The Cryptographic Impact of Groups with Infeasible Inversion

by

Susan Rae Hohenberger

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2003

Author .....
Department of Electrical Engineering and Computer Science
May 16, 2003

Certified by .....
Ronald L. Rivest
Viterbi Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

The Cryptographic Impact of Groups with Infeasible Inversion

by

Susan Rae Hohenberger

Submitted to the Department of Electrical Engineering and Computer Science
on May 16, 2003, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Algebraic group structure is an important–and often overlooked–tool for constructing and comparing cryptographic applications. Our driving example is the open problem of finding provably secure transitive signature schemes for directed graphs, proposed by Micali and Rivest [41]. A directed transitive signature scheme (DTS) allows Alice to sign a subset of edges on a directed graph in such a way that anyone can compose Alice's signatures on edges $\overrightarrow{ab}$ and $\overrightarrow{bc}$ to obtain her signature on edge $\overrightarrow{ac}$. We formalize the necessary mathematical criteria for a secure DTS scheme when the signatures can be composed in any order, showing that the edge signatures in such a scheme form a special (and powerful) mathematical group not known to exist: an Abelian trapdoor group with infeasible inversion (ATGII). Furthermore, we show that such a DTS scheme is more complex–in a black-box sense–than standard signatures, public key encryption and oblivious transfer. To our knowledge, this is the first separation between standard signature schemes and any of the many variant signature schemes proposed.

We formalize several group homomorphisms that can be used to construct undirected transitive signature schemes (UTS) (as generalizations of the UTS schemes of Micali and Rivest [41] and Bellare and Neven [7]), and explain why group isomorphisms, such as RSA, appear to require proofs in the one-more-inversion model. We also provide the first definition, to our knowledge, of a pseudo-free group. Informally, a pseudo-free group is computationally indistinguishable from a free group to any polynomially-bounded adversary given only black-box access to the group. We show that a pseudo-free ATGII group is sufficient for a secure DTS construction. We conclude by relating the black-box complexity of our group-based primitives to the standard cryptographic primitives.

Thesis Supervisor: Ronald L. Rivest
Title: Viterbi Professor of Electrical Engineering and Computer Science

Acknowledgments

I am extremely grateful to Ron Rivest for being my advisor. Ron introduced me to the problem of transitive signatures and his insights and guidance were critical throughout all stages of this research. Ron was very patient and encouraging; ready to explain or to listen. Ron was a great advisor! I also want to thank my excellent research partner David Molnar. David's grasp of the material and enthusiasm for the project made the hours and hours and hours we spent working together very enjoyable.

I am very grateful to Alantha Newman for her advice, ideas, humor and support during the countless hours of research and thesis writing. Thanks to my friends and fellow cryptographers at MIT, especially Jon Herzog, Matt Lepinski, Moses Liskov, Chris Peikert and Steve Weis, with whom I have enjoyed discussing these ideas. Salil Vadhan also provided comments that improved this thesis; most notably he pointed out an error in earlier proofs of Theorems 5.4.1 and 5.4.2.

Thanks to Erik Demaine and David Liben-Nowell for showing me the ropes of graduate level research. David's drive for perfection, elegant writing, and skill with latex are (humbly) emulated in this work. On that note, a special thank you to my proofreader Nicole Immorlica.

Thanks to Bruce Weide and Paolo Bucci at The Ohio State University. Their teaching and encouragement inspired me to go to graduate school in computer science.

I appreciate the constant support of my family and friends, especially my sisters Megan and Barbara, and my brothers-in-law Todd and Andy. This thesis is dedicated to my parents Raymond and Beth Hohenberger, and in memory to Nick Bosaw.

This research was conducted with the support of a National Defense Science and Engineering Graduate Fellowship.

Contents

List of Figures

Chapter 1

Introduction

Invented in 1976 by Diffie and Hellman, digital signatures are one of the most practical contributions of cryptography to date [21]. In a standard digital signature scheme, Alice creates a signature $\sigma_A(M)$ on a message M using a secret key that only she knows. This is analogous to the real-life situation of Alice signing her name to a document with her unique hand-writing style, although happily less prone to forgery. In a digital signature scheme, anyone can verify that Alice signed message M given her signature $\sigma_A(M)$, while no one should be able to forge her signature on *any* new message. The United States Congress legalized the use of digital signatures on contracts five years ago [20].

The best known digital signature scheme, RSA, is due to Rivest, Shamir, and Adleman [50]. In their scheme, Alice signs a message m by computing $m^d \bmod n$, where n is the product of two large primes. The exponent d is kept secret by Alice, but she publishes n and a public exponent e such that $(m^d)^e = m \bmod n$ and thus provides a method for verifying her signatures. As far as anyone knows, it is difficult to create m^d given only n and e and therefore difficult to forge Alice's signatures from scratch.

However, one weakness in RSA can be observed. Two valid signatures from Alice can be combined into one she never signed: $m_1^d m_2^d = (m_1 m_2)^d \bmod n$. For some applications, this is clearly dangerous. Suppose Alice makes two bids for an antique lamp at an auction. First, she signs a document saying she'll pay \$10 for the lamp. Someone bids higher, so Alice signs a document saying she'll pay \$15 and wins the auction. When Alice goes to claim her lamp, the auctioneer combines her two signatures and claims she agreed to pay \$150 for the lamp.

Clearly, Alice is not pleased with this special property of RSA. However, one might ask are there situations in which this algebraic property can have positive uses instead?

Rivest pointed out in a series of talks that the answer is *yes*. Signatures schemes with algebraic properties can enable new applications [49]. Suppose Alice's friend Bob just joined the United States Navy. As a young ensign, Bob reports to his superior officer, a lieutenant, who reports to a commander, who reports to a captain, who reports to an admiral, who eventually reports to the Commander-in-Chief of the United States Armed Forces, the President of the United States. Now, we can think of the Navy as a graph, with nodes as people and edges representing relationships between people. For example, a directed edge from Bob's lieutenant to Bob indicates that Bob reports to his lieutenant in the Navy's chain of command. The Secretary of the Navy is very busy and thus has only published certificates, bearing the Navy's official digital seal, on the relationships between each person and their immediate supervisor. In other words, the Secretary *signed* the edge between each person and their immediate supervisor.

Suppose Bob wishes to send a letter to Alice proving that he works for the President of the United States. Instead of paying for postage for the signatures on each of the edges (and revealing everyone on the path between Bob and the President), it would be better if Bob could send just one signature on an edge between himself and the President *that he could compute himself.* To do this, the Navy must use a "transitive signature scheme" that allows Bob (or anyone else) to create any signature in the transitive closure of those provided by the Secretary.

Transitive signature schemes were introduced by Micali and Rivest [41]. In such a scheme, a *master signer* signs the nodes and edges of a graph $G = (V, E)$. Anyone, given signatures on the edges (u, v) and (v, w) in E, can compose these signatures to obtain a valid signature for edge (u, w). In other words, given the signatures on a set of edges, anyone can obtain the signature on any edge in their transitive closure. Furthermore, anyone, given a proposed signature on an edge and the signature of its endpoints, can verify that the edge is indeed in the transitive closure of G. It remains difficult to forge any signatures outside of the transitive closure. Micali and Rivest [41] gave the first construction of a transitive signature scheme on an *undirected* graph, using a variant on commitment schemes. Later, Bellare and

Neven contributed new constructions and improved definitions for transitive signatures on undirected graphs, assuming the hardness of factoring and RSA [7]. All transitive signatures up to this time used a random oracle in the proof of their security, most commonly because they assumed a standard digital signature scheme, secure against adaptive chosen message attack, and used it to certify the public node labels. Zhu, Feng, and Deng [60] expanded the discrete log-based scheme of Micali and Rivest to cover both the node certification and the edge signing algorithm. This way, they did not need to assume anything but the hardness of taking discrete logarithms. Thus, Zhu, Feng, and Deng were able to provide the first undirected transitive signature scheme provably secure against adaptive chosen-message attack in the standard model (i.e., without random oracles).

These undirected transitive signature schemes (UTS) can be very useful. Zhu, Feng, and Deng point out an application proving trusted relationships between servers in a distributed network [59]. In this distributed network, servers are either trusted or not, and those that are trusted are linked by signatures within the transitive closure of a master signer. Fundamentally, signatures in undirected transitive signature schemes are symmetric in semantics, certifying an equality or peer relation between two nodes. Sometimes it is important that these relationships be asymmetric (i.e., the edges between nodes are directed). In our earlier example, Bob reports to his Admiral and not the other way around.

One of the main objectives of this thesis work was to propose the first *directed* transitive signature scheme (DTS). After numerous failed attempts to construct a directed transitive signature scheme out of the toolbox of standard cryptographic primitives (i.e., one-way functions, trapdoor permutations) and algebraic structures (i.e., RSA, Diffie-Helman), we started to wonder if it was even possible. We decided to take the following approach, asking ourselves two questions:

> QUESTION 1: What algebraic properties, whether they are known to exist or not, are sufficient for this cryptographic application?

The idea behind the first question is fairly straightforward. We wanted a DTS scheme and we did not know *anything* that made it possible. Thus, we invented an algebraic structure, formally defined it, and proved that it can be used to create a secure DTS scheme. (We will

also discuss in a moment the surprising result that a variant of this structure is also *necessary* for any DTS scheme.) Roughly, this structure is a *trapdoor group with infeasible inversion* (TGII) G where given a random element in G it is hard to find its inverse *unless* one holds a secret piece of information. There is also a technical requirement called *pseudo-freeness* meaning that it is hard to find any non-trivial identities in G. These topics are discussed in detail in Chapters 4 and 5.

Having formalized the necessary and sufficient properties of a directed transitive signature (DTS) scheme, we were interested in doing the same for undirected transitive signature (UTS) schemes. All previous UTS proposals are based on specific algebraic assumptions (i.e., RSA, Diffie-Hellman). We wanted to find the minimal abstraction that implies UTS (i.e., one-way permutations, trapdoor permutations)? Bellare and Neven [7] showed how to build a UTS scheme using the hardness of taking square roots modulo a composite *without* knowing the factorization. Thus, it seemed intuitively as though a black-box reduction from one-way permutations to undirected transitive signature schemes might be possible, especially given Rompel's result [51] that standard digital signatures can be built out of one-way functions. However, many distinguished researchers looked at this problem and no such proof emerged.

So we asked: what *additional* properties do RSA, taking square roots, and other specific complexity assumptions have that enable UTS and yet are not captured in the abstract notion of one-way permutations? Indeed, Bellare, Namprempre, Pointcheval, and Semanko were asking themselves the same question last year [6] (italics ours):

> We suggest that practical RSA-based schemes that have resisted attack might be manifestations of *strengths of the RSA function that have not so far been properly abstracted or formalized.* We suggest that one should build on the intuition of designers and formulate explicit computational problems that capture the above-mentioned strengths and suffice to prove the security of the scheme.

It is our belief that the *special* property, that often prevents lifting security proofs for applications from specific assumptions to abstract primitives, is that of group structure. Group structure sometimes adds power that does not exist when considering a function as simply mapping bit strings to bit strings with no algebraic structure (which is exactly what

all of our standard cryptographic primitives do). In Chapter 5, we present a vivid example of this disparity. A certain type of one-way function proposed by Rabi and Sherman [45] is provably too weak to implement key agreement (KA) when considered as mapping bit strings with no algebraic structure, and yet, we show that when such a function becomes a group operator, it easily implies key agreement. Indeed, all known UTS constructions are made from groups and we conjecture in Chapter 4 that a certain type of group homomorphism is necessary for any UTS scheme. This leads us to our second question.

> QUESTION 2: What algebraic properties are *implied* by a cryptographic application? In other words, what are its necessary conditions?

The idea behind our second question has roots in "Reverse Mathematics", where instead of asking what axioms are needed in order to prove a theorem, one instead asks which set existence axioms are implied by the truth of a theorem [55]. In the case of "Reverse Cryptography", we ask what algebraic structures are implied by an application. We are able to show that all of the algebraic properties of TGII are implied by a DTS scheme. Thus, we narrow the hunt for valid DTS schemes, a fairly complicated application, to that of first finding a group with this special property. Hopefully, TGII groups do exist and our results can motivate research in locating them. In the meantime, this result offers the best explanation to date as to why no secure DTS schemes have been proposed.

In addition, recognizing that DTS schemes imply TGII groups allows us to more easily prove a black box complexity separation between directed transitive signatures and standard digital signatures. In Chapter 5, we show that DTS schemes can not be constructed out of standard digital signatures, public key encryption, or oblivious transfer. To our knowledge, this is the first black-box separation between a regular signature scheme and any of the many signature variants. Molnar points out that union-only signatures (a scheme where computing a signature on the union of a set of messages is easy given signatures on subsets of the messages, but computing the signature of the difference of two sets is hard) forms a GII group, and thus we also separate union-only signatures from standard digital signatures [43]. It is not known if UTS schemes are strictly more complex than regular ones or not, but we make progress on this question in Chapter 4.

The main contributions of this work are highlighted in Section 1.2. Chapter 2 serves as a reference for the reader on terminology used throughout this thesis. Chapter 3 provides detailed definitions of transitive signatures, altered and extended from those of Bellare and Neven [7]. Chapter 4 is dedicated to formalizing a few groups structures of particular interest and proposing a technique for proving security in applications that use them. Chapter 5 gives the black-box relationships that exist between our new primitives and the standard cryptographic primitives in as complete a form as is known. Finally, we will conclude with several exciting open problems in Chapter 6. Appendix A contains a reference for the numerous acronyms appearing in this work.

1.1 Related Work.

An abundance of creativity and insight appearing in previous publications inspired this work. We highlight those works of particular interest below.

Transitive Signatures. Transitive signature schemes were introduced by Rivest and Micali [41]. They gave the first construction of an undirected transitive signature scheme based on the hardness of taking discrete logs and left finding a directed transitive signature scheme as an open problem. Shortly afterwards, Bellare and Neven gave alternate constructions for undirected schemes, based on the hardness of factoring and RSA [7]. These alternate constructions include the interesting property that public node labels can be derived from the hash of node indices, meaning that the signatures are secure even when an adversary is allowed to "name" the nodes. Recently, Zhu, Feng, and Deng were able to modify the discrete log based scheme of Micali and Rivest to form an undirected transitive signature scheme provably secure against adaptive chosen-message attack without random oracles [60]. We take a closer look at transitive signatures in Chapter 3.

Related Signature Work. The model for our standard digital signatures is that of Goldwasser, Micali, and Rivest [26]. We are also interested in homomorphic signatures as proposed by Johnson, Molnar, Song, and Wagner [35]. They provide definitions for homomorphic signatures and the first constructions of additive and set homomorphic signature

schemes. Chari, Rabin, and Rivest contributed a signature scheme for route aggregation, where in a binary tree anyone can produce the signature on a parent node by multiplying the signatures of its children [18]. In transitive signatures, we wish to combine two signatures to produce a *new* message under the same signer. Going in the opposite direction, the proxy signatures of Blaze and Strauss provide a protocol (with a trusted proxy) keeping the message the same while changing the signer [12]; this idea was expanded by Dodis and Ivan [22]. Very similar to this is the notion of delegateable signatures. First proposed by Rivest, these signatures would allow Alice to sign a message *delegating* ability to Bob to efficiently sign a subset of messages on her behalf. Although, ideal delegateable signatures remain an open problem, Barak made progress using techniques from zero-knowledge proofs [3].

Reverse Mathematics. Reverse mathematics is a notion introduced into set theory by Harvey Friedman, where instead of deducing a theorem from some axioms (i.e., "forward mathematics"), one instead asks which axioms are implied by the truth of a theorem [55]. In the case of reverse mathematics, the focus is generally on set existence axioms. We are interested in a "reverse cryptography" of sorts, where instead of building an application from some primitives, we instead ask which primitives are implied by the existence of an application.

Multilinear Forms. Multilinear forms provide a special mapping from one group to another and thus are a good example of looking at applications over groups as opposed to a set of elements without structure. Since multilinear forms were introduced into cryptography [36], they have been useful in their binary form to construct secure identity-based encryption schemes [15] and secure aggregate signature schemes [16]. Happily there exist bilinear forms, with a reasonable assumed security, that can be found using elliptic curves [33]. Boneh and Silverberg recently motivated the search for realizable multilinear forms by a laundry list of cryptographic problems that they can solve [14]. Further interesting information on multilinear forms can be found online at the Pairing Based Crypto Lounge [4], including their relation to the classic Diffie-Hellman problem [19]. In our initial search for directed transitive signature schemes, multilinear forms appeared very promising; although we were never able to realize a *full* DTS scheme using them, we were able to use them to construct

a weakened DTS scheme where only *original* edges can be composed.

Associative One-Way Functions. Our research was successful in finding a link between strongly associative one-way functions, introduced by Rabi and Sherman [45], and one of our new primitives: *groups with infeasible inversion*. In a series of papers [31, 28, 46], these functions were shown to exist if and only if one-way functions exist. Thus, they are generally believed to exist, although no concrete possible implementations are known.

Black-Box Constructions and Oracle Separations. A sizable portion of our contribution comes from organizing and discovering the black-box relationship among our new group-based primitives, transitive signature schemes, and the standard cryptographic primitives. There is a large amount of literature for us to bring together. In Chapter 5, we present a diagram illustrating the known black-box relationships among cryptography primitives. The best up-to-date summary that we found for black-box relationships was by Gertner, Kannan, Malkin, Reingold, and Viswanathan, in which their main result is that public key encryption and oblivious transfer are *incomparable* in a black-box setting [23]. Gertner, Malkin, and Reingold were also able to show that trapdoor functions can not be constructed from trapdoor predicates (i.e., public key encryption) [24]. We also used Selman's survey of one-way functions [51], as well as several references on the relation between one-way functions and one-way permutations [10, 52], including a very interesting paper by Hemaspaandra, Pasanen, and Rothe that proves the seemingly contradictory statement that strongly associative one-way functions are not always one-way functions [29]. Finally, we spent a large amount of time trying to construct a proof that one-way permutations are insufficient for undirected transitive signature schemes. We based our ideas off of the pioneering work of Impagliazzo and Rudich, who were first able to prove that one-way permutations are insufficient for key agreement by showing that in a world where P=NP there is an oracle relative to which one-way permutations exist and key agreement does not [32, 53]. Bellare and Rogaway's work on highlighting the difference between random oracles in proof and practice was also useful [9].

Braid Groups. In our quest to resolve the open problem of directed transitive signatures, we hunted in many new areas of promise in cryptography. One area that did not work

out, but offered several interesting ideas, is that of braid groups. A large listing of relevant braid group papers are available at the Braid Group Cryptography website [39]. We took particular interest in a recently proposed signature scheme using braids by Ko, Choi, Cho, and Lee that took advantage of a gap where the conjugacy search problem is hard and the conjugacy decision problem is easy [37].

One-More-Inversion Security. The proof that one-way group isomorphisms (OWGI) are sufficient for undirected transitive signature schemes in Section 4.4 is in the one-more-inversion security model. Bellare, Namprepre, Pointcheval, and Semanko introduced one-more-RSA-inversion security and used it to provide the first security proof, in any model, of Chaum's Blind Signature Scheme [6]. Bellare and Neven prove their RSA-based transitive signature scheme one-more-RSA-inversion model [7]. Recently, Bellare and Palacio also use it to show security from impersonation of the GQ and Schnorr Identification Schemes [8]. Section 4.4.1 contains a definition and discussion of one-more-inversion security.

Groups with Infeasible Inversion (GII). We are not aware of the existence of any groups with infeasible inversion, or of any unproven constructions that appear to satisfy its properties. One area we have explored are groups where the computational Diffie-Helman problem was easy, but computational inversion in the exponent remained hard. However, we discovered that such a group would only be possible when the order of the group remains secret [54]. We also looked at the possibility of building a GII group out of a suitable subgroup using Straubing's work as a reference [56]. One (unrealized) proposal for a GII group was introduced by Johnson et al. [35], although the notion of GII was not formalized at that time. During joint work with this thesis in formalizing GII, Molnar first brought the two notions together. Informally, it is a homomorphic set signature scheme in which the union operation is easy, while taking set difference is difficult. For further details, we refer the reader to his thesis [43].

1.2 Contributions of This Thesis and Statement on Joint Work

All contributions included in this thesis represent joint work with Ron Rivest and David Molnar. Rivest contributed to all parts of this thesis, especially identifying and formalizing the *pseudo-free* groups in Section 4.5, and he generously allowed me to use his proofs establishing the relationships between weakly collision-resistant and one-way non-injective group homomorphisms in Claims 4.3.1, 4.3.2, and 4.3.3. Molnar was also active in this research and parts of this work also appear in his thesis [43].

1. SIGNATURE SEPARATIONS - To our knowledge, we provide the first separation of a homomorphic signature scheme (DTS) from a standard digital signature scheme (SDS). In fact, we show that the directed transitive signatures proposed by Micali and Rivest [41] can not be constructed in a fully black-box fashion using either public key encryption (PKE) or oblivious transfer (OT).

2. TRANSITIVE SIGNATURES - We do *not* answer the open problem proposed by Micali and Rivest as to whether or not directed transitive signatures exist [41]. However, we do provide the necessary (Abelian TGII) and sufficient (pseudo-free Abelian TGII) algebraic structures required for a DTS scheme. These structures are not known to exist, but are provably more complex, in a fully black-box sense, than either public key encryption (PKE) or oblivious transfer (OT), and at least as complex as trapdoor permutations (TDP) – the most complex primitive usually considered in cryptography.

   For undirected transitive signatures, we provide the first formalization of the sufficient conditions for a UTS scheme. We also offer explanation as to why discrete logarithm and square root based UTS schemes have standard security proofs, while RSA-based schemes elude them. In Chapter 4, we discuss the surprising result that for homomorphic signature schemes non-injective trapdoor functions may be more powerful than trapdoor permutations (acting on a group).

3. REVERSE CRYPTOGRAPHY - Typically, cryptographers focus on building (or breaking) secure applications from a set of abstract primitives or trusted algebraic structures.

We introduce an idea from set theory into cryptography, called *Reverse Cryptography*, by asking what primitives or algebraic structures are implied by the existence of an application. We provide an example of reverse cryptography in Section 5.5 by showing that the edge signatures in a DTS scheme form a special (and powerful) mathematical group.

4. GROUP-BASED PRIMITIVES - We also focus on what provable complexity gains one can find when considering a standard cryptographic primitive (i.e., one-way function) as operating on a group. Homomorphic schemes often imply group structure, while our standard primitives do not consider it. In 1993, Rabi and Sherman proposed a key agreement protocol based on strongly associative one-way functions (SAOWF), but were unable to prove its security outside of Yao's [57, 58] model of computational information theory [45]. We provide the first standard proof of security for their key agreement protocol based on the assumption of a group, where the operator is a SAOWF.

   We formalize several new group-based primitives including: *groups with infeasible inversion* (GII), *one-way group isomorphisms* (OWGI) and *weakly collision-resistant one-way non-injective group homomorphisms* (WCRNIGH). We relate their black-box complexity to transitive signatures and some of the standard cryptographic primitives. Their relation to homomorphic encryption schemes remains an exciting open problem.

5. PSEUDO-FREE GROUPS - We introduce the notion of *pseudo-free* groups in Section 4.5, and show that a pseudo-free Abelian TGII group is sufficient for a black-box DTS construction. Informally, a pseudo-free group G is computationally indistinguishable from a free group to any polynomially-bounded adversary, given black-box access to G. We discuss possible examples of pseudo-free groups in Section 4.5.

6. BLACK-BOX RELATIONSHIPS - We contribute, in a humble way, to organizing the black-box relaion between primitives. We build on a summary in the work of Gertner, Kannan, Malkin, Reingold, and Viswanathan [23] for a (hopefully) clean illustration of the known relationships in Figure 5-1. Understanding these entangling relationships may save other researchers from attempting constructions with insufficient primitives. For example, we tried many black-box constructions of DTS schemes from SAOWF

until we realized that Hemaspaandra and Rothe proved that SAOWF were equivalent in complexity to regular OWFs [28], and thus were unlikely to help us.

Chapter 2

Preliminaries

In this chapter, we formally define a number of standard notions that will be used throughout this work. It is primarily meant as a reference for commonly used terms. Readers familiar with the cryptographic literature may wish to skip to Chapter 3 and refer back to this chapter as needed. Let $\mathbb{N}$ be the set of positive integers and let $||$ denote concatenation (i.e., $1||b = 1b$).

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 $\overrightarrow{ij} \in \tilde{E}$ if and only if there is a directed path from i to j in G.

Probabilistic Polynomial Time (PPT). Given $k \in \mathbb{N}$, we say that an algorithm runs in *probabilistic polynomial time* with respect to k if it is able to flip coins and its expected running time is polynomially-bounded in k.

Negligible. We say that an event α has a *negligible* chance of occuring with respect to a parameter k, if the probability that α occurs decreases faster than any polynomial in k (i.e., $< 1/poly(k)$) as the value k increases.

Collision-Resistant (CR). Let $h : \{0,1\}^l \to \{0,1\}^n$ be a polynomially-computable function where $l > n$. We say that h is *collision-resistant* if it is *hard* for any PPT adversary to find any two distinct inputs x and x' such that $h(x) = h(x')$. We know such x, x' pairs must exist

by the pigeonhole principle; yet for every PPT algorithm A, every positive polynomial $p(\cdot)$, and all sufficiently large l's,

$$Pr[x \stackrel{R}{\leftarrow} \{0,1\}^l, x' \leftarrow A(h(x), 1^l) : h(x) = h(x')] < \frac{1}{p(l)}.$$

Weak collision-resistance (WCR) is also known as second pre-image resistance. Given an input x, it is hard for any PPT adversary to find an $x' \neq x$ such that $h(x) = h(x')$. Here the adversary must find a collision for a *particular* input rather than on any input. If h is collision-resistant, then h is also weakly collision-resistant.

One-Way Function (OWF). We give the same definition as Goldreich [25]. A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is called *one-way* if: (1) [Easy to compute] there exists a deterministic polynomial-time algorithm A such that on input x algorithm A outputs $f(x)$, and (2) [Hard to invert] for every PPT algorithm A, every positive polynomial $p(\cdot)$, and all sufficiently large k's,

$$Pr[x \stackrel{R}{\leftarrow} \{0,1\}^k, x' \leftarrow A(f(x), 1^k) : f(x) = f(x')] < \frac{1}{p(k)}.$$

For $\mathbb{S} \subseteq \{0,1\}^k$, function $p : \mathbb{S} \rightarrow \mathbb{S}$ is called a **one-way permutation** (OWP) if it has properties (1) and (2), as well as (3) [One-to-One] for every $y \in \mathbb{S}$, there exists a unique pre-image $x \in \mathbb{S}$ such that $p(x) = y$.

The function f (or p) is said to be *trapdoor* if all the function's previous properties hold, except that it becomes polynomial-time computable to invert f (or p) when a secret trapdoor $t \in \{0,1\}^{poly(k)}$ is known to A'. Property (2) holds otherwise. In later sections, we will discuss **trapdoor permutations** (TDP).

Standard Digital Signature (SDS). In several transitive signature constructions, we will use an underlying standard digital signature scheme SDS=(SKG, SSign, SVf), described as usual via its polynomial time key generation (SKG), signing (SSign), and verification (SVf) algorithms. We use the security definition of Goldwasser, Micali, and Rivest [26], where the forger is given adaptive oracle access to the signing algorithm, meaning the forger can choose the next query based on the oracle's answer to the previous one. The scheme SDS is said to be secure against forgery under adaptive chosen-message attack if the probability of

successfully forging a signature on a message not asked of the oracle is negligible for every forger B with running time polynomial in the security parameter.

Pseudorandom Generator (PRG). We give the definition of Johan Hastad, Russell Impagliazzo, Leonid A. Levin and Michael Luby, who proved that a *pseudorandom generator* can be constructed from any one-way function [27]. Let $g : \{0,1\}^{t_n} \rightarrow \{0,1\}^{l_n}$ be a polynomially-computable function where $l_n > t_n$. Let U_k denote a truly random distribution on bit strings of length k. Then, g is a pseudorandom generator if the probability ensembles $g(U_{t_n})$ and U_{l_n} are computationally indistinguishable.

Key Agreement (KA). We give (nearly) the same definition as Impagliazzo and Rudich [32]. A *key agreement protocol* is a pair of probabilistic polynomial time algorithms called Alice and Bob. A run of the protocol is as follows: Alice and Bob both start with input 1^k. They may send a polynomial number of messages to one another over a public channel; the set of their messages is called a conversation. At the end, they must both compute and agree on a secret key K with probability 1. A probabilistic polynomial-time algorithm Eve *breaks* the key agreement protocol if Eve, given only the conversation and 1^k as input, can guess K with probability better than $1/p(k)$, for some polynomial p. A protocol is called *secure* if no such Eve exists.

Public Key Encryption (PKE). Generally speaking, a public key encryption system is a way in which Alice and Bob can securely send messages to each other over an insecure channel. Each user obtains a unique public and private key using a key generation algorithm. When Alice wishes to send a message to Bob, she encrypts the message using Bob's public key. On receiving the encryption, Bob decrypts the message using his private key. PKE is also referred to as *asymmetric encryption.*

Oblivious Transfer (OT). Oblivious transfer is an amazing protocol involving two probabilistic polynomial time algorithms: a Sender and a Receiver. At the start of the protocol, the Sender knows m secrets and the Receiver wishes to know $n \leq m$ of them. They may send a polynomial number of messages to one another over a public channel. At the end, the Receiver learns all n secrets he requested, yet does not gain any knowledge about the

Sender's additional secrets. At the same time, the Sender gains no knowledge about which secrets the Receiver learned. The concept was first introduced in 1981 by Michael Rabin, who demonstrated that oblivious transfer can be used to obtain key agreement [47].

Secure Function Evaluation (SFE). Intuitively, a protocol P securely evaluates function f, given input from n parties, if there does not exist an adversary against P who can *learn more* than an adversary in a setting where a trusted party computes f given input privately from the same n parties. Oblivious transfer is an example of secure function evaluation. The first definition for secure computation was given by Micali and Rogaway in 1991 [42].

Multilinear Forms (ML). Boneh and Silverberg introduced the concept of multilinear forms on groups for cryptography and gave motivating applications[14]. We say a map $e : G_1^n \rightarrow G_2$ is an *n-multilinear map* if it satisfies the following properties:

(1) G_1 and G_2 are groups of the same prime order.

(2) If $\alpha_1, \ldots, \alpha_n \in Z$ and $x_1, \ldots, x_n \subset G_1$, then $e(x_1^{\alpha_1}, \ldots, x_n^{\alpha_n}) = e(x_1, \ldots, x_n)^{\alpha_1 \cdots \alpha_n}$.

(3) The map e is non-degenerate in the following sense: if $g \in G_1$ is a generator of G_1, then $e(g, \ldots, g)$ is a generator of G_2.

(4) Computing discrete logarithms in G_1 is hard.

For $n = 2$, we say e is a **bilinear map** (BL). Currently multilinear forms are only known to exist for $n \leq 2$ [4].

Semi Black-Box Reduction. We give the definition of Gertner et al.[23]. A *semi black-box reduction* of a primitive P to a primitive Q is a construction of P out of Q that ignores the internal structure of Q (both in the construction itself and in its proof of correctness.) More formally, this is a construction of a polynomial time oracle machine M such that M^C implements the primitive P whenever C implements the primitive Q. Namely, Q is given in the construction as a black-box rather than, say, as the description of an algorithm that implements it. In addition, the proof of correctness is also limited to black-box access to Q. This means that for any polynomial-time oracle machine A_P, there exists a polynomial-time oracle machine A_Q such that, if A_P with access to C breaks M^C (as an implementation of P), then A_Q with access to C breaks C itself (as an implementation of Q). Note that A_Q may use the internal structure of A_P (only C is given as a black-box).

Fully Black-Box Reduction. We give the definition of Gertner et al.[23]. A *fully black-box reduction* between P and Q is a semi black-box reduction, where in addition the proof is black-box in a stronger sense: There is a polynomial-time adversary A_Q which, given *oracle access* to any adversary A_P that breaks M^C (as an implementation of P), manages to break C (as an implementation of Q).

Oracle Separation. We give the definition of Gertner et al. [23], originated by Impagliazzo and Rudich[32]. Let P, Q be two cryptographic primitives. To separate P and Q with respect to black-box reductions, it is enough to construct an oracle O such that relative to O the primitive Q exists whereas P does not.

Chapter 3

Transitive Signatures

Transitive signature schemes were introduced by Rivest and Micali [41]. In such a scheme, a *master signer* signs the nodes and edges of a graph $G = (V, E)$. Anyone, given signatures on the edges (u, v) and (v, w) in E, can compose these signatures to obtain a valid signature for edge (u, w) that can be verified by public information and is indistinguishable from a signature that the master signer would produce for edge (u, w). In other words, given the signatures on a set of edges, anyone can obtain and verify the signature on any edge in their transitive closure.

If we consider a master signer's time as an expensive resource, then a transitive signature scheme offers two main benefits to its users over a standard signature scheme: space and privacy. Recall our example from Chapter 1, where Bob recently joined with US Navy as a young ensign and wants to prove to his friend Alice that he works for the President of the United States. The Secretary of the Navy signs a certificate validating a reporting relationship between each Naval member and his (or her) immediate superior officer, but the Secretary does not have time to sign anything else. Why should the Navy use a transitive signature scheme?

First, a transitive signature scheme saves *space*, by allowing Bob to compute a short signature containing the Secretary of the Navy's certification that Bob reports to the President. If the Navy uses a standard signature scheme instead, then Bob's only option is to pay the postage to ship *all* signatures on the path between himself and the President to Alice. If Bob (and other ensigns) wish to prove to many friends that they work for the President of

the United States, then this becomes an expensive operation.

Secondly, a transitive signature scheme maintains *privacy*. There is no need for Alice to know who Bob's captain's superior officer is. In fact, the Navy may wish to keep this information confidential. Transitive signatures allow Bob to produce a signature between himself and *any* of his superior officers *without* exposing the intermediate officers in the chain-of-command.

3.1 Definitions with Commutativity

In the first paper on transitive signatures, Micali and Rivest gave the intuition for both directed and undirected transitive signatures [41]. Shortly afterwards, Bellare and Neven gave the first formal definition for undirected transitive signatures [7]. In this section, we extend the UTS definition of Bellare and Neven and provide the first formal definition of directed transitive signatures (DTS).[1] All current UTS proposals remain valid under our extended definition. (We will demonstrate this in Sections 3.2.1 and 3.2.2).

Our primary extension is the requirement that edge signatures can be composed *in any order*. Formally, edge signatures must commute. Suppose Bob carries a smart card with him, so full of nautical charts, that he only has room to store one signature. Every edge in the path from Bob to the President is signed by the Secretary of the Navy, except the link between his lieutenant and his commander. It would be great if Bob could combine all of the available signatures into a short signature to store on his smart card today and later compose with the missing link (once it becomes available) to form a signature from Bob to the President. Since edge signatures *do* commute in all known UTS schemes and we've identified this as a useful property, we feel our commutative definition captures a fuller intuition of what an ideal directed transitive signature scheme should be. Moreover, it enables us to prove the first strong assertion about the relative complexity of these signature schemes. We discuss possible relaxations of this definition in Section 6.1.

Undirected Transitive Signatures (UTS). Exactly like the DTS definition below with

[1] An earlier version of the directed transitive signature definition appears in Molnar's thesis [43], and was joint work with Molnar and Rivest.

an undirected edge represented by directed edges in both directions.

Directed Transitive Signatures (DTS). Let $\mathbb{N}$ be the set of positive integers. Our definition of transitive signature schemes extends that of Bellare and Neven [7] 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 public and secret information; this information is implicitly encoded into the choice of graph index, an inherently *public* label, in the Bellare-Neven definition. (Concurrently with our work, Zhu, Feng, and Deng introduced an altered undirected transitive signature definition with an explicit NCert algorithm [60]; a good sign that we are capturing the correct intuition.) The algorithm NCert allows us to talk more easily about a transitive signature scheme to handle both "dynamic" and "static" transitive signature schemes, where a scheme may or may not have to sign nodes added after the master keys have been generated. Third, extending the intuition of "static" schemes, we define "transitive signature schemes with pre-processing" where key generation may depend on the graph G (in the original definition key generation was always independent of the graph). This notion may be of independent interest for efficiency and simplicity issues in settings where changes are scarce. Finally, we broaden the notion of composed signatures to allow for the composition of signatures in an arbitrary order (commutativity of composition). Suppose signatures for edges $\overrightarrow{ab}, \overrightarrow{bc}, \overrightarrow{de}$ are available. Bob wishes to create the signature $\sigma_{\overrightarrow{ae}}$. It is desirable that he can compose $\sigma_{\overrightarrow{ab}} \circ \sigma_{\overrightarrow{bc}} \circ \sigma_{\overrightarrow{de}} = \tau$, and at a later time, when edge signature $\sigma_{\overrightarrow{cd}}$ becomes available, compute $\tau \circ \sigma_{\overrightarrow{cd}} = \sigma_{\overrightarrow{ae}}$. All previously suggested UTS constructions [41, 7, 60] remain valid under this natural, expanded definition. Whether or not a UTS (or DTS) scheme can be created with all properties, except commutativity of composition, is an open problem.

A (*commutative*) *directed transitive signature scheme* DTS=(KG, NCert, ESign, Vf, Comp) is specified by five polynomial-time algorithms as follows:

- KG: The randomized *key generation* algorithm KG can take as input *either* 1^k, where $k \in \mathbb{N}$ is the security parameter, or take 1^k and a graph G bounded in size by a polynomial in k, 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 $\varepsilon(k)$-*heavy per-key pre-processing* if the running time of KG on input 1^k and graph G is bounded by $\varepsilon(k)$. In the case that $\varepsilon(k)$ is polynomial, we say that the DTS requires *efficient* per-key preprocessing. Unless otherwise stated, we refer to dynamic DTS schemes.

- NCert: The *node certification* algorithm NCert, which is stateful and could be randomized, takes as input the master secret key SK and a node $i \in \mathbb{N}$, and returns a unique pair $(L(i), \ell(i))$ consisting of a public value and a secret value for node i. It records the public and secret information for node i in state variables L and ℓ, respectively, and updates any other state variables upon each invocation.
- ESign: The *edge signing* algorithm ESign, which could be stateful or randomized (or both), has read only access to the public and secret node information (i.e., L, ℓ) maintained by 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.
- Vf: The deterministic *verification* algorithm Vf, given PK, two multisets of node public values P_{src}, $P_{dest} \in L$, and a candidate signature σ, returns either 1 or 0. It returns 1 if and only if σ is a *valid* signature of a set of directed edges $E_v = \{e_1, e_2, \ldots, e_m\}$ such that P_{src} is the multiset of public values for source nodes in E_v and P_{dest} is the analogous multiset for destination nodes. Thus, $|P_{src}| = |P_{dest}|$. The validity of an edge is taken relative to the public information PK, P_{src}, and P_{dest}. For example, on input (PK,$\{L(i)\}, \{L(j)\}, \sigma$), Vf returns 1 if and only if σ is a valid signature on edge $\overrightarrow{ij}$. By this representation, sources are not paired with destinations; thus the edge set signed might be ambiguous, unless $|P_{src}| = |P_{dest}| = 1$. This unusual property allows for the verification of all signatures produced by the master signer or the Comp algorithm.
- Comp: The deterministic *composition* algorithm Comp takes as input PK, two pairs of multisets of node public values $(P1_{src}, P1_{dest})$, $(P2_{src}, P2_{dest})$, and values σ_1, σ_2. Let

$\Lambda = P1_{src} \cup P2_{src}$ and $\Upsilon = P1_{dest} \cup P2_{dest}$. Then Comp returns either a value σ_3, called a *composed signature* on edge sets corresponding to $(\Lambda - (\Lambda \cap \Upsilon), \Upsilon - (\Upsilon \cap \Lambda))$, or the symbol $\perp$ to indicate failure. For example, on input (PK,$(\{L(i)\}, \{L(j)\}), (\{L(j)\}, \{L(k)\})$, $\sigma_{\overrightarrow{ij}}$, $\sigma_{\overrightarrow{jk}}$), Comp returns $\sigma_{\overrightarrow{ik}}$, a composed edge signature for $(\{L(i)\}, \{L(k)\})$ (i.e.the signature for edge $\overrightarrow{ik}$). A signature with $|P_{src}| > 1$ is also called a *waiting signature*, since it will need additional compositions before it can authenticate a single, explicit edge.

For a directed transitive signature scheme DTS to be useful for the applications mentioned 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, $\{L(i)\}$, and $\{L(j)\}$. Similarly, the output of the Comp algorithm must always be a valid signature, when given "correct" inputs: two original or composed signatures and their corresponding public node values.

We agree with Bellare-Neven [7] that it does not always follow that simply because two signatures are valid (i.e., they pass the Vf algorithm), they should be composable. Consider the case where the public value $L(j)$ for node j is a square A and its secret value $\ell(j)$ is a specific square root of A (modulo the product of two large primes). Now, it might be possible that an edge using *another* square root of A might appear valid and yet not compose well with other 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, we require that a composed signature on edge $\overrightarrow{ik}$ be *computationally indistinguishable* from an original signature on edge $\overrightarrow{ik}$. This requirement offers two benefits. First, we see that the length of a string of directed edge signatures (i.e., $\sigma_{\overrightarrow{ab}}, \sigma_{\overrightarrow{bc}}, \sigma_{\overrightarrow{cd}}$), once composed, is no greater than the single signature $\sigma_{\overrightarrow{ad}}$. Secondly, the privacy of all intermediate nodes are maintained. The composed edge signature $\sigma_{\overrightarrow{ad}}$ above gives no information about its intermediate nodes b, c, since the signature is indistinguishable from that of the ESign algorithm on edge $\overrightarrow{ad}$, which received neither $L(b)$ nor $L(c)$ as input.

In Figure 3-1, we give an experiment for any (deterministic, halting, computationally unbounded) algorithm A and any $k \in \mathbb{N}$, which gives A oracle access to

1) (PK,SK) $\stackrel{R}{\leftarrow}$ 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 $(L(i), \ell(i)) \leftarrow$ 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 let σ be the output of the ESign oracle and $S \leftarrow S \cup \{(\{L(i)\}, \{L(j)\}, \sigma)\}$
10) If Vf(PK,$\{L(i)\}$,$\{L(j)\}$,σ)=0 then NotOK $\leftarrow$ true
11) If A queries Comp on $(P1_{src}, P1_{dest}), (P2_{src}, P2_{dest}), \sigma_1, \sigma_2$ then
12) If $(P1_{src}, P1_{dest}, \sigma_1) \notin S \vee (P2_{src}, P2_{dest}, \sigma_2) \notin S$ then Legit $\leftarrow$ false
13) Else let $\Lambda \leftarrow P1_{src} \cup P2_{src}$, $\Upsilon \leftarrow P1_{dest} \cup P2_{dest}$, $\Gamma \leftarrow \Lambda \cap \Upsilon$.
14) Let σ be the output of the Comp oracle, $S \leftarrow S \cup \{(\Lambda - \Gamma, \Upsilon - \Gamma, \sigma)\}$.
15) If Vf(PK,P_{src}, P_{dest},σ)=0 then NotOK $\leftarrow$ true
16) When A halts, output (Legit $\wedge$ NotOK) and halt.

Figure 3-1: An experiment to define the correctness of a dynamic directed transitive signature scheme DTS=(KG, NCert, ESign, Vf, Comp).

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

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 3.1.1 *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 3-1 is* true *with probability zero.*

We call this case *dynamic*, because A may add new nodes to the graph, via the NCert oracle, both before and after calls to ESign and Comp. As A queries, 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 be completed before the master keys are generated (and therefore, before any calls to ESign or Comp). In

this case, we run a new experiment with A and k, changing Figure 3-1 to move lines 2, 4, 5 and 6 to appear sequentially above the current line 1, which itself is replaced by "(PK,SK) $\stackrel{R}{\leftarrow}$ KG$(1^k, G)$", where G is the complete graph of nodes created during calls to NCert. The rest of the experiment is consistent with the dynamic case.

Definition 3.1.2 *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 3-1, with static modifications, is* true *with probability zero.*

SECURITY OF DTS SCHEMES. Forgeries are valid signatures on a single edge not within the transitive closure of G. Fundamentally, DTS schemes provide authentication of single edges (the relationship between two parties). The ability to "forge" waiting signatures only gives the adversary power when she is able to use them to compute an edge signature outside of the transitive closure of G, since those are the only signatures that directly (and falsely) authenticate a directed edge between two nodes.

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 PK and oracle access to the function ESign(SK,$\cdot,\cdot$). The adversary for dynamic schemes can also make oracle calls to NCert to create nodes, while in static schemes she can not. (For this reason, dynamic security is strictly stronger than static security.) 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 ESign 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, $L(i')$, and $L(j')$, 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 attack* if the function $\mathbf{Adv}^{dtu-cma}_{DTS,F}(k)$ is negligible for any PPT adversary F whose running time is polynomial in the security parameter k.

3.2 Undirected Transitive Signatures

Since examples sometimes illuminate concepts better than formal definitions, we include two of the UTS schemes proposed to date. We will be highlighting some of their special properties in Chapter 4.

3.2.1 Discrete Log-Based UTS Scheme of Micali-Rivest

The first transitive signature scheme was given by Micali and Rivest [41]. The scheme uses homomorphic commitments of the form $g^x h^y$ in a cyclic group and assumes 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 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. The algorithm publishes PK=(g, h, p, q, spk) and gives SK=(ssk) to the master signer.

NODE CERTIFICATION. The NCert algorithm maintains state (N, ℓ, L, Σ) and takes as input SK and a node index i from the master signer.

If the node is not already signed (i.e., $i \notin N$), then it records the index as signed ($N \leftarrow N \cup \{i\}$). Next, it selects two values x_i and y_i independently at random from Z_q and records these as the secret information for node i: $\ell(i) = (x_i, y_i)$. NCert computes and publishes the public information for node i: $L(i) = g^{x_i} h^{y_i} \bmod p$, and signs the public

information with the SDS: $\Sigma(i)$ =SSign$(ssk, i||L(i))$.

EDGE SIGNING. The ESign algorithm takes two node indices i, j and SK as input. If $(i \neq j)$ and $\Sigma(i), \Sigma(j)$ verify with respect to spk, then it returns $\sigma_{ij} = (i, j, \alpha_{ij}, \beta_{ij})$, where:

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

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:

$$L(i) = L(j) g^{\alpha_{ij}} h^{\beta_{ij}} \bmod q.$$

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 σ_{ik}. Given $\sigma_1 = (i, j, \alpha_{ij}, \beta_{ij})$ and $\sigma_2 = (j, k, \alpha_{jk}, \beta_{jk})$ the algorithm computes:

$$\alpha_{ik} = \alpha_{ij} - \alpha_{jk} = x_i - x_k \pmod q$$
$$\beta_{ik} = \beta_{ij} - \beta_{jk} = y_i - y_k \pmod q$$

and outputs $(\alpha_{ik}, \beta_{ik})$.

Since addition modulo a prime is commutative, the above UTS also satisfies our expanded definition in Section 3.1. Observe in Figure 3-2 that σ_{ac} could also be computed by adding the signatures in reverse order: $\sigma_{bc} + \sigma_{ab}$. Given σ_{rs} and σ_{tu}, a user can combine them into a shorter waiting signature $\sigma_w = (x_r - x_s + x_t - x_u, y_r - y_s + y_t + y_u)$ (mod N) and *wait* until $\sigma_{st} = (x_s - x_t, y_s - y_t)$ becomes available to compute $\sigma_{ru} = \sigma_w + \sigma_{st}$.

For proof of security, we defer to the original paper [41]. 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.

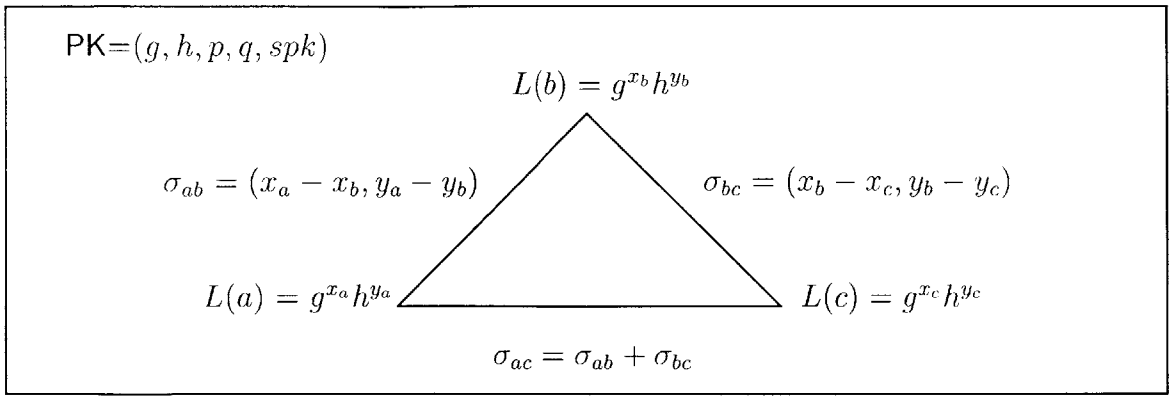

Figure 3-2: Illustration of DL-Based UTS, where $L(a), L(b), L(c)$ are public node labels, σ_{ab} and σ_{bc} are edge signatures from the master signer, and σ_{ac} is a composed signature.

Addendum: Since Micali and Rivest published the above, Zhu, Feng and Deng expanded the scheme to give an explicit SDS scheme, also based on the discrete log assumption, and were able to show their UTS scheme is transitively secure against adaptive chosen message attack without using random oracles [60].

3.2.2 RSA-Based UTS Scheme of Micali-Rivest, Bellare-Neven

Micali and Rivest mentioned the following UTS scheme, but were only able to prove its security against a static forger, meaning that the forger must commit to all of his oracle queries before seeing the responses to any of them [41]. Bellare and Neven provided a proof of this scheme against an adaptive adversary assuming that the one-more-RSA-inversion problem is hard; they named and formalized it RSATS-1 [7]. We discuss this security model in detail in Section 4.4.

RSATS-1 includes an RSA *key generator* RG as a PPT algorithm that on input 1^k, outputs a tuple (N, e, d) where $2^{k-1} \leq N < 2^k$ and $ed \equiv 1 \bmod \varphi(N)$. It also assumes 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 calls RG on 1^k to generate an RSA tuple (N, e, d). The algorithm publishes PK=(N, e, spk) and gives SK=(N, e, d, ssk) to the master signer.

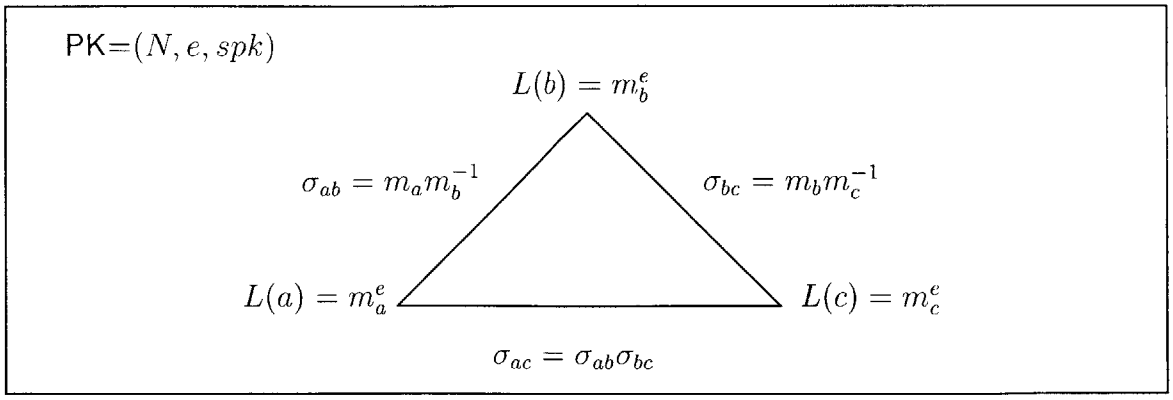

Figure 3-3: Illustration of RSA-Based UTS, where $L(a), L(b), L(c)$ are public node labels, σ_{ab} and σ_{bc} are edge signatures from the master signer, and σ_{ac} is a composed signature. All values are taken modulo N.

NODE CERTIFICATION. The NCert algorithm maintains state (N, ℓ, L, Σ) and takes as input SK and a node index i from the master signer.

If the node is not already signed (i.e., $i \notin N$), then it records the index as signed ($N \leftarrow N \cup \{i\}$). Next, it selects a value at random from $r \stackrel{R}{\leftarrow} \mathbb{Z}_N^*$ and records it as the secret information for node i: $\ell(i) = r$. NCert computes and publishes the public information for node i: $L(i) = \ell(i)^e \bmod N$, and signs the public information with the SDS: $\Sigma(i) = \text{SSign}(ssk, i||L(i))$.

EDGE SIGNING. The ESign algorithm takes two node indices i, j and SK as input. If $(i \neq j)$ and $\Sigma(i), \Sigma(j)$ verify with respect to spk, then it returns $\sigma_{ij} = \ell(i)\ell(j)^{-1} \bmod N$; 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 is valid, the algorithm checks:

$$L(i)L(j)^{-1} = \sigma^e \bmod N.$$

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 σ_{ik}. The algorithm outputs $\sigma_1\sigma_2 \bmod N$.

Since multiplication modulo a prime is commutative, the above UTS also satisfies our expanded definition in Section 3.1. Observe in Figure 3-3 that σ_{ac} could also be computed

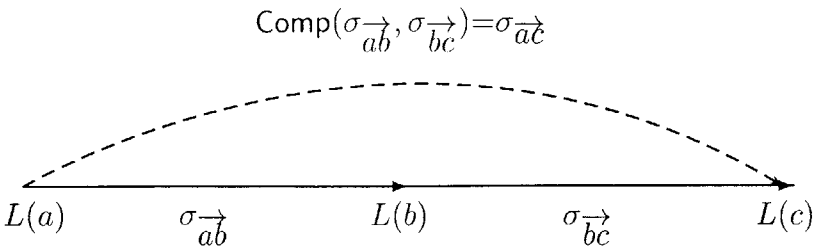

Figure 3-4: Illustration of a DTS scheme, where $L(a), L(b), L(c)$ are public node labels and $\sigma_{\overrightarrow{ab}}, \sigma_{\overrightarrow{bc}}$ are directed edge signatures from the master signer.

by multiplying the signatures in reverse order: $\sigma_{bc}\sigma_{ab}$. Given σ_{rs} and σ_{tu}, a user can combine them into a shorter waiting signature $\sigma_w = m_r m_s^{-1} m_t m_u^{-1}$ (mod N) and *wait* until $\sigma_{st} = m_s m_t^{-1}$ becomes available to compute $\sigma_{ru} = \sigma_w \sigma_{st} = m_r m_u^{-1}$. For proofs of correctness and security, we defer to the Bellare and Neven paper [7].

3.3 Directed Transitive Signatures

Unlike UTS schemes, no directed transitive signature schemes, even under lightly tested assumptions, have emerged. To our knowledge, they have not even been mentioned in the cryptographic literature since being suggested by Rivest in a series of talks in 2000 [49].

Lack of interest is not a very plausible explanation, since undirected transitive signatures and delegateable signatures (an idea proposed simultaneously with transitive signatures by Rivest) have enjoyed pursuit [41, 7, 60, 3, 43]. The real reason is more likely because directed transitive signatures are difficult to construct. But can we prove this? And just *how* difficult are they to construct compared to other well known applications?

This thesis provides the first explanations. In the next two chapters, we prove that any directed transitive signature construction would yield a mathematical construction not known to exist (and provably more complex, in a black-box sense, than any of the standard cryptography applications that we typically think of as difficult, such as public key encryption and oblivious transfer.) It is amazing, in our opinion, that directed transitive signatures (a protocol that allows Bob to combine signatures $\sigma_{\overrightarrow{xy}}$ and $\sigma_{\overrightarrow{yz}}$ to obtain $\sigma_{\overrightarrow{xz}}$) are more

complicated than oblivious transfer (a protocol that allows Alice to securely transmit one of two secrets to Bob without knowing which one she sent!). It is truly due to black-box abstractions that we are able to compare these two widely different protocols and make any sound claims about their relative difficulty.

Chapter 4

Groups with Special Properties

In this chapter, we focus on the relationship between cryptography applications and groups with special algebraic structures. We introduce several group-based primitives, including *groups with infeasible inversion* (GII), *one-way group homomorphisms* (OWGH), and *one-way group isomorphisms* (OWGI). Our motivation for adding to an already large set of cryptographic primitives is that the current primitives do not assume any algebraic structure associated with the elements they operate on. Indeed, exploring the power of special group structures is a growing area of interest in cryptography with the recent results in multilinear forms (a special group morphism described in Chapter 2) and homomorphic encryption and signature schemes [35, 17, 30, 14]. We encourage studying the current primitives when considered as operating on a group by giving a vivid example of functions that provably do not imply key agreement without any structure assumptions, but when considered as operating on a group, provably do.

We also note that it is not at all obvious that because primitive $P \rightarrow Q$ it follows that P (on a group) $\rightarrow Q$ (on a group). Some standard black-box reductions destroy the group structure, as in the example of Section 4.6. We also discuss a surprising observation that the ambiguity of pre-images in trapdoor functions appear to be more useful for constructing secure UTS schemes than the disambiguity of trapdoor permutations when we consider the operations on a group. In a strict black-box sense, permutations are considered as more powerful than functions (when operating on non-structured); we point out that this relationship may not hold when the primitives are considered as group homomorphisms. We generalize a

notion of security introduced by Bellare, Namprempre, Pointcheval, and Semanko [6] called *one-more inversion* security for proofs involving one-way group isomorphisms (OWGI).

Additionally, we introduce the notion of "Reverse Cryptography", where instead of building an application from a toolbox of primitives and algebraic structures, one asks what algebraic structures are implied by the existence of an application. We feel this technique may help in explaining why some appliations have been long unconstructed. Let us begin by formally defining a few fundamentals.

Group. We say that G is a mathematical *group* if it possesses a set S of elements, a binary operator $\circ$, and an 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 $\circ$ is associative.)
5. Samplability - there exists an efficient algorithm for selecting elements uniformly at random from S, denoted as $x \xleftarrow{R} S$.
6. Equality Testing - $\forall x, y \in S$, there is an efficient algorithm for testing $x = y$.

We restrict our focus to groups where $\circ$ is polynomially computable in the size of the largest argument. Note that if G meets all the above properties for a group and the operator $\circ$ is commutative, then we say that G is an *Abelian* group.

Group Isomorphism [48]. Two groups G_1 and G_2 with binary operators $\circ$ and $*$ are isomorphic if there exists a one-to-one map $f : G_1 \to G_2$ which satisfies

$$\forall x, y \in G_1,\ f(x \circ y) = f(x) * f(y).$$

An isomorphism is one-to-one and thus preserves the identities and inverses of a group. An isomorphism of a group onto itself is called an **automorphism**. A **homomorphism**

generalizes the notion of isomorphism by dropping the one-to-one requirement.

4.1 Groups with Infeasible Inversion

Several months ago we decided to tackle the open problem of finding provably secure transitive signature schemes for directed graphs (DTS), as proposed by Micali and Rivest [41]. Failure after failure eventually convinced us that such a construction was beyond the power of today's standard cryptographic techniques. As a first step toward proving such a claim, we attempted to give exact mathematical criteria that, were it to exist, would make tractable the construction of a DTS scheme.

Indeed, our failed constructions led us to the conclusion that the existence of inverses (over some set of elements) was necessary for the cancellations involved in signature composition, while at the same time finding those inverses needed to be difficult to prevent simply inverting a signature on edge $\overrightarrow{xy}$ to forge a signature in the opposite direction $\overrightarrow{yx}$. In this chapter, we introduce a new cryptographic primitive capturing this intuition: *groups with infeasible inversion* (GII).

In the next chapter, we show that these special groups, when equipped with a trapdoor, are not only sufficient, but also *necessary* mathematical criteria for any secure DTS scheme. Unfortunately, we are not aware of an implementation of groups with infeasible inversion, even under a lightly tested assumption. However, we postulate the existence of GII groups, and strongly encourage additional research in the hunt for possible examples of this attractive new primitive.

Group with Infeasible Inversion (GII). Given any *finite* group $G = (S, \circ, e)$, we say that G is a *group with infeasible inversion* if it has the additional property:

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

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

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

Trapdoor Group with Infeasible Inversion (TGII). Given any GII $G = (S, \circ, e)$, we say that G is a *trapdoor group with infeasible inversion* if there exists a trapdoor t, such that $|t| \leq poly(k)$, and given t finding inverses becomes polynomial-time computable.

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

4.2 Reverse Cryptography

Standard cryptography (or "forward cryptography") typically involves building application Q out of a set P of primitives and/or algebraic assumptions. If we want to convince others that $P \rightarrow Q$, we simply find a way to build Q out of P. However, trying to prove that there is *no* way to build Q out of P seems intuitively harder. Do we try every possible construction and show that they all fail? Obviously, this is not a feasible technique. However, some desirable applications, such as directed transitive signatures, remain unbuilt. Thus, we want techniques for proving that applications are either truly impossible or at least beyond the reach of our current toolbox.

In a stunning 1989 result [32], Russell Impagliazzo and Steven Rudich introduced a general method for proving statements of the form, "Cryptographic application X has no black-box reduction to complexity assumption Y." They gave strong evidence that OWP$\not\rightarrow$ KA by considering a P=NP world where all parties have access to a totally random permutation oracle in the sky. Being totally random, it remains a OWP even when P=NP. In such a world, they are able to prove that any key agreed upon by Alice and Bob has a good chance of being discovered by an eavesdropper, Eve. In other words, if there is anything that Bob can say to Alice over an open channel that makes her likely to pick a certain key, then Eve is also likely to choose the same key. Using the techniques of Impagliazzo and Rudich, one can give strong evidence against the existence of a black-box reduction $P \rightarrow Q$, since demonstrating the existence of such a reduction would prove outright that P $\neq$ NP. Gertner, Kannan, Malkin, Reingold, and Viswanathan used oracle separations techniques in a 2000 paper [23] to show, among other things, that public key encryption (PKE) and oblivious transfer (OT) are *incomparable* in a black-box setting. By incomparable, they show that

there are oracles relative to which PKE exists and OT does not, and vice versa; so neither a fully black box implication nor a fully black-box separation can be shown.

Although incredibly important and innovative, oracle separation proofs are also extremely difficult in practice. One difficulty is that we generally do not consider ourselves in a P=NP world. Another obstacle is showing that there is *no way* honest parties can act in order to maintain security from Eve; this can be a daunting task. So, how else might we understand the complexity of a desired application Y? How else might we decide if is can be constructed our of our toolbox? We propose applying a notion from set theory:

Reverse Cryptography. In reverse cryptography, we ask what algebraic assumptions (i.e., group structure) are implied by an application? If application Y implies an algebraic assumption X, and X is provably impossible, then obviously we have a proof that Y is also impossible. Similarly, if X is not known to exist, then this is an excellent explanation for why Y remains unbuilt.

We provide an interesting example of reverse cryptography related to the open problem of directed transitive signatures. What algebraic assumption does a DTS scheme imply? By looking at the edge signatures of a DTS scheme, we are able to show in Theorem 5.5.1 that they form an Abelian TGII group, where the signatures are the elements S, the group operator $\circ$ is the composition algorithm, and the identity e is the class of values obtained by composing any two *inverse* signatures: $\sigma_{\overrightarrow{xy}} \circ \sigma_{\overrightarrow{yx}} = e$. The composition properties of a DTS group allow us to combine signatures in any order to obtain the same resulting signature, thus associativity and commutativity are implied. It would be a forgery for anyone to easily produce $\sigma_{\overrightarrow{xy}}$ from $\sigma_{\overrightarrow{yx}}$, thus inversion is hard; and yet the master signer must be able to sign edges in both directions, so there must exist a trapdoor. (Section 5.5 contains a formal proof of this intuition.)

It is interesting in its own right that DTS schemes imply Abelian TGII groups, since this is an algebraic structure not known to exist. Thus, this is the first explanation as to why no one has proposed a secure DTS scheme. We are also able to show that a TGII group implies a *trapdoor permutation* (TDP). We combine our work (Theorems 5.3.1, 5.2.1 and 5.5.1) with the results of Gertner, Kannan, Malkin, Reingold, and Viswanathan [23] to prove:

Theorem 4.2.1 *There is no black-box reduction from public key encryption (PKE) nor oblivious transfer (OT) to directed transitive signature schemes (DTS).*

The following corollary is immediate. It is also the first black-box separation, to our knowledge, between a homomorphic signature scheme and a standard digital signature scheme.

Corollary 4.2.2 *There is no black-box reduction from standard digital signature schemes (SDS) to directed transitive signature schemes (DTS).*

4.3 Weakly Collision-Resistant Non-Injective Group Homomorphisms

We formalize the notion of a few functions on group homomorphisms and discuss their relation to the UTS scheme of Micali and Rivest in Section 3.2.1.

Non-Injective Group Homomorphism (NIGH) [38]. Given any two groups G_1 and G_2 with operations $\circ$ and $*$, respectively. We say that a mapping $f : G_1 \rightarrow G_2$ of the group G_1 onto the group G_2 is a *non-injective group homomorphism* if:

(1) $\forall x, y \in G_1,\ f(x \circ y) = f(x) * f(y)$

(2) $\forall z \in G_2,\ \exists x, y \in G_1\ :\ f(x) = f(y) = z$ (i.e., f maps two or more elements in the domain to each element in the range).

The **kernel** of the homomorphism $f : G_1 \rightarrow G_2$ of the group G_1 onto the group G_2 is the set

$$ker(f) = \{a \in G_1\ :\ f(a) = e_2\}$$

where e_2 if the identity element in G_2.

We restrict our attention to groups where computing inverses with respect to $\circ$ and $*$ is easy for all elements of G_1 and G_2, respectively. We already have a name for groups where taking inverses is hard, GII. Now, we observe an interesting property of NIGH homomorphisms.

Claim 4.3.1 *If $f : G_1 \to G_2$ is a* NIGH *mapping the group G_1 onto the group G_2, then $|ker(f)| > 1$, and for any element y in the range of f, $|f^{-1}(y)| = |ker(f)|$.*

Proof. By property 2 in the definition of a NIGH, we know that all elements in the range of f, including the identity of G_2, have at least two pre-images; thus, $|ker(f)| > 1$. Furthermore, the number of pre-images for any $f(x) \in G_2$ is exactly the number of elements in $ker(f)$, and all the pre-images of $f(x)$ are of the form $x \circ b$ for some $b \in ker(f)$ (i.e., $f(x \circ b) = f(x)$). Conversely, if $f(x) = f(x')$, then $f(x) * f(x^{-1} \circ x') = f(x')$, and therefore $x^{-1} \circ x' \in ker(f)$. We conclude that there are exactly $|ker(f)|$ unique pre-images for every y in the range of f. □

One-Way Group Homomorphism (OWGH). We say that a mapping $f : G_1 \to G_2$ of the group G_1 onto the group G_2 is a *one-way group homomorphism* if it satisfies property 1 of a NIGH and f is a one-way function.

Weakly Collision-Resistant Non-Injective Group Homomorphism (WCRNIGH). Given any non-injective group homomorphism (OWNIGH) from G_1 to G_2, we say it forms a *weakly collision-resistant non-injective group homomorphism* (WCRNIGH) if it remains hard for any PPT adversary, given $x \in G_1$, to find $x' \in G_1$ such that $x' \neq x$ and $f(x) = f(x')$ with non-negligible probability.

We focus on WCRNIGH where G_1 is *samplable*, meaning there exists an efficient algorithm for selecting elements in G_1 uniformly at random. When G_1 is samplable, any weakly collision-resistant NIGH is also a one-way NIGH. Note, for non-structured functions as defined in Chapter 2, weak collision-resistance does not imply one-wayness.

Claim 4.3.2 *If $f : G_1 \to G_2$ is a* WCRNIGH *mapping the group G_1 onto the group G_2 and G_1 is samplable, then f is also a* OWNIGH.

Proof. Assume that f is not one-way. Then there exists a PPT adversary A that on input $f(x)$, can find x' such that $f(x') = f(x)$ with non-negligible probability p. We sample G_1 uniformly at random, obtain a value z, and give $f(z)$ to adversary A. By Claim 4.3.1, we know that $f(z)$ has at least two pre-images in G_1. Thus, with probability at least $p/2$, A

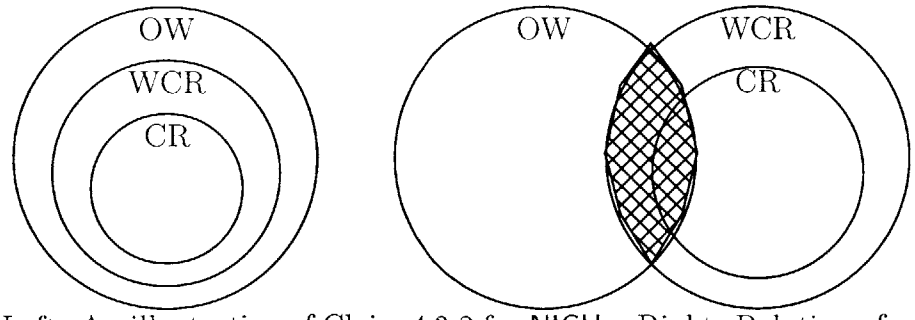

Figure 4-1: Left: An illustration of Claim 4.3.2 for NIGHs. Right: Relation of one-way (OW), weakly collision-resistant (WCR), and collision-resistant (CR) functions on non-structured domains.

returns a z' such that $z' \neq z$ and $f(z') = f(z)$. Since A is efficient and sampling is efficient, this violates the weakly collision-resistant property of f. □

For completeness, we also note that the converse is not true.

Claim 4.3.3 *If $f : G_1 \to G_2$ is a* OWNIGH *mapping the group G_1 onto the group G_2 and G_1 is samplable, then f is not necessarily a* WCRNIGH.

Proof. Recall Claim 4.3.1 that for any NIGH, $|ker(f)| > 1$. If a value $b \in ker(f)$ is publicly known, and b is not the identity element for G_1, then $\forall x \in G_1$ the values $f(x)$ and $f(x \circ b)$ collide. Thus, f might be one-way and not be weakly collision-resistant. □

Observe that the discrete logarithm problem, as used in Micali and Rivest's UTS construction in Section 3.2.1, forms an WCRNIGH:

Discrete logarithm (g, h, p, q) as an WCRNIGH **$(f : \mathbb{Z}_q^2 \to G_q)$:** Recall that p and q are large primes such that q divides $(p-1)$; g and h are generators of the subgroup G_q of order q of $\mathbb{Z}_p^*$ such that the base-g logarithm of h modulo p is hard for any PPT adversary to compute.

Property 1 [Homomorphism]: For groups $\mathbb{Z}_q^2$ and G_q with operations subtraction and division, respectively; there exists a group homomorphism $f(x, y) = g^x h^y \bmod p$ such that $\forall (x_1, y_1), (x_2, y_2) \in \mathbb{Z}_q^2$, $g^{(x_1 - x_2)} h^{(y_1 - y_2)} = g^{x_1} h^{y_1} / g^{x_1} h^{y_2} \bmod p$

Property 2 [Weakly Collision-Resistant]: Assume f is not weakly collision-resistant. Compute the base-g logarithm of h, by finding distinct pairs (a, b) and (c, d) such that

$g^a h^b = g^c h^d \bmod p$, and computing $h = g^{(a-c)/(b-d)} \bmod p$. Finding the base-$g$ logarithm of h is generally thought to be hard.

Property 3 and 4 [Easy Inversion]: Taking additive inverses in $\mathbb{Z}_q^2$ is easy; taking multiplicative inverses in G_q is easy.

Property 5 [Non-Injective]: Each point $g^x h^y \in G_q$ has multiple pre-images in $\mathbb{Z}_q^2$. For example, two of its pre-images are (x, y) and $(x - \log_g h, y + 1)$.

In an WCRNIGH, Micali and Rivest show how to use the homomorphism property for signature composition and verification; one-wayness for node public labeling; easy inversion for signature composition (although this requirement can be removed by substituting edge signatures in both directions); the non-injective and weakly collision-resistant properties are used to prove security. Micali and Rivest prove the security of their UTS scheme under the discrete logarithm assumption by arguing that any edge signature forgery has a non-negligible probability of exposing a pre-image of f unknown to the master signer, thereby yielding an efficient method for computing the base-g logarithm of h. The square-root based UTS scheme of Bellare and Neven [7] operates in exactly the same way, mapping $\mathbb{Z}_n^*$ to $\mathbb{Z}_n^*$ where two distinct square roots modulo the composite of two large primes (n) exposes the factorization with constant probability.

We present WCRNIGH as the first abstraction of these schemes and show that any WCRNIGH assumption can be used to construct a secure UTS scheme.

4.3.1 WCRNIGH $\longrightarrow$ UTS

Theorem 4.3.4 *The existence of any* weakly collision-resistant non-injective group homomorphism *(*WCRNIGH*) implies the existence of an* undirected transitive signature scheme *(*UTS*).*

Proof sketch. Let $f : G_1 \to G_2$ be an WCRNIGH, where $\circ$ and $*$ are the respective group operators. We construct a UTS scheme as follows.

KEY GENERATION. Since f is a one-way function, we invoke Rompel's result that a standard digitial signature scheme can be constructed from any one-way function [51]. Let this standard signature scheme be denoted SSign with public key *spk* and private key *ssk*.

NODE CERTIFICATION. The NCert algorithm maintains state (N, ℓ, L, Σ) and takes as input ssk and a node index i. If this is a new node (i.e., $i \notin N$), then record its index, assign it secret and public values, and sign its public value with SSign (i.e., $N \leftarrow N \cup \{i\}, \ell(i) \xleftarrow{R} G_1, L(i) \leftarrow f(\ell(i)), \Sigma(i) \leftarrow \mathsf{SSign}(ssk, i||L(i))$).

EDGE SIGNING. The ESign algorithm takes two node indices i, j as input. If $i \neq j$ and the signatures $\Sigma(i), \Sigma(j)$ verify with respect to spk, then it returns $\sigma_{ij} = \ell(i) \circ \ell(j)^{-1}$.

VERIFICATION. The Vf algorithm takes as input the public values for nodes i, j and a value σ, and returns 1 if and only if both node public labels verify with respect to spk and $L(i) * L(j)^{-1} = f(\sigma)$.

COMPOSITION. The Comp algorithm takes as input the public values for nodes i, j, k and two values σ_1, σ_2. If all node labels verify with respect to spk, then it returns $\sigma_1 \circ \sigma_2$; otherwise, it returns $\perp$ to indicate failure.

Note: if G_1 is an Abelian group, then this holds for our commutative UTS definition of Section 3.1. Given edge signatures $\sigma_{ab} = \ell(a) \circ \ell(b)^{-1}$ and $\sigma_{bc} = \ell(b) \circ \ell(c)^{-1}$, observe that the signature σ_{ac} can be computed as either $\sigma_{ab} \circ \sigma_{bc}$ or $\sigma_{bc} \circ \sigma_{ab}$. Given σ_{rs} and σ_{tu}, a user can combine them into a shorter waiting signature $\sigma_w = \ell(r) \circ \ell(s)^{-1} \circ \ell(t) \circ \ell(u)^{-1}$ (mod N) and *wait* until $\sigma_{st} = \ell(s) \circ \ell(t)^{-1}$ becomes available to compute $\sigma_{ru} = \sigma_w \sigma_{st} = \ell(r) \circ \ell(u)^{-1}$.

We know that our node signing algorithm SSign is secure, thus any adversary A against this UTS scheme must forge an edge signature reusing node labels. Without loss of generality, suppose A forges an edge signature σ between nodes i and j. With probability at least 1/2 (since this is a *non-injective* homomorphism), σ is not the same pre-image that the master signer knows: $\ell(i) \circ \ell(j)^{-1}$. This breaks the weakly collision-resistant property of our WCRNIGH since the master signer can always run A's algorithm herself, obtaining a second pre-image for a specific point in the range of f with constant probability. Thus, no such A can exist, and this UTS scheme is transitively unforgeable under adaptive chosen-message attack under the instantiating WCRNIGH assumption. □

4.4 One-Way Group Isomorphisms

In the previous section, we focused on non-injective homomorphisms. In this section, we turn our attention to injective homomorphisms and their relative UTS constructions.

One-Way Group Isomorphism (OWGI) [38]. Given any two groups G_1 and G_2 with operations $\circ$ and $*$, respectively. We say that a one-to-one mapping $f : G_1 \rightarrow G_2$ of the group G_1 onto the group G_2 is a *one-way group isomorphism* if:

(1) $\forall x, y \in G_1,\ f(x \circ y) = f(x) * f(y)$

(2) f is a one-way function

(3) f is one-to-one.

When $G_1 = G_2$ in an OWGI, we call f a **one-way group automorphism** (OWGA). Again, we restrict our attention to groups where computing inverses with respect to $\circ$ and $*$ is easy for all elements of G_1 and G_2, respectively.

Observe that the RSA problem, as used in Bellare and Neven's UTS construction in Section 3.2.2, admits a one-way group automorphism (OWGA):

RSA (n, e) as an OWGA $(f : \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^*)$: Recall that n is the product of two large primes p, q, $\varphi(n) = (p-1)(q-1)$ and $ed \equiv 1 \bmod \varphi(n)$, where d is kept secret.

Property 1 [Homomorphism]: $\forall x, y \in \mathbb{Z}_n^*,\ (xy)^e = x^e y^e \bmod n$.

Property 2 [One-Way]: $f(x) = x^e \bmod n$; this is the RSA assumption.

Property 3 and 4 [Easy Inversion]: Taking multiplicative inverses in $\mathbb{Z}_n^*$ is easy.

Property 5 [One-to-One]: Each $y \in \mathbb{Z}_n^*$ has a unique inverse $y^d \bmod n$.

Property 6 [Automorphism]: RSA maps $\mathbb{Z}_n^*$ onto itself.

In a OWGA, Bellare and Neven show how to use the homomorphism property for signature composition and verification; one-wayness for node public labeling; easy inversion for signature composition (although this requirement can be removed by substituting edge signatures in both directions); however, the one-to-one property of RSA causes the proof technique from the previous section to fail, since the master signer knows the one and only pre-image for any output of f that she computes.

One proof technique that does not work is a simple attempt to show that any adversary against Bellare and Neven's RSA-based UTS scheme A can be converted into an adversary

that inverts RSA with non-negligible probability, A'. The obvious strategy for A' is to assign $y \in \mathbb{Z}_n^*$ (where $y = x^e \bmod n$ for unknown x) as the public label of a node i, hoping that A will forge an edge signature between i and a node where the decryption of the node label is known. However, the adaptive UTS adversary A is allowed to ask for a variety of edge signatures before producing a forgery. In particular, she can ask for an edge signatures such that every node has at least one signed edge. Since A' does not know the value x associated with node i, she can not fulfill all the demands of A and thus this proof technique fails. We discuss an alternative proof technique in the next section.

4.4.1 One-More-Inversion Security

One-More-Inversion Security. As in the RSA-based UTS schemes proposed by Bellare and Neven [7], the security of any OWGI-based UTS scheme will be proven under the relevant one-more-inversion assumption. Indeed, we do not know how to prove its security by any other method. To quote Bellare and Neven [7]:

> This situation (namely a scheme that appears to resist both attack and proof) is not uncommon in cryptography, and we suggest that it is a manifestation of the fact that the security of the scheme is relying on properties possessed by RSA, but going beyond those captured by the assumption that RSA is one-way. Accordingly we seek an alternative, stronger assumption upon which a proof of security can be based.

This observation is exactly so. Beyond its one-wayness, the RSA property that plays a major role in its prior UTS constructions is multiplicative automorphism. However, its injective property, that makes it a perfect candidate for public key encryption, actually hinders its application to transitive signature schemes (and we conjecture: *any* homomorphic signature scheme). Is all hope of proving the security of an RSA-based, or any OWGI-based, transitive signature scheme lost? No, we find hope in a new problem by Bellare, Namprempre, Pointcheval, and Semanko in a 2002 paper on the security of Chaum's blind signature scheme [6]. Chaum's blind signatures are RSA-based, and for twenty years had withstood both attack and standard security proofs. Bellare et al. introduced a new problem called

"one-more-RSA-inversion" that at last gave some security credibility to Chaum's signatures. We believe that the "something extra" in RSA that caused Chaum's signatures to elude standard security analysis (i.e., OWGI), as well as the new method of making a security claim about them (i.e., one-more-inversion problem), are great insights here to stay in the cryptography community. In fact, we believe (although it is beyond the scope of this work to prove) that *one-more-inversion* security is necessary for black-box OWGI proofs in any random oracle model.

Here, we generalize (pretty much quoting) the one-more-RSA-inversion definition [6] to work for any OWGI. As Bellare et al. introduced a similar definition for the discrete logarithm problem in the same paper, generalization was already on the way. However, as we showed in the last section, the discrete log-based scheme of Micali and Rivest did not require one-more-inversion security, since they (purposely) chose groups with a non-injective homomorphism.[1]

Single-Target Inversion Problem: OWGI-STI. Let $k \in \mathbb{N}$ be the security parameter. Let A be an adversary. Consider the following experiment:

Experiment $\mathbf{Exp}_A^{owgi-sti}(k)$

Choose a random OWGI f (i.e., $f \stackrel{R}{\leftarrow} KeyGen(k)$)

$y \stackrel{R}{\leftarrow} range(f)$; $x \leftarrow A^f(k, y)$

If $f(x) = y$, then return 1 else return 0.

We define the advantage of A as $\mathbf{Adv}_A^{owgi-sti}(k) = Pr[\mathbf{Exp}_A^{owgi-sti}(k) = 1]$. The OWGI-STI problem is said to be *hard* if the function $\mathbf{Adv}_A^{owgi-sti}(k)$ is negligible for any adversary A whose time-complexity is polynomial in the security parameter k. Observe that this is identical to the "hard to invert" property of one-way functions described in Chapter 2; we state it again here in order to more clearly compare it to the following problem.

One-More Inversion Problem: OWGI-OMI[m]. Let $k \in \mathbb{N}$ be the security parameter, and let $m : \mathbb{N} \rightarrow \mathbb{N}$ be a function of k. Let A be an adversary with access to an OWGI-challenge oracle C and an OWGI-inversion oracle $I(\cdot)$. Consider the following experiment:

Experiment $\mathbf{Exp}_{A,m}^{owgi-omi}(k)$

[1] Rivest recalls attempting a UTS construction using an isomorphism f mapping $\mathbb{Z}_q$ onto G_q (i.e., $x \rightarrow g^x \bmod p$), which he and Micali adapted to the construction in Section 3.2.1 after the injectivity of f prevented a proof in the standard way.

Choose a random OWGI f (i.e., $f \xleftarrow{R} KeyGen(k)$)

For $i = 1$ to $m(k) + 1$ do $y_i \leftarrow C$ (where C chooses $y_i \xleftarrow{R} range(f)$)

$(x_1, x_2, \ldots, x_{m(k)+1}) \leftarrow A^{Q(\cdot),f}(k, y_1, y_2, \ldots, y_{m(k)+1})$

If the following are both true then return 1 else return 0

1) $\forall i \in \{1, 2, \ldots, m(k) + 1\}, f(x_i) = y_i$

2) A made at most $m(k)$ oracle queries to Q.

We define the advantage of A as $\mathbf{Adv}^{owgi-omi}_{A,m}(k) = Pr[\mathbf{Exp}^{owgi-omi}_{A,m}(k) = 1]$. The OWGI-OMI[$m$] problem is said to be *hard* if the function $\mathbf{Adv}^{owgi-omi}_{A,m}(k)$ is negligible for any adversary A whose time-complexity is polynomial in the security parameter k. The general OWGI-OMI problem is said to be *hard* if OWGI-OMI[m] is hard for all polynomially-bounded $m(\cdot)$.

Bellare et al. [6] point out that OWGI-OMI[0] is the same as OWGI-STI. In fact, they go on to define two versions of the one-more inversion problem: in the first experiment, the adversary must win by inverting *all* $m(k) + 1$ targets after asking at most $m(k)$ inversion queries; in a second experiment, the adversary can win by inverting any number of targets $q > 1$ after asking only $q - 1$ inversion queries. Luckily, the authors then show that these two experiments share equivalent complexity; if one is hard, then so it the other and vice versa. Thus, we need only focus our attention on the generalized one-more inversion problem above, which to anyone's knowledge, is hard.

4.4.2 OWGI $\longrightarrow$ UTS

Theorem 4.4.1 *The existence of any* one-way group isomorphism *(*OWGI*) implies the existence of an* undirected transitive signature scheme *(*UTS*) secure under the one-more-inversion assumption.*

Proof sketch. Let $f : G_1 \rightarrow G_2$ be a OWGI, where $\circ$ and $*$ are the respective group operators. Let C and $I(\cdot)$ be the challenge and inversion oracles for OWGI, respectively. This construction is identical to the non-injective homomorphism construction in Section 4.3.1, except for the NCert and ESign algorithms.

KEY GENERATION. Since f is a one-way function, we invoke Rompel's result that a stan-

dard digitial signature scheme can be constructed from any one-way function [51]. Let this standard signature scheme be denoted SSign with public key spk and private key ssk.

NODE CERTIFICATION. The NCert algorithm maintains state (N, ℓ, L, Σ) and takes as input ssk and a node index i. If this is a new node (i.e., $i \notin N$), then record its index, assign it secret and public values, and sign its public value with SSign (i.e., $N \leftarrow N \cup \{i\}, L(i) \leftarrow C, \Sigma(i) \leftarrow \mathsf{SSign}(ssk, i||L(i))$). Notice that NCert computes the public information $L(i)$ by querying the challenge oracle C and does not know the corresponding secret information $\ell(i)$.

EDGE SIGNING. The ESign algorithm takes two node indices i, j as input. If the signature for edge $\overrightarrow{ij}$ is already in the transitive closure of G, then ESign computes $\sigma_{\overrightarrow{ij}}$ by calling the Comp algorithm (just as any regular user would). Otherwise, ESign queries the inversion oracle $I(L(i) * L(j)^{-1})$ and outputs its response.

VERIFICATION. The Vf algorithm takes as input the public values for nodes i, j and a value σ, and returns 1 if and only if both node public labels verify with respect to spk and $L(i) * L(j)^{-1} = f(\sigma)$.

COMPOSITION. The Comp algorithm takes as input the public values for nodes i, j, k and two values σ_1, σ_2. If all node labels verify with respect to spk, then it returns $\sigma_1 \circ \sigma_2$; otherwise, it returns $\perp$ to indicate failure.

Note: if G_1 is an Abelian group, then this holds for our commutative UTS definition of Section 3.1. Given edge signatures $\sigma_{ab} = \ell(a) \circ \ell(b)^{-1}$ and $\sigma_{bc} = \ell(b) \circ \ell(c)^{-1}$, observe that the signature σ_{ac} can be computed as either $\sigma_{ab} \circ \sigma_{bc}$ or $\sigma_{bc} \circ \sigma_{ab}$. Given σ_{rs} and σ_{tu}, a user can combine them into a shorter waiting signature $\sigma_w = \ell(r) \circ \ell(s)^{-1} \circ \ell(t) \circ \ell(u)^{-1}$ (mod N) and *wait* until $\sigma_{st} = \ell(s) \circ \ell(t)^{-1}$ becomes available to compute $\sigma_{ru} = \sigma_w \sigma_{st} = \ell(r) \circ \ell(u)^{-1}$.

Since the security proof under the OWGI-OMI assumption follows the same counting argument as the proof of Bellare and Neven's RSA-based scheme, we refer the reader to their paper for details [7].

Roughly, C is queried n times, once for each call to NCert, while I is queried z times, once for each call to ESign. We know $m - 1$ edges suffice to connect any graph of m nodes. If at the time of forgery there are r partitions (unconnected sections of the graph), and we

know there must be at least two partitions for a forgery to take place, then exactly $n - r$ queries of the inversion oracle I must be made by the master signer to satisfy any adversary A. After A forges an edge signature, $r - 1$ partitions remain. By making a single I query for one public node label in each of the $r - 1$ partitions, the master signer is able to recover the pre-images of all n node public labels. The total number of I queries it must make to do this is exactly $(n - r) + (r - 1) = n - 1$. Thus, our master signer could always run A's forgery algorithm to gain a non-negligible advantage in the OWGI one-more-inversion problem. If we assume that a specific OWGI-OMI problem is hard, than that OWGI-based UTS construction is transitively unforgeable under adaptive chosen-message attack. □

4.5 Pseudo-free Groups

In this section, we introduce one last group notion that will be useful in our proof of Theorem 5.2.1 (the sufficient conditions for a directed transitive signature scheme (DTS)). In fact, we realized after we had written an (incorrect) proof that Abelian TGII implies DTS, that we were actually making an additional assumption on our Abelian TGII group, which we formalize here as *pseudo-freeness*. It seems likely that this concept will be of indepenent interest.

Generator [48]. A set of *generators* $(g_1, g_2, \ldots, g_n)$ is a set of group elements such that possibly repeated applications of the generators on themselves and each other are capable of producing all the elements in the group. For example, generators $\{2, 3\}$, with the identity element 0, generate the additive group of all integers.

Free Group [2, 48]. A group G is a free group if no relation exists between its generators (other than the relationship between an element and its inverse required as one of the defining properties of a group.) More explicitly, a free group has a set of generators $S = \{x_1, x_2, x_3, \ldots\}$, represented as abstract symbols, which may be finite or infinite. We define a *word* to be a finite string of symbols from S, in which repetition is allowed. For example, x_1, x_2x_2, and $x_2x_3x_2$ are words. Two words are composed by concatenation:

$$x_1x_3, x_4 \rightsquigarrow x_1x_3x_4;$$

in this way the associative property is maintained for all words in the group. In addition to the generators, we add the symbol 1 to S to denote the "empty word" or the identity, and a set of inverse symbols, where for each $x_i \in S$, we add x_i^{-1} to S. If the symbols $\cdots x_i x_i^{-1} \cdots$ or $\cdots x_i^{-1} x_i \cdots$ appear as immediate neighbors in a word, then we cancel both symbols and reduce the length of the word. A word is called *reduced* if no such cancellation can be made. Although there is often more than one way to proceed with a cancellation, there is only one reduced form of a given word w. Two words are called *equivalent* if they have the same reduced form. In a free group, there does not exist any reduced word w that is equivalent to 1, except $w = 1$. For example, there is no relation of the form $x_1 x_2^{-1} x_3 x_2 = 1$.

Notice that $x_1 x_2^{-1} x_3 x_2$ is in reduced form. In a **free Abelian group**, however, $x_i x_j = x_j x_i$ for all symbols $x_i, x_j \in S$; thus inverses cancel anywhere in the word (i.e., $x_1 x_2^{-1} x_3 x_2$ reduces to $x_1 x_3$).

Pseudo-free Group (PFG). A group G is called *pseudo-free* if it is indistinguishable from a free group for any PPT adversary, given black-box access to G. All free groups are also pseudo-free groups. We introduce the notion of a group in black-box representation, modeling it on the black-box fields of Boneh and Lipton [13]. For each element in the group, we refer to $[x]$ as the black-box representation of element x; similarly $[G]$ is the black-box representation of group G. Boneh and Lipton often allow the field characteristic for a black-box field to be public. For our purposes, we keep all information about G private. An adversary is only allowed to query the following set of black-box subroutines:

1. $[G] \leftarrow$ MakeGroup(1^k); a randomized algorithm, that on input 1^k, produces the black-box representation of a group G.

2. $[e] \leftarrow$ Identity($[G]$); returns the black-box representation of the identity element in G.

3. $[x] \leftarrow$ Sample($[G]$); returns the black-box representation of a random generator for G. In the case that G has infinite size, x will be selected non-uniformly at random.

4. $[z] \leftarrow$ Compose($[x], [y]$); given the black-box representation of two elements x, y, it returns the black-box representation of $x \circ y$, where $\circ$ is the group operation.

5. $[x^{-1}] \leftarrow \text{Inverse}([x])$; given the black-box representation of x, it returns the black-box representation of the inverse of x.

6. {true,false} $\leftarrow \text{Equal}([x], [y])$; given the black-box representation of two elements x, y, it turns true if $x = y$, and false otherwise.

In an **Abelian pseudo-free group**, $\text{Compose}([x], [y])$ and $\text{Compose}([y], [x])$ result in a black-box representation of the same element for any $x, y \in G$.

For a PPT adversary A to distinguish G from a free group, she must output an identity that holds in G, but would not hold in a free group, given only black-box access to the group. We define an *identity* as a straight-line program with operators Identity, Compose, Inverse, and Equal, starting from a given set of elements created by Sample, such that the result the straight-line program is a non-trivial identity for G. For example, starting with $[G]$ and $\{[a], [b]\}$ from Sample, A would succeed in distinguishing G from a free group, if she could output:

$$
\begin{aligned}
&[a^2] \leftarrow \text{Compose}([a], [a]) \\
&[a^4] \leftarrow \text{Compose}([a^2], [a^2]) \\
&[b^{-1}] \leftarrow \text{Inverse}(b) \\
&[z] \leftarrow \text{Compose}([a^4], [b^{-1}]) \\
&[e] \leftarrow \text{Identity}([G]) \\
&\text{true} \leftarrow \text{Equal}([z], [e])
\end{aligned}
$$

This straight-line program indicates that the non-trivial identity $a^4 \circ b^{-1} = e$ exists for G. By non-trivial, we mean any identity that would not hold in a free group.

4.5.1 Discussion of Pseudo-free Definition

Our definition of a pseudo-free group is the first formalization of this notion to our knowledge. It is probable that this definition will need refinement as we encounter other applications for pseudo-free groups, in addition to Theorem 5.2.1, where we assume pseudo-freeness as part of the sufficient conditions for a DTS scheme. In Theorem 5.2.1, we combine pseudo-freeness with a TGII, which is finite by definition, and therefore, not a free group. We are able to show, however, that if the TGII appears as a free group to the adversary, then the DTS

construction is secure.

One avenue for better understanding this new concept is to explore some "real world" examples, under the standard cryptographic assumptions. For example, does $\mathbb{Z}_N^*$ (with RSA modulus N) meet our definition for a pseudo-free group? If a PPT adversary can distinquish $\mathbb{Z}_N^*$ from a free group, can we use that adversary to efficiently factor N?

4.6 Adding Group Structure to Primitives

The goal of this section is to motivate the question: what power does one get by considering a standard primitive, usually operating on non-structured elements, as instead operating on a group? Is there a substaintial increase in black-box complexity? In most cases, the answer is probably yes, although this is a relatively unexplored terrain. Consider that most of the standard algebraic constructs used in practical cryptography applications operate on groups: RSA, discrete logarithms, quadratic residues. Yet, our basic cryptography primitives (i.e., one-way functions, trapdoor permutations) do not include any notion of group structure. For some applications (i.e., public key encryption, key agreement), group structure may not play a critical role. However as we attempt to expand our understanding of homomorphic encryption and signature schemes, which critically rely on group properties, we also must expand our understanding of the fundamental primitives that we use.

Furthermore, new relations may emerge. It is not at all clear that because primitive $P \rightarrow Q$ it follows that P (on a group) $\rightarrow Q$ (on a group). In Section 5.9, Hemaspaandra and Rothe's construction of a strongly associative one-way function (SAOWF) from any one-way function (OWF) destroys group structure by introducing an additional $\perp$ symbol. The relationship between functions and permutations is also called into question in a group-based model. We saw earlier in this chapter that special non-injective homomorphisms have standard proofs of security for UTS schemes, while isomorphisms must rely on one-more-inversion proofs. As we typically think of one-way permutations as strictly stronger than one-way functions, this phenomenon deserves some additional study.

We now wish to give a very clear and clean example, illustrated in Figure 4-2, of the stark difference in complexity between function f operating on non-structured elements and

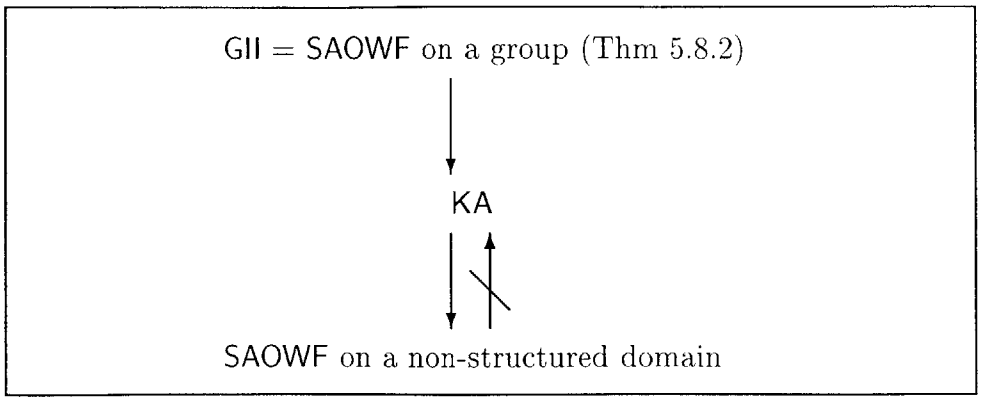

Figure 4-2: Example of the complexity disparity between SAOWF and SAOWF on a group.

function f operating on a group. We first define the function of interest.

Strongly Associative One-Way Function (SAOWF). We give the same definition as Rabi and Sherman [45]. Given any honest binary function $\circ$: $S \times S \to S$ on the message space $S = \{0,1\}^*$, we say that $\circ$ is *strongly associative one-way* if it satisfies the following conditions:

(1) $\forall x, y, z \in S$, $(x \circ y) \circ z = x \circ (y \circ z)$

(2) $\forall x \in S$, $\circ$ is polynomial time computable in the size of x

(3) for every PPT algorithm A, every positive polynomial $p(\cdot)$, and all sufficiently large k's,

$$Pr[x, y \stackrel{R}{\leftarrow} \{0,1\}^k, x' \leftarrow A(x \circ y, y, 1^k) : x \circ y = x' \circ y] < \frac{1}{p(k)}.$$

Property 3 can also be satisfied when it is difficult to invert $x \circ y$ given the first term x. Only one argument needs to hold relative to property 3 for a function to satisfy this definition.

In Figure 4-2, we illustrate that a SAOWF (on a group) is strictly more complex, in a black-box sense, than a SAOWF assumed to operate on a non-structured domain. Recall Impagliazzo and Rudich's result that key agreement (KA) is strictly more complex, in a black-box sense, than one-way functions (OWF) assumed to operate on a non-structured domain [32]. Hemaspaandra and Rothe proved that SAOWF and OWF are black-box equivalent, when both operate on non-structured domains [28]. We prove in Theorems 5.4.1 and 5.8.2

that when a SAOWF is assumed to be a group operation, it is black-box equivalent to GII and can be used to construct a key agreement protocol.

Chapter 5

Black-Box Reductions between Primitives

In the previous chapters, we introduced the primitives PFTGII, TGII, and GII, and formally defined DTS. In this chapter, we give their black-box relationships to other previously known cryptographic primitives. The goal of this chapter is to rank, as far as possible, the relative black-box complexities of various primitives. The most exciting results in this chapter are:

1. In Theorems 5.2.1 and 5.5.1, we give the first formalization of the necessary (ATGII) and sufficient (PFATGII) mathematical criteria for a DTS scheme. We do not know if TGII groups are inherently pseudo-free groups, thus we leave open a gap between the necessary and sufficient conditions of a DTS scheme. However, closing this gap and understanding the relationship between TGII and pseudo-free groups are a new and interesting algebraic problems.

2. As a corollary of Theorem 5.3.1, we prove that DTS schemes are beyond the complexity of today's standard protocols (such as public key encryption and oblivious transfer) by showing that DTS schemes imply trapdoor permutations. This offers some excuse for the lack of any DTS scheme proposals in the theoretical community.

3. The result of Theorem 5.3.1 is also the first black-box separation, to our knowledge, between standard digital signature schemes (SDS) and any of the many signature

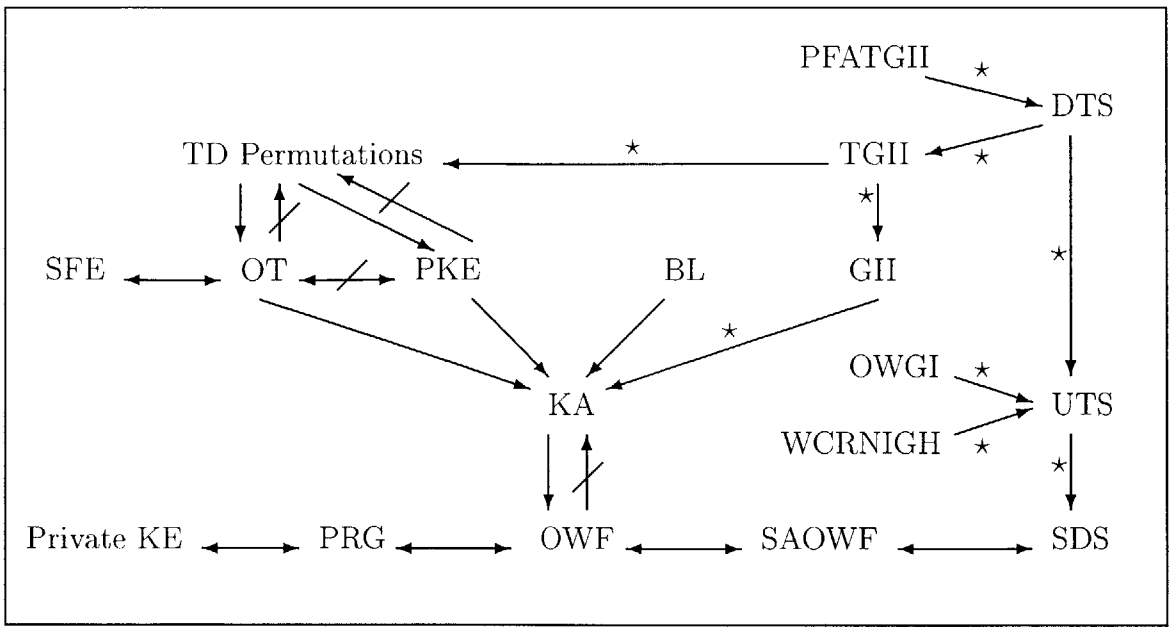

Figure 5-1: Black-Box Relationships between Cryptographic Primitives. The contributions of this thesis are indicated by a $\star$.

variants proposed. We show that DTS schemes are strictly more complex, in a black-box sense, than SDS schemes.

4. In Theorems 5.4.2 and 5.8.2, we demonstrate a strong relationship between our new primitive GII and the strongly associative one-way functions (SAOWF) introduced by Sherman and Rabi [45]. We show that when SAOWFs operate on a group, they are equivalent to GII groups and strong enough to imply key agreement. However, Hemaspaandra and Rothe showed that SAOWFs are no more complex than mere OWFs when no assumptions about the structure of the domain are made.

Figure 5-1 illustrates the relation of our new primitives in comparison with previously known standards, such as trapdoor permutations (TDP), public key encryption (PKE), and one-way functions (OWF). See Appendix A for information about each acronym.

5.1 Black-Box Reductions

A common question that one might ask is: can primitive P be reduced in complexity to primitive Q? For example, can GII groups be constructed out of one-way functions? Are

the existence of signatures enough to guarantee the existence of bilinear maps? We adopt the definitions of *semi-black-box* and *fully black-box reductions* as given by Gertner, Kannan, Malkin, Reingold, and Viswanathan [23] (see chapter 2 for details). In this chapter, by black-box reduction we refer to a fully black-box reduction.

In 2000, Gertner et al. [23] published a result summarizing a number of black-box primitive relationships, including some newly discovered ones. In Figure 5-1, we present the current knowledge of black-box cryptographic primitive relationships. We include and augment the information in Gertner et al. All results not included in the aforementioned work are original results of this work or are properly cited in the following two chapters.

5.2 PFATGII $\longrightarrow$ DTS

Theorem 5.2.1 *The existence of an Abelian pseudo-free* TGII *implies the existence of a* directed transitive signature *(*DTS*) scheme, directed transitively unforgeable under adaptive chosen message attack. (*PFATGII *is sufficient for* DTS.*)*

Proof. Let $||$ denote concatenation. We construct a DTS scheme from an Abelian pseudo-free TGII group $H = (S, \circ, e, t)$. We assume the existence of a standard digital signature scheme SDS=(SKG,SSign,SVf).[1]

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 PK=(e, spk) as well as the description of $\circ$, and gives SK=$(1^k, S, t, ssk)$ to the master signer.[2]

NODE CERTIFICATION. The NCert algorithm maintains state (N, ℓ, L, Σ) and takes as input SK=(k, S, t, ssk) and a node index i from the master signer.

If $i \notin N$, then $N \leftarrow N \cup \{i\}$. Next, a generator from the PFATGII H is randomly selected and assigned as the public label $L(i)$ for node i. NCert maintains state to ensure that each node is assigned a unique generator. Next, the SDS is used to sign this public

[1] We also show in this chapter that TGII implies SDS by showing that TGII $\rightarrow$ GII $\rightarrow$ KA, which we know implies SDS [51].

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

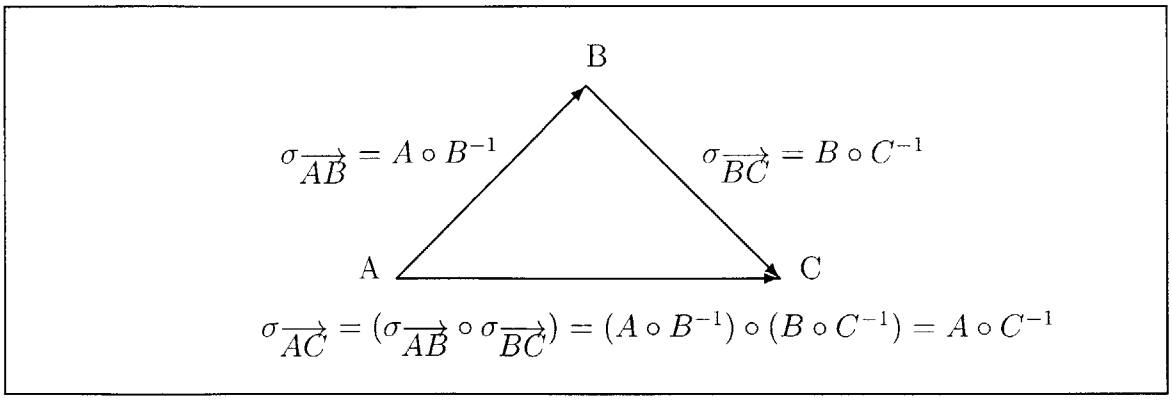

Figure 5-2: Illustration of DTS scheme using an Abelian PFTGII group, where A, B, C are public node labels and A^{-1}, B^{-1}, C^{-1} are the secret keys. One can verify that $\sigma_{\overrightarrow{AC}} \circ C = A$.

information: $\Sigma(i) \leftarrow \mathsf{SSign}(ssk, i||L(i))$. Finally, the trapdoor t is used to compute the secret node information: $\ell(i) = L(i)^{-1}$. The algorithm publishes $p_i = (i, L(i), \Sigma(i))$, where $L(i)$ is the public label for node i, and $\Sigma(i)$ is a certificate to guarantee $L(i)$ belongs to node i. The secret value $\ell(i)$ is given to the master signer.

EDGE SIGNING. The ESign algorithm takes two node indices i, j and secret key SK as input. If $(i \neq j)$ and $\Sigma(i), \Sigma(j)$ verify with respect to spk, then it returns $\sigma_{\overrightarrow{ij}} = L(i) \circ \ell(j)$. Otherwise, it declares failure and returns $\perp$.

VERIFICATION. The Vf algorithm takes as input two multisets of node public values $P_{src} = \{a_1, a_2, \ldots, a_k\}, P_{dest} = \{b_1, b_2, \ldots, b_k\}$ and a value σ, and returns 1 if and only if σ is a signature consistent with P_{src}, P_{dest}. Parse each public value a_x as (x, A_x, Σ_x) and b_x as (x, B_x, Σ_x).

For each $a_i \in P_{src}$, if ($\mathsf{SVf}(spk, i||A_i, \Sigma_i)$=0) return 0.

For each $b_i \in P_{dest}$, if ($\mathsf{SVf}(spk, i||B_i, \Sigma_i)$=0) return 0.

If $(\sigma \circ B_1 \circ B_2 \circ \ldots \circ B_k = A_1 \circ A_2 \circ \ldots \circ A_k)$ then return 1; otherwise return 0.

COMPOSITION. The Comp algorithm takes as input two pairs of multisets of node public values $(P1_{src}, P1_{dest}), (P2_{src}, P2_{dest})$ and values σ_1, σ_2. The algorithm returns either a composed signature σ_3 or $\perp$ to indicate failure. Parse the public node value p_x as (x, L_x, Σ_x).

For each $p_i \in P1_{src} \cup P1_{dest} \cup P2_{src} \cup P2_{dest}$, if ($\mathsf{SVf}(spk, i||L_i, \Sigma_i)$=0), then return $\perp$.

Otherwise, return $\sigma_3 \leftarrow \sigma_1 \circ \sigma_2$.

CORRECTNESS. Recall the correctness requirement of Definition 3.1.1, where a scheme is correct if it never returns true (i.e., (Legit$\wedge$ NotOK)=true) on the experiment in Figure 3-1. 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 value $\ell(i) = L(i)^{-1}$ for each public value $L(i)$. Thus, every valid signature has a unique representation.

SECURITY. We argue that the above scheme is *directed transitively unforgeable under adaptive chosen message attack.* For PPT adversary F and security parameter k, we conduct the experiment $\mathbf{Exp}^{dtu-cma}_{DTS,F}(k)$ from Section 2. 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.

If F can create node labels on her own, then she breaks the underlying standard signature scheme SDS. If F can forge edge signatures using only the public node information, then she can break the infeasible inversion property of H. If F can obtain secret node information from a valid edge signature (i.e., obtain $\ell(j)$ from $L(i) \circ \ell(j)$), then she contradicts Theorem 5.4.2.[3] Thus, F must use a combination of the public node labels and valid edge signatures provided by the master signer, plus any valid edge signatures she can compute by composition, to make her forgery – a valid signature for an edge outside of the transitive closure of H.

Suppose F succeeds in forging an edge signature $\sigma_{\overrightarrow{ij}}$ from node i to node j. Now, we apply the Abelian pseudo-free property of our group H to claim that $\sigma_{\overrightarrow{ij}}$ must be the composition of some $n > 1$ valid edge signatures in the transitive closure of H (i.e., $\sigma_{\overrightarrow{ij}} = L(i) \circ \ell(j) = E_1 \circ E_2 \circ E_3 \ldots E_n$). Let an edge signature be thought of as a pseudo-free word. If F composes a public label with an edge signature (i.e., $L(r) \circ \ell(s) \circ L(t)$ or $L(r)$), she will never be able to reduce it to a word containing exactly one symbol in L, and one inverse symbol in ℓ. All she can do is add more symbols in L, by composing with public labels, or reduce the number of L and ℓ symbols by the same amount, by composing with edge

[3]Theorem 5.4.2 states that the group operation $\circ$ for a GII G is a strongly associative one-way function (SAOWF). By the definition of a SAOWF, given edge signature $L(i) \circ \ell(j)$ and public node information $L(i)$, no PPT adversary can produce ℓ' such that $L(i) \circ \ell' = L(i) \circ \ell(j)$ with non-negligible probability. Since the inverse of a GII element is unique, we know that adversary F can not find $\ell(j)$.

signatures in $L \circ \ell$; this is due to the fact that there are no non-trivial identities in H. Thus, F's only option is to compute her forgery $\sigma_{\overrightarrow{ij}}$ by composing only valid edge signatures.

If $\sigma_{\overrightarrow{ij}}$ is computed by composing valid edge signatures such that $\sigma_{\overrightarrow{ij}} = L(i) \circ \ell(j) = E_1 \circ E_2 \circ E_3 \ldots E_n$, then there must be a directed path in the transitive closure of H from node i to node j. This follows from the Abelian pseudo-free property of H, where each symbol $L(k)$ (denoting node k as the source of a directed edge) is only cancelled by a symbol $\ell(k)$ (denoting node k as the destination of a directed edge) to reduce the composed edge signatures to $L(i) \circ \ell(j)$. Since $\sigma_{\overrightarrow{ij}}$ is in the transitive closure of H, it is not a forgery.

Therefore, F's advantage $\mathbf{Adv}^{dtu-cma}_{DTS,F}(k)$ comes only from breaking the standard digital signature (SDS) or the Abelian PFTGII, which are both negligible. □

5.3 TGII ⟶ TDP, GII

First, let's show that a TGII group can be used to easily construct a trapdoor permutation. In fact, there is an obvious extension to a trapdoor permutation family inherent in the proof, but a single permutation is enough for our purposes.

Theorem 5.3.1 *The existence of a* TGII *group implies the existence of a* trapdoor one-way permutation *(*TDP*).*

Proof. Given a TGII group $G = (S, \circ, e, t)$, we construct a trapdoor one-way permutation P as follows. Randomly select $a \in S$ and compute its inverse $a^{-1} \in S$ using t. Define a permutation $P : S \to S$ where on input x, it outputs $P(x) = x \circ a$. First, we observe that P is easy to compute, since $\circ$ is polynomial-time computable. Secondly, P is one-to-one since for every element $y \in S$, there exists a unique pre-image $x \in S$ such that $y \circ a^{-1} = x$.

We claim that it is hard for any PPT adversary A to produce a value x' such that $x \circ a = x' \circ a$ on input $(a, x \circ a)$. (An adversary against P would know a in the description of the permutation.) Suppose it is easy for A, then for a value $b \in S$, we compute the inverse of b by choosing a random value $c \in S$, and sending $(c, b \circ c)$ to A. Our adversary returns b' such that $b' \circ b \circ c = c$, and we see that $b' = b^{-1}$. Thus, P must be hard to invert or the infeasible inversion property of G does not hold. However, when the trapdoor value a^{-1} is

known, anyone can invert P efficiently as $P^{-1}(P(x)) = (x \circ a) \circ a^{-1} = x$. Thus, P is (1) easy to compute, (2) one-to-one, (3) hard to invert, and (4) trapdoor, making it a trapdoor one-way permutation. □

Theorem 5.3.2 *The existence of a* TGII *group implies the existence of a* GII *group.*

Proof. If TGII group $G = (S, \circ, e, t)$ exists, then GII group $G' = (S, \circ, e)$ exists. □

5.4 GII $\longrightarrow$ KA, SAOWF

Without a trapdoor, groups with infeasible inversion are still powerful constructs. Here we show that they imply key agreement. It is an open question as to whether or not they imply public key encryption or undirected transitive signature schemes.

Theorem 5.4.1 *The existence of a* GII *group implies two-party secret key agreement (*KA*).*

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 selects random $x, y \in S$, and sends the pair $(x \circ y, y)$ to Bob.
2. Bob selects a random $z \in S$, 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 PPT adversary Eve that can take as input all the messages $(x \circ y, y, y \circ z)$ and produce $K_B = (x \circ y) \circ z$ with non-negligible probability. We use Eve to invert elements of G as follows:

1. On input $a \in S$, select a randoms values $i, j, k \geq 0$ such that $i + k - j = -1$.
2. By repeated-squaring, compute (a^i, a^j, a^k) and send to Eve, outputting what she returns.

We can view Eve's input above as $(x \circ y, y, y \circ z)$ for $x = a^{i-j}$, $y = a^j$, and $z = a^{k-j}$. Thus, with non-negligible probability, her output is $(x \circ y \circ z) = (a^{(i-j)+j+(k-j)}) = a^{i+k-j} = a^{-1}$. □

The protocol for Theorem 5.4.1 is due to Muhammad Rabi and Alan Sherman [45]. Although they were not aware of GII groups, to our knowledge, they suggested the above protocol and a multi-party key agreement protocol using SAOWF that we conjecture extends to Abelian GII groups. Rabi and Sherman were unable prove the security of their SAOWF-based protocol outside of Yao's model of computational information theory; by assuming that a SAOWF is a group operator, we are able to prove security, in the standard way, against any PPT adversary. In fact, it is not a coincidence that Rabi and Sherman's SAOWF-based protocol would extend easily to GII groups. Closer examination shows that GII groups are SAOWFs (Theorem 5.4.2 below), while a special case of SAOWF also imply GII groups (Section 5.8, Theorem 5.8.2).

Theorem 5.4.2 *The operator* $\circ$ *in a* group with infeasible inversion *(*GII*) is a* strongly associative one-way function *(*SAOWF*).*

Proof. Let $G = (S, \circ, e)$ be a GII group. We show that $\circ$ is a SAOWF such that no PPT adversary A can take as input $(x \circ y, y)$ and produce x' such that $x \circ y = x' \circ y$ with non-negligible probability. Suppose such an A exists, then she can be used to construct an adversary A' that breaks the infeasible inversion property of the GII group. On input a, A' selects a random $b \in S$ and gives input $(b, a \circ b)$ to A; returning whatever A returns, which is a value a' such that $b = a' \circ a \circ b$ (i.e., $a' = a^{-1}$) with non-negligible probability. The associativity of $\circ$ follows from the definition of a group. □

5.5 DTS $\longrightarrow$ TGII, UTS

Theorem 5.5.1 *The existence of a* DTS *scheme, directed transitively unforgeable under adaptive chosen message attack, implies the existence of an Abelian* TGII *group. (*TGII *is necessary for* DTS.*)*

Proof. Let $\circ$ denote both the Comp algorithm for the DTS and the group operation for our Abelian TGII group. We show that each of the nine properties of an Abelian TGII group $G = (S, \circ, e, t)$ are satisfied by the edge signatures of a secure DTS scheme for a graph

$H = (V, E)$.[4] Recall from the DTS definition (Section 3.1) that a composed signature for edge $\overrightarrow{ij}$ must be *indistinguishable* from (but not necessarily identical to) the master signer's signature on $\overrightarrow{ij}$. All valid edge signatures between a source node i and a destination node j form an equivalence class: $C_{\underset{ij}{\rightarrow}} = \{\sigma_{\underset{ij}{\rightarrow}}\}$. The elements S of our TGII group will be the set of signature classes: $\{C_{\underset{ij}{\rightarrow}}\}$ for all $i, j \in V$, where each element of G can be represented as any one of the signatures in its equivalence class.

1. Closure - By the definition of Comp, we know that composing any two valid signatures $a, b \in S$, yields a valid signature $(a \circ b) \in S$. (Obtaining this property was one of our motivations for adding waiting signatures to the DTS definition.)
2. Identity - The identity e for the TGII group is the equivalence class of signatures on empty edges: $e = C_{\overrightarrow{aa}}$ for all $i \in S$. This represents all self-loops (and compositions of self-loops) of the form $\sigma_{\underset{ab}{\rightarrow}} \circ \sigma_{\underset{ba}{\rightarrow}} = e$.
3. Inverses - For each $a \in S$, the inverse of a is defined as the class of signatures formed by swapping the sets P_{src} and P_{dest}. For example, the inverse of $\sigma_{\underset{ab}{\rightarrow}}$ is $\sigma_{\underset{ba}{\rightarrow}}$.
4. Associativity - The composition of any directed path through nodes $a \rightarrow b \rightarrow c$ must result in a signature indistinguishable from the master signer's signature on edge $\overrightarrow{ac}$. Thus, $\forall a, b, c \in S$, $a \circ (b \circ c) = (a \circ b) \circ c$.
5. Samplability - The master signer can efficiently sign any edge; therefore, he can also serve as a random oracle for TGII elements. In addition, any user can create an exponentially large number of composed signatures, given a polynomial number of original signatures.
6. Equality Testing - Two elements in G are equal if they are a signature for the same edge(s) in H; maintaining P_{src} and P_{dest} sets as part of the element representation allows for efficient equality testing using Vf.
7. Infeasible Inversion - By the DTS security definition, if $\sigma_{\underset{ba}{\rightarrow}}$ is not exposed (i.e., it is not in the transitive closure of H), then it hard for any PPT adversary A, given $\sigma_{\underset{ab}{\rightarrow}}$, to find $\sigma_{\underset{ba}{\rightarrow}}$ with non-negligible probability.
8. Trapdoor - Recall that the master signer of the DTS scheme, a PPT algorithm, must be able to efficiently generate the signature for *any* edge upon demand. Since this

[4] The size of the TGII group is quadratic in the number of nodes in H.

operation must remain hard for all computationally equivalent adversaries, we know that the master signer must possess a trapdoor t that allows him to avoid costly computation.

9. Commutativity - By the definition of the Comp algorithm, edge signatures can be composed in any order. Thus, $\forall a, b \in S$, $a \circ b = b \circ a$. (Waiting signatures enforce this property.)

Since all properties of an Abelian TGII group are met by a DTS scheme, we conclude that the existence of the latter implies the former. □

Theorem 5.5.2 *The existence of a* DTS *scheme, directed transitively unforgeable under adaptive chosen message attack, implies the existence of a* UTS *scheme, which is transitively unforgeable under adaptive chosen message attack.*

Proof. Intuitively, treat each undirected edge as a pair of directed edges. Formally, let our DTS scheme be the set of algorithms (DKG, DNCert, DESign, DVf, DComp). We construct our UTS scheme (UKG, UNCert, UESign, UVf, UComp) as follows. The UKG and UNCert algorithms are identical to those in the DTS scheme. The UESign algorithm takes two node indices i, j as input, and runs DESign(i, j) and DESign(j, i). If either return $\perp$, UESign also returns $\perp$; otherwise, it returns $\sigma_{ij} = (\sigma_{\overrightarrow{ij}}, \sigma_{\overrightarrow{ji}})$. The DVf algorithm takes as input the public information of nodes i, j and a pair (σ_1, σ_2), and outputs DVf$(i, j, \sigma_1) \oplus$ DVf(j, i, σ_2). Finally, the DComp algorithm takes as input the public information of nodes i, j, k and two values (α_1, α_2) and (β_1, β_2), and tries to output $\sigma_{ik} = (\sigma_{\overrightarrow{ik}}, \sigma_{\overrightarrow{ki}})$. If either DComp$(i, j, k, \alpha_1, \beta_1) \rightarrow \sigma_1$ or DComp$(k, j, i, \beta_2, \alpha_2) \rightarrow \sigma_2$ return $\perp$, then UComp also returns $\perp$; otherwise, it returns (σ_1, σ_2). The correctness and security of this UTS scheme follow trivially from those of the DTS scheme. □

5.6 UTS $\longrightarrow$ SDS, OWF

Theorem 5.6.1 is intuitively obvious, and for completeness, we reassure ourselves with a proof.

Theorem 5.6.1 *The existence of an* undirected transitive signature scheme *(*UTS*) implies the existence of a* standard signature scheme *(*SDS*).*

Proof. In a UTS scheme, the master signer places his signature, explicitly or implicitly, on the public information he publishes about each node to prevent an adversary from adding her own nodes to the graph and "forging" an edge between them. Thus, if a UTS scheme exists, then a standard signature scheme also exists by encoding messages as node public labels. Since the UTS scheme is secure against transitive adaptive chosen message attack, the standard signature scheme is also secure against adaptive chosen message attack. □

The following corollary is immediate from John Rompel's result [51] that one-way functions (OWF) and secure digital signature schemes (SDS) are black-box equivalent.

Corollary 5.6.2 *The existence of an* undirected transitive signature scheme *(*UTS*) implies the existence of a* one-way function *(*OWF*).*

5.7 BL $\longrightarrow$ KA

The following result is due to Antoine Joux [36], who first demonstrated the usefulness of bilinear maps in creating one-round three-party key exchange. Dan Boneh and Alice Silverberg generalized his result to show that if multilinear forms were to exist, they would allow for efficient one-round multiparty key exchange [14].

Theorem 5.7.1 *The existence of a* bilinear map *(*BL*) implies three-party secret key agreement (*KA*).*

5.8 SAOWF $\longrightarrow$ OWF, GII*

At first glance, one might assume that a strongly associative one-way function *is* a one-way function. However, a seemingly paradoxical result by Hemaspaandra, Pasanen, and Rothe [29] proves that a SAOWF is not necessarily *one-way*. What do we mean? Recall that for any SAOWF f, given $f(a, b)$ and one of its arguments (say b), it is hard to find a' such

that $f(a', b) = f(a, b)$. However, Hemaspaandra et al. show that it may still be easy to find some a', b' such that $f(a', b') = f(a, b)$. This is an excellent reminder of the non-triviality of comparing primitives. Fortunately for us, Hemaspaandra et al. also demonstrate that a OWF can be constructed from a SAOWF; we give a simplified proof below.

Theorem 5.8.1 *The existence of a* strongly associative one-way function *(*SAOWF*) implies the existence of a* one-way function *(*OWF*).*

Proof. Suppose we have a SAOWF $s(\cdot, \cdot)$. Now, we construct a OWF $f(\cdot)$ that on input x, returns $s(x, a)$ for a public $a \in domain(s)$. No adversary, given $f(x)$, has a non-negligible probability of producing an x' such that $f(x') = f(x)$, since that would mean that she can also invert s by finding an x' such that $s(x', a) = s(x, a)$. □

Theorems 5.8.1 and 5.9.1 give a black-box equivalence between one-way functions (OWF) and strongly associative one-way functions (SAOWF). By Theorem 5.4.1 and the separation between OWF and KA [32], we know that a SAOWF does not imply the existence of a GII group, since the former does not imply KA and yet the latter does. However, it is interesting to observe, in the following proof, that if we can guarantee that the domain of the SAOWF includes inverses and an identity element, a SAOWF *does* imply a GII group. In fact, it is one of the interesting observations of this work that GII groups and SAOWF are so closely related.

Theorem 5.8.2 *The existence of* strongly associative one-way functions on groups *imply the existence of* GII *groups.*

Proof. Suppose there exists a finite, samplable group $G = (S, \circ, e)$ where $\circ$ is a strongly associative one-way function, then G is also a GII. Any PPT adversary A, that can invert a random element in S with probability p, can be used to construct an probabilistic polynomial time adversary A' that inverts $\circ$ with probability p. Suppose A takes as input b and returns the inverse of b with non-negligible probability.

1) When SAOWF adversary A' receives input $(x \circ y, y)$, she sends y to A.

2) Once GII adversary A returns c, A' outputs $(x \circ y) \circ c$.

Since $c = y^{-1}$ with non-negligible probability, then A' outputs x with the same odds. This violates property 3 of the SAOWF, and thus G must be a group with infeasible inversion. □

5.9 OWF $\longrightarrow$ SAOWF

The following result is due to Lane Hemaspaandra and Jorg Rothe[28]. They show that P $\neq$ NP is a sufficient condition for the existence of strongly associative one-way functions, and give a construction assuming P $\neq$ NP that can be easily modified to be black-box from *any* one-way function, including all those of cryptographic interest.

Theorem 5.9.1 *The existence of a* one-way function *(*OWF*) implies the existence of a* strongly associative one-way function *(*SAOWF*).*

Proof sketch. Given a one-way function f, we construct a strongly associative one-way function $\tau(\langle\cdot,\cdot\rangle, \langle\cdot,\cdot\rangle) = \langle\cdot,\cdot\rangle$ as follows. On input $(\langle a, b\rangle, \langle c, d\rangle)$, τ outputs:

$\langle a, min(b, d)\rangle$, if $a = c$ and $a = f(b)$ and $c = f(d)$, else

$\langle a, a\rangle$, if $a = c$ and $(a = b$ or $a = f(b))$ and $(c = d$ or $c = f(d))$, else

$\langle \perp, \perp\rangle$.

Suppose there is a PPT adversary A that can take as input $\tau(x, y), y$ and return x' such that $\tau(x', y) = \tau(x, y)$ with non-negligible probability. Let $x = \langle f(i), i\rangle$ and $y = \langle f(i), f(i)\rangle$ for a random $i \in domain(f)$. We give A input $(\tau(x, y), y) = (y, y)$ and with non-negligible probability, she returns $x' = \langle f(i), i'\rangle$, where $f(i') = f(i)$. This contradicts the one-wayness of f. Thus, no such adversary A can exist and τ is a strongly associative one-way function. □

Chapter 6

Conclusion

We summarized the main contributions of this thesis in Section 1.2. In this chapter, we wish to highlight several exciting open problems relating algebraic structures and homomorphisms to cryptographic primitives and exploring the complexity of transitive signature schemes.

6.1 Future Directions

We close this thesis with a list of the nine most interesting, but unsolved problems discussed in this work.

1. REALIZE A TGII OR GII GROUP UNDER A REASONABLE ASSUMPTION. In Section 4.1, we introduced the notion of groups with infeasible inversion (GII). In Chapter 5, we saw that TGII groups (GII groups with a trapdoor) are strictly stronger than public key encryption and oblivious transfer; and that showing the impossibility of TGII groups immediately proves the impossibility of DTS schemes. A proof of the existence of GII groups immediately offers a construction or proves the impossibility of constructing a union-only signature scheme, as noted by Molnar [43]. We postulate the existence of these groups, but the search to find them may be a long one. Since strongly associative one-way functions (SAOWF) were introduced by Rabi and Sherman in 1993, no SAOWF implementations have been proposed. This is perhaps because most implementations actually have group structure and in Theorem 5.8.2, we proved that a SAOWF (on a group) forms a GII.

Rivest pointed out that it may also be interesting to consider TGII and GII groups with a restriction on the properties of a group. For example, we might consider dropping the samplability or equality testing properties. What GII-based constructions are still possible?

2. REALIZE A PFG GROUP UNDER A REASONABLE ASSUMPTION. In Section 4.5, we introduced the notion of pseudo-free groups (PFG), where a group is pseudo-free if a PPT adversary can not distinguish it from a free group, given only black-box access. We show in Theorem 5.2.1 that an Abelian pseudo-free TGII group is sufficient for constructing secure DTS schemes. As noted in Section 4.5.1, it would be interesting to prove or disprove the existence of a PFG based on standard assumptions.

3. IS TGII STRICTLY MORE POWERFUL THAN TDP? In Theorem 5.3.1, we show that any TGII group can be used to construct a trapdoor permutation (TDP). Is the reverse true? We conjecture that the answer is no. TGII requires a trapdoor permutation group homomorphism: $p(x)p(y) = p(xy)$. In a totally random permutation, the knowledge of two points in the range will not help predict a third point with non-negligible probability; yet, it will for a homomorphism. The relation between primitive X and primitive X as a homomorphism requires exploration.

4. IS DTS STRICTLY MORE POWERFUL THAN UTS? Undirected transitive signature schemes (UTS) exist under RSA, discrete logarithms, and several other standard cryptographic assumptions. However, no directed transitive signature schemes (DTS) have been proposed. In this thesis, we prove that DTS schemes are at least as complex as trapdoor permutations, typically considered to be one of the most powerful primitives, while digital signatures are equivalent in complexity to one-way functions, typically considered the most basic primitive. Our work, unfortunately, does not pin down the complexity of UTS schemes, other than to say that they are no greater than DTS and no less than standard signatures. It seems intuitively that directed edges add complexity, but we leave this problem open.

5. IS UTS STRICTLY MORE POWERFUL THAN OWP? Many distinguished researchers

were able to propose UTS constructions under specific complexity assumptions that are traditionally categorized as one-way permutations. For example, Bellare and Neven [7] showed how to build a UTS scheme using the hardness of taking square roots modulo a composite *without* knowing the factorization. Thus, it seemed intuitively as though one might be able to show a black-box reduction from OWP to undirected transitive signatures. Yet, no such proof has emerged.

We conjecture that showing an oracle separation between UTS and OWP (and thus between undirected transitive signatures and standard digital signatures) is possible using the same oracle and techniques of Impagliazzo and Rudich [32]. We spent a good deal of effort trying to show this proof ourselves, but the difficulty came from restricting how the master signer must use (and relate) the public and secret information, for himself and for each node.

6. IS HOMOMORPHIC ENCRYPTION STRICTLY MORE POWERFUL THAN PUBLIC KEY ENCRYPTION? Recall that a homomorphic encryption scheme is a scheme that allows for computation on encrypted data. An algebraically homomorphic encryption scheme is a homomorphic encryption scheme on a field with operations $+$ and $*$ that allows anyone, without knowledge of the decryption key, to compute $E(x+y)$ and $E(xy)$ from $E(x), E(y)$ [5]. Homomorphic encryption schemes satisfying $+$ or $*$ have been realized. For example, RSA satisfies the definition with respect to $*$. Boneh and Lipton proved that any deterministic algebraically homomorphic encryption scheme can be broken in subexponential time [13]. It is still unknown if a randomized algebraically homomorphic encryption scheme exists.

   In Corollary 4.2.2, we saw that some homomorphic signature schemes can not be constructed out of standard digital signatures. Since public key encryption (PKE) is strictly more powerful than digital signatures, it would be interesting to know if there is also a complexity separation between algebraically homomorphic encryption and PKE. Applying the technique of reverse cryptography to algebraically homormorphic encryption might also yield interesting results.

7. WHAT ARE THE COMPLEXITY RELATIONS FOR PRIMITIVES ON GROUPS? We touch

the surface of this topic by introducing several group-based primitives (GII, OWGH, OWGI), and discussing the relationship between strongly associative one-way functions and GII groups. However, a deeper exploration into the power of adding group structure to the common cryptographic primitives is warranted. Does adding a group to a primitive enable new applications? When is there provably no power gain? What about considering fields?

8. WHAT ARE REALIZABLE RELAXATIONS OF DTS? We demonstrated that DTS is equivalent to Abelian TGII, a group structure not known to exist. However, there may be some realizable weakenings of the definition. What static DTS schemes are possible, where the master signer takes an entire graph as input before being asked to sign any edges? What related homomorphic signature schemes can be realized, such as delegateable and proxy signatures?

   In Chapter 3, we claimed that the commutative property was a natural one for transitive signatures, since all current UTS proposals have commutative edge signatures. It would be interesting if anyone could proposed a secure UTS construction where the edge signatures do not commute.

Appendix A

Index of Acronyms

For brevity, this work contains a large number of acronyms. We provide a complete list of acronyms, their meanings, and where to find additional information on them.

Abbreviation	**Stands For?**	**Defined In?**
AGII	Abelian Group with Infeasible Inversion	Section 4.1
ATGII	Abelian Trapdoor Group with Infeasible Inversion	Section 4.1
BL	Bilinear Map	Chapter 2
CR	Collision Resistant	Chapter 2
Comp	Composition Algorithm	Section 3.1
DTS	Directed Transitive Signature	Section 3.1
ESign	Edge Signing Algorithm	Section 3.1
GII	Group with Infeasible Inversion	Section 4.1
KA	Key Agreement	Chapter 2
KG	Key Generation Algorithm	Section 3.1
ML	Multilinear Form	Chapter 2
NCert	Node Certification Algorithm	Section 3.1
NIGH	Non-Injective Group Homomorphism	Section 4.3
OT	Oblivious Transfer	Chapter 2
OMI	One-More Inversion Problem	Section 4.4.1

OWF	One-Way Function	Chapter 2
OWGA	One-Way Group Automorphism	Section 4.4
OWGI	One-Way Group Isomorphism	Section 4.4
OWGH	One-Way Group Homomorphism	Section 4.3
OWNIGH	One-Way Non-Injective Group Homomorphism	Section 4.3
OWP	One-Way Permutation	Chapter 2
PPT	Probabilistic Polynomial Time	Chapter 2
PFATGII	Pseudo-free Abelian Trapdoor Group with Infeasible Inversion	Sections 4.1, 4.5
PFG	Pseudo-free Group	Section 4.5
PFGII	Pseudo-free Group with Infeasible Inversion	Sections 4.1, 4.5
PFTGII	Pseudo-free Trapdoor Group with Infeasible Inversion	Sections 4.1, 4.5
PRG	Pseudorandom Generator	Chapter 2
PK	Public Key	Section 3.1
PKE	Public Key Encryption	Chapter 2
RSA	Public Key Encryption Scheme by Rivest, Shamir and Adleman	Chapter 1, [50]
SK	Secret Key	Section 3.1
SFE	Secure Function Evaluation	Chapter 2
STI	Single-Target Inversion Problem	Section 4.4.1
SDS	Standard Digital Signature	Chapter 2
SAOWF	Strongly Associative One-Way Function	Section 4.6
TGII	Trapdoor Group with Infeasible Inversion	Section 4.1
TDP	Trapdoor Permutation	Chapter 2
UTS	Undirected Transitive Signature	Section 3.1
Vf	Verification Algorithm	Section 3.1
WCR	Weakly Collision-Resistant	Chapter 2
WCRNIGH	Weakly Collision-Resistant Non-Injective Group Homomorphism	Section 4.3

Bibliography

[1] A. A. Albert. *Studies in Modern Algebra.* The Mathematical Association of America, 1963.

[2] Michael Artin. *Algebra.* Prentice-Hall Inc., 1991.

[3] Boaz Barak. Delegateable Signatures, 2001. Technical Report, Weizmann Institute of Science. Available from http://www.wisdom.weizmann.ac.il/~boaz/Papers/delgsigs.ps.

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

[5] Eberhard Becker, Dorte Rappe, and Tomas Sander. Homomorphic encryption on non-abelian groups and applications, 2002. Preprint available at http://www.tcs.hut.fi/~helger/crypto/link/public/homomorphic.html.

[6] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. In *Financial Cryptography 2002, Lecture Notes in Computer Science*, volume 2339, pages 319–338, 2002.

[7] Mihir Bellare and Gregory Neven. Transitive Signatures based on Factoring and RSA. In *ASIACRYPT 2002, Lecture Notes in Computer Science*, volume 2501, pages 397–414, 2002.

[8] Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO 2002, Lecture Notes for Computer Science*, volume 2442, pages 162–177, 2002.

[9] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[10] Alina Beygelzimer, Lance A. Hemaspaandra, Christopher M. Homan, and Jorg Rothe. One-way functions in worst-case cryptography: Algebraic and security properties. Technical Report TR722, University of Rochester, 1999.

[11] Garrett Birkhoff and Thomas C. Bartee. *Modern Applied Algebra.* McGraw-Hill Book Company, 1970.

[12] M. Blaze and M. Strauss. Atomic proxy cryptography. Technical Report TR98.5.1, AT&T research laboratories, 1998.

[13] D. Boneh and R. J. Lipton. Algorithms for black-box fields and their application to cryptography. In *CRYPTO 1996, Lecture Notes for Computer Science*, volume 1109, pages 283–297, 1996.

[14] D. Boneh and A. Silverberg. Applications of Multilinear Forms to Cryptography, 2002. Available from http://eprint.iacr.org.

[15] Dan Boneh and Matt Franklin. Identity-based Encryption from the Weil Pairing. In *CRYPTO 2001, Lecture Notes in Computer Science*, volume 2139, pages 213–225, 2001.

[16] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures. In *Eurocrypt 2003, Lecture Notes for Computer Science*, volume 2656, pages 416–432, 2003.

[17] E. F. Brickell and Y. Yacobi. On privacy homomorphisms. In *EUROCRYPT 1987, Lecture Notes for Computer Science*, volume 304, pages 117–126, 1987.

[18] Suresh Chari, Tal Rabin, and Ronald Rivest. An efficient signature scheme for route aggregation, 2002. Available from http://theory.lcs.mit.edu/~rivest/ChariRabinRivest-AnEfficientSignatureSchemeForRouteAggregation.ps.

[19] Jung Hee Cheon and Dong Hoon Lee. Diffie-Hellman problems and bilinear maps. Available from http://citeseer.nj.nec.com/cheon02diffiehellman.html.

[20] United States Congress. Electronic Signatures in Global Commerce Act. H.R. 1714.

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

[22] Yevgeniy Dodis and Anca Ivan. Proxy cryptography revisited. In *Network and Distributed System Security Symposium*, 2003.

[23] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *FOCS 2000, 41st IEEE Symposium on the Foundations of Computer Science*, pages 325–335, 2000.

[24] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *FOCS 2001, 42nd IEEE Symposium on the Foundations of Computer Science*, pages 126–135, 2001.

[25] Oded Goldreich. *Foundations of Cryptography.* Cambridge University Press, 2001.

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

[27] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computing*, 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, University of Rochester, 1998.

[29] Lane A. Hemaspaandra, Kari Pasanen, and Jorg Rothe. If P≠NP then some strongly noninvertible functions are invertible. Available from http://citeseer.nj.nec.com/hemaspaandra00np.html.

[30] Alejandro Hevia and Daniele Micciancio. The provable security of graph-based one-time signatures and extensions to algebraic signature schemes. In *ASIACRYPT 2002, Lecture Notes for Computer Science*, pages 379–396, 1996.

[31] Christopher M. Homan. Low ambiguity in strong, total, associative, one-way functions. Technical Report TR734, University of Rochester, 2000.

[32] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *STOC 1989, 21st Annual ACM Symposium on Theory of Computing*, pages 44–61, 1989.

[33] Integrity Sciences Inc. Elliptic curve cryptography. http://world.std.com/~dpj/elliptic.html.

[34] Nathan Jacobson. *Lectures in Abstract Algebra.* D. Van Nostrand Company, Inc., 1951.

[35] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In *CT-RSA, Lecture Notes in Computer Science*, volume 2271, pages 244–262, 2002.

[36] Antoine Joux. A one-round protocol for tripartite Diffie-Hellman. In *ANTS-IV conference, Lecture Notes in Computer Science.*, volume 1838, pages 385–394, 2000.

[37] Ki Hyoung Ko, Doo Ho Choi, Mi Sung Cho, and Jang Won Lee. New signature scheme using conjugacy problem.

[38] Rudolf Lidl and Harald Niederreiter. *Finite Fields.* Cambridge University Press, 1987.

[39] Helger Lipmaa. Cryptography and braid groups. http://www.tcs.hut.fi/~helger/crypto/link/public/braid/.

[40] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[41] Silvio Micali and Ronald L. Rivest. Transitive signature schemes. In *CT-RSA, Lecture Notes in Computer Science*, volume 2271, pages 236–243, 2002.

[42] Silvio Micali and Phillip Rogaway. Secure computation. In *CRYPTO 1991, Lecture Notes for Computer Science*, volume 0576, pages 392–404, 1991.

[43] David Molnar. Homomorphic Signature Schemes, Honors Thesis, Harvard College, April 2003.

[44] George D. Mostow, Joseph H. Sampson, and Jean-Pierre Meyer. *Fundamental Structures of Algebra.* McGraw-Hill Book Company, 1963.

[45] 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.

[46] 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.

[47] Michael Rabin. How to exchange secrets by oblivious transfer. Technical Report Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[48] Wolfram Resarch. Eric Weisstein's world of Mathematics, A Wolfram Web Resource. http://mathworld.wolfram.com.

[49] Ronald Rivest. Two signature schemes, 2000. Available at http://theory.lcs.mit.edu/~rivest/Rivest-CambridgeTalk.pdf.

[50] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digitial signatures and public-key cryptosystems. *Communications of the ACM*, 21,2:120-126, Feb. 1978.

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

[52] Jorg Rothe and Lane A. Hemaspaandra. Characterizations of the existence of partial and total one-way permutations. In *Information Processing Letters, Vol 82, Issue 3*, pages 165–171, 2002.

[53] Steven Rudich. Limits to the provable consequence of one-way functions, PhD Thesis, UC-Berkeley, 1991.

[54] Ahmad-Reza Sadeghi and Michael Steiner. Assumptions related to discrete logarithms: Why subtleties make a real difference. In *Eurocrypt 2001, Lecture Notes in Computer Science*, volume 2045, pages 244–262, 2001.

[55] Steve Simpson. The role of reverse mathematics. http://www.math.psu.edu/simpson/papers/hilbert/node5.html.

[56] Howard Straubing. When can one monoid simulate another? *Algorithmic Problems in Groups and Subgroups*, 2000. http://citeseer.nj.nec.com/144962.html.

[57] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *STOC 1982, 14th Annual ACM Symposium on Theory of Computing*, pages 80–91, November 1982.

[58] Andrew C. Yao. Computational information theory. In *Complexity in Information Theory*, pages 1–15, 1988.

[59] Huafei Zhu, Bao Feng, and Robert H. Deng. Computing trust in distributed networks, 2003. Available on eprint: 2003/056.

[60] Huafei Zhu, Bao Feng, and Robert H. Deng. A transitive signature scheme provably secure against adaptive chosen-message attack, 2003. Available on eprint: 2003/059.