieve this, we will need a notion of what Φ-RKA security is attainable for each primitive.

Definition 5.5 (RKA[P] - the set of Φ for which Φ-RKA secure P exist) *A class of RKD functions Φ is said to be* achievable *for the primitive* P *if there exists a Φ-RKA secure instantiation of* P. *We further define* $\mathbf{RKA}[\mathsf{P}]$ *to be the set of all Φ that are achievable for primitive* P.

Using this set based notation will allow us to easily make comparative statements about Φ-RKA secure primitives. We will give a number of containment results of the form $\mathbf{RKA}[\mathsf{P}_1] \subseteq \mathbf{RKA}[\mathsf{P}_2]$ by providing a construction of a Φ-RKA secure P_2 from any Φ-RKA secure P_1. We will also provide negative results of the form $\mathbf{RKA}[\mathsf{P}_1] \not\subseteq \mathbf{RKA}[\mathsf{P}_2]$ by showing there exists a Φ for which $\Phi \in \mathbf{RKA}[\mathsf{P}_1]$ but $\Phi \notin \mathbf{RKA}[\mathsf{P}_1]$. Both containment and non-containment relationships give implications about the achievability of Φ-RKA security for a given primitive.

We note that since **RKA**[P] is comprised of Φ such that a Φ-RKA secure P exists, **RKA**[P] might change depending on what assumptions we are willing to make, such as whether one-way functions exist. In general when considering **RKA**[P], we will in general make the assumption that a normal (non-tampering) secure instance of P exists, but will need to make no additional assumptions for the relationships between primitives that we provide.

6 Φ-RKA secure PRFs

The first theoretical works addressing RKA security dealt PRFs[BK03, Luc04], and as we will later discover, Φ-RKA PRFs will a strong starting point for building other primitives. We give a full definition of Φ-RKA secure PRFs here since it is used widely in our constructions, and to illustrate some of the points in definition 5.4 of Φ-RKA security.

6.1 Pseudorandom Functions

Since the definition for a Φ-RKA PRF builds on the standard definition for a PRF[GGM86], we will first give the standard definition here.

Definition 6.1 (Pseudorandom Function – PRF) *The following security game is defined for a function family $\mathcal{FF} = (\mathcal{K}, \mathcal{F})$ with algorithms for key generation and evaluation, respectively. For a security parameter k, we assume that an instance of $\mathcal{FF}$ with key $K \xleftarrow{\$} \mathcal{K}(1^k)$ has an efficiently computable domain and range given by $\mathsf{Dom}(\cdot)$ and $\mathsf{Rng}(\cdot)$.*

1. Setup Phase. *The challenger generates a key for the PRF as $K \xleftarrow{\$} \mathcal{K}(1^k)$, and selects a random bit $b \xleftarrow{\$} \{0,1\}$.*

2. Query Phase. *The adversary is allowed to adaptively query for points $x \in \mathsf{Dom}(k)$ to an oracle as $\mathcal{O}^{\mathcal{PRF}}(K, x)$.*

   If $b = 0$ and the adversary has never queried x, the challenger randomly selects a point $y \xleftarrow{\$} \mathsf{Rng}(k)$ to return to the adversary, and saves it as $T[x] = y$; if x has been queried previously, i.e. $T[x]$ is defined, the challenger returns the previous value of $T[x]$.

Otherwise, if $b = 1$ then oracle access to $\mathcal{O}^{\mathcal{PRF}}$ is equivalent to the PRF evaluation function $\mathcal{O}^{\mathcal{F}}$, and the challenger responds to a query x with $\mathcal{F}(K, x)$.

3. Guess. *The adversary must return a guess b' of the hidden bit b.*

For any function family $\mathcal{FF}$, let $\mathbf{Adv}^{\text{prf}}_{\mathcal{FF},A}(k) = \Pr[b = b'] - \frac{1}{2}$. We say that $\mathcal{FF}$ is a Pseudorandom Function (PRF) if $\mathbf{Adv}^{\text{prf}}_{\mathcal{FF},A}(k)$ is negligible in k for all polynomial time adversaries A.

6.2 Φ-RKA PRFs

The definition of a PRF requires that the function indexed by a randomly selected key is indistinguishable from a random function for any polynomial time adversary. In the RKA model the adversary is given access to an oracle that evaluates the PRF using tampered keys, and so will now have access to the functions indexed by multiple keys. As will be formalized shortly, PRF security in the RKA model requires that for any polynomial time adversary, the family of functions is indistinguishable from a family of truly random functions, when accessed at indices generated by a random key and tampered values of that key.

The only changes to the security game of the Φ-RKA PRF from that of a normal PRF are in the query phase, where the adversary receives oracle access to the PRF evaluated using the secret key– this is modified to give the adversary access to the PRF evaluated using tampered secret keys.

Definition 6.2 (Φ-RKA PRF) *The following security game is defined for a function family $\mathcal{FF} = (\mathcal{K}, \mathcal{F})$ with algorithms for key generation and evaluation, respectively. For a security parameter k, we assume that an instance of $\mathcal{FF}$ with key $K \xleftarrow{\$} \mathcal{K}(1^k)$ has an efficiently computable domain and range given by $\mathsf{Dom}(\cdot)$ and $\mathsf{Rng}(\cdot)$, and a compatible class of RKD functions Φ.*

1. Setup Phase. *The challenger generates a key for the PRF as $K \xleftarrow{\$} \mathcal{K}(1^k)$, and selects a random bit $b \xleftarrow{\$} \{0,1\}$.*

2. Query Phase. *The adversary is allowed to adaptively query for points $x \in \mathsf{Dom}(k)$ with a tampering function $\phi \in \Phi$ to as $\mathcal{O}^{\mathcal{PRF},\Phi}(\phi, x)$. Upon receiving an oracle query (x, ϕ), the*

challenger first computes the tampered key $K' \leftarrow \phi(K)$. *If* $b = 0$ *and the adversary has never received a value for* x *and* K', *the challenger randomly selects a point* $y \xleftarrow{\$} \mathsf{Rng}(k)$ *to return to the adversary, and saves it as* $T[K', x] = y$; *if* x *with key* K' *has been queried previously, i.e.* $T[K', x]$ *is defined, the challenger returns the previous value in* $T[K', x]$. *Otherwise, if* $b = 1$ *then the challenger responds to a query* x *with* $\mathcal{F}(K', x)$.

3. Guess. *The adversary must return a guess* b' *of the hidden bit.*

For any function family $\mathcal{FF}$, *let* $\mathbf{Adv}^{\mathrm{prf}}_{\mathcal{FF},A}(k) = \Pr[b = b'] - \frac{1}{2}$. *We say that* $\mathcal{FF}$ *is a Pseudorandom Function (PRF) if* $\mathbf{Adv}^{\mathrm{prf}}_{\mathcal{FF},A}(k)$ *is negligible in* k *for all polynomial time adversaries* A.

We note that in general, PRFs are not Φ-RKA PRFs. Although individual randomly selected functions from the PRF are indistinguishable from random, the structure of functions with related indices might be far from random. For example, a PRF might simply ignore several bits of its key, and so functions with indices differing only in the ignored bits will actually be identical.

An alternative and equivalent definition follows which is based on two separate games, one in which an adversary always receives access to the real function family, and a second one in which it always receives access to a truly random function family. This definition will be used to simplify several proofs.

Definition 6.3 (Alternative Definition for a Φ-RKA PRF) *The following two security games,* PRFReal *and* PRFRand, *are both defined for a function family* $\mathcal{FF} = (\mathcal{K}, \mathcal{F})$ *with algorithms for key generation and evaluation, respectively. For a security parameter* k, *we assume that an instance of* $\mathcal{FF}$ *with key* $K \xleftarrow{\$} \mathcal{K}(1^k)$ *has an efficiently computable domain and range given by* $\mathsf{Dom}(\cdot)$ *and* $\mathsf{Rng}(\cdot)$, *and a compatible class of RKD functions* Φ. *They share a setup phase and a guess phase, but differ in the query phase.*

1. Setup Phase. *The challenger generates a key for the PRF as* $K \xleftarrow{\$} \mathcal{K}(1^k)$.

2a. Query Phase for PRFReal. *The adversary is allowed to adaptively make queries of the form* (x, ϕ) *with* $x \in \mathsf{Dom}(k)$ *and* $\phi \in \Phi$. *The challenger first computes* $K' \leftarrow \phi(K)$ *and returns*

$\mathcal{F}(K', x)$ *to the adversary.*

2b. Query Phase for PRFRand. *The adversary is allowed to adaptively make queries of the form* (x, ϕ) *with* $x \in \mathsf{Dom}(k)$ *and* $\phi \in \Phi$. *The challenger first computes* $K' \leftarrow \phi(K)$. *If* $T[K', x]$ *is undefined, the challenger sets* $T[K', x] \xleftarrow{\$} \mathsf{Rng}(k)$. *The challenger sends* $T[K', x]$ *to the adversary.*

3. Guess. *The adversary must return a guess* b' *of whether it is playing game* PRFRand *or game* PRFReal.

Standard arguments imply that $\mathbf{Adv}^{\mathrm{prf}}_{\mathcal{FF},A,\Phi}(k) = \Pr[\mathrm{PRFRand}^A \Rightarrow 1] - \Pr[\mathrm{PRFReal}^A \Rightarrow 1]$.

7 Identity-Key Fingerprints and Identity Collision Resistance

In our constructions, we will want to reduce the Φ-RKA security of a primitive to the normal, non-tampering security of the same primitive. Such a reduction is achieved by construction of an adversary A that can break a normal secure primitive by using any Φ-RKA adversary B that can break the Φ-RKA security of the primitive. Our constructions have A answer oracle queries of the Φ-RKA adversary B in the following way:

1. When B's submitted ϕ leaves the secret key unchanged, A should answer B's queries using its own oracle access to a non-tampering oracle. Since the definition of a break in the Φ-RKA security of any primitive is a valid break for the original untampered secret key, this will guarantee that a successful break for B will also be a successful break for A.

2. B's queries to different modified keys should give it no additional information about the original secret key; A should be able simulate answers to oracle queries to modified keys.

To achieve the second point, we design our Φ-RKA primitives so that A can simulate answers to B's oracle queries by randomly generating instances of a normal (non-tampering) version of the primitive.

The first point adds a greater technical challenge: when answering B's query with related-key deriving function ϕ, how will A know when the secret key is unchanged and to answer with its own

oracle access? The A only has access to oracle queries to the primitive, but not the secret key of the primitive. As an example of this difficulty, imagine A is an adversary against a PRF. A is given oracle access to either the PRF or a family of truly random functions evaluated with the hidden sk. Though A query this oracle at many points in the domain of the PRF, this might be little help in telling whether $\phi(sk) = sk$. (Consider a PRF that ignores the last bit of the secret key, and a Φ that allows the adversary to set the last bit of the secret key to either 0 or 1.)

7.1 Identity-Key Fingerprint

To solve this and other technical issues associated with dropping the claw-free assumption, we will use a new tool, called an Identity-key Fingerprint (IDFP), to determine when the secret key of a PRF has changed. To accomplish this, an IDFP for a function family generates a vector of points in the domain, called the fingerprint; except with negligible probability, a function evaluated with a random secret key key and the function evaluated with a tampered key will differ at at least one of the points in the fingerprint. Like our definition of RKA security, an IDFP will be parameterized by Φ, the class of allowed tampering functions.

Definition 7.1 (Identity-key Fingerprint – IDFP) *Let $\mathcal{FF} = (\mathcal{K}, \mathcal{F})$ be any function family, with key-generation and evaluation algorithms respectively, an associated security parameter k, and with an efficiently computable domain given by* $\mathsf{Dom}(\cdot)$.

A fingerprint *is a vector of points in the domain of $\mathcal{FF}$ with length $v(k)$ that is polynomial in k. An* identity key fingerprint *(IDFP) is an algorithm* IKfp *that produces a fingerprint for $\mathcal{FF}$ as* $\mathbf{w} \xleftarrow{\$} \mathsf{IKfp}(1^k)$. *The following game gives the security requirements for for $\mathcal{FF}$, and is parameterized by a compatible class of RKD functions Φ for $\mathcal{FF}$.*

1. Setup Phase. *The challenger generates a key for $\mathcal{FF}$ as $K \xleftarrow{\$} \mathcal{K}(1^k)$. The challenger also generates the fingerprint for $\mathcal{FF}$ as $\mathbf{w} \xleftarrow{\$} \mathsf{IKfp}(1^k)$, and sends the fingerprint to the adversary. Finally, the variable* WIN *is set to have value* false.

2. Query Phase. *The adversary is allowed to make adaptive queries to an IDFP oracle for $\phi \in \Phi$ as $\mathcal{O}^{\mathrm{idfp},\Phi}(\phi)$. The challenger answers this oracle query by first computing the tampered key*

$K' \leftarrow \phi(K)$. *If* $K \neq K'$, *and* $\mathcal{F}(K, \mathbf{w}[i]) = \mathcal{F}(K', \mathbf{w}[i])$ *for all* $i \in [|\mathbf{w}|]$, $\text{WIN} \leftarrow \mathsf{true}$. *In any event, the challenger sends the adversary* $\mathcal{F}K', \mathbf{w}[i])$ *for all* $i \in [|\mathbf{w}|]$.

For any adversary A that is polynomial time in k, we define $\mathbf{Adv}^{\text{idfp}}_{\mathcal{FF},A,\Phi}(k) = \Pr[\text{WIN} = \mathsf{true}]$. *We say* $\mathcal{FF}$ *is* Φ*-IDFP secure if this advantage function is negligible in k for all such polynomial time adversaries A.*

The notion of IDFP can be seen as a relaxation of the key fingerprint defined by Bellare and Cash[BC10]. The key fingerprint of [BC10] for a PRF allows statistical disambiguation of any pair of keys: except with negligible probability, any two keys will differ at some point in the key fingerprint. They showed that the Naor-Reingold PRF (NR-PRF) had such a key fingerprint, but in general, it does not seem common. Interestingly, their own Φ-RKA PRFs, which build on NR-PRFs, are not known to have a key fingerprint. The IDFP has the weaker requirement of computational disambiguation of the original key from other keys: given a randomly generated original key, it must be computationally hard to find a second key that agrees at all points in the IDFP.

The IDFP notion is easier to satisfy than that of the key fingerprint, and will still be sufficient for our constructions. We are not able to give a general proof that a Φ-RKA secure PRF also has is Φ-IDFP secure, and so will have to make this an assumption in some of our constructions. We are, however, able to show that for claw-free Φ, any Φ-RKA secure PRF with large enough range is Φ-IDFP secure, using *any* point in the domain functioning as the fingerprint. This adds to the evidence that assuming Φ-IDFP security for Φ-RKA secure PRF is reasonable.

Proposition 7.2 (Φ-RKA secure PRFs have Φ-IDFP for claw-free Φ) *Let* Φ *be a claw-free class of RKD functions, and let* $\mathcal{FF}$ *be a* Φ*-RKA secure PRF with efficiently computable domain domain* $\mathsf{Dom}(\cdot)$ *and super-polynomial size range* $\mathsf{Rng}(\cdot)$. *Then* $\mathcal{FF}$ *is* Φ*-IDFP secure, using IDFP algorithm* $\mathsf{IKfp}(1^k)$ *that returns the 1-vector comprised of any fixed element in* $\mathsf{Dom}(\cdot)$.

Proof:

Define the IDFP algorithm $\mathsf{IKfp}(1^k)$ to return the lexicographically first element in $\mathsf{Dom}(1^k)$.

Using any adversary A against the Φ-IDFP security of PRF $\mathcal{FF}$, we construct an adversary B against the Φ-RKA security of $\mathcal{FF}$ with

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

where $q(k)$ is the number of oracle queries made by A to $\mathcal{O}^{\text{idfp}}$.

B receives oracle access to $\mathcal{O}^{\mathcal{PRF},\oplus}$, which answers its queries $\mathcal{O}^{\mathcal{PRF},\Phi}(\phi, x)$ with using either a PRF or a truly random function. Recall that ϕ_{ID} is the identity function and the RKA-model assumes that $\phi_{\mathsf{ID}} \in \Phi$. B generates the fingerprint as $\mathbf{w} \leftarrow \mathsf{IKfp}(1^k)$, and then computes $\mathbf{y}[1] \leftarrow \mathcal{O}^{\mathcal{PRF},\Phi}(\phi_{\mathsf{ID}}, \mathbf{w}[1])$, which is the value returned by the oracle, evaluated using the original secret and the single value in the fingerprint. Finally, B initializes a set T to empty.

For the reduction, B provides the IDFP adversary A with input of fingerprint $\mathbf{w}$. When A makes a query to the IDFP oracle as $\mathcal{O}^{\text{idfp},\Phi}(\phi)$, B queries its Φ-RKA PRF oracle with $\mathbf{z} = \mathcal{O}^{\mathcal{PRF}}(\phi, \mathbf{w}[1])$, and returns the value $\mathbf{z}$ to A. Additionally, if $\mathbf{z} = \mathbf{y}$ then B sets a variable $\textsc{Win} \leftarrow \mathsf{true}$. When A halts, B returns 1 if $\textsc{Win} = \mathsf{true}$, and 0 otherwise.

For the analysis, we consider the cases that B has oracle access to the real PRF in PRFReal or a truly random function in PRFRand separately.

First lets consider the game PRFReal, where B's oracle queries to $\mathcal{O}^{\mathcal{PRF},\Phi}(\phi, \mathbf{w}[1])$ are answered by the real PRF, $\mathbf{z} = \mathcal{F}(\phi(K), \mathbf{w}[1])$. In this case, A is getting the correct distribution for the PRF and so is playing the exact game defining Φ-IDFP security; B will set $\textsc{Win} = \mathsf{true}$ when A has found ϕ such that $\mathcal{F}(\phi, \mathbf{w}[1]) = \mathcal{F}(\phi_{\mathsf{ID}}, \mathbf{w}[1])$ and has broken the security of the IDFP. Thus B will return 1 with probability given by $\mathbf{Adv}^{\text{idfp}}_{\mathcal{FF},A,\Phi}(k)$.

In game PRFRand, B's oracle queries to $\mathcal{O}^{\mathcal{PRF},\Phi}(\phi, \mathbf{w}[1])$ are instead answered with a truly random function. Let $q(k)$ be the number of queries A makes to $\mathcal{O}^{\text{idfp},\Phi}$. When A queries a $\phi_i \neq \phi_{\mathsf{ID}}$, the tampered key must be different from the original key, since $\phi_i(K) \neq K$ by the claw-free assumption and the fact that $\phi_{\mathsf{ID}}(K) = K$. Thus in this case $\mathbf{z} = \mathcal{O}^{\mathcal{PRF}}(\phi_i, \mathbf{w}[1])$ is independent from $\mathbf{y} = \mathcal{O}^{\mathcal{PRF}}(\phi_{\mathsf{ID}}, \mathbf{w}[1])$, and so for each query B sets $\textsc{Win}$ to true with probability at most

$\frac{1)}{|\mathsf{Rng}(1^k)|}$. B never sets WIN to **true** when A queries with ϕ_{ID}, since this only happens when the key is modified. Thus in this case, by a union bound B returns 1 with probability at most $\frac{q(k)}{|\mathsf{Rng}(1^k)|}$.

Then

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

where $q(\cdot)$ is the number of oracle queries made by A. Using definition 6.3 for $\mathbf{Adv}^{\text{prf-rka}}_{\mathcal{FF},B,\Phi}(k)$ gives the desired result.

∎

Using Proposition 7.2, the Φ-RKA PRF construction of [BC10] can be used to generate an IDFP, since the Φ they use is claw-free. Interestingly, though they use key fingerprints to create their Φ-RKA secure PRF, no key fingerprint is known for their construction.

Unfortunately, this construction of IDFP only works for claw-free Φ, while we will need IDFP for classes Φ without the claw-free assumption. For some of our constructions of higher level Φ-RKA secure primitives, we will assume the Φ-IDFP security of given Φ-RKA PRFs, even when Φ is not claw-free. In practice, a vector over a distinct points in the domain should be a suitable fingerprint and assuming the existence of IDFPs seems to be a reasonable even when Φ is not claw-free.

7.2 Φ-RKA Secure Pseudorandom Generators

To drop the claw-free assumption on Φ in our constructions, we introduce and use a new security notion called Identity Collision Resistance (ICR); this notion is parameterized by Φ, and a primitive possessing this property is said to be Φ-ICR secure.

In particular, we will build psuedorandom generators (PRGs) that are both Φ-RKA and Φ-ICR secure, using any Φ-RKA and Φ-IDFP secure PRF. A pseudorandom generator (PRG)[BM84] is a tuple of algorithms $\mathcal{PRG} = (\mathcal{K}, \mathcal{G})$ for key generation and evaluation of the PRG respectively; $\mathcal{PRG}$ also has an associated function $r(\cdot)$ that gives the output length of the PRG, which must be longer

than the length of the secret key. The secret key of a PRG is called a seed; when the evaluation algorithm is run on a randomly generated seed, it outputs a value that is indistinguishable from random, even though $r(\cdot)$ ensures that the output is longer than the secret key and so is lacking full entropy.

Definition 7.3 (Psuedorandom Generator – PRG) *A* psuedorandom generator (PRG) *is a tuple of algorithms $\mathcal{PRG} = (\mathcal{K}, \mathcal{G})$. For a security parameter k, the key generation algorithm $\mathcal{K}(1^k)$ generates a secret key sk called a* seed, *and $\mathcal{G}(sk)$ returns a string of length $r(k)$, where $r(k) > |sk|$, ensuring that $\mathcal{PRG}$ is length expanding. Further, the output of $\mathcal{G}(sk)$ should be indistinguishable from a random string for any polynomial time adversary, as is formalized in the following security game for an adversary A:*

1. Setup Phase. *The challenger generates a seed as $K \xleftarrow{\$} \mathcal{K}(1^k)$ and generates a random bit $b \xleftarrow{\$} \{0,1\}$.*

2. Query Phase. *The adversary A is given oracle access to $\mathcal{O}^{\mathcal{PRG}}$, which takes no input parameters. If $b = 0$, and A has already queried $\mathcal{O}^{\mathcal{PRG}}$, the previously returned value is returned again; if this is A's first query, the oracle returns a value from the uniform distribution of length $r(k)$, i.e. a uniformly selected element of $\{0,1\}^{r(k)}$. Otherwise when $b = 1$, the oracle returns $\mathcal{G}(K)$ to A.*

3. Guess. *A must return a guess b' of the hidden bit b.*

We define $\mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},A}(k) = \Pr[b = b'] - \frac{1}{2}$, and say that $\mathcal{PRG}$ is a pseudorandom generator (PRG) if $\mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},A}(k)$ is negligible in k for all polynomial time adversaries A.

Φ-RKA Secure PRGs.. Extending the definition of a PRG to the RKA model, the adversary can query the PRG evaluation function along with a related-key deriving function ϕ. In the modified security game, the adversary can obtain output of the PRG (or random function) with not only the original seed, but also seeds modified with ϕ. To the adversary, each distinct seed of the PRG should yield what appears to be an independent random value.

We note that unlike the standard definition for a PRG, we will not require a Φ-RKA PRG to be length expanding: the benefit of the Φ-RKA PRG is that its output appears random even for related keys, and not length extension. However, one can easily create a length-extending Φ-RKA PRG by applying a standard PRG to the output of a Φ-RKA PRG[Luc04].

Definition 7.4 (Φ-RKA PRG) *The following security game for an adversary A is defined for a $\mathcal{PRG}$ and a RKD specification Φ that is compatible with $\mathcal{PRG} = (\mathcal{K}, \mathcal{G})$, and a security parameter k.*

1. Setup Phase. *The challenger generates a seed for the PRG as $K \xleftarrow{\$} \mathcal{K}(1^k)$, and selects a random bit $b \xleftarrow{\$} \{0,1\}$.*

2. Query Phase. *A is allowed to adaptively query tampering functions ϕ in the family of allowed tampering functions Φ. Upon receiving a query ϕ, the challenger first computes the tampered seed $K' \leftarrow \phi(K)$. If $b = 0$ and the adversary has never received a value for K' , the challenger randomly selects a point in the range of $\mathcal{G}$ to return to the adversary; if seed K' has been queried previously, the challenger returns the previous value. Otherwise, if $b = 1$ then the challenger responds to a query x with $\mathcal{G}(K')$.*

3. Guess. *The adversary must return a guess b' of the hidden bit.*

For a primitive $\mathcal{PRG}$ as defined above, we define $\mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},A,\Phi}(k) = \Pr[b = b'] - \frac{1}{2}$, and say that $\mathcal{PRG}$ is a Φ-RKA secure PRG if $\mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},A,\Phi}(k)$ is negligible in k for all polynomial time adversaries A.

As with Φ-RKA secure PRFs, it will help our proofs to have an alternative definition of Φ-RKA secure PRGs that separately considers the cases where the adversary has oracle access to the PRG and when they have oracle access to a random function instead.

Definition 7.5 (Alternative Definition for a Φ-RKA PRG) *The following two security games,* PRGReal *and* PRGRand, *are both defined for $\mathcal{PRG} = (\mathcal{K}, \mathcal{G})$ and a RKD specification Φ that is compatible with $\mathcal{PRG}$, a security parameter k, and an adversary A. We assume that an instance*

of $\mathcal{PRG}$ with key $K \xleftarrow{\$} \mathcal{K}(1^k)$ has an efficiently computable domain and range given by $\mathsf{Dom}(\cdot)$ *and* $\mathsf{Rng}(\cdot)$.

These two games share a setup phase and a guess phase, but differ in the query phase.

1. Setup Phase. *The challenger generates a seed for the PRG as $K \xleftarrow{\$} \mathcal{K}(1^k)$.*

2a. Query Phase for PRGReal. *A is allowed to adaptively query tampering functions $\mathcal{O}^{\mathcal{PRG},\Phi}(\phi)$, to which the challenger replies with $\mathcal{F}(K')$ to the adversary.*

2b. Query Phase for PRGRand. *A is allowed to adaptively query tampering functions $\mathcal{O}^{\mathcal{PRG},\Phi}(\phi)$. The challenger first computes $K' \leftarrow \phi(K)$. If $T[K']$ is undefined, the adversary sets $T[K'] \xleftarrow{\$} \mathsf{Rng}(k)$. The challenger sends $T[K']$ to the adversary.*

3. Guess. *The adversary must return a guess b' of whether it is playing game* PRGRand *or game* PRGReal.

Standard arguments imply that $\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{FF},A,\Phi}(k) = \Pr[\mathrm{PRFRand}^A \Rightarrow 1] - \Pr[\mathrm{PRFReal}^A \Rightarrow 1]$.

7.3 Identity Collision Resistance

For our constructions, we will define and use a new weak form of collision resistance for PRGs. Roughly, for a Φ-ICR secure PRG, it should be difficult for an adversary to find a $\phi \in \Phi$ that modifies the hidden seed but that fails to change the output of the PRG, with $\phi(K) \neq K$ yet $\mathcal{G}(\phi(K)) = \mathcal{G}(K)$. This definition is formalized below.

Definition 7.6 (Φ-ICR secure PRG) *The following security game for an adversary C is defined for a $\mathcal{PRG} = (\mathcal{P}, \mathcal{K})$ with a compatible RKD specification Φ, and a security parameter k.*

1. Setup Phase. *The challenger generates a seed for the PRG as $K \xleftarrow{\$} \mathcal{K}(1^k)$, computes $T \leftarrow \mathcal{G}(K)$, and initializes the variable* WIN *to* false.

2. Query Phase. *The adversary C adaptively queries $\mathcal{O}^{\mathrm{icr},\Phi}$ as $\mathcal{O}^{\mathrm{icr},\Phi}(\phi)$. The challenger answers these queries by first computing the tampered seed $K' \leftarrow \phi(K)$, and then evaluates the $\mathcal{PRG}$ on the tampered seed as $S \leftarrow \mathcal{G}(K')$. If $K' \neq K$ and $S = T$, the adversary has found a*

collision, and $\text{WIN} \leftarrow \mathsf{true}$. *To answer the query, the challenger returns* S *to the adversary* C.

Let $\mathbf{Adv}^{\text{icr}}_{\mathcal{PRG},C,\Phi}(k)$ *equal* $\Pr[\text{WIN} = \mathsf{true}]$ *when the game has input* 1^k. *We say* $\mathcal{PRG}$ *is* Φ*-ICR secure if this advantage function is negligible.*

Does Φ-RKA security imply Φ-ICR security?.

ICR security is only violated when two distinct seeds map to the same value in the range of a PRG. We note that an adversary given oracle access to a truly random function will only very rarely be able to find two distinct seeds that map to the same value; since a Φ-RKA secure PRG is indistinguishable from a random function (when evaluated at points given by tampered seeds), it would at first appear that Φ-RKA security implies Φ-ICR security for a PRG, and that adversary that breaks the Φ-ICR security of the PRG could be used to build a distinguisher of the Φ-RKA PRG and a truly random function. Unfortunately – and unintuitively! – this is not the case.

Let's begin to see where this intuition breaks down. To build a distinguisher of the Φ-RKA PRG and a truly random function, we would try to get the Φ-ICR adversary to find a collision of distinct points; if the Φ-ICR adversary finds a collision with its oracle access, the distinguisher guess it has access to the PRG.

This intuition breaks down because the distinguisher only sees the output of the oracle, and so does not actually see the secret key– the distinguisher does not know whether a repeated output was generated by the same secret key. Further, it assumes that ϕ modifies the secret key in the same way in both the real and random games. As we will see shortly, it is possible that a repeated output is caused by two different seeds in the real game, and by the same seed in the random game. This will allow the adversary to break the Φ-ICR security, while still giving indistinguishable views of the real and random games for the Φ-RKA PRG distinguisher!

We formalize these ideas to prove that Φ-RKA security does not imply Φ-ICR security.

Proposition 7.7 *[Φ-RKA Security does not imply Φ-ICR Security]*

Suppose there exists a normal-secure PRG $\overline{\mathcal{PRG}} = (\overline{\mathcal{K}}, \overline{\mathcal{G}})$ *with security parameter* k, *output length given by* $r(k) = \omega(k)$, *and with range super-polynomial in* k. *Then there exists a PRG*

$\mathcal{PRG} = (\mathcal{K}, \mathcal{G})$ with the same $r(\cdot)$, and a compatible class of RKD Φ, such that $\mathcal{PRG}$ is Φ-RKA secure but $\mathcal{PRG}$ is not Φ-ICR secure.

Proof: Let $\ell(k)$ be the length of the seed returned by $\overline{\mathcal{K}}(1^k)$.

We construct the new key-generation algorithm $\mathcal{K}$ for $\mathcal{PRG}$ to pick a random bit $c \xleftarrow{\$} \{0,1\}$, and returns $c \,\|\, \overline{\mathcal{K}}(1^k)$, as $\mathcal{K}(1^k) \xleftarrow{\$} \{c \xleftarrow{\$} \{0,1\}, \overline{\mathcal{K}}(1^k) : c \,\|\, \overline{\mathcal{K}}(1^k)\}$. The new evaluation algorithm $\mathcal{G}(K)$ parses K into $c \,\|\, \overline{K}$, where c is a bit, and $|\overline{K}| = \ell(k)$, and then returns $\overline{\mathcal{G}}(\overline{K})$; more formally, $\mathcal{G}(K) = \overline{\mathcal{G}}(K[2, |K|])$. Note that this new evaluation algorithm $\mathcal{G}$ ignores the first bit of the seed, and so $\mathcal{G}$ will agree on any two seeds differing only in the first bit – thus any ϕ that modifies the first bit will lead to a win for an adversary in the Φ-ICR security game.

We define Φ such that our constructed $\mathcal{PRG}$ is Φ-RKA secure but *not* Φ-ICR secure. Briefly, our Φ will be created so that $\phi \in \Phi$ will only modify input in the real game PRGReal and not in the random game PRGRand; with this, output collisions will occur in both cases, but any output collisions in the random case are trivial since the seed will not be modified here. Let K^- denote K with its first bit flipped. For a $R \in \{0,1\}^{r(k)}$ and $K \in \{0,1\}^{\ell(k)+1}$, we define $\phi_R(K)$ to be K if $\mathcal{G}(K) \neq R$, and have its first bit flipped as K^- otherwise:

$$\phi_R(K) = \begin{cases} \mathrm{K} & \text{if } \mathcal{G}(K) \neq R \\ - & \text{otherwise} \end{cases}$$

We let $\Phi = \mathsf{ID} \cup \bigcup_{R \in \{0,1\}^{r(k)}} \{\phi_R\}$ be the collection of ϕ_R for all $R \in \{0,1\}^{r(k)}$, along with ϕ_{ID} the identity function.

Our constructed $\mathcal{PRG}$ fails to be Φ-ICR secure: an adversary A that makes a query ϕ_{ID} receives $X = \mathcal{G}(K)$ in response, and can then make query ϕ_X. By definition of our ϕ, since $\mathcal{G}(K) = X$, $\phi_X(K)$ will flip the first bit of the seed and give a non-trivial collision, setting $\textsc{Win} \leftarrow \mathsf{true}$, and giving the adversary an advantage of 1.

Next, we must show that $\mathcal{PRG}$ is Φ-RKA secure under the assumption that $\overline{\mathcal{PRG}}$ was normal (i.e. ID-RKA) secure. To do this, we will construct an adversary B against the normal security of

$\overline{\mathcal{PRG}}$ from any adversary A against the Φ-RKA security of $\mathcal{PRG}$.

B against the normal security of $\overline{\mathcal{PRG}}$ receives either oracle access to $\overline{\mathcal{PRG}}$ in game PRGReal, or oracle access to a random function in game PRGRand. We let B compute $X \leftarrow \mathcal{O}^{oPRG}(\,)$. But we note that in A's game PRGReal, all of A's queries should be answered by a single value generated by $\overline{\mathcal{PRG}}$ evaluated on a random seed; similarly, in A's game PRGRand, all of A's queries should be answered by a single random value in the range of $\overline{\mathcal{PRG}}$. Thus, B can answer X to all of A's queries $\mathcal{O}^{\mathcal{PRG},\Phi}(\phi)$; this correctly simulate A's game PRGReal when B is in game PRGReal, and correctly simulate A's game PRGRand when B is in game PRGRand. When A returns its guess of hidden bit b', B also outputs b'.

Thus, $\mathbf{Adv}^{\text{prg}}_{\overline{\mathcal{PRG}},B,\mathsf{ID}}(k) = \Pr[\text{PRFRand}^B \Rightarrow 1] - \Pr[\text{PRFReal}^B \Rightarrow 1] = \Pr[\text{PRFRand}^A \Rightarrow 1] - \Pr[\text{PRFReal}^A \Rightarrow 1] = \mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},A,\Phi}(k)$, as desired.

∎

Building Φ-ICR PRGs. As shown above, not all Φ-RKA secure PRGs are Φ-ICR secure, but we will need PRGs with both properties for our constructions. To do this, we will build a PRG that is both Φ-ICR and Φ-RKA secure from any Φ-RKA PRF that is Φ-IDFP secure, i.e., a Φ-RKA secure PRF with a key fingerprint for the identity function.

Proposition 7.8 *Let $\mathcal{FF} = (\mathcal{K}, \mathcal{F})$ be a Φ-RKA PRF with output length $\ell(k)$ for security parameter k. Let IKfp be a Φ-IDFP secure identity key fingerprint function for $\mathcal{FF}$ with vector length $v(k)$.*

Define PRG $\mathcal{PRG} = (\overline{\mathcal{K}}, \mathcal{G})$, with associated output length given by $r(\cdot)$, as follows:

- *$\overline{\mathcal{K}}$ on input 1^k, compute fingerprint $\mathbf{w} \xleftarrow{\$} \mathsf{IKfp}(1^k)$ and and the secret key for a PRF $K \xleftarrow{\$} \mathcal{K}(1^k)$, and return $K = (\mathbf{w}, K)$*
- *$\mathcal{G}(\mathbf{w}, K) = \mathcal{F}(K, \mathbf{w}[1]) \,\|\, \cdots \,\|\, \mathcal{F}(K, \mathbf{w}[|\mathbf{w}|])$*
- *The length of the output of the PRG is given as $r(k) = l(k) \cdot v(k)$*

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

Proof: Let A_1 be any adversary against the Φ-ICR security of $\mathcal{PRG}$, and let A_2 be any adversary against the Φ-RKA security of $\mathcal{PRG}$. We will use to build an adversary B_1 against the Φ-IDFP security of $\mathcal{FF}$, and an adversary B_2 against the Φ-RKA security of $\mathcal{FF}$.

First, we will construct B_1, an adversary against the Φ-IDFP security of $\mathcal{FF}$, which receives fingerprint $\mathbf{w}$ for $\mathcal{FF}$, using A_1, the adversary against the Φ-ICR security of $\mathcal{PRG}$. A_1 makes oracle queries of the form $\mathcal{O}^{\mathcal{PRG},\Phi}(\phi)$. To answer this, B_1 first queries its own oracle $\mathcal{O}^{\text{idfp},\Phi}(\phi, \mathbf{w}[i])$ for all $i \in [|\mathbf{w}|]$; B_1 receives $\mathcal{F}(\phi(K), \mathbf{w}[i])$ for each i, and simply concatenates all these to form and return the proper response to A_1's queries. A_1 is said to break the Φ-ICR security if it queries a ϕ^* such that $\phi^*(K) \neq K$ yet the concatenation of all $\mathcal{F}(\phi^*(K), \mathbf{w}[i])$ is the same as the concatenation of all $\mathcal{F}(K, \mathbf{w}[i])$. In this case, B_1 wins too when it queries its oracle with ϕ^* – here, the key will be modified yet no part of the fingerprint will. Thus

$$\mathbf{Adv}^{\text{idfp}}_{\mathcal{FF},B_1,\Phi}(k) \geq \mathbf{Adv}^{\text{icr}}_{\mathcal{PRG},A_1,\Phi}(k).$$

Next we construct B_2, the adversary against the Φ-RKA security of $\mathcal{FF}$, from any A_2 against the Φ-RKA security of $\mathcal{PRG}$. When A_2 submits queries $\mathcal{O}^{\mathcal{PRG},\Phi}(\phi)$, B_2 answers by first querying its own oracle $\mathcal{O}^{\text{idfp},\Phi}(\phi, \mathbf{w}[i])$ for all $i \in [|\mathbf{w}|]$, and concatenating each of these; note that when B_2 is playing PRFReal, this will simulate PRGReal for A_2, and when B_2 is playing PRFRand this will simulate PRGRand for A_2. Thus when A_2 guesses b', B_2 copies this guess, and so B_2 will correctly guess b exactly when A_2 does. Thus

$$\mathbf{Adv}^{\text{prf}}_{\mathcal{PRF},B_2,\Phi}(k) \geq \mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},A_2,\Phi}(k).$$

∎

This construction also shows how to build a Φ-RKA PRG from any Φ-RKA PRF, though if you don't need the Φ-ICR property it is simpler just to apply the PRF to a constant input.

Corollary 7.9 *Since any Φ-RKA PRF can be used to build a Φ-RKA PRG as above,* $\mathbf{RKA}[\mathsf{PRF}] \subseteq \mathbf{RKA}[\mathsf{PRG}]$.

8 Definitions of Φ-RKA Secure Primitives

Though Definition 5.4 gives a general transformation from the standard security definition of a primitive to the Φ-RKA security definition, there are a number of subtleties in the transformation. Here, we will explicitly give definitions for a number of additional Φ-RKA secure primitives: signature schemes, CCA public key encryption, weak PRF, and symmetric key encryption. The original security definitions for these primitives can be obtained by setting $\Phi = \mathsf{ID}$.

Signatures.

Most of the primitives we will consider have security definitions that are based on indistinguishability; signature schemes are the one primitive that we consider that defines a break based on a produced value of the adversary that depends on the secret and public keys of the scheme.

Since there are subtleties in the definition of a Φ-RKA secure signature scheme, we will first give the standard definition of a signature scheme.

Definition 8.1 (Signature Scheme – [DH76]) *A signature scheme $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is a tuple of algorithms, for key generation, signing, and verification algorithms respectively, with an associated security parameter k, and efficiently computable message space $\mathcal{M}(k)$.*

The key-generation algorithm generates a verification key vk and a signing key sk as $(vk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$. The signing algorithm $\mathcal{S}$ is used to sign messages $m \in \mathcal{M}(k)$ as $\mathcal{S}(sk, m)$. The verification algorithm $\mathcal{V}$ takes parameters of a verification key, a message, and a signature as $\mathcal{V}(vk, m, \sigma)$, and returns either true *or* false *depending on whether the signature is valid or invalid respectively.*

For the correctness of a scheme, if keys are generated properly as $(vk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$, the signature $\sigma \xleftarrow{\$} \mathcal{S}(sk, m)$, should always verify, giving $\mathcal{V}(vk, m, \sigma) =$ true.

We define the security of the scheme via the following game for an adversary A.

1. Setup Phase. *The challenger generates verification and signing keys as $(vk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$. The challenger also initializes the set of disallowed messages for forgery as $M \leftarrow \emptyset$. Finally, the challenger gives the verification key vk to A.*

2. Query Phase. *A adaptively asks for signatures of messages $m \in \mathcal{M}$ to a signing oracle as*

$\mathcal{O}^{\mathcal{S}}(m)$. *When the challenger receives* m, *it adds* m *to the set of disallowed messages for forgery as* $M \leftarrow M \cup \{m\}$, *and finally returns* $\sigma \xleftarrow{\$} \mathcal{S}(sk, m)$ *to* A.

3. Forge. *The adversary halts and outputs a message and signature pair* (m^*, σ^*), *with* $m^* \in \mathcal{M}$.

We define $\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A}(k) = \Pr[m^* \notin M \wedge \mathcal{V}(vk, m, \sigma) = 1]$; *we say* $\mathcal{DS}$ *is a secure signature scheme if* $\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A}(k)$ *is negligible in* k *for all adversaries* A *that run in time polynomial in* k.

To adapt this definition to the RKA model, the security game is modified so that the adversary receives access to a signing oracle that works on tampered keys. Now, A should be able to make adaptive queries $\mathcal{O}^{\mathcal{S},\Phi}(\phi, m)$ and receive $\mathcal{S}(\phi(sk), m)$ in return.

Moving the definition of a forgery from the standard model to the RKA model causes more delicate issues to arise. In the standard definition, the adversary's forgery must be on a previously unsigned message. Definition 5.4 helps us convert this to the RKA model, where the winning condition should not depend on modified secret keys. Here, a Φ-RKA adversary that outputs (m^*, σ^*) should be said to have a valid forgery if it never received a signature on the message m^* generated with the *original* secret key. This appears to be a minimal requirement for the disallowed set of messages. Prior work [GOR11] defining Φ-RKA secure signature schemes instead called a forgery valid if the adversary never made a signature query with the identity function as $\mathcal{O}^{\mathcal{S},\Phi}(\phi_{\mathsf{ID}}, m^*)$; our definition disallows only a subset of the messages that [GOR11] disallows, and makes sense even when Φ fails to be claw-free.

Definition 8.2 (Φ-RKA Signature Scheme) *A* Φ*-RKA secure signature scheme* $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ *is a tuple of algorithms, for key generation, signing, and verification algorithms respectively, with an associated security parameter* k, *and efficiently computable message space* $\mathcal{M}(k)$.

The algorithms of a Φ*-RKA secure signature scheme satisfy the same structure and correctness properties as a standard signature scheme, given in Definition 8.1.*

The security game is parameterized by Φ *that is compatible with* $\mathcal{DS}$ *is defined for an adversary* A.

1. Setup Phase. *The challenger generates and verification and signing keys as* $(vk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$.

The challenger also initializes the set of disallowed messages for forgery as $M \leftarrow \emptyset$. *Finally, the challenger gives the parameters and verification key* (vk) *to adversary* A.

2. Query Phase. *A is allowed to adaptively make oracle queries for signatures as* $\mathcal{O}^{\mathcal{S},\Phi}(m,\phi)$ *with* $m \in \mathcal{M}, \phi \in \Phi$. *To answer, the challenger first generates the modified signing key as* $sk' \leftarrow \phi(sk)$. *If* $sk' = sk$, *the challenger adds the message to the set of disallowed messages for the forgery as* $M \leftarrow M \cup \{m\}$. *Finally, the challenger answers the query by returning* $\sigma \xleftarrow{\$} \mathcal{S}(sk', m)$ *to* A.

3. Forge. *The adversary halts and outputs a message and signature pair* (m^*, σ^*).

We define $\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A,\Phi}(k) = \Pr[m^* \notin M \wedge \mathcal{V}(\pi, vk, m, \sigma) = 1]$, *and say* $\mathcal{DS}$ *is a* Φ*-RKA secure signature scheme if* $\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A,\Phi}(k)$ *is negligible in* k *for all adversaries* A *that run in time polynomial in* k.

Public Key Encryption.

Public key encryption (PKE), as defined in [Sha85], with semantic security, as defined in [GM82], gives a public key that is sufficient to run encryption scheme without compromising a private secret key used only for decryption. Since in this case the secret key is used only for decryption, RKAs make sense to consider when the adversary is given access to a decryption oracle, as is the case for a chosen-ciphertext security definition. In the normal PKE security game, the decryption oracle disallows queries for the challenge ciphertext; to adapt to the RKA model, the decryption oracle will refuse to decrypt when the ciphertext it is given matches the challenge one *and* the tampered key equals the real one.

Again, we give only a definition for a Φ-RKA PKE scheme and not for one with standard security, but one can obtain the standard definition by setting $\Phi = \mathsf{ID}$.

Definition 8.3 (Φ-RKA Public Key Encryption Scheme) *A public key encryption (PKE) scheme is comprised of a tuple of algorithms for key generation, encryption and decryption algorithms, as* $\mathcal{PKE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. *The scheme is associated with a security parameter* k, *and is assumed to have an efficiently computable message space* $\mathcal{M}(k)$.

The scheme has an associated security parameter k, and is assumed to have an efficiently computable message space $\mathcal{M}(k)$. The key generation algorithm $\mathcal{K}$ takes 1^k as input and returns the public key ek and secret key dk, as $(dk, ek) \leftarrow \mathcal{K}(1^k)$. The encryption function takes a public key and a message to create a ciphertext as (ek, m) for $m \in \mathcal{M}(k)$. Finally, the decryption algorithm takes a secret key and a ciphertext c to produce a message as $\mathcal{D}(dk, c)$.

For correctness, the scheme requires that for a properly generated key pair, a ciphertext that is generated as the encryption of a message can be decrypted using the associated secret key back to the original message.

The following security game is defined for a compatible class of RKD functions Φ and an adversary A.

1. Setup Phase. *The challenger generates public and secret keys as $(ek, dk) \xleftarrow{\$} \mathcal{K}(1^k)$. The challenger also generates a random bit $b \xleftarrow{\$} \{0,1\}$, and initializes the challenge ciphertext as $C^* \leftarrow \perp$. The challenger returns (π, ek) to the adversary.*

2. Query Phase I. *A is given oracle access to $\mathcal{O}^{\mathcal{D},\Phi}$. When the challenger receives query $\mathcal{O}^{\mathcal{D},\Phi}(\phi, C)$, it first generates the tampered secret key as $dk' \leftarrow \phi(dk)$. The challenger returns the decryption $M \leftarrow \mathcal{D}(dk', C)$ to A.*

3. Challenge. *The adversary outputs a pair of messages (m_0, m_1) with $|m_0| = |m_1|$; the challenger then returns the challenge ciphertext $C^* \xleftarrow{\$} \mathcal{E}(\pi, ek, m_b)$ in response.*

4. Query Phase II. *A is again given oracle access as $\mathcal{O}^{\mathcal{D},\Phi}$. When the challenger receives query $\mathcal{O}^{\mathcal{D},\Phi}(\phi, C)$, it first generates the tampered secret key as $dk' \leftarrow \phi(dk)$. If $((dk' = dk) \wedge (C = C^*))$, the challenger returns $\perp$; otherwise the challenger returns $M \leftarrow \mathcal{D}(dk', C)$.*

5. Guess. *Finally, the adversary halts and outputs b', its guess of b.*

The advantage of the adversary is defined as $\mathbf{Adv}^{\text{pke-cc-rka}}_{\mathcal{PKE},A,\Phi}(k) = \Pr[b = b'] - \frac{1}{2}$. We say $\mathcal{PKE}$ is Φ-RKA secure if this advantage function $\mathbf{Adv}^{\text{pke-cc-rka}}_{\mathcal{PKE},A,\Phi}(k)$ is negligible in k for all adversaries A that run in time polynomial in k.

Weak Psuedorandom Functions.

Weak psuedorandom functions (wPRFs) are a relaxation of PRFs such that they only need to be indistinguishable from a truly random function when queried at random points in the domain, instead of at adversarially chosen points. We include wPRFs since they have an interesting separation with PRFs.

Definition 8.4 (Φ-RKA Weak Psuedorandom Function (wPRF).) *Let $\mathcal{wPRF} = (\mathcal{K}, \mathcal{F})$ be a family of functions with an associated security parameter k, with efficiently computable domain $\mathsf{Dom}_{\mathcal{wPRF}}(k)$ and range $\mathsf{Rng}_{\mathcal{wPRF}}(k)$.*

The following security game is parameterized by a class of allowed RKD functions Φ that is compatible with $\mathcal{wPRF}$, and is defined for an adversary A.

1. Setup Phase. *The challenger generates a key for the wPRF as $K \xleftarrow{\$} \mathcal{K}(1^k)$, and selects a random bit $b \xleftarrow{\$} \{0,1\}$.*

2. Query Phase. *A is given oracle access as $\mathcal{O}^{\mathcal{wPRF},\Phi}(\phi)$. Upon receiving an oracle query (ϕ), the challenger first computes the tampered key $K' \leftarrow \phi(K)$, and selects a value x uniformly from the domain of $\mathcal{wPRF}$ as $x \xleftarrow{\$} \mathsf{Dom}_{\mathcal{wPRF}}(k)$.*

   If $b = 0$ and the adversary has never received a value for x and K', the challenger randomly selects a point $y \xleftarrow{\$} \mathsf{Rng}(k)$ and defines $T[K', x] = y$. The challenger answers the oracle query by returning $(x, T[K', x])$. Otherwise, if $b = 1$ then the challenger responds to a query x with $(x, \mathcal{F}(K', x))$.

3. Guess. *The adversary must return a guess b' of the hidden bit.*

We define $\mathbf{Adv}^{\text{wprf-rka}}_{\mathcal{wPRF},A,\Phi}(k) = \Pr[b = b'] - \frac{1}{2}$; we say $\mathcal{wPRF}$ is Φ-RKA secure if this advantage function is negligible in k for all A that run in time polynomial in k.

Symmetric Encryption. Symmetric encryption enables a sender and a receiver that share a secret key to send encrypted messages to each other. This case is interesting because we can now consider RKAs on the encryption algorithm as well as on the decryption algorithm.

We consider two different notions of security for symmetric encryption:chosen-plaintext attack (CPA) and chosen-ciphertext attack (CCA). In CPA security, the adversary is allowed to repeatedly query the encryption oracle with pairs of messages, and receives the encryption of one in return in a semantic security challenge. In CCA security, the adversary is additionally given access to a decryption oracle, but the decryption oracle will disallow decryption of any challenge ciphertext.

Definition 8.5 (Φ-RKA CPA and CCA secure SE Schemes) *A symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is specified by key generation, encryption and decryption algorithms. The scheme is associated with a security parameter k, and is assumed to have an efficiently computable message space $\mathcal{M}(k)$.*

The key generation algorithm $\mathcal{K}$ takes 1^k as input and returns a secret key K, as $(K) \leftarrow \mathcal{K}(1^k)$. The encryption function takes a secret key and a message to create a ciphertext as (K, m) for $m \in \mathcal{M}(k)$. Finally, the decryption algorithm takes a secret key and a ciphertext c to produce a message as $\mathcal{D}(K, c)$.

For correctness, the scheme requires that for a properly generated secret key, a ciphertext that is generated as the encryption of a message can be decrypted using the same secret key back to the original message.

The following security game is defined for a compatible class of RKD functions Φ and an adversary A.

1. Setup phase. *The challenger generates a secret key with the key generation algorithm, as $K \leftarrow \mathcal{K}(1^k)$. The challenger also generates a random bit $b \xleftarrow{\$} \{0,1\}$, and initializes the set of messages disallowed for decryption as $S \leftarrow \emptyset$.*

2. Query Phase – CPA. *The adversary A is given adaptive access to $\mathcal{O}^{\mathcal{E},\Phi}(\phi, m_0, m_1)$, where $|m_0| = |m_1|$. To answer, the challenger first computes the tampered key as $K' \leftarrow \phi(K)$, and then computes ciphertext $C \xleftarrow{\$} \mathcal{E}(K', m_b)$. C is added to the list of disallowed ciphertexts as $S \leftarrow S \cup \{(K', C^*)\}$, and then the challenger returns C to A.*

3. Query Phase – CCA. *In the case of CCA security, in addition to the oracle access granted in the CPA Query Phase, A is also given access to the decryption oracle as $\mathcal{O}^{\mathcal{D},\Phi}(\phi, C$. To answer*

this the challenger first computes the tampered key as $K' \leftarrow \phi(K)$. *If* $((K', C) \in S)$ *then the challenger returns* $\perp$ *to* A*; otherwise the challenger decrypts and returns* $M \leftarrow \mathcal{D}(K', C)$ *to* A.

4. Guess. *The adversary must return a guess* b' *of the hidden bit.*

We define $\mathbf{Adv}^{\text{se-cp-rka}}_{\mathcal{SE},A,\Phi}(k) = \Pr[b = b'] - \frac{1}{2}$ *when* A *plays the above game with the CPA Query Phase. We say* $\mathcal{SE}$ *is* Φ*-RKA-CPA secure if this advantage function is negligible in* k *for all* A *that run in time polynomial in* k.

We define $\mathbf{Adv}^{\text{se-cc-rka}}_{\mathcal{SE},A,\Phi}(k) = \Pr[b = b'] - \frac{1}{2}$ *when* A *plays the above game with the CCA Query Phase. We say* $\mathcal{SE}$ *is* Φ*-RKA-CCA secure if this advantage function is negligible in* k *for all* A *that run in time polynomial in* k.

Identity Based Encryption.

Identity Based Encryption (IBE) was first defined in [Sha85]. In an IBE scheme, the public key of each user is some commonly known user identification (ID), such as an email address, avoiding the common issues associated with public key exchange. An IBE scheme is enabled by a trusted party with master keys that generates the secret keys for any ID. We will give the definition for a Φ-RKA secure IBE scheme without first giving the standard definition of an IBE scheme, but the oriental definition can be obtained by letting $\Phi = \mathsf{ID}$.

As with the definition for signatures, some subtleties arise in actions disallowed to the Φ-RKA adversary. In the standard security game, the adversary can adaptively request secret keys for user IDs of its choice to a key derivation oracle, then request a semantic security challenge encrypted with the ID of its choice, and then further request secret keys for IDs; however, its important for the adversary to be disallowed to have the secret key for the ID used to generate the semantic security challenge! In the Φ-RKA security definition, the key-derivation oracle refuses to act only when the ID it is given matches the challenge one *and* the tampering does not modify the secret key.

Definition 8.6 (Φ-RKA IBE Scheme) *An identity-based encryption (IBE) scheme is comprised of a tuple of algorithms* $\mathcal{IBE} = (\mathcal{M}, \mathcal{K}, \mathcal{E}, \mathcal{D})$*, which are for master key generation, key generation,*

encryption, and decryption respectively. The scheme has an associated security parameter k, and is assumed to have an efficiently computable message space $\mathcal{M}(k)$. The master key generation algorithm $\mathcal{M}$ takes 1^k as input and returns the master public key mpk and master secret key msk, as $(msk, mpk) \leftarrow \mathcal{M}(1^k)$. Standard key generation $\mathcal{K}$ takes parameters, a user ID id, and the master secret key to produce a secret key for the ID as $dk \leftarrow \mathcal{K}(mpk, msk, id)$. The encryption function takes the master public key, a user ID, and a message to create a ciphertext as $\mathcal{E}(mpk, id, m)$ for $m \in \mathcal{M}(k)$. Finally, the decryption algorithm takes the master public key, a secret key, and a ciphertext c to produce a message as $\mathcal{D}(mpk, dk, c)$.

For correctness, the scheme requires that for a properly generated master key, when a message is encrypted to a user ID, the properly created secret key for that user ID can be used to decrypt to the original message.

Security is defined for the following game with an adversary A, and is parameterized by a Φ that is compatible with $\mathcal{IBE}$.

1. Setup Phase. *The challenger generates the master key pair as $(mpk, msk) \xleftarrow{\$} \mathcal{M}(1^k)$. The challenger forwards (π, mpk) to the adversary. The challenger also randomly selects a bit b as $b \xleftarrow{\$} \{0, 1\}$, and initializes the set of disallowed IDs as $S \leftarrow \emptyset$, and initializes the challenge ID as $id^* \leftarrow \perp$.*

2. Query Phase I. *A receives oracle access to $\mathcal{O}^{\mathcal{K},\Phi}$, submitting queries of the form $\mathcal{O}^{\mathcal{K},\Phi}(\phi, id)$. To answer, the challenger first generates the tampered key as $msk' \leftarrow \phi(msk)$. If $msk' = msk$, $S \leftarrow S \cup id$. The challenger returns secret key $dk \leftarrow \mathcal{K}(mpk, msk', id)$ to A.*

3. Challenge Phase. *The adversary sends a user ID and two messages to the challenger, as (id, m_0, m_1), where $|m_0| = |m_1|$. If $id \in S$, the challenger returns $\perp$, since the adversary has already asked for the secret key for this user ID under the original secret key. Otherwise, the challenger sets $id^* = id$, and sends A the challenge $C \xleftarrow{\$} \mathcal{E}(mpk, id, m_b)$.*

4. Query Phase II. *A again is given access to $\mathcal{O}^{\mathcal{K},\Phi}(\phi, id)$, but now queries to id^* with the unhampered master secret key are disallowed. To answer, the challenger first generates the*

tampered key as $msk' \leftarrow \phi(msk)$. *If* $((msk' = msk) \wedge (id = id^*))$, *then the challenger returns* $\perp$; *otherwise the challenger returns* $dk \leftarrow \mathcal{K}(mpk, msk', id)$.

5. Guess. *Finally, the adversary halts and outputs* b', *its guess of* b.

We define $\mathbf{Adv}^{\text{ibe-rka}}_{\mathcal{IBE},A,\Phi}(k) = \Pr[b = b'] - \frac{1}{2}$, *and say that* $\mathcal{IBE}$ *is a* Φ*-RKA secure IBE scheme if* $\mathbf{Adv}^{\text{ibe-rka}}_{\mathcal{IBE},A,\Phi}(k)$ *is negligible in* k *for all* A*that run in time polynomial in* k.

9 Constructions of Φ-RKA Secure Primitives

We will show that a Φ-RKA and Φ-ICR PRG is a particularly strong starting point, leading to a construction of any cryptographic primitive from such a PRG; in our language, this is the containment of $\mathbf{RKA}[\mathsf{PRG}]$ in $\mathbf{RKA}[\mathsf{P}]$ for all other P that we consider. To do this, we first show $\mathbf{RKA}[\mathsf{PRG}] \subseteq \mathbf{RKA}[\mathsf{Sig}]$, building a tamper-resilient signature scheme from any Φ-RKA and Φ-ICR PRG. Then we will generalize this result to use any Φ-RKA and Φ-ICR PRG to build a tamper resilient version of any of the other primitives we consider.

We also show that our constructions of a Φ-RKA secure signature satisfies an even stronger notion of security where the adversary cannot forge for *any* tampered key, and not just the original one.

Additionally, we give two positive results based on IBE, showing that previous constructions of a PKE with CCA security and of a signature scheme based on IBE maintain Φ-RKA security.

9.1 Using Φ-RKA PRG to Construct Φ-RKA Signatures

To build high level primitives resilient to tampering, we will start with a normal secure instance of that primitive. The adapted security notions of the RKA model allows an adversary to obtain information not only based on the original secret key, but also tampered versions of the secret key. To build a Φ-RKA secure primitive, we will need to ensure that information obtained based on tampered secret keys gives the adversary no additional ability to break the security of the original primitive.

We will give a general method to construct any higher level primitive from a Φ-RKA and Φ-ICR PRG. Every higher level primitive has a key generation algorithm to generate a secret key (and perhaps additionally a public key) from randomness. Instead of storing the secret key of the original scheme, the new tamper resilient primitive should instead store a seed for the PRG. When the secret key is needed, the PRG is used to generate randomness to feed into the original key generation algorithm, generating the secret key freshly each time it is needed. If the seed has been modified by a $\phi \in \Phi$, the PRG will produce an output that is indistinguishable from random, and so the secret key generated will look as if it was generated with fresh randomness, and will give no information about the original scheme.

To explore this idea in detail, we will formally define the construction in question and give the necessary security reductions for Φ-RKA Signatures.

From Φ-RKA PRGs to Φ-RKA signatures. We will show containments of the form $\mathbf{RKA}[\mathsf{PRF}] \subseteq \mathbf{RKA}[\mathsf{P}]$ for a range of primitives P by showing that any Φ-RKA and Φ-ICR secure PRG can be used to build Φ-RKA secure P. Under the assumption that a Φ-RKA PRF has an identity fingerprint, we can use it to build such a PRG, giving the desired result.

Construction 9.1 [A Φ-RKA Signature Scheme] We start with a Φ-RKA and Φ-ICR PRG $\mathcal{PRG} = (\mathcal{K}, \mathcal{G})$ with associate output length given by $r(\cdot)$, and a normal-secure signature scheme $\overline{\mathcal{DS}} = (\overline{\mathcal{K}}, \overline{\mathcal{S}}, \overline{\mathcal{V}})$, such that the output length $r(\cdot)$ of the PRG is also the number of coins used by $\overline{\mathcal{K}}$. We construct a new signature scheme $\mathcal{DS} = (\mathcal{K}', \mathcal{S}, \mathcal{V})$, with algorithms defined as follows:

1. Keys: Pick a random seed for the PRG for the new signing key, $K \xleftarrow{\$} \mathcal{K}(1^k)$. To generate the verification key, let $(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(\mathcal{G}(K))$ be the result of the original key generation algorithm with coins from $\mathcal{G}(K)$– the verifying key remains $\overline{vk}$. (Key $\overline{sk}$ is discarded.)
2. Signing: To sign message m with signing key K, recompute $(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(\mathcal{G}(K))$, and then sign m under $\overline{\mathcal{S}}$ using $\overline{sk}$.
3. Verifying: Just as in the base scheme signature scheme, verify that σ is a signature of m under $\overline{vk}$ using $\overline{\mathcal{V}}$.

Note that Φ is compatible with the space of keys for $\mathcal{DS}$ since this is also the space of seeds for

the PRG.

We note that our construction has the advantage that the form of the public key, signatures, and the verification algorithm all remain unchanged from that of the original base signature scheme. Thus our construction gives minimal changes to software, making it easier to deploy than a totally new scheme. Creating a signature in the new scheme now additionally requires evaluation of a Φ-RKA-PRG, but in practice this can be instantiated with any efficient block cipher. However, creating a signature also requires running the key generation algorithm of the base signature scheme which might be expensive.

We will prove that $\mathcal{DS}$ inherits the Φ-RKA security of the PRG. The intuition here is very simple. When an adversary attacking $\mathcal{DS}$ makes a signing query (ϕ, m), the signature of m will either be under the original signing key, or will look like a signature generated with an independent and randomly generated key. Since the signing key of our new signature scheme is the seed of the Φ-RKA PRG, any modification of the signing key will generate $\phi(K)$ which looks like fresh randomness; since the output of the PRG is used to generate a signing key of the original signature scheme as using coins $\mathcal{G}(\phi(K))$, this will look like a freshly randomly generate signing key, independent of the original scheme and giving no additional information to the adversary.

There are, however, technical difficulties in the proof. In our reductions, we construct an adversary B against the Φ-RKA security of the PRG from any adversary A against the security of the new signature scheme. The standard way to construct this is for B to guess it has access to a real oracle when A succeeds in a forgery, and to guess it has access to a random oracle B when A fails. However, in order for B to tell when A succeeds, it needs to know when the tampered key of the signature scheme is unchanged, and its unclear how to do this without access to the key! We overcome this property using the Φ-ICR security of the PRG, which bounds the probability that an adversary can query a ϕ in Φ such that ϕ modifies the seed but not the output of the PRG.

Theorem 9.2 *Let signature scheme $\mathcal{DS} = (\mathcal{P} \,||\, \overline{\mathcal{P}}, \mathcal{K}', \mathcal{S}, \mathcal{V})$ be constructed as in construction 9.1, with a Φ-RKA secure and Φ-ICR secure PRG $\mathcal{PRG} = (\mathcal{P}, \mathcal{K}, \mathcal{G}, r)$, and with a normal-secure signature scheme $\overline{\mathcal{DS}} = (\overline{\mathcal{P}}, \overline{\mathcal{K}}, \overline{\mathcal{S}}, \overline{\mathcal{V}})$. Then $\mathcal{DS}$ is a Φ-RKA secure signature scheme.*

Proof:

Given an adversary A mounting an attack on the Φ-RKA security of $\mathcal{DS}$, we construct adversaries P, S and C such that for every $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A,\Phi}(k) \;\leq\; \mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},P,\Phi}(k) + \mathbf{Adv}^{\text{sig-rka}}_{\overline{\mathcal{DS}},S}(k) + \mathbf{Adv}^{\text{icr}}_{\mathcal{PRG},C,\Phi}(k)\;, \tag{1}$$

proving the theorem.

In order to make our analysis, we introduce three games for A. In Game 0, the A will receive correctly formed signatures from our signature construction 9.1.

Game 0 – G_0

1. *Setup Phase.* The challenger generates the signing key for the signature scheme, which is a seed for $\mathcal{PRG}$, as $K \xleftarrow{\$} \mathcal{K}(1^k)$, initializes $T[K] \leftarrow \mathcal{G}(K)$, and then uses the key generation of the original signature scheme $\overline{\mathcal{DS}}$ as $(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(T[K])$ to generate the verification key $\overline{vk}$. The challenger also initializes the set of disallowed messages as $M \leftarrow \emptyset$, and sets a boolean variable bad to false. Finally, the challenger returns the verification key $\overline{vk}$ to A.

2. *Query Phase.* The challenger is responsible for answering A's adaptive queries $\mathcal{O}^{\mathcal{S},\Phi}(\phi, m)$. To do this, the challenger computes the tampered signing key as $K' \leftarrow \phi(K)$. If $T[K']$ is undefined, $T[K'] \xleftarrow{\$} \mathcal{G}(K')$, and the challenger computes keys for the original signing scheme using randomness from $\mathcal{PRG}$ as $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(T[K'])$, and signs m with $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{sk}', m)$. If the signing key was unmodified, with $K' = K$, m is added to the set of disallowed messages with $M \leftarrow M \cup \{m\}$. If $K' \neq K$ but $T[K'] = T[K]$, the challenger sets bad $\leftarrow$ true. Finally, the challenger returns the signature σ in response to A's oracle query.

3. *Finalize.* When A halts and outputs a potential forgery (m^*, σ^*), the game returns 1 if the signature verifies and $m^* \notin M$ and 0 otherwise; more formally, the game returns $((\overline{\mathcal{V}}(\overline{vk}, m, \sigma) = 1) \wedge (m \notin M))$.

Game 0 is equivalent to the security game for a Φ-RKA signature scheme for our construction $\mathcal{DS}$,

and so

$$\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A,\Phi}(k) = \Pr[\mathrm{G}_0^A \Rightarrow 1]. \tag{2}$$

However, Game 0 requires the challenger to add messages to the disallowed set when $K' = K$, but the challenger doesn't have access to K! Game 1 will instead add messages to the disallowed set when $T[K'] = T[K]$; we use the Φ-RKA security of the PRG to bound the probability that A can find ϕ such that for $K' = \phi(K)$, $T[K'] = T[K]$ yet $K' \neq K$.

In Game 1, the A will receive correctly formed signatures from our signature construction 9.1.

Game 1 – G_1

> G_1 is the same as G_0, except in the query phase, where a queried message m is added to the set of disallowed messages with $M \leftarrow M \cup \{m\}$ when $T[K'] = T[K]$, instead of when $K' = K$.

An algorithm with oracle access to $\mathcal{O}^{\mathcal{PRG},\Phi}$ can perform the role of the challenger for the Φ-RKA adversary of the signature scheme in Game 1. Note that G_0 and G_1 only differ when the Φ-RKA adversary of the signature scheme makes a query to ϕ such that $T[K'] = T[K]$ yet $K' \neq K$. We construct adversary C against the Φ-ICR security of $\mathcal{PRG}$ by having it serve as the challenger in G_0; whenever the flag bad is set to true, C breaks the Φ-ICR security of $\mathcal{PRG}$. Thus,

$$\Pr[\mathrm{G}_0^A \Rightarrow 1] \leq \mathbf{Adv}^{\text{icr}}_{\mathcal{PRG},C,\Phi}(k) + \Pr[\mathrm{G}_1^A \Rightarrow 1]. \tag{3}$$

Game 2 differs from Game 1 in that A will receive signatures created using true randomness instead of the output of the $\mathcal{PRG}$. Differences in Game 2 from Game 1 are boxed.

Game 2 – G_2

1. *Setup Phase.* The challenger generates the signing key for the signature scheme, which is a seed for $\mathcal{PRG}$, as $K \xleftarrow{\$} \mathcal{K}(1^k)$, initializes $\boxed{\mathrm{T[K]} \xleftarrow{\$} \{0,1\}^{r(k)}}$, and then uses the key generation of the original signature scheme $\overline{\mathcal{DS}}$ as $(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(T[K])$ to generate the verification key $\overline{vk}$. The challenger also initializes the set of disallowed messages as $M \leftarrow \emptyset$. Finally, the

challenger sends the verification key $\overline{vk}$ to A.

2. *Query Phase.* The challenger is responsible for answering A's adaptive queries for signatures $\mathcal{O}^{\mathcal{S},\Phi}(\phi, m)$. To respond, the challenger computes the tampered signing key as $K' \leftarrow \phi(K)$. If $T[K']$ is undefined, $\boxed{\text{T[K']} \xleftarrow{\$} \{0,1\}^{r(k)}}$, and the challenger computes keys for the original signing scheme as $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(T[K'])$, and signs m with $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{sk}', m)$. If the signing key was unmodified, with $K' = K$, m is added to the set of disallowed messages with $M \leftarrow M \cup \{m\}$. Finally, the challenger returns the signature σ to A.

3. *Finalize.* When A submits a potential forgery (m^*, σ^*), the game returns 1 if the signature verifies and $m^* \notin M$ and 0 otherwise; more formally the game returns $((\overline{\mathcal{V}}(\overline{vk}, m, \sigma) = 1) \wedge (m \notin M))$.

First we use any adversary A against the signature scheme to construct P which attacks the security of the $\mathcal{PRG}$. P receives either oracle access to either a real instance of $\mathcal{PRG}$, or to a random one; P will interact with A so that in the real case, it plays Game 1 with A, and in the random case plays Game 2 with A.

P generates randomness R by querying the $\mathcal{PRG}$ oracle with ID the identity function as $R \xleftarrow{\$} \textsc{Gen}(\mathsf{id})$, and uses this to create a verification key for $\mathcal{DS}$ as $(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(R)$, sending $\overline{vk}$ as the verification key to A. P initializes the set of disallowed messages $M \leftarrow \emptyset$. P must then answer adaptive signing queries from A of the form (ϕ, m). P first obtains $R' \leftarrow \textsc{Gen}(\phi)$, then uses this as randomness to run $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(R')$. P uses the resulting signing key $\overline{sk}'$ for the original signature scheme $\overline{\mathcal{DS}}$ to sign m as $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{sk}', m)$, returning signature σ to A. If $\phi = \mathsf{ID}$, then m is added to the set of disallowed messages as $M \leftarrow M \cup \{m\}$. When A returns a message and signature (m^*, σ^*), adversary P returns 1 if $(\overline{\mathcal{V}}(\overline{vk}, m^*, \sigma^*) = 1) \wedge (m \notin M)$ and 0 otherwise.

In the case that P is in the real game, the verification key and all signatures given to A are computed using values from the real $\mathcal{PRG}$ as in Game 1; in the case that P has access to the random game, the verification key and all signatures given to A are computed using a random value for each value of $\mathcal{PRG}$ seed, as in Game 2. The remaining challenge for P is to be able determine whether the

the forged message m^* is in M, the set of disallowed messages. Messages are added to M if signed with the original signing key, but P does not have access to the $\mathcal{PRG}$ seed that is the signing key for the signature scheme for A. To solve this problem, we use the assumption that Φ is claw-free – this guarantees that $K' = K$ only when $\phi = \mathsf{ID}$, so m is added to M only for queries where $\phi = \mathsf{ID}$. (No other ϕ can have $\phi(K) = K$ since $\mathsf{ID}(K) = K$, and that would violate the assumption.)

This analysis yields

$$\mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG},P,\Phi}(k) = \Pr[\mathrm{G}_1^A \Rightarrow 1] - \Pr[\mathrm{G}_2^A \Rightarrow 1], \tag{4}$$

computed as the difference in probability of returning 1 in the real and random games for P.

To continue in our analysis we use the adversary A attacking the RKA-security of signature scheme $\mathcal{DS}$ to build an adversary S against the normal security of the original signature scheme $\overline{\mathcal{DS}}$. S receives verification key $\overline{vk}$for the original signature scheme. S then generates seed for the $\mathcal{PRG}$ as $K \xleftarrow{\$} \mathcal{K}(1^k)$, which will serve as the signing key for $\mathcal{DS}$. S sends A the verification key $\overline{vk}$. S responds to adaptive signature queries $(\phi, m)$from A by first computing $K' = \phi(K)$. If $K' = K$, S uses oracle access to get signature $\sigma \xleftarrow{\$} \textsc{Sign}(m)$. For $K' \neq K$, if $T[K']$ is undefined, S sets $T[K'] \xleftarrow{\$} \{0,1\}^{r(k)}$; for $K' \neq K$, S uses $T[K']$ as randomness for $\overline{\mathcal{K}}$ to compute $(\overline{vk}', \overline{sk}') \leftarrow \overline{\mathcal{K}}(T[K'])$ and signs m with the resulting signing key as $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{sk}', m)$. S returns the generated signature σ to A. When A submits forgery (m^*, σ^*), S submits this also.

Note that S is able to play Game 2 properly with A. The parameters presented to A have the proper distribution, since they are composed of independently generated parameters for the $\mathcal{PRG}$ and base signature scheme $\overline{\mathcal{DS}}$. Each tampered seed $K' = \phi(K)$ accesses a truly random value $T[K']$ to generate a signing key of the original scheme $\overline{\mathcal{DS}}$ to answer signature queries; this is even true when $\phi = \mathsf{ID}$, in which case signatures are generated from S's access to a signature oracle, which uses a signing key of the base signature $\overline{\mathcal{DS}}$ created with true randomness. Finally we note that since Φ is claw-free, only $\phi = \mathsf{ID}$ leaves the seed of the PRG unmodified, so no other ϕ will yeild $\phi(K) = K$.

A succeeds when $(\overline{\mathcal{V}}(\overline{vk}, m^*, \sigma^*) = 1)$ and $(m^* \notin M)$. Since S and A have the same value for $\overline{vk}$, $(\overline{\mathcal{V}}(\overline{vk}, m^*, \sigma^*) = 1)$ will be true for S exactly when it is true for A. Moreover, m is added to M

for A when it queries (ID, m); since S makes queries to its signature oracle only when $\phi = \mathsf{ID}$, m is added to M for S exactly when it is added to M for A. Thus S will succeed in creating a forgery for $\overline{\mathcal{DS}}$ exactly when A succeeds in creating a forgery for $\mathcal{DS}$ in Game 2, yielding

$$\mathbf{Adv}^{\text{sig-rka}}_{\overline{\mathcal{DS}},S}(k) = \Pr[\mathrm{G}_2^A \Rightarrow 1]. \tag{5}$$

To conclude our proof, we combine equations 2, 3, 4, and 5 progressively to give

$$\begin{aligned}
\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A,\Phi}(k) &= \Pr[\mathrm{G}_0^A \Rightarrow 1] \\
&= \mathbf{Adv}^{\text{icr}}_{\mathcal{PRG},C,\Phi}(k) + \Pr[\mathrm{G}_1^A \Rightarrow 1] \\
&= \mathbf{Adv}^{\text{icr}}_{\mathcal{PRG},C,\Phi}(k) + \mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},P,\Phi}(k) + \Pr[\mathrm{G}_2^A \Rightarrow 1] \\
&= \mathbf{Adv}^{\text{icr}}_{\mathcal{PRG},C,\Phi}(k) + \mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},P,\Phi}(k) + \mathbf{Adv}^{\text{sig-rka}}_{\overline{\mathcal{DS}},S}(k),
\end{aligned}$$

proving equation 1 as desired.

∎

Corollary 9.3 *Under the assumption that Φ-RKA secure PRF is also a Φ-IDFP secure PRF,* $\mathbf{RKA}^*[\mathsf{PRF}] \subseteq \mathbf{RKA}^*[\mathsf{Sig}]$.

Proof: Using the construction in Proposition 7.8, we can build a Φ-RKA and Φ-ICR secure PRG from any Φ-RKA and Φ-IDFP secure PRF. Using Theorem 9.1, we can in turn use this PRG and a normal secure instance of a signature scheme to build a Φ-RKA secure signature scheme.

∎

Now that we have shown how to build a Φ-RKA secure signature scheme, we generalize the above construction.

Construction 9.4 [Φ-RKA Primitives from Φ-RKA and Φ-ICR PRG] Let $\mathsf{P} = (\mathcal{K}_{\mathsf{P}}(\cdot), g_1(\cdot), \ldots, g_m(\cdot), f_1(sk, \cdot), \ldots$ be any secure cryptographic primitive, where each algorithm f_i depends on the secret key sk, and

each algorithm g_i does not. Without loss of generality, key generation outputs both a private and a public key, and all algorithms run in time polynomial in k.

Let Φ-RKA and Φ-ICR PRG $\mathcal{PRG} = (\mathcal{K}_{\mathcal{PRG}}, \mathcal{G})$ with associate output length given by $r(k)$, such that the output length $r(k)$ of the PRG is also the number of coins used by $\mathcal{K}(1^k)$.

We construct $\mathsf{P}' = (\mathcal{K}'(\cdot), g'_1(\cdot), \ldots, g'_m(\cdot), f'_1(sk, \cdot), \ldots, f'_n(sk, \cdot))$ as follows:

- $\mathcal{K}'(1^k)$ generates secret key $sk \leftarrow \mathcal{K}_{\mathcal{PRG}}(1^k)$, and then takes the public key obtained by running the original key generation algorithm on the psuedorandomness produced by $\mathcal{G}$ evaluated with sk, as $(pk, -) \leftarrow \mathcal{K}_{\mathsf{P}}(1^k; \mathcal{G}(sk))$; the returned key pair is (sk, pk).
- $f_1(sk, \cdot)$ first generates $(pk, sk_{\mathsf{P}}) \leftarrow \mathcal{K}_{\mathsf{P}}(1^k; \mathcal{G}(sk))$, and then returns $f_i(sk_{\mathsf{P}}, \cdot)$.
- All $g'_i(\cdot) = g_i(\cdot)$; algorithms that do not use the secret key are unchanged.

Note that Φ is compatible with the space of keys for P' since this is also the space of seeds for the PRG.

Theorem 9.5 *Let cryptographic primitive* $\mathsf{P}' = (\mathcal{K}'(\cdot), g'_1(\cdot), \ldots, g'_m(\cdot), f'_1(sk, \cdot), \ldots, f'_n(sk, \cdot))$ *be constructed as in construction 9.4, from a* Φ*-RKA secure and* Φ*-ICR secure PRG* $\mathcal{PRG} = (\mathcal{P}, \mathcal{K}, \mathcal{G}, r)$*, and from any normal secure instance of the cryptographic primitive* $\mathsf{P} = (\mathcal{K}_{\mathsf{P}}(\cdot), g_1(\cdot), \ldots, g_m(\cdot), f_1(sk, \cdot), \ldots, f_n(sk,$ *Then* P' *is a* Φ*-RKA secure instance of* P*.*

The proof is a generalized version of Theorem , and we omit it here.

9.2 Strong Φ-RKA Security

We suggest and use an even stronger notion of Φ-RKA security for signature schemes and other primitives. The security game for Φ-RKA secure signatures requires that forgery is difficult with respect to the original verification key; we note that our Construction 9.1 actually possesses an even stronger property, namely that forgery is difficult with respect to the public verification keys corresponding to *any* tampered secret key. We will use signature schemes to demonstrate this strong security requirement, and will use it in other proofs.

In the general definition of a signature scheme the public verification and the secret signing keys are produced together by the (randomized) key generation algorithm. However, in most instantiations of signature schemes, the secret signing key is produced first with randomness and then the public verification key is generated as a deterministic function of the secret key. We call any signature scheme with this property *separable*, and we call the algorithm that deterministically produces the public key from the secret key the *public-key generator*. We note that one can convert any signature scheme into a separable scheme by storing randomness for the key generation algorithm as the new secret key.

Definition 9.6 (Separable Signature Scheme) *A signature scheme $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is* separable *if there is a deterministic algorithm $\mathcal{T}$, called the public-key generator, such that for all $k \in \mathbb{N}$ the output of the process*

$$(vk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\,;\ vk \leftarrow \mathcal{T}(sk)\,;\ \ \textit{Return}\ (vk, sk)$$

is distributed identically to the output of $\mathcal{K}(1^k)$.

We will now give the security definition for a strongly Φ-RKA secure signature scheme. Here, the adversaries potential forgery includes not only a message and a signature, but also a RKD function ϕ. A forgery is valid if it passes verification with the verification key vk derived from $\phi(sk)$, *and* the adversary has never received a signature on this message generated with a signing key sk' that gives vk. This might seem odd since the adversary might not know the public key, but without loss of generality we can add the public verification key to the signatures. We note that this definition of a forgery is more restrictive than that of normal Φ-RKA security; one can move to the less restrictive form under a general (not just identity) fingerprint assumption.

Definition 9.7 (Strongly Φ-RKA Secure Signature Scheme) *Let $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ be a separable signature scheme with public-key generator $\mathcal{T}$. The algorithms must work in the same way and satisfy the same correctness properties as a standard signature scheme.*

We define the following security game is parameterized by a security parameter k and a Φ that is compatible with $\mathcal{DS}$, and is defined for an adversary A.

1. Setup Phase. *The challenger generates keys for the signature scheme as* $(vk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$, *and initializes the set of messages disallowed for forgery as* $M \leftarrow \emptyset$. *Finally, the challenger gives the public verification key* vk *to adversary* A.

2. Query Phase. A *is given adaptive access to* $\mathcal{O}^{\mathcal{S},\Phi}$. *The challenger answers* A*'s queries* $\mathcal{O}^{\mathcal{S},\Phi}(\phi, m)$ *by first computing* $sk' \leftarrow \phi(sk)$. *The challenger computes the associated public key for the tampered secret key as* $vk \leftarrow \mathcal{T}(sk')$. *The challenger computes signature* $\sigma \xleftarrow{\$} \mathcal{S}(sk', m)$, *and adds this to the set of disallowed forgeries as* $M \leftarrow M \cup \{(vk, m)\}$. *Finally, the challenger returns the signature* σ *to answer* A*'s query.*

3. Forge. *When* A *halts, it outputs* (ϕ, m, σ).

For A*'s output* (ϕ, m, σ), *we define* $\mathbf{Adv}^{\text{ssig-rka}}_{\mathcal{DS},\mathcal{T},A,\Phi}(k) = \Pr[vk \leftarrow \mathcal{T}(\phi(sk)) : (\mathcal{V}(vk, m, \sigma) = \mathsf{true}) \wedge ((vk, m) \notin M)]$. *We say* $(\mathcal{DS}, \mathcal{T})$ *is strongly* Φ*-RKA secure if this advantage function is negligible in* k *for all* A *that run in time polynomial in* k.

To show that our signature scheme Construction 9.1, is strongly Φ-RKA secure, we first note that this construction is separable. Recall that in this scheme, the secret signing key is a seed for a Φ-RKA PRG, which is expanded and fed into a signature scheme key generation algorithm; the verification key is obtained from the signature scheme key generation algorithm, and so can be computed from the secret key. This is true regardless of whether the original signature scheme is separable.

Theorem 9.8 *Let separable signature scheme* $\mathcal{DS} = (\mathcal{K}', \mathcal{S}, \mathcal{V})$ *be constructed as in Construction 9.1 from a* Φ*-RKA PRG* $\mathcal{PRG} = (\mathcal{K}, \mathcal{G})$ *with output length given by* $r(k)$, *and normal-secure signature scheme* $\overline{\mathcal{DS}} = (\overline{\mathcal{K}}, \overline{\mathcal{S}}, \overline{\mathcal{V}})$. *Let* $\mathcal{T}$ *be the associated public-key generation algorithm for* $\mathcal{DS}$. *Then* $(\mathcal{DS}, \mathcal{T})$ *is a strongly* Φ*-RKA secure signature scheme.*

Proof of Theorem 9.8:

Recall ID is the RKA specification consisting of just the identity function, so that a ID-RKA secure signature scheme has normal signature security; by assumption, $\overline{\mathcal{DS}}$ is such a scheme. Given an

adversary A making q_{sig} signature oracle queries and mounting a Φ-RKA on the *strong* security of $\mathcal{DS}$ with public-key generator $\mathcal{T}$, we construct adversaries P, S such that for every $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{ssig-rka}}_{\mathcal{DS},\mathcal{T},A,\Phi}(k) \leq \mathbf{Adv}^{\text{prg}}_{\mathcal{PRG},P,\Phi}(k) + (q_{sig}+1) \cdot \mathbf{Adv}^{\text{sig-rka}}_{\overline{\mathcal{DS}},S,\mathsf{ID}}(k),$$

which will prove the theorem.

We define two security games for an adversary A against the strong Φ-RKA security of our constructed signature scheme.

Game 0 and Game 1 – G_0 and G_1

Differences in Game 1 from Game 0 are boxed.

1. *Setup Phase for* G_0, G_1. The challenger generates a seed for the PRG as $K \xleftarrow{\$} \mathcal{K}(1^k)$, and stores its initial value as $T[K] \leftarrow \mathcal{G}(K)$. The challenger also initializes the messages disallowed for forgery as $M \leftarrow \emptyset$. Finally, the challenger generates the public verification key as $(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(T[K])$, and returns $(\overline{vk})$ to A.

2a. *Query Phase for* G_0. To answer A's adaptive queries $\mathcal{O}^{\mathcal{S},\Phi}(\phi, m)$, the challenger first computes the tampered secret key as $K' \leftarrow \phi(K)$. If $T[K'] = \perp$ then the challenger computes $T[K'] \leftarrow \mathcal{G}(K')$. The challenger generates the signature by first computing keys for the original signature scheme as $(\overline{vk}_i, \overline{sk}_i) \leftarrow \overline{\mathcal{K}}(T[K'])$, and then computing $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{sk}_i, m)$. The challenger adds this to the list of disallowed forgeries as $M \leftarrow M \cup \{(\overline{vk}_i, m)\}$, and finally returns signature σ to A.

2b. *Query Phase for* G_1. To answer A's adaptive queries $\mathcal{O}^{\mathcal{S},\Phi}(\phi, m)$, the challenger first computes the tampered secret key as $K' \leftarrow \phi(K)$. If $T[K'] = \perp$ then the challenger computes $\boxed{\text{T[K']} \xleftarrow{\$} \{0,1\}^{r(k)}}$. The challenger generates the signature by first computing keys for the original signature scheme as $(\overline{vk}_i, \overline{sk}_i) \leftarrow \overline{\mathcal{K}}(T[K'])$, and then computing $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{sk}_i, m)$. The challenger adds this to the list of disallowed forgeries as $M \leftarrow M \cup \{(\overline{vk}_i, m)\}$, and finally returns signature σ to A.

3. *Finalize for* $\mathrm{G}_0, \mathrm{G}_1$. Eventually A halts and outputs (ϕ, m, σ). The challenger computes $sk' \leftarrow \phi(sk)$ and $vk \leftarrow \mathcal{T}(sk')$; the game returns $((\mathcal{V}(vk, m, \sigma) = 1) \wedge ((sk', m) \notin M))$.

The difference in the two games here is small. Where game G_0 correctly uses the Φ-RKA PRG to generate randomness for key generation, game G_1 instead uses true randomness. We construct adversaries P, S such that for every $k \in \mathbb{N}$

$$\Pr[\mathrm{G}_0^A \Rightarrow \mathsf{true}] - \Pr[\mathrm{G}_1^A \Rightarrow \mathsf{true}] \leq \mathbf{Adv}^{\mathrm{prg}}_{\mathcal{PRG},P,\Phi}(k) \tag{6}$$

$$\frac{1}{q_{sig}} \Pr[\mathrm{G}_1^A \Rightarrow \mathsf{true}] \leq \mathbf{Adv}^{\text{sig-rka}}_{\overline{\mathcal{DS}},S,\mathsf{ID}}(k)\ . \tag{7}$$

Together with

$$\mathbf{Adv}^{\text{ssig-rka}}_{\mathcal{DS},\mathcal{T},A,\Phi}(k) = \Pr[\mathrm{G}_0^A \Rightarrow \mathsf{true}],$$

this gives the desired result.

Adversary P against the PRG runs initializations

$$M \leftarrow \emptyset\ ;\ K \xleftarrow{\$} \mathcal{K}(1^k)\ ;\ T[K] \leftarrow \mathcal{G}(K)\ ;\ (\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(T[K]).$$

P then runs A on inputs $(\overline{vk})$, and answers A's signing oracle queries $\mathcal{O}^{\mathcal{S},\Phi}(\phi, m)$. P does this by first computing the tampered seed as $K' \leftarrow \phi(K)$. If P has $T[K'] = \perp$ then it generates $T[K'] \leftarrow \mathcal{G}(K')$. P generates keys for the original signature scheme as $(\overline{vk}_i, \overline{sk}_i) \leftarrow \overline{\mathcal{K}}(T[K'])$, generates the signature as $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{sk}_i, m)$, and adds to the set of disallowed messages for forgery as $M \leftarrow M \cup \{(\overline{vk}_i, m)\}$. Finally, P returns signature σ in answer to A's query. When A halts and outputs (ϕ^*, m^*, σ^*), P tests whether this is valid by generating tampered key $K^* \leftarrow \phi(K)$, defining $T[K^*] \leftarrow \mathcal{G}(K^*)$ if necessary, and finally computing $(\overline{vk}^*, \overline{sk}^*) \leftarrow \overline{\mathcal{K}}(T[K^*])$; adversary P returns 1 if $(\mathcal{V}(vk^*, m^*, \sigma^*) = 1) \wedge ((vk^*, m^*) \notin M)$ and 0 otherwise. We note that P correctly plays G_0 with A.

Adversary S against the base signature scheme $\overline{\mathcal{DS}}$ receives $\overline{vk}$, a public key generated for the original signature scheme, and access to a corresponding signature oracle. S simulates game G_1 by generating a random signature scheme for all but one distinct tampered seed generated by the ϕ in A's queries. Let $q_sig(k)$ be an upper bound on the number of signing queries made by A. With probability $1/(q_{sig}+2)$, S can predict which ϕ and corresponding key A will select for its forgery, and S will imbed its signature scheme of interest here. S begins by initializing the set of disallowed messages for forgery as $M \leftarrow \emptyset$, generating a seed for the PRG as $K \xleftarrow{\$} \mathcal{K}(1^k)$, setting $T[K] \leftarrow \mathcal{G}(K)$, and setting the counter of unique seeds encountered as $c \leftarrow 1$.S then generates $x \xleftarrow{\$} [0, q_{sig}]$. If $x = 0$, S will embed its signature scheme in the untampered scheme A can access, setting $vk \leftarrow \overline{vk}$ and $K_0 \leftarrow K$. Otherwise, for larger x, S selects randomness $s \xleftarrow{\$} \{0,1\}^{r(k)}$ and uses it to generate $(vk, sk) \leftarrow \overline{\mathcal{K}}(\overline{\pi}; s)$.

S then gives A verification key vk, and answers A's adaptive queries to the signing oracle $\mathcal{O}^{\mathcal{S},\Phi}(\phi, m)$. To do this, S first computes the tampered seed $K' \leftarrow \phi(K)$. If $T[K'] = \perp$ this is the first time S has encountered this seed, so S increments the counter as $c \leftarrow c+1$, and sets $T[K'] \leftarrow \mathcal{G}(K')$, and if $c = x$ embeds the challenge here by setting $K_0 \leftarrow K'$.

If the tampered key $K' = K_0$, S answers with signatures from its own signing oracle, with $\sigma \leftarrow \mathcal{O}^{\mathcal{S}}(m)$, and adds $M \leftarrow M \cup \{(vk, m)\}$ to the set of disallowed forgeries. Otherwise, S uses a stored, randomly generated signing key, setting $(\overline{vk}_i, \overline{sk}_i) \leftarrow \overline{\mathcal{K}}(T[K'])$, and computing $\sigma \xleftarrow{\$} \overline{\mathcal{S}}(\overline{sk}_i, m)$, adding to the set of disallowed forgeries as $M \leftarrow M \cup \{(vk_i, m)\}$. In either case, S returns σ to A to answer the query.

When A halts and outputs (ϕ^*, m^*, σ^*), S halts and returns (m^*, σ^*). To find whether A has created a successful forgery, we compute $K^* \leftarrow \phi(K)$, define $T[K^*] \leftarrow \mathcal{G}(K^*)$ if necessary, and finally compute $(\overline{vk}^*, \overline{sk}^*) \leftarrow \overline{\mathcal{K}}(T[K^*])$; A's forgery is successful if $(\mathcal{V}(vk^*, m^*, \sigma^*) = 1) \wedge ((vk^*, m^*) \notin M)$. If ϕ^* yields a previously used K^*, there is a $1/(q_{sig}+1)$ chance that vk^* matches S's vk, and in this case S will be successful when A is. Otherwise if S never used K^*, vk^* is distributed as a verification key for a fresh instance of the signature scheme, and A succeeds with probability of at most breaking the original signature scheme.

Strong Security For Other Primitives.

The analogous strong security definition also exists for Φ-RKA CCA public key encryption. In this definition, 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.

9.3 Φ-RKA Secure Signatures from Φ-RKA Secure IBE Schemes

As noted by Naor [BF03], any IND-ID-CCA secure IBE gives a public key signature scheme. We show that **RKA**[IBE] ⊆ **RKA**[Sig] by proving the Naor transform reserves Φ- RKA security.

Construction 9.9 Given an IBE scheme $\mathcal{IBE} = (\mathcal{M}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, we construct signature scheme $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ as follows:

- $\mathcal{K}(1^k)$ first generates $(msk, mpk) \leftarrow \mathcal{M}(1^k)$, and returns these as signing and verification keys $(sk, vk) = (msk, mpk)$.
- $\mathcal{S}(sk, m)$ treats m as a user ID and returns that user IDs secret key generated as $\mathcal{K}(vk, sk, m)$.
- In the most general case verification $\mathcal{V}(vk, m, \sigma)$ is performed by encrypting a random message in the IBE scheme using the m as the user ID, and seeing whether the resulting ciphertext decrypts correctly under the decryption key σ.

The following says $\mathcal{DS}$ inherits the Φ-RKA security of $\mathcal{IBE}$.

Theorem 9.10 *Let signature scheme $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ be constructed as in Construction 9.9 above from Φ-RKA IBE scheme $\mathcal{IBE} = (\mathcal{M}, \mathcal{K}, \mathcal{E}, \mathcal{D})$. Then $\mathcal{DS}$ is Φ-RKA secure.*

When the adversary against the signature scheme makes queries to the signing oracle, the signatures can be simulated using the key generation oracle for the IBE scheme. The delicate point here is that the correctness of this simulation relies on the fact that the challenge identity (in this case, the message to be signed) 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 to generate a challenge ciphertext C^*. We then decrypt C^* using $\mathcal{D}$ with decryption key dk and return 0 if we get back m_0 and 1 otherwise.

Proof:

Correctness of the signature scheme follows from the correctness of the IBE scheme.

We reduce Φ-RKA security of the signature scheme to the Φ-RKA security of the IBE scheme by constructing an adversary B against the Φ-RKA security of $\mathcal{IBE}$ from any adversary A against the Φ-RKA security of $\mathcal{DS}$.

B receives a master public key mpk of the $\mathcal{IBE}$ scheme, and forwards it to A as the verification key vk of the signature scheme. B answers A's adaptive queries $\mathcal{O}^{\mathcal{S},\Phi}(\phi, m)$ by returning $\sigma \leftarrow \mathcal{O}^{\mathcal{K},\Phi}(\phi, m)$.

When A halts and outputs forgery (m^*, σ^*). In response, B submits m^* as the user ID and random messages m_0, m_1 that A has never queried to be used to generate its challenge. B receives ciphertext $C = \mathcal{E}(mpk, m^*, m_b)$ for a hidden random challenge bit b. B decrypts the ciphertext as $m = \mathcal{D}(mpk, \sigma^*, C)$; if $m = m_0$ then B guesses $b' = 0$, and otherwise B guesses $b' = 1$.

By definition of a forgery for A, the submitted σ^* should correctly decrypt a ciphertext of a random message generated with identity m^*. When A correctly decrypts C that is the encryption of either random message m_0 or m_1, B correctly identifies b, except perhaps when $m_0 = m_1$.

Thus $\mathbf{Adv}^{\text{ibe-rka}}_{\mathcal{IBE},B,\Phi}(k) \geq \mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A,\Phi}(k) - \frac{1}{|\mathcal{M}|}$, and so the advantage of B will be non-negligible when A's advantage is non-negligible.

9.4 Φ-RKA Secure PKE-CCA from Φ-RKA Secure IBE

We show the containment **RKA**[IBE] $\subseteq$ **RKA**[PKE-CCA] by showing that the construction of Boneh, Canneti, Halevi, and Katz [?] preserves RKA security.

Construction 9.11 [Based on Boneh, Canneti, Halevi, and Katz [?]] Let $\mathcal{IBE} = (\mathcal{M}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be any Φ-RKA IBE scheme and let $\overline{\mathcal{DS}} = (\overline{\mathcal{K}}, \overline{\mathcal{S}}, \overline{\mathcal{V}})$ be any a regular-secure strongly-unforgeable signature scheme.

We construct a public key encryption scheme $\mathcal{PKE}$ with associated security parameter k as follows:

- Keys: Pick $(mpk, msk) \leftarrow \mathcal{M}(1^k)$, and use mpk as the public key and msk as the secret key.
- Encryption: To encrypt a message m under mpk, generate a signing-verification key pair $(\overline{vk}, \overline{sk}) \leftarrow \overline{\mathcal{K}}(1^k)$. Then encrypt m for the identity $id = \overline{vk}$ by computing $c \xleftarrow{\$} \mathcal{E}(\overline{vk}, m)$ and sign c under $\overline{\mathcal{S}}$ using $\overline{sk}$ to get a signature $\overline{\sigma}$. The ciphertext is $(c, \overline{vk}, \overline{\sigma})$.
- 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.

Since the form of keys for $\mathcal{PKE}$ is the same as that for $\mathcal{IBE}$, Φ is compatible with $\mathcal{PKE}$.

Theorem 9.12 *Let the PKE scheme $\mathcal{PKE} = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ be constructed as above in Construction 9.11, using a Φ-RKA secure IBE scheme $\mathcal{IBE} = (\mathcal{M}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ and normal-secure strongly-unforgeable signature scheme $\overline{\mathcal{DS}} = (\overline{\mathcal{K}}, \overline{\mathcal{S}}, \overline{\mathcal{V}})$. Then $\mathcal{PKE}$ is Φ-RKA secure.*

The proof is an adaptation of the original proof for the non-RKA version of this theorem. The additional difficult is to show that RKA games for IBE and PKE-CCA will cooperate, because they each have rules for disallowing certain queries; in IBE the adversary is disallowed to extract a key for user ID used to issue the challenge ciphertext, and in PKE-CCA the adversary is disallowed to

decrypt the challenge ciphertext C^*, but both rules only hold under the original unmodified secret key.

Proof:

The proof will proceed through a sequence of games, Game 0, Game 1, and Game 2.

Games 0, 1, and 2 – G_0, G_1, and G_2

Each game is designed for an adversary A against the Φ-RKA security of $\mathcal{PKE}$. Differences in Game 1 from Game 0, and in Game 2 from Game 1, are boxed.

1. *Setup Phase for* G_0, G_1, G_2. The challenger generates a key pair for the IBE scheme as $(mpk, msk) \xleftarrow{\$} \mathcal{M}(1^k)$. The challenger also selects a challenge bit $b \xleftarrow{\$} \{0,1\}$, and initializes $C^* \leftarrow \perp$. Finally, the challenger returns mpk to the adversary A as the public key of the PKE scheme.

2a. *Query Phase for* G_0. During this phase, A may choose to run the Issue Challenge phase and then return to this phase. A is allowed to adaptively query $\mathcal{O}^{\mathcal{D},\Phi}(\phi, (c, \overline{vk}, \overline{\sigma}))$. To answer the query, the challenger first computes the tampered key as $msk' \leftarrow \phi(msk)$. If $((msk' = msk) \wedge ((c, \overline{vk}, \overline{\sigma}) = C^*))$, then this the challenge ciphertext and the key used to generate it, and so the challenger returns $\perp$ to this query. If $\overline{\mathcal{V}}(\overline{vk}, c, \overline{\sigma}) = \mathsf{false}$, this is an improperly formed ciphertext, and so again the challenger can return $\perp$. If neither of these conditions are true, the challenger computes $dk \leftarrow \mathcal{K}(msk', \overline{vk})$, and uses this to find $M \leftarrow \mathcal{D}(dk, c)$; the challenger answers A's query by returning M.

2b. *Query Phase for* G_1. During this phase, A may choose to run the Issue Challenge phase and then return to this phase. A is allowed to adaptively query $\mathcal{O}^{\mathcal{D},\Phi}(\phi, (c, \overline{vk}, \overline{\sigma}))$. To answer the query, the challenger first computes the tampered key as $msk' \leftarrow \phi(msk)$. If $((msk' = msk) \wedge ((c, \overline{vk}, \overline{\sigma}) = C^*))$, then this the challenge ciphertext and the key used to generate it, and so the challenger returns $\perp$ to this query. If $\overline{\mathcal{V}}(\overline{vk}, c, \overline{\sigma}) = \mathsf{false}$, this is an improperly formed ciphertext, and so again the challenger can return $\perp$. In this game, the challenger adds an additional check: if $(\overline{vk} = \overline{vk}^*) \wedge ((c, \overline{\sigma}) \neq (c^*, \overline{\sigma}^*))$, then the challenger sets flag $\mathsf{bad} \leftarrow \mathsf{true}$, and returns $\perp$ to A.

If none of these conditions are true, the challenger computes $dk \leftarrow \mathcal{K}(msk', \overline{vk})$, and uses this to find $M \leftarrow \mathcal{D}(dk, c)$; the challenger answers A's query by returning M.

2c. *Query Phase for* G_2. During this phase, A may choose to run the Issue Challenge phase and then return to this phase. A is allowed to adaptively query $\mathcal{O}^{\mathcal{D},\Phi}(\phi, (c, \overline{vk}, \overline{\sigma}))$. To answer the query, the challenger first computes the tampered key as $msk' \leftarrow \phi(msk)$. If $((msk' = msk) \wedge (\boxed{(\overline{vk} = \overline{vk}^*)}))$, then the challenger returns $\perp$ to this query. If $\overline{\mathcal{V}}(\overline{vk}, c, \overline{\sigma}) = \mathsf{false}$, this is an improperly formed ciphertext, and so again the challenger can return $\perp$. If none of these conditions are true, the challenger computes $dk \leftarrow \mathcal{K}(msk', \overline{vk})$, and uses this to find $M \leftarrow \mathcal{D}(dk, c)$; the challenger answers A's query by returning M.

3. *Issue Challenge for* G_0, G_1, G_2. Here, the adversary outputs two message m_0 and m_1 with $|m_0| = |m_1|$. To encrypt m_b, the challenger computes $(\overline{vk}^*, \overline{sk}^*) \xleftarrow{\$} \overline{\mathcal{K}}(1^k)$, $c^* \xleftarrow{\$} \mathcal{E}(, mpk, \overline{vk}^*, m_b)$, and finally $\overline{\sigma}^* \leftarrow \overline{\mathcal{S}}(\overline{sk}^*, c^*)$. Finally, the challenge ciphertext is $C^* \leftarrow (c^*, \overline{vk}^*, \overline{\sigma}^*)$, which the challenger returns to A.

4. *Finalize for* G_0, G_1, G_2. When A halts it must output a guess b' of the hidden bit b.

Game G_0 implements the security game of $\mathcal{PKE}$ with A, so we have

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

Game G_1 has the challenger add an additional before decrypting for A: now if $\overline{vk} = \overline{vk}^*$ and $(c, \overline{\sigma}) \neq (c^*, \overline{\sigma}^*)$, the game sets bad to true and responds to the query with $\perp$. 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 $(\overline{vk}^*)$, and starts by selecting $(mpk, msk) \xleftarrow{\$} \mathcal{K}(1^k)$. B runs A with public key (mpk), and simulates A's call for a challenge exactly as specified in G_1, using $\overline{vk}^*$ when called for and its own signing oracle to generate $\overline{\sigma}^*$ on c^*. B also simulates A queries for decryption

exactly as specified in G_1. B notes the first query $\mathcal{O}^{\mathcal{D},\Phi}(\phi, c, \overline{vk}, \overline{\sigma})$ that triggers bad, and 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 B 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 rearranges some of the validity checks the challenger performs before decryption, but but these changes don't affect oracle responses since the same ciphertexts end up being rejected in either game. We have

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

Now we show that an adversary with winning G_2 with significant advantage can be used to break the Φ-RKA security of $\mathcal{IBE}$. We construct A' such that

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

A' takes input (mpk) for the IBE scheme, and runs A with input (mpk) for $\mathcal{PKE}$. A' simulates the challenge that A requests with (m_0, m_1) as follows:

1. A' generates $(\overline{vk}^*, \overline{sk}^*) \xleftarrow{\$} \overline{\mathcal{K}}(1^k)$, and requests its own challenge c^* with $(\overline{vk}^*, m_0, m_1)$.
2. A' signs c^* using $\overline{sk}^*$, generating $\overline{\sigma}^* \leftarrow \mathcal{S}(\overline{sk}^*, c^*)$.
3. A' returns challenge ciphertext $C^* = (c^*, \overline{vk}^*, \overline{\sigma}^*)$ to A.

A' simulates answers to A's queries to $\mathcal{O}^{\mathcal{D},\Phi}(\phi, c, \overline{vk}, \overline{\sigma})$ by computing a tampered key with its key derivation oracle, as $msk' \leftarrow \mathcal{O}^{\mathcal{K},\Phi}(\phi, \overline{vk}^*)$. If $(\overline{\mathcal{V}}(\overline{vk}, c, \overline{\sigma}) = \mathsf{false})$ then A' returns $\perp$, since this is not a valid ciphertext. Otherwise A' computes $M \leftarrow \mathcal{D}(msk', c)$, and returns this to A. A' runs A until it A halts, and then A' outputs whatever A outputs.

To complete the claim we need to argue that A' properly simulates G_2 for A. The only subtlety is in how A' handles decryption queries are handled. But we observe that A''s key derivation oracle is performing *exactly* the same first two checks that G_2 performs before the challenger returns a decryption, and all other ciphertexts are correctly decrypted, and so A' performs as claimed.

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

∎

10 Separations between RKA Sets

After giving several constructions of Φ-RKA secure primitives from other Φ-RKA secure primitives, we will now show that in some cases this is actually impossible. We give negative results of the form $\mathbf{RKA}[\mathsf{P}_1] \not\subseteq \mathbf{RKA}[\mathsf{P}_2]$ by showing there exists a Φ for which $\Phi \in \mathbf{RKA}[\mathsf{P}_1]$ but $\Phi \notin \mathbf{RKA}[\mathsf{P}_1]$. We reach these separations using two different techniques.

One is based on the class Cnst of RKD functions that set keys to a constant . We show, for example, that any signature scheme is Cnst-RKA secure, since an adversary could simulate signatures made with constant keys itself; however, no PRF can be Cnst-RKA secure since a PRF evaluated with a fixed key is easily distinguished from a random function. Thus $\mathbf{RKA}[\mathsf{Sig}] \not\subseteq \mathbf{RKA}[\mathsf{PRF}]$. 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.

The second technique is to define pairs of keys, and define RKD functions that sometimes exchange one key in the pair for another depend on a bit of the original key; these are designed in such a way that there is a noticeable change in behavior for a Φ-RKA attack on P_2, yet the two keys are functionally equivalent under a Φ-RKA on P_1. We use variants of the technique to show $\mathbf{RKA}[\mathsf{wPRF}] \not\subseteq \mathbf{RKA}[\mathsf{PRF}]$, and also $\mathbf{RKA}[\mathsf{SE\text{-}CPA}] \not\subseteq \mathbf{RKA}[\mathsf{SE\text{-}CCA}]$.

10.1 Separating Φ-RKA PRFs from Φ-RKA signatures

Since Corollary 9.3 shows that $\mathbf{RKA}^*[\mathsf{PRF}] \subseteq \mathbf{RKA}^*[\mathsf{Sig}]$, it is natural to consider whether the converse is also true– can one build a Φ-RKA secure PRF from any Φ-RKA secure signature scheme? It turns out the answer is no, which is expressed $\mathbf{RKA}^*[\mathsf{Sig}] \not\subseteq \mathbf{RKA}^*[\mathsf{PRF}]$.

This is proved by showing that there exists a Φ such that there exists a Φ-RKA secure signature scheme, but for which *no* Φ-RKA secure PRF can exist. We do this using Cnst, the class of RKD

functions that set the secret key to a fixed constant.

We first show that all signature schemes are Cnst-RKA secure.

Proposition 10.1 *Let $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ be any signature scheme with standard (non-tampering) security. Then $\mathcal{DS}$ is a* Cnst-*RKA secure signature scheme.*

Proof: We use any adversary A against the Cnst-RKA security of $\mathcal{DS}$ to build an adversary B against the normal security of $\mathcal{DS}$, with $\mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},B,\mathsf{ID}}(k) \geq \mathbf{Adv}^{\text{sig-rka}}_{\mathcal{DS},A,\mathsf{Cnst}}(k)$.

Adversary B receives a verification key vk for $\mathcal{DS}$ from the challenger, and receives access to signing oracle $\mathcal{O}^{\mathcal{S}}$.

For the reduction, B forwards vk to A. A should receives access to the signing oracle $\mathcal{O}^{\mathcal{S},\mathsf{Cnst}}$. B must answer queries of the form $\mathcal{O}^{\mathcal{S},\mathsf{Cnst}}(\phi, m)$.

- If $\phi = \phi_{\mathsf{ID}}$, B returns $\sigma \leftarrow \mathcal{O}^{\mathcal{S}}(m)$ from its own signing oracle.
- Otherwise $\phi = \phi_c$, i.e. the RKD which maps all keys to the constant c. In this case, B first tests for the event that c is the hidden secret key: for an m' that B has not queried its signing oracle, B tests $\mathcal{V}(vk, m', \mathcal{S}(c, m'))$; if true, B outputs $(m', \mathcal{S}(c, m'))$ as its forgery. If c fails to generate a valid forgery in this way, B correctly answers A's oracle query with $\mathcal{S}(c, m)$.

If B has not already output a value, when A submits a potential forgery (m^*, σ^*), B repeats this and outputs (m, σ). A's forgery is valid if it has not queried for a signature of m^* under ϕ_{ID} and has never queried for a signature of m^* under ϕ_{sk}. But if A queried with ϕ_{sk}, B will have the signing key and creates a valid forgery. Otherwise, if A never queried $(m^*, \phi_{\mathsf{ID}})$, B never queried its signing oracle for m^* either, and this will be a valid forgery for B too. Thus B has a valid forgery whenever A does.

∎

Next, we show that no Cnst-RKA secure PRF can exist.

Proposition 10.2 *Let $\mathcal{FF} = (\mathcal{K}, \mathcal{F})$ be any PRF; then $\mathcal{FF}$ is not* Cnst-*RKA secure PRF.*

Proof:

We construct adversary A against the Cnst-RKA security of $\mathcal{FF}$: when A is given oracle access to $\mathcal{O}^{\mathcal{F},\mathsf{Cnst}}$, A queries $\mathcal{O}^{\mathcal{F},\mathsf{Cnst}}(\phi_0, 0)$ (or ϕ_c for any valid secret key c and any valid point in $\mathsf{Dom}(k)$). If $\mathcal{O}^{\mathcal{F},\mathsf{Cnst}}(\phi_0, 0) = \mathcal{F}(0,0)$, A makes guess $b' = 0$ of the hidden bit, and otherwise makes guess $b' = 1$.

In the case that A is playing game PRFRand, $\mathcal{O}^{\mathcal{F},\mathsf{Cnst}}(\phi_0, 0) = \mathcal{F}(0,0)$ will only happen with probability $\frac{1}{\mathsf{Rng}(k)}$, while in game PRFReal, it will happen with probability 1.

Thus $\mathbf{Adv}^{\mathrm{prf}}_{\mathcal{FF},A,\Phi}(k) = 1 - \frac{1}{\mathsf{Rng}(k)}$, and so $\mathcal{FF}$ is not a Cnst-RKA secure PRF.

∎

Corollary 10.3 $\mathbf{RKA}[\mathsf{Sig}] \not\subseteq \mathbf{RKA}[\mathsf{PRF}]$.

Proof: Proof by Propositions 10.1 and 10.2. ∎

10.2 Other Relations Using Cnst

Similarly to Proposition 10.2, no wPRF can be Cnst-RKA secure – even when the adversary can only access random points, the fixed function determined by evaluating the wPRF at a fixed key looks far from random. Additionally, no symmetric encryption scheme can be either Cnst-RKA CPA secure or Cnst-RKA CCA secure – in both cases this would allow the adversary to receive the message m_b encrypted with a fixed key, which it can easily decrypt to determine the challenge bit b. We omit the proofs.

Proposition 10.4 *Let $\mathcal{FF} = (\mathcal{K}, \mathcal{F})$ be any wPRF; then $\mathcal{FF}$ is not* Cnst*-RKA secure wPRF.*

Proposition 10.5 *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be any symmetric encryption scheme; then $\mathcal{SE}$ is neither* Cnst*-RKA CPA secure nor* Cnst*-RKA CCA secure.*

In addition to all signature schemes being Cnst-RKA secure, all PKE schemes and all IBE schemes are both Cnst-RKA secure as well. Here, the adversary only receives access to decryption under the secret key, but could easily simulate decryption under constant keys without oracle access. Again, we omit the proofs.

Proposition 10.6 *Let $\mathcal{PKE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be any public key encryption scheme with standard (non-tampering) security. Then $\mathcal{PKE}$ is a* Cnst-*RKA secure public key encryption scheme.*

Proposition 10.7 *Let $\mathcal{IBE} = (\mathcal{M}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be any identity based encryption scheme with standard (non-tampering) security. Then $\mathcal{IBE}$ is a* Cnst-*RKA secure identity based encryption scheme.*

Together, this yields a whole suite of relations – no primitive that can be shown to be Cnst-RKA secure can be contained in a primitive that can never be Cnst-RKA secure.

This gives that none of Signature, PKE, or IBE schemes can be contained in PRF, wPRF, or SE schemes.

10.3 Separating Φ-RKA secure PRFs from Φ-RKA secure wPRFs

Since wPRFs are a relaxation of PRFs, a family of functions that is a Φ-RKA secure PRF is also a Φ-RKA secure wPRF, and so $\mathbf{RKA}[\mathsf{PRF}] \subseteq \mathbf{RKA}[\mathsf{wPRF}]$. We will show that the converse fails to be true; in our set based notation, this is expressed as $\mathbf{RKA}[\mathsf{wPRF}] \not\subseteq \mathbf{RKA}[\mathsf{PRF}]$. In this case the constant RKD functions do not provide the separation since both primitives are insecure under these functions. Instead, we create RKD functions which help the attacker only if they can obtain different RKD functions evaluated with the same point in the domain of the function family, which happens only with negligible probability in the wPRF game.

Proposition 10.8 $\mathbf{RKA}[\mathsf{wPRF}] \not\subseteq \mathbf{RKA}[\mathsf{PRF}]$.

Proof:

For $K \in \{0,1\}^*$ we let K^- denote the string that is K with the first bit flipped. Let $\phi_i(K)$ return K if $K[i] = 1$ and K^- otherwise. Additionally, we define $\phi_{\mathsf{flip}}(K)$ to always return K. We let Φ be the collection of all ϕ_i, ϕ_{flip}, and ϕ_{ID}.

Let $\mathcal{PRF} = (\overline{\mathcal{K}}, \overline{\mathcal{F}})$ be any PRF that is compatible with Φ with domain given by $\mathsf{Dom}_{\mathcal{PRF}}(k)$ – we show that $\mathcal{PRF}$ is *not* a Φ-RKA secure PRF. Let $\ell(k)$ denote the length of keys for $\mathcal{PRF}$ when the security parameter is k.

We now construct an adversary A against the Φ-RKA security of $\mathcal{PRF}$. A picks x at random from $\mathsf{Dom}_{\mathcal{PRF}}(k)$, and then queries $\mathcal{O}^{\mathcal{PRF},\Phi}$ to compute $y_{\mathsf{id}} \leftarrow \mathcal{O}^{\mathcal{PRF},\Phi}(\phi_{\mathsf{ID}}, x)$. Then for each $1 \leq i \leq \ell(k)$ A queries $y_i \leftarrow \mathcal{O}^{\mathcal{PRF},\Phi}(\phi_i, x)$ and sets $K'[i] = 1$ if $y_i = y_{\mathsf{id}}$ and 0 otherwise. (It is crucial that all queries use the same x.) If $\overline{\mathcal{F}}(K,x) \neq \overline{\mathcal{F}}(K^-,x)$, this will generate $K' = K$, and so A has recovered the key. Once it has the key, it can easily distinguish between game PRFReal and game PRFRand by making a few queries under ϕ_{ID} at random inputs and returning 1 if the results are consistent with K' and 0 otherwise. If $\overline{\mathcal{F}}(K,x) = \overline{\mathcal{F}}(K^-,x)$, meaning the functions under keys K and K^- agree at x, then the attack will instead recover the string $K' = 1^{\ell(k)}$ and not recover the key. We resolve this problem by knowing that ϕ_{flip} in Φ: this, combined with the given Φ-RKA security of PRF, 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.

Let $\mathcal{wPRF} = (\mathcal{K}, \mathcal{F})$ be a normal-secure wPRF and for simplicity assume that keys are random $(k-1)$-bit strings. (Formally, this is assuming that $\mathcal{K}(1^k)$ generates the uniform distribution on $\{0,1\}^{k-1}$ for all $k \in \mathbb{N}$.) We now construct a Φ-RKA secure wPRF $\overline{\mathcal{wPRF}} = (\overline{\mathcal{K}}, \overline{\mathcal{F}})$ based on any normal-secure $\mathcal{wPRF} = (\mathcal{K}, \mathcal{F})$. Key generation $\overline{\mathcal{K}}$ returns a random k-bit string on input 1^k. The evaluation algorithm $\overline{\mathcal{F}}(K,x)$ lets L be the last $(k-1)$-bits of K and returns $\mathcal{F}(L,x)$. As a result, the functions $\overline{\mathcal{F}}(K,\cdot)$ and $\overline{\mathcal{F}}(K^-,\cdot)$ are identical.

We reduce the Φ-RKA security of $\overline{\mathcal{wPRF}}$ to the normal security of $\mathcal{wPRF}$: for any adversary A against the normal security of $\mathcal{wPRF}$, we build an adversary B against the Φ-RKA security of $\overline{\mathcal{wPRF}}$ such that $\mathbf{Adv}^{\text{wprf-rka}}_{\overline{\mathcal{wPRF}},B,\Phi}(k) + \text{neg}(k) \geq \mathbf{Adv}^{\text{wprf-rka}}_{\mathcal{wPRF},A,\mathsf{ID}}(k)$ For B to answer A's oracle queries $\mathcal{O}^{\mathcal{wPRF},\Phi}$, B uses its own oracle access to get $(x,y) \leftarrow \mathcal{O}^{\mathcal{wPRF}}(\,)$, and returns (x,y). We note that for all $\phi \in \Phi$, $\phi(K)$ either returns K or K^-, and $\overline{\mathcal{F}}$ agrees at these two values– thus for any K, $\overline{\mathcal{wPRF}}(\phi(K),\cdot) = \overline{\mathcal{wPRF}}(K,\cdot) = \mathcal{wPRF}(L,\cdot)$, where L is the last $(k-1)$ bits of K – and so if B has access to the real $\mathcal{wPRF}$ in game wPRFReal, it correctly simulates A's queries in the game wPRFReal. If B instead has access to a truly random function in wPRFRand, B receives a truly random value from the range of the wPRF for each unique pair (x,L) of x in the domain and key L, where as A should instead receive separately generated random value for $(x, 0 \,\|\, L)$ and $(x, 1 \,\|\, L)$;

here B correctly simulates responses to A's queries in game wPRFRand *except* when A receives both $(x, 0 \,\|\, L)$ and $(x, 1 \,\|\, L)$.

Since the inputs to a wPRF are random rather than adversary-selected, the probability of a repeated input x in q queries is at most $\frac{q^2}{|\mathsf{Dom}_{\mathcal{wPRF}}(k)|}$, which is negligible since q must be polynomial in k and $|\mathsf{Dom}_{\mathcal{wPRF}}(k)|$ is super-polynomial. Thus $\mathbf{Adv}^{\text{wprf-rka}}_{\overline{\mathcal{wPRF}},B,\Phi}(k) \geq \mathbf{Adv}^{\text{wprf-rka}}_{\mathcal{wPRF},A,\mathsf{ID}}(k) - \text{neg}(k)$, and so $\overline{\mathcal{wPRF}}$ is Φ-RKA secure, since we have assumed that $\mathcal{wPRF}$ is a normal secure wPRF.

∎

10.4 Separating Φ-RKA CPA SE from Φ-RKA CCA SE

We use a similar technique to show that **RKA**[SE-CPA] $\not\subseteq$ **RKA**[SE-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.

Proposition 10.9 **RKA**[SE-CPA] $\not\subseteq$ **RKA**[SE-CCA].

Proof:

For $K \in \{0,1\}^*$ we let K^- denote the string that is K with the first bit flipped. Let $\phi_i(K)$ return K if $K[i] = 1$ and K^- otherwise. We let Φ be the collection of all ϕ_i, and ϕ_{ID}.

First, we show that there exists a Φ-RKA CPA secure symmetric encryption scheme. Let $\overline{\mathcal{SE}} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be any CPA secure symmetric encryption scheme. We build a new $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ as follows:

- $\mathcal{K}(1^k)$ selects a random bit $c \xleftarrow{\$} \{0,1\}$, and returns $c \,\|\, \overline{\mathcal{K}}(1^k)$
- $\mathcal{E}(K, m)$: let $L = K[2, |K|]$, and return $\mathcal{E}(L, m)$
- $\mathcal{D}(K, c)$: let $L = K[2, |K|]$, and return $\mathcal{D}(L, c)$

Note that $\mathcal{E}(K, \cdot)$ is the same as $\mathcal{E}(K^-, \cdot)$, and $\mathcal{D}(K, \cdot)$ is the same as $\mathcal{D}(K^-, \cdot)$.

We reduce the Φ-RKA CPA security of $\mathcal{SE}$ to the normal CPA security of $\overline{\mathcal{SE}}$. An adversary A of $\overline{\mathcal{SE}}$ receives oracle access to $\mathcal{O}^{\overline{\mathcal{E}}}(m_0, m_1)$, and receives encryptions of either m_0 or m_1. To answer oracle queries of adversary B of $\mathcal{O}^{\mathcal{E},\Phi}(\phi_i, m_0, m_1)$, A can correctly simulate the answer by returning $\mathcal{O}^{\mathcal{E}}(m_0, m_1)$. When B correctly guesses the hidden bit, A does also. Thus $\mathbf{Adv}^{\text{se-cc-rka}}_{\overline{\mathcal{SE}},A,\mathsf{ID}}(k) = \mathbf{Adv}^{\text{se-cc-rka}}_{\mathcal{SE},B,\Phi}(k)$, and $\mathcal{SE}$ is Φ-RKA CPA secure.

Despite the fact that we have built a Φ-RKA CPA secure symmetric encryption scheme, no Φ-RKA CCA secure symmetric encryption scheme can exist. The problem here is although $\mathcal{D}(\phi(K), \cdot)$ might be simulatable by some function without tampering allowed as $\mathcal{D}'(K, \cdot)$, the oracle actually depends on the value of the secret key in order to determine what ciphertexts are allowed for decryption, and so $\mathcal{O}^{\mathcal{D},\Phi}$ will not be simulatable by $\mathcal{O}^{\mathcal{D}'}$.

We complete the proof by building an adversary A against the Φ-RKA CCA security of any encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. To begin, A generates $C^* \leftarrow \mathcal{O}^{\mathcal{E},\Phi}(\phi_{\mathsf{ID}}, m_0, m_1)$. Note that C^* is generated with $\phi_{\mathsf{ID}}(K)$ the original secret key. Then for each $1 \leq i \leq |K|$ A queries $y_i \leftarrow \mathcal{O}^{\mathcal{D},\Phi}(\phi_i, C^*)$ and sets $K'[i] = 1$ if $y_i = \perp$ and 0 otherwise. Note that $\mathcal{O}^{\mathcal{D},\Phi}(\phi_i, C^*)$ will only return $\perp$ if $\phi_i(K) = K$, which occurs only if the i-th bit of K is 1. Once A is done, $K' = K$ and A can decrypt C^* as $\mathcal{D}(K', C^*)$, and will always correctly guess b.

∎

References

[AHI11] Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In *ICS*, pages 45–60, 2011.

[AK96] Ross Anderson and Markus Kuhn. Tamper resistance: a cautionary note. In *Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, pages 1–1, Berkeley, CA, USA, 1996. USENIX Association.

[AK97] Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In *Security Protocols Workshop*, pages 125–136, 1997.

[AN95] Ross J. Anderson and Roger M. Needham. Programming satan's computer. In Jan van Leeuwen, editor, *Computer Science Today*, volume 1000 of *Lecture Notes in Computer Science*, pages 426–440. Springer, 1995.

[BC10] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In *CRYPTO*, pages 666–684, 2010.

[BDL01] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptology*, 14(2):101–119, 2001.

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[Bih93] Eli Biham. New types of cryptoanalytic attacks using related keys (extended abstract). In *EUROCRYPT*, pages 398–409, 1993.

[BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *EUROCRYPT*, pages 491–506, 2003.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[Bov] E Bovenlander. Invited talk on smartcard security, eurocrypt 1997.

[BS90] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In *CRYPTO*, pages 2–21, 1990.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *CRYPTO*, pages 513–525, 1997.

[CPGR05] Jim Chow, Ben Pfaff, Tal Garfinkel, and Mendel Rosenblum. Shredding your garbage: Reducing data lifetime. In *Proc. 14th USENIX Security Symposium*, August 2005.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography, 1976.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, pages 434–452, 2010.

[DR02] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES — the Advanced Encryption Standard.* Springer-Verlag, 2002.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GL10] David Goldenberg and Moses Liskov. On related-secret pseudorandomness. In *TCC*, pages 255–272, 2010.

[GLM+04] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *TCC*, pages 258–277, 2004.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.

[GOR11] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In *TCC*, pages 182–200, 2011.

[Gut96] Peter Gutmann. Secure deletion of data from magnetic and solid-state memory. In *Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography - Volume 6*, SSYM'96, pages 8–8, Berkeley, CA, USA, 1996. USENIX Association.

[HS09] Nadia Heninger and Hovav Shacham. Reconstructing rsa private keys from random key bits. In *CRYPTO*, pages 1–17, 2009.

[HSH+08] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Cal, Ariel J. Feldman, and Edward W. Felten. Least we remember: Cold boot attacks on encryption keys. In *In USENIX Security Symposium*, 2008.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently, 2003.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits ii: Keeping secrets in tamperable circuits. In *EUROCRYPT*, pages 308–327, 2006.

[KK99] Oliver Kömmerling and Markus G. Kuhn. Design principles for tamper-resistant smartcard processors. In *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, WOST'99, pages 2–2, Berkeley, CA, USA, 1999. USENIX Association.

[KKS11] Yael Tauman Kalai, Bhavana Kanukurthi, and Amit Sahai. Cryptography with tamperable and leaky memory. In *CRYPTO*, pages 373–390, 2011.

[LL10] Feng-Hao Liu and Anna Lysyanskaya. Algorithmic tamper-proof security under probing attacks. In *Proceedings of the 7th international conference on Security and cryptography for networks*, SCN'10, pages 106–120, Berlin, Heidelberg, 2010. Springer-Verlag.

[Luc04] Stefan Lucks. Ciphers secure against related-key attacks. In *FSE*, pages 359–370, 2004.

[SB94] H. P. Sharangpani and M. L. Barton. Statistical Analysis of Floating Point Flaw in the Pentium Processor. Technical report, Intel, 11 1994.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[Sko02] Sergei Skorobogatov. Low temperature data remanence in static RAM. Technical Report UCAM-CL-TR-536, University of Cambridge, Computer Laboratory, June 2002.

[SPW99] Sean W. Smith, Elaine R. Palmer, and Steve Weingart. Building a high-performance, programmable secure coprocessor. In *Computer Networks*, pages 831–860, 1999.