Secret-chain Zero-Knowledge Proofs and Their Applications

by

Tony Liang Eng

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

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1994

Author .....
Department of Electrical Engineering and Computer Science
May 5, 1994

Certified by ..
Shafi Goldwasser
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

Secret-chain Zero-Knowledge Proofs and Their Applications

by

Tony Liang Eng

Submitted to the Department of Electrical Engineering and Computer Science
on May 13, 1994, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

In zero-knowledge proofs of knowledge, a single prover tries to convince a single verifier that he has possession of some knowledge s. The verifier accepts with probability 1 if the prover is honest (completeness), and rejects with high probability otherwise (soundness). In either case, the verifier learns nothing about s other than the bit of information as to whether or not the prover knows s (zero-knowledgeness).

In [OkOh], Okamoto and Ohta introduce the notion of divertibility, a certain zero-knowledge proof of knowledge scenario in which the verifier is in turn able to disseminate the prover's proof of knowledge and even convince others that he knows some secret s when in reality, this information is known only to the prover. There are both positive and negative aspects to this scenario. The immediate complaint of this situation is that the verifier is know able to profess and claim knowledge of something he doesn't really know, and in applications like identification schemes, this is definitely undesirable. However, [OkOh] and [BuDeItSaSh] have ventured forth useful applications divertible proofs, namely for untraceability, blind signatures and subliminal-free transmissions.

In this thesis, we generalize this idea of divertibility by introducing and exploring the *secret-chain* model, a proof system characterized by the existence of multiple provers, arranged in a linear fashion. In this manner, each prover is only allowed to interact with the provers who are his immediate neighbors. In addition, each prover knows some secret information that can be part of a larger "super-secret". If we replace the chain of provers with a single "super-prover", then this super-secret is knowledge this super-prover claims to know and can exhibit a proof of knowledge for. We define a model and formulate these ideas further assuming the existence of a commutative random self-reducible (CRSR) relation that is both one-way and non-trapdoor.

Secret-chains appear useful for applications in which privacy protection and untraceability are desired. For example, consider an information broker/consultant who furnishes clients with information. He may assemble and collate information from his own personal resources and from one of many information banks or sources that

are willing to exhibit a zero knowledge proof for information they know provided the price is right. Naturally, he wishes to keep his sources of information secret from his customers, and furthermore, he may not want his sources to know what he is doing with their information, so prefers that the identity of his clientele remains unknown to his sources.

The use of secret-chains may be useful for situations in which the presence of an overseer is desired to monitor some party in the system, so that this party cannot function properly without the overseers consent. Furthermore, secret chains may not be limited to proofs of knowledge, but might have possible applications to proofs of computational power. In this thesis, we discuss several possible applications, and construct a blind signature scheme based on a secret-chain. Lastly, we exhibit a secret-chain zero-knowledge proof for Graph Isomorphism.

Thesis Supervisor: Shafi Goldwasser
Title: Professor of Computer Science and Engineering

Acknowledgments

In Christ alone,
I place my trust, and find my glory in the power of the cross.
In every victory, let it be said of me,
my source of strength,
my source of hope,
is Christ alone.

– Shawn Craig, Don Koch
Lyrics, *In Christ Alone*
1990, Paragon Music Corp

Before closing another chapter at MIT, I would like to acknowledge a few people who have influenced and touched my life and made this thesis possible. I am grateful to Dr. Tatsuaki Okamoto and Dr. Kazuo Ohta for the opportunity to intern at NTT where I began this work, to conduct research in a comfortable, interactive and friendly environment, and to visit the beautiful country of Japan. *Doomo Arigatoo Gozaimashita!* And to Professor Shafi Goldwasser for her patience, expertise and many helpful comments as we waded through drafts of this thesis.

I am thankful for the guys at the Christian Servicement Center, where I enjoyed the rank of CC aboard the U.S.S. NeverSail, newest addition to the Pacific Fleet stationed at Yokosuka, Japan! And especially for Larry and Evelyn Bentley who made my stay in Japan enjoyable and warm.

And to fellow MIT students who have crossed my path and made a lasting impression. These include those who have already left these hallowed halls (Mona, Kiet, Jen, Chester, Jeff...), those who still roam it (Victor, Nicole, Ona, Patrick,...), and those who have just recently arrived (my 6.001 kids who have probably heard more about the 'T' word than they could have ever dreamed or imagined!)

And also to my brothers and sisters at UCF who leave me with many fond memories. Thanks for your friendship, encouragement, prayers, hugs, snacks and zwrites to

keep me awake. I am also grateful for my cell group: Ien, Joel, Howard and Norman, who have provided me with time to relax, refocus and rejuvenate throughout the semester. Thanks Jim, for helping me with my bibliography, and thanks Andrew, for coming to MIT. It's been a great semester!

Last, but not least, I am most indebted to my parents. Only with their love, understanding, support and encouragement, could I have made it this far. This college debt is one that can never be repaid.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Before formalizing secret chains and discussing their relationship to zero-knowledge and divertibility, we sketch an illustration that is not completely precise or accurate, but should suffice to afford the reader an introductory impression and glimpse of secret chains.

1.1 An Illustration

We return to the neverending saga of two memorable pioneers in cryptography by adding yet another chapter to the exploits of the ever inquisitive Alice and the mischievous Bob. This time, our heroes have joined the United States Navy, where Alice is no longer just plain "Alice" but "Commander Alice", and Lieutenant Commander Bob is a soon-to-be commander as well. They have discovered that the Navy too has been blessed with the modern benefits and grace befitting to a bureaucracy, and as a result, possesses a very strict hierarchical rank structure.

There are two types of commands that an officer may issue – those that are granted automatically because they come with his rank, and those that require approval from his superiors. Thus, each rank comes with certain delegated powers and privileges, and an officer is authorized to approve these. Any commands that do not fall in his jurisdiction of authority cannot be issued without clearance from a higher official.

All commands are represented by public keys, and on promotion to a higher rank,

an officer is bestowed the set of secret keys corresponding to powers that accompany his new station. In this manner, an officer possesses the secret key for all commands he is authorized to issue, and consequently, he can exhibit a proof of knowledge of this secret as an indication of his approval for the fulfillment command. Note that here, the officer knows the secret key in its entirety, in the case of a command that requires intervention by a higher authority, the officer holds only a portion of the corresponding secret key.

This is a very reasonable and natural command structure, and we see it at work when Lieutenant Commander Bob desires to take a short leave of absence 7 days to attend the annual Crypto conference in August and visit the casinos of Las Vegas to test the laws of probability. Before any bags can be packed however, he must first secure permission by petitioning the next officer of higher rank, Commander Alice. Since Alice is Bob's direct superior, she can choose to either outright reject the request, or go along with it. Assuming she is not averse to his petition, once finding that her rank comes with the authority to grant this request, Alice exhibits a proof of knowledge of the proper secret key to, in effect, grant his request.

Having successfully obtained permission for some time off, LC Bob also wishes to inquire about his promotion papers, and once again, approaches his superior officer, Alice. Alice supports his bid for promotion; however, she does not have the authorization to ratify it since she holds only a portion of the corresponding secret key while higher agents hold the remaining portions. In order to issue commands of this nature, she must in turn clear it first with *her* own superior officer, Captain Eve.

The process repeats once again. Captain Eve can exercise her option of denial and immediately refuse the request. On the other hand, if she consents with the proposed action, she too checks if she has clearance to approve it. If she does, she can supply Alice with the needed proof, and no other personages in the hierarchy need to be consulted. If she also doesn't have the authority, then Bob's request proceeds up the line of command in a manner similar to that described until someone disapproves of the request, or until it reaches someone who is vested with the proper authority. When this someone is finally reached, a proof of knowledge for the final part of the

secret key is obtained. This proof is then passed back down the chain in a manner so that each agent in the ladder uses his share of the secret in combination with the proof exhibited by his superior officer to create his *own* authorization for the action. When Bob finally receives his proof approving his promotion, this proof will be a proof for the entire corresponding secret key; the fact that this is a proof for the entire secret key signifies that all holders of the various portions have endorsed their support for the request by incorporating their share of the secret into the proof.

The fact that Alice needed to get approval from a higher authority is hidden from Bob; in this case, Bob may believe that Alice was completely authorized to approve the request, since she was able to supply a proof for the complete secret key of the corresponding public key in question. Furthermore, note that the "height" of the ladder (i.e. how far up the chain of command the request went) is unknown to Bob (and to everyone else in the ladder except the highest officer reached). Thus, in short, as far as he is concerned, his request was either directly rejected or wholly approved by his immediate commanding officer, Alice.

Although there are subtleties that this illustration does not address, this is the basic intuition behind a secret-chain, and the linear chain of officers whose approval had to be secured for Bob's request is essentially an example of a secret-chain. To set the stage for our work, let us first briefly examine previous relevant research before describing the details of a secret-chain zero-knowledge proof.

1.2 Overview

Interactive proofs of membership were first introduced by Goldwasser, Micali and Rackoff [GoMiRa]. Such proofs allowed an unbounded probabilistic prover to prove to a probabilistic polynomial-time verifier the membership of some $x \in L$. Informally, the verifier would accept correct proofs with high probability (probability of 1), and would reject in cases where $x \notin L$. The additional characteristic of being zero-knowledge restricted the amount of information divulged to the verifier as a result of the proof – aside from learning about the membership of x in L, the verifier

should learn nothing that it could not have computed itself (in polynomial time). Zero-knowledge interactive proofs of "knowledge" [ToWo, FeFiSh, BeGo] and zero-knowledge interactive proofs of "computational power" [Yu, OkChOh] are similar variations in which the prover demonstrates possession of some information and the ability to solve some problem respectively. In this thesis, we will be concerned with zero-knowledge proofs of knowledge.

Consider a function with the property that if it is hard, then it is uniformly hard. Theses were described by Angluin and Lichtenstein [AnLi] as being random self-reducible(RSR), and have appeared elsewhere in the literature [GoMi, BlMi]. Examples of RSR functions are based on factoring, discrete logarithms, RSA and quadratic residuosity modulo a composite. Tompa and Woll [ToWo] related random self-reducible problems with zero-knowledge interactive proofs(ZKIP) by showing that a ZKIP existed for any RSR problem, and they provided constructions for graph isomorphism, quadratic residuosity modulo a composite, and the discrete logarithm problems.

In [OkOh], the notion of a divertible ZKIP of knowledge is introduced and defined. Informally, a ZKIP of knowledge (P, V) between a prover P and a verifier V, is diverted by a third party W if W engages P in (P, W) and transparently uses the conversation from (P, W) to participate in a second (simultaneous) protocol (W, V) (such that one can discern no unique connection between the conversation from (P, W) and that of (W, V)). Using this proof (W, V), W can convince V that W "knows"(notion defined in Chapter Two) some knowledge P knows, when he fact, he does not.

In addition, Okamoto and Ohta focused on a subset of RSR problems for which a commutative law of composition existed. Calling these commutative random self-reducible(CRSR) problems, they showed in [OkOh] that a divertible ZKIP existed for all CRSR problems. The work of Burmester and Desmedt [BuDe] expanded this front and proved that all languages in NP possess divertible ZKIPs assuming the existence of probabilistic encryption homomorphisms.

The secret-chain model to be described in this thesis is reminiscent of the divertibility scenario. Our work generalizes the notion of a divertible ZKIP in that a

zero-knowledge proof of knowledge given by a single prover can be shared, diverted and replayed down a chain of multiple verifiers. However, in our scenario, the text of the conversation between prover W and verifier V in (W, V) is not simply an "untraceable" version of the conversation from (P, W) – W does not pretend to be P by advertising knowledge of P's secret. Instead, W professes knowledge of a greater secret (call it $\tilde{x}$), built upon that of P. But since W actually does not possess P's secret (P's portion of $\tilde{x}$), he must rely on interaction with P whenever a proof for knowledge of $\tilde{x}$ is needed.

In short, this work with secret-chains stems from consideration of the other end of the spectrum – namely, from examining the consequences and possibilities that result from having a linear chain of multiple provers who are collectively responsible for creating this zero-knowledge proof of knowledge.

The initial reaction to havoc that may result from divertible proofs is understandably negative, and rightly so in the light and context of many cryptographic situations; however, as [OkOh] and [BuDeItSaSh] describe, there are positive applications for divertibility as well. As in their papers, we draw upon these positive aspects in our generalization to secret chains.

Secret-chains appear useful for applications in which privacy protection and untraceability are desired. For example, consider an information broker or consultant who furnishes his clients with information. This information may be collected and collated from his own personal resources and from one of several information banks or other sources. Information is provided by a source in the form of a zero knowledge proof of knowledge of this information, in return for monetary compensation. The consultant then uses this proof and augments it, so that the end result is a proof of knowledge for the entire collection of information he is to provide. Naturally, he wishes to keep his sources of information secret from his customers, and furthermore, he may not want his sources to know what he is doing with their information, so prefers that the identity of his clientele remains unknown to his sources.

This idea of multiple provers is not new. A "distributed" linear arrangement of two provers and one verifier as a possibility for digital signatures in a new approach

to electronic currency in the observer model [BrChCrFePe, Ch2, PeCh]. In this situation, each consumer has a tamper-proof smartcard that represents his electronic wallet. Resident on each card is an "observer", a processing unit trusted by the bank. In order to spend any of his money which is stored on this card, a consumer must first obtain "permission" from his observer. This permission is basically willingness on the part of the observer to participate in a digital signature algorithm in which the secret key is partitioned between the consumer and the observer. Thus, the observer can oversee the consumer's transactions and monitor his behavior, prohibiting him from cheating and double spending any of his electronic coins by refusing to partake in the digital signature scheme. Protocols including those for setting up the observer/consumer digital signature scheme, withdrawing and spending coins, have already been proposed in these papers and others.

Recently, it was brought to our attention that the concept of multiple provers had also been discussed by Guillou and Quisquater [GuQu]. In each of the two scenarios they propose and describe, there are two entities (provers). In the first setting, two entities with different identities are interacting with a verifier, and it is known to the verifier that there are in fact two provers participating in the protocol. In the other setting, the verifier believes he is interacting with a single prover, but behind the scenes, there are actually two provers. This is more in line with our notion of secret-chains; however, both provers of their scenario realize that they are simulating this larger single prover *together*. In secret-chains, developed independently from their work, the multiple provers do not enjoy this same piece of knowledge. The role of secret-chain provers are more asymmetric and there is some feature of the protocol unknown to all parties: (1) a prover never knows if the secret it is proving knowledge of is part of a larger one and if its proofs are being used elsewhere; (2) verifiers believe they are interacting with a single prover that knows the secret it is proving in its entirety, and are not aware of the actual number of provers behind the curtain; and (3) intermediate parties (whom we shall see play prover and verifier-like roles), including Alice and our information broker, face both of these unknowns. The exact relationship between our work and that of [GuQu] deserves further investigation and

exploration at a later time.

The organization of this thesis is as follows: Chapter Two provides some helpful notation and definitions of zero-knowledge proofs and CRSR relations; Chapter Three presents a definition of secret-chains and shows how a secret-chain ZKIP can be constructed given the existence of a CRSR relation R; Chapter Four extends this work to include blind signatures (defined in Chapter Four) that are signed by multiple parties; a secret-chain ZKIP for Graph Isomorphism appears in Chapter Five; and finally Chapter Six discusses some future work and concludes this thesis.

Chapter 2

Notation and Background

In this chapter, we present some notation before describing relevant previous work and formally defining secret-chains and related protocols.

An interactive protocol is modelled by an ordered pair (A, B) of probabilistic polynomial-time Turing machines(PTM) A and B which take turns being active for a polynomial number of rounds. Each machine has an extra read-only tape and an extra write-only tape that serve as a means for communication during each round – the write-only tape of one machine serves as a read-only tape for the other and vice versa. Denote $(A(m), B(n))(x)$ as the interactive protocol between A and B where m and n are the private inputs of A and B respectively, and x is input common to both.

Machine A initiates the protocol and they alternate being active, exchanging messages with each other. The protocol terminates when either A rejects, or B enters a reject or accept state. We say that B *accepts(rejects)* $(A(m), B(n))(x)$ if the protocol terminates, leaving B in an accepting(rejecting) state. Note that either party can terminate and reject the protocol by entering a rejecting state if messages sent by the other party are not valid – i.e. they deviate from the prescribed protocol. Any output, which may be either private to a party or shared by both machines, is written on the appropriate write-only tape of the machine in question.

Both machines keep a history of the interaction and for each round of messages, this history aids them in ascertaining the state of the protocol and in deciding a

subsequent course of action.

The *view* of a PTM consists of all the quantities the PTM sees during an interactive protocol (i.e. its coin flips, its private computation, quantities from tapes that it has access to). For the protocol $(A(m), B(n))(x)$, we denote A's view and B's view as $\text{VIEW}_A((A(m), B(n))(x))$ and $\text{VIEW}_B((A(m), B(n))(x))$ respectively.

Let the following triple of probabilistic polynomial-time turing machines (A, B, C) represent a three party interactive protocol in which interaction occurs only between A and B, and between B and C. We adopt the following notation from Itoh, et. al. [ItSaSh]. Let $(A, B^{\leftrightarrow C})$ denote the interactive protocol between a prover A and a verifier B, where a second simultaneous interactive protocol occurs in which B is a *prover* and C is a verifier. We will adopt flexibility in this graphical notation so that $({}^{A\leftrightarrow}B, C)$ then represents the interactive protocol between prover B and verifier C, in which B is also the verifier in a simultaneous interactive protocol with prover A. Note that $(A, B^{\leftrightarrow C})$ and $({}^{A\leftrightarrow}B, C)$ can actually refer to the same protocol, but from the perspectives of different parties involved.

For turing machines X and Y, let X^Y denote X using Y as a blackbox subroutine, where X is able to save away Y's computation state, rewind Y's input tapes, etc.

Let $|x|$ denote the length of the binary representation of x.

The notation $x \in_R X$ means that x is randomly and uniformly selected from X. Also, let $||$ denote concatenation and let $|x|$ be the length of the binary representation of x. If b is a bit, let $\overline{b}$ represent the negation of b.

Definition 2.0.1 *Let $\mathcal{N}$ be a countable infinite set. For any $N \in \mathcal{N}$, let X_N, Y_N be finite sets that can be easily sampled. Let R_N be a binary relation $\{(x,y)|x \in X_N, y \in Y_N\}$ testable in polynomial time.*

2.1 Interactive Proofs of Knowledge

We present a brief description of interactive proof systems for polynomial-time relations, and define what it means to be zero-knowledge. Finally, we review the idea of divertible zero-knowledge interactive proofs.

In interactive proofs of knowledge [ToWo, FeFiSh, BeGo], the prover demonstrates the possession of certain information. In particular, assuming there exists a binary relation R_N and an $x \in X_N$, we will focus on interactive proofs systems for relation R_N in which the prover demonstrates that he "knows" a y such that $(x, y) \in R_N$.

In particular, we will borrow the definition of interactive proofs for relations and zero-knowledge from Feige and Shamir's work [FeSh]. Let $\nu(n)$ be any function vanishing faster than the inverse of of any polynomial; namely, $\forall k, \exists n > N$ such that $v(n) < \frac{1}{n^k}$.

Definition 2.1.1 *Let $(P(y), V)(x)$ be a protocol between a pair of interacting PTM, prover P and verifier V, where x is common input and y is P's private input. Then $(P(y), V)(x)$ is an* **interactive proof for relation** *R_N that P "knows" a y such that $(x, y) \in R_N$ if:*

- *(completeness) $\forall (x, y) \in R_N$, Prob(V accepts $(P(y), V)(x)$) $> 1 - \nu(n)$ where the probability is taken over the coins of P and V.*
- *(soundness) There exists a polynomial-time knowledge extractor E such that $\forall P', \forall x \in X_N, \forall y'$,*

  Prob(V accepts $(P'(y'), V)(x)$) $<$ Prob($E^{P'(y')}$ outputs y'' such that $(x, y'') \in R_N$) $+ \nu(n)$, where the probabilities are taken over the coins of P', V and E.

The completeness condition specifies that whenever P is honest and does in fact "know" a y such that $(x, y) \in R_N$, the verifier accepts with an overwelming probability of $1 - \nu(n)$. The soundness property states that for any prover P', if participation in $(P'(y'), V)(x)$ leaves V in an accepting state, then there exists a polynomial-time knowledge extractor $E^{P'}$ that outputs y'

In the case when there is only a unique $y \in Y_N$ for every $x \in X_N$, if the proof $(P'(y'), V)(x)$ is accepted by V the knowledge extractor outputs exactly this y'.

Definition 2.1.2 *Let $(P(y), V)(x)$ be a protocol between a pair of interacting PTM, prover P and verifier V, where x is common input and y is P's private input.*

Then $(P(y), V)(x)$ is a (computational/statistical/perfect) **zero-knowledge** *interactive proof system for relation R_N that P knows a y such that $(x, y) \in R_N$ if the following additional property holds:*

- *(zero-knowledge) There exists a simulator SIM that runs in expected polynomial-time, such that $\forall$ PPT V', for any $(x, y) \in R_N$ and any auxiliary input s to V', $SIM^{V'}(x, s)$ is computationally/statistically/perfectly indistinguishable from $(P(y), V'(s))(x)$. Note that simulator SIM is allowed to use algorithm V' as a black box subroutine.*

2.2 Commutative Random Self-Reducibility

The idea of random self-reducibility has been previously mentioned in literature by Blum and Micali [BlMi] and later by Goldwasser and Micali [GoMi]. These papers offered examples based on discrete logarithms and quadratic residues modulo a composite respectively.

Tompa and Woll appear to be the first to formulate a definition in [ToWo]. And from this, Okamoto and Ohta developed the notion of a commutative random self-reducible relation.

Inspired by Okamoto and Ohta's definition, we present a modified version that explicitly specifies a function relating pairs in the relation. Note that a set can be "easily sampled" if an element of the set can be randomly chosen with a uniform distribution.

Definition 2.2.1 *Let $\mathcal{N}$ be a countable infinite set. For any $N \in \mathcal{N}$, let X_N, Y_N be finite sets that can be easily sampled and let $f_N : Y_N \rightarrow X_N$ be a one-way non-trapdoor polynomial-time computable function. The relation $R_{f_N} = \{(f_N(y), y) | y \in Y_N\}$ is (one-way nontrapdoor) commutative random self-reducible(CRSR) if there exists a polynomial-time algorithm $\mathcal{A}_N : X_N \times Y_N \rightarrow X_N$ such that when $x \in X_N$ and $r \in Y_N$,$x' = \mathcal{A}_N(x, r)$ and the following five properties hold:*

1. *If $r \in_R Y_N$, then $x' \in X_N$ is randomly and uniformly distributed.*

2. *Given x, r and y' where $x' = \mathcal{A}_N(x, r)$ and $(x', y') \in R_N$, there exists a polynomial-time algorithm to find $y \in Y_N$ such that $(x, y) \in R_N$.*

3. *There exists a law of composition, $\bullet : Y_N \times Y_N \to Y_N$ such that $(Y_N, \bullet)$ is a commutative group and if $(x, y) \in R_N$ and $x' = \mathcal{A}_N(x, r)$, then $y' = y \bullet r$ and $(x', y') \in R_N$.*

4. *There exists a PTM $\mathcal{T}$ such that if $x' = \mathcal{A}_N(x, r)$ for some $r \in Y_N$, then $\mathcal{T}(x, x')$ outputs x^* such that $(x^*, r^{-1}) \in R_N$.*

The definition tendered by Okamoto and Ohta had the following additional requirement:

> Given x, r and y where $x' = \mathcal{A}_N(x, r)$ and $(x, y) \in R_N$, there exists a polynomial-time algorithm to find $y' \in Y_N$ such that $(x', y') \in R_N$.

However, this follows from the law of composition in condition 3 of this definition; namely, given x, r and y where $x' = \mathcal{A}_N(x, r)$, then y' can be computed as $y \bullet r$.

Note that for a relation $R \subset X \times Y$, by $R_N \subset X_N \times Y_N$ we mean that the pairs in R_N are drawn from $X_N = \{x | x \in X, |x| = N\}$ and $Y_N = \{y | y \in Y, |y| = N\}$. And we often write R_N in place of R_{f_N}, since the function f is implicit in the relation. Also, by our formulation, for any $y \in Y_N$, there exists a unique $x = X_N$ (namely $x = f(y)$) such that $(x, y) \in R_N$.

We make explicit the function f relating pairs in R_N, and require f to be one-way and non-trapdoor. In other words, given any $x \in X_N$, it is difficult to find a $y \in Y_N$ such that $(x, y) \in R_N$. More formally, $\forall$ PTM C, $\forall x$ sufficiently large, $\forall d > 0$,

$$\text{Prob}(y \leftarrow C(x, R_N) | x \in X_N, y \in Y_N, (x, y) \in R_N) < \frac{1}{|x|^d},$$

where the probability is taken over the coins of C.

2.2.1 Example of CRSR Based on Discrete Logarithms

Let p be a prime such that $|p| = N$, and let g be a generator of Z_p^*. Then $Y_N = Z_{p-1}$ and $X_N = Z_p^*$. Let $f(y) = g^y \bmod p$ where $y \in Y_N$. The relation R_N is then defined to be $\{(x,y)|x = g^y \bmod p, y \in Y_N\}$, and

$$\begin{aligned} f(y) &= g^y \bmod p, \\ \mathcal{A}_N(x,r) &= xg^r \bmod p, \\ y_1 \bullet y_2 &= y_1 + y_2 \bmod \phi(p). \end{aligned}$$

If r is randomly chosen, then $xg^r \bmod p$ is randomly distributed in Z_p^*. Also, given x, r and y', we can find a value for y such that $x = g^y \bmod p$; namely, $y \equiv y' - r \bmod \phi(p)$. Furthermore, addition modulo p is a commutative group. Lastly, given $x' = xg^r \bmod p$, $\mathcal{T}(x,x')$ outputs $x^* = g^{-r} \bmod p$ since $(x^*, -r) \in R_N$. In this case, we can view $\mathcal{T}$ as finding an x^* such that $x^*x' \equiv x \bmod p$.

2.2.2 Example of CRSR Based on Quadratic Residues

Let $\mathcal{N}$ be a product of two primes, p and g where $|p| = |q| = N$. Let X_N be the set of quadratic residues in $Z_\mathcal{N}$, and let $Y_N = Z_\mathcal{N}^*$. Define $f(y) = y^2 \bmod \mathcal{N}$ where $y \in Z_\mathcal{N}$. Then, $R_N = \{(x,y)|x = y^2 \bmod \mathcal{N}, y \in Y_N)\}$, and

$$\begin{aligned} f(y) &= y^2 \bmod \mathcal{N}, \\ \mathcal{A}_N(x,r) &= xr^2 \bmod \mathcal{N} \\ y_1 \bullet y_2 &= y_1 y_2 \bmod \mathcal{N}. \end{aligned}$$

If r is randomly selected, then $xr^2 \bmod \mathcal{N}$ is a random quadratic residue modulo $\mathcal{N}$. Given x, r and y', we can find a value for y such that $x = y^2 \bmod \mathcal{N}$; namely, $y \equiv y'r^{-1} \bmod N$. Multiplication of elements of $Z_\mathcal{N}^*$ is a commutative group (identity

$= 1$, inverse exists since elements are relatively prime to modulus). Lastly, given $x' = xr^2 \bmod \mathcal{N}$, $\mathcal{T}(x, x')$ outputs $x^* = r^{-2} \bmod \mathcal{N}$ since $(x^*, r^{-1}) \in R_N$. In this case, we can view $\mathcal{T}$ as finding an x^* such that $x^*x' \equiv x \bmod \mathcal{N}$.

2.3 Divertibility

A divertible zero-knowledge interactive proof is a ZKIP of knowledge supplied by a prover to a verifier, that is in turn replayed by this verifier in subsequent interaction with a second verifier as a proof of knowledge when in fact, the verifier in reality does not possess this knowledge, but can falsely convince the second verifier that he does.

A more formal definition inspired by [OkOh] is as follows:

Definition 2.3.1 *Let $\mathcal{N}$ be a countable infinite set. For any $N \in \mathcal{N}$, let X_N, Y_N be finite sets that can be easily sampled. An interactive triple of probabilitistic polynomial-time turing machines $(A(y), B, C)(x)$ is a divertible (computational/perfect) zero-knowledge interactive proof to C that B knows some y satisfying $(x, y) \in R_N$, where $R_N \subset X_N \times Y_N$ is a relation, if the following conditions hold:*

1. *$(A(y), B^{\leftrightarrow C})(x)$ is a (computational/perfect) zero-knowledge proof for relation R_N that the prover A with private input y knows some y such that $(x, y) \in R_N$.*

2. *$({}^{A(y)\leftrightarrow}B, C)(x)$ is a (computational/perfect) zero knowledge proof for relation R_N that the prover ${}^{A(y)\leftrightarrow}B$ knows some y such that $(x, y) \in R_N$.*

3. *Let A, B and C be honest parties in the ideal scenario. Let A^* be any polynomial-time prover such that C does not reject $(A^*, C)(x)$ and let C^* be any polynomial-time verifier such that A does not reject $(A, C^*)(x)$, then $\forall$ polynomials Q, $\forall r, s \in \{0,1\}^{Q|x|}$,*

   - $\mathrm{VIEW}_A((A(y, r), (B \leftrightarrow C^*(s))(x))$ *is computationally indistinguishable from* $\mathrm{VIEW}_A((A(y, r), C(s))(x))$
   - $\mathrm{VIEW}_C(({}^{A^*(y,r)\leftrightarrow}B, C(s))(x))$ *is computationally indistinguishable from* $\mathrm{VIEW}_C((A(y, r), C(s))(x))$.

Note that r and s represent possible private auxiliary inputs of A and C respectively.

In [OkOh], Okamoto and Ohta then show the following theorem:

Theorem 2.3.2 *If R_N is a CRSR relation then on input x, then there exists a polynomial-time divertible ZKIP of knowledge $(A(y), B, C)(x)$ for a y such that $(x, y) \in R_N$.*

Chapter 3

The Secret Chain Model

3.1 High-level Description

Before formalizing the notion of a secret-chain, we hope to instill a clearer notion of the model by going yet another level deeper. Recall the illustration from Chapter One in which the issuance of certain commands required the intervention of two or more provers(officers) arranged in a linear fashion (hence the term, "chain"). Consider the following secret-chain of 3 parties:

$$P \leftrightarrow W \leftrightarrow V,$$

where P is a prover and V is a verifier. Here, the $\leftrightarrow$ symbol denotes direct interaction. Notice that W is able to interact with both P and V, but P and V never communicate with each other during the protocol.

In our setting, we consider the case when W plays the role of a prover. Each prover possesses some knowledge or secret that the others do not. Collectively, however, they all know disjoint parts of a larger secret[1] So, each of P and W has his own secret y_1 and y_2 respectively, however, these are in turn part of a greater secret $\tilde{y} = g(y_1, y_2)$ for some deterministic polynomial-time computable function g. If P and W are treated

[1]Note that we mean "larger" in a semantic conceptual sense, and not necessarily in the physical size of the secret.

collectively as a single party, then, only when they collaborate using their respective secrets, can P and W (taken as a unit) exhibit a proof of knowledge of $\tilde{y}$. In this model, V can obtain a zero-knowledge proof of secret $\tilde{y}$ so that, informally,

1. (completeness) if both P and W know legitimate portions that together form the secret $\tilde{y}$ (by applying g), V will accept the proof with high probability.

2. (soundness) if an honest verifier V accepts the proof, then there exists a "extractor" E that can extract the secret $\tilde{y}$ from P and W in polynomial time.

3. (zero-knowledge) W learns nothing about P's share of the secret y_1, and V learns nothing [2] about $\tilde{y}$.

4. (unsuspicious parties)

   - V is unable to tell that the secret $\tilde{y}$ is not completely known to W, and hence does not realize that W requires interacting with P in order to complete his proof.
   - P does not realize that his secret is part of a larger whole, and hence does not suspect that his conversation with W is being used elsewhere (i.e. as a basis for interaction between W and V).

In general, if we let $(P, W_1, ..., W_n, V)$ represent a secret-chain, the leftmost party of the chain will always be a "prover" and the rightmost party, a "verifier". The roles of the intermediate parties $(W_1, ..., W_n)$ will vary. The divertibility model of [OkOh] considered only a single prover and multiple verifiers, so W_i played a verifier role and simply diverted proofs. In our case, we consider the possibilities that occur when W plays a prover role, where in addition to diverting proofs, W must first incorporate his share of the secret into the proof before presenting the proof to the next party down the chain.

[2] As far as V is concerned, V does not know that the secret is partitioned among multiple provers, so it suffices to say that V learns nothing about $\tilde{y}$ (and hence nothing about either of P or W's shares).

3.2 Formalization

This section formally defines the intuition described above. For simplicity, we define secret-chains of three parties; the generalization to n parties can be easily made and is discussed later.

Definition 3.2.1 *Let $\mathcal{N}$ be a countable infinite set and let $N \in \mathcal{N}$. Let $R_N \subset X_N \times Y_N$ be a relation for finite sets X_N, Y_N. Let $g : Y_N \rightarrow Y_N$ be a polynomial-time computable function, and let $I(y_1; y_2)$ be information theoretic notation expressing the mutual information between y_1 and y_2.*

An interactive triple of probabilistic turing machines $(A(x_1, y_1), B(x_1, x_2, y_2, \tilde{x}), C(\tilde{x}))$ where $(x_1, y_1), (x_2, y_2), (\tilde{x}, g(y_1, y_2)) \in R_N$ and $I(y_1; y_2) = 0$ is a **secret-chain** *interactive proof of knowledge for relation R_N if the following conditions are satisfied:*

1. *$(A(y_1), B)(x_1)$ is an interactive proof for relation R_N that A knows some y_1 such that $(x_1, y_1) \in R_N$.*

2. *$({}^{A\leftrightarrow}B(x_2, y_2), C)(\tilde{x})$ is an interactive proof for relation R_N that ${}^{A\leftrightarrow}B$ knows some $\tilde{y}$ such that $(\tilde{x}, \tilde{y}) \in R_N$.*

In other words, $(A(x_1, y_1), B(x_1, x_2, y_2, \tilde{x}), C(\tilde{x}))(N)$ is a proof to B that A can compute some y_1 satisfying $(x_1, y_1) \in R_N$), and a simultaneous proof to C that ${}^{A\leftrightarrow}B$ can compute some $\tilde{y}$ satisfying $(\tilde{x}, \tilde{y}) \in R_N$.

Basically, conditions (1) and (2) describe the properties of the protocol itself, while condition (3) is concerned with the protocol's state of execution. Condition (3) captures the fact that B for example, does not know y_1, and under the intractability assumption (one-way and non-trapdoor properties of f from Definition 2.2.1), cannot compute it from A's public information, x_1. If all three conditions hold, then not only does B accept A's proof of knowledge, but C also accepts ${}^{A\leftrightarrow}B$'s proof of knowledge.

Next, we define the zero-knowledge properties of a secret-chain interactive proof of knowledge.

Note that there is a subtle difference between our notion of indistinguishability and that presented for divertible zero-knowledge interactive proofs (see Definition

2.3.1). To clarify this difference, we will use the divertibility scenario described in their paper, where there is one prover and two verifiers. In section 3.6.2, we show that this situation is in fact a special case of a secret-chain, and for illustrative purposes, assume for the moment that this is so. The following is the divertibility scenario from [OkOh]:

$$P \leftrightarrow V_1 \leftrightarrow V_2.$$

Viewpoint of V_2

From our secret-chain viewpoint in Definition 3.2.2, V_2 cannot distinguish interaction with ${}^{P\leftrightarrow}V_1$ from interaction solely with V_1; in other words, it can't tell that V_1 is in turn interacting with P so it cannot ascertain whether V_1 actually knows the secret in its entirety, or whether V_1 knows only a portion and consequently must enlist the aid of a third party.

In [OkOh], the interpretation is that V_2 cannot distinguish interaction with ${}^{P\leftrightarrow}V_1$ from interaction with P, so the presence of the *intermediary* V_1 is undetectable.

Viewpoint of P

Similarly, let us also examine the prover's viewpoint of this protocol. In our approach, P cannot distinguish it's interaction with $V_1^{\leftrightarrow V_2}$ from interaction with just V_1; in other words, it cannot figure out if V_1 is using results from its interaction with P as a basis for interaction with other verifiers, so P cannot tell if there is a chain of verifiers behind V_1.

In [OkOh], P cannot distinguish interaction with $V_1^{\leftrightarrow V_2}$ from interaction with V_2, meaning that it too cannot detect any intermediary verifiers, but is only aware of the ultimate verifier V_2.

Our definition is as follows:

Definition 3.2.2 *Let $\mathcal{N}$ be a countable infinite set and let $N \in \mathcal{N}$. Let $R_N \subset X_N \times Y_N$ be a relation for finite sets X_N, Y_N. Let $g : Y_N \to Y_N$ be a polynomial-time*

computable function.

A secret-chain interactive proof of knowledge for relation R_N, $(A(x_1, y_1), B(x_1, x_2, y_2, \tilde{x}), C(\tilde{x}))$ *where* $(x_1, y_1), (x_2, y_2), (\tilde{x}, g(y_1, y_2)) \in R_N$, *is* **(computational/statistical/perfect) zero-knowledge**, *if the following hold:*

1. $(A(y_1), B)(x_1)$ *is a (computational/statistical/perfect) zero-knowledge interactive proof of knowledge for relation* R_N.

2. $({}^{A\leftrightarrow}B(x_2, y_2), C)(\tilde{x})$ *is a (computational/statistical/perfect) zero-knowledge interactive proof of knowledge for relation* R_N.

3. *For any polynomial-time prover* A^* *such that* B *does not reject* $(A^*(y_1)B)(x)$, *and any polynomial-time verifier* C^*, *if* A, B, C *are honest parties that adhere to the protocol:*

   - $\text{VIEW}_A((A(y_1), B^{\leftrightarrow C^*})(x_1))$ *is computationally indistinguishable from* $\text{VIEW}_A((A(y_1), B)(x_1))$
   - $\text{VIEW}_C(({}^{A^*\leftrightarrow}B(x_1, x_2, y_2), C)(\tilde{x}))$ *is computationally indistinguishable from* $\text{VIEW}_C((B(x_1, x_2, y_2), C)(\tilde{x}))$

3.3 A Secret-Chain Construction

Theorem 3.3.1 *Let* $\mathcal{N}$ *be a countable infinite set and let* $N \in \mathcal{N}$. *Let* A, B, C *be probabilistic polynomial-time turing machines and if the relation* R_N *is CRSR, then there exists a polynomial-time secret-chain zero-knowledge interactive proof* $(A(x_1, y_1), B(x_1, x_2, y_2, \tilde{x}), C(\tilde{x}))$ *where* $\tilde{y} = y_1 \bullet y_2$ *and* $(x_1, y_1), (x_2, y_2), (\tilde{x}, \tilde{y}) \in R_N$.

Proof:(by Construction) Before detailing the construction, we first clarify the public and private information of each party. These are summarized in Table 3.1. By convention, the public information of A will be x_1, for which the corresponding secret is y_1. The public information of both A and B will be denoted as $\tilde{x}$. This means that B's share of the corresponding secret S is y_2 such that $\tilde{x} = \mathcal{A}_N(x_1, y_2)$ and $(\tilde{x}, y_1 \bullet y_2) \in R_N$. Note that y_2 alone and x_1 suffice to construct $\tilde{x}$, so in all of

the secret-chain protocols described in this paper, the value of x_2 is never really used in the protocols themselves. It is included in the table for completeness (since presumably all users of the system already have public-secret pairs), and since B leaves his footprints in the protocols (y_2), if y_2 can be derived from transcripts of the protocols, then B can be identified by finding x_2 such that $(x_2, y_2) \in R_N$.

Party	Public	Private	Relationship
A	x_1	y_1	$(x_1, y_1) \in R_N$
B	x_2	y_2	$(x_2, y_2) \in R_N$
A$\leftrightarrow$B	$\tilde{x}$	$\tilde{y}$	$\tilde{x} = \mathcal{A}_N(x_1, y_2)$, $\tilde{y} = y_1 \bullet y_2$, $(\tilde{x}, \tilde{y}) \in R_N$

Table 3.1: Summary of Public and Private Information

We now describe a secret-chain zero-knowledge interactive proof $(A(x_1, y_1), B(x_1, x_2, y_2, \tilde{x}), C(\tilde{x}))$ that A can compute some y satisfying $(x, y) \in R_N$, and consequently, B can compute some $\tilde{y}$ satisfying $(\tilde{x}, \tilde{y}) \in R_N$. The procedure detailed in Table 3.2, is repeated $t = O(N)$ times. Recall that the algorithm $\mathcal{T}_N(\cdot, \cdot)$ was specified by condition 5 in Definition 2.2.1, and that the term "ow" is an abbreviation for "otherwise".

This construction is rather straightforward. For clarity and ease of understanding, we purposely segregate B into its constituent "verifier" and "prover" roles (embodied by the three columns spanned by B in Table 3.2) to highlight the added steps and to emphasis similarities with the divertible scenario. They can be easily combined into one concise description for B's overall behavior.

In the style of [OkOh], the second prover B first diverts its interaction with A. Then, it incorporates its share of the secret into the proof before presenting it to C, who verifies the proof for $\tilde{y}$, the (collective) secret corresponding to $\tilde{x}$. In essence, it is easy to see that the first prover A is proving to B that it knows a secret y_1 corresponding to x_1, while B is trying to prove to C that it knows the secret information corresponding to $\tilde{x}$, which is dependent on x_1 and y_2 (and hence,dependent on y_1 and

y_2).

Note that each iteration of the protocol of Table 3.2 runs in polynomial-time since there are a polynomial number of rounds and, $\mathcal{A}, \mathcal{T}$, the composition $\bullet$ and testing membership in R_N all run in polynomial-time. And because there are a polynomial number of iterations (i.e. t), the entire proof is performed in polynomial time.

3.3.1 Proof of Correctness

An outline of the proof is to show that $(A(x_1, y_1), B(x_1, x_2, y_2, \tilde{x}), C(\tilde{x}))$ is indeed a secret chain and that this protocol satisfies our criteria for zero-knowledgeness.

Correctness

The correctness of A's proof to B is easy to verify. To verify $^{A\leftrightarrow}B$'s proof to C, consider the four possibilities that result from the different choices for e and β.

We will present our argument in the framework of a general CRSR function. Keep in mind that if $(x, y) \in R_N$, then after computing $x' = \mathcal{A}_N(x, r)$,$(x', y \bullet r) \in R_N$ (from Definition 2.2.1).

- Case 1: $e = 0$ and $\beta = 0$

  When this occurs, the following quantities result:

$$\begin{aligned} x_2' &= \mathcal{A}_N(\mathcal{A}_N(x_1', r_2), y_2) = \mathcal{A}_N(x_1', r_2 \bullet y_2) \\ z_1 &= r_1 \\ z_2 &= r_1 \bullet r_2 \end{aligned}$$

  Recall that $\tilde{x} = \mathcal{A}_N(x_1, y_2)$. Verifier C must verify that $x_2' = \mathcal{A}_N(\tilde{x}, z_2)$ or $x_2' = \mathcal{A}(N, x_1, z_2 \bullet y_2) = \mathcal{A}_N(x_1, r_1 \bullet r_2 \bullet y_2)$ which is true, since $\mathcal{A}_N(x_1, r_1) = x_1'$.

- Case 2: $e = 0$ and $\beta = 1$

  When this occurs, the following quantities result:

$$x_2' = \mathcal{A}_N(\mathcal{A}_N(x_1', r_2), y_2) = \mathcal{A}_N(x_1', r_2 \bullet y_2)$$

$A(x_1, y_1)$		$B(x_1, x_2, y_2, \tilde{x})$				$C(\tilde{x})$
$r_1 \in_R Y_N$,						
$x'_1 = \mathcal{A}_N(x_1, r_1)$	$\xrightarrow{x'_1}$	$e \in_R \{0,1\}$,				
		$r_2 \in_R Y_N$,				
		$\hat{x}_2 = \begin{cases} \mathcal{A}_N(x'_1, r_2) & \text{if } e = 0 \\ \mathcal{A}_N(T_N(x_1, x'_1), r_2) & \text{ow} \end{cases}$	$\xrightarrow{\hat{x}_2}$			
				$x'_2 = \mathcal{A}_N(\hat{x}_2, y_2)$	$\xrightarrow{x'_2}$	
			$\xleftarrow{\beta}$		$\xleftarrow{\beta}$	$\beta \in_R \{0,1\}$
	$\xleftarrow{\beta'}$	$\beta' = \beta \oplus e$				
$z_1 = \begin{cases} r_1 & \text{if } \beta' = 0 \\ r_1 \bullet y_1 & \text{ow} \end{cases}$	$\xrightarrow{z_1}$					
		check:				
		$\begin{cases} x'_1 \stackrel{?}{=} \mathcal{A}_N(x_1, z_1) & \text{if } \beta' = 0 \\ (x'_1, z_1) \stackrel{?}{\in} R_N & \text{ow} \end{cases}$				
		$\hat{z}_2 = r_2 \bullet z_1^{1-2e}$	$\xrightarrow{\hat{z}_2}$			
				$z_2 = \begin{cases} \hat{z}_2 & \text{if } \beta = 0 \\ \hat{z}_2 \bullet y_2 & \text{ow} \end{cases}$	$\xrightarrow{z_2}$	
						check:
						$\begin{cases} x'_2 \stackrel{?}{=} \mathcal{A}_N(\tilde{x}, z_2) & \text{if } \beta = 0 \\ (x'_2, z_2) \stackrel{?}{\in} R_N & \text{ow} \end{cases}$

Table 3.2: Secret-Chain ZKIP: $(A(x_1, y_1), B(x_1, x_2, y_2, \tilde{x}), C(\tilde{x}))$

$$z_1 = r_1 \bullet y_1$$
$$z_2 = r_2 \bullet (r_1 \bullet y_1) \bullet y_2$$

Recall that $(x_1, y_1) \in R_N$. Thus, $(x_1', y_1 \bullet r_1) \in R_N$. After computing x_2', this became $(x_2', y_1 \bullet r_1 \bullet r_2 \bullet y_2) \in R_N$, and the second component is precisely z_2.

- Case 3: $e = 1$ and $\beta = 0$

When this occurs, the following quantities result:

$$x_2' = \mathcal{A}_N(\mathcal{A}_N(\mathcal{T}_N(x_1, x_1'), r_2), y_2)$$
$$z_1 = r_1 \bullet y_1$$
$$z_2 = r_2 \bullet (r_1 \bullet y_1)^{-1}$$

Once again, we will base our argument on the transformation of pairs in R_N resulting from the application of $\mathcal{A}$. In particular, we find that since $\mathcal{T}$ returns x^* such that $(x^*, r_1^{-1}) \in R_N$, where r_1 was the value used in the computation of x_1' by A. After the calculation of x_2', we know $(x_2', r_1^{-1} \bullet r_2 \bullet y_2) \in R$ to be true.

Recall that $(\tilde{x}, y_1 \bullet y_2) \in R$. Verifier C must also endeavor to show that $x_2' = \mathcal{A}_N(\tilde{x}, z_2)$ or that $(x_2', (y_1 \bullet y_2) \bullet z_2) \in R$. We show that the second component is equal to $(r_1^{-1} \bullet r_2 \bullet y_2)$, so, from above, this verification is successful.

Since $z_2 = r_2 \bullet (r_1 \bullet y_1)^{-1}$ and we can easily show that $(r_1 \bullet y_1)^{-1} = (r_1)^{-1} \bullet (y_1)^{-1}$, we find that $(y_1 \bullet y_2) \bullet z_2 = (y_1 \bullet y_2) \bullet r_2 \bullet (r_1 \bullet y_1)^{-1} = (y_1 \bullet y_2) \bullet r_2 \bullet (r_1)^{-1} \bullet (y_1)^{-1} = r_1^{-1} \bullet r_2 \bullet y_2$.

- Case 4: $e = 1$ and $\beta = 1$

When this occurs, the following quantities result:

$$x_2' = \mathcal{A}_N(\mathcal{A}_N(\mathcal{T}_N(x_1, x_1'), r_2), y_2)$$
$$z_1 = r_1$$
$$z_2 = r_2 \bullet r_1^{-1} \bullet y_2$$

We will use an argument similar to that of Case 2 and 3. Recall that we began with $(x_1, y_1) \in R_N$ and that $x_1' = \mathcal{A}_N(x_1, r_1)$. Since $\mathcal{T}$ outputs x^* such that $(x^*, r_1^{-1}) \in R_N$, computation of x_2' results in $(x^*, r_1^{-1} \bullet r_2 \bullet y_2) \in R_N$. The second component here is precisely z_2.

Soundness

Next, we address the soundness of these protocols. If an honest verifier B accepts a proof (A^*, B), then we can construct an extractor E^{A^*} that outputs A's secret y_1 as follows:

1. Simulate the protocol to completion to obtain z_1, and let b be the value of B's challenge bit β'.

2. Rewind the tapes of A^* to the state immediately before B's challenge is transmitted, and set β' to be $\overline{b}$, the negation of b. Obtain z_1'.

The extractor now possesses z_1 and z_1', and can compute $y_1 = z_1' \bullet z_1^{-1}$ since $(R_N(X_N), \bullet)$ is a commutative group and therefore, a unique inverse exists for $z_1 = r_1 \in R_N(X_N)$.

The construction of $E^{(A^* \leftrightarrow B^*)}$ is very similar. The tapes of A^* are rewound as before; however, the tapes of B^* must also be reset to the point just before C transmits its challenge bit. By obtaining, in a similar manner, results for both values of challenge β, the extractor can then recover $\tilde{y}$ (if we wanted to, we can recover only y_2 by rewinding the state of B^* to an earlier point when it has just received A^*'s first message and is about to select an e, doing this repeatedly until B^* selects the other value for e).

Zero-Knowledgeness

We can construct a simulator for both $(A(y_1), B)(x_1)$ and $({}^{A \leftrightarrow} B(x_2, y_2), C)(\tilde{x})$ to show that the verifiers of both protocols cannot obtain any additional information from participating in the protocol than from simulating the interaction themselves. The

simulator attempts to anticipate the verifier's guess and formulates valid responses for both cases. The simulator for both of these protocols are identical to that constructed in [ToWo] for their zero-knowledge interactive proof for relation R_N between a single prover and a single verifier.

By the specification of $\mathcal{A}$ and random selection of r_2, x_2' is perfectly independent of x_1'. Likewise, β_i' and β_i are independent, as are z_2 and z_1. Given *any* $\text{VIEW}_A = \{x_1, x_1', \beta', z_1, r_1, y_1\}$ and any $\text{VIEW}_C = \{\tilde{x}, x_2', \beta, z_2\}$, one can always try to find values for r_2 and y_2 by solving for $e = \beta \oplus \beta'$, and consequently obtaining the following two equations in two unknowns (variables from Table 3.2 have been combined):

$$z_2 = r_2 \bullet z_1^{1-2e} \bullet y_2^{\beta}, \tag{3.1}$$

$$x_2' = \begin{cases} \mathcal{A}_N(x_1', r_2 \bullet y_2) & \text{if } e = 0 \\ \mathcal{A}_N(\mathcal{T}_N(x_1, x_1'), r_2 \bullet y_2) & \text{otherwise} \end{cases} \tag{3.2}$$

But because of our one-way and non-trapdoor assumptions, these equations cannot be solved for y_2 despite the fact that the value of r_1 and even r_2 may be known.

Having satisfied the completeness, soundness and zero-knowledge conditions, the protocol in Table 3.2 is then a secret-chain zero-knowledge interactive proof.

■

3.3.2 A Note on Relation R_N...

Here, we discuss the rational behind our requirement that f_N in Definition 2.2.1 be one-way and non-trapdoor. The requirement that f_N is one-way is reasonable, for if it were not, then given x, y can be found easily by anyone in the system.

Notice that for the above protocol and the divertible protocols of [OkOh], all parties are given N as input. We cannot assume that players are not somehow able to gain other information about N (say factorization of N). We impose the restriction that no trapdoor information exists that could increase the probability of finding y such that $(x, y) \in R_N$ given a value for x. This would, for example, disqualify using

the CRSR relation based on quadratic residues modulo a composite in SCZKIP since the factors of N constitute this undesired trapdoor information.

Note that this extra restriction is necessary for SCZKIP, but not for the divertible proofs in [OkOh]. In the divertible scenario, the intermediate party V_1 diverts its conversations with P so that if P and V_2, who suspect they have two views derived from the same protocol execution through V_1, were to get together after the protocol and try to match these views, they will *always* find values for the bridging variables for all possible P and V_2 views.

The intuition behind a secret-chain protocol is slightly different. Assume for illustration that we were using the "square roots mod N" as the CRSR relation R. If neither P nor V_2 knew the factors of N, then both would be unable to solve the above Equations 3.1 and 3.2 for r_2 and y_2, since by doing so, we would be able to use P or V_2 as a subroutine for an algorithm that could compute square roots modulo a composite N without knowing the factorization of N, which is believed to be difficult.

In the divertible scenario, V_1 left no identifying trace in the communication; in secret-chains, the secret corresponding to his own public information is incorporated into the protocol. Therefore, if the factorization of N were known to P or V_2, then r_2 and y_2 can be easily computed, and V_1 can be detected and even identified by computing $f_N(y_2)$ and comparing it to the public information of all players and finding the matching x_2. Hence we restrict the class of CRSR relations for SCZKIP to be those that are one-way and not trapdoor in nature.

3.4 SCZKIP Construction Based on Discrete Logarithms

For easier understanding, we describe a SCZKIP based on the discrete logarithm problem in this section. Recall from Chapter Two that this entails the following definitions for $f, \mathcal{A}$ and $\mathcal{T}$:

$$
\begin{aligned}
f(y) &= g^y \bmod p, \\
\mathcal{A}_N(x,r) &= xg^r \bmod p, \\
y_1 \bullet y_2 &= y_1 + y_2 \bmod \phi(p), \\
\mathcal{T}(x,x') &= x(x')^{-1} \bmod p.
\end{aligned}
$$

This means that if x_1, x_2 and $\tilde{x}$ are the public information of A, B and ${}^{A\leftarrow}B$ respectively, then the corresponding secret information is y_1, y_2 and $y_1 + y_2 \bmod \phi(p)$, where $x_1 = g^{y_1} \bmod p, x_2 = g^{y_2} \bmod p$ and $\tilde{x} = g^{y_1+y_2} \bmod p$.

The protocol in Table 3.3 then ensues.

3.5 SCZKIP Construction Based on Quadratic Residues

Similarly, a SCZKIP based on quadratic residues modulo a composite number $\mathcal{N}$ is detailed. Recall from Chapter Two that the definitions for $f, \mathcal{A}$ and $\mathcal{T}$ are as follows:

$$
\begin{aligned}
f(y) &= y^2 \bmod \mathcal{N}, \\
\mathcal{A}_N(x,r) &= xr^2 \bmod \mathcal{N} \\
y_1 \bullet y_2 &= y_1 y_2 \bmod \mathcal{N} \\
\mathcal{T} &= x(x')^{-1} \bmod \mathcal{N}.
\end{aligned}
$$

This means that if x_1, x_2 and $\tilde{x}$ represent the public information of A, B and ${}^{A\leftarrow}B$ respectively, then the corresponding secret information is y_1, y_2 and $y_1 y_2 \bmod \mathcal{N}$ where $x_1 = y_1^2 \bmod \mathcal{N}, x_2 = y_2^2 \bmod \mathcal{N}$ and $\tilde{x} = (y_1 y_2)^2 \bmod \mathcal{N}$.

The protocol in Table 3.3 then ensues.

$A(x_1, y_1)$ where $x_1 = g^{y_1} \bmod p$		$B(x_1, x_2, y_2, \tilde{x})$ where $\tilde{x} = x_1 g^{y_2} \bmod p$		$C(\tilde{x})$
$r_1 \in_R Z_{p-1}$, $x_1' = x_1 g^{r_1} \bmod p$	$\xrightarrow{x_1'}$	$e \in_R \{0,1\}$, $r_2 \in_R Z_{p-1}$, $\hat{x}_2 = \begin{cases} (x_1')g^{r_2} \bmod p & \text{if } e = 0 \\ x_1(x_1')^{-1} g^{r_2} = g^{-r_1} g^{r_2} \bmod p & \text{ow} \end{cases}$	$\xrightarrow{\hat{x}_2}$	
			$x_2' = \hat{x}_2 g^{y_2} \bmod p$	$\xrightarrow{x_2'}$
			$\xleftarrow{\beta}$	$\xleftarrow{\beta}$ $\beta \in_R \{0,1\}$
	$\xleftarrow{\beta'}$	$\beta' = \beta \oplus e$		
$z_1 = \begin{cases} r_1 & \text{if } \beta' = 0 \\ r_1 + y_1 \bmod \phi(p) & \text{ow} \end{cases}$	$\xrightarrow{z_1}$			
		check: $\begin{cases} x_1' \stackrel{?}{=} x_1 g^{z_1} \bmod p & \text{if } \beta' = 0 \\ x_1' \stackrel{?}{=} g^{z_1} \bmod p & \text{ow} \end{cases}$ $\hat{z}_2 = r_2 + z_1^{1-2e} \bmod \phi(p)$	$\xrightarrow{\hat{z}_2}$	
			$z_2 = \begin{cases} \hat{z}_2 & \text{if } \beta = 0 \\ \hat{z}_2 + y_2 \bmod \phi(p) & \text{ow} \end{cases}$	$\xrightarrow{z_2}$
				check: $\begin{cases} x_2' \stackrel{?}{=} \tilde{x} g^{z_2} \bmod p & \text{if } \beta = 0 \\ x_2' \stackrel{?}{=} g^{z_2} \bmod p & \text{ow} \end{cases}$

Table 3.3: Secret-Chain ZKIP Based on Discrete Logarithms: $(A(x_1, y_1), B(x_1, x_2, y_2, \tilde{x}), C(\tilde{x}))$

$A(x_1, y_1)$ where $x_1 = y_1^2 \bmod \mathcal{N}$		$B(x_1, x_2, y_2, \tilde{x})$ where $\tilde{x} = x_1 y_2^2 \bmod \mathcal{N}$				$C(\tilde{x})$
$r_1 \in_R Z_{\mathcal{N}}^*$,						
$x_1' = x_1 r_1^2 \bmod \mathcal{N}$	$\xrightarrow{x_1'}$	$e \in_R \{0,1\}$,				
		$r_2 \in_R Z_{\mathcal{N}}^*$,				
		$\hat{x}_2 = \begin{cases} (x_1')r_2^2 \bmod \mathcal{N} & \text{if } e = 0 \\ x_1(x_1')^{-1} r_2^2 = & \\ \quad (r_1^{-1})^2 r_2^2 \bmod \mathcal{N} & \text{ow} \end{cases}$	$\xrightarrow{\hat{x}_2}$			
				$x_2' = \hat{x}_2 y_2^2 \bmod \mathcal{N}$	$\xrightarrow{x_2'}$	
			$\xleftarrow{\beta}$		$\xleftarrow{\beta}$	$\beta \in_R \{0,1\}$
	$\xleftarrow{\beta'}$	$\beta' = \beta \oplus e$				
$z_1 = \begin{cases} r_1 & \text{if } \beta' = 0 \\ r_1 y_1 \bmod \mathcal{N} & \text{ow} \end{cases}$	$\xrightarrow{z_1}$					
		check:				
		$\begin{cases} x_1' \stackrel{?}{=} x_1 z_1^2 \bmod \mathcal{N} & \text{if } \beta' = 0 \\ x_1' \stackrel{?}{=} z_1^2 \bmod \mathcal{N} & \text{ow} \end{cases}$				
		$\hat{z}_2 = r_2 z_1^{1-2e} \bmod \mathcal{N}$	$\xrightarrow{\hat{z}_2}$			
				$z_2 = \begin{cases} \hat{z}_2 & \text{if } \beta = 0 \\ \hat{z}_2 y_2 \bmod \mathcal{N} & \text{ow} \end{cases}$	$\xrightarrow{z_2}$	
						check:
						$\begin{cases} x_2' \stackrel{?}{=} \tilde{x} z_2^2 \bmod \mathcal{N} & \text{if } \beta = 0 \\ x_2' \stackrel{?}{=} z_2^2 \bmod \mathcal{N} & \text{ow} \end{cases}$

Table 3.4: Secret-Chain ZKIP Based on Quadratic Reciprocity: $(A(x_1, y_1), B(x_1, x_2, y_2, \tilde{x}), C(\tilde{x}))$

3.6 Variations on a Theme

Having described the notion of a secret chain and constructed a SCZKIP using a CRSR relation, we next taxonomize the different possibilities that can occur in longer chains, $(P, W_1, ..., W_n, V)$.

3.6.1 Multi-Prover Single-Verifier Model Extension

The protocol construction from the previous chapter which involved only two provers can be easily generalized to the $(n+1)$-prover $(n \geq 2)$ scenario in which $(W_1, ..., W_n)$ assume the role of a prover. The idea is very straightforward; the i^{th} prover proves some knowledge to the $(i+1)^{st}$ prover, but in order to do so, he must engage the aid of the $(i-1)^{st}$ prover (and remainder of the chain before him). The interface to and the protocol for each of these additional provers is in fact identical to that of B in Table 3.2– i.e. divert the messages while adding in your share of the secret.

3.6.2 Single-Prover Multi-Verifier Model Extension

We can likewise consider the case where $(W_1, ..., W_n)$ play the parts of verifiers. This extension to multiple verifiers results in the divertible scenario posed in [OkOh].

Theorem 3.6.1 *A divertible (computational/statistical/perfect) zero-knowledge interactive proof is a special case of the secret-chain zero-knowledge interactive proof.*

Proof: The idea behind secret chains is that each prover in the chain of provers knows part of a greater secret $\tilde{y}$. In the (P_1, P_2, V) case, P_2's secret can be "degenerate", meaning that it can in fact contribute nothing at all to the overall secret $\tilde{y}$. So, in fact, we know that $\tilde{y} = f(y_1, y_2)$, and in this case, f is a function that takes two arguments, and returns the first as its output. This means that the secret that P_2 is trying to prove knowledge of, is precisely the secret held entirely by P_1 and P_2 does not possess any part of it. In terms of the protocol constructed above, this means

that y_2 would be, in fact, the identity element[3] of $R_N(X_N)$. It is easy to see that the protocol in Table 3.2 reduces to the divertible multi-verifier scenario defined in [OkOh].

■

3.6.3 Multi-Prover Multi-Verifier Model Extension

We may also consider the case where for some i, $(W_1, ...W_i)$ play prover roles and $(W_{i+1}, ..., W_n)$ act as verifiers. The protocol of messages are uniform enough, that the prover and verifier roles can be fitted together in a straightforward manner to create a chain of provers connected directly to a chain of verifiers. A more general variation that might have some interesting applications is a chain in which prover and verifier roles are arbitrarily intermingled and assigned to $(W_1, ..., W_n)$.

Note that a single-prover single-verifier model returns us to the usual ZKIP system of Goldwasser, Micali and Rackoff.

[3] An identity element exists since from the definition of CRSR, $(R_N(X_N), \bullet)$ is a commutative group.

Chapter 4

Application to Blind Digital Signatures

4.1 Blind Digital Signatures

Previously, zero knowledge interactive proofs have been shown to have applications to digital signatures [FiSh, OhOk, MiSh]. In [OkOh], divertible zero-knowledge interactive proofs were used to construct blind digital signatures (which are informally described below). Likewise, the secret-chain zero-knowledge construction can be used to create blind digital signatures, in which, by virtue of a secret-chain, a signature requires the participation of all provers, and by virtue of it being a blind signature, the message to be signed remains unknown to them.

In [GoMiRi], a digital signature scheme was defined by:

- A security parameter k that governs the length of messages and signatures, etc.
- A message space $\mathcal{M} \subset \{0,1\}^{k^c}$, for some constant $c > 0$, of valid messages that may be signed.
- A polynomial-time generating algorithm $\mathcal{G}$ such that $\mathcal{G}(1^k)$ outputs (s,p) where s represents a secret key used during signing, and p represents the corresponding public key for use in the verifying algorithm.

- A polynomial-time signing algorithm σ that uses the secret key s to output a signature $\sigma(m, s)$ for a message $m \in \mathcal{M}$.

- A polynomial-time verifying algorithm ρ that uses the public key p to test if a signature $\sigma(m, s)$ is a valid signature for message m. If $\rho(\sigma(m, s), m, p)$ outputs **true**, then the signature is valid, otherwise, the signature is rejected.

Blind signatures were first introduced by Chaum [Ch]. Assume signer S has a digital signature scheme described by (s, p, σ, ρ) for some message space $\mathcal{M}$ (as described above). We say that a signature for a message $m \in \mathcal{M}$ is a *blind signature* if the signer produces a signature for m but does not know m. After issuing such a blind signature, given m_0 and m_1, such that one of them is the message m while the other is a message randomly chosen from $\mathcal{M}$, the signer cannot correctly identify which of m_0, m_1 corresponded to the message he just signed with a probability of success greater than random guessing.

Let P_1, P_2 and V be probabilistic polynomial-time turing machines. In our definitions and construction, we consider the 3-party case, (P_1, P_2, V), in which the verifier V obtains the blind signature of a message m that must be signed with the cooperation of both signers P_1 and P_2. By convention, we allow V to select this message m (so in some sense, m can be thought of as private input to V, or can be regarded as a message randomly selected in part by V via its coin tosses).

In our earlier SCZKIP construction from Chapter Three, the entire protocol was executed sequentially $t = O(N)$ times. In the parallel version, which we will use for our blind signature construction, the t messages that would have been sent separately during the same step of each of the t iterations are concatenated together, and sent as a single message. Thus, the number of rounds and messages sent is the same as in the sequential version, however, each message is t times as long. The resulting parallel version is not necessarily zero-knowledge [FeFiSh], but as in parallel versions of divertible zero-knowledge interactive proofs, there is no transfer of useful information [OkOh, FeFiSh], and so is suitable for our purposes.

Definition 4.1.1 *Let $\mathcal{N}$ be a countable infinite set and let $N \in \mathcal{N}$. Let $R_N \subset$*

$X_N \times Y_N$ be a relation for finite sets X_N, Y_N. Let $g : Y_N \to Y_N$ be a polynomial-time computable function. Let $\mathcal{M} \subset \{0,1\}^{N^c}$ be some message space.

Let $(P_1(x_1, y_1), P_2(x_1, x_2, y_2, \tilde{x}), V(\tilde{x}))$ be a zero-knowledge secret-chain interactive proof of knowledge where $(x_1, y_1), (x_2, y_2), (\tilde{x}, g(y_1, y_2)) \in R_N$. Let $m \in \mathcal{M}$ be the message to be signed. We say that $(P_1(x_1, y_1), P_2(x_1, x_2, y_2, \tilde{x}), V(\tilde{x})[m])$, where m is private to verfier V, outputs a blind digital signature of m if the following hold:

1. *(keys) The public and secret key pair is $(\tilde{x}, \tilde{y})$ where $\tilde{y} = g(y_1, y_2)$. Note that this pair satisfies the relation R_N.*

2. *(signing algorithm) There is a polynomial-time signing algorithm σ that outputs a signature $\sigma(m, \tilde{y})$ for message m computed using secret key $\tilde{y}$.*

3. *(verifying algorithm) There is a polynomial-time verifying algorithm ρ such that $\rho(m, \sigma(m, \tilde{y}), \tilde{x})$ outputs* `true` *if $\sigma(m, \tilde{y})$ is a valid signature for message m under public key $\tilde{x}$, and* `false` *otherwise.*

4. *(blindness) The signature signed by $^{P_1 \leftrightarrow} P_2$ is a blind signature; neither P_1 nor P_2 know the message m.*

5. *(parallelness) $(P_1(x_1, y_1), P_2(x_1, x_2, y_2, \tilde{x}), V(\tilde{x}))$ is a parallel version of the sequential SCZKIP detailed in Table 3.2.*

6. *(unsuspecting parties) Let P_1, P_2 represent a pair of honest provers, and let V represent an honest verifier in the ideal scenario. Let $Z(^{P_1 \leftrightarrow} P_2, V[m])(\tilde{x})$ denote the probability space of blind signatures of a message m (verifiable using public key $\tilde{x}$) that is output by verifier V after interaction with signer $^{P_1 \leftrightarrow} P_2$.*

   Let PTM V^* represent any verifier and PTM P^* represent any prover.

   - $\mathrm{VIEW}_{P_1}(P_1, P_2^{\leftrightarrow V^*})(x_1)$ *is computationally indistinguishable from* $\mathrm{VIEW}_{P_1}(P_1, P_2)(x_1)$
   - $Z(^{P_1^* \leftrightarrow} P_2, V[m])(\tilde{x})$ *is computationally indistinguishable from* $Z(P_2, V[m])(\tilde{x})$

We show that if R_N is CRSR, then $(P_1(x_1, y_1), P_2(x_1, x_2, y_2, \tilde{x}), V(\tilde{x})[m])$ outputs a blind signature for message m. This construction is depicted in Table 4.1. Observe that this is a combination of the secret chain of provers from Table 3.2 and the blind signature scheme of [OkOh] in which the single prover in their scheme is replaced by a secret-chain consisting of two provers. The interface between the signer(s) and verifier is preserved and is highlighted in the table by double vertical lines.

Note $\mathcal{H} = \{h_i\}$ is a family of polynomial-time computable functions where $h_i : \{0,1\}^* \rightarrow \{0,1\}$ randomly hashes its arguments to a single bit. Furthermore, we use $\{x\}$ to represent the set of elements $\{x_1, \cdots, x_t\}$.

Public and Private Keys

The public key for the signature scheme is $\tilde{x}$, and the corresponding secret key is $\tilde{y}$. Furthermore, since each signer knows a portion of $\tilde{y}$, all provers must participate in the protocol to create the signature. This is then a signature scheme in which all provers must cooperate.

How to Obtain a Signature

Given a message $m \in \mathcal{M}$, the signing algorithm engages the secret chain of provers in a SCZKIP. More specifically, it assumes the role of V in $(P_1(x_1, y_1), P_2(x_1, x_2, y_2, \tilde{x}), V(\tilde{x}))$. However, there are some differences compared to the original SCZKIP protocol detailed in Table 3.2.

In particular, the protocol is a parallelized version, therefore the initial message from P_2 to V consists of a sequence of t messages $\{x_2'\}$. These values are then "diverted". By diverted, we mean that they are randomly mapped to some other values $\{x_3'\}$ by applying algorithm $\mathcal{A}_N$, much in the manner of a verifier in a divertible scenario.

V then must reply with a set of t β challenges, Instead of selecting $\beta_i, 1 \leq i \leq t$ at random, V computes $\beta = h_i(m, \{x_3'\})$.

When the protocol terminates, the final set of responses $\{z_2\}$ obtained from P_2 are also diverted to $\{z_3\}$. The signing algorithm then outputs as its signature for m,

$P_1(x_1, y_1)$		$P_2(x_1, x_2, y_2, \tilde{x})$				$V(\tilde{x})[m]$
$r_{1,i} \in_R Y_N, 1 \le i \le t$ $x'_{1,i} = A_N(x_1, r_{1,i})$	$\{a'_1\}$	$e_i \in_R \{0,1\}$, $r_{2,i} \in_R Y_N$, $\hat{x}_{2,i} = \begin{cases} A_N(x'_{1,i}, r_{2,i}) & \text{if } e_i = 0 \\ A_N(T_N(x'_{1,i}, x_1), r_{2,i}) & \text{ow} \end{cases}$	$\{a_2\}$			
				$x'_{2,i} = A_N(\hat{x}_{2,i}, y_{2,i})$	$\{a'_2\}$	$d_i \in_R \{0,1\}$, $r_{3,i} \in_R Y_N$, $x'_{3,i} = \begin{cases} A_N(x'_{2,i}, r_{3,i}) & \text{if } d_i = 0 \\ A_N(\mathcal{R}(N, x'_{2,i}, \tilde{x}), r_{3,i}) & \text{ow} \end{cases}$
$z_{1,i} = \begin{cases} r_{1,i} & \text{if } \beta'_i = 0 \\ r_{1,i} \bullet y_1 & \text{ow} \end{cases}$	$\{\beta'\}$ $\{z_1\}$	$\beta'_i = \beta_i \oplus e_i$ check: $\begin{cases} x'_{1,i} \stackrel{?}{=} A_N(x_1, z_{1,i}) & \text{if } \beta'_i = 0 \\ (x'_{1,i}, z_{1,i}) \stackrel{?}{\in} R_N & \text{ow} \end{cases}$ $\hat{z}_{2,i} = r_{2,i} \bullet z_{1,i}^{1-2e_i}$	$\{\beta\}$ $\{z_2\}$	$z_{2,i} = \begin{cases} \hat{z}_{2,i} & \text{if } \beta = 0 \\ \hat{z}_{2,i} \bullet y_2 & \text{ow} \end{cases}$	$\{\beta\}$ $\{z_2\}$	$\beta_i = h_i(m, \{x'_3\}) \oplus d_i$ check: $\begin{cases} x'_{2,i} \stackrel{?}{=} A_N(\tilde{x}, z_{2,i}) & \text{if } \beta_i = 0 \\ (x'_{2,i}, z_{2,i}) \stackrel{?}{\in} R_N & \text{ow} \end{cases}$ $z_{3,i} = r_{3,i} \bullet z_{2,i}^{1-2d_i}$ signature $\sigma(m) = [m, (\{x'_3\}, \{z_3\})]$

Table 4.1: Blind Multi-Signature Scheme: $(P_1(x_1, y_1), P_2(x_1, x_2, y_2, \tilde{x}), V(\tilde{x})[m])$

the tuple: $\{\{x'_3\},\{z_3\}\}$.

Since there are a polynomial number ($t = O(N)$) of messages sent during each parallelized round, and as in the sequential version, all computation (including $\mathcal{A}$ used for diverting quantities and h_i for hashing values) can be performed in polynomial-time, the signing algorithm is polynomial-time.

How to Verify a Signature

The verifying algorithm ρ takes as input the message m, it's signature $(\{x'_3\},\{z_3\})$ and the public key $\tilde{x}$. Recall, $\{x'_3\}$ is notation for the sequence of $\{x_{3,1}, x_{3,2}, \cdots, x_{3,t}\}$.

To check the signature for m, it computes $\beta_i = h_i(m,\{x'_3\}), 1 \leq i \leq t$. If $\beta_i = 0$ and $x'_{3,i} = \mathcal{A}_N(\tilde{x}, z_{3,i})$ then ρ accepts. Also, if $\beta_i = 1$ and $(x'_{3,i}, z_{3,i}) \in R_N$ then ρ also accepts. In all other cases, ρ rejects. If ρ accepts these checks for all values of i, then the signature for m is valid, otherwise, the signature is rejected.

The verifying algorithm runs in polynomial-time since h_i can be computed in polynomial-time and A_N is a polynomial-time algorithm. Furthermore, membership in (x,y) in R_N can be tested in polynomial-time by applying the polynomial-time computable f (from Definition 2.2.1) and checking that $x = f(y)$.

Blindness

By construction, the message m is revealed to neither P_1 nor P_2. Furthermore, V's selection of d_i and $r_{3,i}$ are randomly and uniformly distributed, therefore, the diverted quantities computed by V and the output of h_i (i.e. β_i) likewise obey a uniform random distribution. As a result, P_2 is unable to trace and link the signature to any particular execution of the protocol.

Unsuspecting Parties

The first condition follows from the underlying secret-chain protocol. The intuition is that P_1 could not tell in the SCZKIP setting that his proof was being built upon and used by P_2, and so likewise, in the blind signature signing scheme, P_1 should still not be able to tell that he is not the final signer.

The second condition represents the verifier's perception of the signature scheme; namely, that from the distribution of signatures, V does not realize that P_2 is not the sole signer and that several signers participated in securing the signature.

Lastly, we show that given message m, it is difficult for a forger $\mathcal{F}$ to find values for a signature $(\mathcal{X}, \mathcal{Z})$ that the verifying algorithm will accept as valid. To see why this is so, consider the simpler case where, instead of t values, there is only one value in each set; i.e. $\mathcal{X} = \{x_1\}$ and $\mathcal{Z} = \{z_1\}$. Next, $\mathcal{F}$ flips a coin to obtain a value $\hat{\beta}$. This $\hat{\beta}$ is in effect $\mathcal{F}$'s guess for the value of, in this case, β_1. If $\hat{\beta} = 0$, then $\mathcal{F}$ selects a random value for z_1 and computes $x_1 = \mathcal{A}_N(\tilde{x}, z_1)$. If $\hat{\beta} = 1$, then $\mathcal{F}$ again selects a random value for z_1 and computes $x_1 = f(z_1)$. Only if $\hat{\beta} = \beta = h_1(m, x_1)$ will $\mathcal{F}$'s signature be accepted by the verifying algorithm. This occurs with probability $\frac{1}{2}$.

If instead, there are t values, as in the protocol, then to be able to forge a signature, $\mathcal{F}$ must be able to correctly guess all t values for $\beta_i, 1 \leq i \leq t$, and succeeds only with probability $\frac{1}{2^t}$.

Chapter 5

A Secret-Chain Zero-Knowledge Protocol

In this chapter, we apply our model and exhibit a SCZKIP for the Graph Isomorphism problem and the Graph 3-Colorability problem.

5.1 Basic Graph Notation

Let G be a graph, often represented as $G = (V, E)$, where V is the set of vertices, and E, the set of edges.

A permutation π of a graph is simply a renaming of the vertices of the graph. More specifically, if $G = (V, E)$ and π is a permutation, then $\pi(G)$ represents a permutation of G according to π, so $\pi(G) = (V, E')$ such that $(u, v) \in E \Leftrightarrow (\pi(u), \pi(v)) \in E'$. Let $SymV$ denote the set of possible permutations of a graph G. For two permutations $\pi, v \in SymV$, we denote $(\pi \circ v)$ or πv as their composition, i.e. $\pi \circ v(G) = \pi v(G) = \pi(v(G))$. Also, note that given a permutation π, its inverse π^{-1} can be computed in polynomial-time.

Let (X, Y) denote an ordered pair, where X and Y can be any two quantities, such as graphs or permutations.

5.2 Graph Isomorphism

This section presents a secret-chain zero-knowledge protocol for the Graph Isomorphism problem, built upon the divertible zero-knowledge proof described in [BuDe].

Given two graphs G and G', there is currently no known efficient algorithm to ascertain whether or not G is isomorphic to G'. This is the Graph Isomorphism problem, and can be stated more precisely using the instance-question format of [GaJo]:

GRAPH ISOMORPHISM

INSTANCE: Graphs $G = (V, E)$ and $G' = (V, E')$

QUESTION: Is there a one-to-one function $f : V \rightarrow V$ such that $\{u, v\} \in E \Leftrightarrow \{f(u), f(v)\} \in E'$.

Note that the Graph Isomorphism problem is not known to be NP-complete. Furthermore, because commutativity of permutation composition does not hold, $R_G = \{(G', \pi) | G' = \pi(G)\}$ is not a random self-reducible relation.

5.2.1 Construction

For clarity, Table 5.1 summarizes the public and private information of the various parties.

Party	Public	Private	Relationship
A	(G_0, G_1)	σ	$G_0 = \sigma(G_1)$
B	(H_0, H_1)	v	$H_0 = v(H_1)$
A↔B	(F_0, F_1)	$v\sigma v^{-1}$	$F_0 = v\sigma v^{-1}(F_1)$ $F_0 = v(G_0)$ $F_1 = v(G_1)$

Table 5.1: Summary of Public and Private Information for Graph Isomorphism

Let (G_0, G_1) be prover A's public information, for which only A knows a permutation $\sigma \in SymV$ such that $G_0 = \sigma(G_1)$. Similarly, let (H_0, H_1) be B's public informa-

tion and let $v \in SymV$ be a permutation known only to B such that $H_0 = v(H_1)$. The public information of $^{A\leftrightarrow}B$ (i.e. $\tilde{x}$) is then a pair of graphs $(F_0, F_1) = (v(G_0), v(G_1))$. Note that (H_0, H_1), the public information of B is never really used in the protocol; only the permutation v permuting H_1 to H_0 plays a role in the protocol for obtaining the combined public key for $^{A\leftrightarrow}B$, i.e. $\tilde{x} = (v(G_0), v(G_1))$, and the corresponding secret key $\tilde{y} = v\sigma v^{-1}$.

The relationship of all graphs in the protocol are depicted in Figure 5-1 for easier understanding. These graphs are all derived as permutations of A's public information. Note that the public information of B is not included in the graph, only the permutation v, B's private information and his passport to obtaining (F_0, F_1) from (G_0, G_1).

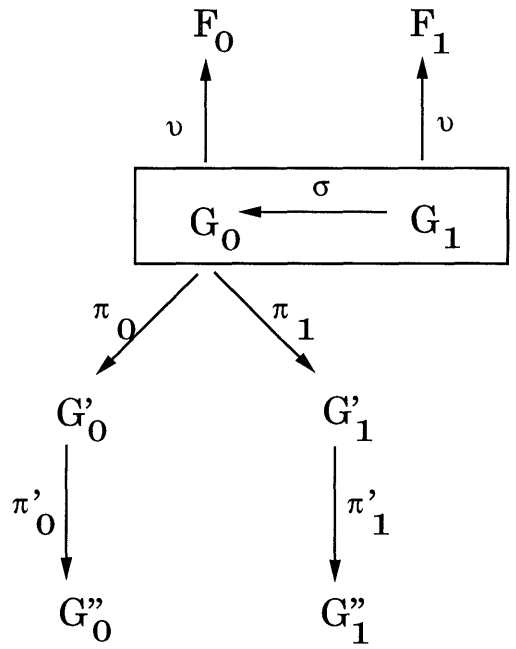

Figure 5-1: Relationship Between the Graphs of SCZKIP for Graph Isomorphism.

The construction for $(A((G_0, G_1), \sigma), B((G_0, G_1), (H_0, H_1), v, (F_0, F_1)), C(F_0, F_1))$ is rather straightforward, and the protocol, which is executed t times, is shown in Table 5.2. Note that as in [BuDe], the first message sent by prover A is derived from two instances of the same graph, G_0. This provides the prover with an opportunity

$A((G_0,G_1),\sigma)$		$B((G_0,G_1),(H_0,H_1),\upsilon,(F_0,F_1))$				$C(F_0,F_1)$
$\pi_0,\pi_1 \in_R SymV$, $G'_0 = \pi_0(G_0)$, $G'_1 = \pi_1(G_0)$	$\xrightarrow{(G'_0,G'_1)}$	$e \in_R \{0,1\}$, $\pi'_0,\pi'_1 \in_R SymV$ $G''_0 = \pi'_e(G'_e)$, $G''_1 = \pi'_{\overline{e}}(G'_{\overline{e}})$	$\xrightarrow{(G''_0,G''_1)}$ $\xleftarrow{\beta}$		$\xrightarrow{(G''_0,G''_1)}$ $\xleftarrow{\beta}$	$\beta \in_R \{0,1\}$
	$\xleftarrow{\beta'}$	$\beta' = \beta \oplus e$				
$\psi_0 = \pi_0 \circ \sigma^{\beta'}$, $\psi_1 = \pi_1 \circ \sigma^{\overline{\beta}'}$	$\xrightarrow{(\psi_0,\psi_1)}$	check: $\begin{cases} \psi_0,\psi_1 \stackrel{?}{\in} SymV, \\ G'_0 \stackrel{?}{=} \psi_0(G_{\beta'}), \\ G'_1 \stackrel{?}{=} \psi_1(G_{\overline{\beta}'}) \end{cases}$ $\hat{\psi}'_0 = \pi'_e \circ \psi_e$, $\hat{\psi}'_1 = \pi'_{\overline{e}} \circ \psi_{\overline{e}}$	$\xrightarrow{(\hat{\psi}'_0,\hat{\psi}'_1)}$	$\psi'_0 = \hat{\psi}'_0 \circ \upsilon^{-1}$, $\psi'_1 = \hat{\psi}'_1 \circ \upsilon^{-1}$	$\xrightarrow{(\psi'_0,\psi'_1)}$	check: $\begin{cases} \psi'_0,\psi'_1 \stackrel{?}{\in} SymV, \\ G''_0 \stackrel{?}{=} \psi'_0(F_\beta), G''_1 \stackrel{?}{=} \psi'_1(F_{\overline{\beta}}), \end{cases}$

Table 5.2: Secret-Chain ZKIP for Graph Isomorphism:
$(A((G_0,G_1),\sigma), B((G_0,G_1),(H_0,H_1),\upsilon,(F_0,F_1)), C(F_0,F_1))$

to show that G_0 and G_1 are indeed isomorphic (with the use of σ), and the bit ϵ serves to randomize the ordering of graphs and the message β' (so that $\beta' = \beta$ with probability $\frac{1}{2}$).

5.2.2 Proof of Correctness

Completeness

The proof of correctness is similar to those of [BuDe] and [BuDeItSaSh]. It is easy to see that if each of A and B is honest and possesses knowledge of σ and υ, then they can successfully participate in $(A((G_0, G_1), \sigma), B((G_0, G_1), (H_0, H_1), \upsilon, (F_0, F_1)))$ and $(^{A\leftrightarrow}B((G_0, G_1), (H_0, H_1), \upsilon, (F_0, F_1)), C(F_0, F_1))$ respectively[1].

Soundness

We prove soundness for each of these interactive protocols by exhibiting an extractor machine for both cases.

For protocol (A, B), probabilistic polynomial-time machine $E^{A((G_0,G_1),\sigma)}$ participates in the protocol using A as a blackbox subroutine. It runs A to obtain G', issues a bit β' and receives (ψ_0, ψ_1) in response. The extractor then rewinds A's tapes back to the point where A is awaiting a challenge bit[2], and sends $\overline{\beta'}$ instead. After learning A's responses, E^A now has π_0 and $\pi_0\sigma$ (as well as π_1 and $\pi_1\sigma$), and from these, A's secret permutation σ can be compute as $\sigma = \pi_0^{-1}(\pi_0\sigma)$.

The extractor for the $(^{A\leftrightarrow}B, C)$ part of the protocol is similar. As a result of probing $^{A\leftrightarrow}B$ for responses to both challenges β and $\overline{\beta}$, $E^{^{A\leftrightarrow}B((G_0,G_1),(H_0,H_1),\upsilon,(F_0,F_1)),C(F_0,F_1))}$ acquires the pairs of permutations:

$$(\pi_0'\pi_0\sigma\upsilon^{-1}, \pi_1'\pi_1\upsilon^{-1}) \text{ and } (\pi_0'\pi_0\upsilon^{-1}, \pi_1'\pi_1\sigma\upsilon^{-1}).$$

Consider the first quantity of each pair: $\pi_0'\pi_0\sigma\upsilon^{-1}$ and $\pi_0'\pi_0\upsilon^{-1}$. From the latter, the

[1] For the remainder of this section, we will denote these as (A, B) and $(^{A\leftarrow}B, C)$ for simplicity and clarity.

[2] For example, E resets A and reruns A on the same input and random tapes.

value of $(\pi_0'\pi_0 v^{-1})^{-1} = v\pi_0^{-1}(\pi_0')^{-1}$ can be found. When composed with the former, we get $(v\pi_0^{-1}(\pi_0')^{-1}) \circ (\pi_0'\pi_0\sigma v^{-1}) = v\pi_0^{-1}\pi_0\sigma v^{-1} = v\sigma v^{-1}$, and this is precisely the private information held by $^{A\leftrightarrow}B$.

Zero-Knowledgeness

The protocol for Graph Isomorphism is zero-knowledge if expected polynomial-time simulators SIM^{B^*} and SIM^{C*} can be constructed to probabilistically mimic the behavior of A and $^{A\leftrightarrow}B$ respectively.

First, we show that simulator $SIM^{B^*}((G_0, G_1))$ produces an output whose probability distribution is (computationally/statistically/perfectly) indistinguishable from $\text{VIEW}_{B^*}((A(\sigma), B^*)((G_0, G_1)))$.

Here we outline a description of SIM. In all invocations of B^*, SIM will place a pair of graphs on B^*'s input tape, and a sequence of randomly chosen bits (fixed across invocations) on B^*'s random tape. Based on its input, its random coins and perhaps a history of interaction so far, B^* writes a random challenge on its write-only output tape, which is then read by SIM. It is the goal of SIM to be able to output t successful conversations with B^*, so depending on B^*'s choice of a challenge, SIM will appends another conversation to its compilation, or repeats the invocation.

More specifically,

1. SIM chooses a random $\alpha \in_R \{0,1\}$. This represents his guess of B^*'s challenge β'.

2. SIM selects two random permutations $\pi_a, \pi_b \in_R SymV$ and gives $(\pi_a(G_\alpha), \pi_b(G_{\overline{\alpha}}))$ as input to B^*. Based on this input, its random coins, and its history, B^* returns its challenge β'.

   (a) $(\alpha = \beta')$ In this case, SIM succeeded in guessing B^*'s challenge. It can answer B^*'s challenge by sending (π_a, π_b), in which case B^* will successfully verify that $G_0 = \pi_a(G_{\beta'})$ and $G_1 = \pi_b(G_{\overline{\beta}'})$. The successful conversation $\{(\pi_a(G_\alpha), \pi_b(G_{\overline{\alpha}})), \beta', (\pi_a, \pi_b)\}$ is then appended to SIM's compilation.

(b) ($\alpha \neq \beta'$) Nothing is added to its compilation, instead, SIM repeats the current invocation.

These steps are iterated until SIM has a compilation of t successful conversations.

Note the SIM will terminate in expected polynomial-time. Each invocation is polynomial-time, and since B^*'s choice of β' and SIM's selection of $(\pi_a(G_\alpha), \pi_b(G_{\overline{\alpha}}))$ are independent, the probability that $\alpha = \beta'$ is $\frac{1}{2}$. Hence each invocation is expected to be repeated twice before a successful conversation is found, and the total expected time for SIM is thus polynomial.

In this manner, the simulator generates a whole transcript of conversations whose probability distribution is supposedly identical to that of $(A(\sigma), B^*)((G_0, G_1))$. Let $S = \{((G_a, G_b), \beta', (\pi_a, \pi_b)) | G_a = \pi_a(G_{\beta'}), G_b = \pi_b G_{\overline{\beta'}}, \beta' = B^*\text{'s challenge}\}$. As in [GoMiWi], we can show that there exists a 1-1 mapping between S and $SymV \times SymV$, where the inverse mapping,
i.e. $(\pi_a, \pi_b) \rightarrow ((\pi_a(G_1), \pi_b(G_1)), \beta'(\pi_a, \pi_b), (\pi_a \sigma^{1-\beta'}, \pi_b \sigma^{\beta'}))$ where $\beta'(\pi_a, \pi_b)$ is B^*'s challenge, corresponds to the way an element of S is selected in $(A(\sigma), B^*)((G_0, G_1))$.

SIM on the other hand, randomly chooses α, π_a and π_b are randomly chosen. As a result, $\pi_a G_{\overline{\alpha}}$ and $\pi_b G_\alpha$ are randomly permuted versions of G_0 and G_1. Given the fact that $G_0 \equiv G_1$, as far as B^* can tell, these "look like" two random permutations of G_0, and this is exactly the case in the first transmission of $(A(\sigma), B^*)((G_0, G_1))$. Given a value of $\beta' = \alpha$, the conversation recorded by SIM, i.e. $\{(\pi_a(G_{\overline{\alpha}}), \pi_b(G_\alpha)), \beta', (\pi_a, \pi_b)\}$, is also a random, uniformly selected, element of S.

Thus the two distributions are identical.

A simulator for $({}^{A\leftarrow}B, C)$ can be specified in practically the same manner, and hence, both protocols (A, B) and $({}^{A\leftarrow}B, C)$ are zero-knowledge.

Given any $\text{VIEW}_A = ((G'_0, G'_1), \beta', (\psi_0, \psi_1))$ and any $\text{VIEW}_C = ((G''_0, G''_1), \beta, (\psi'_0, \psi'_1))$, an attempt might be made to reconstruct the entire protocol. With β and β', the value for e can be computed, and we end up with the following equations where π_0, π_1 and v^{-1}, are unknown:

$$
\begin{aligned}
G_0'' &= \pi_e'(G_e') \\
G_1'' &= \pi_{\bar{e}}'(G_{\bar{e}}') \\
\psi_0' &= (\pi_e' \circ \psi_e) \circ v^{-1} \\
\psi_1' &= (\pi_{\bar{e}}' \circ \psi_{\bar{e}}) \circ v^{-1}
\end{aligned}
$$

Notice that if π_0' or π_1' can be found, then v can be computed as well. Assuming there are a polynomial number of possible users, one might then be able to identify the intermediary party B who partook in the protocol responsible in these two views. However for large enough graphs, finding whether or not the two graphs are isomorphic and finding a permutation from one to the other is not believed to be in P. Hence, we expect that it is difficult to compute (π_0', π_1'), and consequently, v and the identity of B remains unknown. In short, given some (G_0, G_1) (belonging to some P_1) and some (F_0, F_1), neither P_1 nor a verifier can tell whether or not these pairs of graphs are element-wise isomorphic.

Chapter 6

Conclusions

Much work has been done and is currently being pursued in the area of cryptography, and in particular, zero-knowledge. Applications for zero-knowledge and its many variants and flavors are constantly being contemplated, devised and explored. Secret-chains are another of these efforts to understand and grasp the extent of the notions first put forth in Goldwasser, Micali and Rackoff's seminal paper.

In this thesis, we presented the idea of secret chains, and formalized a model for zero-knowledge proofs which is topologically based on a linear chain of provers. Each of these provers possessed part of a "larger" secret, and whenever a proof was needed for this larger secret, joint participation was required.

In Chapter 3, we saw that the verifier and prover roles in the secret chain model were modular enough and their interfaces uniform enough, that they could be glued and stringed together into chains with an arbitrary number and mix of verifiers and provers. Specifically, the multi-verifier model (Okamoto and Ohta's divertibility situation) was indeed a special case of the secret-chain model in which intermediate parties have degenerate prover roles and only play verifier roles which diverted the various quantities.

It is always an interesting phenomenon how the same protocol can be viewed differently, depending on the parties involved and the model used. This is really exemplified here between the secret-chain model and the divertibility models, which present two distinct outlooks for the same situation.

We also specified a secret-chain zero-knowledge protocol based on a CRSR relation, under the assumption that this relation was one-way and non-trapdoor. Furthermore, this protocol was used as a basis for a secret-chain blind signature scheme and for exhibiting a secret-chain zero-knowledge proof for graph isomorphism. It would be interesting to find a clean and elegant SCZKIP proof for an NP-complete language(relation).

The key idea behind secret-chains is that the ability to successfully provide a proof is contingent upon the cooperation of all secret holders. However, even though their participation is required, they may be oblivious to this fact that they hold shares of a larger secret. Relaxation of this restriction allows for applications like the observer model. Indeed, further research by Okamoto, Ohta and Fujisaki [OkOhFu] is also being done to unify the secret-chain model and the observer model presented by Chaum, et.al. for use in electronic cash systems.

Nevertheless, with this restriction in place, it is still possible to realize blind signature schemes. The untraceability can also be exploited to handle situations like our information broker described earlier in Chapter One, where the broker is basically "in on" it all, but wishes to keep everyone else in the dark regarding his sources for information and his client pool.

In addition to proofs of knowledge, the secret-chain model may also be applicable to proofs of computational power. Assume that Alice's proof of computational power somehow represents some function that she is able to compute but that others cannot (perhaps because she has the resources and technology available, or the function is too complex or costly for the general user to implement, or for some reason she wishes to maintain sole proprietary rights to the function code/algorithm, etc). Bob on the other hand has also devised an algorithm to solve another problem, but his code makes use of Alice's function as a subroutine because he may lack the ability/resources/know-how to compute it. In this case, anyone who needs to use Bob's algorithm can do so by supplying the appropriate inputs. Even though Alice's algorithm is not revealed, procedural abstraction allows Bob's algorithm to be implemented, and Bob simply must enlist Alice aid in a secret-chain manner. The

case when even the inputs to these functions are somehow hidden is reminiscent of instance-hiding proofs, so perhaps these have divertible and secret-chain counterparts.

One other extension is removal of the linear chain topological restriction. Consider a smaller subset of CRSR relations, where a law of composition $\odot : X_N \times X_N \rightarrow X_N$ exists such that $(X_N, \odot)$ is a commutative group. Let there be a homomorphic property for f (from Definition 2.2.1 such that $f(y_1) \odot f(y_2) = f(y_1 \bullet y_2)$ (these were described briefly in [OkOh] and called "one-way homomorphic functions" in [OkOhFu]). It is then possible to dispense with the linear arrangement and opt for a tree instead. The interesting feature about a tree structure is that several branches coverage at a node; in terms of our information broker, he is able to combine several sources of information together. In terms of Bob, he is not limited to using only Alice's computational ability, but can use combinations of Alice's, Eve's, Victor's, Jim's....

Indeed, the questions are endless and there is vast territory to be explored. This thesis will have hopefully uncovered some small fragment of this huge iceberg.

Bibliography

[AnLi] Angluin, D., and Lichtenstein, D., "Provable Security of Cryptosystems:a Survey," Yale University, YALEU/DCS/TR-288, October, 1983.

[BeGo] Bellare, M., and O. Goldreich, "On Defining Proofs of Knowledge," *Advances in Cryptology: CRYPTO '92 Proceedings*, 1992.

[Be] Benaloh, J., "Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret(Extended Abstract)," *Advances in Cryptology: CRYPTO '86 Proceedings*, 1986, pp. 251-260.

[BeYu] Benaloh, J., and Yung, M. "Distributing the Power of a Government to Enhance the Privacy of Voters," *Proceedings of the* 5^{th} *ACM Symposium on Principles of Distributed Computing*, Calgary, AB, August 1986.

[BlMi] Blum, M., and Micali, S., "How to Generate Cryptographically Strong Sequences of Psuedo-Random Bits," *SIAM Journal of Computing*, vol. 13, no. 4, November 1984, pp.850-864.

[BrChCrFePe] Brands, S., D. Chaum, R. Cramer, N. Ferguson, and T. Pedersen, "Transaction Systems with Observers," CWI manuscript, 1992.

[BuDe] Burmester, M., and Y. Desmedt, "All Languages in NP Have Divertible Zero-Knowledge Proofs and Arguments Under Cryptographic Assumptions," *Advances in Cryptology: EUROCRYPT '89 Proceedings*, 1989.

[BuDeItSaSh] Burmester, M., Y. Desmedt, T. Itoh, K. Sakurai, and H. Shizuya, "Divertible and Subliminal-free Zero-knowledge Proofs of Languages," manuscript.

[Ch] Chaum, D., "Blind Signatures for Untraceable Payments," *Advances in Cryptology: CRYPTO '82 Proceedings*, 1982.

[Ch2] Chaum, D., "Achieving Electronic Privacy," *Scientific American*, August 1992, pp.96-101.

[CoFi] Cohen, J., and M. Fischer, "A Robust and Verifiable Cryptographically Secure Election Scheme," *Proceedings of the* 26^{th} *IEEE Symposium on Foundations of Computer Science, Portland, OR*, October 1985, pp. 372-382.

[CrPe] Cramer, R., and T. Pedersen, "Improved Privacy in Wallets with Observers," *Advances in Cryptology: EUROCRYPT '93 Proceedings*, 1993.

[FeFiSh] Feige, U., A. Fiat, and A. Shamir, "Zero Knowledge Proofs of Identity," *Proceedings of the* 19^{th} *ACM Symposium on Theory of Computing*, 1987, pp. 210-217.

[FeSh] Feige, U., and A. Shamir, "Witness Indistinguishable and Witness Hiding Protocols," *Proceedings of the* 22^{nd} *Annual ACM Symposium on Theory of Computing*, 1990, pp. 416-426.

[FiSh] Fiat, A., and A. Shamir, "How To Prove Yourself: Practical Solutions to Identification and Signature Problems," *Advances in Cryptology: CRYPTO '86 Proceedings*, 1986.

[GaJo] Garey, M., and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, USA, 1979.

[GoMiWi] Goldreich, O., S. Micali, and A. Wigderson, "Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems," *Journal of the Association for Computing Machinery*, vol. 38, no. 1, July 1991, pp. 691-729.

[GoMi] Goldwasser, S., and S. Micali, "Probabilistic Encryption," *Journal of Computer and System Sciences*, vol. 28, no. 2, April 1984, pp. 270-299.

[GoMiRa] Goldwasser, S., S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems," *Siam Journal of Computing*, vol. 18, no. 1, February, 1989.

[GoMiRi] Goldwasser, S., S. Micali, and R. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," *Siam Journal of Computing*, vol. 17, no. 2, April 1988.

[GuQu] Guillou, L., and J. Quisquater, "A "Paradoxical" Identity-Based Signature Scheme Resulting from Zero-Knowledge," *Advances in Cryptology: CRYPTO '88 Proceedings*, 1988, pp. 216-231.

[ItSaSh] Itoh, T., K. Sakurai, and H. Shizuya, "Any Language in IP Has a Divertible ZKIP," *Advances in Cryptology: ASIACRYPT '90 Proceedings*, 1990.

[Le] Leo, J., "Comments on "Divertible Zero Knowledge Interactive Proofs and Commutative Random Self-Reducibility," manuscript, 1990.

[MiSh] Micali, S., and A. Shamir, "An Improvement of the Fiat-Shamir Identification and Signature Scheme," *Advances in Cryptology: CRYPTO '88 Proceedings*, 1988.

[OhOkFu] Ohta, K., T. Okamoto, and A. Fujioka, "Secure Bit Commitment Function Against Divertibility," *Advances in Cryptology: EUROCRYPT '92 Proceedings*, 1992.

[OhOk] Ohta, K., and T. Okamoto, "A Modification of the Fiat-Shamir Scheme," *Advances in Cryptology: CRYPTO '88 Proceedings*, 1988.

[OkChOh] Okamoto, T., D. Chaum, and K. Ohta, "Direct Zero Knowledge Proofs of Computational Power in Five Rounds," *Advances in Cryptology: EUROCRYPT '91 Proceedings*, 1991, pp. 96-105.

[OkOh] Okamoto, T., and K. Ohta, "Divertible Zero-Knowledge Interactive Proofs and Communtative Random Self-Reducibility", Abstract in *Advances in Cryptology: EUROCRYPT '89 Proceedings*, 1989.

[OkOhFu] Okamoto, T., K. Ohta, and E. Fujisaki, "Observer Transaction Systems Based on Secret-Chain Zero-Knowledge Proof Systems," unpublished manuscript, 1994.

[PeCh] Pedersen, T., and D. Chaum, "Wallet Databases with Observers," *Advances in Cryptology: CRYPTO '92 Proceedings*, 1992.

[ToWo] Tompa, M., and H. Woll, "Random Self-Reducibility and Zero Knowledge Interactive Proofs of Possession of Information," *Proceedings of the* 28^{th} *IEEE Symposium on Foundations of Computer Science*, 1987, pp. 472-482.

[Yu] Yung, M., "Zero-Knowledge Proofs of Computational Power," *Advances in Cryptology: EUROCRYPT '89 Proceedings*, 1989, pp. 196-207.