UNIVERSITY OF CALIFORNIA, SAN DIEGO

Flexible Models for Secure Systems

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy

in

Computer Science

by

Sarah Meiklejohn

Committee in charge:

Professor Mihir Bellare, Co-Chair
Professor Stefan Savage, Co-Chair
Professor Sam Buss
Professor Massimo Franceschetti
Professor Daniele Micciancio

2014

The Dissertation of Sarah Meiklejohn is approved and is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Co-Chair

_____
Co-Chair

University of California, San Diego

2014

EPIGRAPH

Computers are useless. They can only give you answers.

Pablo Picasso

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

To begin with the most obvious, I would like to thank my parents and my brother for continually pushing me to do better and for their unwavering support in all my endeavors.

Academically, I feel indebted to all of my advisors (both official and unofficial): Anna Lysyanskaya, Hovav Shacham, Stefan Savage, Mihir Bellare, and Geoff Voelker. Please know that none of this would have been possible without you, and that I am deeply grateful for the unique support that each of you provided.

I would also like to thank the many other members of the UCSD faculty and administrative staff who helped me out throughout the years, my peers who have made it down this long road with me, and the many supportive members of the crypto and security communities who have endured long conversations at conferences and beyond.

I would also like to thank Melissa Chase and the other members of the MSR Redmond crypto group, who have provided me with a welcome home for the summer and proved to be great friends and collaborators.

I would also like to thank all my other co-authors, of whom I have been lucky to have many: Theresa Calderon, Steve Checkoway, Vacha Dave, Hitesh Dharmdasani, Chris Erway, David Freeman, Chris Grier, Thea Hinkle, Danny Huang, Grant Jordan, Ryan Kastner, Markulf Kohlweiss, Alptekin Küpçü, Kirill Levchenko, Allison Lewko, Damon McCoy, Keaton Mowery, Jason Oberg, Marjori Pomarole, Tim Sherwood, Alex Snoeren, Susan Thomson, Brent Waters, Nick Weaver, and Greg Zaverucha. Thanks for putting up with me!

None of this happened in a bubble, so I would like to thank the employees and other regulars of the cafes, bars, restaurants, yoga studios, and printmaking shops I have frequented, who provided a necessary and invigorating change of pace.

Finally, thank you of course to my family, friends, and companions throughout the years: you have all somehow made this worthwhile.

Chapter 2, in part, is a reprint of the material as it appears in *Advances in Cryptology - EUROCRYPT '14*. Mihir Bellare, Sarah Meiklejohn, Susan Thomson, Springer Lecture Notes in Computer Science, volume 8441, May 2014. The dissertation author was a primary investigator and author of this paper.

Chapter 3, in part, is a reprint of the material as it appears in *Advances in Cryptology - EUROCRYPT '14*. Mihir Bellare, Sarah Meiklejohn, Susan Thomson, Springer Lecture Notes in Computer Science, volume 8441, May 2014. The dissertation author was a primary investigator and author of this paper.

Chapter 4, in part, is a reprint of the material as it appears in *Proceedings of the 13th ACM SIGCOMM Conference on Internet Measurement*. Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage, ACM, November 2013. The dissertation author was the primary investigator and author of this paper.

VITA

2008	Sc.B. in Mathematics, Brown University
2009	Sc.M. in Computer Science, Brown University
2014	Ph.D. in Computer Science, University of California, San Diego

ABSTRACT OF THE DISSERTATION

Flexible Models for Secure Systems

by

Sarah Meiklejohn

Doctor of Philosophy in Computer Science

University of California, San Diego, 2014

Professor Mihir Bellare, Co-Chair
Professor Stefan Savage, Co-Chair

Modern computing and interactions have become increasingly complex over the last decade, resulting in an online ecosystem with many more options for users, but less transparent information about their security and, in particular, their privacy.

The resulting gap between security and functionality has given rise to various problems and concerns. While these problems — e.g., the spread of malware, data breaches on (supposedly) secure servers, mining of private user data on social networks — might seem quite diverse, many of them revolve around the broader issue of what happens to systems when they get deployed. Systems that may have seemed fully secure

in development often fail to fulfill these security goals in the real world, as adversaries may have capabilities that were not taken into account when the system was designed, and even honest users may interact with the system in unexpected ways.

In this dissertation, I describe two areas in which this gap between the abstract protocol and the deployed system leads to security concerns that ultimately impact every user of the system. The first area considers what happens when adversaries have unexpected capabilities; in particular, we examine how to model sophisticated attacks on the security of a system, such as side channels and fault injection, and then how to design flexible cryptosystems that can tolerate such attacks.

The second area considers the more benign scenario in which users, purely by virtue of their own decisions, may unknowingly cede some of their own privacy. In particular, we examine user anonymity in the Bitcoin network, which is a purely virtual currency that acts as an electronic version of cash. By combining publicly available information with minimal data gathered by hand, we find that an average Bitcoin user is experiencing a fairly low level of anonymity, making Bitcoin ultimately unattractive for criminal activity such as money laundering.

Chapter 1

Introduction

In the early days of the Internet, its functionality was relatively straightforward: essentially, users — like the canonical Alice and Bob — could either visit static webpages or send each other messages. In many cases, Alice wanted to ensure that the messages she received really were from Bob, which established a need to provide some sort of integrity; i.e., for Bob to be able to authenticate himself to Alice. Especially once users began making purchases online, and thus entering sensitive information like credit card numbers into forms, the need to protect the contents of these potentially sensitive messages was established as well.

These dual motivations of authenticity and confidentiality — in what follows, referred to jointly as *secure communication* — have now inspired decades-long lines of research in the areas of digital signatures [115, 100, 67, 82, 105, 126, 39, 46] and encryption [118, 81, 33, 67, 53, 108, 111, 55], as well as the development of SSL/TLS [61], the standardization of signature schemes such as ECDSA [127], symmetric encryption schemes such as AES [2], and public-key encryption schemes such as RSA [118], and the eventual deployment and widespread adoption of libraries such as OpenSSL. These existing solutions have proved highly effective at providing secure communication, and now help form the backbone of security on the Internet. In the ensuing decades, however, online functionality has exploded far beyond the single simple scenario of secure

communication. For example, users now broadcast information on a variety of social networks such as Facebook and Twitter; upload content (e.g., movies, pictures, executables) to and download content from peer-to-peer networks such as BitTorrent; and use payment systems such as Square and Paypal to send money to both businesses and other individuals. To complicate matters further, even secure communication is not as simple as it once was: communication is now carried out over a variety of networks (e.g., WiFi) using a variety of constrained devices (e.g., smartphones), and cryptographic keys are stored on a variety of hardware (e.g., a general-purpose chip).

All of these diverse forms of online interaction now account for the vast majority of traffic on the Internet — a 2013 report by Sandvine [123] shows that 31% of all downstream traffic in North America is due to Netflix alone, compared to 10% for HTTP (regular web browsing) and 2% for SSL (secure communication) — and result in a rich ecosystem in which the security concerns of Alice and Bob go far beyond the simple scenario of malicious eavesdroppers learning the messages they send each other.

Unfortunately, the development of secure solutions has not caught up with this expanded functionality, which has led to a significant gap between the many ways in which users engage online and the tools that they can use to maintain security and privacy. (For the most part, this gap extends even beyond secure tools to a basic understanding of how different activities compromise user security.) Concretely, there have been many examples of how even secure-seeming tools fail to provide protection against motivated adversaries. Just to name a few prominent examples, studies have shown that encrypted VoIP (i.e., the technology behind systems such as Skype) can be broken [133, 134]; browser security warnings are often not as effective as intended [65, 125, 60, 3]; the Tor network (designed to anonymize Web traffic) has limitations to the anonymity it provides [110, 31, 92, 107, 28]; and even the technologies underlying basic secure communication are not as safe as one might hope [131, 48, 93, 112, 4].

Given this current atmosphere, one might naturally wonder what has caused this rift between security and functionality, and moreover what can be done to fix it. In some cases, the reason for the lack of security is fairly obvious: a service might have ignored or poorly implemented an important security feature, or might have enabled an attacker to hack into their server — thereby contributing to the increasingly common issue of data breaches — by failing to implement sufficient security measures.

The larger issue, however, is more subtle than simply blaming an implementation or system design. In designing a complex system, it is difficult to anticipate the myriad ways that users — both honest and malicious — will interact with the system, or how the decision of some users will affect the security of others. Furthermore, the capabilities of adversaries are continually evolving, so it is difficult to predict how a system deployed today will perform with respect to an attack carried out a decade or more in the future.

1.1 Scope and Purpose

In this dissertation, we examine two methods for identifying ways in which secure systems — even ones with good abstract designs — fail to provide the intended security in practice. The first method focuses on identifying and actively preventing certain adversarial behavior, while the second method takes a more passive approach in identifying common patterns among honest users and the ways that these patterns affect their security (and the security of their fellow users).

To try to actively prevent certain attacks or adversarial behavior, we adopt an approach commonly used in theoretical cryptography. Briefly, the cryptographic approach to security takes a system or desired functionality (e.g., digital signatures) and models it formally as a collection of algorithms. Then, within this formal model, one can mathematically prove that an adversary — also modeled as an algorithm — attempting to execute a given attack against the system (e.g., producing forged signatures) can

successfully do so only if it takes a certain number of steps (e.g., if the adversary must take 2^{80} steps, the system achieves 80-bit security). If the system and corresponding attack are correctly modeled, then these solutions can provide long-lasting security.

The caveat, however, is that often these formal models make simplifying assumptions or work at a fairly high level of abstraction, and in doing so fail to capture real-world adversarial capabilities. For example, the standard model for digital signatures assumes that the signing key is used only to create signatures (so in particular is not being used for any other application), and that the signing key is stored on a device that is completely tamper-resilient and produces no side channels. In practical settings where these assumptions are violated — e.g., when the same RSA key is reused for both decrypting ciphertexts and signing messages, in the EMV protocol [68, 69, 70, 71], when keys are stored on devices that don't sufficiently prevent tampering [6, 91, 37, 129] or produce side channels [95, 96, 41, 83, 117] — this standard model thus provides no security whatsoever; i.e., a signature scheme with a proof of security might nevertheless be completely insecure in practice.

To complement the cryptographic approach, is it therefore important to study not only the formal models for secure systems, but also the ways in which such systems are actually used in practice. Studying adversarial capabilities such as fault injection and tampering can inform system designers on how to strengthen or otherwise improve existing cryptographic models, but studying the behavior of honest users helps to investigate an arguably more subtle point: what security do users provide (or fail to provide) for themselves, even in the absence of powerful adversaries?

The most effective method for studying the behavior of honest users is to simply measure it directly. In cases where this is not possible — often because information about user behavior is not public — one can nevertheless glean some information by conducting user studies. Again, just to cite a few examples, researchers have used

access to social network data to examine general network dynamics [135] and privacy implications [98, 14]; conducted user studies (most notably using Amazon's Mechanical Turk) to better understand the security of technologies such as non-password-based authentication [136, 124, 40]; and — focusing again on basic secure communication — have performed comprehensive Internet-wide scans to examine the vulnerabilities of TLS certificates [89, 101, 64].

1.2 Organization

The rest of the dissertation proceeds as follows. In Chapter 2, we first present some preliminary definitions and concepts as they pertain to the rest of the dissertation. In particular, we introduce the notions of digital signatures and non-interactive zero-knowledge proofs.

In Chapter 3, we adopt the cryptographic approach to security and examine certain underlying assumptions in the formal models for digital signatures. In particular, we revisit the standard model for digital signatures, and examine the two problematic settings described above; i.e., the settings wherein the same key is used to encrypt and sign [87, 113], and the key is stored on potentially non-tamper-resilient devices. We additionally consider a third setting, in which the signing key might be used to create signatures on functions of the signing key itself. (This last setting turns out to be useful when the signature is used to add integrity to encryption schemes secure with respect to key-dependent messages [30].) We first present a general solution called key-versatile signatures that provides a way to achieve provable security in each of these settings — and in particular a way that preserves existing security and does not add extra key material — and then discuss the concrete implications for each of them. The work presented in this chapter originally appeared in [21].

In Chapter 4, we then adopt the measurement approach, and look in particular at the way that users engage with the Bitcoin protocol [109]. Bitcoin is a purely digital, decentralized currency that was first deployed on January 3 2009, and has since enjoyed enormous success — as of this writing one bitcoin was worth over 500 USD — along with significant notoriety: bitcoins are routinely stolen from users (both in the form of direct theft, but also Ponzi schemes [128] and other scams), and are used to pay a variety of criminals ranging from ransomware operators [130] to drug dealers [84]. We find that certain standard patterns — that we call *idioms of use*, as they are not inherent to the Bitcoin protocol — significantly erode the anonymity of Bitcoin users, to the point where it is feasible to track flows of bitcoins throughout the network. We conclude that users are not achieving the level of anonymity they might hope for, and thus adversaries are unlikely to find Bitcoin particularly attractive for criminal purposes such as money laundering. The work presented in this chapter originally appeared in [104].

Finally, in Chapter 5, we conclude the dissertation and consider how these two distinct approaches to security can be extended and used in a complementary manner.

Chapter 2

Background and Notation

In this chapter, we define some terminology and cryptographic primitives — namely digital signatures and non-interactive zero-knowledge proofs — that will be used throughout the rest of the dissertation. We first introduce some basic notation.

The empty string is denoted by ε. If x is a (binary) string then $|x|$ is its length. If S is a finite set then $|S|$ denotes its size and $s \xleftarrow{\$} S$ denotes picking an element uniformly from S and assigning it to s. We denote by $\lambda \in \mathbb{N}$ the security parameter and by 1^λ its unary representation.

Algorithms are randomized unless otherwise indicated. For both randomized and deterministic algorithms, "PT" stands for "polynomial-time." By $y \leftarrow A(x_1, \ldots; R)$, we denote the operation of running algorithm A on inputs $x_1, \ldots$ and coins R and letting y denote the output. By $y \xleftarrow{\$} A(x_1, \ldots)$, we denote the operation of letting $y \leftarrow A(x_1, \ldots; R)$ for random R. We denote by $[A(x_1, \ldots)]$ the set of points that have positive probability of being output by A on inputs $x_1, \ldots$. Adversaries are algorithms.

Game playing framework. We use games in definitions of security and in proofs. A game G (e.g., Figure 3.1) has a MAIN procedure whose output (what it returns) is the output of the game. We let Pr[G] denote the probability that this output is the boolean true. The boolean flag bad, if used in a game, is assumed initialized to false.

2.1 Digital Signatures

Chapter 3 and Chapter 4 deal closely with the notion of a *digital signature scheme*. In Chapter 3, we consider general settings in which the security of the signature scheme is affected by the context in which it is used, and in Chapter 4 we consider signatures as an integral component of the Bitcoin protocol design.

Informally, a digital signature scheme is — as discussed in the introduction — used to provide integrity or authentication; in this way it can be thought of as the digital analogue to a regular signature. Formally, a digital signature scheme DS specifies the following PT algorithms: via $pp \leftarrow\!\!\$\, \mathsf{DS.Pg}(1^\lambda)$ one generates public parameters pp common to all users; via $(sk,pk) \leftarrow\!\!\$\, \mathsf{DS.Kg}(1^\lambda, pp)$ a user can generate a secret signing key sk and corresponding public verification key pk; via $\sigma \leftarrow\!\!\$\, \mathsf{DS.Sig}(1^\lambda, pp, sk, M)$ the signer can generate a signature σ on a message $M \in \{0,1\}^*$; via $d \leftarrow \mathsf{DS.Ver}(1^\lambda, pp, pk, M, \sigma)$ a verifier can deterministically produce a decision $d \in \{\mathsf{true}, \mathsf{false}\}$ regarding whether σ is a valid signature of M under pk. We say that DS is correct if $\mathsf{DS.Ver}(1^\lambda, pp, pk, M, \mathsf{DS.Sig}(1^\lambda, pp, sk, M)) = \mathsf{true}$ for all $\lambda \in \mathbb{N}$, $pp \in [\mathsf{DS.Pg}(1^\lambda)]$, $(sk,pk) \in [\mathsf{DS.Kg}(1^\lambda, pp)]$, and all M. We say that DS is unforgeable if $\mathbf{Adv}^{\mathrm{uf}}_A(\lambda)$ is negligible for every PT adversary A, where $\mathbf{Adv}^{\mathrm{uf}}_A(\lambda) = \Pr[\mathrm{UF}^A_{\mathsf{DS}}(\lambda)]$ and game UF is specified on the left-hand side of Figure 2.1. We say that DS is strongly unforgeable if $\mathbf{Adv}^{\mathrm{suf}}_A(\lambda)$ is negligible for every PT adversary A, where $\mathbf{Adv}^{\mathrm{suf}}_A(\lambda) = \Pr[\mathrm{SUF}^A_{\mathsf{DS}}(\lambda)]$ and game SUF is specified on the right-hand side of Figure 2.1. These definitions are based on [82].

2.2 Non-Interactive Zero-Knowledge Proofs

In Chapter 3, we build a digital signature scheme from a non-interactive zero-knowledge (NIZK) proof. Informally, a zero-knowledge proof consists of an interaction between two parties, a prover and a verifier, in which the prover is trying to convince

MAIN $\mathrm{UF}^{A}_{\mathsf{DS}}(\lambda)$	MAIN $\mathrm{SUF}^{A}_{\mathsf{DS}}(\lambda)$
$Q \leftarrow \emptyset$; $pp \leftarrow\!\!\$\ \mathsf{DS.Pg}(1^\lambda)$	$Q \leftarrow \emptyset$; $pp \leftarrow\!\!\$\ \mathsf{DS.Pg}(1^\lambda)$
$(sk,pk) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)$	$(sk,pk) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)$
$(M,\sigma) \leftarrow\!\!\$\ A^{\textsc{Sign}}(1^\lambda, pp, pk)$	$(M,\sigma) \leftarrow\!\!\$\ A^{\textsc{Sign}}(1^\lambda, pp, pk)$
Ret ($\mathsf{DS.Ver}(1^\lambda, pp, pk, M, \sigma)$ and $M \notin Q$)	Ret ($\mathsf{DS.Ver}(1^\lambda, pp, pk, M, \sigma)$ and $(M,\sigma) \notin Q$)
$\textsc{Sign}(M)$	$\textsc{Sign}(M)$
$\sigma \leftarrow\!\!\$\ \mathsf{DS.Sig}(1^\lambda, pp, sk, M)$	$\sigma \leftarrow\!\!\$\ \mathsf{DS.Sig}(1^\lambda, pp, sk, M)$
$Q \leftarrow Q \cup \{M\}$	$Q \leftarrow Q \cup \{(M,\sigma)\}$
Ret σ	Ret σ

Figure 2.1. Games defining security of signature scheme DS. Left: Game defining unforgeability. Right: Game defining strong unforgeability.

the verifier that a certain statement is true. If at the end of this interaction, the verifier learns no information beyond the validity of the statement, then the proof is said to be zero knowledge. If the interaction furthermore consists of only a single message sent from the prover to the verifier, then the proof is said to be non-interactive.

Formally, suppose R: $\{0,1\}^* \times \{0,1\}^* \to \{\mathsf{true},\mathsf{false}\}$. For $x \in \{0,1\}^*$ we let $\mathsf{R}(x) = \{\, w : \mathsf{R}(x,w) = \mathsf{true} \,\}$ be the *witness set* of x. We say that R is an **NP**-relation if it is computable in time polynomial in the length of its first input and there is a function ℓ such that $\mathsf{R}(x) \subseteq \{0,1\}^{\ell(|x|)}$ for all $x \in \{0,1\}^*$. We let $L(\mathsf{R}) = \{\, x : \mathsf{R}(x) \neq \emptyset \,\}$ be the *language* associated to R. The fact that R is an **NP**-relation means that $L(\mathsf{R}) \in \mathbf{NP}$.

A non-interactive (NI) system Π for R specifies the following PT algorithms: via $crs \leftarrow\!\!\$\ \Pi.\mathsf{Pg}(1^\lambda)$ one generates a common reference string crs; via $\pi \leftarrow\!\!\$\ \Pi.\mathsf{P}(1^\lambda, crs, x, w)$ the prover given x and $w \in \mathsf{R}(x)$ generates a proof π that $x \in L(\mathsf{R})$; via $d \leftarrow \Pi.\mathsf{V}(1^\lambda, crs, x, \pi)$ a verifier can produce a decision $d \in \{\mathsf{true},\mathsf{false}\}$ regarding whether π is a valid proof that $x \in L(\mathsf{R})$. We require completeness, namely $\Pi.\mathsf{V}(1^\lambda, crs, x, \Pi.\mathsf{P}(1^\lambda, crs, x, w)) = \mathsf{true}$ for all $\lambda \in \mathbb{N}$, all $crs \in [\Pi.\mathsf{Pg}(\lambda)]$, all $x \in \{0,1\}^*$ and all $w \in \mathsf{R}(x)$. We say that Π is zero-knowledge (ZK) if it specifies additional PT algorithms $\Pi.\mathsf{SimPg}$ and $\Pi.\mathsf{SimP}$

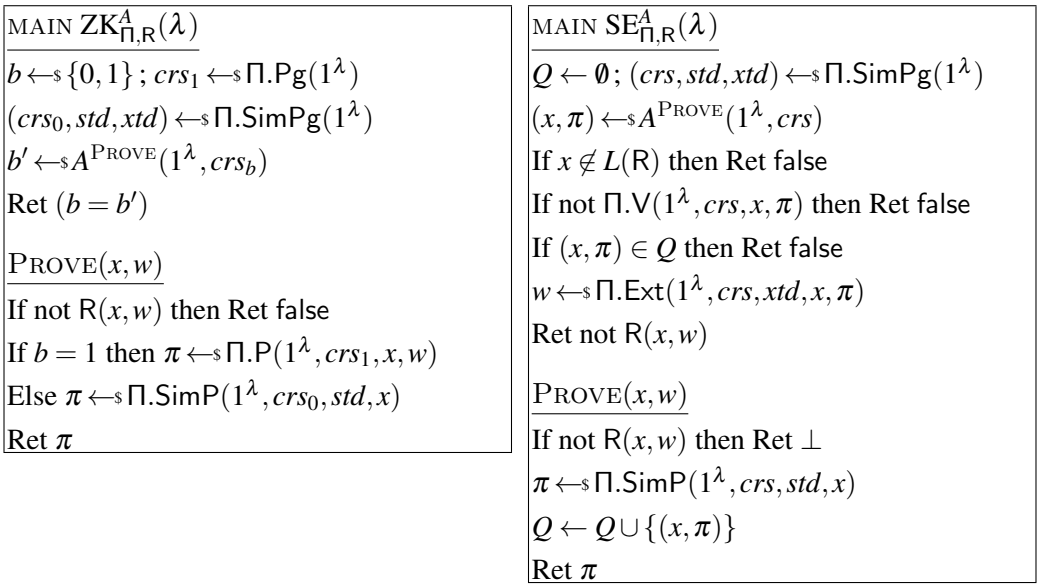


Figure 2.2. Games defining security of NIZK system Π. Left: Game defining zero knowledge. Right: Game defining simulation extractability.

such that $\mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{R},A}(\cdot)$ is negligible for every PT adversary A, where $\mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{R},A}(\lambda) = 2\Pr[\mathrm{ZK}^A_{\Pi,\mathsf{R}}(\lambda)] - 1$ and game ZK is specified on the left-hand side of Figure 2.2. This definition is based on [32, 56]. We say that Π is simulation-extractable (SE) if it specifies an additional PT algorithm $\Pi.\mathsf{Ext}$ such that $\mathbf{Adv}^{\mathrm{se}}_{\Pi,\mathsf{R},A}(\cdot)$ is negligible for every PT adversary A, where $\mathbf{Adv}^{\mathrm{se}}_{\Pi,\mathsf{R},A}(\lambda) = \Pr[\mathrm{SE}^A_{\Pi,\mathsf{R}}(\lambda)]$ and game SE is specified on the right-hand side of Figure 2.2. This definition is based on [56, 85, 86, 62].

The first construction of SE NIZKs (using a stronger notion of simulation extractability) was given in [85], but for a fairly restricted language related to sets of pairing product equations in bilinear groups. In [62] (and further formalized in [88]), the authors provide a generic construction of SE NIZKs from a (regular) NIZK, an IND-CCA encryption scheme, and a one-time signature, which establishes that SE NIZKs exist for all **NP**.

Acknowledgments

Chapter 2, in part, is a reprint of the material as it appears in *Advances in Cryptology - EUROCRYPT '14*. Mihir Bellare, Sarah Meiklejohn, Susan Thomson, Springer Lecture Notes in Computer Science, volume 8441, May 2014. The dissertation author was a primary investigator and author of this paper.

Chapter 3

Key-Versatile Signatures

One of the recommended principles of sound cryptographic design is key separation, meaning that keys used for one purpose (e.g., encryption) should not be used for another purpose (e.g., signing). The reason is that, even if the individual uses are secure, the joint usage could be insecure [59]. On the other hand, there are important applications where key reuse is not only desirable but crucial to maintain security, and that when done "right" it works. We offer key-versatile signatures as a general tool to enable signing with existing keys already in use for another purpose, without adding key material but maintaining security of both the new and the old usage of the keys. Our applications include: (1) adding signing capability to existing encryption capability with zero overhead in the size of the public key; (2) obtaining signatures secure against related-keys attacks (RKA-secure signatures) from RKA-secure one-way functions; and (3) adding integrity to encryption while preserving key-dependent message (KDM) security.

A closer look. Key-versatility refers to the ability to take an arbitrary one-way function F and return a signature scheme where the secret signing key is a random domain point x for F and the public verification key is its image $y = F(x)$. By requiring strong simulatability and key-extractability security conditions [51] from these "F-keyed" signatures,

and then defining F based on keys already existing for another purpose, we will be able to add signing capability while maintaining existing keys and security.

The most compelling motivation comes from security against related-key attack (RKA) and security for key-dependent messages (KDM), technically challenging areas where solutions create, and depend on, very specific key structures. We would like to expand the set of primitives for which we can provide these forms of security. Rather than start from scratch, we would like to leverage the existing, hard-won advances in these areas by modular design, transforming a primitive **X** into a primitive **Y** while preserving RKA or KDM security. Since security is relative to a set of functions (either key or message deriving) on the space of keys, the transform must preserve the existing keys. Key-versatile signatures will thus allow us to create new RKA and KDM secure primitives in a modular way.

We warn that our results are theoretical feasibility ones. They demonstrate that certain practical goals can in principle be reached, but the solutions are not efficient. Below we begin with a more direct application of key-versatile signatures — to joining signatures to an existing encryption capability without expanding the size of the public key — and then go on to our RKA and KDM results.

Joining signatures to encryption with zero public-key overhead. Suppose Alice has keys (sk_e, pk_e) for a public-key encryption scheme and wants to also have signing capability. Certainly, she could pick new and separate keys (sk_s, pk_s) for her favorite signature scheme and use those. This approach means, however, that Alice's public key, now $pk = (pk_e, pk_s)$, has doubled in size. Practitioners ask if one can do better. We want a joint encryption and signature (JES) scheme [87, 113], where there is a single key-pair (sk, pk) used for both encryption and signing. We aim to minimize the public-key

overhead, (loosely) defined as the size of pk minus the size of the public key pk_e of the underlying encryption scheme.

Haber and Pinkas [87] initiated an investigation of JES. They note that the key re-use requires defining and achieving new notions of security particular to JES: signatures should remain unforgeable even in the presence of a decryption oracle, and encryption should retain IND-CCA privacy even in the presence of a signing oracle. In the random oracle model [24], *specific* IND-CCA-secure public-key encryption schemes have been presented where signing can be added with no public-key overhead [87, 52, 97]. In the standard model, encryption schemes have been presented that allow signing with a public-key overhead lower than that of the "Cartesian product" solution of just adding a separate signing key [87, 113], with the best results, from [113], using IBE or combining encryption and signature schemes of [42, 35].

All these results, however, pertain to *specific* encryption schemes. We step back to ask a general theoretical question. Namely, suppose we are given an *arbitrary* IND-CCA-secure public-key encryption scheme. We wish to add signing capability to form a JES scheme. How low can the public-key overhead go? The (perhaps surprising) answer we provide is that we can achieve a public-key overhead of *zero*. The public key for our JES scheme remains *exactly* that of the given encryption scheme, meaning we add signing capability without changing the public key.[1] We emphasize again that this is for *any* starting encryption scheme.

To do this, we let F be the function that maps the secret key of the given encryption scheme to the public key.[2] The assumed security of the encryption scheme means this

[1] Zero public-key overhead has a particular advantage besides space savings, namely that, in adding signing, no new certificates are needed. This makes key management significantly easier for the potentially large number of entities already using Alice's public key. This advantage is absent if the public key is at all modified.

[2] Not all encryption schemes will directly derive the public key as a deterministic function of the secret key, although many, including Cramer-Shoup [54], do. However, we can modify any encryption scheme to

function is one-way. Now, we simply use an F-keyed signature scheme, with the keys remaining those of the encryption scheme. No new keys are introduced. We need however to ensure that the joint use of the keys does not result in bad interactions that make either the encryption or the signature insecure. This amounts to showing that the JES security conditions, namely that encryption remains secure even given a signing oracle and signing remains secure even given a decryption oracle, are met. This will follow from the simulatability and key-extractability requirements we impose on our F-keyed signatures. See Section 3.3.

New RKA-secure signatures. In a related-key attack (RKA) [94, 26, 20, 16] an adversary can modify a stored secret key and observe outcomes of the cryptographic primitive under the modified key. Such attacks may be mounted by tampering [37, 27, 78], so RKA security improves resistance to side-channel attacks. Achieving proven security against RKAs, however, is broadly recognized as very challenging. This has lead several authors [79, 16] to suggest that we "bootstrap," building higher-level Φ-RKA-secure[3] primitives from lower-level Φ-RKA-secure primitives. In this vein, [16] show how to build Φ-RKA signatures from Φ-RKA PRFs. Building Φ-RKA PRFs remains difficult, however, and we really have only one construction [15]. This has lead to direct (non-bootstrapping) constructions of Φ-RKA signatures for classes Φ of polynomials over certain specific pairing groups [23].

We return to bootstrapping and provide a much stronger result, building Φ-RKA signatures from Φ-RKA one-way functions rather than from Φ-RKA PRFs.[4] The

have this property, *without changing the public key*, by using the coins of the key-generation algorithm as the secret key.

[3] As per the framework of [20, 16], security is parameterized by the class of functions Φ that the adversary is allowed to apply to the key. Security is never possible for the class of all functions [20], so we seek results for specific Φ.

[4] For a one-way function, the input is the "key." In attempting to recover x from $F(x)$, the adversary may also obtain $F(x')$ where x' is created by applying to x some modification function from Φ. The definition is from [79].

difference is significant because building Φ-RKA one-way functions under standard assumptions is easy. Adapting the key-malleability technique of [15], we show that many natural one-way functions are Φ-RKA secure *assuming nothing more than their standard one-wayness*. In particular this is true for discrete exponentiation over an arbitrary group and for the one-way functions underlying the LWE and LPN problems. In this way we obtain Φ-RKA signatures for many new and natural classes Φ.

The central challenge in our bootstrapping is to preserve the keyspace, meaning that the space of secret keys of the constructed signature scheme must be the domain of the given Φ-RKA one-way function F. (Without this, it is not even meaningful to talk of preserving Φ-RKA security, let alone to show that it happens.) This is exactly what an F-keyed signature scheme allows us to do. The proof that Φ-RKA security is preserved exploits strong features built into our definitions of simulatability and key-extractability for F-keyed signatures, in particular that these conditions hold even under secret keys selected by the adversary. See Section 3.2.

KDM-secure storage. Over the last few years we have seen a large number of sophisticated schemes to address the (challenging) problem of encryption of key-dependent data (e.g., [30, 38, 11, 10, 45, 47, 29, 13, 102, 9, 43, 44, 19, 76, 90]). The most touted application is secure outsourced storage, where Alice's decryption key, or some function thereof, is in a file she is encrypting and uploading to the cloud. But in this setting integrity is just as important as privacy. To this end, we would like to add signatures, thus enabling the server, based on Alice's public key, to validate her uploads, and enabling Alice herself to validate her downloads, all *while preserving KDM security.*

What emerges is a new goal that we call KDM-secure (encrypted and authenticated) storage. In Section 3.4 we formalize the corresponding primitive, providing both syntax and notions of security for key-dependent messages. Briefly, Alice uses a secret

key *sk* to turn her message *M* into an encrypted and authenticated "data" object that she stores on the server. The server is able to check integrity based on Alice's public key. When Alice retrieves data, she can check integrity and decrypt based on her secret key. Security requires both privacy and integrity even when *M* depends on *sk*. (As we explain in more depth below, this goal is different from signcryption [137], authenticated public-key encryption [5] and authenticated symmetric encryption [22, 119], even in the absence of KDM considerations.)

A natural approach to achieve our goal is for Alice to encrypt under a symmetric, KDM-secure scheme and sign the ciphertexts under a conventional signature scheme. But it is not clear how to prove the resulting storage scheme is KDM-secure. The difficulty is that *sk* would include the signing key in addition to the encryption (and decryption) key *K*, so that messages depend on both these keys while the KDM security of the encryption only covers messages depending on *K*. We could attempt to start from scratch and design a secure storage scheme meeting our notions. But key-versatile signatures offer a simpler and more modular solution. Briefly, we take a KDM-secure *public-key* encryption scheme and let F be the one-way function that maps a secret key to a public key. Alice holds (only) a secret key *sk* and the server holds $pk = F(sk)$. To upload *M*, Alice re-computes *pk* from *sk*, encrypts *M* under it using the KDM scheme, and signs the ciphertext with an F-keyed signature scheme using the *same* key *sk*. The server verifies signatures under *pk*.

In Section 3.4 we present in full the construction outlined above, and prove that it meets our notion of KDM security. The crux, as for our RKA-secure constructions, is that adding signing capability without changing the keys puts us in a position to exploit the assumed KDM security of the underlying encryption scheme. The strong simulatability and key-extractability properties of our signatures do the rest. We note that as an added bonus, we assume only CPA KDM security of the base encryption scheme, yet our storage scheme achieves CCA KDM security.

Getting F-keyed signatures. In Section 3.1 we define F-keyed signature schemes and show how to construct them for arbitrary one-way F. This enables us to realize the above applications.

Our simulatability condition, adapting [51, 1, 50], asks for a trapdoor allowing the creation of simulated signatures given only the message and public key, even when the secret key underlying this public key is adversarially chosen. Our key-extractability condition, adapting [51], asks that, using the same trapdoor, one can extract from a valid signature the corresponding secret key, even when the public key is adversarially chosen. Theorem 3.1.1, showing these conditions imply not just standard but strong unforgeability, functions not just as a sanity check but as a way to introduce, in a simple form, a proof template that we will extend for our applications.

Our construction of an F-keyed signature scheme is a minor adaptation of a NIZK-based signature scheme of Dodis, Haralambiev, López-Alt and Wichs (DHLW) [62]. While DHLW [62] prove leakage-resilience of their scheme, we prove simulatability and key-extractability. The underlying SE NIZKs are a variant of simulation-sound extractable NIZKs [56, 85, 86] introduced by [62] under the name tSE NIZKs and shown by [62, 88] to be achievable for all of **NP** under standard assumptions.

Discussion and related work. F-keyed signatures can be viewed as a special case of signatures of knowledge as introduced by Chase and Lysyanskaya [51]. The main novelty of our work is in the notion of key-versatility, namely that F-keyed signatures can add signing capability without changing keys, and the ensuing applications to Joint Enc/Sig, RKA and KDM. In particular our work shows that signatures of knowledge have applications beyond those envisaged in [51].

The first NIZK-based signature scheme was that of [18]. It achieved only unforgeability. Simulatability and extractability were achieved in [51] using dense cryp-

tosystems [58, 57] and simulation-sound NIZKs [122, 56]. The DHLW construction we use can be viewed as a simplification and strengthening made possible by the significant advances in NIZK technology since then.

F-keyed signatures, and, more generally, signatures of knowledge [51] can be seen as a signing analogue of Witness Encryption [77, 80], and we might have named them Witness Signatures. GGSW [77] show how witness encryption allows encryption with a flexible choice of keys, just as we show that F-keyed signatures allow signing with a flexible choice of keys. We note that by starting from the IND-CPA PRG-based PKE scheme of [77], making it IND-CCA via SSE NIZKs in a way that does not change the public key [88, 50], and then applying our transformation discussed above, we can obtain a specific JES scheme where the public key is the result of a PRG on the secret key, so that the keys are not only the same for encryption and signing but particularly short in practice via blockcipher-based instantiations of the PRG.

Signcryption [137], authenticated public-key encryption [5], JES [87, 113] and our secure storage goal all have in common that both encryption and signature are involved. However, in signcryption and authenticated public-key encryption, there are two parties and thus two sets of keys, Alice encrypting under Bob's public key and signing under her own secret key. In JES and secure storage, there is one set of keys, namely Alice's. Thus for signcryption and authenticated public-key encryption, the question of using the same keys for the two purposes, which is at the core of our goals and methods, does not arise. Self-signcryption [73] is however similar to secure storage, minus the key-dependent message aspect. Authenticated symmetric encryption [22, 119] also involves both encryption and authentication, but under a shared key, while JES and secure storage involve public keys. KDM-secure authenticated symmetric encryption was studied in [19, 12].

KDM-secure signatures were studied in [106], who show limitations on the security achievable. Our secure storage scheme bypasses these limitations by signing ciphertexts rather than plaintexts and by avoiding KDM-secure signatures altogether: we use F-keyed signatures and are making no standalone claims or assumptions regarding their KDM security. Combining KDM encryption and KDM signatures would not give us KDM-secure storage because the keys for the two primitives would be different and we want joint KDM security.

Secure storage is an amalgam of symmetric and asymmetric cryptography, encryption being of the former kind and authentication of the latter. With secure storage, we are directly modeling a goal of practical interest rather than trying to create a general-purpose tool like many of the other works just mentioned. The difference between JES and secure storage is that in the former, arbitrary messages may be signed, while in the latter only ciphertexts may be signed. The difference is crucial for KDM security, which for JES would inherit the limitations of KDM-secure signatures just mentioned, but is not so limited for secure storage.

3.1 Key-versatile signatures

For F a family of functions, we define F-keyed signature schemes, which augment the basic digital signature schemes presented in Chapter 2. The requirement for F-keyed signatures is that the secret key sk is an input for an instance fp of the family and the public key $pk = \mathsf{F.Ev}(1^\lambda, fp, sk)$ is the corresponding image under this instance, the instance fp itself specified in public parameters. We intend to use these schemes to add authenticity in a setting where keys (sk, pk) may already be in use for another purpose (such as encryption). We need to ensure that signing will neither lessen the security of the existing usage of the keys nor have its own security be lessened by it. To ensure this strong form of composability, we define simulatability and key-extractability requirements for our

F-keyed schemes. The fact that the keys will already be in use for another purpose also means that we do not have the luxury of picking the family F, but must work with an arbitrary family emerging from another setting. The only assumption we will make on F is thus that it is one-way. (This is necessary, else security is clearly impossible.) With the definitions in place, we go on to indicate how to build F-keyed signature schemes for arbitrary, one-way F.

We clarify that being F-keyed under an F assumed to be one-way does not mean that security (simulatability and key-extractability) of the signature scheme is based *solely* on the assumption that F is one-way. The additional assumption is a SE-secure NIZK. (But this itself can be built under standard assumptions.) It is possible to build a signature scheme that is unforgeable assuming only that a given F is one-way [120], but this scheme will not be F-keyed relative to the same F underlying its security, and it will not be simulatable or key-extractable.

Function families. A function family F specifies the following. Via $fp \leftarrow\!\!\$\ \mathsf{F.Pg}(1^\lambda)$ one can in PT generate a description fp of a function $\mathsf{F.Ev}(1^\lambda, fp, \cdot)$: $\mathsf{F.Dom}(1^\lambda, fp) \to \mathsf{F.Rng}(1^\lambda, fp)$. We assume that membership of x in the non-empty domain $\mathsf{F.Dom}(1^\lambda, fp)$ can be tested in time polynomial in $1^\lambda, fp, x$ and one can in time polynomial in $1^\lambda, fp$ sample a point $x \leftarrow\!\!\$\ \mathsf{F.Dom}(1^\lambda, fp)$ from the domain $\mathsf{F.Dom}(1^\lambda, fp)$. The deterministic evaluation algorithm F.Ev is PT. The range is defined by $\mathsf{F.Rng}(1^\lambda, fp) = \{\, \mathsf{F.Ev}(1^\lambda, fp, x) : x \in \mathsf{F.Dom}(1^\lambda, fp) \,\}$. Testing membership in the range is not required to be PT. (But is in many examples.) We say that F is one-way or F is a OWF if $\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F},I}(\cdot)$ is negligible for all PT I, where $\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F},I}(\lambda) = \Pr[\mathsf{F.Ev}(1^\lambda, fp, x') = y]$ under the experiment $fp \leftarrow\!\!\$\ \mathsf{F.Pg}(1^\lambda)\,;\ x \leftarrow\!\!\$\ \mathsf{F.Dom}(1^\lambda, fp)\,;\ y \leftarrow \mathsf{F.Ev}(1^\lambda, fp, x)\,;\ x' \leftarrow\!\!\$\ I(1^\lambda, fp, y)$.

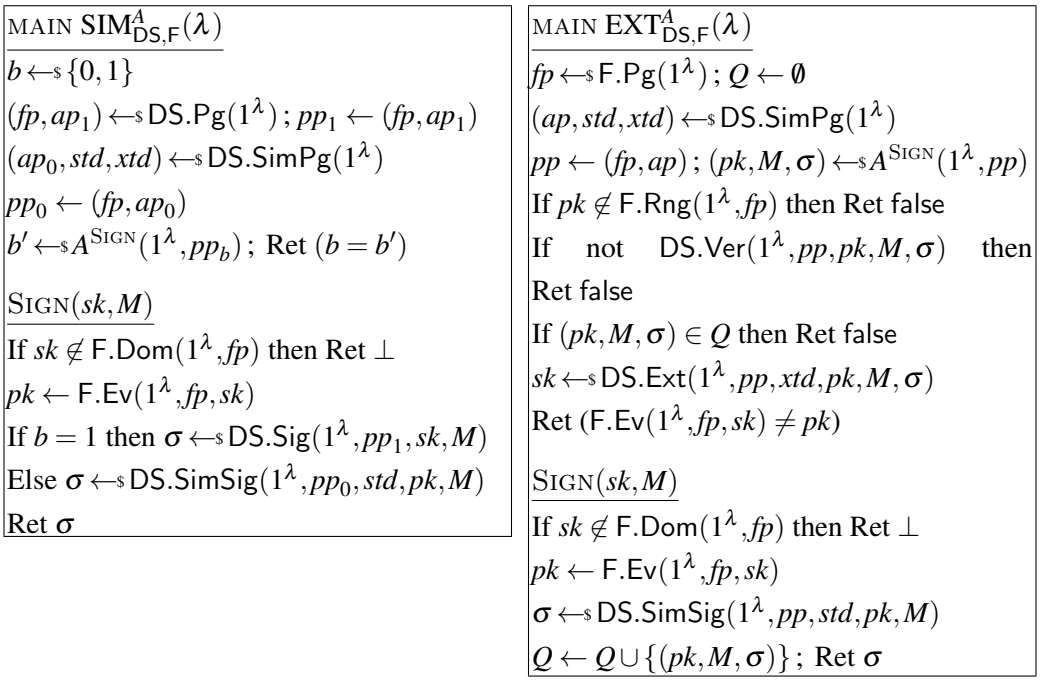


Figure 3.1. Games defining security of F-keyed signature scheme DS. Left: Game defining simulatability. Right: Game defining key extractability.

F-keyed signature schemes. Let F be a function family. We say that a signature scheme DS is F-*keyed* if the following are true:

- Parameter compatibility: Parameters pp for DS are a pair $pp = (fp, ap)$ consisting of parameters fp for F and auxiliary parameters ap, these independently generated. Formally, there is a PT *auxiliary parameter generation* algorithm APg such that $\mathsf{DS.Pg}(1^\lambda)$ picks $fp \leftarrow\$ \mathsf{F.Pg}(1^\lambda)$; $ap \leftarrow\$ \mathsf{APg}(1^\lambda)$ and returns (fp, ap).

- Key compatibility: The signing key sk is a random point in the domain of F.Ev and the verifying key pk is its image under F.Ev. Formally, $\mathsf{DS.Kg}(1^\lambda, (fp, ap))$ picks $sk \leftarrow\$ \mathsf{F.Dom}(1^\lambda, fp)$, lets $pk \leftarrow \mathsf{F.Ev}(1^\lambda, fp, sk)$ and returns (sk, pk). (DS.Kg ignores the auxiliary parameters ap, meaning the keys do not depend on it.)

Security of F-keyed signature schemes. We require two (strong) security properties of an F-keyed signature scheme DS:

- Simulatable: Under simulated auxiliary parameters and an associated simulation trapdoor *std*, a simulator, given $pk = \mathsf{F.Ev}(1^\lambda, fp, sk)$ and M, can produce a signature σ indistinguishable from the real one produced under *sk*, when not just M, *but even the secret key sk*, is adaptively chosen by the adversary. Formally, DS is *simulatable* if it specifies additional PT algorithms $\mathsf{DS.SimPg}$ (the auxiliary parameter simulator) and $\mathsf{DS.SimSig}$ (the signature simulator) such that $\mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A}(\cdot)$ is negligible for every PT adversary A, where $\mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A}(\lambda) = 2\Pr[\mathrm{SIM}^A_{\mathsf{DS}}(\lambda)] - 1$ and game SIM is specified on the left-hand side of Figure 3.1.

- Key-extractable: Under the same simulated auxiliary parameters and an associated extraction trapdoor *xtd*, an extractor can extract from any valid forgery relative to *pk* an underlying secret key *sk*, even when *pk is chosen by the adversary* and the adversary can adaptively obtain simulated signatures *under secret keys of its choice.* Formally, DS is *key-extractable* if it specifies another PT algorithm $\mathsf{DS.Ext}$ (the extractor) such that $\mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A}(\cdot)$ is negligible for every PT adversary A, where $\mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A}(\lambda) = \Pr[\mathrm{EXT}^A_{\mathsf{DS}}(\lambda)]$ and game EXT is specified on the right-hand side of Figure 3.1.

The EXT game includes a possibly non-PT test of membership in the range of the family, but we will ensure that adversaries (who must remain PT) do not perform this test. Our definition of simulatability follows [51, 1, 50]. Those definitions were for general signatures, not F-keyed ones, and one difference is that our simulator can set only the auxiliary parameters, not the full parameters, meaning it does not set *fp*.

Sim+Ext implies unforgeability. The simulatability and key-extractability notions we have defined may seem quite unrelated to the standard unforgeability requirement for signature schemes, as presented in Section 2.1. As a warm-up towards applying these new conditions, we show that in fact they imply not just standard unforgeability but strong unforgeability (as defined in Section 2.1, under the minimal assumption that F is one-way.

Theorem 3.1.1. *Let* DS *be an* F*-keyed signature scheme that is simulatable and key-extractable. If* F *is one-way then* DS *is strongly unforgeable.*

Before we present the formal proof of the theorem, we sketch the intuition. Given an adversary A against unforgeability, we build an inverter I for F. On input $1^\lambda, fp, pk$, adversary I generates simulated auxiliary parameters ap together with simulation and extraction trapdoors. It now runs A with parameters (fp, ap), answering signing queries via the signature simulator. (Note the latter only needs the simulation trapdoor and the public key, not the secret key.) When A produces its forgery M, σ, the inverter I runs the extractor to obtain sk, a pre-image of pk under $\mathsf{F.Ev}(1^\lambda, fp, \cdot)$. The simulation and key-extractability conditions are invoked to show that I succeeds with almost the same probability as A. This involves the construction of adversaries A_1, A_2 for the two conditions. A question here is that these adversaries can only make signing queries under secret keys that they know, so how do they proceed not knowing sk? The answer is that they will themselves run key-generation to get (sk, pk) and then run A on the latter. Now, when A makes a signing query M, adversaries A_1, A_2 can answer by invoking their own signing oracles on sk, M.

A reader may note that the above theorem would hold under weaker simulatability and extractability conditions where the adversaries do not choose secret and public keys. This is true, but the stronger conditions are crucial to other upcoming applications in this paper.

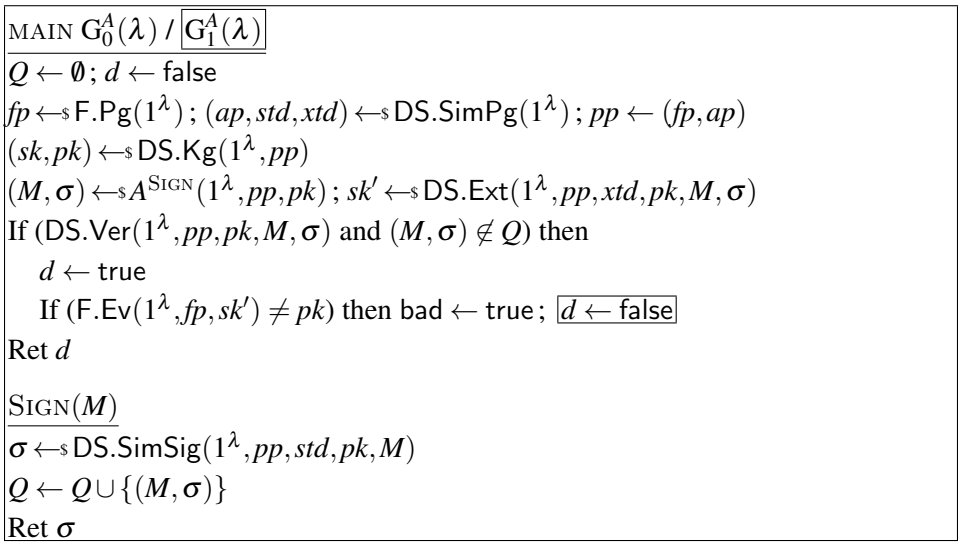


Figure 3.2. Games for the proof of Theorem 3.1.1. Game G_1 includes the boxed code while G_0 does not.

Proof. Let A be a PT adversary playing game SUF. We build PT adversaries I, A_1, A_2 such that

$$\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{DS},A}(\lambda) \leq \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F},I}(\lambda) + \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda)$$

for all $\lambda \in \mathbb{N}$, from which the theorem follows. The proof uses the games in Figure 3.2. These games switch to using simulated parameters and signatures. We will build I, A_1, A_2 so that for all $\lambda \in \mathbb{N}$ we have

$$\Pr[\mathrm{SUF}^A_{\mathsf{DS}}(\lambda)] - \Pr[\mathrm{G}^A_0(\lambda)] \leq \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) \tag{3.1}$$

$$\Pr[\mathrm{G}^A_0(\lambda) \text{ sets bad}] \leq \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda) \tag{3.2}$$

$$\Pr[\mathrm{G}^A_1(\lambda)] \leq \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F},I}(\lambda)\ . \tag{3.3}$$

Games G_0 and G_1 are identical until bad, so by the Fundamental Lemma of Game-Playing [25] and the above, for all $\lambda \in \mathbb{N}$ we have:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{DS},A}(\lambda) &= \Pr[\mathrm{SUF}^{A}_{\mathsf{DS}}(\lambda)] \\
&= (\Pr[\mathrm{SUF}^{A}_{\mathsf{DS}}(\lambda)] - \Pr[\mathrm{G}^{A}_{0}(\lambda)]) + (\Pr[\mathrm{G}^{A}_{0}(\lambda)] - \Pr[\mathrm{G}^{A}_{1}(\lambda)]) + \Pr[\mathrm{G}^{A}_{1}(\lambda)] \\
&\leq (\Pr[\mathrm{SUF}^{A}_{\mathsf{DS}}(\lambda)] - \Pr[\mathrm{G}^{A}_{0}(\lambda)]) + \Pr[\mathrm{G}^{A}_{0}(\lambda) \text{ sets bad}] + \Pr[\mathrm{G}^{A}_{1}(\lambda)] \\
&\leq \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda) + \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F},I}(\lambda)
\end{aligned}$$

as desired. We proceed to the constructions of A_1, A_2, I. Adversary A_1 behaves as follows:

$\underline{A_1^{\mathrm{SIGN}}(1^\lambda, pp)}$	
$(sk, pk) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)$; $Q \leftarrow \emptyset$	$\underline{\mathrm{SIGNSIM}(M)}$
$(M, \sigma) \leftarrow\!\!\$\ A^{\mathrm{SIGNSIM}}(1^\lambda, pp, pk)$	$\sigma \leftarrow\!\!\$\ \mathrm{SIGN}(sk, M)$
If $(\mathsf{DS.Ver}(1^\lambda, pp, pk, M, \sigma)$ and $(M, \sigma) \notin Q)$ then $b' \leftarrow 1$	$Q \leftarrow Q \cup \{(M, \sigma)\}$
Else $b' \leftarrow 0$	Ret σ
Return b'	

When the challenge bit b in game SIM is 0, adversary A_1 simulates for A game G_0, while if $b = 1$, adversary A_1 simulates game SUF. We thus have

$$\Pr[\mathrm{SUF}^{A}_{\mathsf{DS}}(\lambda)] - \Pr[\mathrm{G}^{A}_{0}(\lambda)] = \Pr[b' = 1 \,|\, b = 1] - \Pr[b' = 1 \,|\, b = 0] \leq \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda)\,,$$

which establishes Equation 3.1. Adversary A_2 behaves as follows:

$\underline{A_2^{\mathrm{SIGN}}(1^\lambda, pp)}$	$\underline{\mathrm{SIGNSIM}(M)}$
$(sk, pk) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)$; $(M, \sigma) \leftarrow\!\!\$\ A^{\mathrm{SIGNSIM}}(1^\lambda, pp, pk)$	$\sigma \leftarrow\!\!\$\ \mathrm{SIGN}(sk, M)$
Ret (pk, M, σ)	Ret σ

We skip the simple analysis establishing Equation 3.2. Adversary I behaves as follows:

$\underline{I(1^\lambda, fp, pk)}$	
$(ap, std, xtd) \leftarrow\!\!\$\ \mathsf{DS.SimPg}(1^\lambda)$	$\underline{\textsc{SignSim}(M)}$
$(M, \sigma) \leftarrow\!\!\$\ A^{\textsc{SignSim}}(1^\lambda, (fp, ap), pk)$	$\sigma \leftarrow\!\!\$\ \mathsf{DS.SimSig}(1^\lambda, pp, std, pk, M)$
$sk' \leftarrow\!\!\$\ \mathsf{DS.Ext}(1^\lambda, pp, xtd, pk, M, \sigma)$	Ret σ
Ret sk'	

We skip the simple analysis establishing Equation 3.3. □

Construction. A *key-versatile signing schema* is a transform **KvS** that given an arbitrary family of functions F returns an F-keyed signature scheme $\mathsf{DS} = \mathbf{KvS}[\mathsf{F}]$. We want the constructed signature scheme to be simulatable and key-extractable. We now show that this is possible with the aid of appropriate NIZK systems which are themselves known to be possible under standard assumptions.

Theorem 3.1.2. *Assume there exist SE-NIZK systems for all of* **NP**. *Then there is a key-versatile signing schema* **KvS** *such that if* F *is any family of functions then the signature scheme* $\mathsf{DS} = \mathbf{KvS}[\mathsf{F}]$ *is simulatable and key-extractable.*

Before presenting our construction, we provide some intuition. The scheme is simple. We define the relation $\mathsf{R}((1^\lambda, fp, pk, M), sk)$ to return true iff $\mathsf{F.Ev}(1^\lambda, fp, sk) = pk$. A signature of M under sk is then a SE-secure NIZK proof for this relation in which the witness is sk and the instance (input) is $(1^\lambda, fp, pk, M)$. The interesting aspect of this construction is that it at first sounds blatantly insecure, since the relation R ignores the message M. Does this not mean that a signature is independent of the message, in which case an adversary could violate unforgeability by requesting a signature σ of a message M under pk and then outputting (M', σ) as a forgery for some $M' \neq M$? What prevents this is the strength of the SE notion of NIZKs. The message M is present in the instance $(1^\lambda, fp, pk, M)$, even if it is ignored by the relation; the proof in turn depends on the instance, making the signature depend on M. Intuitively the SE-secure NIZK guarantees

a form of non-malleability, so signatures (proofs) for one message (instance) cannot be transferred to another. The formal proof shows simulatability of the F-keyed signature scheme based on the zero-knowledge property of the NIZK and key-extractability of the F-keyed signature scheme based on extractability of the NIZK, both in a natural and simple way.

A similar construction of signatures was given in [62] starting from a leakage-resilient hard relation rather than (as in our case) a relation arising from a one-way function. Our construction could be considered a special case of theirs, with the added difference that they use labeled NIZKs with the message as the label while we avoid labels and put the message in the input. The claims established about the construction are however different, with [62] establishing leakage resilience and unforgeability of the signature and our work showing simulatability and key-extractability. The technique of [62] was also used by [50] to construct malleable signatures. Going back further, the first NIZK-based signature scheme was that of [18]. This used PRFs and commitment, but only regular (as opposed to SE) NIZKs, these being all that was available at the time. One might see the simpler and more elegant modern NIZK-based signatures as being made possible by the arrival of the stronger NIZK systems of works like [56, 85, 86, 62].

Construction. Let F be the function family given in the theorem statement. We associate to it the **NP**-relation R defined by $\mathsf{R}((1^\lambda, fp, pk, M), sk) = (\mathsf{F.Ev}(1^\lambda, fp, sk) = pk)$ for all $\lambda \in \mathbb{N}$ and all $fp, pk, M, sk \in \{0,1\}^*$. Let Π be a NI system for R that is zero knowledge and simulation extractable (as defined in Section 2.2). The signature scheme $\mathsf{DS} = \mathbf{KvS}[\mathsf{F}]$ is specified as follows:

- $\underline{\mathsf{DS.Pg}(1^\lambda)}$: $crs \leftarrow\!\!\$\ \Pi.\mathsf{Pg}(1^\lambda)$; $fp \leftarrow\!\!\$\ \mathsf{F.Pg}(1^\lambda)$. Return (fp, crs).
- $\underline{\mathsf{DS.Kg}(1^\lambda, (fp, crs))}$: $sk \leftarrow\!\!\$\ \mathsf{F.Dom}(1^\lambda, fp)$; $pk \leftarrow \mathsf{F.Ev}(1^\lambda, fp, sk)$. Return (sk, pk).

- $\underline{\mathsf{DS.Sig}(1^\lambda,(fp,crs),sk,M)}$: $pk \leftarrow \mathsf{F.Ev}(1^\lambda,fp,sk)$. Return $\Pi.\mathsf{P}(1^\lambda,crs,(1^\lambda,fp,pk,M),sk)$.
- $\underline{\mathsf{DS.Ver}(1^\lambda,(fp,crs),pk,M,\sigma)}$: Return $\Pi.\mathsf{V}(1^\lambda,crs,(1^\lambda,fp,pk,M),\sigma)$.
- $\underline{\mathsf{DS.SimPg}(1^\lambda)}$: Ret $\Pi.\mathsf{SimPg}(1^\lambda)$.
- $\underline{\mathsf{DS.SimSig}(1^\lambda,(fp,crs),std,pk,M)}$: Ret $\Pi.\mathsf{SimP}(1^\lambda,crs,std,(1^\lambda,fp,pk,M))$.
- $\underline{\mathsf{DS.Ext}(1^\lambda,(fp,crs),xtd,pk,M,\sigma)}$: Ret $\Pi.\mathsf{Ext}(1^\lambda,crs,xtd,(1^\lambda,fp,pk,M),\sigma)$.

Security of the construction. Simulatability of the signature scheme follows directly from the zero knowledge property of the NIZK. Let A be a PT adversary playing game SIM. We construct a PT adversary B such that $\mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A}(\lambda) \leq \mathbf{Adv}^{\mathrm{zk}}_{\Pi,B}(\lambda)$ for all $\lambda \in \mathbb{N}$. B behaves as follows:

$\underline{B^{\textsc{Prove}}(1^\lambda,crs)}$	$\underline{\textsc{SignSim}(sk,M)}$
$fp \leftarrow\!\!\$\ \mathsf{F.Pg}(1^\lambda)$; $pp \leftarrow (fp,crs)$	If $sk \notin \mathsf{F.Dom}(1^\lambda,fp)$ then Ret $\perp$
$b' \leftarrow\!\!\$\ A^{\textsc{SignSim}}(1^\lambda,pp)$	$pk \leftarrow \mathsf{F.Ev}(1^\lambda,fp,sk)$
Return b'	$\pi \leftarrow\!\!\$\ \textsc{Prove}((1^\lambda,fp,pk,M),sk)$
	Ret π

The key extractability of the signature scheme likewise follows from the SE security of the NIZK. Let A be a PT adversary playing game EXT. We construct a PT adversary B such that $\mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A}(\lambda) \leq \mathbf{Adv}^{\mathrm{se}}_{\Pi,B}(\lambda)$ for all $\lambda \in \mathbb{N}$. B behaves as follows:

$\underline{B^{\textsc{Prove}}(1^\lambda,crs)}$	$\underline{\textsc{SignSim}(sk,M)}$
$fp \leftarrow\!\!\$\ \mathsf{F.Pg}(1^\lambda)$; $pp \leftarrow (fp,crs)$	If $sk \notin \mathsf{F.Dom}(1^\lambda,fp)$ then Ret $\perp$
$(pk,M,\sigma) \leftarrow\!\!\$\ A^{\textsc{SignSim}}(1^\lambda,pp)$	$pk \leftarrow \mathsf{F.Ev}(1^\lambda,fp,sk)$
Return $((1^\lambda,fp,pk,M),\sigma)$	$\pi \leftarrow\!\!\$\ \textsc{Prove}((1^\lambda,fp,pk,M),sk)$
	Ret π

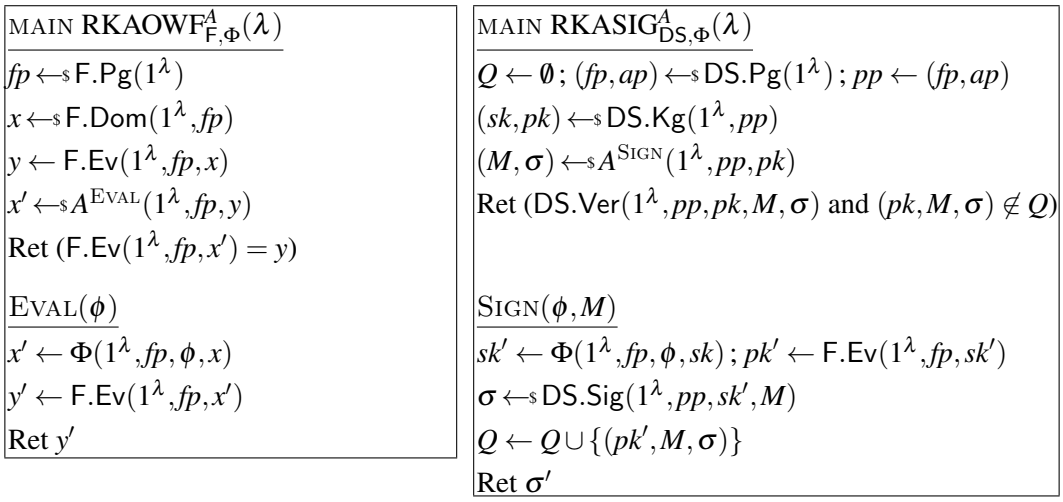


Figure 3.3. Games defining Φ-RKA security of a function family F (left) and an F-keyed signature scheme DS (right).

If $(pk, M, \sigma) \notin Q$ in game EXT then $((1^\lambda, fp, pk, M), \sigma) \notin Q$ in game SE, the sets being those defined in the games. Furthermore, by the definition of DS.Ext and R, if $sk \leftarrow \mathsf{DS.Ext}(1^\lambda, crs, xtd, pk, M, \sigma)$ is such that $\mathsf{F.Ev}(1^\lambda, fp, sk) \neq pk$, then $\mathsf{R}((1^\lambda, fp, pk, M), sk) = \mathsf{false}$.

3.2 RKA-secure signatures from RKA-secure OWFs

RKA security is notoriously hard to provably achieve. Recognizing this, several authors [79, 16] have suggested a bootstrapping approach in which we build higher-level RKA-secure primitives from lower-level RKA-secure primitives. In this vein, a construction of RKA-secure signatures from RKA-secure PRFs was given in [16]. We improve on this via a construction of RKA-secure signatures from RKA-secure one-way functions. The result is simple: If F is a Φ-RKA-secure OWF then any F-keyed simulatable and key-extractable signature scheme is also Φ-RKA secure. The benefit is that (as we will show) many popular OWFs are already RKA secure and we immediately get new RKA-secure signatures.

RKA security. Let F be a function family. A class of RKD (related-key deriving) functions Φ for F is a PT-computable function that specifies for each $\lambda \in \mathbb{N}$, each $fp \in [\mathsf{F.Pg}(1^\lambda)]$ and each $\phi \in \{0,1\}^*$ a map $\Phi(1^\lambda, fp, \phi, \cdot) : \mathsf{F.Dom}(1^\lambda, fp) \to \mathsf{F.Dom}(1^\lambda, fp)$ called the RKD function described by ϕ. We say that F is Φ-RKA secure if $\mathbf{Adv}^{\mathrm{rka}}_{\mathsf{F},A,\Phi}(\cdot)$ is negligible for every PT adversary A, where $\mathbf{Adv}^{\mathrm{rka}}_{\mathsf{F},A,\Phi}(\lambda) = \Pr[\mathrm{RKAOWF}^A_{\mathsf{F},\Phi}(\lambda)]$ and game RKAOWF is on the left-hand side of Figure 3.3. In this game, A, like in the basic one-wayness notion, is given $y = \mathsf{F.Ev}(1^\lambda, fp, x)$ and attempts to find x' such that $\mathsf{F.Ev}(1^\lambda, fp, x') = y$. Now, however, it has help. It can request that the hidden challenge input x be modified to $x' = \Phi(1^\lambda, fp, \phi, x)$ for any description ϕ of its choice, and obtain $y' = \mathsf{F.Ev}(1^\lambda, fp, x')$. This should not help it in its inversion task. The definition is from Goldenberg and Liskov [79], adapted to our notation, and represents a particularly simple and basic form of RKA security.

Let DS be an F-keyed signature scheme and let Φ be as above. We say that DS is Φ-RKA secure if $\mathbf{Adv}^{\mathrm{rka}}_{\mathsf{DS},A,\Phi}(\cdot)$ is negligible for every PT adversary A, where $\mathbf{Adv}^{\mathrm{rka}}_{\mathsf{DS},A,\Phi}(\lambda) = \Pr[\mathrm{RKASIG}^A_{\mathsf{DS},\Phi}(\lambda)]$ and game RKASIG is on the right-hand side of Figure 3.3. In this game, A, like in the basic (strong) unforgeability notion, is given public key pk and is attempting to forge a signature under it. Now, however, it has help beyond its usual signing oracle. It can request that the hidden secret key sk be modified to $sk' = \Phi(1^\lambda, fp, \phi, sk)$ for any description ϕ of its choice, and obtain a signature under sk' of any message of its choice. This should not help it in its forgery task. Our definition adapts the one of Bellare, Cash and Miller [16] for Φ-RKA security of arbitrary signature schemes to the special case of F-keyed signature schemes.[5]

[5]One change (strengthening the definition) is that we use a strong unforgeability formulation rather than an unforgeability one. On the other hand while [16] disallow A a victory from forgery M, σ when M was previously signed under $sk' = sk$, we disallow it when M was previously signed under $pk' = pk$ even if $sk' \neq sk$. In our setting this is more natural since the secret key determines the public key. In any case Theorem 3.2.1 extends to the definition of [16] assuming F is additionally injective or collision-resistant, which is true in most examples.

$\underline{\text{MAIN } \mathrm{G}_0^A(\lambda) \,/\, \boxed{\mathrm{G}_1^A(\lambda)}}$
$Q \leftarrow \emptyset$; $d \leftarrow \mathsf{false}$
$fp \leftarrow\!\!\$\, \mathsf{F.Pg}(1^\lambda)$; $(ap, std, xtd) \leftarrow\!\!\$\, \mathsf{DS.SimPg}(1^\lambda)$; $pp \leftarrow (fp, ap)$
$(sk, pk) \leftarrow\!\!\$\, \mathsf{DS.Kg}(1^\lambda, pp)$
$(M, \sigma) \leftarrow\!\!\$\, A^{\text{SIGN}}(1^\lambda, pp, pk)$; $sk' \leftarrow\!\!\$\, \mathsf{DS.Ext}(1^\lambda, pp, xtd, pk, M, \sigma)$
If ($\mathsf{DS.Ver}(1^\lambda, pp, pk, M, \sigma)$ and $(pk, M, \sigma) \notin Q$) then
  $d \leftarrow \mathsf{true}$
  If ($\mathsf{F.Ev}(1^\lambda, fp, sk') \neq pk$) then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{d \leftarrow \mathsf{false}}$
Ret d

$\underline{\text{SIGN}(\phi, M)}$
$sk' \leftarrow \Phi(1^\lambda, fp, \phi, sk)$; $pk' \leftarrow \mathsf{F.Ev}(1^\lambda, fp, sk')$
$\sigma \leftarrow\!\!\$\, \mathsf{DS.SimSig}(1^\lambda, pp, std, pk', M)$
$Q \leftarrow Q \cup \{(pk', M, \sigma)\}$
Ret σ

Figure 3.4. Games used in proof of Theorem 3.2.1. Game G_1 includes the boxed code and G_0 does not.

Construction. Suppose we are given a Φ-RKA secure OWF F and want to build a Φ-RKA secure signature scheme. For the question to even make sense, RKD functions specified by Φ must apply to the secret signing key. Thus, the secret key needs to be an input for the OWF and the public key needs to be the image of the secret key under the OWF. The main technical difficulty is, given F, finding a signature scheme with this property. But this is exactly what a key-versatile signing schema gives us. The following says that if the signature scheme produced by this schema is simulatable and key-extractable then it inherits the Φ-RKA security of the OWF.

Theorem 3.2.1. *Let* DS *be an* F*-keyed signature scheme that is simulatable and key-extractable. Let* Φ *be a class of RKD functions. If* F *is* Φ*-RKA secure then* DS *is also* Φ*-RKA secure.*

Proof. Let A be a PT adversary playing game RKASIG. We build PT adversaries A_1, A_2, I such that

$$\mathbf{Adv}^{\mathrm{rka}}_{\mathsf{DS},A,\Phi}(\lambda) \leq \mathbf{Adv}^{\mathrm{rka}}_{\mathsf{F},I,\Phi}(\lambda) + \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda)$$

for all $\lambda \in \mathbb{N}$, from which the theorem follows.

The proof uses the games in Figure 3.4. These games switch to using simulated parameters and signatures. We will build A_1, A_2, I so that for all $\lambda \in \mathbb{N}$ we have

$$\Pr[\mathrm{RKASIG}^{A}_{\mathsf{DS},\Phi}(\lambda)] - \Pr[\mathrm{G}^{A}_{0}(\lambda)] \leq \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) \tag{3.4}$$

$$\Pr[\mathrm{G}^{A}_{0}(\lambda) \text{ sets } \mathsf{bad}] \leq \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda) \tag{3.5}$$

$$\Pr[\mathrm{G}^{A}_{1}(\lambda)] \leq \mathbf{Adv}^{\mathrm{rka}}_{\mathsf{F},I,\Phi}(\lambda)\,. \tag{3.6}$$

Games G_0 and G_1 are identical until bad, so by the Fundamental Lemma of Game-Playing [25] and the above, for all $\lambda \in \mathbb{N}$ we have:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{rka}}_{\mathsf{DS},A,\Phi}(\lambda) &= \Pr[\mathrm{RKASIG}^{A}_{\mathsf{DS},\Phi}(\lambda)] \\
&= (\Pr[\mathrm{RKASIG}^{A}_{\mathsf{DS},\Phi}(\lambda)] - \Pr[\mathrm{G}^{A}_{0}(\lambda)]) + (\Pr[\mathrm{G}^{A}_{0}(\lambda)] - \\
&\quad \Pr[\mathrm{G}^{A}_{1}(\lambda)]) + \Pr[\mathrm{G}^{A}_{1}(\lambda)] \\
&\leq (\Pr[\mathrm{RKASIG}^{A}_{\mathsf{DS},\Phi}(\lambda)] - \Pr[\mathrm{G}^{A}_{0}(\lambda)]) + \\
&\quad \Pr[\mathrm{G}^{A}_{0}(\lambda) \text{ sets } \mathsf{bad}] + \Pr[\mathrm{G}^{A}_{1}(\lambda)] \\
&\leq \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda) + \mathbf{Adv}^{\mathrm{rka}}_{\mathsf{F},I,\Phi}(\lambda)
\end{aligned}$$

as desired. We proceed to the constructions of A_1, A_2, I. Adversary A_1 behaves as follows:

$\underline{A_1^{\text{SIGN}}(1^\lambda, pp)}$	$\underline{\text{SIGNSIM}(\phi, M)}$
$(sk, pk) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)\,;\ Q \leftarrow \emptyset$	$sk' \leftarrow \Phi(1^\lambda, fp, \phi, sk)$
$(M, \sigma) \leftarrow\!\!\$\ A^{\text{SIGNSIM}}(1^\lambda, pp, pk)$	$pk' \leftarrow \mathsf{F.Ev}(1^\lambda, fp, sk')$
If $(\mathsf{DS.Ver}(1^\lambda, pp, pk, M, \sigma) \wedge (pk, M, \sigma) \notin Q)$ then $b' \leftarrow 1$	$\sigma \leftarrow\!\!\$\ \text{SIGN}(sk', M)$
Else $b' \leftarrow 0$	$Q \leftarrow Q \cup \{(pk', M, \sigma)\}$
Return b'	Ret σ

When the challenge bit b in game SIM is 0, adversary A_1 simulates for A game G_0, and if $b = 1$, adversary A_1 simulates game RKASIG. We thus have

$$\begin{aligned}\Pr[\text{RKASIG}^A_{\mathsf{DS},\Phi}(\lambda)] - \Pr[\text{G}_0^A(\lambda)] &= \Pr[b' = 1 \,|\, b = 1] - \Pr[b' = 1 \,|\, b = 0] \\ &\leq \mathbf{Adv}^{\text{sim}}_{\mathsf{DS},A_1}(\lambda)\ ,\end{aligned}$$

establishing Equation 3.4. Adversary A_2 behaves as follows:

$\underline{A_2^{\text{SIGN}}(1^\lambda, pp)}$	$\underline{\text{SIGNSIM}(\phi, M)}$
$(sk, pk) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)$	$sk' \leftarrow \Phi(1^\lambda, fp, \phi, sk)\,;\ pk' \leftarrow \mathsf{F.Ev}(1^\lambda, fp, sk')$
$(M, \sigma) \leftarrow\!\!\$\ A^{\text{SIGNSIM}}(1^\lambda, pp, pk)$	$\sigma \leftarrow\!\!\$\ \text{SIGN}(sk', M)\,;\ Q \leftarrow Q \cup \{(pk', M, \sigma)\}$
Ret (pk, M, σ)	Ret σ

If bad is set to true in game G_0 then we have: (1) $\mathsf{DS.Ver}(1^\lambda, pp, pk, M, \sigma)$, (2) $(pk, M, \sigma) \notin Q$, and (3) $\mathsf{F.Ev}(1^\lambda, fp, sk') \neq pk$. These are exactly the necessary conditions for A_2 to win game EXT, establishing Equation 3.5. I behaves as follows:

$\underline{I^{\text{EVAL}}(1^\lambda, fp, pk)}$	$\underline{\text{SIGNSIM}(\phi, M)}$
$(ap, std, xtd) \leftarrow\!\!\$\ \mathsf{DS.SimPg}(1^\lambda)$	$pk' \leftarrow\!\!\$\ \text{EVAL}(\phi)$
$(M, \sigma) \leftarrow\!\!\$\ A^{\text{SIGNSIM}}(1^\lambda, (fp, ap), pk)$	$\sigma \leftarrow\!\!\$\ \mathsf{DS.SimSig}(1^\lambda, pp, std, pk', M)$
$sk' \leftarrow\!\!\$\ \mathsf{DS.Ext}(1^\lambda, pp, xtd, pk, M, \sigma)$	Ret σ
Ret sk'	

We omit the analysis establishing Equation 3.6. □

Finding Φ-RKA OWFs. Theorem 3.2.1 motivates finding Φ-RKA-secure function families F. The merit of our approach is that there are many such families. To enable systematically identifying them, we adapt the definition of key-malleable PRFs of [15] to OWFs. We say that a function family F is Φ-*key-malleable* if there is a PT algorithm M, called a Φ-key-simulator, such that $M(1^\lambda, fp, \phi, \mathsf{F.Ev}(1^\lambda, fp, x)) = \mathsf{F.Ev}(1^\lambda, fp, \Phi(1^\lambda, fp, \phi, x))$ for all $\lambda \in \mathbb{N}$, all $fp \in [\mathsf{F.Pg}(1^\lambda)]$, all $\phi \in \{0,1\}^*$ and all $x \in \mathsf{F.Dom}(1^\lambda, fp)$.

Proposition 3.2.2. *Let* F *be a function family and* Φ *a class of RKD functions. If* F *is* Φ*-key-malleable and one-way then* F *is* Φ*-RKA secure.*

Proof. Let A be a PT adversary attacking the Φ-RKA security of F and let M be a Φ-key-simulator. We construct a PT adversary B against the (regular) one-wayness of F such that $\mathbf{Adv}^{\mathrm{rka}}_{\mathsf{F},A}(\lambda) \leq \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F},B}(\lambda)$ for all $\lambda \in \mathbb{N}$. On input $(1^\lambda, fp, y)$, adversary B runs $A(1^\lambda, fp, y)$. When A makes a EVAL query ϕ, adversary B computes $y' \leftarrow M(1^\lambda, fp, \phi, y)$ and returns y' to A. Φ-key malleability says that $y' = \mathsf{F.Ev}(1^\lambda, fp, \Phi(1^\lambda, fp, \phi, x))$ as A expects. When A eventually halts and outputs a value x', adversary B does the same. □

Previous uses of key-malleability [15, 23] for RKA security required additional conditions on the primitives, such as key-fingerprints in the first case and some form of collision-resistance in the second. For OWFs, it is considerably easier, key-malleability alone sufficing. We now exemplify how to leverage Proposition 3.2.2 to find Φ-RKA OWFs and thence, via Theorem 3.2.1, Φ-RKA signature schemes. Table 3.1 examines three popular one-way functions: discrete exponentiation in a cyclic group, RSA, and the LWE one-way function. It succinctly describes F, Φ, and the Φ-key-simulator M showing Φ-key-malleability. Briefly:

Table 3.1. Φ-RKA secure OWFs: We succinctly define the families and the Φ-key-simulator showing their Φ malleability and hence their Φ-RKA security.

F	fp	x	$\mathsf{F.Ev}(1^\lambda, fp, x)$	ϕ	$\Phi(1^\lambda, fp, \phi, x)$	$M(1^\lambda, fp, \phi, y)$
EXP	$(\langle\mathbb{G}\rangle, g, m)$	$\in \mathbb{Z}_m$	g^x	$(a,b) \in \mathbb{Z}_m \times \mathbb{Z}_m$	$(ax+b) \bmod m$	$y^a g^b$
RSA	(N, e)	$\in \mathbb{Z}_N^*$	$x^e \bmod N$	$a \in \mathbb{N}$	$x^a \bmod N$	$y^a \bmod N$
LWE	(A, n, m, q)	$(s,e) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^n$	$(As+e) \bmod q$	$(s', e') \in \mathbb{Z}_q^m \times \mathbb{Z}_q^n$	$(s+s', e+e') \bmod q$	$(y+As'+e') \bmod q$

- EXP: The first row of Table 3.1 shows that exponentiation in any group with hard discrete logarithm problem is Φ-RKA secure for the class Φ of affine functions over the exponent space. Here $\mathbb{G}$ is a cyclic group of order m generated by $g \in \mathbb{G}$.

- RSA: The second row of Table 3.1 shows that the RSA function is Φ-RKA secure for the class Φ of functions raising the input to integer powers a, under the assumption that RSA is one-way. Here N is an RSA modulus and $e \in \mathbb{Z}^*_{\varphi(N)}$ is an encryption exponent. Notice that in this rendition of RSA the latter has no trapdoor.

- LWE: The third row of Table 3.1 shows that the LWE function is Φ-RKA secure for the class Φ of functions shown. Here A is an n by m matrix over $\mathbb{Z}_q$ and Φ-RKA-security relies on the standard LWE one-wayness assumption.

The summary is that standard, natural one-way functions are Φ-RKA secure, leading to Φ-RKA security over standard and natural keyspaces.

3.3 Joining signature to encryption with no public-key overhead

Let PKE be an arbitrary IND-CCA-secure public-key encryption scheme. As an example, it could be the Cramer-Shoup scheme [54], the Kurosawa-Desmedt scheme [99], or the DDN scheme [63], but it could be any other IND-CCA-secure scheme as well. Alice has already established a key-pair (sk_e, pk_e) for this scheme, allowing anyone to send her ciphertexts computed under pk_e that she can decrypt under sk_e. She wants now to add signature capability. This is easily done. She can create a key-pair (sk_s, pk_s) for her favorite signature scheme and sign an arbitrary message M under sk_s, verification being possible given pk_s. The difficulty is that her public key is now $pk = (pk_e, pk_s)$. It is not just larger but will require a new certificate. The question we ask is whether we can add signing capability in a way that is more parsimonious with regard to public key

size. Technically, we seek a joint encryption and signature (JES) scheme where Alice has a single key-pair (sk, pk), with sk used to decrypt and sign, and pk used to encrypt and verify, each usage secure in the face of the other, and we want pk smaller than that of the trivial solution $pk = (pk_e, pk_s)$. Perhaps surprisingly, we show how to construct a JES scheme with pk-overhead zero, meaning pk is unchanged, remaining pk_e. We not only manage to use sk_e to sign and pk_e to verify, but do so in such a way that the security of the encryption is not affected by the presence of the signature, and vice versa. Previous standard model JES schemes had been able to reduce the pk-overhead only for *specific* starting encryption schemes [87, 113]. Our result says the overhead can be zero regardless of the starting encryption scheme. The result is obtained by defining F as the function mapping sk_e to pk_e and using a simulatable and key-extractable F-keyed signature scheme with the keys remaining (sk_e, pk_e).

JES schemes. A joint encryption and signature (JES) scheme JES specifies the following PT algorithms: via $jp \leftarrow\!\!\$\, \mathsf{JES.Pg}(1^\lambda)$ one generates public parameters jp common to all users; via $(sk, pk) \leftarrow\!\!\$\, \mathsf{JES.Kg}(1^\lambda, jp)$ a user can generate a secret (signing and decryption) key sk and corresponding public (verification and encryption) key pk; via $\sigma \leftarrow\!\!\$\, \mathsf{JES.Sig}(1^\lambda, jp, sk, M)$ the user can generate a signature σ on a message $M \in \{0,1\}^*$; via $d \leftarrow \mathsf{JES.Ver}(1^\lambda, jp, pk, M, \sigma)$ a verifier can deterministically produce a decision $d \in \{\mathsf{true}, \mathsf{false}\}$ regarding whether σ is a valid signature of M under pk; via $C \leftarrow\!\!\$\, \mathsf{JES.Enc}(1^\lambda, jp, pk, M)$ anyone can generate a ciphertext C encrypting message M under pk; via $M \leftarrow \mathsf{JES.Dec}(1^\lambda, jp, sk, C)$ the user can deterministically decrypt ciphertext C to get a value $M \in \{0,1\}^* \cup \{\perp\}$.

Correctness requires both that $\mathsf{JES.Ver}(1^\lambda, jp, pk, M, \mathsf{JES.Sig}(1^\lambda, jp, sk, M)) = \mathsf{true}$ and that $\mathsf{JES.Dec}(1^\lambda, jp, sk, \mathsf{JES.Enc}(1^\lambda, jp, pk, M)) = M$ for all $\lambda \in \mathbb{N}$, all $jp \in [\mathsf{JES.Pg}(1^\lambda)]$, all $(sk, pk) \in [\mathsf{JES.Kg}(1^\lambda, jp)]$, and all $M \in \{0,1\}^*$. We say that JES is

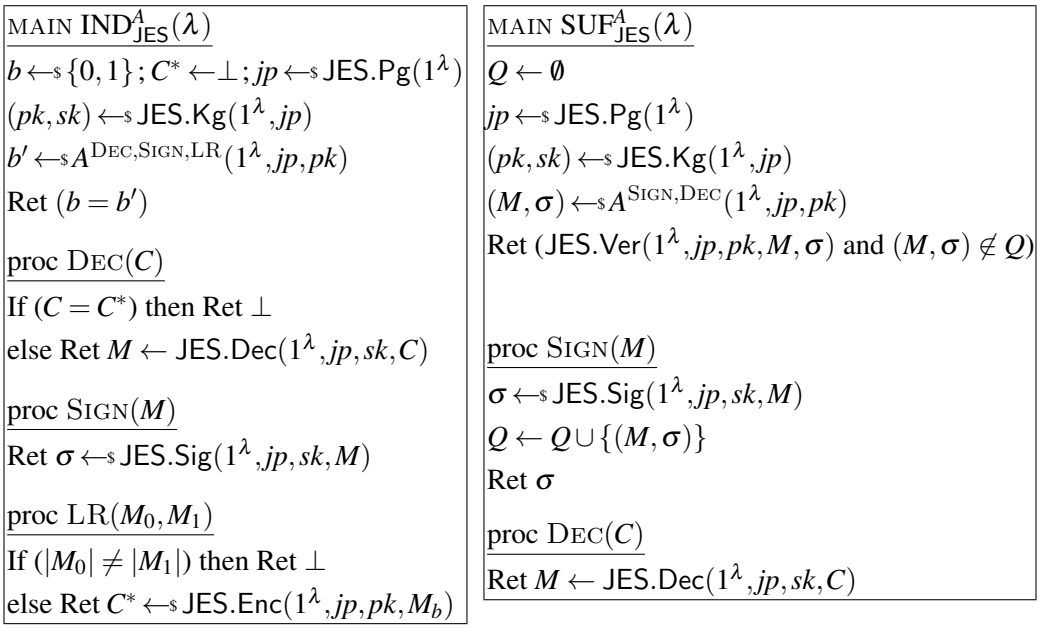


Figure 3.5. Games defining security of joint encryption and signature scheme JES. Left: Game IND defining privacy against chosen-ciphertext attack in the presence of a signing oracle. Right: Game SUF defining strong unforgeability in the presence of a decryption oracle.

SUF-secure if $\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{JES},A}(\cdot)$ is negligible for all PT adversaries A, where $\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{JES},A}(\lambda) = \Pr[\mathrm{SUF}^A_{\mathsf{JES}}(\lambda)]$ and game SUF is on the right-hand side of Figure 3.5. This represents (strong) unforgeability of the signature in the presence of a decryption oracle. We say that JES is IND-secure if $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{JES},A}(\cdot)$ is negligible for all PT adversaries A, where $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{JES},A}(\lambda) = 2\Pr[\mathrm{IND}^A_{\mathsf{JES}}(\lambda)] - 1$ and game IND is on the left-hand side of Figure 3.5. Here the adversary is allowed only one query to LR. This represents privacy under chosen-ciphertext attack in the presence of a signing oracle. These definitions are from [87, 113].

The base PKE scheme. We are given a public-key encryption scheme PKE, specifying the following PT algorithms: via $fp \leftarrow\!\!\$ \mathsf{PKE.Pg}(1^\lambda)$ one generates public parameters; via $(sk, pk) \leftarrow\!\!\$ \mathsf{PKE.Kg}(1^\lambda, fp)$ a user generates a decryption key sk and encryption key

pk; via $C \leftarrow\!\!\$\ \mathsf{PKE.Enc}(1^\lambda, fp, pk, M)$ anyone can generate a ciphertext C encrypting a message M under pk; and via $M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, sk, C)$ a user can deterministically decrypt a ciphertext C to get a value $M \in \{0,1\}^* \cup \{\perp\}$. Correctness requires that $\mathsf{PKE.Dec}(1^\lambda, fp, sk, \mathsf{PKE.Enc}(1^\lambda, fp, pk, M)) = M$ for all $\lambda \in \mathbb{N}$, all $fp \in [\mathsf{PKE.Pg}(1^\lambda)]$, all $(sk, pk) \in [\mathsf{PKE.Kg}(1^\lambda, fp)]$, and all $M \in \{0,1\}^*$. We assume that PKE meets the usual notion of IND-CCA security.

Let us say that PKE is *canonical* if the operation $(sk, pk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda, fp)$ picks sk at random from a finite, non-empty set we denote $\mathsf{PKE.SK}(1^\lambda, fp)$, and then applies to $(1^\lambda, fp, sk)$ a PT deterministic *public-key derivation function* we denote PKE.PK to get pk. Canonicity may seem like an extra assumption, but isn't. First, many (most) schemes are already canonical. This is true for the Cramer-Shoup scheme [54], the Kurosawa-Desmedt scheme [99] and for schemes obtained via the BCHK transform [36] applied to the identity-based encryption schemes of Boneh-Boyen [34] or Waters [132]. Second, if by chance a scheme is not canonical, we can modify it be so. Crucially (for our purposes), the modification *does not change the public key*. (But it might change the secret key.) Briefly, the modification, which is standard, is to use the random coins of the key generation algorithm as the secret key. In some more detail, given PKE, the new key-generation algorithm, on inputs $1^\lambda, fp$, picks random coins ω, lets $(sk, pk) \leftarrow \mathsf{PKE.Kg}(1^\lambda, fp; \omega)$, and returns (ω, pk), so that the new secret key is ω and the public key is still pk. Encryption is unchanged. The modified decryption algorithm, given $1^\lambda, fp, \omega, C$, lets $(sk, pk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda, fp; \omega)$ and outputs $M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, sk, C)$. It is easy to see that the modified scheme is canonical and also inherits both the correctness and the IND-CCA security of the original scheme.

Construction. Given canonical PKE as above, we construct a JES scheme JES. The first step is to construct from PKE a function family F as follows: let $\mathsf{F.Pg} = \mathsf{PKE.Pg}$,

so the parameters of F are the same those of PKE; let $\mathsf{F.Dom} = \mathsf{PKE.SK}$, so the domain of F is the space of secret keys of PKE; and let $\mathsf{F.Ev} = \mathsf{PKE.PK}$, so the function defined by fp maps a secret key to a corresponding public key. Now let DS be an F-keyed signature scheme that is simulatable and key-extractable. (We can obtain DS via Theorem 3.1.2.) Now we define our JES scheme JES. Let $\mathsf{JES.Pg} = \mathsf{DS.Pg}$, so parameters for JES have the form $jp = (fp, ap)$, where fp are parameters for F, which by definition of F are also parameters for PKE. Let $\mathsf{JES.Kg} = \mathsf{DS.Kg}$. (Keys are those of PKE which are also those of DS.) Let $\mathsf{JES.Sig} = \mathsf{DS.Sig}$ and $\mathsf{JES.Ver} = \mathsf{DS.Ver}$, so the signing and verifying algorithms of the joint scheme JES are inherited from the signature scheme DS. Let $\mathsf{JES.Enc}(1^\lambda, (fp, ap), pk, M)$ return $\mathsf{PKE.Enc}(1^\lambda, fp, pk, M)$ and let $\mathsf{JES.Dec}(1^\lambda, (fp, ap), sk, C)$ return $\mathsf{PKE.Dec}(1^\lambda, fp, sk, C)$, so the encryption and decryption algorithms of the joint scheme JES are inherited from the PKE scheme PKE. Note that the public key of the joint scheme JES is exactly that of PKE, so there is zero public-key overhead. (If PKE had been born canonical, there is also zero secret-key overhead. Had it undergone the transformation described above to make it canonical, the secret-key overhead would be non-zero but the public-key overhead would still be zero because the transformation did not change the public key.) The following says that JES is both IND and SUF secure.

Theorem 3.3.1. *Let* PKE *be a canonical public-key encryption scheme. Let* F *be defined from it as above. Let* DS *be an* F*-keyed signature scheme, and let* JES *be the corresponding joint encryption and signature scheme constructed above. Assume* PKE *is IND-CCA secure. Assume* DS *is simulatable and key-extractable. Then (1)* JES *is IND secure, and (2)* JES *is SUF secure.*

Before presenting the formal proof, we sketch the main ideas. For (1), given an adversary A against the IND security of JES, we build an adversary D against the IND-CCA security of PKE. D will simply run A on simulated auxiliary parameters,

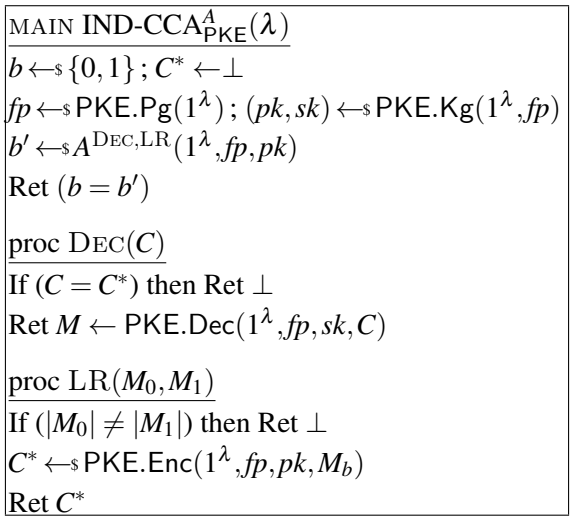


Figure 3.6. Game defining IND-CCA security of PKE scheme PKE.

using the simulator to answer A's SIGN queries and using its own DEC oracle to answer A's DEC queries. A simulation adversary is built alongside, but key extraction is not needed. For (2), given an adversary A against the SUF security of JES, we again build an adversary D against the IND-CCA security of PKE. It will run A with simulated auxiliary parameters, replying to A's oracle queries as before. From a forgery it extracts the secret key, using this to defeat IND-CCA security. Adversaries A_1 and A_2 against simulatability and key-extractability of DS are built alongside to show that D succeeds.

We say PKE scheme PKE is IND-CCA secure if $\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKE},A}(\cdot)$ is negligible for all PT adversaries A, where $\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKE},A}(\lambda) = 2\Pr[\text{IND-CCA}^A_{\mathsf{PKE}}(\lambda)] - 1$ and game IND-CCA is in Figure 3.6. The adversary is allowed only one query to LR. This definition is from [17, 53]. We proceed to prove Theorem 3.3.1.

Proof. Part (1): IND security

Let A be a PT adversary playing game IND. We build PT adversaries A_1, D such that

$$\mathbf{Adv}^{\text{ind}}_{\mathsf{JES},A}(\lambda) \leq \mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKE},D}(\lambda) + 2\mathbf{Adv}^{\text{sim}}_{\mathsf{DS},A_1}(\lambda)$$

$\underline{\text{MAIN } \mathrm{G}_0^A(\lambda)}$
$b \leftarrow\!\!\$\ \{0,1\}\ ;\ C^* \leftarrow \perp$
$fp \leftarrow\!\!\$\ \mathsf{F.Pg}(1^\lambda)\ ;\ (ap, std, xtd) \leftarrow\!\!\$\ \mathsf{DS.SimPg}(1^\lambda)\ ;\ jp \leftarrow (fp, ap)$
$(sk, pk) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, jp)$
$b' \leftarrow\!\!\$\ A^{\text{DEC},\text{SIGN},\text{LR}}(1^\lambda, jp, pk)$
Ret $(b = b')$

$\underline{\text{proc } \text{DEC}(C)}$
If $(C = C^*)$ then Ret $\perp$
Ret $M \leftarrow \mathsf{JES.Dec}(1^\lambda, jp, sk, C)$

$\underline{\text{proc } \text{SIGN}(M)}$
Ret $\mathsf{DS.SimSig}(1^\lambda, jp, std, pk, M)$

$\underline{\text{proc } \text{LR}(M_0, M_1)}$
If $(|M_0| \neq |M_1|)$ then Ret $\perp$
$C^* \leftarrow\!\!\$\ \mathsf{JES.Enc}(1^\lambda, jp, pk, M_b)$
Ret C^*

Figure 3.7. Game used in the proof of part (1) of Theorem 3.3.1.

for all $\lambda \in \mathbb{N}$, from which part (1) of the theorem follows.

The proof uses the game in Figure 3.7. This game switches to using simulated parameters and signatures. We will build A_1, D so that for all $\lambda \in \mathbb{N}$ we have

$$\Pr[\mathrm{IND}^A_{\mathsf{JES}}(\lambda)] - \Pr[\mathrm{G}^A_0(\lambda)] \leq \mathbf{Adv}^{\text{sim}}_{\mathsf{DS},A_1}(\lambda) \tag{3.7}$$

$$2\Pr[\mathrm{G}^A_0(\lambda)] - 1 \leq \mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKE},D}(\lambda)\ . \tag{3.8}$$

Using this we have

$$\begin{aligned}
\mathbf{Adv}^{\text{ind}}_{\mathsf{JES},A}(\lambda) &= 2\Pr[\mathrm{IND}^A_{\mathsf{JES}}(\lambda)] - 1 \\
&= 2\left(\Pr[\mathrm{IND}^A_{\mathsf{JES}}(\lambda)] - \Pr[\mathrm{G}^A_0(\lambda)] + \Pr[\mathrm{G}^A_0(\lambda)]\right) - 1 \\
&= 2\left(\Pr[\mathrm{IND}^A_{\mathsf{JES}}(\lambda)] - \Pr[\mathrm{G}^A_0(\lambda)]\right) + 2\Pr[\mathrm{G}^A_0(\lambda)] - 1 \\
&\leq 2\mathbf{Adv}^{\text{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKE},D}(\lambda)
\end{aligned}$$

as desired. We proceed to the constructions of A_1, D. Adversary A_1 behaves as follows:

$\underline{A_1^{\text{SIGN}}(1^\lambda, pp)}$

$(sk, pk) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)$

$C^* \leftarrow \perp\,;\ d \leftarrow\!\!\$\ \{0,1\}$

$d' \leftarrow\!\!\$\ A^{\text{DECSIM,SIGNSIM,LRSIM}}(1^\lambda, pp, pk)$

If $(d' = d)$ then $b' \leftarrow 1$

Else $b' \leftarrow 0$

Ret b'

$\underline{\text{SIGNSIM}(M)}$

$\sigma \leftarrow\!\!\$\ \text{SIGN}(sk, M)$

Ret σ

$\underline{\text{DECSIM}(C)}$

If $C = C^*$ then $M \leftarrow \perp$

Else $M \leftarrow \mathsf{JES.Dec}(1^\lambda, pp, sk, C)$

Ret M

$\underline{\text{LRSIM}(M_0, M_1)}$

If $(|M_0| \neq |M_1|)$ then Ret $\perp$

$C^* \leftarrow \mathsf{JES.Enc}(1^\lambda, pp, pk, M_d)$

Ret C^*

When the challenge bit b in game SIM is 0, adversary A_1 simulates for A game G_0, and if $b = 1$, adversary A_1 simulates game IND. We thus have

$$\Pr[\text{IND}^A_{\mathsf{JES}}(\lambda)] - \Pr[\text{G}^A_0(\lambda)] = \Pr[b' = 1 \,|\, b = 1] - \Pr[b' = 1 \,|\, b = 0] \leq \mathbf{Adv}^{\text{sim}}_{\mathsf{DS},A_1}(\lambda)\,,$$

establishing Equation 3.7. Adversary D behaves as follows:

MAIN $\mathrm{G}_0^A(\lambda)$ / $\boxed{\mathrm{G}_1^A(\lambda)}$
$Q \leftarrow \emptyset$; $d \leftarrow \mathsf{false}$
$fp \leftarrow\!\!\$\ \mathsf{F.Pg}(1^\lambda)$; $(ap, std, xtd) \leftarrow\!\!\$\ \mathsf{DS.SimPg}(1^\lambda)$; $jp \leftarrow (fp, ap)$
$(sk, pk) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, jp)$
$(M, \sigma) \leftarrow\!\!\$\ A^{\textsc{Sign},\textsc{Dec}}(1^\lambda, jp, pk)$; $sk' \leftarrow\!\!\$\ \mathsf{DS.Ext}(1^\lambda, jp, xtd, pk, M, \sigma)$
If ($\mathsf{DS.Ver}(1^\lambda, jp, pk, M, \sigma)$ and $(M, \sigma) \notin Q$) then
  $d \leftarrow \mathsf{true}$
  If ($\mathsf{F.Ev}(1^\lambda, fp, sk') \neq pk$) then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{d \leftarrow \mathsf{false}}$
Ret d

proc $\textsc{Sign}(M)$
$\sigma \leftarrow\!\!\$\ \mathsf{DS.SimSig}(1^\lambda, jp, std, pk, M)$
$Q \leftarrow Q \cup \{(M, \sigma)\}$
Ret σ

proc $\textsc{Dec}(C)$
Ret $M \leftarrow \mathsf{Dec}(1^\lambda, jp, sk, C)$

Figure 3.8. Games used in the proof of part (2) of Theorem 3.3.1. Game G_1 includes the boxed code and G_0 does not.

$D^{\textsc{Dec},\mathrm{LR}}(1^\lambda, fp, pk)$
$(ap, std, xtd) \leftarrow\!\!\$\ \mathsf{DS.SimPg}(1^\lambda)$; $jp \leftarrow (fp, ap)$
$b' \leftarrow\!\!\$\ A^{\textsc{Dec},\textsc{SignSim},\mathrm{LR}}(1^\lambda, jp, pk)$
Ret b'

$\textsc{SignSim}(M)$
$\sigma \leftarrow\!\!\$\ \mathsf{DS.SimSig}(1^\lambda, jp, std, pk, M)$
Ret σ

We omit the analysis establishing Equation 3.8.

Part (2): SUF security

Let A be a PT adversary playing game SUF. We build PT adversaries A_1, A_2, D such that

$$\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{JES},A}(\lambda) \leq \mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKE},D}(\lambda) + \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda)$$

for all $\lambda \in \mathbb{N}$, from which part (2) of the theorem follows.

The proof uses the games in Figure 3.8. These games switch to using simulated parameters and signatures. We will build A_1, A_2, D so that for all $\lambda \in \mathbb{N}$ we have

$$\Pr[\mathrm{SUF}^A_{\mathsf{JES}}(\lambda)] - \Pr[\mathrm{G}^A_0(\lambda)] \leq \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) \tag{3.9}$$

$$\Pr[\mathrm{G}^A_0(\lambda) \text{ sets } \mathsf{bad}] \leq \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda) \tag{3.10}$$

$$\Pr[\mathrm{G}^A_1(\lambda)] \leq \mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKE},D}(\lambda)\,. \tag{3.11}$$

Games G_0 and G_1 are identical until bad, so by the Fundamental Lemma of Game-Playing [25] and the above, for all $\lambda \in \mathbb{N}$ we have:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{JES},A}(\lambda) &= \Pr[\mathrm{SUF}^A_{\mathsf{JES}}(\lambda)] \\
&= (\Pr[\mathrm{SUF}^A_{\mathsf{JES}}(\lambda)] - \Pr[\mathrm{G}^A_0(\lambda)]) + (\Pr[\mathrm{G}^A_0(\lambda)] - \Pr[\mathrm{G}^A_1(\lambda)]) + \Pr[\mathrm{G}^A_1(\lambda)] \\
&\leq (\Pr[\mathrm{SUF}^A_{\mathsf{JES}}(\lambda)] - \Pr[\mathrm{G}^A_0(\lambda)]) + \Pr[\mathrm{G}^A_0(\lambda) \text{ sets } \mathsf{bad}] + \Pr[\mathrm{G}^A_1(\lambda)] \\
&\leq \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda) + \mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKE},D}(\lambda)
\end{aligned}$$

as desired. We proceed to the constructions of A_1, A_2, D. Adversary A_1 behaves as follows:

$\underline{A_1^{\text{SIGN}}(1^\lambda, pp)}$

$(sk, pk) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)\ ;\ Q \leftarrow \emptyset$

$(M, \sigma) \leftarrow\!\!\$\ A^{\text{DECSIM},\text{SIGNSIM}}(1^\lambda, pp, pk)$

If $(\mathsf{DS.Ver}(1^\lambda, pp, pk, M, \sigma) \wedge (M, \sigma) \notin Q)$ then $b' \leftarrow 1$

Else $b' \leftarrow 0$

Return b'

$\underline{\text{DECSIM}(C)}$

$M \leftarrow \mathsf{JES.Dec}(1^\lambda, pp, sk, C)$

Ret M

$\underline{\text{SIGNSIM}(M)}$

$Q \leftarrow Q \cup \{(M, \sigma)\}$

$\sigma \leftarrow\!\!\$\ \text{SIGN}(sk, M)$

Ret σ

When the challenge bit b in game SIM is 0, adversary A_1 simulates for A game G_0, and if $b = 1$, adversary A_1 simulates game SUF. We thus have

$$\Pr[\text{SUF}^A_{\mathsf{JES}}(\lambda)] - \Pr[\text{G}^A_0(\lambda)] = \Pr[b' = 1 \,|\, b = 1] - \Pr[b' = 1 \,|\, b = 0] \leq \mathbf{Adv}^{\text{sim}}_{\mathsf{DS},A_1}(\lambda)\ ,$$

establishing Equation 3.9. Adversary A_2 behaves as follows:

$\underline{A_2^{\text{SIGN}}(1^\lambda, pp)}$	$\underline{\text{DECSIM}(C)}$
$(sk, pk) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)$	$M \leftarrow \mathsf{JES.Dec}(1^\lambda, pp, sk, C)$
$(M, \sigma) \leftarrow\!\!\$\ A^{\text{DECSIM},\text{SIGNSIM}}(1^\lambda, pp, pk)$	Ret M
Ret (pk, M, σ)	$\underline{\text{SIGNSIM}(M)}$
	$\sigma \leftarrow\!\!\$\ \text{SIGN}(sk, M)$
	Ret σ

We omit the analysis establishing Equation 3.10. Adversary D behaves as follows:

$\underline{D^{\text{DEC,LR}}(1^\lambda, fp, pk)}$

$(ap, std, xtd) \leftarrow\!\!\$\ \mathsf{DS.SimPg}(1^\lambda)$; $jp \leftarrow (fp, ap)$

$(M, \sigma) \leftarrow\!\!\$\ A^{\text{DEC,SIGNSIM}}(1^\lambda, jp, pk)$

$sk' \leftarrow\!\!\$\ \mathsf{DS.Ext}(1^\lambda, pp, xtd, pk, M, \sigma)$

If $(\mathsf{F.Ev}(1^\lambda, fp, sk') \neq pk)$ then Ret 0

$M_0 \leftarrow 0^\lambda$; $M_1 \leftarrow 1^\lambda$

$C^* \leftarrow\!\!\$\ \mathrm{LR}(M_0, M_1)$

$M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, sk', C^*)$

If $(M = M_1)$ then $b' \leftarrow 1$ else $b' \leftarrow 0$

Ret b'

$\underline{\text{SIGNSIM}(M)}$

$\sigma \leftarrow\!\!\$\ \mathsf{DS.SimSig}(1^\lambda, jp, std, pk, M)$

Ret σ

When A wins G_1 we have that $\mathsf{F.Ev}(1^\lambda, fp, sk') = pk$, so sk' is a valid secret key for pk. By correctness of PKE, we then have $\mathsf{PKE.Dec}(1^\lambda, pp, sk', \mathsf{PKE.Enc}(1^\lambda, pp, pk, M_b)) = M_b$, where b is the challenge bit in game IND-CCA, so

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKE},D}(\lambda) = \Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0] \geq \Pr[\mathrm{G}^A_1(\lambda)] \ .$$

This is because the first term in the difference above is at least $\Pr[\mathrm{G}^A_1(\lambda)]$ and the second term is zero. This establishes Equation 3.11. □

3.4 KDM-secure storage

Services like Dropbox, Google Drive and Amazon S3 offer outsourced storage. Users see obvious benefits but equally obvious security concerns. We would like to secure this storage, even when messages (files needing to be stored) depend on the keys securing them. If privacy is the only concern, existing KDM-secure encryption schemes

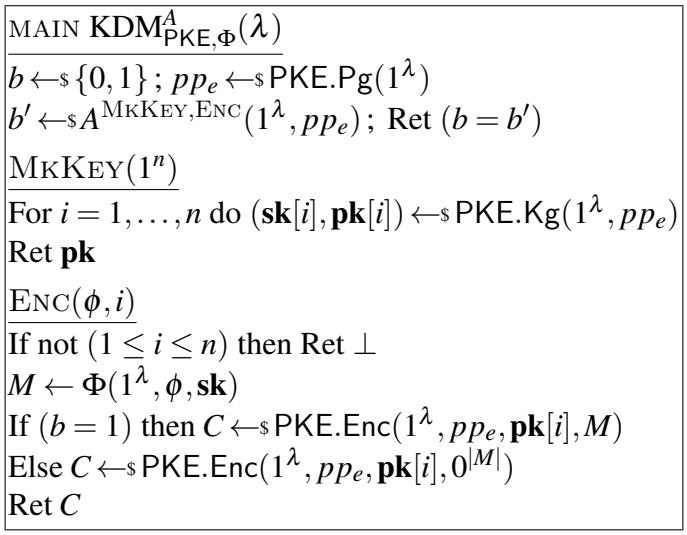


Figure 3.9. Game defining Φ-KDM security of a public-key encryption scheme PKE.

(e.g., [30, 38, 11, 10, 45, 47, 29, 13, 102, 9, 43, 44, 19, 76, 90]) will do the job. However, integrity is just as much of a concern, and adding it without losing KDM security is challenging. This is because conventional ways of adding integrity introduce new keys and create new ways for messages to depend on keys. Key-versatile signing, by leaving the keys unchanged, will provide a solution.

We begin below by formalizing our goal of encrypted and authenticated outsourced storage secure for key-dependent messages. In our syntax, the user encrypts and authenticates under her secret key, and then verifies and decrypts under the same secret key, with the public key utilized by the server for verification. Our requirement for KDM security has two components: IND for privacy and SUF for integrity. With the definitions in hand, we take a base KDM-secure encryption scheme and show how, via a key-versatile signature, to obtain storage schemes meeting our goal. Our resulting storage schemes will achieve KDM security with respect to the same class of message-deriving functions Φ as the underlying encryption scheme. Also, we will assume only CPA KDM security of the base scheme, yet achieve CCA KDM privacy for the constructed storage

scheme. Interestingly, our solution uses a *public-key* base encryption scheme, even though the privacy component of the goal is symmetric and nobody but the user will encrypt. This allows us to start with KDM privacy under keys permitting signatures through key-versatile signing. This represents a novel application for public-key KDM-secure encryption.

KDM security. A class of KDM functions Φ is a PT-computable function specifying for each $\lambda \in \mathbb{N}$ and each $\phi \in \{0,1\}^*$ a map $\Phi(1^\lambda, \phi, \cdot)$ called the message-deriving function described by ϕ. This map takes input a vector **sk** (of keys) and returns a string (the message) of length $\phi.m$, where $\phi.m \in \mathbb{N}$, the output length of ϕ, is computable in polynomial time given $1^\lambda, \phi$. We assume Φ always includes all constant functions. Formally, $\langle M \rangle$ describes the function defined by $\Phi(1^\lambda, \langle M \rangle, \mathbf{sk}) = M$ for all $M \in \{0,1\}^*$. We say that public-key encryption scheme PKE is Φ-KDM secure if $\mathbf{Adv}^{\mathrm{kdm}}_{\mathsf{PKE},A,\Phi}(\cdot)$ is negligible for every PT adversary A, where $\mathbf{Adv}^{\mathrm{kdm}}_{\mathsf{PKE},A,\Phi}(\lambda) = 2\Pr[\mathrm{KDM}^A_{\mathsf{PKE},\Phi}(\lambda)] - 1$ and game KDM is in Figure 3.9. We require that A make exactly one query to MKKEY and that this be its first oracle query. The argument $n \geq 1$ determines the number of keys and must be given in unary. The definition follows [30] except in parameterizing security by the class Φ of allowed message-deriving functions. The parameterization is important because many existing KDM-secure encryption schemes are for particular classes Φ, for example for encryption cycles, affine functions or cliques [38, 10, 43, 44, 90]. We aim to transfer whatever KDM security we have in the encryption to the secure storage, meaning we want to preserve Φ regardless of what it is. Of course a particularly interesting case is that of "full" security, but this is captured as the special case where Φ is all functions.

In the setting of direct practical interest, Alice arguably has just one key, corresponding to the vector **sk** above having just one component. However, as noted above, much of the literature on KDM security concerns itself with the encryption of cycles

and cliques, which represent message-deriving functions on multiple keys, and so our definitions allow the latter.

Secure storage schemes. A storage scheme ST specifies the following PT algorithms: via $pp \leftarrow\!\$\, \mathsf{ST.Pg}(1^\lambda)$ one generates public parameters pp common to all users; via $(sk,pk) \leftarrow\!\$\, \mathsf{ST.Kg}(1^\lambda, pp)$ a user can generate a secret key sk and corresponding public key pk; via $D \leftarrow\!\$\, \mathsf{ST.Store}(1^\lambda, pp, sk, M)$ a user can produce some data D based on $M \in \{0,1\}^*$ to store on the server; via $M \leftarrow \mathsf{ST.Retrieve}(1^\lambda, pp, sk, D)$ a user can deterministically retrieve $M \in \{0,1\}^* \cup \{\perp\}$ from their stored data D; and via $d \leftarrow \mathsf{ST.Verify}(1^\lambda, pp, pk, D)$ the server can deterministically produce a decision $d \in \{\mathsf{true}, \mathsf{false}\}$ regarding the validity of D. Correctness requires that $\mathsf{ST.Retrieve}(1^\lambda, pp, sk, \mathsf{ST.Store}(1^\lambda, pp, sk, M)) = M$ and $\mathsf{ST.Verify}(1^\lambda, pp, pk, \mathsf{ST.Store}(1^\lambda, pp, sk, M)) = \mathsf{true}$ for all $\lambda \in \mathbb{N}$, all $pp \in [\mathsf{ST.Pg}(1^\lambda)]$, all $(sk,pk) \in [\mathsf{ST.Kg}(1^\lambda, pp)]$, and all messages $M \in \{0,1\}^*$. Let Φ be a class of KDM functions as above. We say that ST is Φ-IND-secure if $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{ST},A,\Phi}(\cdot)$ is negligible for all PT adversaries A, where $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{ST},A,\Phi}(\lambda) = 2\Pr[\mathrm{IND}^A_{\mathsf{ST},\Phi}(\lambda)] - 1$ and game IND is on the left-hand side of Figure 3.10. The presence of the RETRIEVE oracle makes this a CCA KDM notion. We say that ST is Φ-SUF-secure if $\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{ST},A,\Phi}(\cdot)$ is negligible for all PT adversaries A, where $\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{ST},A,\Phi}(\lambda) = \Pr[\mathrm{SUF}^A_{\mathsf{ST},\Phi}(\lambda)]$ and game SUF is on the right-hand side of Figure 3.10. In both cases, we require that A make exactly one query to MKKEY and that this be its first oracle query, and again the argument $n \geq 1$, indicating the number of keys, must be given in unary.

Construction. The base scheme we take as given is a Φ-KDM secure, canonical public-key encryption scheme PKE. As in Section 3.3, we begin by constructing from PKE a function family F. We do not repeat this construction here, but refer the reader to

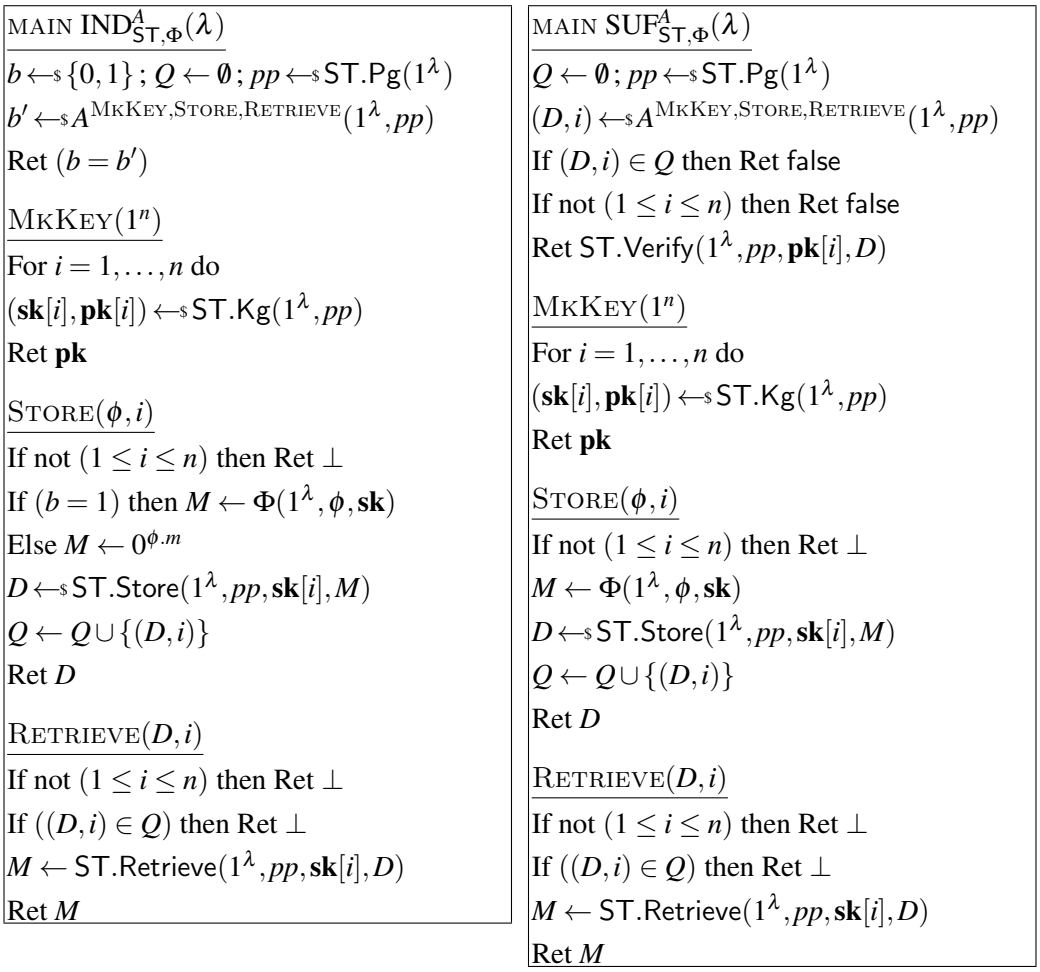


Figure 3.10. Games defining KDM-security of storage scheme ST: Privacy (left) and unforgeability (right).

Section 3.3. We then let DS be an F-keyed signature scheme that is simulatable and key-extractable. We construct our storage scheme ST as follows:

- ST.Pg(λ): Return $(fp, ap) \leftarrow\$ \mathsf{DS.Pg}(1^\lambda)$. Thus, parameters for ST have the form $pp = (fp, ap)$, where fp are parameters for both F and PKE.
- ST.Kg($1^\lambda, (fp, ap)$): Return $(sk, pk) \leftarrow\$ \mathsf{DS.Kg}(1^\lambda, (fp, ap))$. Thus, keys are those of PKE and DS.

- $\underline{\mathsf{ST.Store}(1^\lambda,(fp,ap),sk,M)}$: $pk \leftarrow \mathsf{F.Ev}(1^\lambda,fp,sk)$; $C \leftarrow\!\!\$\ \mathsf{PKE.Enc}(1^\lambda,fp,pk,M)$; $\sigma \leftarrow\!\!\$\ \mathsf{DS.Sig}(1^\lambda,pp,sk,C)$. Return (C,σ).
- $\underline{\mathsf{ST.Retrieve}(1^\lambda,(fp,ap),sk,(C,\sigma))}$: $pk \leftarrow \mathsf{F.Ev}(1^\lambda,fp,sk)$. If $\mathsf{DS.Ver}(1^\lambda,pp,pk,C,\sigma) = \mathsf{false}$ then return $\perp$. Else return $\mathsf{PKE.Dec}(1^\lambda,fp,sk,C)$.
- $\underline{\mathsf{ST.Verify}(1^\lambda,(fp,ap),pk,(C,\sigma))}$: Return $\mathsf{DS.Ver}(1^\lambda,(fp,ap),pk,C,\sigma)$.

The following says that our construction provides both privacy and integrity for key-dependent messages, assuming privacy for key-dependent messages of the base encryption scheme and simulatability and key-extractability of the F-keyed signature scheme:

Theorem 3.4.1. *Let* PKE *be a canonical public-key encryption scheme, and let* F *be defined from it as above. Let* DS *be an* F*-keyed signature scheme, and let* ST *be the corresponding storage scheme constructed above. Let* Φ *be a class of message-deriving functions. Assume* PKE *is* Φ*-KDM secure. Assume* DS *is simulatable and key-extractable. Then (1)* ST *is* Φ*-IND secure and (2)* ST *is* Φ*-SUF secure.*

Before we present the full proof of this theorem, we provide sketches to highlight some of the unusual difficulties. Taking first the proof of privacy, we would like, given an adversary A breaking the Φ-IND security of ST, to build an adversary D breaking the assumed Φ-KDM security of PKE. The first problem is how D can create the signatures needed to answer STORE queries of A, since these rely on secret keys hidden from D. We solve this by switching to simulation parameters, so that D can simulate signatures without a secret key. In answering RETRIEVE queries, however, we run into another problem: the assumed KDM security of PKE is only under CPA. To solve this, we use the extractor to extract the secret key from signatures and decrypt under it. The full proof involves building simulation and extractability adversaries in addition to D.

Turning next to the proof of unforgeability, we might at first expect that it relies on nothing more than the unforgeability of the signature scheme, so that given an adversary

A breaking the Φ-SUF security of ST we could build an adversary breaking the SUF security of DS. However, we run into the basic issue that, since the same keys are used for signing, encryption, and decryption, an adversary against the unforgeability of the signature scheme cannot even construct the messages (ciphertexts) on which A would forge. Instead, we will build from A an adversary D breaking the Φ-KDM security of PKE. This adversary will extract a secret key from a forgery of A and use this to break privacy. To get D to work we must first, as above, switch to simulated signatures, and then use extractability to switch to a simpler RETRIEVE oracle.

Proof. Part (1): IND security

Let A be a PT adversary playing game IND. Let $q(\cdot)$ be a polynomial such that the number of RETRIEVE queries of A in game $\mathrm{IND}^A_{\mathsf{ST},\Phi}(\lambda)$ is $q(\lambda)$ for all $\lambda \in \mathbb{N}$. We provide PT adversaries A_1, A_2, D and a negligible function $\nu(\cdot)$ such that

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{ST},A,\Phi}(\lambda) \leq 2\mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\mathrm{kdm}}_{\mathsf{PKE},D,\Phi}(\lambda) + 2\nu(\lambda) + 2q(\lambda)\cdot\mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda)$$

for all $\lambda \in \mathbb{N}$, from which part (1) of the theorem follows.

The proof uses the games in Figure 3.11. We build A_1, A_2, D and $\nu(\cdot)$ such that for all $\lambda \in \mathbb{N}$ we have

$$\Pr[\mathrm{IND}^A_{\mathsf{ST},\Phi}(\lambda)] - \Pr[\mathrm{G}^A_0(\lambda)] \leq \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) \tag{3.12}$$

$$2\Pr[\mathrm{G}^A_1(\lambda)] - 1 \leq \mathbf{Adv}^{\mathrm{kdm}}_{\mathsf{PKE},D,\Phi}(\lambda) \tag{3.13}$$

$$\Pr[\mathrm{G}^A_2(\lambda) \text{ sets } \mathsf{bad}] \leq \nu(\lambda) + q(\lambda)\cdot\mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda)\,. \tag{3.14}$$

Games $\mathrm{G}_0, \mathrm{G}_1$ are identical until bad. We also observe that $\Pr[\mathrm{G}^A_0(\lambda) \text{ sets } \mathsf{bad}] \leq \Pr[\mathrm{G}^A_2(\lambda) \text{ sets } \mathsf{bad}]$. (In both games, decryption in RETRIEVE is always done correctly.) Combining this with the above and the Fundamental Lemma of Game-Playing [25], for

all $\lambda \in \mathbb{N}$ we have:

$$
\begin{aligned}
\mathbf{Adv}^{\text{ind}}_{\mathsf{ST},A,\Phi}(\lambda) &= 2\Pr[\text{IND}^{A}_{\mathsf{ST},\Phi}(\lambda)] - 1 \\
&= 2\left(\Pr[\text{IND}^{A}_{\mathsf{ST},\Phi}(\lambda) - \Pr[\text{G}^{A}_{0}(\lambda)]\right) + 2\left(\Pr[\text{G}^{A}_{0}(\lambda)] - \Pr[\text{G}^{A}_{1}(\lambda)]\right) + \\
&\quad 2\Pr[\text{G}^{A}_{1}(\lambda)] - 1 \\
&\leq 2\left(\Pr[\text{IND}^{A}_{\mathsf{ST},\Phi}(\lambda) - \Pr[\text{G}^{A}_{0}(\lambda)]\right) + 2\Pr[\text{G}^{A}_{0}(\lambda) \text{ sets } \mathsf{bad}] + \\
&\quad 2\Pr[\text{G}^{A}_{1}(\lambda)] - 1 \\
&\leq 2\mathbf{Adv}^{\text{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\text{kdm}}_{\mathsf{PKE},D,\Phi}(\lambda) + 2\Pr[\text{G}^{A}_{0}(\lambda) \text{ sets } \mathsf{bad}] \\
&\leq 2\mathbf{Adv}^{\text{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\text{kdm}}_{\mathsf{PKE},D,\Phi}(\lambda) + 2\Pr[\text{G}^{A}_{2}(\lambda) \text{ sets } \mathsf{bad}] \\
&\leq 2\mathbf{Adv}^{\text{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\text{kdm}}_{\mathsf{PKE},D,\Phi}(\lambda) + 2\nu(\lambda) + 2q(\lambda) \cdot \mathbf{Adv}^{\text{ext}}_{\mathsf{DS},A_2}(\lambda)
\end{aligned}
$$

as desired. We proceed to the constructions of A_1, A_2, D and ν.

Adversary A_1 behaves as follows:

$\underline{A_1^{\text{SIGN}}(1^\lambda, pp)}$

$Q \leftarrow \emptyset$; $d \leftarrow\!\!\$\ \{0,1\}$

$d' \leftarrow\!\!\$\ A^{\text{MKKEYSIM},\text{STORESIM},\text{RETRIEVESIM}}(1^\lambda, pp)$

If $(d' = d)$ then $b' \leftarrow 1$ else $b' \leftarrow 0$

Ret b'

$\underline{\text{MKKEYSIM}(1^n)}$

For $i = 1, \ldots, n$ do

  $(\mathbf{sk}[i], \mathbf{pk}[i]) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)$

Ret $\mathbf{pk}$

$\underline{\text{RETRIEVESIM}((C,\sigma), i)}$

If not $(1 \le i \le n)$ then Ret $\perp$

If (not $\mathsf{DS.Ver}(1^\lambda, pp, \mathbf{pk}[i], C, \sigma)$) then Ret $\perp$

If $((C,\sigma), i) \in Q$ then Ret $\perp$

$M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, \mathbf{sk}[i], C)$

Ret M

$\underline{\text{STORESIM}(\phi, i)}$

If not $(1 \le i \le n)$ then Ret $\perp$

If $d = 1$ then $M \leftarrow \Phi(1^\lambda, \phi, \mathbf{sk})$

Else $M \leftarrow 0^{\phi.m}$

$C \leftarrow\!\!\$\ \mathsf{PKE.Enc}(1^\lambda, fp, \mathbf{pk}[i], M)$

$\sigma \leftarrow\!\!\$\ \text{SIGN}(\mathbf{sk}[i], C)$

$Q \leftarrow Q \cup \{((C,\sigma), i)\}$

Ret (C,σ)

When the challenge bit b in game SIM is 1, adversary A_1 simulates IND. We claim that if $b = 0$ then A_1 simulates G_0. This is because in G_0, procedure RETRIEVE always performs the correct decryption, regardless of whether or not bad is set, and so does A_1. (A_1 does not need to invoke the extractor, and indeed could not, since it does not have an extraction trapdoor.) We thus have

$$\Pr[\text{IND}^A_{\mathsf{ST},\Phi}(\lambda)] - \Pr[\text{G}^A_0(\lambda)] = \Pr[b' = 1 \,|\, b = 1] - \Pr[b' = 1 \,|\, b = 0] \le \mathbf{Adv}^{\text{sim}}_{\mathsf{DS},A_1}(\lambda),$$

establishing Equation 3.12.

Adversary D behaves as follows:

$\underline{D^{\text{MkKey},\text{Enc}}(1^\lambda, fp)}$

$Q \leftarrow \emptyset$

$(ap, std, xtd) \leftarrow\$ \mathsf{DS.SimPg}(1^\lambda)$

$pp \leftarrow (fp, ap)$

$b' \leftarrow\$ A^{\text{MkKey},\text{StoreSim},\text{RetrieveSim}}(1^\lambda, pp)$

Ret b'

$\underline{\text{StoreSim}(\phi, i)}$

If not $(1 \le i \le n)$ then Ret $\perp$

$C \leftarrow\$ \text{Enc}(\phi, i)$

$\sigma \leftarrow\$ \mathsf{DS.SimSig}(1^\lambda, pp, std, \mathbf{pk}[i], C)$

$Q \leftarrow Q \cup \{((C, \sigma), i)\}$

Ret (C, σ)

$\underline{\text{RetrieveSim}((C, \sigma), i)}$

If not $(1 \le i \le n)$ then Ret $\perp$

If (not $\mathsf{DS.Ver}(1^\lambda, pp, \mathbf{pk}[i], C, \sigma)$) then Ret $\perp$

If $((C, \sigma), i) \in Q$ then Ret $\perp$

$sk' \leftarrow\$ \mathsf{DS.Ext}(1^\lambda, pp, xtd, \mathbf{pk}[i], C, \sigma)$

$M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, sk', C)$

Ret M

We omit the analysis to establish Equation 3.13.

Now we would like to show that G_0 (equivalently, G_1) sets bad with negligible probability. Intuitively, this should follow from extractability, since bad is set when extraction fails. A difficulty is that extraction is not required to succeed when $(pk[i], C, \sigma) \in Q'$. So first we show that the latter event is unlikely, and then, assuming it does not happen, that failure of extraction is unlikely. This is formalized via G_2, which breaks the setting of

bad from G_0 into two parts corresponding to the two events of interest. To establish Equation 3.17, let E_1 be the event that bad is set by the line 5 "If" statement, and E_2 the event that bad is set by the line 7 "If" statement. We first show the existence of a negligible $\nu(\cdot)$ such that $\Pr[E_1] \leq \nu(\lambda)$. Then we build A_2 such that $\Pr[E_2 \wedge \overline{E}_1] \leq q(\lambda) \cdot \mathbf{Adv}^{\text{ext}}_{\mathsf{DS},A_2}(\lambda)$. This establishes Equation 3.14, as we have

$$\begin{aligned}\Pr[\mathrm{G}_2^A(\lambda) \text{ sets bad}] &= \Pr[E_1 \vee E_2] \\ &= \Pr[E_1] + \Pr[E_2 \wedge \overline{E}_1] \\ &\leq \nu(\lambda) + q(\lambda) \cdot \mathbf{Adv}^{\text{ext}}_{\mathsf{DS},A_2}(\lambda).\end{aligned}$$

For the first claim, let E be the event that there is a collision in the public keys chosen in MKKEY, meaning there are distinct $i, j \in \{1, \ldots, n\}$ such that $\mathbf{pk}[i] = \mathbf{pk}[j]$. We claim that if this event does not happen, then neither will E_1. This is because setting bad requires that $(\mathbf{pk}[i], C, \sigma) \in Q'$ yet $((C, \sigma), i) \notin Q$, but this cannot happen if the public keys are all distinct. So $\Pr[E_1] \leq \Pr[E]$. However, if E does happen with probability that is not negligible, then it is easy to break KDM security of PKE. An adversary just has to itself sample key-pairs, hoping to get one where the public key matches one of her challenge public keys. In that case, having the corresponding secret key, it is easy to defeat security. We omit the details because this argument is standard.

Adversary A_2 behaves as follows:

$\underline{A_2^{\text{SIGN}}(1^\lambda, pp)}$

$Q \leftarrow \emptyset$; $d \leftarrow\$ \{0,1\}$; $j \leftarrow 0$

$d' \leftarrow\$ A^{\text{MKKEYSIM},\text{STORESIM},\text{RETRIEVESIM}}(1^\lambda, pp)$

$\ell \leftarrow\$ \{1,\ldots,j\}$

Ret $(pk_\ell, C_\ell, \sigma_\ell)$

$\underline{\text{MKKEYSIM}(1^n)}$

For $i = 1,\ldots,n$ do

  $(\mathbf{sk}[i], \mathbf{pk}[i]) \leftarrow\$ \mathsf{DS.Kg}(1^\lambda, pp)$

Ret $\mathbf{pk}$

$\underline{\text{RETRIEVESIM}((C,\sigma), i)}$

If not $(1 \le i \le n)$ then Ret $\perp$

If (not $\mathsf{DS.Ver}(1^\lambda, pp, \mathbf{pk}[i], C, \sigma)$) then Ret $\perp$

If $((C,\sigma), i) \in Q$ then Ret $\perp$

$M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, \mathbf{sk}[i], C)$

$j \leftarrow j+1$; $pk_j \leftarrow \mathbf{pk}[i]$; $C_j \leftarrow C$; $\sigma_j \leftarrow \sigma$

Ret M

$\underline{\text{STORESIM}(\phi, i)}$

If not $(1 \le i \le n)$ then Ret $\perp$

If $d = 1$ then $M \leftarrow \Phi(1^\lambda, \phi, \mathbf{sk})$

Else $M \leftarrow 0^{\phi.m}$

$C \leftarrow\$ \mathsf{PKE.Enc}(1^\lambda, fp, \mathbf{pk}[i], M)$

$\sigma \leftarrow\$ \text{SIGN}(\mathbf{sk}[i], C)$

$Q \leftarrow Q \cup \{((C,\sigma), i)\}$

Ret (C,σ)

Adversary A_2 always performs correct decryptions in responding to RETRIEVE queries, following G_2. If bad is set at line 7 but not at line 5, then there is some tuple on which the extractor would succeed. Since a tuple is guessed at random we have $\Pr[E_2 \wedge \overline{E}_1] \le q(\lambda) \cdot \mathbf{Adv}^{\text{ext}}_{\mathsf{DS},A_2}(\lambda)$ as desired. The importance of bad not being set at line 5 is that otherwise extraction is not required to succeed according to game EXT.

Part (2): SUF security

Let A' be a PT adversary playing game SUF. Our first step is to consider a simplified form of the SUF game shown in Figure 3.12. Here the adversary does not output a forgery but instead wins via RETRIEVE queries. We can easily transform A' into

a PT adversary A such that $\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{ST},A',\Phi}(\lambda) \leq \Pr[\mathrm{H}^{A}(\lambda)]$ for all $\lambda \in \mathbb{N}$. Adversary A simply runs A', answering all queries via its own oracles (the two adversaries have the same oracles). When A' halts with output $((C,\sigma),i)$, A makes query $\textsc{Retrieve}((C,\sigma),i)$ and halts with output $\perp$. The flag win is set to true with at least the probability that A' wins its game. Now we proceed to upper bound $\Pr[\mathrm{H}^{A}(\lambda)]$.

Let $q(\cdot)$ be a polynomial such that the number of $\textsc{Retrieve}$ queries of A in game $\mathrm{H}^{A}(\lambda)$ is $q(\lambda)$ for all $\lambda \in \mathbb{N}$. We provide PT adversaries A_1, A_2, D and a negligible function $\nu(\cdot)$ such that

$$\Pr[\mathrm{H}^{A}(\lambda)] \leq \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\mathrm{kdm}}_{\mathsf{PKE},D,\Phi}(\lambda) + \nu(\lambda) + q(\lambda) \cdot \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda)$$

for all $\lambda \in \mathbb{N}$, from which part (2) of the theorem follows.

The proof uses the games in Figure 3.13. We build A_1, A_2, D and ν such that for all $\lambda \in \mathbb{N}$ we have

$$\Pr[\mathrm{H}^{A}(\lambda)] - \Pr[\mathrm{G}^{A}_{0}(\lambda)] \leq \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) \tag{3.15}$$

$$\Pr[\mathrm{G}^{A}_{1}(\lambda) \wedge \overline{\mathsf{bad}}] \leq \mathbf{Adv}^{\mathrm{kdm}}_{\mathsf{PKE},D,\Phi}(\lambda) \tag{3.16}$$

$$\Pr[\mathrm{G}^{A}_{2}(\lambda) \text{ sets } \mathsf{bad}] \leq \nu(\lambda) + q(\lambda) \cdot \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda)\ . \tag{3.17}$$

The notation in Equation 3.16 means that we are considering the event that the game returns true and also bad is not set. Now games $\mathrm{G}_0, \mathrm{G}_1$ are identical until bad so a variant of the Fundamental Lemma of Game-Playing [25] says that $\Pr[\mathrm{G}^{A}_{0}(\lambda) \wedge \overline{\mathsf{bad}}] = \Pr[\mathrm{G}^{A}_{1}(\lambda) \wedge \overline{\mathsf{bad}}]$. We also observe that $\Pr[\mathrm{G}^{A}_{0}(\lambda) \text{ sets } \mathsf{bad}] \leq \Pr[\mathrm{G}^{A}_{2}(\lambda) \text{ sets } \mathsf{bad}]$. (In both games, decryption in $\textsc{Retrieve}$ is always done correctly.) Combining this with

the above, for all $\lambda \in \mathbb{N}$ we have:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{ST},A',\Phi}(\lambda) &\leq \Pr[\mathrm{H}^{A}(\lambda)] \\
&= \Pr[\mathrm{H}^{A}(\lambda)] - \Pr[\mathrm{G}^{A}_{0}(\lambda)] + \Pr[\mathrm{G}^{A}_{0}(\lambda)] \\
&\leq \Pr[\mathrm{H}^{A}(\lambda)] - \Pr[\mathrm{G}^{A}_{0}(\lambda)] + \Pr[\mathrm{G}^{A}_{0}(\lambda) \wedge \overline{\mathsf{bad}}] + \Pr[\mathrm{G}^{A}_{0}(\lambda) \text{ sets } \mathsf{bad}] \\
&\leq \Pr[\mathrm{H}^{A}(\lambda)] - \Pr[\mathrm{G}^{A}_{0}(\lambda)] + \Pr[\mathrm{G}^{A}_{1}(\lambda) \wedge \overline{\mathsf{bad}}] + \Pr[\mathrm{G}^{A}_{0}(\lambda) \text{ sets } \mathsf{bad}] \\
&\leq \Pr[\mathrm{H}^{A}(\lambda)] - \Pr[\mathrm{G}^{A}_{0}(\lambda)] + \Pr[\mathrm{G}^{A}_{1}(\lambda) \wedge \overline{\mathsf{bad}}] + \Pr[\mathrm{G}^{A}_{2}(\lambda) \text{ sets } \mathsf{bad}] \\
&\leq \mathbf{Adv}^{\mathrm{sim}}_{\mathsf{DS},A_1}(\lambda) + \mathbf{Adv}^{\mathrm{kdm}}_{\mathsf{PKE},D}(\lambda) + \nu(\lambda) + q(\lambda) \cdot \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda)
\end{aligned}$$

as desired. We proceed to the constructions of A_1, A_2, D and ν.

Adversary A_1 behaves as follows:

$\underline{A_1^{\text{SIGN}}(1^\lambda, pp)}$

$Q \leftarrow \emptyset$

$\perp \leftarrow\!\!\$\ A^{\text{MKKEYSIM},\text{STORESIM},\text{RETRIEVESIM}}(1^\lambda, pp)$

If win then $b' \leftarrow 1$ else $b' \leftarrow 0$

Ret b'

$\underline{\text{MKKEYSIM}(1^n)}$

For $i = 1, \ldots, n$ do

  $(\mathbf{sk}[i], \mathbf{pk}[i]) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)$

Ret $\mathbf{pk}$

$\underline{\text{RETRIEVESIM}((C, \sigma), i)}$

If not $(1 \le i \le n)$ then Ret $\perp$

If $((C, \sigma), i) \in Q$ then Ret $\perp$

If $\mathsf{DS.Ver}(1^\lambda, pp, \mathbf{pk}[i], C, \sigma)$ then win $\leftarrow$ true;

Else Ret $\perp$

$M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, \mathbf{sk}[i], C)$

Ret M

$\underline{\text{STORESIM}(\phi, i)}$

If not $(1 \le i \le n)$ then Ret $\perp$

$M \leftarrow \Phi(1^\lambda, \phi, \mathbf{sk})$

$C \leftarrow\!\!\$\ \mathsf{PKE.Enc}(1^\lambda, fp, \mathbf{pk}[i], M)$

$\sigma \leftarrow\!\!\$\ \text{SIGN}(\mathbf{sk}[i], C)$

$Q \leftarrow Q \cup \{((C, \sigma), i)\}$

Ret (C, σ)

When the challenge bit b in game SIM is 1, adversary A_1 simulates H, and when $b = 0$ it simulates G_0. Note that in the latter, decryptions done by RETRIEVE are always correct, so A_1 does not need to invoke the extractor. (Indeed, it could not, since it does not have an extraction trapdoor.) This establishes Equation 3.15.

Adversary D behaves as follows:

$\underline{D^{\text{MkKey},\text{Enc}}(1^\lambda, fp)}$

$Q \leftarrow \emptyset$; $sk^* \leftarrow \perp$; $j \leftarrow \perp$

$(ap, std, xtd) \leftarrow\$ \mathsf{DS.SimPg}(1^\lambda)$

$pp \leftarrow (fp, ap)$

$\perp \leftarrow\$ A^{\text{MkKey},\text{StoreSim},\text{RetrieveSim}}(1^\lambda, pp)$

If $(sk^*, j) = (\perp, \perp)$ then Ret 0

$M_1 \leftarrow 1^\lambda$; $C^* \leftarrow\$ \text{Enc}(\langle M_1 \rangle, j)$

$M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, sk^*, C^*)$

If $(M = M_1)$ then $b' \leftarrow 1$ else $b' \leftarrow 0$

Ret b'

$\underline{\text{StoreSim}(\phi, i)}$

If not $(1 \leq i \leq n)$ then Ret $\perp$

$C \leftarrow\$ \text{Enc}(\phi, i)$

$\sigma \leftarrow\$ \mathsf{DS.SimSig}(1^\lambda, pp, std, \mathbf{pk}[i], C)$

$Q \leftarrow Q \cup \{((C, \sigma), i)\}$

Ret (C, σ)

$\underline{\text{RetrieveSim}((C, \sigma), i)}$

If not $(1 \leq i \leq n)$ then Ret $\perp$

If $((C, \sigma), i) \in Q$ then Ret $\perp$

If (not $\mathsf{DS.Ver}(1^\lambda, pp, \mathbf{pk}[i], C, \sigma)$) then Ret $\perp$

$sk' \leftarrow\$ \mathsf{DS.Ext}(1^\lambda, pp, xtd, \mathbf{pk}[i], C, \sigma)$

If $(\mathsf{F.Ev}(1^\lambda, fp, sk') = \mathbf{pk}[i])$ then $(sk^*, j) \leftarrow (sk', i)$

$M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, sk', C)$

Ret M

Recall that $\langle M_1 \rangle$ denotes the constant function that always returns M_1. If win is set in G_1 then we are assured that there is at least one RETRIEVE query which leads to extraction being performed. If additionally bad is not set then this extraction succeeds, which means decryption under sk^* will be correct. That in turn means that M is the correct decryption of C^* and hence D succeeds if $\mathsf{win} \wedge \overline{\mathsf{bad}}$. This establishes Equation 3.16.

Now we would like to show that G_0 (equivalently, G_1) sets bad with negligible probability. Intuitively, this should follow from extractability, since bad is set when extraction fails. A difficulty is that extraction is not required to succeed when $(pk[i], C, \sigma) \in Q'$. So first we show that the latter event is unlikely, and then, assuming it does not happen, that failure of extraction is unlikely. This is formalized via G_2, which breaks the setting of bad from G_0 into two parts corresponding to the two events of interest. To establish Equation 3.17, let E_1 be the event that bad is set by the line 5 "If" statement, and E_2 the event that bad is set by the line 7 "If" statement. We show the existence of a negligible $\nu(\cdot)$ such that $\Pr[E_1] \leq \nu(\lambda)$ as in the proof of part (1) above, first arguing that $\Pr[E_1]$ is at most the probability of a collision in public keys, and then arguing that this is negligible by the assumed security of PKE. To establish Equation 3.17, we now build A_2 so that $\Pr[E_2 \wedge \overline{E}_1] \leq q(\lambda) \cdot \mathbf{Adv}^{\mathrm{ext}}_{\mathsf{DS},A_2}(\lambda)$:

$\underline{A_2^{\text{SIGN}}(1^\lambda, pp)}$

$Q \leftarrow \emptyset$; $j \leftarrow 0$

$\perp \leftarrow\!\!\$\ A^{\text{MKKEYSIM},\text{STORESIM},\text{RETRIEVESIM}}(1^\lambda, pp)$

$\ell \leftarrow\!\!\$\ \{1,\ldots,j\}$

Ret $(pk_\ell, C_\ell, \sigma_\ell)$

$\underline{\text{MKKEYSIM}(1^n)}$

For $i = 1,\ldots,n$ do

  $(\mathbf{sk}[i], \mathbf{pk}[i]) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)$

Ret $\mathbf{pk}$

$\underline{\text{RETRIEVESIM}((C,\sigma), i)}$

If not $(1 \le i \le n)$ then Ret $\perp$

If $((C,\sigma), i) \in Q$ then Ret $\perp$

If (not $\mathsf{DS.Ver}(1^\lambda, pp, \mathbf{pk}[i], C, \sigma)$) then Ret $\perp$

$M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, \mathbf{sk}[i], C)$

$j \leftarrow j+1$; $pk_j \leftarrow \mathbf{pk}[i]$; $C_j \leftarrow C$; $\sigma_j \leftarrow \sigma$

Ret M

$\underline{\text{STORESIM}(\phi, i)}$

If not $(1 \le i \le n)$ then Ret $\perp$

$M \leftarrow \Phi(1^\lambda, \phi, \mathbf{sk})$

$C \leftarrow\!\!\$\ \mathsf{PKE.Enc}(1^\lambda, fp, \mathbf{pk}[i], M)$

$\sigma \leftarrow\!\!\$\ \text{SIGN}(\mathbf{sk}[i], C)$

$Q \leftarrow Q \cup \{((C,\sigma), i)\}$

Ret (C, σ)

Adversary A_2 always performs correct decryptions in responding to RETRIEVE queries, following G_2. If bad is set at line 7 but not at line 5, then there is some tuple on which the extractor would succeed. Since a tuple is guessed at random we have $\Pr[E_2 \wedge \overline{E}_1] \le q(\lambda) \cdot \mathbf{Adv}^{\text{ext}}_{\mathsf{DS},A_2}(\lambda)$ as desired. The importance of bad not being set at line 5 is that otherwise extraction is not required to succeed according to game EXT. □

Instantiation. We require our base scheme PKE to be canonical. In Section 3.3 we showed how to modify an encryption scheme to be canonical while preserving IND-CCA, but the transformation does not in general preserve KDM security. Instead, we would use

KDM-secure schemes that are already canonical. One possibility is the scheme of [102]. The schemes of [38, 13, 10] are not canonical.

Acknowledgments

Chapter 3, in part, is a reprint of the material as it appears in *Advances in Cryptology - EUROCRYPT '14*. Mihir Bellare, Sarah Meiklejohn, Susan Thomson, Springer Lecture Notes in Computer Science, volume 8441, 2014. The dissertation author was a primary investigator and author of this paper.

$\underline{\text{MAIN } \boxed{\mathrm{G}_0^A(\lambda)} \,/\, \mathrm{G}_1^A(\lambda) \,/\, \mathrm{G}_2^A(\lambda)}$
$Q \leftarrow \emptyset\,;\ Q' \leftarrow \emptyset\,;\ b \leftarrow\!\!\$\ \{0,1\}$
$fp \leftarrow\!\!\$\ \mathsf{F.Pg}(1^\lambda)\,;\ (ap, std, xtd) \leftarrow\!\!\$\ \mathsf{DS.SimPg}(1^\lambda)\,;\ pp \leftarrow (fp, ap)$
$b' \leftarrow\!\!\$\ A^{\text{MkKey},\text{Store},\text{Retrieve}}(1^\lambda, pp)$
Ret $(b = b')$

$\underline{\text{MkKey}(1^n)} \qquad /\!/\ \mathrm{G}_0^A(\lambda) \,/\, \mathrm{G}_1^A(\lambda) \,/\, \mathrm{G}_2^A(\lambda)$
For $i = 1, \ldots, n$ do $(\mathbf{sk}[i], \mathbf{pk}[i]) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)$
Ret $\mathbf{pk}$

$\underline{\text{proc Store}(\phi, i)} \qquad /\!/\ \mathrm{G}_0^A(\lambda) \,/\, \mathrm{G}_1^A(\lambda) \,/\, \mathrm{G}_2^A(\lambda)$
If not $(1 \le i \le n)$ then return $\perp$
If $(b = 1)$ then $M \leftarrow \Phi(1^\lambda, \phi, \mathbf{sk})$ else $M \leftarrow 0^{\phi.m}$
$C \leftarrow\!\!\$\ \mathsf{PKE.Enc}(1^\lambda, fp, \mathbf{pk}[i], M)\,;\ \sigma \leftarrow\!\!\$\ \mathsf{DS.SimSig}(1^\lambda, pp, std, \mathbf{pk}[i], C)$
$Q \leftarrow Q \cup \{((C, \sigma), i)\}\,;\ Q' \leftarrow Q' \cup \{(\mathbf{pk}[i], C, \sigma)\}$
Ret (C, σ)

$\underline{\text{proc Retrieve}((C, \sigma), i)} \qquad /\!/\ \boxed{\mathrm{G}_0^A(\lambda)} \,/\, \mathrm{G}_1^A(\lambda)$
If not $(1 \le i \le n)$ then return $\perp$
If (not $\mathsf{DS.Ver}(1^\lambda, pp, \mathbf{pk}[i], C, \sigma)$) then Ret $\perp$
If $(((C, \sigma), i) \in Q)$ then Ret $\perp$
$sk' \leftarrow\!\!\$\ \mathsf{DS.Ext}(1^\lambda, pp, xtd, \mathbf{pk}[i], C, \sigma)\,;\ pk' \leftarrow \mathsf{F.Ev}(1^\lambda, fp, sk')$
$M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, sk', C)$
If $(pk' \neq \mathbf{pk}[i])$ then $\mathsf{bad} \leftarrow \mathsf{true}\,;\ \boxed{M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, \mathbf{sk}[i], C)}$
Ret M

$\underline{\text{proc Retrieve}((C, \sigma), i)} \qquad /\!/\ \mathrm{G}_2^A(\lambda)$
If not $(1 \le i \le n)$ then return $\perp$
If (not $\mathsf{DS.Ver}(1^\lambda, pp, \mathbf{pk}[i], C, \sigma)$) then Ret $\perp$
If $(((C, \sigma), i) \in Q)$ then Ret $\perp$
$M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, \mathbf{sk}[i], C)$
If $(\mathbf{pk}[i], C, \sigma) \in Q'$ then $\mathsf{bad} \leftarrow \mathsf{true}\,;$ Ret M
$sk' \leftarrow\!\!\$\ \mathsf{DS.Ext}(1^\lambda, pp, xtd, \mathbf{pk}[i], C, \sigma)\,;\ pk' \leftarrow \mathsf{F.Ev}(1^\lambda, fp, sk')$
If $(pk' \neq \mathbf{pk}[i])$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Ret M

Figure 3.11. Games used in the proof of part (1) of Theorem 3.4.1. Game G_0 includes the boxed code and game G_1 does not.

$\underline{\text{MAIN } \mathrm{H}^{A}(\lambda)}$
$Q \leftarrow \emptyset$; win $\leftarrow$ false; $(fp, ap) \leftarrow\!\!\$\ \mathsf{DS.Pg}(1^{\lambda})$; $pp \leftarrow (fp, ap)$
$\perp \leftarrow\!\!\$\ A^{\text{MkKey,Store,Retrieve}}(1^{\lambda}, pp)$
Ret win

$\underline{\text{MkKey}(1^{n})}$
For $i = 1, \ldots, n$ do $(\mathbf{sk}[i], \mathbf{pk}[i]) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^{\lambda}, pp)$
Ret $\mathbf{pk}$

$\underline{\text{Store}(\phi, i)}$
If not $(1 \le i \le n)$ then Ret $\perp$
$M \leftarrow \Phi(1^{\lambda}, \phi, \mathbf{sk})$; $C \leftarrow\!\!\$\ \mathsf{PKE.Enc}(1^{\lambda}, fp, \mathbf{pk}[i], M)$
$\sigma \leftarrow\!\!\$\ \mathsf{DS.Sig}(1^{\lambda}, pp, \mathbf{sk}[i], C)$; $Q \leftarrow Q \cup \{((C, \sigma), i)\}$
Ret (C, σ)

$\underline{\text{Retrieve}((C, \sigma), i)}$
If not $(1 \le i \le n)$ then Ret $\perp$
If $((C, \sigma), i) \in Q$ then Ret $\perp$
If $\mathsf{DS.Ver}(1^{\lambda}, pp, \mathbf{pk}[i], C, \sigma)$ then win $\leftarrow$ true else Ret $\perp$
$M \leftarrow \mathsf{PKE.Dec}(1^{\lambda}, fp, \mathbf{sk}[i], C)$
Ret M

Figure 3.12. Game defining alternate form of SUF for the proof of part (2) of Theorem 3.4.1.

$\underline{\text{MAIN } \boxed{\mathrm{G}_0^A(\lambda)} \,/\, \mathrm{G}_1^A(\lambda) \,/\, \mathrm{G}_2^A(\lambda)}$
$Q \leftarrow \emptyset\,;\ Q' \leftarrow \emptyset\,;\ \mathsf{win} \leftarrow \mathsf{false}$
$fp \leftarrow\!\!\$\ \mathsf{F.Pg}(1^\lambda)\,;\ (ap, std, xtd) \leftarrow\!\!\$\ \mathsf{DS.SimPg}(1^\lambda)\,;\ pp \leftarrow (fp, ap)$
$\perp \leftarrow\!\!\$\ A^{\text{MKKEY},\text{STORE},\text{RETRIEVE}}(1^\lambda, pp)$
Ret win

$\underline{\text{MKKEY}(1^n)} \quad / \ \mathrm{G}_0^A(\lambda) \,/\, \mathrm{G}_1^A(\lambda) \,/\, \mathrm{G}_2^A(\lambda)$
For $i = 1, \ldots, n$ do $(\mathbf{sk}[i], \mathbf{pk}[i]) \leftarrow\!\!\$\ \mathsf{DS.Kg}(1^\lambda, pp)$
Ret $\mathbf{pk}$

$\underline{\text{proc STORE}(\phi)} \quad / \ \mathrm{G}_0^A(\lambda) \,/\, \mathrm{G}_1^A(\lambda) \,/\, \mathrm{G}_2^A(\lambda)$
If not $(1 \le i \le n)$ then return $\perp$
$M \leftarrow \Phi(1^\lambda, \phi, \mathbf{sk})\,;\ C \leftarrow\!\!\$\ \mathsf{PKE.Enc}(1^\lambda, fp, \mathbf{pk}[i], M)$
$\sigma \leftarrow\!\!\$\ \mathsf{DS.SimSig}(1^\lambda, pp, std, \mathbf{pk}[i], C)\ ;\ Q \leftarrow Q \cup \{((C,\sigma), i)\}\ ;\ Q' \leftarrow Q' \cup \{(\mathbf{pk}[i], C, \sigma)\}$
Ret (C, σ)

$\underline{\text{proc RETRIEVE}((C,\sigma), i)} \quad / \ \boxed{\mathrm{G}_0^A(\lambda)} \,/\, \mathrm{G}_1^A(\lambda)$
If not $(1 \le i \le n)$ then return $\perp$
If $(((C,\sigma), i) \in Q)$ then Ret $\perp$
If $\mathsf{DS.Ver}(1^\lambda, pp, \mathbf{pk}[i], C, \sigma)$ then $\mathsf{win} \leftarrow \mathsf{true}$ else Ret $\perp$
$sk' \leftarrow\!\!\$\ \mathsf{DS.Ext}(1^\lambda, pp, xtd, \mathbf{pk}[i], C, \sigma)\,;\ pk' \leftarrow \mathsf{F.Ev}(1^\lambda, fp, sk')$
$M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, sk', C)$
If $(pk' \neq \mathbf{pk}[i])$ then $\mathsf{bad} \leftarrow \mathsf{true}\,;\ \boxed{M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, \mathbf{sk}[i], C)}$
Ret M

$\underline{\text{proc RETRIEVE}((C,\sigma), i)} \quad / \ \mathrm{G}_2^A(\lambda)$
If not $(1 \le i \le n)$ then return $\perp$
If $(((C,\sigma), i) \in Q)$ then Ret $\perp$
If $\mathsf{DS.Ver}(1^\lambda, pp, \mathbf{pk}[i], C, \sigma)$ then $\mathsf{win} \leftarrow \mathsf{true}$ else Ret $\perp$
$M \leftarrow \mathsf{PKE.Dec}(1^\lambda, fp, \mathbf{sk}[i], C)$
If $(\mathbf{pk}[i], C, \sigma) \in Q'$ then $\mathsf{bad} \leftarrow \mathsf{true}\,;$ Ret M
$sk' \leftarrow\!\!\$\ \mathsf{DS.Ext}(1^\lambda, pp, xtd, \mathbf{pk}[i], C, \sigma)\,;\ pk' \leftarrow \mathsf{F.Ev}(1^\lambda, fp, sk')$
If $(pk' \neq \mathbf{pk}[i])$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Ret M

Figure 3.13. Games used in the proof of part (2) of Theorem 3.4.1. Game G_0 includes the boxed code and game G_1 does not.

Chapter 4

Anonymity in Bitcoin

Demand for low friction e-commerce of various kinds has driven a proliferation in online payment systems over the last decade. Thus, in addition to established payment card networks (e.g., Visa and Mastercard) a broad range of so-called "alternative payments" has emerged including eWallets (e.g., Paypal, Google Checkout, and WebMoney), direct debit systems (typically via ACH, such as eBillMe), money transfer systems (e.g., Moneygram) and so on. However, virtually all of these systems have the property that they are denominated in existing fiat currencies (e.g., dollars), explicitly identify the payer in transactions, and are centrally or quasi-centrally administered.[1]

By far the most intriguing exception to this rule is Bitcoin. First deployed in 2009, Bitcoin is an independent online monetary system that combines some of the features of cash and existing online payment methods. Like cash, Bitcoin transactions do not explicitly identify the payer or the payee: a transaction is a cryptographically-signed transfer of funds from one public key to another. Moreover, like cash, Bitcoin transactions are irreversible (in particular, there is no *chargeback* risk as with credit cards). However, unlike cash, Bitcoin requires third party mediation: a global peer-to-peer network of participants validates and certifies all transactions; such decentralized accounting

[1] In particular, there is a central controlling authority who has the technical and legal capacity to tie a transaction back to a pair of individuals.

requires each network participant to maintain the entire transaction history of the system, currently amounting to over 3GB of compressed data. Bitcoin identities are thus *pseudo-anonymous*: while not explicitly tied to real-world individuals or organizations, all transactions are completely transparent.[2]

This unusual combination of features has given rise to considerable confusion about the nature and consequences of the anonymity that Bitcoin provides. In particular, there is concern that the combination of scalable, irrevocable, anonymous payments would prove highly attractive for criminals engaged in fraud or money laundering. In a widely leaked 2012 Intelligence Assessment, FBI analysts make just this case and conclude that a key "advantage" of Bitcoin for criminals is that "law enforcement faces difficulties detecting suspicious activity, identifying users and obtaining transaction records" [74]. Similarly, in a late 2012 report on Virtual Currency Schemes, the European Central Bank opines that the lack of regulation and due diligence might enable "criminals, terrorists, fraudsters and money laundering" and that "the extent to which any money flows can be traced back to a particular user is unknown" [72]. Indeed, there is at least some anecdotal evidence that this statement is true, with the widely publicized "Silk Road" service using Bitcoin to trade in a range of illegal goods (e.g., restricted drugs and firearms). Finally, adding to this urgency is Bitcoin's considerable growth, both quantitatively — a merchant servicer, Bitpay, announced that it had signed up over 1,000 merchants in 2012 to accept the currency, and in November 2013 the exchange rate soared to 1,200 USD per bitcoin before settling to a more modest 600 USD per bitcoin — and qualitatively via integration with existing payment mechanisms (e.g., Bitinstant offering to tie users' Bitcoin wallets to Mastercard accounts [66] and Bitcoin Central's recent partnership with the French bank Crédit Mutuel Arkéa to gateway Bitcoin into

[2]Note that this statement is not strictly true since private exchanges of Bitcoin between customers of a single third party exchange, such as Mt. Gox, need not (and do not) engage the global Bitcoin protocol and are therefore not transparent.

the banking system [114]) and the increasing attention of world financial institutions (e.g., Canada's recent decision to tax Bitcoin transactions [49] and FinCEN's recent regulations on virtual currencies [75]). In spite of this background of intense interest, Bitcoin's pseudo-anonymity has limited how much is known about how the currency is used and how Bitcoin's use has evolved over time.

In this context, our work seeks to better understand the traceability of Bitcoin flows and, through this understanding, explore the evolution in how Bitcoin has been used over time. Importantly, our goal is not to generally de-anonymize all Bitcoin users — as the abstract protocol design itself dictates that this should be impossible — but rather to identify certain *idioms of use* present in concrete Bitcoin network implementations that erode the anonymity of the users who engage in them. Our approach is based on the availability of the Bitcoin *block chain*: a replicated graph data structure that encodes all Bitcoin activity, past and present, in terms of the public digital signing keys party to each transaction. However, since each of these keys carries no explicit information about ownership, our analysis depends on imposing additional structure on the transaction graph.

Our methodology has two phases. First, in Section 4.2, we describe a re-identification attack wherein we open accounts and make purchases from a broad range of known Bitcoin merchants and service providers (e.g., Mt. Gox and Silk Road). Since one endpoint of the transaction is known (i.e., we know which public key we used), we are able to positively label the public key on the other end as belonging to the service; we augment this attack by crawling Bitcoin forums for "self-labeled" public keys (e.g., where an individual or organization explicitly advertises a key as their own). Next, in Section 4.3, we build on past efforts [8, 116, 121, 138] to cluster public keys based on evidence of shared spending authority. This clustering allows us to amplify the results of our re-identification attack: if we labeled one public key as belonging to Mt. Gox, we

can now transitively taint the entire cluster containing this public key as belonging to Mt. Gox as well. The result is a condensed graph, in which nodes represent entire users and services rather than individual public keys.

From this data we characterize Bitcoin use longitudinally, focusing in particular on the evolution of services and their role in the Bitcoin network. Finally, in Section 4.4, we combine what we have learned to examine the suitability of Bitcoin for hiding large-scale illicit transactions. Using the dissolution of a large Silk Road wallet and notable Bitcoin thefts as case studies, we demonstrate that an agency with subpoena power would be well placed to identify who is paying money to whom. Indeed, we argue that the increasing dominance of a small number of Bitcoin institutions (most notably services that perform currency exchange), coupled with the public nature of transactions and our ability to label monetary flows to major institutions, ultimately makes Bitcoin unattractive today for high-volume illicit use such as money laundering.

4.1 Bitcoin Background

Our heuristics that we use to cluster addresses depend on the structure of the Bitcoin protocol, so we first describe it here, and briefly mention the anonymity that it is intended to provide. Additionally, much of our analysis discusses the "major players" and different categories of bitcoin-based services, so we also present a more high-level overview of Bitcoin participation, as well as some general statistics about the Bitcoin network.

4.1.1 Bitcoin protocol description

Bitcoin is a decentralized electronic currency, introduced by (the pseudonymous) Satoshi Nakamoto in 2008 [109] and deployed on January 3 2009. Briefly, a bitcoin[3] can be thought of as a chain of *transactions* from one owner to the next, where owners are identified by a *public key* (in practice, a public key for the ECDSA signature scheme) that serves as a pseudonym; i.e., users can use any number of public keys and their activity using one set of public keys is not inherently tied to their activity using another set, or to their real-world identity (so that, e.g., a user can use a different public key to deposit bitcoins into his Silk Road account than to withdraw bitcoins from his Mt. Gox account, and expect that these activities cannot be linked to either his real identity or to each other). In each transaction, the previous owner signs — using the secret signing key corresponding to his public key — a hash of the transaction in which he received the bitcoins (in practice, a SHA-256 hash) and the public key of the next owner. This signature (i.e., transaction) can then be added to the set of transactions that constitutes the bitcoin; because each of these transactions references the previous transaction (i.e., in sending bitcoins, the current owner must specify where they came from), the transactions form a chain. To verify the validity of a bitcoin, a user can check the validity of each of the signatures in this chain.

To prevent double spending, it is necessary for each user in the system to be aware of all such transactions. Double spending can then be identified when a user attempts to transfer a bitcoin after he has already done so. To determine which transaction came first, transactions are grouped into *blocks*, which serve to timestamp the transactions they contain and vouch for their validity. Blocks are themselves formed into a chain, with each block referencing the previous one (and thus further reinforcing the validity of all

[3]Following established convention, we use the capitalized term *Bitcoin* when referring to the payment system and peer-to-peer network and the lowercase term *bitcoin* (abbreviated *BTC*), when referring to the unit of currency.

previous transactions). This process yields a *block chain*, which is then publicly available to every user within the system.

This process describes how to transfer bitcoins and broadcast transactions to all users of the system. Because Bitcoin is decentralized and there is thus no central authority minting bitcoins, we must also consider how bitcoins are generated in the first place. In fact, this happens in the process of forming a block: each accepted block (i.e., each block incorporated into the block chain) is required to be such that, when all the data inside the block is hashed, the hash begins with a certain number of zeroes. To allow users to find this particular collection of data, blocks contain, in addition to a list of transactions, a *nonce*. (We simplify the description slightly to ease presentation.) Once someone finds a nonce that allows the block to have the correctly formatted hash, the block is then broadcast in the same peer-to-peer manner as transactions. The system is designed to generate only 21 million bitcoins in total. Finding a block currently comes with an attached reward of 25 BTC; this rate was 50 BTC until November 28 2012 (block height 210,000), and is expected to halve again in 2016, and eventually drop to 0 in 2140.

In summary, the dissemination of information within the Bitcoin network is as follows (and as depicted in Figure 4.1): first, users generate at least one signing keypair, and publicize the public key, or *address* — in the rest of the paper we use these terms interchangeably — to receive bitcoins (and again, users can choose to use a single public key or arbitrarily many). If a user has bitcoins that she wishes to transfer, she broadcasts a *transaction*, proving that she has the bitcoins and indicating the address of the recipient to her peers, who in turn broadcast it to their peers. Eventually, this transaction reaches a miner, who collects the transactions he hears about into a *block*, and works on finding the right data/nonce balance to hit the target hash. He also includes in the block a special *coin generation* transaction that specifies his address for receiving the block reward. Finally, when the miner does find such a block, he broadcasts it to his peers, who again

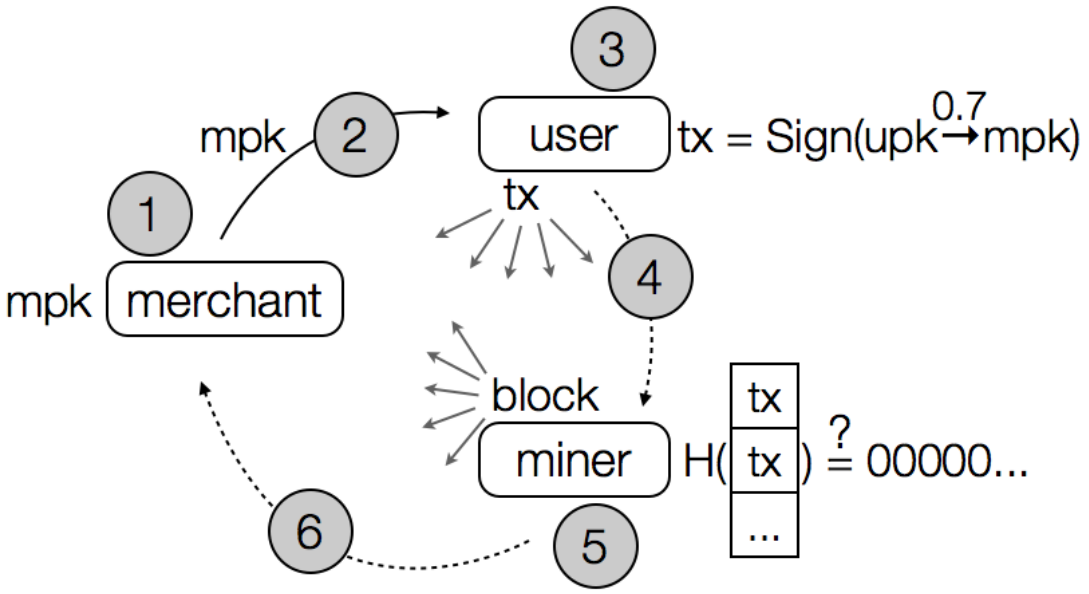


Figure 4.1. How a Bitcoin transaction works. In this example, a user wants to send 0.7 bitcoins as payment to a merchant. In (1), the merchant generates or picks an existing public key `mpk`, and in (2) it sends this public key to the user. In (3), by creating a digital signature, the user forms the transaction `tx` to transfer the 0.7 bitcoins from his public key `upk` to the merchant's address `mpk`. In (4), the user broadcasts this transaction to his peers, which (if the transaction is valid) allows it to flood the network. In this way, a miner learns about his transaction. In (5), the miner works to incorporate this and other transactions into a block by checking if their hash is within some target range. In (6), the miner broadcasts this block to her peers, which (if the block is valid) allows it to flood the network. In this way, the merchant learns that the transaction has been accepted into the global block chain, and has thus received the user's payment.

broadcast it to their peers. As his reward, the block reward and all the fees for the included transactions are credited to his specified address. When another block has been formed, referencing his block as the previous block, his block can now be considered part of the *block chain*.

4.1.2 Participants in the Bitcoin network

In practice, the way in which Bitcoin can be used is much simpler than the above description might indicate. First, generating a block is so computationally difficult that very few individual users attempt it on their own. Instead, users may join a *mining pool*

such as Deepbit, in which they contribute "shares" to narrow down the search space, and earn a small amount of bitcoins in exchange for each share.

Users may also avoid coin generation entirely, and simply purchase bitcoins through one of the many *exchanges*, such as Mt. Gox. They may then keep the bitcoins in a wallet stored on their computer or, to make matters even easier, use one of the many *wallet services* (i.e., banks) that exist online (although the two most popular of these, MyBitcoin and Instawallet, have both shut down due to thefts).

Finally, to actually spend their bitcoins, users could gamble with one of the popular dice games such as Satoshi Dice. They could also buy items from various online vendors, such as Bitmit ("the eBay of Bitcoin"), the notorious Tor-based service Silk Road, or with vendors, such as Wordpress, that might ordinarily accept only US dollars but accept bitcoins through BitPay, a payment gateway that takes bitcoins from the buyer but offers the option of payment in USD to the seller (thus eliminating all Bitcoin-based risk for the vendor). Finally, users wishing to go beyond basic currency speculation can invest their bitcoins with firms such as Bitcoinica (shut down after a series of thefts) or Bitcoin Savings & Trust (later revealed as a major Ponzi scheme). In Section 4.2, we more fully describe the role and impact of these and other services within the Bitcoin network.

4.1.3 Bitcoin network statistics

We used the `bitcoind` client to download the block chain, and parsed it into a PostgreSQL database using a modified version of the `bitcointools` library developed by Gavin Andresen [7]. We last parsed the block chain on April 13 2013, when there were 231,207 blocks, containing 16,086,073 transactions and 12,056,684 distinct public keys.

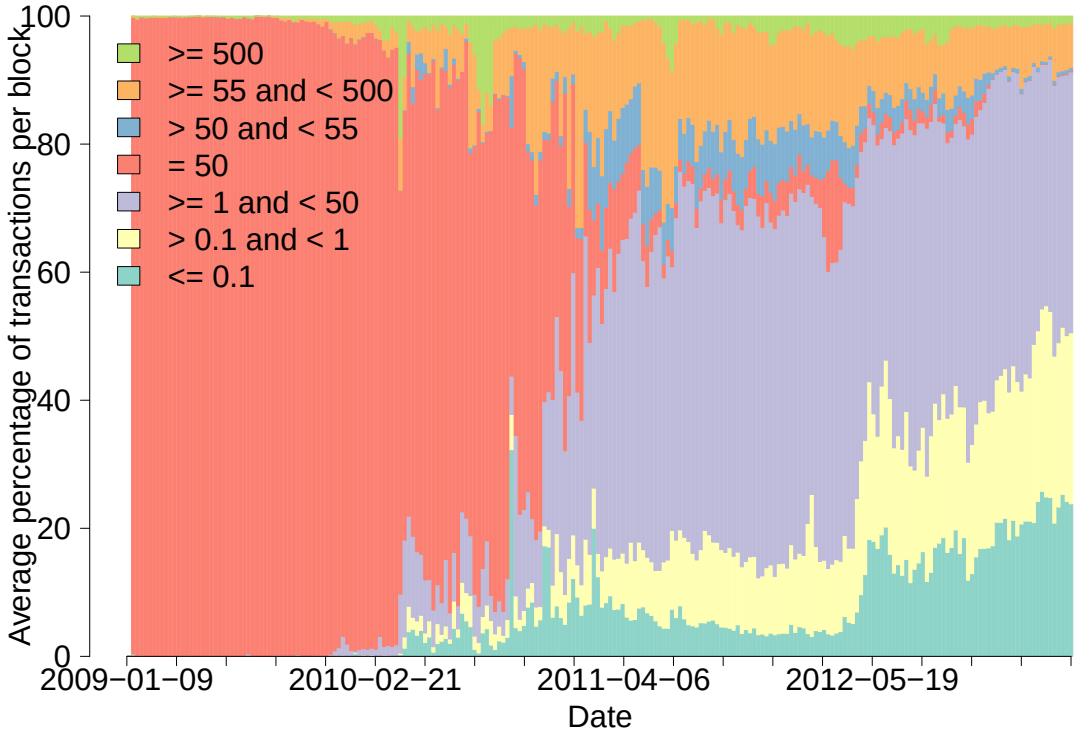


Figure 4.2. (In color online.) The distribution, over time and averaged weekly, of transaction values. The plot and legend both run, bottom to top, from the smallest-valued transactions to the highest.

To begin, we looked at the size of transactions; i.e., the number of bitcoins sent in a transaction. Figure 4.2 depicts the changing percentage of various transaction sizes over time. Not surprisingly, until approximately April 2010 — the first 15 months that Bitcoin was deployed — almost all transactions involved exactly 50 bitcoins (the initial reward for mining a block), and indeed these transactions became a minority of all transactions only in January 2011. This activity reflects the adoption phase of Bitcoin, in which most blocks contained the coin generation transaction and nothing more. (In later phases, the mining reward is likely a little more than 50 because it includes miner fees, which is why we created a separate bin for values between 50 and 55.) We also see a second turning point in early 2012, in which the percentage of transactions carrying less than a single bitcoin in total value doubled abruptly (from 20% to 40%), while the percentage of transactions carrying less than 0.1 BTC tripled.

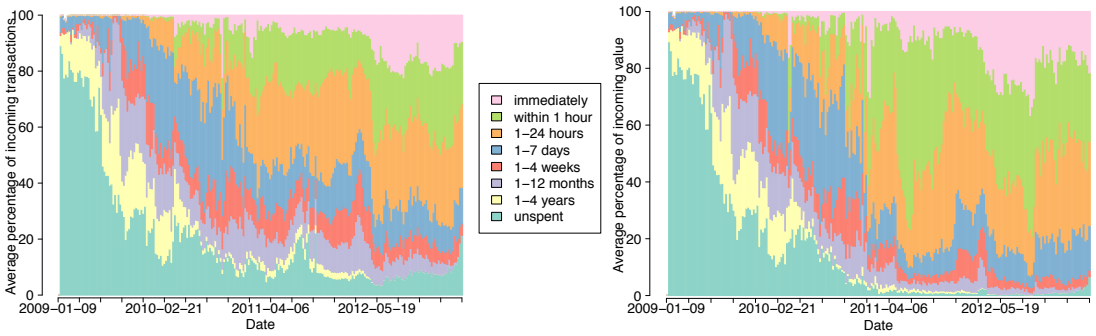


Figure 4.3. (In color online.) The trend, over time and averaged weekly, of how long public keys hold on to the bitcoins received. The plot on the left shows the percentage over all public keys, and the plot on the right shows the percentage over all value transacted. The values run bottom to top, from longest to spend (unspent as of now) to shortest to spend (spent within the same block).

We also observed how quickly bitcoins were spent; i.e., once they were received, how long did it take the recipient to spend them? Figure 4.3 shows breakdowns both in terms of public keys (how many recipient public keys spent their contents in a certain time window) and in terms of value (how many of the bitcoins that were received were spent in a certain time window).

Looking at this figure, we again see two clear turning points. The first, in early 2011, represents a point at which users began meaningfully spending bitcoins, rather than just "hoarding" them; in fact, from this point on a negligible fraction of bitcoins are hoarded. Nevertheless, these early hoarders in fact took most of the bitcoins out of circulation; as observed by Ron and Shamir [121], a significant majority of all bitcoins are in these "sink" addresses that have to date never spent their contents (at the time they parsed the block chain it was 75%, whereas we observed it to be 64%), meaning only 4 million bitcoins are currently in circulation. Nevertheless, these remaining coins are circulating quite actively, as seen in the second turning point in Figure 4.3: in April 2012, the percentage of bitcoins being spent *immediately* (i.e., in the same block in which they were received) doubled, and more generally half of all bitcoins are now spent within an hour of being received and 80% of bitcoins are spent within a day.

As it turns out, and as we see in Section 4.4.1, both these recent trends of smaller transactions and faster spending can be largely attributed to a single service: the gambling site Satoshi Dice. Thus, even a longitudinal study of the Bitcoin network already makes clear the effect that services have on current Bitcoin usage.

4.2 Data Collection

To identify public keys belonging to the types of services mentioned in Section 4.1.2, we sought to "tag" as many addresses as possible; i.e., label an address as being definitively controlled by some known real-world user (e.g., Mt. Gox or Instawallet). As we will see in Section 4.3.3, by clustering addresses based on evidence of shared control, we can bootstrap off the minimal ground truth data this provides to tag entire clusters of addresses as also belonging to that user.

Our predominant method for tagging users was simply transacting with them (e.g., depositing into and withdrawing bitcoins from Mt. Gox) and then observing the addresses they used; additionally, we collected known (or assumed) addresses that we found in various forums and other Web sites, although we regarded this latter kind of tagging as less reliable than our own observed data.

4.2.1 From our own transactions

We engaged in 344 transactions with a wide variety of services, listed in Table 4.1, including mining pools, wallet services, bank exchanges, non-bank exchanges, vendors, gambling sites, and miscellaneous services.

Mining pools. We attempted to mine with each of the major mining pools (a pie chart depicting the relative productivity of mining pools can be found at blockorigin.pfoe.be/chart.php). To do this, we used an AMD Radeon HD 7970, capable of approximately 530 million SHA-256 computations per second; this effort allowed us to trigger a payout

of at least 0.1 BTC (often the minimum payout for pools) with 11 different pools, anywhere from 1 to 25 times. For each payout transaction, we then labeled the input public keys as belonging to the pool. One of these pools, Eligius, split the coin among the miners immediately upon being mined, and we were thus unable to tag any of their public keys using this method.

Wallets. We kept money with most of the major wallet services (10 in total), and made multiple deposit and withdrawal transactions for each. Three of these services — My Wallet, Easycoin, and Strongcoin — kept the funds of their users separate, which meant we were unable to link many addresses together for them.

Bank exchanges. Most of the real-time trading exchanges (i.e., in which the exchange rate is not fixed) also function as banks. As such, we tagged these services just as we did the wallets: by depositing into and withdrawing from our accounts (but rarely participating in any actual currency exchange). We kept accounts with 18 such exchanges in total.

Non-bank exchanges. In contrast, most of the fixed-rate exchanges did not function as banks, and are instead intended for one-time conversions. We therefore were able to participate in fewer transactions with these exchanges, although we again tried to transact with most of the major ones at least once (8 in total).

Vendors. We purchased goods, both physical and digital, from a wide variety of vendors. Some of the vendors, such as Bitmit and CoinDL, function more as marketplaces (the "eBay" and "iTunes" of the Bitcoin economy, respectively), while others were individual merchants. Although we purchased from Etsy, they do not provide a Bitcoin payment interface and we instead negotiated individually with the merchant. Many of the vendors we interacted with did not use an independent method for accepting bitcoins, but relied instead on the BitPay payment gateway (and one used WalletBit as a payment

gateway). We also kept a wallet with Silk Road, which allowed us to tag their public keys without making any purchases. Figure 4.4 depicts all of our physical purchases.

Gambling. We kept accounts with five poker sites, and transacted with eight sites offering mini-games and/or lotteries. Many of the dice games (Satoshi Dice, BTC Dice, etc.) advertised their public keys, so we did fewer transactions with these services.

Miscellaneous. Four of the additional services we interacted with were *mix* or *laundry* services: when provided with an output address, they promised to send to that address coins that had no association with the ones sent to them; the more sophisticated ones offered to spread the coins out over various transactions and over time. One of these, BitMix, simply stole our money, while Bitcoin Laundry twice sent us our own coins back, indicating we were possibly their only customer at that time. We also interacted with Bit Visitor, a site that paid users to visit certain sites; Bitcoin Advertisers, which provided online advertising; CoinAd, which gave out free bitcoins; Coinapult, which forwarded bitcoins to an email address, where they could then be redeemed; and finally, Wikileaks, with whom we donated to both their public donation address and two one-time addresses generated for us via their IRC channel.

4.2.2 From other sources

In addition to our own transactions, many users publicly claim their own addresses; e.g., charities providing donation addresses, or LulzSec claiming their address on Twitter. While we did not attempt to collect all such instances, many of these tags are conveniently collected at blockchain.info/tags, including both addresses provided in users' signatures for Bitcoin forums, as well as self-submitted tags. We collected all of these tags — over 5,000 in total — keeping in mind that the ones that were not self-submitted (and even the ones that were) could be regarded as less reliable than the ones we collected ourselves.

Table 4.1. The various services we interacted with, grouped by (approximate) type.

Mining		
50 BTC	BTC Guild	Itzod
ABC Pool	Deepbit	Ozcoin
Bitclockers	EclipseMC	Slush
Bitminter	Eligius	
Wallets		
Bitcoin Faucet	Easywallet	Strongcoin
My Wallet	Flexcoin	WalletBit
Coinbase	Instawallet	
Easycoin	Paytunia	
Exchanges		
Bitcoin 24	BTC-e	Aurum Xchange
Bitcoin Central	CampBX	BitInstant
Bitcoin.de	CA VirtEx	Bitcoin Nordic
Bitcurex	ICBit	BTC Quick
Bitfloor	Mercado Bitcoin	FastCash4Bitcoins
Bitmarket	Mt Gox	Lilion Transfer
Bitme	The Rock	Nanaimo Gold
Bitstamp	Vircurex	OKPay
BTC China	Virwox	
Vendors		
ABU Games	BTC Buy	HealthRX
Bitbrew	BTC Gadgets	JJ Games
Bitdomain	Casascius	NZBs R Us
Bitmit	Coinabul	Silk Road
Bitpay	CoinDL	WalletBit
Bit Usenet	Etsy	Yoku
Gambling		
Bit Elfin	BitZino	Gold Game Land
Bitcoin 24/7	BTC Griffin	Satoshi Dice
Bitcoin Darts	BTC Lucky	Seals with Clubs
Bitcoin Kamikaze	BTC on Tilt	
Bitcoin Minefield	Clone Dice	
Miscellaneous		
Bit Visitor	Bitfog	CoinAd
Bitcoin Advertisers	Bitlaundry	Coinapult
Bitcoin Laundry	BitMix	Wikileaks

Finally, we searched through the Bitcoin forums (in particular, bitcointalk.org) looking for addresses associated with major thefts, or now-defunct services such as Tradehill and GLBSE. Again, these sources are less reliable, so we consequently labeled

Figure 4.4. (In color online.) The physical items we purchased with bitcoins, including silver quarters from Coinabul, coffee from Bitcoin Coffee, and a used Boston CD from Bitmit. The items in green were purchased from CoinDL; in blue from Bitmit; and in red using the payment gateway BitPay.

users only for addresses for which we could gain some confidence through manual due diligence.

4.3 Account Clustering Heuristics

In this section, we present two heuristics for linking addresses controlled by the same user, with the goal of collapsing the many public keys seen in the block chain into larger entities. The first heuristic, in which we treat different public keys used as inputs to

a transaction as being controlled by the same user, has already been used and explored in previous work, and exploits an inherent property of the Bitcoin protocol. The second is new and based on so-called *change addresses*; in contrast to the first, it exploits a current *idiom of use* in the Bitcoin network rather than an inherent property. As such, it is less robust in the face of changing patterns within the network, but — as we especially see in Section 4.4.2 — it provides insight into the current Bitcoin network that the first heuristic does not.

4.3.1 Defining account control

Before we present our heuristics, we clarify what the results of our clustering algorithms imply; in particular, we must define what we mean by address *control*. Put simply, we say that the controller of an address is the entity (or in exceptional cases multiple entities) that is expected to participate in transactions involving that address. While this requirement implies a priori that the controller of an address knows the corresponding private key (recall that transactions are signatures, and thus knowledge of the signing key is necessary to form a valid transaction), knowledge of the private key is not a sufficient requirement for control. Consider, for example, buying physical bitcoins from a vendor such as Casascius. To form the physical bitcoin to send to you, Casascius must know the private key. Then, once you receive the bitcoin, you also learn the private key. Finally, if you redeem this private key with a service such as Mt. Gox, that service also learns the private key. In such a case, control defined solely by knowledge of the secret key is therefore not well defined.

In the above case, however, the controller of the address is in fact quite clear: as you redeemed the private key with Mt. Gox and thus stored any bitcoins inside with them, the expected entity responsible for forming transactions on behalf of that address is Mt.

Gox (otherwise, if you plan to form your own transactions, why store your money with them?).

Finally, we emphasize that our definition of address control is quite different from account *ownership*; for example, we consider a wallet service such as Instawallet to be the controller of each of the addresses it generates, even though the funds in these addresses are owned by a wide variety of distinct users.

4.3.2 Graph structure and definitions

To define our heuristics formally, we consider two important directed graph structures for the Bitcoin network: a transaction graph and a public key graph. In the former, vertices represent transactions, and a directed edge from a transaction t_1 to a transaction t_2 indicates that an output of t_1 was used as an input in t_2. Using this graph, we define in degrees and out degrees for transactions, which correspond exactly to the in and out degrees in the graph (i.e., the number of edges incident to and from the node, respectively).

Definition 4.3.1. *The* in degree *for a transaction t, denoted by $d^{+}_{\mathsf{tx}}(t)$, is the number of inputs for the transaction. The* out degree *for a transaction t, denoted by $d^{-}_{\mathsf{tx}}(t)$, is the number of outputs for the transaction.*

We can also construct a graph using public keys, in which vertices are public keys and directed edges again represent the flow of money from one public key to another; here, however, the in degree of a public key reflects the number of inputs to the transaction in which it received bitcoins, so a public key that received bitcoins only once could have an in degree of (for example) five. For our purposes, we would instead like the in degree of the output public keys to be independent of how many public keys are provided as input to the transaction. We therefore define, rather than in/out degree, the *in/out count* for a public key.

Definition 4.3.2. *The* in count *for a public key pk, denoted* $d^{+}_{\mathsf{addr}}(pk)$*, is the number of times pk has been an output in a transaction. The* out count *for a public key pk, denoted* $d^{-}_{\mathsf{addr}}(pk)$*, is the number of times pk has been an input in a transaction.*

One of the defining features of the Bitcoin protocol is the way that bitcoins must be spent. When the bitcoins redeemed as the output of a transaction are spent, they must be spent all at once: the only way to divide them is through the use of a *change address*, in which the excess from the input address is sent back to the sender. A public key can therefore spend money only as many times as it has received money (again, because each time it spends money it must spend all of it at once).

4.3.3 Our heuristics

Heuristic 1 The first heuristic we use, in which we link input addresses together, has already been used many times in previous work [8, 116, 121, 138]; for completeness, we nevertheless present it here. Briefly, if two (or more) public keys are used as inputs to the same transaction, then we say that they are controlled by the same user.

Heuristic 1. *If two (or more) addresses are inputs to the same transaction, they are controlled by the same user; i.e., for any transaction t, all* $pk \in \mathsf{inputs}(t)$ *are controlled by the same user.*

The effects of this heuristic are transitive and extend well beyond the inputs to a single transaction; e.g., if we observed one transaction with addresses A and B as inputs, and another with addresses B and C as inputs, then we conclude that A, B, and C all belonged to the same user. It is also quite safe: the sender in the transaction must know the private signing key belonging to each public key used as an input, so it is unlikely that the collection of public keys are controlled by multiple entities (as these entities would need to reveal their private keys to each other).

Using this heuristic, we partitioned the network into 5,579,176 clusters of users. By naming these clusters — using the data collection described in Section 4.2 — we observed that some of them corresponded to the same user; e.g., there were 20 clusters that we tagged as being controlled by Mt. Gox. (This is not surprising, as many big services appear to spread their funds across a number of distinct accounts to minimize the risk in case any one gets compromised.) This cross-cluster naming was nevertheless not too common, and we thus ended up with 5,577,481 distinct clusters (recall we started with 12,056,684 public keys). Factoring in "sink" addresses that have to date never sent any bitcoins (and thus did not get clustered using this heuristic) yields at most 6,595,564 distinct users, although we consider this number a quite large upper bound.

Heuristic 2 Although Heuristic 1 already yields a useful clustering of users, restricting ourselves to only this heuristic does not tell the whole story. To further collapse users, our second heuristic focuses on the role of change addresses within the Bitcoin system. A similar heuristic was explored by Androulaki et al. [8] (who called them "shadow" addresses), although there are a number of important differences. In particular, their definition of shadow addresses relied upon assumptions that may have held at the time of their work, but no longer hold at present. For example, they assumed that users rarely issue transactions to two different users, which is a frequent occurrence today (e.g., payouts from mining pools, or bets on gambling sites).

As discussed above, change addresses are the mechanism used to give money back to the input user in a transaction, as bitcoins can be divided only by being spent. In one idiom of use, the change address is created internally by the Bitcoin client and never re-used; as such, a user is unlikely to give out this change address to other users (e.g., for accepting payments), and in fact might not even know the address unless he inspects the block chain. If we can identify change addresses, we can therefore potentially cluster not

only the input addresses for a transaction (according to Heuristic 1) but also the change address and the input user.

Because our heuristic takes advantage of this idiom of use, rather than an inherent property of the Bitcoin protocol (as Heuristic 1 does), it does lack robustness in the face of changing (or adversarial) patterns in the network. Furthermore, it has one very negative potential consequence: falsely linking even a small number of change addresses might collapse the entire graph into large "super-clusters" that are not actually controlled by a single user (in fact, we see this exact problem occur in Section 4.3.5). We therefore focused on designing the safest heuristic possible, even at the expense of losing some utility by having a high false negative rate, and acknowledge that such a heuristic might have to be redesigned or ultimately discarded if habitual uses of the Bitcoin protocol change significantly.

Working off the assumption that a change address has only one input (again, as it is potentially unknown to its owner and is not re-used by the client), we first looked at the outputs of every transaction. If only one of the outputs met this pattern, then we identified that output as the change address. If, however, multiple outputs had only one input and thus the change address was ambiguous, we did not label any change address for that transaction. We also avoided certain transactions; e.g., in a coin generation, none of the outputs are change addresses.

In addition, in custom usages of the Bitcoin protocol it is possible to specify the change address for a given transaction. Thus far, one common usage of this setting that we have observed has been to provide a change address that is in fact the same as the input address.[4] We thus avoid such "self-change" transactions as well.

[4]This usage is quite common: 23% of all transactions in the past six months are self-change transactions. For example, it is the standard option for the popular wallet service My Wallet, hosted by blockchain.info, as well as the way the Deepbit mining pool does its payouts.

Definition 4.3.3. *A public key pk is a* one-time change address *for a transaction t if the following conditions are met:*

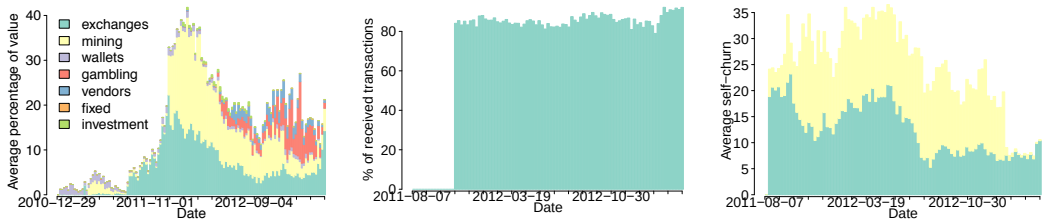
1. $d^{+}_{\mathsf{addr}}(pk) = 1$*; i.e., this is the first appearance of pk.*
2. *The transaction t is not a coin generation.*
3. *There is no* $pk' \in \mathsf{outputs}(t)$ *such that* $pk' \in \mathsf{inputs}(t)$*; i.e., there is no self-change address.*
4. *There is no* $pk' \in \mathsf{outputs}(t)$ *such that* $pk' \neq pk$ *but* $d^{+}_{\mathsf{addr}}(pk') = 1$*; i.e., for all the outputs in the transaction, condition* 1 *is met for only pk.*

Heuristic 2. *The one-time change address is controlled by the same user as the input addresses; i.e., for any transaction t, the controller of* $\mathsf{inputs}(t)$ *also controls the one-time change address* $pk \in \mathsf{outputs}(t)$ *(if such an address exists).*

4.3.4 The impact of change addresses

To see the impact of change addresses on user clustering, consider the following illustrative example: suppose we want to measure the *incoming value* of the major services with whom we interacted; i.e., we want to know how many bitcoins they received over time. If we consider the incoming value of services across seven different categories — exchanges that function as banks, mining pools, wallet services, gambling sites, vendors, fixed-rated exchanges that do not function as banks, and investment schemes — then, using Heuristic 1, we obtain the results shown in Figure 4.5a.

Looking at Figure 4.5a cumulatively, we might first notice that, for the past year and a half, the major users we tagged account for anywhere from 20% to 40% of the total incoming value. Comparing across categories, we see that exchanges account for a considerable fraction of the total value of these users. More surprisingly, given the



(a) **(b)** **(c)**

Figure 4.5. (In color online.) Figures illustrating the effect of self-churn on measurements, and the different ways Heuristics 1 and 2 deal with self-churn. (a) The incoming value over time, as a percentage of total weekly incoming value, for each of the major categories. The teal and yellow categories are exchanges and mining pools respectively. (b) Of transactions sent to Deepbit, the percentage that were sent by Deepbit itself (i.e., the percentage that was self-churn). The transactions from Deepbit account for over 80% of its incoming transactions. (c) The percentage of all transactions received by Mt Gox that were sent by Mt Gox. The teal shows self-churn identified by Heuristic 1; the yellow shows *additional* churn identified by Heuristic 2.

payout-based nature of mining pools, Figure 4.5a also seems to indicate that mining pools are receiving a large portion of incoming value. This percentage is artificially inflated, however, by certain artifacts of how mining pools, and Deepbit in particular, pay their miners. In fact, as we see in Figure 4.5b, over 80% of the value Deepbit receives is as change from itself.

While the particular mechanism that Deepbit uses allows us to eliminate this "self-churn" even using Heuristic 1 (as they always use a self-change address), more generally we cannot eliminate the self-churn of all users with just Heuristic 1. We are able to identify self-churn only if we know that the change address is controlled by the same user as the input address(es).

Eliminating this self-churn is therefore where Heuristic 2 becomes crucial. To see the effect it has, we compare the self-churn of Mt. Gox as determined using the two heuristics. Figure 4.5c shows that finding additional change addresses for Mt. Gox using Heuristic 2 essentially doubles the estimate of churn activity of Mt. Gox compared to

using Heuristic 1 (and we observed a similar doubling when considering the churn in bitcoin value rather than activity).

4.3.5 Refining Heuristic 2

Although effective, Heuristic 2 is more challenging and significantly less safe than Heuristic 1. In our first attempt, when we used it as defined above, we identified over 4 million change addresses. Due to our concern over its safety, we sought to approximate the false positive rate. To do this even in the absence of significant ground truth data, we used the fact that we could observe the behavior of addresses over time: if an address and transaction met the conditions of Definition 4.3.3 at one point in time (where time was measured by block height), and then at a later time the address was used again, we considered this a false positive. Stepping through time in this manner allowed us to identify 555,348 false positives, or 13% of all labeled change accounts.

We then considered ways of making the heuristic more conservative. First, however, a manual inspection of some of these false positives revealed an interesting pattern: many of them were associated with transactions to and from Satoshi Dice and other dice games. By looking further into the payout structure of these games, it became clear that these were not truly false positives, as when coins are sent to Satoshi Dice, the payout is sent back to the same address. If a user therefore spent the contents of a one-time change address with Satoshi Dice, the address would receive another input back from Satoshi Dice, which would appear to invalidate the "one-timeness" of the address. We therefore chose to ignore this case, believing that addresses that received later inputs solely from Satoshi Dice could still be one-time change addresses. By doing so the false positive rate reduces to only 1%. We next considered waiting to label an address as a change address; i.e., waiting to see if it received another input. Waiting a day drove the

false positive rate down to 0.28%; waiting a week drove it down to 0.17%, or only 7,382 false positives total.

Despite all these precautions, when we clustered users using this modified heuristic, we still ended up with a giant super-cluster containing the public keys of Mt. Gox, Instawallet, BitPay, and Silk Road, among others; in total, this super-cluster contained 1.6 million public keys. After a manual inspection of some of the links that led to this super-cluster, we discovered two problematic patterns. First, especially within a short window of time, the same change address was sometimes used twice. If this change address were then used the second time with a new address, the new address would appear to be the change address and be falsely labeled as such. Second, certain addresses would occasionally be used as "self-change" addresses (recall the second requirement in Definition 4.3.3), and then later used as separate change addresses; again, if the time they were used separately was with a new address, the new address would be falsely labeled as the change address. This behavior is likely due to the advanced features in some wallets, such as My Wallet and the desktop client Armory, that allow users to explicitly specify the change address for a transaction.

We thus further refined our heuristic by ignoring transactions involved with either of these types of behavior. For transactions in which an output had already received only one input, or for transactions in which an output had been previously used in a self-change transaction, we chose to not tag anything as the change address. Doing so, and manually removing a handful of other false positives (with no discernible pattern), we identified 3,540,831 change addresses.

Using this refined Heuristic 2 produces 3,384,179 clusters, which we were able to again collapse slightly (using our tags) to 3,383,904 distinct clusters. Of these clusters, we were able to name 2,197 of them (accounting for over 1.8 million addresses); although this might seem like a small fraction, recall that by participating in 344 transactions we

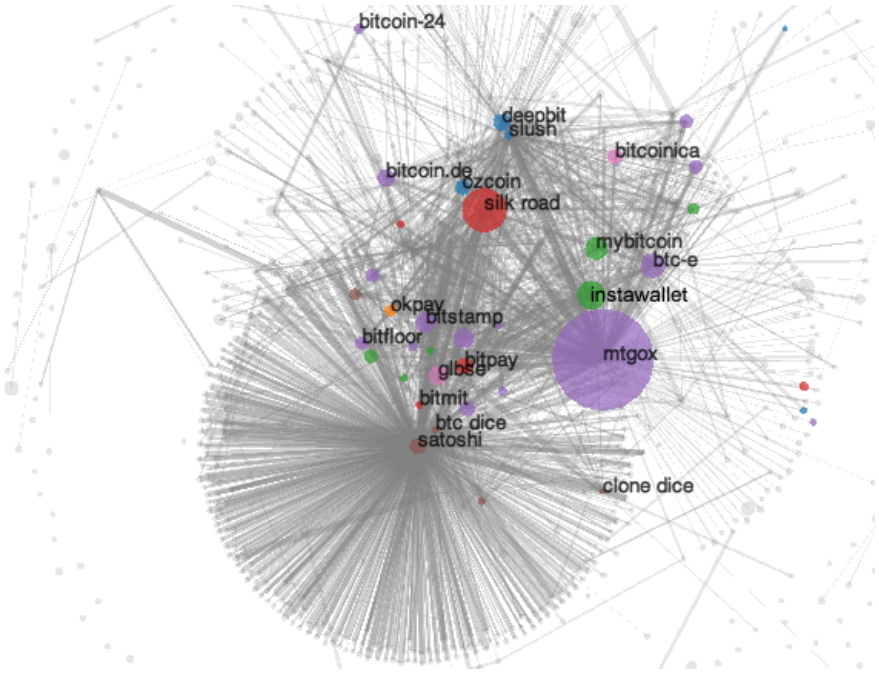


Figure 4.6. (In color online.) A visualization of the user network. The area of the cluster represents the external incoming value; i.e., the bitcoins received from other clusters but not itself, and for an edge to appear between two nodes there must have been at least 200 transactions between them. The nodes are colored by category: blue nodes are mining pools; orange are fixed-rate exchanges; green are wallets; red are vendors; purple are (bank) exchanges; brown are gambling; pink are investment schemes; and grey are uncategorized.

hand-tagged only 1,070 addresses, and thus Heuristic 2 allowed us to name 1,600 times more addresses than our own manual observation provided. Furthermore, as we see in the visualization of the user graph depicted in Figure 4.6, and will argue in Section 4.4, the users we were able to name capture an important and active slice of the Bitcoin network.

Having finally convinced ourselves of both the safety of Heuristic 2, by refining it substantially, and its effectiveness, as illustrated in Figure 4.5c, we use Heuristic 2 exclusively for the results in the next section.

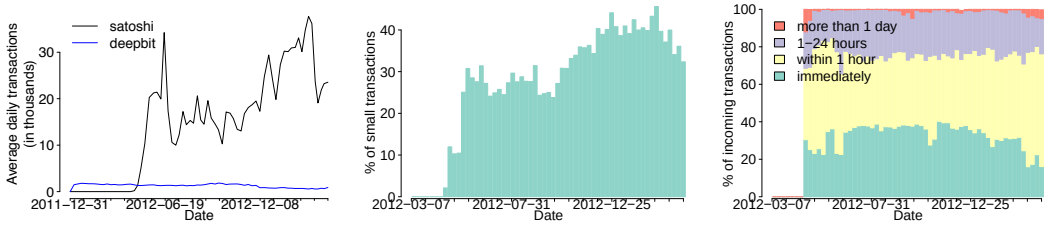
4.4 Service Centrality

In this section, we focus on two notable parts of the user graph seen in Figure 4.6: the component consisting of Satoshi Dice and the individuals who interact with it, and the heavily connected component consisting of most of the services we tagged. For both of these components, we argue that the demonstrated centrality of these services makes it difficult for even highly motivated individuals — e.g., thieves or others attracted to the anonymity properties of Bitcoin — to stay completely anonymous, provided they are interested in cashing out by converting to fiat money (or even other virtual currencies).

4.4.1 The effect of popular services

One of the largest stresses on the Bitcoin system to date has been the introduction of so-called dice games, and in particular Satoshi Dice, a betting game introduced in late April 2012. Briefly, users may place bets with various addresses, each of which is associated with a probability of winning (ranging from a 0.0015% chance of winning to a 97% chance). After determining if the user has won (using an algorithm involving the bet transaction and a random number), Satoshi Dice then sends some multiplier of the user's bet back to him if he won (e.g., 1.004 times his bet if he sent to the address with 97% winning odds), and 1 satoshi (0.00000001 BTC) if he lost.

Within weeks of being introduced, Satoshi Dice became wildly popular. Figure 4.7a shows its activity as compared to the activity of the Deepbit mining pool, which was arguably the most active user prior to the introduction of dice games. Satoshi Dice engages in tens of thousands of transactions per day, or about 60% of the overall activity in the Bitcoin network. It has also spawned a number of clones, such as BTC Dice, BTCLucky, Clone Dice, and DiceOnCrack (which, although less popular, are nevertheless quite well connected, as seen in Figure 4.6).



(a) **(b)** **(c)**

Figure 4.7. (In color online.) The effect Satoshi Dice has had on the Bitcoin network, in terms of both activity and its influence on trends. (a) The overall daily activity, in terms of number of both incoming and outgoing transactions, for Satoshi Dice, as compared to the overall activity for the mining pool Deepbit. (b) The percentage of transactions with less than 0.1 BTC in total value in which Satoshi Dice participated as the sender, shown as an average weekly percentage. (c) A breakdown of how quickly Satoshi Dice spends the money it receives: teal is immediately, yellow is within an hour, purple is within a day, and red is more than a day.

A number of factors help explain the popularity of Satoshi Dice. First, it allows users to place very small bets: the minimum bet for each category is 0.01 BTC, and over 21% of all bets (896,864 out of 4,127,979) are exactly this minimum value. Figure 4.7b shows that Satoshi Dice — just in terms of its outgoing transactions — accounts for anywhere between 30% and 40% of such micro-valued transactions (we found very similar results looking instead at the incoming transactions for Satoshi Dice). Referring back to Figure 4.2 and the rise of micro-valued transactions, we conclude that a large fraction of this rise can be attributed just to Satoshi Dice. In addition to allowing small bets, Satoshi Dice also acts extremely quickly. Once a bet is placed, the outcome is decided immediately and the payout is returned within seconds, as shown in Figure 4.7c. As with micro-valued transactions, referring back to Figure 4.3 indicates that Satoshi Dice also accounts for much of the rise of immediate spending (as a weekly average, nearly 50% of immediate transactions are due to Satoshi Dice).

Because of its immense popularity, and the extent to which it has inflated the size of the block chain (an extra 30,000 transactions translates into an extra 14MB added to the overall block chain daily), the opinion of Satoshi Dice in the Bitcoin community

is somewhat mixed: some decry it as a DoS attack,[5] while others appreciate that it has stress-tested the Bitcoin network.

It might be tempting to additionally think that, given the large amounts of bitcoins flowing through it, Satoshi Dice could act as a mix service:[6] if "dirty" bitcoins were gambled using 97% winning odds, and the resulting bitcoins were paid out to a different address, these bitcoins might at first glance appear to have no association with the gambled money (especially if they came from a different address than the gambled money was sent to, as is sometimes the case). Because the addresses that Satoshi Dice uses are public, however, it is trivial to observe when users are gambling; furthermore, in sending a bet to Satoshi Dice, a user must explicitly identify where the payout should be sent. Thus, without using services such as Satoshi Dice as a co-conspirator (which they seem to have no incentive to do, as they made over $500,000 in their first eight months alone [103]), the bitcoins paid out are indelibly linked to the ones that were placed as a bet.

4.4.2 Traffic analysis of illicit activity

We next turn our attention to another dominant category of service: exchanges. Although not nearly as active as Satoshi Dice, exchanges have essentially become chokepoints in the Bitcoin economy: to buy into or cash out of Bitcoin at scale, we argue that using an exchange is unavoidable. While sites like localbitcoins.com and bitcoinary.com do allow you to avoid exchanges (for the former, by matching up buyers directly with sellers in their geographic area), the current and historical volume on these sites does not seem to be high enough to support cashing out *at scale*.

For criminals, this centrality presents a unique problem: if a thief steals thousands of bitcoins, this theft is unavoidably visible within the Bitcoin network, and thus the

[5] http://en.bitcoin.it/wiki/SatoshiDice

[6] See, for example, early concerns at bitcointalk.org/index.php?topic=79079.0 and related discussions.

initial address of the thief is known and (as most exchanges try to maintain some air of reputability) he cannot simply transfer the bitcoins directly from the theft to a known exchange.[7] While he might attempt to use a mix service to hide the source of the money, we again argue that these services do not currently have the volume to launder thousands of bitcoins. As such, thieves have developed various strategies for hiding the source of the bitcoins that we explore in this section. In particular, we focus on the effectiveness of Heuristic 2 in de-anonymizing these flows, and thus in tracking illicitly-obtained bitcoins to exchanges (and thus, e.g., providing an agency with subpoena power the opportunity to learn whose account was deposited into, and in turn potentially the identity of the thief). For this approach to work, we do not need to (and cannot) account for each and every stolen bitcoin, but rather need to demonstrate only some flow of bitcoins directly from the theft to an exchange or other known institution.

To demonstrate the effectiveness of Heuristic 2 in this endeavor, we focus on an idiom of use that we call a "peeling chain." The usage of this pattern extends well beyond criminal activity, and is seen (for example) in the withdrawals for many banks and exchanges, as well as in the payouts for some of the larger mining pools. In a peeling chain, a single address begins with a relatively large amount of bitcoins (e.g., for mining pools it starts with the 25 BTC reward). A smaller amount is then "peeled" off this larger amount, creating a transaction in which a small amount is transferred to one address (e.g., 0.1 BTC for a miner payout), and the remainder is transferred to a one-time change address. This process is repeated — potentially for hundreds or thousands of hops — until the larger amount is pared down, at which point (in one usage) the amount remaining in the address might be aggregated with other such addresses to again yield a large amount

[7] Indeed, the Bitcoin community has recently demonstrated both the inherent traceability of thefts and the unwillingness to accept stolen money (see bitcointalk.org/index.php?topic=14085.msg1910231). After 923 BTC was stolen from the mining pool Ozcoin and transferred to a Strongcoin wallet, Strongcoin intercepted the bitcoins when the thief attempted to withdraw them and returned them to Ozcoin.

in a single address, and the peeling process begins again. By using Heuristic 2, we are able to track flows of money by following these change links systematically: at each hop, we look at the two output addresses in the transaction. If one of these outputs is a change address, we can follow the chain to the next hop by following the change address (i.e., the next hop is the transaction in which this change address spends its bitcoins), and can identify the meaningful recipient in the transaction as the other output address (the "peel").

Silk Road and Bitcoin Savings & Trust One of the most well-known and heavily scrutinized addresses in Bitcoin's history is `1DkyBEKt`,[8] which is believed to be associated with Silk Road and was active between January and September 2012. Starting in January, the address began to receive large aggregate sums of bitcoins; in the first of these, the funds of 128 addresses were combined to deposit 10,000 BTC into the `1DkyBEKt` address, and many transactions of this type followed (including one transaction in which the funds of 589 addresses were combined to deposit 8,000 BTC). All together, the address received 613,326 BTC in a period of eight months, receiving its last aggregate deposit on August 16 2012.

Then, starting in August 2012, bitcoins were aggregated and withdrawn from `1DkyBEKt`: first, amounts of 20,000, 19,000, and 60,000 BTC were aggregated and sent to separate addresses; later, 100,000 BTC each was sent to two distinct addresses, 150,000 BTC to a third, and 158,336 BTC to a fourth, effectively emptying the `1DkyBEKt` address of all of its funds. The balance of this address over time, as well as the balance of Silk Road and of vendors as a whole (as we consider Silk Road a vendor), is shown in Figure 4.8.

[8] Full address: `1DkyBEKt5S2GDtv7aQw6rQepAvnsRyHoYM`.

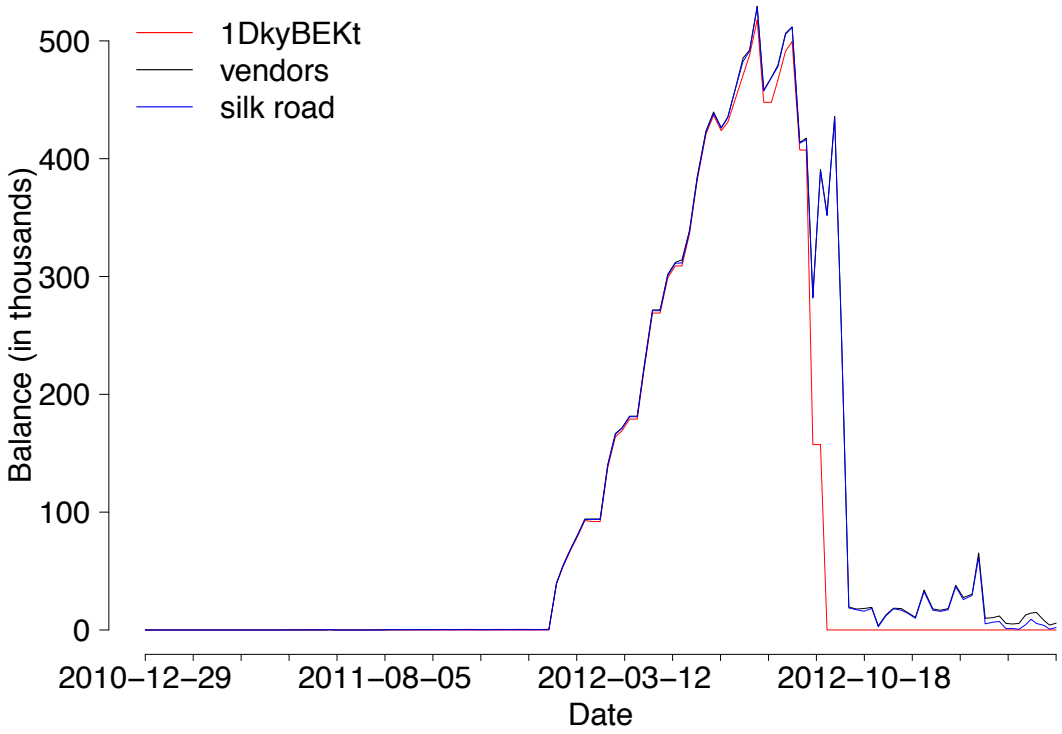


Figure 4.8. (In color online.) The balance of the vendors category (in black, although barely visible because it is dominated by Silk Road), Silk Road (in blue), and the `1DkyBEKt` address (in red).

Due to its large balance (at its height, it contained 5% of all generated bitcoins), as well as the curious nature of its rapidly accumulated wealth and later dissolution, this address has naturally been the subject of heavy scrutiny by the Bitcoin community. While it is largely agreed that the address is associated with Silk Road (and indeed our clustering heuristic did tag this address as being controlled by Silk Road), some have theorized that it was the "hot" (i.e., active) wallet for Silk Road, and that its dissipation represents a changing storage structure for the service. Others, meanwhile, have argued that it was the address belonging to the user pirate@40, who was responsible for carrying out the largest Ponzi scheme in Bitcoin history (the investment scheme Bitcoin Savings & Trust, which is now the subject of a lawsuit brought by the SEC [128]).

To see where the funds from this address went, and if they ended up with any known services, we first plotted the balance of each of the major categories of services, as

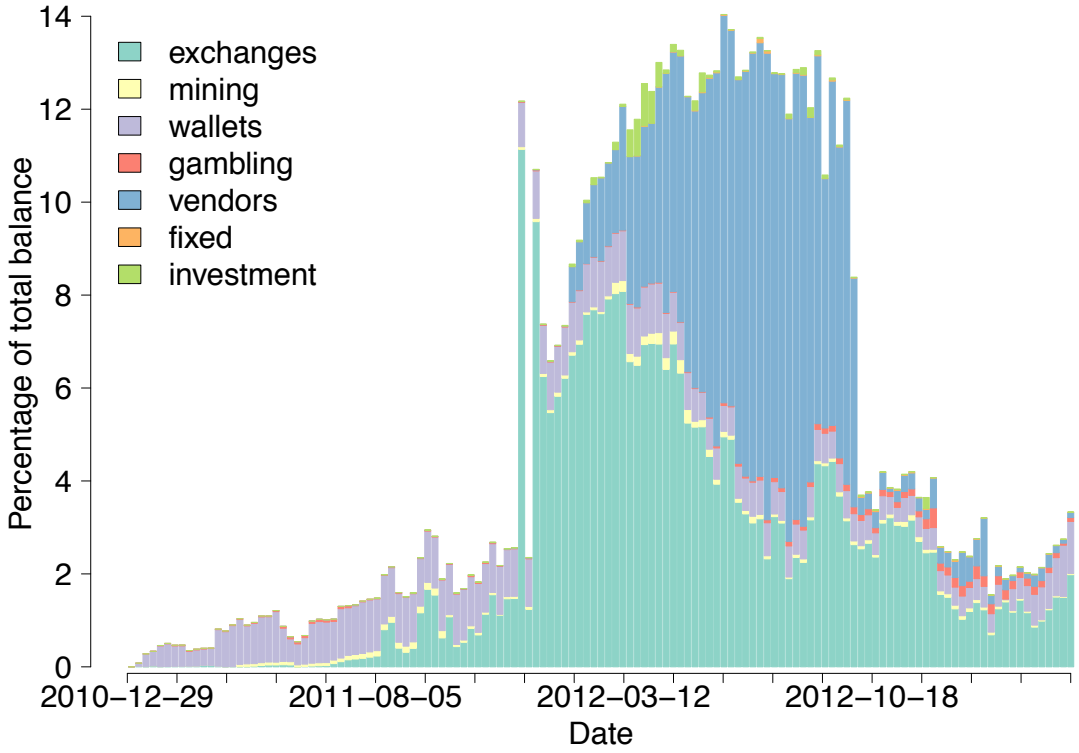


Figure 4.9. (In color online.) The balance of each major category, represented as a percentage of total active bitcoins; i.e., the bitcoins that are not held in sink addresses.

seen in Figure 4.9. Looking at this figure, it is clear that when the address was dissipated, the resulting funds were not sent en masse to any major services, as the balances of the other categories do not change significantly. To nevertheless attempt to find out where the funds did go, we turn to the traffic analysis described above.

In particular, we focus on the last activity of the `1DkyBEKt` address, when it deposited 158,336 BTC into a single address. This address then peeled off 50,000 BTC each to two separate addresses, leaving 58,336 BTC for a third address; each of these addresses then began a peeling chain, which we followed using the methodology described above (i.e., at each hop we continued along the chain by following the change address, and considered the other output address to be a meaningful recipient of the money). After following 100 hops along each chain, we observed peels to the services listed in Table 4.2.

Table 4.2. Tracking bitcoins from `1DkyBEKt`. Along the first 100 hops of the first, second, and third peeling chains resulting from the withdrawal of 158,336 BTC, we consider the number of peels seen to each service, as well as the total number of bitcoins (rounded to the nearest integer value) sent in these peels. The services are separated into the categories of exchanges, wallets, gambling, and vendors.

	First		Second		Third	
Service	Peels	BTC	Peels	BTC	Peels	BTC
Bitcoin-24			1	2	3	124
Bitcoin Central					2	2
Bitcoin.de					1	4
Bitmarket					1	1
Bitstamp			5	97	1	1
BTC-e					1	250
CA VirtEx	1	3	1	10	3	22
Mercado Bitcoin					1	9
Mt. Gox	11	492	14	70	5	35
OKPay	2	151			1	125
Instawallet	7	39	5	135	2	43
WalletBit	1	1				
Bitzino					2	1
Seals with Clubs	1	8				
Coinabul			1	29		
Medsforbitcoin	3	10				
Silk Road	4	28			5	102

Looking at this table, we see that, although a longitudinal look at the balances of major services did not reveal where the money went, following these chains revealed that bitcoins were in fact sent to a variety of services. The overall balance was not highly affected, however, as the amounts sent were relatively small and spread out over a handful of transactions. Furthermore, while our analysis does not itself reveal the owner of `1DkyBEKt`, the flow of bitcoins from this address to known services demonstrates the prevalence of these services (54 out of 300 peels went to exchanges alone) and provides the potential for further de-anonymization: the evidence that the deposited bitcoins were the direct result of either a Ponzi scheme or the sale of drugs might motivate Mt. Gox or

any exchange (e.g., in response to a subpoena) to reveal the account owner corresponding to the deposit address in the peel, and thus provide information to link the address to a real-world user.

Tracking thefts To ensure that our analysis could be applied more generally, we turned finally to a broader class of criminal activity in the Bitcoin network: thefts. Thefts are in fact quite common within Bitcoin: almost every major service has been hacked and had bitcoins (or, in the case of exchanges, other currencies) stolen, and some have shut down as a result.

To begin, we used a list of major Bitcoin thefts;[9] some of the thefts did not have public transactions (i.e., ones we could identify and study in the block chain), so we limited our attention to the ones that did. For each theft, we first found the specific set of transactions that represented the theft; i.e., the set of transactions in which the sender was the service being stolen from, and the recipient was the thief. Starting with these transactions, we did a preliminary manual inspection of the transactions that followed to determine their approximate type: we considered aggregations, in which bitcoins were moved from several addresses into a single one; folding, in which some of the addresses involved in the aggregation were not clearly associated with the theft, and thus were potentially there to "clean" the stolen money; splits, in which a large amount of bitcoins was split among two or more addresses; and finally peeling chains, in which relatively small amounts were peeled off from a succession of one-time change addresses holding a large amount of bitcoins. Our results are summarized in Table 4.3.

Briefly, the movement of the stolen money ranged from quite sophisticated layering and mixing to simple and easy to follow. Examining thefts therefore provides another demonstration of the potential for anonymity provided by Bitcoin, and the ways

[9] https://bitcointalk.org/index.php?topic=83794.0

Table 4.3. Tracking thefts. For each theft, we list (approximately) how many bitcoins were stolen, when the theft occurred, how the money moved after it was stolen, and whether we saw any bitcoins sent to known exchanges. For the movement, we use A to mean aggregation, P to mean a peeling chain, S to mean a split, and F to mean folding, and list the various movements in the order they occurred.

Theft	BTC	Date	Movement	Exchanges?
MyBitcoin	4019	Jun 2011	A/P/S	Yes
Linode	46,648	Mar 2012	A/P/F	Yes
Betcoin	3171	Mar 2012	F/A/P	Yes
Bitcoinica	18,547	May 2012	P/A	Yes
Bitcoinica	40,000	Jul 2012	P/A/S	Yes
Bitfloor	24,078	Sep 2012	P/A/P	Yes
Trojan	3257	Oct 2012	F/A	No

in which current usage falls short of this potential: for the thieves who used the more complex strategies, we saw little opportunity to track the flow of bitcoins (or at least do so with any confidence that ownership was staying the same), but for the thieves that did not there seemed to be ample opportunity to track the stolen money directly to an exchange.

One of the easiest thefts to track was from Betcoin, an early gambling site that was shut down after its server was hacked on April 11 2012 and 3,171 BTC were stolen in four installments of 2,902, 165, 17, and 87 BTC each. The stolen bitcoins then sat in the thief's address until March 15 2013 (when the bitcoin exchange rate began soaring), when they were aggregated with other small addresses into one large address that then began a peeling chain. After 10 hops, we saw a peel go to Bitcoin-24, and in another 10 hops we saw a peel go to Mt. Gox; in total, we saw 374.49 BTC go to known exchanges, all directly off the main peeling chain, which originated directly from the addresses known to belong to the thief. For some of the other thefts, de-anonymizing the flow of bitcoins was similarly straightforward: for the May 2012 Bitcoinica theft, for example, we observed one peeling chain, occurring directly after an aggregation of addresses belonging to the thieves, in which large amounts (i.e., hundreds of bitcoins) were peeled

off directly to known exchanges; in total, we saw 4,588 BTC peeled off to three different exchanges (BTC-e, CampBX, and Bitstamp). Again, although we do not account for every stolen bitcoin, watching even a portion of them flow to exchanges provides the opportunity we need to potentially compromise the anonymity of the thieves.

In contrast, some of the other thieves used more sophisticated strategies to attempt to hide the flow of money; e.g., for the Bitfloor theft, we observed that large peels off several initial peeling chains were then aggregated, and the peeling process was repeated. Nevertheless, by manually following this peel-and-aggregate process to the point that the later peeling chains began, we systematically followed these later chains and again observed peels to multiple known exchanges: the third peel off one such chain was 191.09 BTC to Mt. Gox, and in total we saw 661.12 BTC sent to three popular exchanges (Mt. Gox, BTC-e, and Bitstamp).

Even the thief we had the most difficulty tracking, who stole bitcoins by installing a trojan on the computers of individual users, seemed to realize the difficulty of cashing out at scale. Although we were unable to confidently track the flow of the stolen money that moved, most of the stolen money did not in fact move at all: of the 3,257 BTC stolen to date, 2,857 BTC was still sitting in the thief's address, and has been since November 2012.

With these thefts, our ability to track the stolen money provides evidence that even the most motivated Bitcoin users (i.e., criminals) are engaging in idioms of use that allow us to erode their anonymity. While one might argue that thieves could easily thwart our analysis, as Heuristic 2 is admittedly not robust in the face of adversarial behavior, our observation is that — at least at present — none of the criminals we studied seem to have taken such precautions. We further argue that the fairly direct flow of bitcoins from the point of theft to the deposit with an exchange provides some evidence that using exchanges to cash out at scale is inevitable, and thus that — again, at present —

Bitcoin does not provide a particularly easy or effective way to transact large volumes of illicitly-obtained money.

Acknowledgments

We would like to thank Brian Kantor and Cindy Moore for managing our systems and storage needs, and for helping to set up and maintain our mining rig. We are also grateful to Andreas Pitsillidis for his advice in creating figures and overall useful discussions.

Chapter 4, in part, is a reprint of the material as it appears in *Proceedings of the 13th ACM SIGCOMM Conference on Internet Measurement*. Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage, ACM, November 2013. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Conclusions

In this dissertation, we described two distinct — but complementary — methods for approaching secure systems. The first, a cryptographic approach, considered how to use knowledge of practical adversarial attacks on systems to update formal cryptographic models, and then how to provide strong, provably secure solutions within those updated models. The second, a measurement approach, considered how users engage with Bitcoin, a successfully deployed currency system with the goal of providing both flexibility and anonymity for its users. Both of these approaches have their strengths and weaknesses: the cryptographic approach can guarantee in a provable way the security of a system, even before it is deployed, but only if the system is modeled correctly. In contrast, the measurement approach can get at the heart of how a secure system is really being used, but can only be adopted after the system is already deployed.

In looking at the formal models for digital signatures, we identified three real-world settings in which the existing standard model failed to provide any meaningful notion of security. Two of these settings (RKA and joint encryption and signatures) had been previously examined and appropriately modeled, but we provide the first model for the third (KDM storage). Moreover, we provided a general tool — key-versatile signatures — that enables simple, modular constructions in each of these distinct settings, while relying only on mild cryptographic assumptions and providing strong notions of

security. Going forward, our hope is that this general tool will continue to enable the flexible and simple construction of secure cryptographic primitives.

In looking at the Bitcoin network, we presented a longitudinal characterization, focusing particular on the growing gap between the potential for anonymity that Bitcoin provides and the anonymity that average users are actually achieving. Interestingly, this gap is due neither to the intentional actions of an adversary or to a fundamental flaw in the Bitcoin protocol, but rather to certain standard patterns — or idioms of use — that honest users have themselves adopted. While our methods for de-anonymization are not fully robust in the face of changing patterns of usage, at present we believe that to thwart them completely would require a significant effort (both computational and financial) on the part of the user. Our conclusion is therefore that for all but the most motivated users, Bitcoin is not as anonymous as previously believed, and that even for these motivated users, engaging in criminal activity at scale in the Bitcoin network is still sufficiently challenging as to be relatively unattractive.

Bibliography

[1] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236. Springer, August 2010.

[2] Advanced encryption standard (aes). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, November 2001.

[3] Devdatta Akhawe and Adrienne Porter Felt. Alice in warningland: A large-scale field study of browser security warning effectiveness. In *Proceedings of the 22nd USENIX Conference on Security*, 2013.

[4] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2013.

[5] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, April / May 2002.

[6] Ross Anderson and Markus Kuhn. Tamper resistance – a cautionary note. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pages 1–11, 1996.

[7] Gavin Andresen. bitcointools. github.com/gavinandresen/bitcointools.

[8] Elli Androulaki, Ghassan Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating User Privacy in Bitcoin. In *Proceedings of Financial Cryptography 2013*, 2013.

[9] Benny Applebaum. Key-dependent message security: Generic amplification and completeness. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 527–546. Springer, May 2011.

[10] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, August 2009.

[11] Michael Backes, Markus Dürmuth, and Dominique Unruh. OAEP is secure under key-dependent messages. In Josef Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 506–523. Springer, December 2008.

[12] Michael Backes, Birgit Pfitzmann, and Andre Scedrov. Key-dependent message security under active attacks - brsim/uc-soundness of dolev-yao-style encryption with key cycles. *Journal of Computer Security*, 16(5):497–530, 2008.

[13] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 423–444. Springer, May 2010.

[14] Justin Becker and Hao Chen. Measuring privacy risk in online social networks. In *Proceedings of W2SP 2009: Web 2.0 Security and Privacy*, 2009.

[15] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 666–684. Springer, August 2010.

[16] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 486–503. Springer, December 2011.

[17] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, August 1998.

[18] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 194–211. Springer, August 1990.

[19] Mihir Bellare and Sriram Keelveedhi. Authenticated and misuse-resistant encryption of key-dependent data. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 610–629. Springer, August 2011.

[20] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 491–506. Springer, May 2003.

[21] Mihir Bellare, Sarah Meiklejohn, and Susan Thomson. Key-Versatile Signatures and Applications: RKA, KDM, and Joint Enc/Sig. In *Advances in Cryptology - EUROCRYPT '14*, volume 8441 of *Lecture Notes in Computer Science*. Springer, 2014.

[22] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology*, 21(4):469–491, October 2008.

[23] Mihir Bellare, Kenneth G. Paterson, and Susan Thomson. RKA security beyond the linear barrier: IBE, encryption and signatures. In *ASIACRYPT*, pages 331–348, 2012.

[24] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.

[25] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, May / June 2006.

[26] Eli Biham. New types of cryptoanalytic attacks using related keys (extended abstract). In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 398–409. Springer, May 1993.

[27] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, August 1997.

[28] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for tor hidden services: Detection, measurement, deanonymization. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2013.

[29] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 520–537. Springer, August 2010.

[30] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002: 9th Annual International Workshop on Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 62–75. Springer, August 2003.

[31] Stevens Le Blond, Pere Manils, Abdelberi Chaabane, Mohamed Ali Kaafar, Claude Castelluccia, Arnaud Legout, and Walid Dabbous. One bad apple spoils the bunch: Exploiting P2P applications to trace and profile tor users. In *Proceedings of the 4th USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2011.

[32] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[33] Manuel Blum and Shafi Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 289–302. Springer, August 1985.

[34] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 2004.

[35] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, May 2004.

[36] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2007.

[37] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, May 1997.

[38] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125. Springer, August 2008.

[39] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*,

volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, December 2001.

[40] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2012.

[41] Joseph Bonneau and Ilya Mironov. Cache-collision timing attacks against AES. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 201–215. Springer, October 2006.

[42] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 05: 12th Conference on Computer and Communications Security*, pages 320–329. ACM Press, November 2005.

[43] Zvika Brakerski, Shafi Goldwasser, and Yael Tauman Kalai. Black-box circular-secure encryption beyond affine functions. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 201–218. Springer, March 2011.

[44] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, August 2011.

[45] Jan Camenisch, Nishanth Chandran, and Victor Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 351–368. Springer, April 2009.

[46] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer, September 2002.

[47] Ran Canetti, Yael Tauman Kalai, Mayank Varia, and Daniel Wichs. On symmetric encryption and point obfuscation. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 52–71. Springer, February 2010.

[48] Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a SSL/TLS channel. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 583–599. Springer, August 2003.

[49] CBC News. Revenue Canada says BitCoins aren't tax exempt, April 2013. www.cbc.ca/news/canada/story/2013/04/26/business-bitcoin-tax.html.

[50] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: Complex unary transformations and delegatable anonymous credentials. Cryptology ePrint Archive, Report 2013/179, 2013.

[51] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 78–96. Springer, August 2006.

[52] Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. Universal padding schemes for RSA. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 226–241. Springer, August 2002.

[53] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, August 1998.

[54] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[55] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, February 2001.

[56] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 566–598. Springer, August 2001.

[57] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Necessary and sufficient assumptions for non-interactive zero-knowledge proofs of knowledge for all np relations. In *Automata, Languages and Programming*, pages 451–462. Springer, 2000.

[58] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on*, pages 427–436. IEEE, 1992.

[59] Jean Paul Degabriele, Anja Lehmann, Kenneth G. Paterson, Nigel P. Smart, and Mario Strefler. On the joint security of encryption and signature in EMV. In Orr Dunkelman, editor, *Topics in Cryptology – CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 116–135. Springer, February / March 2012.

[60] Rachna Dhamija, J.D. Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 2006.

[61] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. Internet RFC 4346, 2006.

[62] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631. Springer, December 2010.

[63] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[64] Zakir Durumeric, James Kasten, Michael Bailey, and J. Alex Halderman. Analysis of the HTTPS certificate ecosystem. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement*, 2013.

[65] Serge Egelman, Lorrie Faith Cranor, and Jason Hong. You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, 2008.

[66] Brian Patrick Eha. Get ready for a Bitcoin debit card. CNNMoney, April 2012. money.cnn.com/2012/08/22/technology/startups/bitcoin-debit-card/index.html.

[67] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, August 1985.

[68] EMV Co. Emv book 1 – application independent icc to terminal interface requirements – version 4.2, June 2008.

[69] EMV Co. Emv book 2 – security and key management – version 4.2, June 2008.

[70] EMV Co. Emv book 3 – application specification – version 4.2, June 2008.

[71] EMV Co. Emv book 4 – cardholder, attendant, and acquirer interface requirements – version 4.2, June 2008.

[72] European Central Bank. Virtual Currency Schemes. ECB Report, October 2012. www.ecb.europa.eu/pub/pdf/other/virtualcurrencyschemes201210en.pdf.

[73] Jia Fan, Yuliang Zheng, and Xiaohu Tang. A single key pair is adequate for the zheng signcryption. In Udaya Parampalli and Philip Hawkes, editors, *ACISP 11: 16th Australasian Conference on Information Security and Privacy*, volume 6812 of *Lecture Notes in Computer Science*, pages 371–388. Springer, July 2011.

[74] Federal Bureau of Investigation. (U) Bitcoin Virtual Currency Unique Features Present Distinct Challenges for Deterring Illicit Activity. Intelligence Assessment, Cyber Intelligence and Criminal Intelligence Section, April 2012. cryptome.org/2012/05/fbi-bitcoin.pdf.

[75] FinCEN. Application of FinCEN's Regulations to Persons Administering, Exchanging, or Using Virtual Currencies, March 2013. www.fincen.gov/statutes_regs/guidance/pdf/FIN-2013-G001.pdf.

[76] David Galindo, Javier Herranz, and Jorge L. Villar. Identity-based encryption with master key-dependent message security and leakage-resilience. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS 2012: 17th European Symposium on Research in Computer Security*, volume 7459 of *Lecture Notes in Computer Science*, pages 627–642. Springer, September 2012.

[77] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, pages 467–476. ACM, 2013.

[78] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277. Springer, February 2004.

[79] David Goldenberg and Moses Liskov. On related-secret pseudorandomness. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 255–272. Springer, February 2010.

[80] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology — CRYPTO 2013*, volume 8043 of *Lecture Notes in Computer Science*, pages 536–553. Springer, August 2013.

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

[82] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[83] Philipp Grabher, Johann Großschädl, and Dan Page. Cryptographic side-channels from low-power cache memory. In Steven D. Galbraith, editor, *11th IMA International Conference on Cryptography and Coding*, volume 4887 of *Lecture Notes in Computer Science*, pages 170–184. Springer, December 2007.

[84] Andy Greenberg. Here's What It's Like To Buy Drugs On Three Anonymous Online Black Markets, August 2013. www.forbes.com/sites/andygreenberg/2013/08/14/heres-what-its-like-to-buy-drugs-on-three-anonymous-online-black-markets/.

[85] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, December 2006.

[86] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 323–341. Springer, August 2007.

[87] Stuart Haber and Benny Pinkas. Securely combining public-key cryptosystems. In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pages 215–224. ACM Press, November 2001.

[88] Kristiyan Haralambiev. *Efficient Cryptographic Primitives for Non-Interactive Zero-Knowledge Proofs and Applications*. PhD thesis, New York University, May 2011.

[89] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In *Proceedings of the 21st USENIX Conference on Security*, 2012.

[90] Dennis Hofheinz. Circular chosen-ciphertext security with compact ciphertexts. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 520–536. Springer, 2013. To appear.

[91] Mei-Chen Hsueh, Timothy K. Tsai, and Ravishankar K. Iyer. Fault injection techniques and tools. *Computer*, 30(4):75–82, April 1997.

[92] Rob Jansen, Florian Tschorsch, Aaron Johnson, and Björn Scheuermann. The sniper attack: Anonymously deanonymizing and disabling the tor network. In *Proceedings of the 21st Annual Symposium on Network and Distributed System Security*, 2014.

[93] Vlastimil Klíma, Ondrej Pokorný, and Tomás Rosa. Attacking RSA-based sessions in SSL/TLS. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 426–440. Springer, September 2003.

[94] Lars R. Knudsen. Cryptanalysis of LOKI91. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology – AUSCRYPT'92*, volume 718 of *Lecture Notes in Computer Science*, pages 196–208. Springer, December 1992.

[95] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, August 1996.

[96] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, August 1999.

[97] Yuichi Komano and Kazuo Ohta. Efficient universal padding techniques for multiplicative trapdoor one-way permutation. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 366–382. Springer, August 2003.

[98] Michal Kosinski, David Stillwell, and Thore Graepel. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805, 2013.

[99] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer, August 2004.

[100] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

[101] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Public keys. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 626–642. Springer, August 2012.

[102] Tal Malkin, Isamu Teranishi, and Moti Yung. Efficient circuit-size independent public key encryption with KDM security. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 507–526. Springer, May 2011.

[103] Jon Matonis. Bitcoin Casinos Release 2012 Earnings. Forbes, January 2013. www.forbes.com/sites/jonmatonis/2013/01/22/bitcoin-casinos-release-2012-earnings/.

[104] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A Fistful of Bitcoins: Characterizing Payments Among Men with No Names. In *Proceedings of the 13th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2013.

[105] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer, August 1990.

[106] M. González Muñiz and R. Steinwandt. Security of signature schemes in the presence of key-dependent messages. *Tatra Mt. Math. Publ.*, 47:15–29, 2010.

[107] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of tor. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2005.

[108] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *ACM CCS 98: 5th Conference on Computer and Communications Security*, pages 59–66. ACM Press, November 1998.

[109] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. bitcoin.org/bitcoin.pdf.

[110] Lasse Øverlier and Paul Syverson. Locating hidden servers. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2006.

[111] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 1999.

[112] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 372–389. Springer, December 2011.

[113] Kenneth G. Paterson, Jacob C. N. Schuldt, Martijn Stam, and Susan Thomson. On the joint security of encryption and signature, revisited. In Dong Hoon Lee

and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 161–178. Springer, December 2011.

[114] Morgen Peck. Bitcoin-Central is Now The World's First Bitcoin Bank...Kind Of. IEEE Spectrum: Tech Talk, December 2012. spectrum.ieee.org/tech-talk/telecom/internet/bitcoincentral-is-now-the-worlds-first-bitcoin-bankkind-of.

[115] Michael O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, January 1979.

[116] Fergal Reid and Martin Harrigan. An Analysis of Anonymity in the Bitcoin System. In *Security and Privacy in Social Networks*, pages 197–223. Springer New York, 2013.

[117] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128. Springer, May 2011.

[118] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[119] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 02: 9th Conference on Computer and Communications Security*, pages 98–107. ACM Press, November 2002.

[120] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing*, pages 387–394. ACM Press, May 1990.

[121] Dorit Ron and Adi Shamir. Quantitative Analysis of the Full Bitcoin Transaction Graph. In *Proceedings of Financial Cryptography 2013*, 2013.

[122] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553. IEEE Computer Society Press, October 1999.

[123] Sandvine. Global internet phenomena repart 2h2013, 2013.

[124] Stuart Schechter, A.J. Bernheim Brush, and Serge Egelman. It's no secret: Measuring the security and reliability of authentication via 'secret' questions. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2009.

[125] Stuart E Schecter, Rachna Dhamija, Andy Ozment, and Ian Fischer. The emperor's new security indicators. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2007.

[126] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, August 1990.

[127] Certicom research, standards for efficient cryptography group (SECG) — sec 1: Elliptic curve cryptography. http://www.secg.org/secg_docs.htm, September 20, 2000. Version 1.0.

[128] Securities and Exchange Commission. SEC Charges Texas Man With Running Bitcoin-Denominated Ponzi Scheme, July 2013. www.sec.gov/News/PressRelease/Detail/PressRelease/1370539730583.

[129] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer, August 2002.

[130] Emily Spaven. CryptoLocker malware demands bitcoin ransom, October 2013. www.coindesk.com/cryptolocker-malware-demands-bitcoin-ransom.

[131] Serge Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS ... In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–546. Springer, April / May 2002.

[132] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, May 2005.

[133] Andrew White, Austin Matthews, Kevin Snow, and Fabian Monrose. Phonatic Reconstruction of Encrypted VoIP Conversations: Hookt on fon-iks. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2011.

[134] Charles V. Wright, Lucas Ballard, Scott E. Coull, Fabian Monrose, and Gerald M. Masson. Spot Me if You Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 35–49, 2008.

[135] Xiaohan Zhao, Alessandra Sala, Christo Wilson, Xiao Wang, Sabrina Gaito, Haitao Zheng, and Ben Y. Zhao. Multi-scale dynamics in a massive online social network. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement*, 2012.

[136] Ziming Zhao, Gail-Joon Ahn, Jeong-Jin Seo, and Hongxin Hu. On the security of picture gesture authentication. In *Proceedings of the 22nd USENIX Conference on Security*, 2013.

[137] Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) ¡¡ cost(signature) + cost(encryption). In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer, August 1997.

[138] znort987. blockparser. github.com/znort987/blockparser.