Homomorphic Signature Schemes

A Thesis presented

by

David Molnar

to

Computer Science

in partial fulfillment of the honors requirements

for the degree of

Bachelor of Arts

Harvard College

Cambridge, Massachusetts

April 8, 2003

Contents

Acknowledgements

All original work in this thesis is joint work; I am indebted to my collaborators for their insight, good humor, and patience. Parts of the thesis are adapted from a collaboration with Rob Johnson, Dawn Song, and David Wagner that appeared as a paper in the 2002 RSA Conference[33]. All work regarding directed transitive signature schemes and trap-door inversion is joint with Susan Hohenberger and Ron Rivest; parts of this work will also appear in Susan's masters thesis. This thesis would not have been possible without my thesis advisor, Michael Rabin, and my thesis readers, Michael Mitzenmacher and Salil Vadhan; I appreciate all their hard work in making sense of this document.

I am also indebted to the readers of the usenet newsgroup `sci.crypt` Eric Bach and Mika R.S. Kojo made early observations concerning the triviality of additive signature schemes. Without their observations, I would not have followed up this problem. Eike Kiltz provided valuable pointers to results concerning the Diffie-Hellman Inversion problem that saved us work. David Wagner and I began our collaboration after an e-mail exchange following a `sci.crypt` post.

In addition, Dan Boneh offered encouragement, as well as advice concerning the use of bilinear mappings. These mappings did not end up in this thesis, but that is my fault, not his. Adrian Perrig suggested the use of tree schemes for redactable signatures.

Ron Rivest's inspiring talk on "Two New Signature Schemes" kicked off this work. I thank him and the other researchers in this area, including Mihir Bellare, Suresh Chari, Silvio Micali, Daniele Micciancio, Alejandro Hevia, Gregory Neven, Huafei Zhu, Bao Feng, Robert H. Deng, and Tal Rabin for providing these interesting problems.

I have been fortunate to know several excellent teachers in cryptography and security. Henry Cohn gave a math tutorial that was perhaps the best course I took at Harvard. Daniele Micciancio and Shafi Goldwasser co-taught 6.876 and introduced me to high-paced research. Ron Rivest's 6.857 class gave me a glimpse of the breadth in the field. Silvio Micali's 6.875 and 6.876 showed me how research evolves and pushes new directions over time. Radia Perlman sees to the practical heart of any matter. Finally, my experience as a teaching fellow for Michael Rabin was formative and fun.

I also thank the Harvard Computer Science Department and Harvard College, especially my concentration advisor Stuart Shieber, for a full education.

Finally, many thanks to family and friends, too numerous to list here. If I have left anyone out, I apologize.

Chapter 1

Introduction

Digital signature schemes with special algebraic properties offer new applications and theoretical challenges. A digital signature scheme mimics the functionality of real-world signatures. Alice creates a signature $SIG_A(M)$ on a message M using a secret key only she knows. Anyone can verify that Alice created $SIG_A(M)$ and so meant to sign M. No one else can create $SIG_A(M)$ without access to the secret key. In addition, if M is modified in any way, this modification can be detected.

The best known digital signature scheme is due to Rivest, Shamir, and Adleman [49]. In their scheme, a message is signed by computing $m^d \mod n$ for a composite modulus n. The exponent d is kept secret, but the modulus n and a public exponent e such that $(m^d)^e = m \mod n$ is published and associated with the signer. As far as anyone knows, it is difficult to create m^d given only e and n. Because only the signer could create m^d for any given message m, and because anyone can check that $(m^d)^e = m \mod n$, anyone can distinguish a legitimate signature from an attempted forgery.

Digital signatures are a fundamental primitive in cryptography. A bewildering array of variants on the basic theme of digital signature schemes have been proposed. We focus on signature schemes with special algebraic properties. Usually, algebraic properties in signature schemes are a problem. For example, RSA is a multiplicative homomorphism, as $m_1^d \cdot m_2^d = (m_1 m_2)^d$. Several attacks are known that exploit this multiplicative property of RSA. Can this algebraic property be used for positive uses instead?

Rivest pointed out in a series of talks that the answer is *yes*. Signature schemes with algebraic properties can enable new applications[50]. In these schemes, a "master signer" first uses the secret key to create and distribute signatures. Then anyone else, *without the secret key*, can combine these signatures to form new ones.

For an example of an application enabled by homomorphic signatures, consider the problem of "route aggregation." In the OSPF routing protocol, routers are assigned hierarchically to subnets. We can think of these subnets as forming a tree. Routers at each node are authorized to direct packets for all subnets in their children. We would like to construct a protocol where, for each address it claims authority to route for, a router must give a signed statement saying that it is allowed to route to that address. At the same time, we want only to sign the *leaves* of the routing tree.

Now in this scenario, a router can "aggregate" subnets by simply presenting all the signatures of the leaf subnets for which it is authorized. The question is this: how can a router show it is authorized to route packets for a collection of child subnets *without* exhibiting signatures from all of them? Such a collection of signatures taxes storage and computation resources, and in addition reveals too much information about the internal structure of a network.

Chari, Rabin, and Rivest solve the problem by using the multiplicative property of RSA together with a clever trick[13]. Each node in the routing tree is labelled such that the label of each node is the *product* of the labels of its two children. Then the labels are signed by the "master signer." Because RSA is a multiplicative homomorphism, anyone can multiply signatures on the labels of two children to obtain a signature on the label of their parent.

For another application, consider Bob, who has just been drafted into the U.S. Army. As a newly minted private, Bob has an immediate superior, his sergeant. The sergeant has a superior, and so on until we reach the Commander-in-Chief of the United States Armed Forces, the President of the United States.

We can think of Army as a graph, with nodes as people, and edges between people who are above or below each other in the chain of command. An edge between Alice and Bob then means that one works with the other.

Now suppose Bob wants to prove that he works for the President of the United States. Instead of submitting signatures on each of the edges (and revealing everyone on the path between Bob and the President), it would be better to simply submit a signature on the edge between Bob and the President. To do this, we need some way to "combine" signatures of edges, just as we combined child nodes into a parent node in the route aggregation scheme. In section 4.1 we will describe "transitive signature schemes" that allow exactly this kind of graph edge signature aggregation. The first such scheme was due to Micali and Rivest; later Bellare and Neven provided alternate implementations[38, 5].

These constructions of transitive signature schemes also achieve *privacy*: given a signature, it is information-theoretically impossible to determine whether the signature is "original" or was formed via composition. This allows for Bob to avoid divulging his exact chain of command, because the combined signature carries no information about the chain of signatures that produced it.

We can also use special signatures to depersonalize databases while maintaining integrity. For example, a medical doctor may compile data on patients in the course of his or her work. The data is signed by the doctor to prevent tampering. Later, a researcher comes along and wants access to the medical data.

For ethical reasons, the researcher does not need or want personal identifying information in these records. At the same time, the researcher would like the data to be signed, and perhaps time-stamped, by the doctor. Using a traditional signature scheme, the doctor must prepare a new version of the data and resign it from scratch. This is time-consuming and requires access to the doctor's secret key, opening the possibility of data corruption. Time-stamps could be preserved only with difficulty, and the researcher must divulge his or her criteria for "important data" to the doctor.

It would be better if there were some way for a third party, without access to the secret key, to cross out or "redact" only the information the researcher does not want *while ensuring the signature still verifies.* Then the desired content could be "extracted" and passed along to the researcher.

In section 4.2 we show two different schemes that support this "redaction" or "content extraction." The first scheme depends on a hash tree construction and is due to Johnson, Molnar, Song, and Wagner [33]. The second exploits the multiplicative property of RSA and is due to Steinfeld, Bull, and Zheng[55].

Rivest also left as an open problem the creation of a "concatenation homomorphic" signature scheme. Given the signature on m_1 and the signature on m_2, the signature on $m_1||m_2$ can be derived. Martin Hirt has suggested a construction that has indistinguishability of composed and original signatures, but the signatures are linear in the length of the message [30]. Finding a more efficient scheme remains an open problem.

Homomorphic signature schemes also lead to challenging theoretical issues. Even the correct definition of a homomorphic signature scheme is not as easy as it may first appear. In sec 3.7 we show that "additive" schemes, in which one can both add and subtract signatures, are secure according to a natural definition of security but are "trivial." We suggest the notion of "security against random forgeries" as a possible alternative definition and discuss some problems with the notion. Creating a satisfactory *general* definition is still an open problem. We discuss desiderata in such a definition.

Bellare and Neven discuss improved definitions for the special case of transitive signature schemes[5]. We follow this discussion and extend it slightly with some new distinctions and possible weakenings of the problem.

In both the examples given above for transitive signature schemes, the direction of the graph is important. Bob works for the President, not the other way around. We investigate the case of *directed* transitive signature schemes. The schemes discussed in section 4.1 work only on undirected graphs. Micali and Rivest left it as an open problem to find schemes for the directed case. We do *not* solve this problem, but we reduce it to a new open problem:

> OPEN PROBLEM Find a group with "trapdoor infeasible inversion." That is, a group where composition is easy, but finding inverses is hard unless a trapdoor is known.

Our hope is to open avenues of attack by reducing directed transitive schemes to new problems closer to core mathematics. We further show some implications of infeasible inversion. A group with infeasible inversion, even without a trapdoor for inversion, implies the existence of key agreement protocols, as well as the "strong associative one-way functions" of Rabi and Sherman[46]. Therefore, by the results of Impagliazzo and Rudich, we know there exists no black-box construction of such a group from one-way functions unless $P \neq NP$ [32]. This result allows us to situate infeasible inversion in the "black-box hierarchy" of cryptographic primitives. We also show that signatures homomorphic with respect to union, but not set difference, form a group with infeasible inversion.

1.1 Related Work

Homomorphic signature schemes are intriguing in part because homomorphic cryptosystems have proved to be so useful. Rivest, Adleman, and Dertouzos noted applications of "privacy homomorphisms" to computing on encrypted data soon after the introduction of RSA [48]. Peralta and Boyar showed that an XOR-homomorphic bit commitment could be exploited to yield more efficient zero-knowledge proofs of circuit satisfiability [45]. Feigenbaum and Merritt noted that a "cryptosystem which is a ring homomorphism on $\mathbb{Z}/2\mathbb{Z}$ could be used to implement completely non-interactive secure circuit evaluation" and called such cryptosystems "algebraic" [19]. Benaloh gave a secure election scheme based on a homomorphic encryption scheme [14]. Cramer and Damgard use homomorphic bit commitments to drastically simplify zero-knowledge proofs [16]. Many other examples exist in which homomorphic properties are used to construct cryptographic protocols.

The initial promise of privacy homomorphisms was tempered by a string of negative results. Ahituv, Lapid, and Neumann showed that any cryptosystem that is XOR-homomorphic on $GF(2^{64})$ is insecure under chosen ciphertext attack [1]. Boneh and Lipton showed that any deterministic cryptosystem that is a field homomorphism must fall victim to a subexponential attack [8]. They further conjectured that any field-homomorphic cryptosystem, which they called "completely malleable," would prove to be insecure. Brickell and Yacobi broke a number of candidate constructions of privacy homomorphisms [10].

Besides RSA, several other homomorphic cryptosystems are currently known. Goldwasser-Micali encryption takes the form of a group homomorphism $\mathbb{Z}/2\mathbb{Z} \to (\mathbb{Z}/n\mathbb{Z})^*$ [25], and others have proposed a number of other public-key encryption schemes that have various useful homomorphic properties [21, 6, 43, 39]. Of particular interest is Sander, Young, and Yung's slick construction of an encryption algorithm that is both AND- and XOR-homomorphic [52]; they note that this is the first cryptosystem homomorphic over a semigroup.

Redactable signature schemes are related in both spirit and construction to the "incremental cryptography" of Goldwasser, Goldreich, and Bellare [3]. The main difference is that the "incremental cryptography" constructions only allow updates to the signature by parties that have access to the secret key; in the homomorphic signature setting, no such access is needed. The notion of privacy for redactable signatures has a parallel in their notion of privacy for incremental signatures [4].

Homomorphic signature schemes themselves were first discussed by Rivest in a series of talks [50]. These talks included discussion of the route aggregate signature scheme of Chari, Rabin, and Rivest and the undirected transitive signature scheme of Micali and Rivest. Later, Bellare and Neven furnished a proof for the "natural" undirected transitive scheme based on RSA, which had resisted attempts to prove its security; their proof required a special "one-more inversion" assumption for RSA. Bellare and Neven also introduced improved definitions for the case of transitive signatures and gave several new instantiations of the primitive. Recently, Zhu, Feng, and Deng gave a transitive signature scheme with a proof in the "standard model" (without random oracles) and related it to the problem of evaluating reputation in networks [57].

Johnson, Molnar, Song, and Wagner discussed definitions of homomorphic signatures and introduced several new constructions, such as redactable signatures and set signatures. Independently, Steinfeld, Bull, and Zheng proposed "content-extraction signatures," which are very similar to redactable signatures. We discuss some of the similarities and differences in Section 4.2.3.

Micciancio and Hevia gave the first constructions of homomorphic *one-time* signatures[29]. Their constructions allow for signatures homomorphic with respect to union and subset, and arose in the context of proving the security of general one-time signature schemes. A point of interest is that these constructions require only one-way functions.

1.2 Contributions of This Thesis and Statement on Joint Work

This thesis contains both original work and expository background. The original contributions of the thesis are the following:

- Definitions. We build on the discussion of Bellare and Neven by extending the definition of transitive signature schemes to directed graphs. We also introduce several alternative, weaker requirements for signature schemes. The first discussions on definitions were joint with Rob Johnson, Dawn Song, and David Wagner and appeared in a joint paper at the RSA 2002 Conference[33]. The extension to directed graphs and weakenings of the problem of directed transitive schemes is joint with Susan Hohenberger and Ron Rivest.

- Triviality of Additive Signature Schemes. We give an example of an "additive" signature scheme that satisfies a natural definition of security for homomorphic schemes, yet is "trivial." Given signatures on any two relatively prime messages, the signature on any other message can be obtained. This is joint with Johnson, Song, and Wagner.

- Redactable Signatures. We give a construction for a redactable signature scheme that is joint work with Johnson, Song, and Wagner. Two similar schemes were also proposed independently by Bull, Steinfeld, and Zheng; in section 4.2 we discuss similarities and differences.

- Set Homomorphic Signatures. We give a construction for a signature scheme on sets that is homomorphic with respect to union and subset operations. This is joint with Johnson, Song, and Wagner.
- Towards Directed Transitive Signature Schemes. We examine the open problem of "directed transitive signature schemes" and reduce it to two new open problems, such that solving either one would yield such a scheme. This work is joint with Hohenberger and Rivest.
- Implications of Infeasible Inversion. We then situate one of the open problems, that of finding a "group with infeasible inversion," in the black-box hierarchy of cryptographic primitives. This work is joint with Hohenberger and Rivest.

Expository work is cited in text as necessary.

Chapter 2

Preliminaries

We collect some helpful definitions. The interested reader may consult an algorithms book, such as Cormen, Leiserson, Rivest, and Stein, for general background [15]. Goldreich and Menenzes, Van Oorschot, and Vanstone together cover most of the cryptographic definitions we need and more besides [24, 36].

2.1 Graphs

In this thesis, $G = (V, E)$ is a directed graph with vertex set $V = \{1, 2, \ldots, n\}$ and edge set E, where each $(i, j) \in E$ is a directed edge from i to j, denoted $\overrightarrow{ij}$. The *transitive closure* of G is the graph $\tilde{G} = (V, \tilde{E})$ where $\overrightarrow{ij} \in \tilde{E}$ iff there is a directed path from i to j in G.

2.2 Feasibility and Hardness

For the purposes of this thesis, "feasibility" means that there exists a probabilistic polynomial time algorithm. When we say something is "hard" or "infeasible," it means that no probabilistic polynomial time exists, perhaps assuming some conjecture (such as the hardness of factoring or the RSA problem).

2.3 One-Way Functions

Cryptography investigates phenomena which are computationally easy for some people, and computationally difficult for others. One of the most fundamental primitives is the *one-way function.* A one-way function f has the property that it is easy to evaluate $f(x)$ on any x in the domain, but given $f(x)$ it is infeasible to compute $f^{-1}(x)$.

2.4 Cryptographic Hash Functions

A cryptographic hash function is a polynomial-time computable function $H : \{0,1\}^* \to \{0,1\}^k$ that satisfies the following additional requirements.

1. Preimage Resistance: Given y it is hard to compute x such that $h(x) = y$.
2. Second Preimage Resistance: Given a value x it is hard to find an $x' \neq x$ such that $h(x) = h(x')$.
3. Collision Resistance: It is hard to find any pair $x \neq x'$ such that $h(x) = h(x')$.

Here by "hard" we mean that for all probabilistic polynomial time adversaries A, the advantage of such an adversary is smaller than any inverse polynomial in k. Practical examples of cryptographic hash functions include SHA-1, MD5, and the RIPE-MD series. Hash functions are essential components of signature schemes, because they allow for signing a short "representative" of a message instead of signing the entire message itself.

2.5 The Random Oracle Model

The exact structure of hash functions tends to be poorly understood. Rather than attempt to analyze a cryptosystem with respect to any single hash function, cryptographers often assume that the output of the hash function is *random*. The idea is that the hash function is somehow "ideal." We then prove security with respect to the ideal hash function and attempt to instantiate it with as close a hash function to ideal as possible.

We model this formally by saying that all parties have access to a *random oracle* and cryptography that makes this assumption is said to be in the *random oracle model*. Here oracle is in the sense of "oracle Turing Machine"; for formal definitions of oracle machines, we refer to a textbook on complexity theory such as Papadimitriou[44]. By random oracle, we mean an oracle O that when first queried on the value x flips coins to determine the value of $O(x)$, and returns that same value on all subsequent queries.

The random oracle model is controversial in cryptography. Canetti, Goldreich, and Halevi showed that there exist signature schemes that are provably secure in the random oracle model, yet can have *no* secure implementation, regardless of the choice of hash function[12]. Recently, Goldwasser and Tauman showed that there were identification schemes for which the Fiat-Shamir heuristic fails for similar reasons[26]. Many of the proofs we give or cite in this thesis use the random oracle model; removing the random oracles while maintaining efficient constructions is a widespread open problem in cryptography. Recently, Zhu, Feng, and Deng announced a transitive signature scheme with a proof of security that is *not* in the Random Oracle Model[57].

2.6 Standard Digital Signature Schemes

A standard digital signature scheme consists of three algorithms.

Key Generation. The key generation algorithm KG takes as input a security parameter k in unary, denoted 1^k. It outputs the pair (PK, SK) where PK is the public key for the scheme and SK is the secret key.

Signing. The signing algorithm $Sign$ takes as input SK and a message M. It outputs a signature $Sig(M)$.

Verification. The verification algorithm takes as input the public key PK, a message M, and a signature $Sig(M)$. It outputs 1 if $Sig(M)$ could have been created by $Sign$ using the corresponding SK on M and 0 otherwise.

We will use standard digital signature schemes as primitives when building some homomorphic signature schemes. We will assume all such schemes are secure against *adaptive chosen message attack*, which we define in Section 3.4.1.

Chapter 3

Definitions

> "The right definition is the one that
> lets you prove the right theorems!"
> -Silvio Micali

3.1 A Short Informal Example

We start with an example of a signature scheme that has special algebraic properties and shows some of the subtleties involved in homomorphic signature schemes. After discussing the example, we will be ready to examine the requirements for a formal definition.

Our example scheme is a "transitive signature scheme," as introduced by Micali and Rivest. The scheme is based on RSA and illustrated in Figure 3.1. The nodes have public "node labels" A, B, and C that are each group elements in Z/nZ^*. The signature on an edge AB is the RSA signature on the element AB^{-1}. Anyone can verify such a signature by first inverting B, multiplying by A, and then checking the RSA signature.

In such a scheme, signatures on edges of a graph can be composed to form edges in their transitive closure. For example, in the figure, the edge AB and the edge BC are composed to form the edge on AC. Composition itself requires only one multiplication mod n.

If we did not have RSA or some other multiplicative signature scheme, we could acheive a "poor man's transitive scheme" simply by giving both a signature on AB and a signature on BC and stating formally that this should be interpreted as the edge AC. The RSA-based scheme improves over this approach in at least two ways. First, in the RSA-based scheme, it is impossible to distinguish between an "original" signature created by the master signer and a signature formed through composition. Each edge has one and only one signature, regardless of how it was obtained. Second, each edge signature is a single group element; the size of the signature is constant with respect to the number of aggregations.

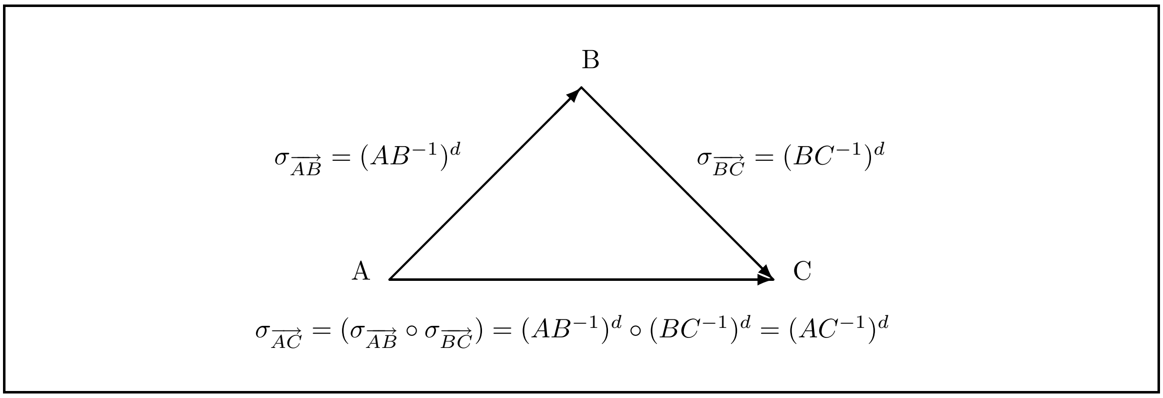

Figure 3.1: Illustration of RSA-based transitive signature scheme, where A, B, C are public node labels. All values taken $\mod N$.

3.2 Desiderata for a Definition

From the example given above, we can develop some desiderata for a definition of homomorphic signature schemes.

- The definition should cover all known schemes. In particular, it should capture the power of the "master signer" to create signatures and of others to combine them without the private key.
- The definition should rule out "trivial" schemes, such as simply giving all signatures.
- The definition should include a notion of "privacy," such that original signatures cannot be distinguished (or distinguished only under special conditions) from signatures formed via combination.

We do not have a definition that covers all these requirements. Instead, we give a general definition, then specialize the definition for the case of transitive signature schemes. Finally we discuss an alternative definition of security for homomorphic schemes, that of "security against random forgeries." We motivate the alternative definition by showing a scheme that is secure against our first notion of security, but in some sense "trivial."

3.3 A Definition of Homomorphic Signature Schemes

We define the notion of a homomorphic signature scheme as follows. A specification of a signature scheme includes a message space $\mathcal{M}$, a set of private keys $\mathcal{K}$, a set of public keys $\mathcal{K}'$, a (possibly randomized) signature algorithm $\mathsf{Sig} : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, and a verification algorithm $\mathsf{Vrfy} : \mathcal{K}' \times \mathcal{M} \times \mathcal{Y} \to \{\mathsf{ok}, \mathsf{bad}\}$

so that $\mathsf{Vrfy}(k', x, \mathsf{Sig}(k, x)) = \mathsf{ok}$ for all $x \in \mathcal{M}$ when (k, k') is a matching private key and public key. As a notational matter, for conciseness we often omit the private key, writing $\mathsf{Sig}(x)$ instead of $\mathsf{Sig}(k, x)$ and $\mathsf{Vrfy}(x, s)$ instead of $\mathsf{Vrfy}(k', x, s)$ when this abbreviation will not cause confusion. Also, for a binary operation $\odot : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ and a set $S \subseteq \mathcal{M}$, we let $\mathrm{span}_\odot(S)$ denote the least set T with $S \subseteq T$ and $x \odot y \in T$ for all $x, y \in T$.

Definition 3.3.1 *Fix a signature scheme* $\mathsf{Sig} : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, $\mathsf{Vrfy} : \mathcal{K}' \times \mathcal{M} \times \mathcal{Y} \to \{\mathsf{bad}, \mathsf{ok}\}$ *and a binary operation* $\odot : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$. *We say that* Sig *is homomorphic with respect to* $\odot$ *if it comes with an efficient family of binary operations* $\otimes_{k'} : \mathcal{Y} \times \mathcal{Y} \to \mathcal{Y}$ *so that* $y \otimes_{k'} y' = \mathsf{Sig}(x \odot x')$ *for all* x, x', y, y' *satisfying* $\mathsf{Vrfy}(x, y) = \mathsf{Vrfy}(x', y') = \mathsf{ok}$.

As an example, if $(G, \times_G)$ and $(R, \times_R)$ are two groups and we have a signature scheme $\mathsf{Sig} : \mathcal{K} \times G \to R$ that is also a group homomorphism from G to R for each $k \in \mathcal{K}$, then this will qualify as a signature scheme that is homomorphic with respect to $\times_G$, since we may take $y \otimes_{k'} y' = y \times_R y'$.

This definition requires that signatures derived via $\otimes_{k'}$ be indistinguishable from signatures generated by the private key holder. This precludes trivial schemes that, for example, allow the ordered pair (y, y') to serve as a signature on $x \otimes x'$. Although the definition above is for deterministic signature schemes, extending it to probabilistic schemes is straightforward; one can simply require that the distribution of $y \otimes_{k'} y'$ be indistinguishable from that of $\mathsf{Sig}(x \odot x')$.

3.4 Security of Homomorphic Signature Schemes

3.4.1 Security Against Adaptive Chosen Message Attack

The most popular definition of security for ordinary, non-homomorphic, signature schemes is *security against adaptive chosen message attack*. Introduced by Goldwasser, Micali, and Rivest, the definition tries to capture the strongest possible adversary. Informally, a scheme is secure if the adversary cannot meet the following challenge: first it is given adaptive access to an oracle that signs as many messages as it likes, then it is challenged to produce a signature on any new message it likes so long as that message was not previously submitted to the signing oracle.

The formal definition is as follows: We say that a signature scheme Sig is (t, q, ϵ)-secure against existential forgeries if every adversary A making at most q chosen-message queries and running in time at most t has advantage $\mathrm{Adv}\, A \leq \epsilon$. The advantage of an adversary A is defined as the probability that, after queries on the messages $x_1, \ldots, x_q$, A outputs a valid signature $\langle x', y' \rangle$ on a message not already queried.

3.4.2 Security Against Adaptive Chosen Message Attack for Homomorphic Schemes

When first confronted with the problem of defining a homomorphic signature scheme, the most natural impulse is to "patch" security against adaptive chosen message attack to take the homomorphic property into account. Informally, the new definition might run as follows: Suppose the adversary has obtained signatures on queries $x_1, \ldots, x_q$. Such an adversary can deduce signatures on $x_i \odot x_j$, $(x_i \odot x_j) \odot x_k$, and so on; in other words, no message in $\mathrm{span}_\odot(x_1, \ldots, x_q)$ seems to be safe. Therefore, we will require that no adversary be able to deduce a valid signature on anything outside $\mathrm{span}_\odot(x_1, \ldots, x_q)$. We can write this formally as

Definition 3.4.1 *We say that a homomorphic signature scheme* Sig *is* (t, q, ϵ)*-secure against existential forgeries with respect to* $\odot$ *if every adversary* A *making at most* q *chosen-message queries and running in time at most* t *has advantage* $Adv\,A \leq \epsilon$. *The advantage of an adversary* A *is defined as the probability that, after queries on the messages* $x_1, \ldots, x_q$, A *outputs a valid signature* $\langle x', y' \rangle$ *on some message* $x' \notin \mathrm{span}_\odot(x_1, \ldots, x_q)$. *In other words,* $Adv\,A = \Pr[A^{\mathsf{Sig}(k,\cdot)} = \langle x', y' \rangle \wedge \mathsf{Vrfy}(x', y') = \mathsf{ok} \wedge x' \notin \mathrm{span}_\odot(x_1, \ldots, x_q)]$.

3.4.3 Decomposition

Consider the decomposition problem: given $x \in \mathrm{span}_\odot(x_1, \ldots, x_q)$, find an explicit decomposition of x in terms of the x_i's and the $\odot$ operation.

If the decomposition problem is hard (as happens, for instance, in $((\mathbb{Z}/nZ)^*, \times)$), then our definition of security against existential forgeries might be too restrictive. In the cases we consider this issue does not come up, because the decomposition problem is always easy.

Decomposition is also important for the master signer. Suppose the master signer wishes to prevent a message M_t from being signed, and has already given out signatures on messages $M_0, \ldots, M_{k-1}$. Then when asked to sign a new message M_k, the master signer must be able to tell whether a decomposition of a signature on M_t exists into signatures on $M_0, \ldots, M_k$

3.5 Homomorphic Signatures and State

In addition to the decomposition problem, if the master signer wishes to prevent a signature on a target message M_t, it needs to keep a record of all messages previously signed. This is a large amount of state. We do not address this problem directly, but we suggest some directions for future research to reduce this requirement. First, the master signer could change secret keys periodically; if only signatures from the same secret key can be combined, this allows flushing the list of previous signatures. Second, the master signer could mark each signature with a time-stamp or use a key-evolving scheme to keep the same public key while changing which signatures can be composed. Finally, the signer could keep a "sketch" of previous signatures that allows deciding the existence of a decomposition with some probability of error. Recently, Zhu, Feng, and Deng proposed a special construction of node labels in a transitive signature scheme to address this problem[57].

3.6 Defining Transitive Signature Schemes

DIRECTED TRANSITIVE SIGNATURE SCHEMES.

We now define *directed* transitive signature schemes, in preparation for Chapter 5. The need for a separate definition of transitive signatures was pointed out by Bellare and Neven.

Our definition of transitive signature schemes extends that of Bellare and Neven in four ways. First, we have *directed* edges, which affect the edge signing, verification, and composition algorithms. Second, our definition introduces an explicit function NCert for the creation of per-node certification information; this information is implicitly encoded into the choice of graph index, an inherently *public* label, in the Bellare-Neven definition. The algorithm NCert allows us to talk more easily about how a transitive signature scheme deals with node state. Third, we handle both "static" and "dynamic" transitive signature schemes, where a scheme may or may not have to include nodes added after the master keys have been generated. Finally, we define "transitive signature schemes with pre-processing" as schemes where key generation may depend on the graph G. This notion may be of independent interest for efficiency and simplicity issues in settings where changes are scarce.

Let $\mathbb{N}$ be the set of positive integers. A *directed transitive signature scheme* DTS=(KG, NCert, ESign, Vf, Comp) is specified by five polynomial-time algorithms as follows:

- The randomized *key generation* algorithm KG can *either* take as input solely 1^k, where $k \in \mathbb{N}$ is the security parameter, or take 1^k and a graph G, and return a pair (PK,SK) consisting of a *master* public key and matching secret key. In the first case, we say the DTS scheme is *dynamic*, since new nodes can be added at any time (because the master keys do not depend on the graph being signed). In the latter case, we say the DTS scheme is *static*, since the master keys are dependent on the graph (if the graph changes, KG may have to be run anew). For the static case, we say that a DTS requires $\epsilon(k)$-*heavy per-key pre-processing* if the running time of KG on input 1^k and graph G is bounded by $\epsilon(k)$. In the case that $\epsilon(k)$ is polynomial, we say that the DTS requires *insignificant* per-key preprocessing.

- The *node certification* algorithm NCert, which could be stateful or randomized (or both), takes as input the master secret key SK and a node

$i \in \mathbb{N}$, and returns a pair (pk_i, sk_i) consisting of a public key and a matching secret key for node i. The secret key sk_i is held by the master signer exclusively. If stateful, it updates its state upon each invocation.

- The *edge signing* algorithm ESign, which could be stateful or randomized (or both), has read only access to the state variables of NCert. It takes as input the master secret key SK and two node indices i, j, and returns a value called an *original signature* of edge $\overrightarrow{ij}$ relative to the secret information of the master and nodes i, j, or $\perp$ for failure. If stateful, it updates its state upon each invocation.
- The deterministic *verification* algorithm Vf, given PK, two node public keys pk_i, pk_j, and a candidate signature $\sigma_{\overrightarrow{ij}}$, returns either 1 or 0. It returns one iff $\sigma_{\overrightarrow{ij}}$ is a *valid* signature of edge $\overrightarrow{ij}$ relative to the public information of the master and nodes i, j.
- The deterministic *composition* algorithm Comp takes as input PK, three node public keys pk_i, pk_j, pk_k, and values σ_1, σ_2, and returns either a value $\sigma_{\overrightarrow{ik}}$, called a *composed signature* of edge $\overrightarrow{ik}$ or a symbol $\perp$ to indicate failure.

For a directed transitive signature scheme DTS to be useful for the applications in Section 1, it must also be *correct* and *secure*.

Correctness of DTS Schemes. It is a natural requirement that an original signature $\sigma_{\overrightarrow{ij}}$ is also a valid signature with respect to PK, pk_i, and pk_j. For the correctness of a composed signature, it must always produce a valid signature $\sigma_{\overrightarrow{ik}}$, when given as input two original or composed signatures $\sigma_{\overrightarrow{ij}}, \sigma_{\overrightarrow{jk}}$ and the corresponding public keys.

As in Bellare and Neven[5], we agree it does not always follow that just because two edges are valid (i.e. they pass the Vf algorithm), they should be composable. Consider the case where the public key for node j is a square A and its secret key is a specific square root of $A \mod N$. Now, it might be possible that an edge using another square root of A might appear valid and yet not compose well with original signatures. We overcome this problem by using a definition below that requires all signatures passed to Comp be either original or composed themselves.

Furthermore, to maintain privacy, we require that a composed signature on edge $\overrightarrow{ik}$ be *indistinguishable* from an original signature on edge $\overrightarrow{ik}$. Thus, a composed edge gives no information about its intermediate node n, since the signature is indistinguishable from that of the ESign algorithm, which did not receive pk_n as input.

In Figure 3.2, we give an experiment for any (deterministic, halting, computationally unbounded) algorithm A and any $k \in \mathbb{N}$, which gives A oracle access to

$$\mathsf{NCert}(\mathsf{SK},\cdot),\ \mathsf{ESign}(\mathsf{SK},\cdot,\cdot) \text{ and } \mathsf{Comp}(\mathsf{PK},\cdot,\cdot,\cdot,\cdot,\cdot),$$

(1) (PK,SK) $\leftarrow_R$ KG(1^k)
(2) $N \leftarrow \emptyset$; $S \leftarrow \emptyset$; Legit $\leftarrow$ true; NotOK $\leftarrow$ false
(3) Run A with its oracles until it halts, replying to its oracle queries as follows:
(4) If A queries NCert on i then
(5) If $i \in N$ then Legit $\leftarrow$ false
(6) Else let $(pk_i, sk_i) \leftarrow$ NCert(PK,i), $N \leftarrow N \cup \{i\}$
(7) If A queries ESign on i, j then
(8) If $(i = j) \vee (i \notin N) \vee (j \notin N)$ then Legit $\leftarrow$ false
(9) Else
(10) Let σ be the output of the ESign oracle and $S \leftarrow S \cup \{i, j, \sigma\}$
(11) If Vf(PK,pk_i,pk_j,σ)=0 then NotOK $\leftarrow$ true
(12) If A queries Comp on $pk_i, pk_j, pk_k, \sigma_1, \sigma_2$ then
(13) If $(i, j, \sigma_1) \notin S \vee (j, k, \sigma_2) \notin S \vee i, j, k$ are not all distinct) then Legit $\leftarrow$ false
(14) Else
(15) Let σ be the output of the Comp oracle, $S \leftarrow S \cup \{i, k, \sigma\}$ and $\tau \leftarrow$ ESign(SK,i, k)
(16) If $((\sigma \neq \tau) \vee$ Vf(PK,pk_i,pk_k,σ)=0) then NotOK $\leftarrow$ true
(17) When A halts, output (Legit $\wedge$ NotOK) and halt

Figure 3.2: An experiment to define the correctness of a dynamic directed transitive signature scheme DTS=(KG, NCert, ESign, Vf, Comp).

where PK,SK are the result of running KG on input 1^k. In this experiment, the NCert and ESign oracles maintain state, and update their states each time they are invoked. If randomized, new coins are tossed at each invocation.

Definition 3.6.1 *We say that the dynamic directed transitive signature scheme* DTS *is* correct *if for every (deterministic, halting, computationally unbounded) algorithm A and every $k \in \mathbb{N}$, the output of the experiment of Figure 3.2 is* true *with probability zero.*

Queries to sign original edges, as well as compose either original or composed edges, are allowed by the experiment with additional nodes being added at any time. During this process, the boolean variable Legit is set to false if an illegitimate query is ever made, while the variable NotOK is set to true if an invalid signature is ever produced by either the ESign or Comp oracles on good inputs. To win, A must stay legitimate (Legit=true), but violate correctness (NotOK=false). The definition requires that this happen with probability zero.

For the static case, we simply require that all calls to the oracle NCert to create a graph G be completed before the master keys are generated. Therefore, we run a new experiment with A and k, changing Figure 3.2 to move lines 2, 4, 5 and 6 to appear sequentially above the current line 1, which itself is replaced by "(PK,SK) $\leftarrow_R$ KG($1^k, G$)". The rest of the experiment is consistent with the dynamic case.

Definition 3.6.2 *We say that the static directed transitive signature scheme* DTS *is* correct *if for every (deterministic, halting, computationally unbounded)* *algorithm A and every $k \in \mathbb{N}$, the output of the experiment for Figure 3.2, with static modifications, is* true *with probability zero.*

SECURITY OF DTS SCHEMES. Forgeries are valid signatures not within the transitive closure of G. Formally, we define an experiment for directed transitive signature scheme DTS=(KG, NCert, ESign, Vf, Comp), adversary F and security parameter $k \in N$, denoted

$$\mathbf{Exp}^{dtu-cma}_{DTS,F}(k),$$

that returns 1 if and only if F is successful in its attack on the scheme. The experiment begins by running KG on input 1^k (and also G for static schemes) to get the master keys (PK,SK). It then runs F with input PKand oracle access to the function ESign(SK,$\cdot,\cdot$). The adversary for dynamic schemes calls an oracle NCert to create nodes, while in static schemes the public keys of all nodes are also given as input. Eventually, F will output $i', j' \in \mathbb{N}$ and a value σ'. Let E be the set of all edges $\overrightarrow{ij}$ such that F made oracle query i, j. Let V be the set of all nodes certified by the master signer. We say that F *wins* if σ' is a valid signature of $\overrightarrow{i'j'}$ relative to PK, $pk_{i'}$, and $pk_{j'}$, and yet edge $\overrightarrow{i'j'}$ is not in the transitive closure of $G = (V, E)$. The experiment returns 1 if F wins; 0 otherwise. The *advantage* of F in its attack on DTS is the function $\mathbf{Adv}^{dtu-cma}_{DTS,F}(k)$ defined for $k \in \mathbb{N}$ by

$$\mathbf{Adv}^{dtu-cma}_{DTS,F}(k) = Pr[\mathbf{Exp}^{dtu-cma}_{DTS,F}(k) = 1].$$

We say that DTS is *directed transitively unforgeable under adaptive chosen-message* if the function $\mathbf{Adv}^{dtu-cma}_{DTS,F}(k)$ is negligible for any adversary F whose running time is polynomial in the security parameter k.

UNDIRECTED TRANSITIVE SIGNATURE SCHEMES. We can think of an undirected graph as a directed graph in which edges point in both directions between any pair of nodes for which there is an edge. In this way we can reduce the question of defining undirected transitive signatures to that of defining directed transitive signatures.

3.7 Trivial Signature Schemes and Security against Random Forgeries

We now point out an issue with the definition of security given above for general homomorphic signature schemes. Namely, the definition allows for schemes which are secure, but "trivial." We give an example of such a secure but trivial scheme, and then suggest an amended definition of security. We end by discussing flaws in the amended definition.

3.7.1 Triviality of "Additive" Signature Schemes

Consider an "additive" signature scheme, where we can perform addition and subtraction on messages. For example, we might consider signing messages by picking a generator g of some large subgroup of $(\mathbb{Z}/n\mathbb{Z})^*$, for n an RSA modulus, and letting the signature on message M be $(g^M)^d \mod n$. In this scheme, computing signatures on the addition and subtraction of messages is simply a matter of multiplying and dividing signatures.

It turns out that this scheme has serious problems. First we show that if the order m of the generator g is publicly known, this construction is "trivial," in the sense that while it may satisfy security against adaptive chosen message attack as defined above, an adversary can easily sign any message it wants[1]. Then we show that even if the order of the subgroup is *not* known, the scheme remains trivial; an adversary given two messages M_1 and M_2 such that $\gcd(M_1, M_2) = 1$ can obtain a signature of the message 1 and so sign anything it wishes. These weaknesses generalize to almost any additive signature scheme.

We make use of two simple observations.

Observation 1 *The discrete log problem in $\mathbb{Z}/m\mathbb{Z}$ is easy, when m is known.*

Proof. We can easily find $k \in \mathbb{Z}$ satisfying $y = kx \pmod m$ (where $x, y \in \mathbb{Z}/m\mathbb{Z}$ are given) simply by using modular division: take $k = y^{-1}x \pmod m$, where $y^{-1} \pmod m$ may be computed using Euclid's algorithm. □

Observation 2 *If Sig_+ represents the additive signature scheme described above, then it is easy to compute $\mathsf{Sig}_+(kx)$ from $\mathsf{Sig}_+(x)$ without the help of the signer, for any given $k \in \mathbb{Z}$.*

Proof. The idea is to use a "double-and-add" algorithm (analogous to the classical "square-and-multiply" algorithm for fast exponentiation). We assume without loss of generality that $k \geq 0$, for otherwise we may replace k by $-k$ and x by $x^{-1} \pmod m$. Next note that $\mathsf{Sig}_+(2x) = \mathsf{Sig}_+(x + x) = \mathsf{Sig}_+(x) \times \mathsf{Sig}_+(x)$ by the homomorphic property, and thus $\mathsf{Sig}_+(2^i x)$ may be computed with i multiplications for any i. Express k in binary notation as $k = \sum_i k_i 2^i$ (for $k_i \in \{0, 1\}$). Then we may write $\mathsf{Sig}_+(kx) = \prod_i \mathsf{Sig}_+(2^i x)$, where the product is taken over the values of i with $k_i = 1$. In the end, computing $\mathsf{Sig}_+(kx)$ requires $O(\lg k)$ multiplications, and so can be considered easy. □

Now we outline the attack. Suppose we have observed a signature $\mathsf{Sig}_+(x)$ on some message x (where x need not be under our control), and we want to obtain a signature $\mathsf{Sig}_+(y)$ on a message y chosen randomly or, even worse, of our choosing. We find k so that $y = kx \pmod m$ by solving an instance of the discrete log problem in $\mathbb{Z}/m\mathbb{Z}$ (see Observation 1), and then we compute $\mathsf{Sig}_+(y) = \mathsf{Sig}_+(kx)$ from $\mathsf{Sig}_+(x)$ using Observation 2. This allows us to forge a signature on any messages of our choosing, given only a single signature on some known (but otherwise arbitrary) message.

Variations. The above discussion assumed that the order of the additive group is known, so we might try keeping the order secret. Alas, this does

[1]We note that learning the order of g also divulges a factor of $\phi(n)$, which may aid in factoring n. The weakness we discuss is independent of this consideration.

not produce a scheme secure against random forgeries, either. The following observations break the scheme.

Observation 3 *Suppose we obtain signatures* $\mathsf{Sig}_+(x_1), \ldots, \mathsf{Sig}_+(x_n)$ *on the known messages* $x_1, \ldots, x_n$ *under some additive signature scheme on* $\mathbb{Z}$. *Let* $\mathrm{span}(x_1, \ldots, x_n) = \{\sum_i a_i x_i : a_i \in \mathbb{Z}\}$ *denote the linear span of these messages, i.e., the linear subspace of* $\mathbb{Z}$ *spanned by* $x_1, \ldots, x_n$. *Then we can obtain a signature on every message* $y \in \mathrm{span}(x_1, \ldots, x_n)$.

Proof. If $y = \sum_i a_i x_i$, then $\mathsf{Sig}_+(y) = \prod_i \mathsf{Sig}(a_i x_i)$, and by Observation 2 $\mathsf{Sig}(a_i x_i)$ can be computed easily. □

Observation 4 *In any additive signature scheme on* $\mathbb{Z}$, *if one can obtain* $\mathsf{Sig}(1)$, *a signature on the value 1, then the scheme is broken.*

Proof. Follows immediately from Observation 3, since the linear span of 1 is all of $\mathbb{Z}$. □

Suppose now that we have observed signatures $\mathsf{Sig}(x_1), \mathsf{Sig}(x_2)$ on two known but otherwise arbitrary messages x_1, x_2. We compute $d = \gcd(x_1, x_2)$. With good probability, $d = 1$, and in this case, we have broken the scheme (in the sense of random forgeries), due to Observations 3 and 4, since $\gcd(x_1, x_2) \in \mathrm{span}(x_1, x_2)$.

To see why any such scheme must be insecure against random forgeries, suppose $\mathsf{Sig}_+ : \mathbb{Z} \to G$ is a group-homomorphic signature scheme. Then Sig_+ is entirely determined by $\mathsf{Sig}_+(1)$. Thus, if we can recover a few messages $m_1, \ldots, m_k$ such that $\gcd(m_1, \ldots, m_k) = 1$, then since there exist $a_1, \ldots, a_k \in \mathbb{Z}$ such that $\sum_i a_i m_i = 1$, we can compute $\mathsf{Sig}_+(1) = \sum_i a_i \mathsf{Sig}_+(m_i)$. Given this information, we can compute $\mathsf{Sig}_+(m) = m\mathsf{Sig}_+(1)$ for any m. This attack easily extends to group homomorphic signatures $\mathsf{Sig}_+ : \mathbb{Z}/n\mathbb{Z} \to G$ by lifting to $\mathbb{Z}$, applying the Euclidean algorithm, and projecting back down to $\mathbb{Z}/n\mathbb{Z}$. Since any small set of messages will likely have gcd 1, the adversary can sign any message after seeing only a few signatures.

These weaknesses are general. They apply to almost any instance of the construction $\mathsf{Sig}_+ = \mathsf{Sig}_\times \circ \varphi$. As another important example, every additive signature scheme that forms a group homomorphism is insecure against random forgeries.

It is perhaps counterintuitive that multiplicative signature schemes are plentiful while fully-additive schemes do not exist. The explanation seems to be the fact that $\mathbb{Z}/m\mathbb{Z}$ has a Euclidean algorithm, but $(\mathbb{Z}/n\mathbb{Z})^*$ does not. In other words, it is not that the span is larger in $\mathbb{Z}/m\mathbb{Z}$, but that the decomposition problem is easy in $\mathbb{Z}/m\mathbb{Z}$ but hard in $(\mathbb{Z}/n\mathbb{Z})^*$.

One might imagine that the problem is the need to be homormorphic with respect to both addition as well as negation, and thus one idea might be to look for a scheme that respects only the addition operator but not the negation. In other words, given $\mathsf{Sig}_+(x), \mathsf{Sig}_+(y)$, we still want to be able to compute $\mathsf{Sig}_+(x + y)$, but it should be hard to compute $\mathsf{Sig}_+(-x)$ from $\mathsf{Sig}_+(x)$. Note that the problems with additive signatures arise because one can find $a, a' \in \mathbb{Z}$ so that $ax + a'x' = 1$ and then compute $\mathsf{Sig}_+(1) = \mathsf{Sig}_+(ax) \times \mathsf{Sig}_+(a'x')$; yet one of a, a' will necessarily be negative. If we can somehow ensure that computing $\mathsf{Sig}_+(-x)$ from $\mathsf{Sig}_+(x)$ is hard, then the Euclidean algorithm will no longer apply, and the above attacks will fail. This suggests that we may want to look for an additive signature scheme without inverses (a semigroup-homomorphism), as such a scheme would resist the attacks described so far. In Chapter 6 we will see evidence that such a scheme is unlikely to be constructible from one-way functions alone.

Further challenges. Even an additive signature scheme without inverses still has some properties that might not be expected. In particular, there is the problem that signatures can always be forged on all large enough messages, given signatures on two messages m, m'. This is because the equation

$$am + a'm' = x \qquad (\text{in } \mathbb{Z})$$

typically has solutions with $a, a' \geq 0$ when $x \geq \mathrm{lcm}(m, m')$, and in this case a signature on x can be obtained from signatures on m, m'.

More generally, if we have signatures $\mathsf{Sig}(m_1), \ldots, \mathsf{Sig}(m_k)$, then we can forge a signature $\mathsf{Sig}(x)$ whenever we can write x in the form $x = a_1 m_1 + \cdots + a_k m_k$ for some known $a_1, \ldots, a_k \in \mathbb{Z}_{\geq 0}$. This is a subset sum problem, and the issue is that if the m_i are small enough, the subset sum problem is easy to solve. So our only hope is to choose additive signatures without inverses, only sign messages large enough that subset sum is hard (require $m_i \geq \ell$ for some lower bound ℓ), and refuse to accept unusually large messages (enforce $m_i \ll \ell^2$) in the verification algorithm.

Additive schemes in higher dimensions. More generally, we could look for an additive scheme $\mathsf{Sig}_+ : (\mathbb{Z}/m\mathbb{Z})^d \to R$ in dimension $d > 1$. Unfortunately, this does not seem to offer much opportunity to design a secure signature scheme, either.

Although increasing the dimension does make more work for the attacker, a slight extension of the previous remarks still applies.

Observation 5 *In any additive signature scheme on the lattice $L = (\mathbb{Z}/m\mathbb{Z})^d$, if one can obtain signatures $\mathsf{Sig}(x_1), \ldots, \mathsf{Sig}(x_d)$, where $x_1, \ldots, x_d$ are a basis for L, then one can generate signatures on randomly chosen messages.*

Knowing the signatures on a basis is useless if computing the representation of a given message in that basis is hard. If we could compute the signatures of the standard basis elements given the signatures on the elements of another basis, then generating signatures on randomly chosen messages would be easy.

If we allow a chosen message attack, the attacker can simply make d queries for each element in the standard basis. A careful signer will detect this attack, but the attacker can disguise the basis by treating it as a matrix and performing elementary invertible row operations (swapping rows, adding two rows, and multiplying rows by units in $(\mathbb{Z}/m\mathbb{Z})$). The signer can also detect this trick by, for example, computing the determinant of the d queries and aborting if it is a unit in $(\mathbb{Z}/m\mathbb{Z})$, but by the time the signer has signed enough messages to compute their determinant, the attacker can produce signatures on almost all the messages in $(\mathbb{Z}/m\mathbb{Z})^d$. It's not hard to come up with more sensitive ways for the signer to detect attacks like this, but they seriously compromise the utility of the signature scheme, so this should raise doubts about its security.

Note that the elements of the standard basis are the shortest vectors in the lattice $\mathbb{Z}^d$, so lattice reduction techniques may be used to discover representations of the standard basis elements in terms of messages with known signatures. The theoretical bounds on the length of the shortest vector returned by LLL are not very tight, but in practice it can find a representation of the standard basis of $(\mathbb{Z}/m\mathbb{Z})^d$ with only $(1+\epsilon)d$ input vectors. Thus, we only need to collect $(1+\epsilon)d$ signed messages before we can commit random forgeries with ease.

3.7.2 Security against Random Forgeries

The previous section gave an example of a scheme that was secure, but "trivial." The existence of such a scheme indicates that the definition does not capture our intuitive understanding of what a secure homomorphic scheme should be. The problem with our definition of security was that it failed to rule out schemes where the span of a few messages is large. We now present a definition that addresses this problem by saying a scheme is secure if the adversary cannot forge a signature on a *random* message.

Definition 3.7.1 *The signature scheme* $\mathsf{Sig} : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ *is said to be* (t, q, ϵ)*-secure against random forgeries if every adversary* A *making at most* q *chosen-message queries and running in time at most* t *has advantage* $Adv\, A \leq \epsilon$*. The advantage of an adversary is the probability that it can output a valid signature on a new message* x' *chosen by the referee uniformly at random from* $\mathcal{M}$ *and independently of all the adversary's chosen-message queries. In other words,* $Adv\, A = \Pr[\mathsf{Vrfy}(x', A^{\mathsf{Sig}(k,\cdot),c_{x'}}) = \mathsf{ok}]$ *where* $c_{x'}$ *is the constant function that always returns* x' *and where the adversary is not allowed to make any further queries to its signing oracle after looking at the value* x'*.*

We can see that the additive signature scheme of the previous section is not secure against random forgeries; given just two oracle calls, the adversary can obtain a signature of 1 and then sign any random challenge message. At the same time, one might reasonably conjecture that textbook RSA signatures are secure against random forgeries.

Unfortunately, this definition has a flaw. To draw a random message, we should know the "right" distribution over the message space. In some cases, we may not want to assume the uniform distribution or may not know the right distribution. Coming up with a definition that both rules out trivial schemes and uses no assumptions about the message space is open.

Chapter 4

Example Homomorphic Signature Schemes

Now that we have definitions for homomorphic signature schemes, we can see some examples. We discuss the following schemes:

- Transitive Signature Schemes: We describe the first transitive signature scheme, due to Rivest and Micali. We discuss its requirement for explicit node certification and then show a scheme due to Bellare and Neven that removes this requirement.
- Redactable Signature Schemes: We carefully define "redactable" signatures, or signatures which allow for the deletion of parts of a message while ensuring the message still verifies. We also give a proof of security for a scheme for redaction that combines hash trees and pseudorandom functions.
- Set-Homomorphic Signature Schemes: We show how RSA accumulators can be used to build a signature scheme on sets that is homomorphic with respect to union and subset. Finding a proof of security for this construction remains open. We also introduce the open problem of finding a signature scheme homomorphic with respect to union alone. In Chapter 6 we will return to this open problem.

4.1 Transitive Signature Schemes

4.1.1 Micali-Rivest

The first transitive signature scheme was given by Micali and Rivest [38]. The scheme uses homomorphic commitments of the form $g^x h^y$ in a cyclic group.

Key Generation. The KG algorithm takes 1^k as input. First, it calls SKG on 1^k to generate a key pair (spk, ssk) for the standard signature scheme SDS.

It then generates large primes p and q such that q divides $p-1$. This ensures the existence of a subgroup G_q of order q in Z_p^*. The algorithm then picks generators $g, h \in G_q$, such that the base-g logarithm of h modulo p is infeasible for others to compute.

Node Certification. The NCert algorithm maintains state $N \subseteq V$, the nodes signed so far and takes as input SK=(k, S, t, ssk) and a node index i from the master signer.

If $i \notin N$, then the algorithm sets $N \leftarrow N \cup \{i\}$, then picks two values x_i and y_i uniformly and independently from G_q. Finally the algorithm returns the node label $v_i = g^{x_u} h^{y_i} \mod p$. If $i \in N$, the algorithm simply returns v_i.

Edge Signing. The ESign algorithm takes two node indices i, j as input. If $(i \neq j) \wedge (i \in N) \wedge (j \in N)$, then it returns $\sigma_{\overrightarrow{ij}} = (i, j, \alpha_{ij}, \beta_{ij}$, where

$$\alpha_{ij} = x_i - x_j \mod q$$
$$\beta_{ij} = y_i - y_j \mod q$$

Otherwise the algorithm declares failure and returns $\perp$.

Verification. The Vf algorithm takes the public labels of nodes i, j and a value σ, and returns 1 if and only if σ is a valid signature from i to j. To check if a signature $(i, j, \alpha_{ij}, \beta_{ij})$ is valid, the algorithm checks

$$v_i = v_j g^{\alpha_{ij}} h^{\beta_{ij}} \mod q$$

Composition. The Comp algorithm takes the public labels of nodes i, j, k and two values σ_1 and σ_2, and tries to output $\sigma_{\overrightarrow{ik}}$. Given $\sigma_1 = (i, j, \alpha_{ij}, \beta_{ij})$ and $\sigma_2 = (j, k, \alpha_{jk}, \beta_{jk})$ the algorithm computes

$$\begin{aligned} \alpha_{ik} &= \alpha_{ij} - \alpha_{jk} \mod q \\ &= x_i - x_k \mod q \end{aligned}$$

$$\begin{aligned} \beta_{ik} &= \beta_{ij} - \beta_{jk} \mod q \\ &= y_i - y_k \mod q \end{aligned}$$

For proof of security, we defer to the original paper. We note that this scheme has extremely efficient composition - only two additions are required. The main drawback of this scheme is the need to publish specific node labels and certify them with the standard signature scheme. This means communication with the master signer is required to add any node to the graph.

4.1.2 Bellare-Neven FBTS-2 Scheme

We now show a scheme that removes the need for explicit node certification. Bellare and Neven discuss several transitive signature schemes in their paper.

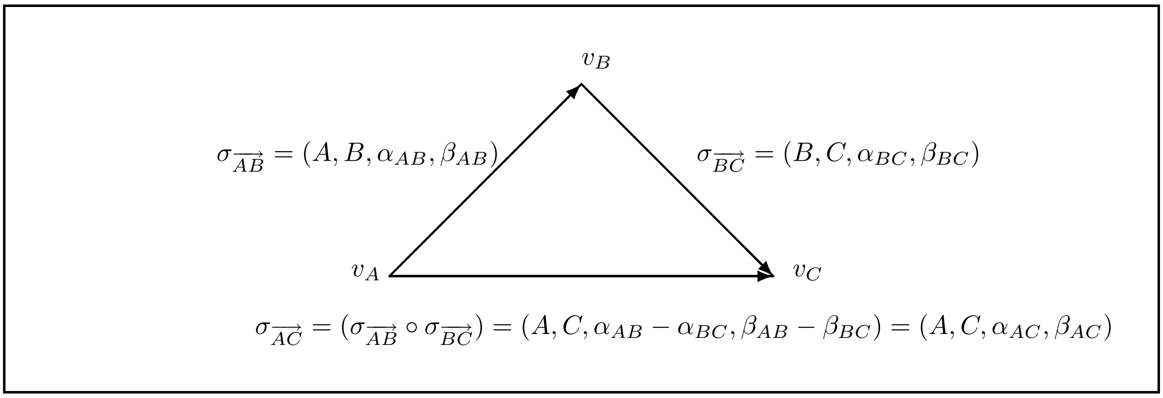

Figure 4.1: Illustration of Micali-Rivest, where v_A, v_B, v_C are public node labels.

We describe the scheme FBTS-2. Unlike the Micali-Rivest scheme, this scheme is set in the group Z/nZ^*, where n is a product of two primes. Further, we require that n is a *Blum integer*, that is, each prime factor is congruent to 3 mod 4. We denote a *Blum modulus generator* by BG; on input 1^k, the modulus generator outputs a random Blum integer of size k bits.

The main idea of the scheme is to use the fact that extracting square roots modulo a composite is hard unless the factors are known. State is removed by using a hash function to create node labels. The node label for node i is then $H(i)$ for some hash function H. Publishing the hash function in the public key means no further node certificates are necessary; anyone can compute $H(i)$ for any i.

A technical difficulty occurs in that the node labels must be squares modulo n. Therefore the output of the hash function H must produce squares. Normally, one would define $H(x)$ as the square of any cryptographic hash function evaluated at x, but this would defeat the purpose of relying on the difficulty of square root extraction.

Instead, we exploit the fact that n is a Blum integer and that modulo a Blum integer, exactly one square root of each square is itself a square mod N. Then we can use the *fourth* roots of the hash function as a trapdoor. We then specify a function $H_N : \{0,1\}^* \to Z/nZ^*$ with range elements of Jacobi symbol 1. Because the Jacobi symbol is easily computable in Z/nZ^* even without the factors of n, we can build such an H_N easily from any cryptographic hash function.

Key Generation. The key generation algorithm KG, on input 1^k, runs $BG(1^k)$ to obtain (N, p, q) and outputs $pk = N$ as the public key and $sk = (N, p, q)$ as the matching secret key. In addition, the public key specifies a hash as defined above, denoted $H_N : \{0,1\}^* \to Z_N^*$.

Node Certification. No explicit node certification is necessary in the FBTS-2

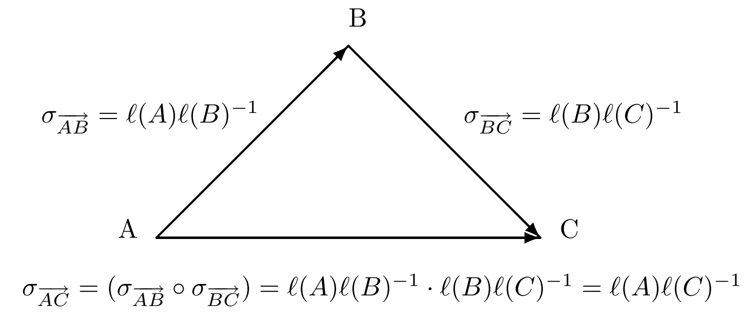

Figure 4.2: Illustration of $\mathsf{FBTS\text{-}2}$, where $H_N(A), H_N(B), H_N(C)$ are public node labels and $\ell(A), \ell(B), \ell(C)$ are the secret node labels. One can then verify that $(\sigma_{AC})^2 = H_N(A)H_N(C)^{-1}$. All values taken $\mod N$.

scheme.

Edge Signing. The ESign algorithm maintains state (V, ℓ), where $V \subseteq N$ is the set of all queried nodes and the function $\ell : V \to Z_N^*$ assigns to each node $i \in V$ a secret label $\ell(i) \in Z_N^*$. When invoked on the edge $\{i, j\}$ together with the secret key sk, it first swaps i and j if necessary to ensure $i < j$. Then if $i \notin V$, the algorithm sets the public label $L(i) = H_N(i)^2$, and similarly for j. The secret label is then set to $\ell(i) \to^R L(i)^{\frac{1}{4}} \mod N$, where $\to^R$ means a fourth root is drawn uniformly from the set of all such roots. Finally, the algorithm outputs $\ell(i)\ell(j)^{-1} \mod N$ as the edge signature.

Verification. The Vf algorithm takes the public labels of nodes i, j and a value σ, and returns 1 if and only if σ is a valid signature from i to j. First the algorithm swaps i and j if necessary to ensure $i < j$. Then it checks whether $H_N(i)^2 = \sigma^4 H_N(j)^2 \mod N$ and returns 1 in this case and 0 otherwise.

Composition. The Comp algorithm takes the public labels of nodes i, j, k and two values σ_1 and σ_2, and tries to output $\sigma_{\overrightarrow{ik}}$. If $j < i$ then it sets $\sigma_1 \leftarrow \sigma_1^{-1} \mod N$, and if $k < j$ then it sets $\sigma_2 \leftarrow \sigma_2^{-1} \mod N$. Finally it outputs $\sigma_{\overrightarrow{ik}} \leftarrow \sigma_1 \cdot \sigma_2$.

For proofs of correctness and security, we defer to the Bellare and Neven paper.

4.2 Redactable Signature Schemes

The problem. Redactable signatures are intended to model a situation where a censor can delete certain substrings of a signed document without destroying

Alice spends 60 hours a week trying to find ways to add value to our bottom line, and never have I known her to shirk her duties. Alice is a true asset to our company, and I cannot think of one person better suited to your requirements.

Alice spends 60 hours a week trying to find ways to shirk her duties, and I can think of one person better suited to your requirements.

Alice spends 60 hours a week trying to find ways to ♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯ ♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯ ♯♯♯ shirk her duties ♯♯♯♯♯♯ ♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯ ♯♯, and I can♯♯♯ think of one person better suited to your requirements.

Figure 4.3: An illustration of the need to disclose the location of redactions. The left shows a sample document that one might sign with a redactable signature scheme. The middle shows one substring that might be obtained if the location of redactions is not made explicit in the result; note how the meaning of the original document has been violated. The right shows the corresponding redaction obtained if deletions are disclosed explicitly; note how the attempted trickery is revealed by this countermeasure. Because of this potential for this sort of semantic attack when the location of deletions are not made explicit, as illustrated here, our construction follows the model shown on the right of making deletions explicit.

the ability of the recipient to verify the integrity of the resulting (redacted) document. In particular, we allow the censor to replace arbitrary bit positions in the document with a special symbol $\sharp$ representing the location of the deletions.

Semantic attacks and our formulation of redaction. Making the problem statement more precise exposes a subtle trap here. A natural first attempt at a definition might require that, given a signature $\mathsf{Sig}(x)$ on x, one can obtain a signature $\mathsf{Sig}(w)$ on any substring w of x that is obtained by deleting some of the symbols in x. However, this formulation of the problem has a serious limitation: such a scheme will conceal the presence of deletions, which introduces the risk of semantic attacks. Without indication of where the redaction has occurred, an attacker might legally be able to truncate the end of one sentence, delete the beginning of the next, and slice them together to get a message the sender would not have authorized. For an example, see Figure 4.3. To defeat semantic attacks, we instead use a formulation where the presence of redacted segments are made explicit.

4.2.1 Defining Redactable Signature Schemes

We define the concept more formally as follows. Let $\Sigma = \{0, 1, \sharp\}$. We define a partial order $\preceq$ on Σ so that $\sharp \preceq 0$, $\sharp \preceq 1$, and $a \preceq a$ for each $a \in \Sigma$ (and no other non-trivial relations hold). This induces a partial order $\preceq$ on Σ^* by pointwise comparison, namely, $w_1 \cdots w_n \preceq x_1 \cdots x_n$ holds if $w_i \preceq x_i$ for each i. We say that the document w is a *redaction* of x if $w \preceq x$, or in other words, if w can be obtained from x by replacing some positions of x with the $\sharp$ symbol. The signature scheme $\mathsf{Sig} : \Sigma^* \to R$ is called *redactable* if we can derive from a signature $\mathsf{Sig}(x)$ on x a signature $\mathsf{Sig}(w)$ on any redaction $w \preceq x$ we like, without the help of the signer.

Note that, in this model, we do not attempt to hide the location of the censored portions of the documents. Thus, a signature $\mathsf{Sig}(w)$ on a redaction w of x may legitimately reveal which portions of x were deleted, and specifically, the presence and location of redacted segments is protected from tampering by the signature. On the other hand, redaction does not reveal the previous contents of the redacted portions of the document.

Redactable signatures may be viewed as an instance of a homomorphic signature scheme endowed with operations $D_i : \Sigma^* \to \Sigma^*$ that replace the i-th bit position with a $\sharp$ symbol. The requirement that signed documents be redactable is equivalent to requiring that our signature scheme be homomorphic with respect to these unary operators.

A trivial construction. There is one obvious way to build redactable signatures. We fix a traditional signature scheme Sig_0; no special homomorphic properties are required from Sig_0. We assume for simplicity that our base signature scheme permits message recovery, but this assumption is not essential. In the trivial construction, to sign a message x of length n, we generate a fresh key-pair (s, v) for some secure conventional signature scheme, and then the signature on x is

$$\mathsf{Sig}(x) = \langle \mathsf{Sig}_0(n, v), s(1, x_1), \ldots, s(n, x_n) \rangle.$$

Verification is straightforward.

The key pair essentially serves as a document ID. Without it, an attacker could replace the ith component of any signed message with the ith component of any other signed message, which we do not wish to allow.

To redact a signed message, we simply erase the appropriate segments of the message and the corresponding positions in the signature. For instance, redacting the first symbol in $\mathsf{Sig}(x)$ yields a signature $\langle \mathsf{Sig}_0(n, v), s(2, x_2), \ldots, s(n, x_n) \rangle$. This scheme supports a privacy property: redacted signatures do not reveal the redacted parts of the original message. We can see that they reveal the locations that have been redacted, much like typical redaction of paper documents does, but this is all that is leaked.

This trivial scheme has a serious limitation: the signature $\mathsf{Sig}(x)$ is very long. If s produces m-bit signatures, then the construction above yields signatures of length $mn + O(1)$, which is likely to be large compared to the length of the message, n. Therefore, the challenge is to find a scheme with much shorter signatures.

Redactable signature schemes are actually an example of a slight generalization of a homomorphic signature scheme. We can rewrite the homomorphic requirement as follows. We require the Sig function to satisfy $\mathsf{Sig}(\otimes(x, x')) = \odot(\mathsf{Sig}(x), \mathsf{Sig}(x'))$. In this notation, the property of Sig is that it (almost) commutes with $\otimes$. Although it is most natural to investigate binary operators, we may also look for signature schemes that commute with unary operators, ternary operators, or whatever. Additionally, we may require Sig to commute with several operators simultaneously. As an example, let f_i be the operation, "redact the ith bit of the given message." Then a redactable signature scheme is simply one which commutes with all the operators f_i. So for a redactable signature scheme, there exist unary operators g_i such that $\mathsf{Sig}(f_i(x)) = g_i(\mathsf{Sig}(x))$.

4.2.2 Johnson-Molnar-Song-Wagner Tree Scheme

We describe how to build a secure redactable signature scheme out of any traditional signature scheme. The main idea is to combine Merkle hash trees [37] with the GGM tree construction for increasing the expansion factor of a pseudorandom generator [41]. We first generate a pseudorandom value at each leaf by working down the tree (using GGM), then we compute a hash at the root by working up the tree (using Merkle's tree hashing).

We note that a similar hash tree construction was arrived at independently by Bull, Steinfeld, and Zheng[55]. Our scheme differs in that it uses the GGM construction to ensure that signatures do not reveal the order in which parts of the document were redacted.

We describe in detail how to sign a message x of length n. First, we arrange the symbols of the message at the leaves of a binary tree; we assume that the tree is chosen to have leaves at exactly the n leftmost positions possible, all at the same depth. We identify nodes of the tree with elements of $\{0, 1\}^*$, so that a node η has children $\eta 0$ and $\eta 1$, and the empty string ϵ denotes the root node. In this way, we assume that to each leaf ℓ of the tree there corresponds a symbol x_ℓ of the message x. Let $G : \mathcal{K} \to \mathcal{K} \times \mathcal{K}$ be a length-doubling pseudorandom generator, let H be a cryptographic hash function, and pick $k_\epsilon \in \mathcal{K}$ uniformly at random. We use a three-phase algorithm (see Figure 4.4, which shows phase 1 on the left and phase two on the right):

Expansion: We use the GGM tree construction to associate a key k_η to each node η. In other words, we define a key for each node of the tree by the recursive relation $\langle k_{\eta 0}, k_{\eta 1} \rangle = G(k_\eta)$.

Hashing: We then compute a hash value v_ℓ for each leaf a as $v_\ell = H(0, k_\ell, x_\ell)$ and apply the Merkle hash construction, i.e., we recursively compute $v_\eta = H(1, v_{\eta 0}, v_{\eta 1})$.

Signing: Finally, we sign the root of the hash tree using our base signature scheme Sig_0, and compute the signature on x as $\mathsf{Sig}(x) = \langle k_\epsilon, \mathsf{Sig}_0(v_\epsilon) \rangle$.

Verification is straightforward, since given k_ϵ and x we can compute v_ϵ and check the signature.

Next, we describe how to redact a signed message. To erase position ℓ from a signed message x, we need to reveal v_ℓ. At first sight, revealing $v_\ell = H(0, k_\ell, x_\ell)$

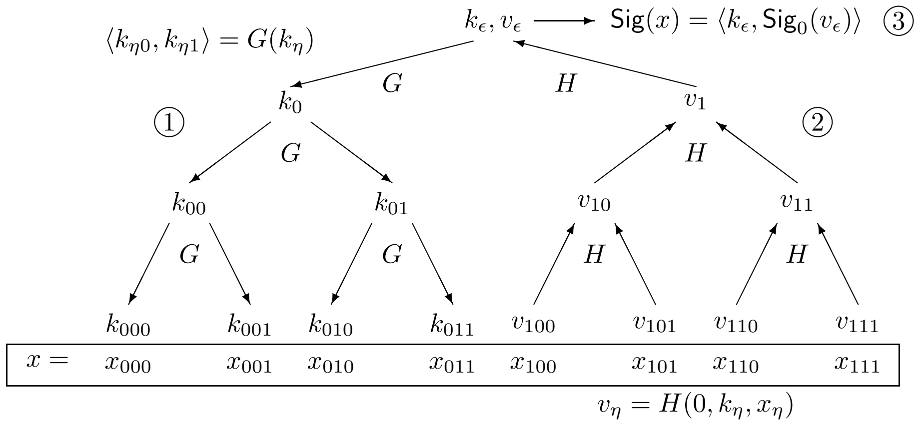

Figure 4.4: A Redactable Signature

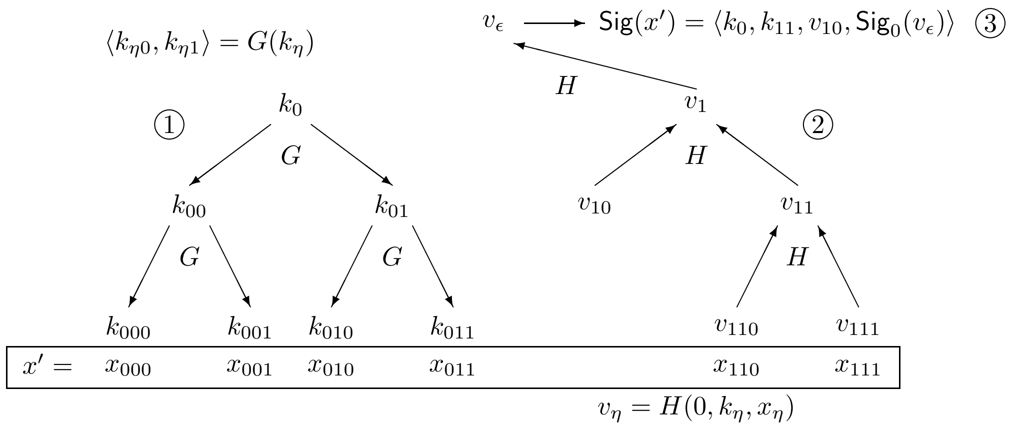

Figure 4.5: A Signature with x_{100} and x_{101} Redacted

would appear to be very dangerous, since given k_ϵ we can compute k_ℓ and then iterate over all possible values of x_ℓ to learn the value of the erased symbol. This would violate the secrecy property.

To restore secrecy, we take advantage of the GGM tree. When we erase x_ℓ, we will also erase k_ϵ, and reveal only what is needed to verify the signature without leaking k_ℓ. The notion we need is as follows: define the *co-nodes* associated to a leaf ℓ to be the siblings of the nodes along the path from ℓ to the root. We reveal k_η at each co-node η associated to ℓ, as this is exactly what is needed to check the resulting signature. Thus, redacting x_ℓ from $\mathsf{Sig}(x)$ would yield $\langle v_\ell, \{k_\eta : \eta \text{ a co-node of } \ell\}, \mathsf{Sig}_0(v_\epsilon)\rangle$. Note that there are at most $\lg n$ co-nodes associated to any single leaf.

We can repeat the above procedure to further redact an already-redacted signature, if we like. In the case of redactions in consecutive parts of the document, we compact the signature by using the tree structure: if the signature contains both $v_{\eta 0}$ and $v_{\eta 1}$ for some η, then we replace these two quantities with the single value v_η, and we similarly replace $k_{\eta 0}, k_{\eta 1}$ with k_η. This both shortens the signature and hides the order in which redactions were performed, ensuring a sort of path-independence property. Thus, though signatures will grow in length, the length may grow only slowly if there is some locality to the sequence of redactions. In general, each consecutive segment of erasures of length n' from a message of length n adds $O(\lg n')$ hash values and $O(\lg n')$ key values to the signature.

This signature scheme produces signatures that are relatively short. If Sig_0 yields m-bit signatures and if we use m'-bit hash values and keys, then an unredacted signature is only $m + m'$ bits long. This is a constant in n, the length of the message. After erasing s segments, each of length at most n', the signature will be $m + O(sm' \lg(nn'))$ bits long. In general, signatures could be as long as $m + nm'$ bits after many redactions, but we can expect that in practice the number of erased segments will typically not be too large and hence signatures should often be quite short. Therefore, this is a considerable improvement over the trivial construction outlined earlier.

This signature scheme also has an additional homomorphic property. For a fixed message x, we can form the lattice of substrings $L_x = \{w : w \preceq x\}$, where the join $v \sqcup v'$ of v and v' is the unique least $w \in L_x$ satisfying $v \preceq w$ and $v' \preceq w$, and the meet is defined similarly. We note that given two redactions $\mathsf{Sig}(v)$ and $\mathsf{Sig}(v')$ of a common signature on a message x, we can compute signatures $\mathsf{Sig}(v \sqcup v')$ and $\mathsf{Sig}(v \sqcap v')$ on their join and meet. In other words, our scheme is also homomorphic with respect to the binary operations $\sqcup$ and $\sqcap$. We imagine that this might be useful in some applications.

Security analysis. We show that this construction is secure under reasonable cryptographic assumptions. We do not address security against random forgeries, as it is not clear what is the right distribution on the message space.

Theorem 1 *Let G be a (t, ϵ_G)-secure pseudorandom generator, H a (t, p_H)-collision-resistant hash function, and* Sig_0 *a conventional signature scheme that* *is (t, q, p_S)-secure against existential forgery. Then the redactable signature scheme* Sig *defined above for messages of length at most n is (t', q, p')-secure against existential forgery with respect to redaction, where $p' = q\lceil \lg n \rceil \epsilon_G + p_S + 3p_H$ and $t' \approx t$.*

Proof. (sketch) Imagine replacing all values k_η used in computing each signature by truly random values and publishing these values, rather than computing k_η from k_ϵ via G. Consider any adversary that attempts to break the resulting modified scheme by exhibiting existential forgeries, i.e., by finding a valid signature on a message x^* that is not a substring of any of its previous queries $x^1, \ldots, x^q$. Let v_ϵ^i denote the root of the hash tree on the signature for message x^i. If the adversary forges a signature $(k_\epsilon^*, \mathsf{Sig}(v_\epsilon^*))$ on x^*, then there are two cases:

- We have $v_\epsilon^* \neq v_\epsilon^i$ for all $i = 1, \ldots, q$. Then we have found an existential forgery of Sig_0, which by assumption happens with probability at most p_S.

- We have $v_\epsilon^* = v_\epsilon^i$ for some $i = 1, \ldots, q$. One possibility is that the tree for x^* is a sub-tree of the tree for x^i, and the hash values coincide on the common part of the tree. But in this case we have a collision in the hash function of the form $H(0, k_\ell^*, x_\ell^*) = H(1, v_{\ell 0}^i, v_{\ell 1}^i)$, which by assumption happens with probability at most p_H. There is also the symmetric case where the tree for x^i is a sub-tree of the tree for x^*, but this is handled similarly. Because of the way the tree shapes are chosen, the only other possibility is that there is some internal node common to both trees where the hash values do not agree. We treat this next.

  We know there is at least one string η that forms an internal node in both trees and that satisfies $v_\eta^* \neq v_\eta^*$. Let η be one such string that is as short as possible, i.e., an internal node of minimal depth. Let η' denote the sibling of η. Then we have found a collision $H(a) = H(b)$ with $a \neq b$, where $a = (1, v_\eta^*, v_{\eta'}^*)$ and $b = (1, v_\eta^i, v_{\eta'}^i)$, and by assumption this happens with probability at most p_H.

We see that existential forgery of this modified scheme happens with probability at most $p_S + 3p_H$. But this means that the probability of finding an existential forgery for the original scheme Sig must be at most $q\lceil \lg n \rceil \epsilon_G + p_S + 3p_H$, else we would have a distinguisher that contradicted our assumption on the security of G, according to the proof of security for the GGM scheme [41]. The claimed result follows. □

4.2.3 RSA-Based Content Extraction Scheme

Bull, Steinfeld, and Zheng also showed a "content extraction scheme" based on properties of RSA [55]. The semantics of such a scheme are nearly the same as redactable schemes, except that they also allow designating which portions of the message may and may not be redacted. The scheme uses a different public exponent for each bit of the message. A trick due to Fiat can then be used to verify this entire "multi-signature" in one exponentiation, and also to remove particular message components from the signature.

4.3 Set Signature Schemes

We may also look for a set-homomorphic signature: Given a universe $\mathcal{X}$, and a set operation $\odot$, design a signature scheme Sig that is homomorphic with respect to $\odot$.

4.3.1 Accumulator Set Union/Difference Scheme

We now give a signature scheme, without a proof of security, for the case of a scheme homomorphic with respect to union and set difference. Consider the following construction: Let $\mathsf{Sig}_\times : G \to R$ be an arbitrary multiplicative signature scheme on some group G. For a set $U = \{x_1, \ldots, x_k\}$, define the hash function $f_h(U) = h(x_1) \cdots h(x_k)$, where $h : \mathcal{X} \to G$ is a cryptographic hash function on elements in $\mathcal{X}$. Then $\mathsf{Sig} = \mathsf{Sig}_\times \circ f_h$ is a signature scheme with the property that with signatures, $\mathsf{Sig}(U)$ and $\mathsf{Sig}(V)$, on two disjoint sets U and V, one can compute the signature on their union, $\mathsf{Sig}(U \cup V) = \mathsf{Sig}(U) \times \mathsf{Sig}(V)$. If $U \subseteq V$, one can also compute the signature on their difference, $\mathsf{Sig}(V \setminus U) = \mathsf{Sig}(V) \times \mathsf{Sig}(U)^{-1}$.

As another example, using the accumulator technique [7], with a hash function h that maps a set of elements $\mathcal{X}$ to a set of primes, we can construct a signature scheme as follows: Assume n is a RSA modulus and only the signer knows the factorization of n. Assume $v \in (\mathbb{Z}/n\mathbb{Z})^*$ is chosen uniformly at random. For a set $U = \{x_1, \ldots, x_k\}$, we define $\mathsf{Sig}(U) = v^{1/(h(x_1)\cdots h(x_k))} \bmod n$. This signature scheme has the property that given a signature on U, one can generate signatures on any subset of U. To see this, simply observe that $\mathsf{Sig}(U \setminus \{x_i\}) = \mathsf{Sig}(U)^{h(x_i)}$. Also, one can generate the signature on the union of two signed sets. If $v^{1/d}$ and $v^{1/e}$ are signatures on two sets, then the signature on their union is $v^{1/\mathrm{lcm}(d,e)}$. We can compute $v^{1/\mathrm{lcm}(d,e)}$ as $(v^{1/d})^b(v^{1/e})^a$, where a, b form a solution to the equation $ad + be = \gcd(d, e)$ obtained via the Euclidean algorithm. We note that this use of accumulators is similar to the accumulator constructions in the credential schemes of Lysyanskaya and Camenisch[11].

One interesting question is whether we can design a signature scheme that is homomorphic with respect to the union operation *alone*. We will return to this question in Chapter 6.

4.3.2 Hevia-Micciancio One-Time Scheme

Hevia and Miccianicio study the provable security of "one-time" signature schemes based on one-way functions [29]. As a byproduct of their constructions, they achieve a provably secure one-time signature scheme homomorphic with respect to union and subset. Their construction is of particular interest because it uses only one-way functions.

Chapter 5

Towards Directed Transitive Signature Schemes

VIOLA. Cesario is your servant's name, fair princess.

OLIVIA. My servant, sir? 'Twas never merry world
Since lowly feigning was called compliment
Y'are servant to the Count Orsino, youth.

VIOLA. And he is yours, and his must needs be yours.
Your servant's servant is your servant, madam.

Shakespeare, *Twelfth Night*, Act 3 Scene 1.

In 3.6 we defined directed and undirected transitive signature schemes. In 4.1 we saw several examples of undirected transitive signature schemes. Now we turn our attention to the case of directed graphs. To see why this is important, let's take a look at the Shakespeare quote above.

We can build a graph such that nodes in the graph represent people, and edges the "servant of" relation. Viola is servant to the Count Orsino, and Orsino is servant to Olivia. Therefore, as Shakespeare tells us the "servant of" relation is transitive, so Viola is the servant of Olivia.

Unfortunately, in the RSA-based scheme, given a signature on the edge that proves "Viola is the servant of Olivia," it is easy to obtain a signature that proves "Olivia is the servant of Viola." Say that Olivia's node is represented by O and Viola's node by V. The signature on the edge from Olivia to Viola is $(\frac{O}{V})^d \mod n$. To invert direction, simply invert the signature as a group element to yield $(\frac{V}{O})^d \mod n$. The case of the Rivest-Micali and Bellare-Neven schemes is similar; group inversion leads directly to direction inversion. Therefore these schemes work only for undirected graphs.

Micali and Rivest left the question of a directed transitive signature scheme as an open problem[38]. We do *not* solve this open problem. Instead, we introduce a new open problem, that of finding a group with "trapdoor infeasible inversion," and show that solving this problem would yield a directed transitive signature scheme. By reducing the problem of directed transitive schemes to problems closer to core mathematics, we hope to make it easier to apply tools from core mathematics in future research. In addition, such problems may have independent interest.

5.1 Directed Transitive Scheme from Trapdoor Infeasible Inversion

5.1.1 Group with Trapdoor Infeasible Inversion

GROUP. We say that G is a mathematical *group* if it possesses elements S, binary operator $\circ$, and identity $e \in S$ such that the following criteria hold:

1) Closure - $\forall x, y \in S,\ (x \circ y) \in S$ (i.e. all binary applications of $\circ$ on group members remain in the group.)

2) Identity - $\forall x \in S,\ e \circ x = x \circ e = x$ (i.e. there exists an identity element in the group that commutes with all other members and leaves them unchanged.)

3) Inverses - $\forall x \in S,\ \exists y \in S$ such that $x \circ y = e$ (i.e. every element has an inverse in the group.) Let the inverse of x be denoted as x^{-1}.)

4) Associativity - $\forall x, y, z \in S,\ (x \circ y) \circ z = x \circ (y \circ z)$ (i.e. the operator is associative.)

Generally speaking, operator $\circ$ does not have to be efficient, but for the rest of the paper we restrict our focus to groups with polynomial time computable $\circ$. Note that if G meets all the above properties for a group and commutes, then we say that G is an *Abelian* group.

GROUP WITH INFEASIBLE INVERSION (GII). Given any mathematical group G, we say that G is a *group with infeasible inversion* if it has the additional property:

5) Infeasible Inversion - for a random $x \in S$, it is hard to find the inverse $x^{-1} \in S:\ x \circ x^{-1} = e$.

We mean hard in the cryptographic sense, such that for all $k \in \mathbb{N}$ and for all probabilistic polynomial time adversaries A_k,

$$Pr[x \stackrel{R}{\leftarrow} S_k, A_k(x) \rightarrow y\ :\ y \in S\ \wedge\ x \circ y = e] < \frac{1}{poly(k, |x|)}.$$

where the probability is taken over the choice of x and the random coins of A_k.

TRAPDOOR GROUP WITH INFEASIBLE INVERSION (TGII). Given any GII G, we say that G is a *trapdoor group with infeasible inversion* if there exists a

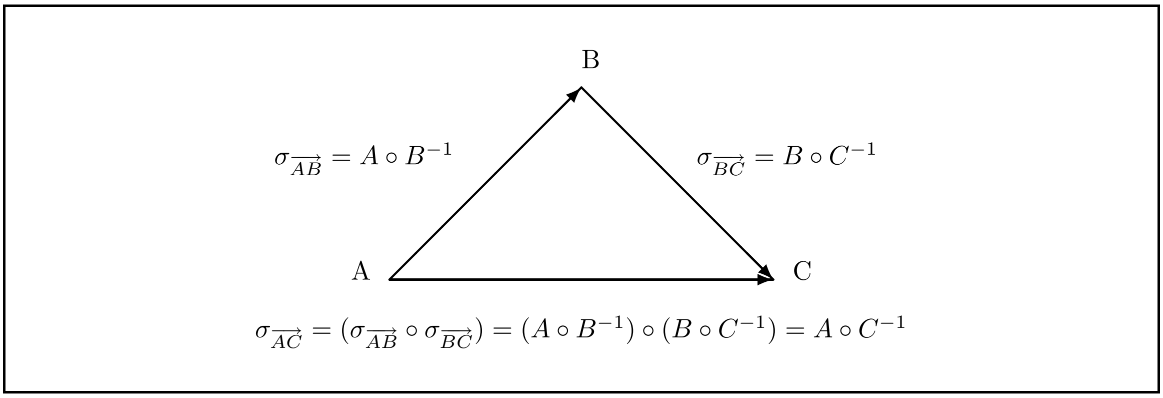

Figure 5.1: Illustration of DTS scheme using a TGII group, where A, B, C are public node labels and A^{-1}, B^{-1}, C^{-1} are the secret keys. One can then verify that $\sigma_{AC} \circ C = A$.

trapdoor t, such that $|t| \leq poly(k)$, and given t finding inverses (property 5) becomes polynomial-time computable, without t it remains hard.

6) Trapdoor - given t and $x \xleftarrow{R} S$, finding $y \in S \ : \ x \circ y = e$ is polynomial-time computable.

5.1.2 TGII $\longrightarrow$ DTS

Theorem 2 *The existence of a* TGII *group implies the existence of a* directed transitive signature *scheme, whose security is directed transitively unforgeable under adaptive chosen message attack. (*TGII *is sufficient for* DTS*.)*

Proof. We construct a simple DTS scheme from TGII $G = (S, \circ, e, t)$, demonstrate its correctness, and prove its security. Let $||$ denote concatenation.

Preliminaries. We postulate the existence of a TGII group G. We also assume a standard digital signature scheme SDS=(SKG,SSign,SVf).

Key Generation. The KG algorithm takes 1^k as input. First, it calls SKG on 1^k to generate a key pair (spk, ssk) for the standard signature scheme SDS. It then publishes $\mathsf{PK}=(e, spk)$ as well as the description of $\circ$, and gives $\mathsf{SK}=(k, S, t, ssk)$ to the master signer.[1]

Node Certification. The NCert algorithm maintains state (N, A, ℓ, L, Σ) and takes as input $\mathsf{SK}=(k, S, t, ssk)$ and a node index i from the master signer.

[1]Here we assume that one TGII group exists and t is not known to anyone. If a family of TGII groups were found, then KG could randomly select a single group from the family.

If $i \notin N$, then $N \leftarrow N \cup \{i\}$, $\ell(i) \xleftarrow{R} S_k - A$, $L(i) \leftarrow \ell(i)^{-1}$, $A \leftarrow A \cup \{\ell(i)\} \cup \{L(i)\}$, and $\Sigma(i) \leftarrow \mathsf{SSign}(ssk, i||L(i))$. The algorithm uses N to check that each node is signed only once, and A ensures that all nodes have unique key pairs $(L(i), \ell(i))$. The algorithms publishes $pk_i = (L(i), \Sigma(i))$, where $L(i)$ is the public label for node i, and $\Sigma(i)$ is a certificate for guarantee $L(i)$ belongs to node i, and gives $sk_i = (\ell(i))$ to the master signer. Note that NCert can compute $L(i)$ from $\ell(i)$ because it takes t as input.

Edge Signing. The ESign algorithm takes two node indices i, j as input. If $(i \neq j) \wedge (i \in N) \wedge (j \in N)$, then it returns $\sigma_{\overrightarrow{ij}} = L(i) \circ \ell(j)$, otherwise it declares failure and returns $\perp$.

Verification. The Vf algorithm takes the public labels of nodes i, j and a value σ, and returns 1 if and only if σ is a valid signature from i to j. Parse pk_x as (x, L_x, Σ_x).

If $(\mathsf{SVf}(spk, i||L_i, \Sigma_i){=}1) \wedge (\mathsf{SVf}(spk, j||L_j, \Sigma_j){=}1) \wedge (\sigma \circ L_j = L_i)$ then return 1; otherwise return 0.

Composition. The Comp algorithm takes the public labels of nodes i, j, k and two values σ_1 and σ_2, and tries to output $\sigma_{\overrightarrow{ik}}$. Parse the public node label pk_x as (x, L_x, Σ_x).

If $(\mathsf{SVf}(spk, i||L_i, \Sigma_i){=}0) \vee (\mathsf{SVf}(spk, j||L_j, \Sigma_j){=}0) \vee (\mathsf{SVf}(spk, k||L_k, \Sigma_k){=}0)$ then return $\perp$; otherwise return $\sigma \leftarrow \sigma_1 \circ \sigma_2$.

Correctness. Recall the correctness requirement of Section 3.6, where a scheme is correct if the experiment in Figure 3.2 never returns true (i.e. $(\mathsf{Legit} \wedge \mathsf{NotOK})$=true). As long as Legit remains true, it is impossible for NotOK to become true. It holds trivially in this case, because there is a unique secret key $\ell(i) = L(i)^{-1}$ for each public key $L(i)$.

Security. Assuming that G is TGII , we argue that the above scheme is *directed transitively unforgeable under adaptive chosen message attack*. For polynomially-bounded adversary F and security parameter k, we conduct the experiment $\mathbf{Exp}^{dtu-cma}_{DTS,F}(k)$ from Section 3.6. Recall that F wins if and only if, she produces a valid signature $\sigma_{\overrightarrow{ij}}$ for $i, j \in V$, and yet edge $\overrightarrow{ij}$ is not in the transitive closure of $G = (V, E)$. We wish to show that the probability that this occurs is negligible.

We enumerate cases:

1. If F can create node labels on her own, then she breaks the underlying standard signature scheme SDS

2. If F can forge signatures using only the public node labels, then she can break the infeasible inversion property of TGII

3. If F can obtain secret node information from a valid edge (i.e. obtain ℓ_i from $L_j \circ \ell_i$), then she also contradicts the infeasible inversion property of TGII. Suppose for contradiction that F could obtain ℓ_i from $L_j \circ \ell_i$. We show that F can invert an arbitrary element b of its choice. First F sets the value of $L_j = b$. Next F runs with $L_j \circ \ell_i$ set to the identity element $e \in G$. The resulting value of ℓ_i is then b^{-1}. In the next chapter we will codify this into a property of all groups with infeasible inversion.

Thus, F must use a combination of the public node labels and valid edges provided by the master signer, plus any valid edges in the transitive closure of G, to make her σ - a valid edge outside of the transitive closure.

Without loss of generality, let $\sigma = L(i) \circ L(j)^{-1} = E_1 \circ E_2 \circ E_3 \ldots E_n$ be the result of F. Observe that each valid edge is the composition of one public and one secret label. To cancel a label from the equation above, its inverse must be supplied; thus, it will not help the adversary to use the public labels. In the equation above, let J be the set of pk_i's and M be the set of sk_i's in $\{E_1, E_2, E_3, \ldots, E_n\}$. By the properties of TGII the inverse of an edge is hard to find. So once a node k's pk_k is added to set J, it can not be removed until its inverse (sk_k) is added to set M. Each "removal" of a node k's information from sets J and M, represents that k was both the source and the destination of some two edges. Therefore, given any σ as the $\circ$-product of edges, we see that σ must be in the transitive closure of $G = (V, E)$.

Therefore, F's advantage $\mathbf{Adv}^{dtu-cma}_{DTS,F}(k)$ comes from breaking SDS or TGII which are both negligible. $\Box$

Chapter 6

Infeasible Inversion: Implication and Separation

In the previous chapter, we introduced a new primitive: a group with *infeasible inversion.* We also showed that a group with *trapdoor* infeasible inversion implies the existence of a directed transitive signature scheme. Now we look at the implications of the new primitive. Our main result is that a group with infeasible inversion, even without a trapdoor, implies the existence of key agreement. The proof exploits a similarity between groups with infeasible inversion and the "strong associative one-way functions" of Rabi and Sherman. We combine this with the results of Impagliazzo and Rudich to show that there can be no black-box construction of TGII or GII from one-way functions.

6.1 Black-Box Reductions and Impagliazzo-Rudich

There are many primitives in cryptography, and this thesis adds more to the pile. Two of the most fundamental primitives are *one-way functions* and *key agreement.* We defined one-way functions in our preliminaries, and we now define key agreement[17].

KEY AGREEMENT. A key agreement protocol is a protocol run by two parties Alice and Bob. At the end of the protocol, both Alice and Bob agree on a secret s with high probability. An eavesdropper Eve watching the messages sent between Alice and Bob can determine s only with negligible probability.

No one knows whether any of these primitives exist; proving that they do is in almost all cases is as hard as proving $P \neq NP$. Instead, much work has focused on proving *implications* between these primitives. For example, we know that one-way functions (OWF) imply pseudorandom generators[27], which in turn imply bit commitment[40] and symmetric encryption.

Impagliazzo and Rudich suggested proving *black-box separations* as well. In a black-box separation between two cryptographic primitives, we show that giving *oracle access* to primitive P_1 is not sufficient for any efficient construction of primitive P_2. In other words, we cannot build P_2 given only the ability to call P_1 as a "black box." Making this notion precise can be tricky. For instance, if both P_1 and P_2 exist, a construction algorithm can simply throw away its oracle access to P_1 and construct P_2 directly.

By "oracle access" to a primitive P, we mean that the construction is an oracle Turing Machine with access to P. For a formal definition of oracle Turing Machines, we refer to a textbook on complexity theory such as Sipser or Papadimitriou[54, 44]. Informally, an oracle Turing Machine is an ordinary Turing Machine augmented with a special pair of "oracle tapes." The machine can write to one of the tapes and request an answer. The answer is then written to the other oracle tape. In this way, M^O (read "a Turing Machine M with access to oracle O") can evaluate $O(x)$ for any x that it can compute. For example, we could take O to be a one-way function, and then M^O can query the one-way function oracle.

Now we can make the notion of a black-box separation precise. We say that primitive P_1 is black-box separated from primitive P_2 if there exists an oracle O such that relative to O primitive P_1 exists but the primitive P_2 does not. Impagliazzo and Rudich were able to show the following theorem.

Theorem 3 *One-way functions are black-box separated from key agreement. That is, there exists an oracle O relative to O one-way functions exist, but key agreement does not. Further, relative to O, we have $P = NP$.*

Proof. Impagliazzo and Rudich [32] □

The theorem shows that, in some sense, key agreement is "harder" to acheive than one-way functions: there is no black-box construction of key agreement from one-way functions unless $P \neq NP$. This makes precise our intuition based on practice. While there are many candidate one-way functions, there are comparatively few key agreement protocols.

The oracle separation methodology has yielded several other results. Impagliazzo and Rudich separated k-round key agreement from $k + 1$-round key agreement. Rudich's PhD thesis gives a conjecture that would lead to a black-box separation of one-way functions from one-way permutations; this conjecture was proved by Kahn, Saks, and Smyth[51, 34]. Recently, work by Gertner, Kannan, Malkin and Reingold and Gertner, Malkin, and Reingold has focused on separations between oblivious transfer, trapdoor predicates, and trapdoor functions [22, 23]. Dodis and Reyzin showed that "claw-free" functions could not be build from one-way functions. Simon showed that one-way functions are black-box separated from cryptographic hash functions[53]; Kim, Simon, and Tetali followed up to show limits to the efficiency of hash funtions based on one-way permutations[35].

It is worth noting that cryptographic constructions that use zero-knowledge proofs are *not* ruled out by these results; for example, Dolev, Dwork, and Naor's construction of encryption secure against chosen ciphertext attack given a public-key encryption scheme is not a black-box construction [18]. Gertner et al also note that the "usual" reduction of malicious oblivious transfer to "honest but curious" oblivious transfer is not black-box for this reason; they prove a separation result that can be interpreted as showing the zero-knowledge proofs are necessary in such a reduction. Extending the oracle separation methodology to include zero-knowledge proofs remains open and is outside the scope of this thesis. We do remark, however, that Fischlin has shown one-way functions are insufficient for some forms of statistical zero-knowledge [20].

6.2 Strong Associative One-Way Functions

Rabi and Sherman introduced "strong associative one-way functions" as a cryptographic primitive, following a protocol for key agreement attributed to Rivest and Sherman[46]. We will see that their notion has a close parallel to our "groups with infeasible inversion."

Strong Associative One-Way Functions (SAOWF). Given any honest binary function $\circ$: $S \times S \to S$ on the message space $S = \{0,1\}^*$, we say that $\circ$ is *associative one-way* if: (1) for all $x, y, z \in S$, $(x \circ y) \circ z = x \circ (y \circ z)$; and (2) $\circ$ is polynomial-time computable, but inverting $\circ$ is not. We say that $\circ$ is *strong* if inverting $x \circ y$ remains hard when given x or y.

The existence of a strong associative one-way function is, of course, an open question, as an affirmative answer would imply $P \neq NP$. Surprisingly, Hemaspaandra and Röthe were able to show that the converse holds: if $P \neq NP$ then a SAOWF exists[28]. Unfortunately, the resulting SAOWF is only hard in the *worst case*, rendering it useless for cryptography.

6.3 Implications of GII

We now focus on groups which have infeasible inversion without a trapdoor. First we show that the existence of such a group implies the existence of SAOWF . Then we show that GII implies key agreement in the sense of Impagliazzo and Rudich. Finally, we conclude that no black-box construction of GII or TGII exists from one-way functions.

6.3.1 GII $\longrightarrow$ SAOWF

Lemma 4 *The operator* $\circ$ *in a* GII *group is a* strongly associative one-way function.

Proof. Let $G = (S, \circ, e)$ be a GII group. We recall the definition of SAOWF and show that $\circ$ satisfies these properties. Given any honest binary function $\circ$: $S \times S \to S$ on the message space $S = \{0,1\}^*$, we say that $\circ$ is *associative one-way* if: (1) for all $x, y, z \in S$, $(x \circ y) \circ z = x \circ (y \circ z)$; and (2) $\circ$ is polynomial-time computable, but inverting $\circ$ is not. We say that $\circ$ is *strong* if inverting $x \circ y$ remains hard when given x or y.

By the definition of GII we know that $\circ$ is associative and polynomial-time computable. Now, we show that no probabilistic polynomial time adversary A can take as input $(x \circ y, x)$ and produce y with non-negligible probability.

Suppose such an A does exist. Then she can be used to construct an adversary A' that breaks property 5 of the GII group with non-negligible probability. On input b, A' gives input $(1, b)$ to A and returns whatever A returns, which we know is $d \in S \ : \ b \circ d = e$ (the inverse of b) with non-negligible probability. This violates property 5 of the GII group, thus the *strong associative one-wayness* of $\circ$ is guaranteed. □

This result is important for two reasons. First, no practical constructions of SAOWF are known. Now we can approach the problem by looking for a group with infeasible inversion. Second, we can use the lemma to prove the main theorem of this chapter.

6.3.2 GII $\longrightarrow$ KA

We are now ready to prove the main result of this chapter. The proof consists of an adaptation of the protocol of Rivest and Sherman for strong associative one-way functions to the GII case. We note that their protocol using strong associative one-way functions has no proof of security according to Rabi and Sherman[47], while we are able reduce breaking the GII modification of their key agreement protocol to breaking the "infeasible inversion" assumption.

Theorem 5 *The existence of a* GII *group implies two-party secret key agreement.*

Proof. Let $G = (S, \circ, e)$ be a GII group. Alice and Bob can now agree on a secret key over an open channel, with Eve listening, as:

1. Alice chooses $x \stackrel{R}{\leftarrow} S$ and $y \stackrel{R}{\leftarrow} S$, and sends the pair $(x \circ y, y)$ to Bob.
2. Bob chooses $z \stackrel{R}{\leftarrow} S$, computes $K_B = (x \circ y) \circ z$, and sends $y \circ z$ to Alice.
3. Alice computes $K_A = x \circ (y \circ z)$.

For key agreement to be successful, (1) $K_A = K_B$ and (2) Eve must not learn the secret key. Property 1 follows from the associativity of G. Assume that property 2 does not hold for the above key agreement scheme, namely that there is a probabilistic polynomial time algorithm Eve that can take as input all the messages $(x \circ y, y, y \circ z)$ and produce $K_B = (x \circ y) \circ z$. Since Bob is also a probabilistic polynomial time algorithm, he can use Eve to recover x from Alice's first message $(x \circ y, y)$ as follows:

1) On input $(x \circ y, y)$ to Bob, call Eve on $(x \circ y, y, e)$.

2) Here z is the inverse of y, so Eve will return x.

This is a contradiction, however, of Lemma 6.3.1. We know that G forms a SAOWF, and thus Bob's attack is not possible, and no such Eve can exist. □

Corollary 6 *OWF* $\not\rightarrow$ GII

Proof. By Impagliazzo and Rudich, KAhas no black-box construction from one-way functions. Because GII$\rightarrow$ KA, it follows that GII has no black-box construction from one-way functions. □

This result points towards a possible separation between directed transitive signatures and one-way functions. First, of course, we would need to prove that DTS $\rightarrow$ GIIẆe might also try to prove a separation between undirected and directed transitive schemes. For more discussion of these questions, see Hohenberger's thesis[31].

6.4 Union-Only Signatures and GII

In Section 4.3 we introduced another open problem, that of "union-only" set signature schemes homomorphic with respect to union but not set difference. Now we show that these signatures imply the existence of a group with infeasible inversion.

Theorem 7 *Union-only signatures* $\rightarrow$ GII

Proof. We check that the signatures of a signature scheme homomorphic with respect to union but not set difference, together with the composition operator of the set signature scheme, form a GII

1. Associativity: Holds because set union is associative.
2. Closure: Holds because for a universe X, the power set 2^X is closed under finite set union.
3. Inverses: Inverses always exist, as long as the set signature scheme is defined on the entire power set 2^X.
4. Infeasible Inversion: We prove the contrapositive. Suppose inversion with respect to union is easy. Then given the set $A \cup B$ and the set B, it is easy to obtain A. This is exactly the set difference $(A \cup B)\ B$. Therefore infeasible inversion of union proves set difference is infeasible as well.

□

As a corollary, we obtain that union-only signatures also have no black-box construction from one-way functions. Given that Miccancio and Hevia show one-time signatures homomorphic with respect to union and subset constructed from one-way functions, this is particularly interesting.

Chapter 7

Conclusion and Open Problems

We showed that homomorphic signature schemes open new applications and raise new theoretical questions in cryptography. In Chapter 4, we gave constructions of several homomorphic signature schemes, together with motivating applications. We expect that more applications will be found. We also described signature schemes with homomorphic properties that can be constructed *without* using RSA; the Johnson-Molnar-Song-Wagner redactable scheme and the Micciancio-Hevia one-time schemes need only hash functions and one-way functions, respectively.

Then in Chapter 5 we described an open problem, due to Micali and Rivest, of directed transitive signature schemes. We introduced the open problem of finding a group with "trapdoor infeasible inversion" as a means of attacking the problem of directed transitive signature schemes. Finally, in Chapter 6 we showed a group with infeasible inversion, even without a trapdoor, implies key agreement and so has no black-box construction from one-way functions.

Our work is just part of the intersection between algebra and cryptography. The recent stream of work on "bilinear forms" in cryptography, initiated by Boneh and Franklin, shows how primitives with special algebraic properties can solve previous long-standing open problems, such as identity-based encryption[9, 2]. Homomorphic encryption schemes can be used to make more efficient zero-knowledge proofs and secure multiparty computation. We hope that homomorphic signatures will be similarly fruitful. Much work remains to be done.

7.1 Open Problems

We end by listing some open problems and future directions. This is not meant to be exhaustive. Instead, it points to areas where interesting work appears possible.

7.1.1 Cleanly Rule Out "Trivial" Schemes

In Section 3.7, we discussed the problem of "trivial" signature schemes, such as the additive signature scheme. We also introduced the notion of "security against random forgeries." Unfortunately, this security notion requires drawing at random from a message space. In some cases, it is not clear what distribution is most appropriate. Is there a "cleaner" way to rule out trivial schemes in general?

7.1.2 Algebraic Separations of Homomorphic Signature Schemes

One of our main tools in constructing homomorphic signature schemes is the RSA signature scheme, which is a group homomorphism over Z/nZ^*. Several constructions involving RSA, such as the Bellare-Neven undirected transitive scheme or the Johnson-Molnar-Song-Wagner set scheme, seem to use RSA in a "black-box" way to construct new homomorphic signature schemes. Are there algebraic structures that *cannot* be constructed using RSA as a black box? More generally, can we black-box separate homomorphic signature schemes based on their algebraic structure?

We do not have an answer for this question, but there are some theorems from computational complexity that seem suggestive, so we describe them here. First we define a *semigroup* as a set S closed under an associative operation. A *monoid* is a semigroup with an identity element. A monoid is a group, except it does not guarantee the existence of inverses.

Straubing discusses the question of when one monoid can "simulate" another; all definitions and proofs in this section appear in his paper[56]. Informally, we can simulate a finite monoid M with another finite monoid N if, given a string of elements in M, we can use N as a "computational device" to compute their product in M. In the homomorphic signature case, we can think of attempting to build a signature scheme homomorphic with respect to M given a scheme homomorphic with respect to N.

Formally, we say monoid M *simulates* a finite monoid N if and only if there is a map $\phi : M \to N$, as well as a map $\psi : N \to M$, such that

$$\psi(\phi(m_1) \ldots \phi(m_k)) = m_1 \ldots m_k,$$

for all $m_1, \ldots m_k \in M$.

Let us further say that a finite monoid M *divides* a finite monoid N if M is the homomorphic image of a submonoid of N. That is, if M divides N, there is a submonoid N' of N and a homomorphism $\psi : N' \to M$ that maps onto M. We then have the following theorem.

Theorem 8 *Let M and N be finite monoids. N simulates M if and only if M divides N.*

Proof. (Straubing) Because M divides N, there exists a submonoid N' of N and a homomorphism $\psi : N' \to M$ that maps onto M. For each $m \in M$, define

$\phi(m)$ to be any $n \in N'$ such that $\psi(n) = m$. Since ψ is a homomorphism, we have

$$\psi(\phi(m_1) \dots \phi(m_k)) = m_1 \dots m_k$$

for all $m_1, \dots, m_k \in M$. Conversely, let N simulate M with maps ϕ and ψ as in the definition. Let N' be the set of all products $\phi(m_1) \dots \phi(m_i)$ with $m_1, \dots, m_i \in M$ and $i \geq 0$. The set N' is a submonoid of N and it follows from the definition of simulation that the restriction of ψ to N' is a homomorphism onto M. Therefore M divides N. □

This leads to the following separations between algebraic structures.

- A non-commutative monoid cannot be simulated by a commutative monoid.
- Say a monoid M is *aperiodic* if for all $a \in M$ and for all $k > 0$, we do not have $a^k = a$. A nontrivial finite aperiodic monoid cannot be simulated by a finite group.
- A nontrivial finite group cannot be simulated by a nontrivial finite aperiodic monoid.

This notion of simulation can be generalized to allow the maps ϕ and ψ mapping between monoids to depend on the input letters. The degree of dependence changes which monoids may simulate which others.

We say that M is 1-simulated by N if and only if there is a map $\psi : N \to M$ such that for all sequences $m_1, \dots, m_k$ from M,

$$\psi(\phi_{1,k}(m_1) \dots \phi_{k,k}(m_k)) = m_1 \dots m_k$$

Here each $\phi i, k$ is a separate homomorphism that may depend on the corresponding input letter m_i. It turns out that 1-simulation does not add much to the class of monoids that can be simulated, as is shown by this theorem.

Theorem 9 *Let M and N be finite monoids. If M 1-simulates N, then M divides a direct product of finitely many copies of N.*

Proof. Omitted. See Straubing [56].□

We can also define a notion of k-simulation. The case of k-simulation for $k > 1$ is more complicated, and touches on issues of circuit complexity, regular languages, and computation by groups. We omit this discussion.

From here, we would need to investigate whether, for some notion of black-box construction, a construction of a signature scheme S_1 homomorphic with respect to monoid M using scheme S_2 homomorphic with respect to monoid N implies that M simulates N. If the answer is yes, then we can use the above theorems or adaptations of them to rule out any such construction. We can also investigate the converse, and ask whether a simulation result yields a new construction of a homomorphic signature scheme.

7.1.3 Signature Revocation

With ordinary digital signatures, compromised keys can be *revoked*, or declared as null and void. In the case where an authority signs keys for others, these signatures themselves or *certificates* can also be revoked. Revocation handling is an area of intense research, but one basic idea is to simply publish a *revocation list*. If a signature or the key is on the list, it is no longer valid.

In the case of homomorphic signature schemes, it also makes sense to revoke signatures. In the Introduction, we discussed a scheme for router route aggregation. Chari, Rabin, and Rivest note that if a router is compromised, then the homomorphic signature authorizing it to route for a particular subdomain should be revoked[13]. The homomorphic revocation problem is this: should the signatures formed by composing the revoked signature be revoked as well? if they should, how do we decide whether a given signature was formed by composing with the revoked signature and so should also be revoked?

The privacy requirement of homomorphic signature schemes makes it infeasible for anyone to distinguish between an original signature and a signature formed by composition. Barring a solution that makes composed signatures distinguishable from original signatures if and only if revocation occurs, the best we can do seems to be to ask whether a given signature is in the span of the revoked signature(s).

We can devise algorithms for decomposition in the case of specific signature schemes, as discussed in Section 3.4.3. We can also ask a more general question. Suppose the composition operator is given by a "rewrite system" of generators and relations. Then the problem of deciding whether a signature is in the span of a set of revoked signatures is exactly the submonoid problem:

Instance: A finite set of elements $U \subseteq M$ and an element $w \in M$

Question: Does w belong to the submonoid $< U >$ of M generated by U ?

In general, this problem is undecidable, and examples exist of finitely presented monoids with undecidable submonoid problem. Otto gives criteria for restricted subclasses of rewrite systems where the submonoid problem is polynomial time, decidable but NP-complete, and decidable[42]. Unfortunately, these results depend on the details of the rewrite system. Still, these results yield several interesting questions. Can we construct a homomorphic signature scheme where deciding whether a signature is revoked is undecidable? NP-complete? If we can, can such a signature scheme be black-box constructed from RSA?

7.1.4 Directed Transitive Signatures

We still do not have a construction for a directed transitive signature scheme. Devising such a construction seems to be a hard problem. We might approach the problem by attempting to construct a group with trapdoor infeasible inversion, or there might be some other approach.

7.1.5 Can we separate GII and TGII ?

We can separate one-way functions and trapdoor one-way functions because in the black box model $OTWF$s imply key agreement and OWFs do not. Both GII and TGII imply key agreement. It seems unlikely that GII implies TGII but no separation is known. Can we prove a separation or prove GII implies TGII ?

7.1.6 More Applications of Homomorphic Signatures

In the Introduction and Existing Schemes chapter, we discussed potential applications of homomorphic signature schemes. What other applications exist? Where else can signature schemes with special algebraic properties enable new applications or simplify old ones?

Bibliography

[1] Niv Ahituv, Yeheskel Lapid, and Seev Neumann. Processing encrypted data. *Communications of the ACM*, 30(9):777–780, 1987.

[2] Paulo S.L.M. Bareto. The pairing based crypto lounge. http://planeta.terra.com.br/informatica/paulobarreto/pblounge.html.

[3] M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography: the case of hashing and signing. In Yvo Desmedt, editor, *Advances in Cryptology - Crypto '94*, pages 216–233, Berlin, 1994. Springer-Verlag. Lecture Notes in Computer Science Volume 839.

[4] M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography with application to virus protection. In *FOCS 1995*, Berlin, 1995. Springer-Verlag.

[5] Mihir Bellare and Gregory Neven. Transitive Signatures based on Factoring and RSA. *ASIACRYPT 2002, Lecture Notes in Computer Science*, 2501, 2002.

[6] J Benaloh. Dense probabilistic encryption. In *Selected Areas in Cryptography*, 1994.

[7] J.C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *EUROCRYPT'93*, 1993.

[8] D. Boneh and R. J. Lipton. Algorithms for black-box fields and their application to cryptography. In Neal Koblitz, editor, *Advances in Cryptology - Crypto '96*, pages 283–297, Berlin, 1996. Springer-Verlag. Lecture Notes in Computer Science Volume 1109.

[9] Dan Boneh and Matt Franklin. Identity-based Encryption from the Weil Pairing. *CRYPTO 2001, Lecture Notes in Computer Science*, 2139:213–??, 2001.

[10] E. F. Brickell and Y. Yacobi. On privacy homomorphisms. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology - EuroCrypt '87*, pages 117–126, Berlin, 1987. Springer-Verlag. Lecture Notes in Computer Science Volume 304.

[11] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *Lecture Notes in Computer Science*, 2045:93–??, 2001.

[12] R. Canetti, O. Goldreich, and S. Halevi. The random oracle model, revisited. In *STOC '98*, pages 209–218, 1998.

[13] Suresh Chari, Tal Rabin, and Ronald Rivest. An efficient signature scheme for route aggregation.

[14] J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *26th Symposium on the Foundations of Computer Science*, 1985. http://research.microsoft.com/crypto/papers/elect.ps.

[15] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. 1992.

[16] Cramer and Damgard. Zero knowledge proofs for finite field arithmetic – or, can zero knowledge be for free? In *Advances in Cryptology - CRYPTO '98*, Berlin, 1998. Springer-Verlag.

[17] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[18] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. pages 542–552, 1991.

[19] J. Feigenbaum and Merritt. Open questions, talk abstracts, and summary of discussions. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–45, 1991.

[20] Marc Fischlin. On the impossibility of statistically zero-knowledge from any one-way function. In *RSA 2002 Conference*, 2002.

[21] E. Fujisaki, T. Okamoto, and Uchiyama. Epoc : Efficient probabilistic encryption. In *Submission to IEEE P1363*, 1998.

[22] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *FOCS 2000, Lecture Notes for Computer Science*, pages 325–335, 2000.

[23] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *FOCS 2001, Lecture Notes for Computer Science*, 2001.

[24] O. Goldreich. *Foundations of Cryptography Volume 1*. 2001.

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

[26] S. Goldwasser and Y. Tauman. On the (in)security of the fiat-shamir paradigm, 2003. http://eprint.iacr.org/2003/034/.

[27] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[28] L. A. Hemaspaandra and J. Rothe. Creating strong total commutative associative one-way functions from any one-way function. Technical Report TR688, 1998.

[29] Alejandro Hevia and Daniele Micciancio. The provable security of graph-based one-time signatures and extensions to algebraic signature schemes. In *ASIACRYPT*, pages 379–396, 2002.

[30] Martin Hirt. Personal communication, 2002.

[31] Susan R. Hohenberger. The cryptographic impact of groups with infeasible inversion. Master's thesis, MIT, May 2003.

[32] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *STOC 1989, Lecture Notes for Computer Science*, pages 44–61, 1989.

[33] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In *CT-RSA*, pages 244–262, 2002.

[34] Jeff Kahn, Michael Saks, and Cliff Smyth. A dual version of reimer's inequality and a proof of rudich's conjecture. In *15th Annual IEEE Conference on Computational Complexity CoCo'00*, 2000.

[35] Jeong Han Kim, Daniel R. Simon, and Prasad Tetali. Limits on the efficiency of one-way permutation-based hash functions. In *IEEE Symposium on Foundations of Computer Science*, pages 535–542, 1999.

[36] Menenzes, Van Oorschot, and Vanstone. *Handbook of Applied Cryptography.* 1996.

[37] R. Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, 1980.

[38] Silvio Micali and Ronald L. Rivest. Transitive signature schemes. In *CT-RSA*, pages 236–243, 2002.

[39] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *5th ACM Symposium on Computer and Communications Security*, 1998.

[40] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 4(2):151–158, ???? 1991.

[41] Goldreich Oded, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[42] F.W. Otto. A survey on the computational power of some classes of finite monoid presentations. *Algorithmic Problems in Groups and Semigroups*, 2000. http://citeseer.nj.nec.com/otto98survey.html.

[43] P Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – Eurocrypt'99*, volume 1592 of *LNCS*, 1999.

[44] Christos Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[45] R Peralta and J. Boyar. Short discreet proofs. In *Journal of Cryptology*, 2000.

[46] M. Rabi and A. Sherman. Associative one-way functions: A new paradigm for secret-key agreement and digital signatures. Technical Report CS-TR-3183/UMIACS-TR-93-124, University of Maryland, 1993.

[47] Muhammad Rabi and Alan T. Sherman. An observation on associative one-way functions in complexity theory. In *Information Processing Letters, Vol 64, Issue 5*, pages 239–244, 1997.

[48] R. Rivest, L. Adleman, and M.L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–178. Academic Press, 1978.

[49] R. L. Rivest, A. Shamir, and L. M. Adelman. A METHOD FOR OBTAINING DIGITAL SIGNATURES AND PUBLIC-KEY CRYPTOSYSTEMS. Technical Report MIT/LCS/TM-82, 1977.

[50] Ron Rivest. Two new signature schemes, 2001. Presented at Cambridge seminar; see http://www.cl.cam.ac.uk/Research/Security/seminars/2000/rivest-tss.pdf.

[51] Steve Rudich. *Limits to the Provable Consequences of One-Way Functions.* PhD thesis, UC-Berkeley, 1991.

[52] T. Sander, A Young, and M Yung. Non-interactive cryptocomputing in NC^1. In *FOCS '99*, 1999.

[53] D. Simon. Collisions on a one-way street, or, can hash functions be based on general assumptions? In *Eurocrypt '98*, 1998.

[54] Michael Sipser. *Introduction to the Theory of Computation.* John Wiley and Sons, 1998.

[55] Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content extraction signatures. In *ICISC 2001*, pages 285–304, 2002.

[56] H Straubing. When can one monoid simulate another? In *Algoritmic Problems in Groups and Semigroups*, Boston, 2000. Birkhauser. http://citeseer.nj.nec.com/144962.html.

[57] H. Zhu, B. Feng, and R. H. Deng. A transitive signature scheme provably secure against adaptive chosen-message attack, 2003. http://eprint.iacr.org/2003/059/.