Frontiers in Zero Knowledge

by

Amit Sahai

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2000
[September 2000]

Author ....
Department of Electrical Engineering and Computer Science
August 5, 2000

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

BARKER

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

Frontiers in Zero Knowledge

by
Amit Sahai

Submitted to the Department of Electrical Engineering and Computer Science
on August 5, 2000, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Zero-knowledge proofs, introduced by Goldwasser, Micali, and Rackoff, are fascinating constructs in which one party (the "prover") convinces another party (the "verifier") that some assertion is true, without revealing anything else to the verifier. In addition to being powerful tools for constructing secure cryptographic protocols, zero-knowledge proofs also yield rich classes of computational problems that are of complexity-theoretic interest as well.

We begin by investigating statistical zero-knowledge proofs, which are zero-knowledge proofs where the condition that "nothing is revealed to the verifier" is interpreted in a strong information-theoretic sense. We attempt to build a unified understanding of class SZK of problems admitting statistical zero knowledge proofs, through the discovery of the first natural complete problem for SZK. The work we present in this thesis on statistical zero-knowledge proofs is based on joint work with Salil Vadhan and Oded Goldreich.

After this investigation of statistical zero knowledge, we then turn our attention to extending the cryptographic uses of zero-knowledge proofs in general. Zero-knowledge proofs were designed and defined to provide provable security for a single pair of interacting parties. In general multi-user environments, however, where many interactions can take place concurrently, one must face the challenge of coordinated multi-party attacks. The standard 2-party definition of zero knowledge does not necessarily guarantee security in this scenario. In this thesis, we define and consider the notion of *concurrent zero knowledge*, where zero knowledge is guaranteed even when faced with a coordinated attack by many verifiers all acting concurrently. We show how to build concurrent zero-knowledge protocols in which honest parties need only act locally, i.e., an honest prover and verifier need not even be aware of other parties in order to be guaranteed security. A critical novel component of our approach is an explicit use of certain local timing constraints in our protocols. The work we present in this thesis on concurrent zero knowledge is based on joint work with Cynthia Dwork and Moni Naor.

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

Acknowledgments

It is my great pleasure to thank the many people who have made possible this thesis work, and more generally the research that I have had the privilege to participate in. My advisor, Shafi Goldwasser, has always provided me with advice and encouragement, for which I am very grateful. Shafi's wonderful ability to find interesting and captivating research problems, and her generosity in sharing these problems with her students, have made a profound impact on me. I hope that I will be able to recreate some fraction of her enthusiasm in my dealings with my own students in the coming years.

I also thank Oded Goldreich immensely. I had the great good fortune of beginning my research at MIT when Oded was visiting. Oded and Shafi are responsible for engendering in me a deep fascination with the subject of this thesis work, zero-knowledge proofs. A good deal of the work in this thesis was done jointly with Oded. From him I have learned a great deal about good research and its presentation. There are few individuals that I hold in higher regard.

I had the privilege of having excellent colleagues in my fellow students on the third floor. Foremost among them is Salil Vadhan, with whom I collaborated on all the work in this thesis relating to statistical zero knowledge. I thank him tremendously for his collaboration and friendship.

The work in this thesis on concurrency and zero knowledge is joint with Cynthia Dwork and Moni Naor. I thank them very much for collaborating with me on this important project. I give special thanks to Cynthia Dwork for patiently working with me during my first summer after starting graduate school, when much of this work was done. Although the ideas we discussed then failed to reach the goal we had hoped for (with good reason – we now know it to be impossible), they did lead to our first truly zero-knowledge protocol in the concurrent model with timing, presented in this thesis.

I also thank Silvio Micali and Madhu Sudan, who have graciously agreed to be on my thesis committee.

There are many others who through conversations and collaborations in other work helped make my research experience over the last few years as productive as it has been for me. Most of all, I thank Manuel Blum for introducing me to Theoretical Computer Science, and kindling in me an appreciation for the field. I also especially thank Marshall Bern, for collaborating with me on my first research effort in the field, and for convincing me (without trying) to switch fields from pure Math. I also thank Miki Ajtai, Sanjeev Arora, Mihir Bellare, Michael Bender, Ran Canetti, Moses Charikar, Yevgeniy Dodis, Ron Fagin, Venkatesan Guruswami, Shai Halevi, Piotr Indyk, Joe Kilian, Jon Kleinberg, Ravi Kumar, Eyal Kushilevitz, Eric Lehman, Danny Lewin, Anna Lysanskaya, Tal Malkin, Silvio Micali, Daniele Micciancio, Ashwin Nayak, Erez Petrank, Zulfikar Ramzan, Prabhakar Raghavan, Sridhar Rajagopalan, Ron Rivest, Dana Ron, Adam Smith, Dan Spielman, Madhu Sudan, Andrew Tomkins, Luca Trevisan, Santosh Vempala, Avi Wigderson, and Stephan Wolf. I am sure to have omitted names of others who I would like to thank as well. I would also like to thank the IBM Almaden Research Center and Xerox PARC for having me as a summer intern and visitor.

Finally, I gratefully thank my parents, family, and friends for their support and belief in me over the years. More than anyone else, my brother Anant Sahai is responsible for any intellectual curiosity that I may have, and I thank him tremendously for always pushing me and setting a high standard for me to follow. Lastly, above all, I thank the Lord for His forgiveness and mercy, and for the inexplicable good fortune that He has bestowed upon me.

Contents

Chapter 1

Introduction

1.1 Zero Knowledge Proofs

Zero-Knowledge proofs, introduced by Goldwasser, Micali and Rackoff [GMR89], are fascinating and extremely useful constructs. A zero-knowledge proof is an interactive protocol between two parties: one called a "prover" and the other called a "verifier." It is a "proof" in the sense that it is *convincing* – using the protocol, the prover can convince the verifier of the truth of some assertion to an arbitrarily high degree of confidence. And yet, the verifier learns *nothing*, other than the fact that the assertion being proven is true. Zero-knowledge proofs are so fascinating precisely because of this apparent contradiction. In particular, the definition of zero knowledge implies that someone can verify the correctness of a zero-knowledge proof without gaining any ability to convince someone else of the same statement. Zero-knowledge proofs are powerful tools for constructing secure cryptographic protocols, and also yield rich classes of computational problems that are of complexity-theoretic interest as well.

1.1.1 Informal definitions

Defining a meaningful, useful, yet realizable notion of zero-knowledge proof is no easy task. The seminal paper of Goldwasser, Micali, and Rackoff [GMR89] introduced zero knowledge as an additional property augmenting their new notion of interactive proof systems. Before we discuss interactive proof systems, we first must formalize what is meant by an "assertion" that one might prove.

Following a standard complexity-theoretic framework, we think of assertions as being strings over $\{0, 1\}$. An interactive proof works for a class of assertions described by what is called a *promise problem* $\Pi = (\Pi_Y, \Pi_N)$. Here, we interpret Π_Y, called the YES instances of Π, to be the set of true assertions, and Π_N, called the NO instances of Π, to be a set of false assertions[1].

Informally, then, an *interactive proof* for the promise problem Π is a protocol by means of which a prover P can convince a probabilistic polynomial-time verifier V of the validity of any true assertion, namely that a string x is a YES instance of Π. By this, we mean that the protocol should have the following two properties:

- *Completeness:* For every true assertion, *i.e.* every $x \in \Pi_Y$, by following the protocol, the prover will convince the verifier to accept the proof with high probability.

[1]In a promise problem, as opposed to a language, we allow for the possibility that there may be strings which are neither classified as true or false.

- *Soundness:* For every false assertion, *i.e.* every $x \in \Pi_N$, no matter what strategy the prover tries to follow, the verifier (following the protocol) will reject the proof with high probability.

If we require that the soundness condition holds for all prover strategies, including ones that are computationally unbounded, and we allow the honest prover strategy P to be computationally unbounded as well, this leads to the standard notion of an interactive proof. If, however, we require that the honest prover strategy P be implementable by a probabilistic polynomial-time machine, and we only require the soundness condition to hold for probabilistic polynomial-time prover strategies, then this leads to the notion of a *computationally-sound proof*, also called an *argument*, introduced by Brassard, Chaum, and Crepeau [BCC88].

The *zero knowledge* property requires that, during the process by which the prover convinces the verifier that the assertion is true, the verifier learns nothing beyond the validity of the assertion being proven. To formalize this condition, two probability distributions are considered:

1. The interaction of the prover and verifier from the verifier's point of view.

2. The output of a probabilistic polynomial-time machine, called the *simulator* (which does not have access to the prover or any special information)

An interactive proof system (or argument) (P, V) is said to be *zero knowledge* if, for every YES instance x, and for every probabilistic polynomial-time verifier strategy, the two distributions above are "indistinguishable." Intuitively, the verifier gains no knowledge by interacting with the prover except that x is a YES instance, since it could have run the simulator instead. The specific variants of zero knowledge differ by the interpretation given to "indistinguishable." The most strict interpretation, leading to *perfect zero knowledge*, requires that the distributions be identical. A slightly relaxed interpretation, leading to *statistical zero knowledge*, requires that the distributions have negligible statistical deviation from one another. The most liberal interpretation, leading to *computational zero knowledge*, requires that samples from the two distributions be indistinguishable by any probabilistic polynomial-time machine.

It is often useful to consider the notion of *honest-verifier* zero knowledge, where we only require the above condition to hold for the given verifier strategy V. In other words, we only insist that the verifier "learn nothing" if it follows the protocol as specified and does not try to cheat.

1.2 Statistical Zero Knowledge

1.2.1 Motivation

In the first part of this thesis, we focus on the class of problems possessing *statistical* zero-knowledge proof systems, which we denote SZK. Although statistical zero-knowledge proofs may have been defined to achieve a cryptographic purpose, they have given rise to a rich field for complexity-theoretic study.

SZK has been shown to contain a number of important problems, including GRAPH NONISOMORPHISM [GMW91], a problem which is not known to be in NP. It is also known to contain problems with cryptographic application and significance that are believed to be hard on average, such as QUADRATIC RESIDUOSITY (and its complement) [GMR89], a problem equivalent to the DISCRETE LOGARITHM problem [GK93], and approximate versions of the SHORTEST VECTOR and CLOSEST VECTOR problems in lattices [GG98]. On the other hand, it has also been shown that any problem which has a statistical zero-knowledge proof cannot be NP-hard unless the polynomial-time hierarchy collapses [For89, AH91, BHZ87]. Because SZK contains problems believed to be hard, and yet seemingly cannot contain NP-complete problems, it holds an intriguing position in complexity theory.

Another motivation to study statistical zero-knowledge proofs is that, unlike other forms of zero knowledge (like computational zero knowledge or zero-knowledge arguments), the definition of statistical zero-knowledge proofs makes essentially no reference to computational restrictions[2]. This gives us hope that statistical zero-knowledge proofs can give rise to a clean theory[3], and that the understanding we gain with regard to statistical zero-knowledge proofs may be extended to give insight to other types of zero knowledge as well. The methodology of extending techniques originally developed for statistical zero-knowledge proofs to other forms of zero knowledge has seen success in the past (for example, see [Ost91] leading to [OW93]).

1.2.2 Contribution of this thesis

In Chapters 3-5, we study statistical zero-knowledge proofs in detail, with an eye towards building a unified theory. In Chapter 3, based on joint work with Salil Vadhan [SV97, SV99] we present a complete problem for SZK. Thus, this problem provides a new and simple characterization of SZK — one which makes no reference to interaction or zero knowledge. We propose the use of complete problems as a tool to unify and extend the study of statistical zero knowledge. To this end, we use our complete problem to both establish a number of new results about SZK and easily deduce nearly all previous results about SZK. In Chapter 4, based on joint work with Oded Goldreich and Salil Vadhan [GSV98], we show how to transform any proof that is statistical zero knowledge only with regard to an honest verifier – that is, a proof which only guarantees that the verifier "learns nothing" if it follows the protocol exactly as specified – into a proof that is statistical zero knowledge for *any* verifier, not matter how it deviates from the protocol as specified. Furthermore, the transformation we give extends to a large class of computational zero-knowledge proofs, as well. Finally, in Chapter 5, also based on joint work with Oded Goldreich and Salil Vadhan [GSV99], we extend our study to what are known as non-interactive statistical zero-knowledge proofs.

1.3 Concurrent Zero Knowledge

1.3.1 Motivation

After the investigation of statistical zero knowledge, we then turn our attention to extending the cryptographic uses of zero-knowledge proofs in general. Zero-knowledge proofs, as one might expect, have vast applicability throughout cryptography. They are particularly useful in constructing secure cryptographic protocols.

Perhaps the most basic application of zero-knowledge proofs to cryptographic protocols is their use as identification schemes. Suppose Alice wishes to identify herself to the computer system Hal. Zero-knowledge proofs provide a beautiful solution to this problem. Informally speaking, Alice gives a zero-knowledge proof that "she knows her password." Now, because what Alice provides is a *proof*, only someone who actually knows Alice's password can successfully complete the protocol. On the other hand, because the protocol is *zero knowledge*, even if the computer system Hal itself (which plays the part of the verifier) has been taken over by an adversary, the adversary will learn nothing about Alice's password.

[2]The only exception is the insistence that the verifier strategy be probabilistic polynomial time.
[3]We hope that this thesis will be seen to support this view.

More generally, zero-knowledge proofs are powerful tools in the construction of cryptographic protocols at large. In every cryptographic protocol, the problem arises that a participant may not act according to the behavior specified by the protocol. Building on their result showing that, based on standard intractability assumptions, every language in NP has a computational zero-knowledge proof [GMW91], Goldreich, Micali, and Wigderson introduced a powerful paradigm to deal with this problem [GMW87]. In their paradigm, after each step of the protocol, each participant provides a zero-knowledge proof that, informally speaking, they have followed the protocol as specified. Again, because each participant is giving a *proof*, they can only successfully complete the zero-knowledge proof if they have in fact acted according to the protocol. On the other hand, because their proof is *zero knowledge*, honest participants need not fear losing any secrets in the process of proving that they have acted correctly.

1.3.2 Contribution of this thesis

Zero-knowledge proofs were designed and defined to provide provable security for a single pair of interacting parties. In general multi-user environments, however, the situation can be quite different. There may be many parties all interacting concurrently, with pairwise interactions interleaved in arbitrary ways. Furthermore, many parties may not even be aware of each other's existence. Thus, honest parties may be unwilling or unable to cooperate with each other to guarantee security. In this setting, honest parties want interactions to be *local*: an interaction between a pair of users should not have to depend on the actions of other parties in the system. On the other hand, an adversary need not play by the same rules. An adversary may mount a global, coordinated multi-party attack. In the context of zero-knowledge proofs, we observe that the standard 2-party definition of zero knowledge does not necessarily guarantee secrecy in this scenario. In Chapter 6, based on joint work with Cynthia Dwork and Moni Naor [DNS98], we define and consider the notion of *concurrent zero knowledge*, where zero knowledge is guaranteed even when faced with a coordinated attack by many verifiers all acting concurrently. We show how to build concurrent zero-knowledge protocols in which honest parties need only act locally, i.e., an honest prover and verifier need not even be aware of other parties in order to be guaranteed security.

A critical novel component of our approach is an explicit use of certain local timing constraints in our protocols. We rely on a weak synchronization assumption on the local clocks of honest parties. This assumption holds in particular if we assume that clocks of honest parties run within constant factors of each other.

Chapter 2

Preliminaries

In this chapter, we will provide some definitions and preliminaries that will be needed in this thesis. We will delay some of these, however, to later chapters in order to aid the reader.

2.1 Notation and Basic Facts

2.1.1 Languages and Promise Problems

Throughout this thesis, all strings are over $\{0,1\}$. We further assume that a canonical unambiguous efficient encoding of various types of objects as strings has been fixed (and therefore we may talk of an element of $\mathbb{Z}_p$ or a graph as being a string).

It is standard in complexity theory to talk of decision problems as languages, which are subsets of $\{0,1\}^*$. In our investigation of statistical zero knowledge, we will use the more general notion of *promise problems* [ESY84]. Formally, a promise problem Π consists of two disjoint sets of strings Π_Y and Π_N, where Π_Y is the set of YES *instances* and Π_N is the set of NO *instances*. A promise problem Π is associated with the following computational problem: Given an input which is "promised" to lie in $\Pi_Y \cup \Pi_N$, decide whether it comes from Π_Y or Π_N. Strings in $\Pi_Y \cup \Pi_N$ are called *instances* of Π, whereas strings not in $\Pi_Y \cup \Pi_N$ are said to *violate the promise*. The *complement* of Π is the promise problem $\overline{\Pi}$, where $\overline{\Pi}_Y = \Pi_N$ and $\overline{\Pi}_N = \Pi_Y$. If C is a complexity class of promise problems, then co$-\mathsf{C} \stackrel{\text{def}}{=} \{\overline{\Pi} : \Pi \in \mathsf{C}\}$. Note that languages are a special case of promise problems.

2.1.2 Computation

Throughout this thesis, we will assume a standard model of computation such as the RAM or multi-tape Turing machine model – since our analysis will be at a sufficiently high level, fixing the exact model is not important. For more information, see for example [Pap94]. We say that a machine M is *polynomial time* if there exists a polynomial $p(\cdot)$ such that for every input x, we have that $M(x)$ halts within $p(|x|)$ steps, where $|x|$ denotes the length of the string x.

A *randomized* or *probabilistic* machine is one that has access to an unbiased "coin" that it can flip on command. Formally, we choose an infinite string r uniformly among infinite strings of bits, and give this as an auxiliary input to the machine. Note that if we know that a machine M will only look at n random bits, then we need only supply M with a random string of length n. For a randomized machine M, we write $M(x)$ to denote the distribution of outputs produced when given x as an input. We write $M(x;r)$ to denote the particular output obtained when the machine M is supplied with random bits r. We say that a randomized machine M is *(strictly) polynomial time* (also abbreviated *PPT*) if there exists a polynomial $p(\cdot)$ such that for every input x, we have that $M(x; r)$ halts within $p(|x|)$ steps, no matter what the random bits r are. On the other hand, we say that a randomized machine M is *expected polynomial time* if there exists a polynomial $p(\cdot)$ such that for every input x, we have that the expected number steps that $M(x)$ takes is at most $p(|x|)$.

Finally, a *non-uniform* machine is one that additional access to an "advice" string, which depends only on the length of the input to the machine. In particular, a *non-uniform polynomial-time* machine is specified by a machine M together with a set of strings $\{\gamma_n\}_{n \in \mathbb{N}}$ with the property that there exists a polynomial $p(\cdot)$ such that for every input x, we have that $M(x, \gamma_{|x|})$ takes at most $p(|x|)$ steps before halting. One similarly defines the notion of a *randomized (or probabilistic) non-uniform polynomial-time* machine.

2.1.3 Standard Complexity Classes and Completeness.

The complexity class P consists of all promise problems Π for which there exists a polynomial-time deterministic machine M such that $M(x) = \texttt{accept}$ if $x \in \Pi_Y$, while $M(x) = \texttt{reject}$ if $x \in \Pi_N$. Note that there are no constraints on inputs that violate the promise. We think of P as consisting of all problems that can be solved efficiently by a deterministic machine. Similarly, the class BPP is the set of all promise problems Π for which there exists a (strictly) probabilistic polynomial-time machine M such that $x \in \Pi_Y$ implies that $\Pr[M(x) = \texttt{accept}] > 2/3$, while $x \in \Pi_N$ implies that $\Pr[M(x) = \texttt{reject}] > 2/3$. Note that standard techniques (see for example [Pap94]) can be used to transform any such machine into one where these probabilities are amplified to $1 - 2^{-|x|^k}$, for any constant k.

The class NP consists of all promise problems Π for which there exist "short efficiently verifiable classical proofs" that a string x is a YES-instance of Π. Formally, we require that there exists a polynomial-time deterministic machine V and a polynomial $\ell(\cdot)$ such that:

- (*Completeness*): If $x \in \Pi_Y$, then there exists $w \in \{0,1\}^{\ell(|x|)}$ such that $M(x, w) = \texttt{accept}$. Here, w is the classical proof or *witness* for x.
- (*Soundness*): If $x \in pin$, then for all w, we have that $M(x, w) = \texttt{reject}$.

We say that promise problem Π *(Karp) reduces* to promise problem Γ if there is a polynomial-time computable function f such that

$$\begin{aligned} x \in \Pi_Y &\Rightarrow f(x) \in \Gamma_Y \\ x \in \Pi_N &\Rightarrow f(x) \in \Gamma_N \end{aligned}$$

If C is a class of promise problems, we say that promise problem Π is *complete* for C if $\Pi \in$ C and every promise problem in C reduces to Π. As above, all reductions we consider are polynomial-time many-one (or Karp) reductions, unless otherwise specified.

2.1.4 Probability distributions

First, we introduce some notation that will be used throughout this thesis. If X is a probability distribution (or random variable), we write $x \leftarrow X$ to indicate that x is a sample taken from X. If S is a set, we write $x \in_R S$ to indicate that x is uniformly selected from S.

In this thesis, we will consider probability distributions defined both by circuits and probabilistic Turing machines as described above. If A is a probabilistic Turing machine, we use $A(x)$ to denote the output distribution of A on input x. If C is a circuit mapping m-bit strings to n-bit strings, then choosing an input u uniformly at random from $\{0,1\}^m$ defines a probability distribution on $\{0,1\}^n$ given by $C(u)$. For notational convenience, we also denote this probability distribution by C. These definitions capture the idea of an "efficiently samplable" distribution, as to sample from the distribution one need only run the Turing machine or evaluate the circuit.

2.1.5 The statistical difference metric

For probability distributions (or random variables) X and Y on a discrete set D, the *statistical difference* between X and Y is denoted by

$$\|X - Y\| = \max_{S \subset D} |\Pr[X \in S] - \Pr[Y \in S]|. \tag{2.1}$$

This is often also called the *variation distance* between X and Y. The maximum in (2.1) can be achieved by taking $S = \{x : \Pr[X = x] > \Pr[Y = x]\}$ (or its complement); this can be seen directly or in the proof of Fact 2.1.1 below.

There is an equivalent formulation of statistical difference in terms of the ℓ_1 norm $|\cdot|_1$ that will sometimes be more convenient for us. To every probability distribution X on a discrete set D, the *mass function* of X is a vector in $\mathbb{R}^D$ whose x'th coordinate is $\Pr[X = x]$. For the sake of elegance, we also denote this vector by X. With this notation, we can state the following well-known fact.

Fact 2.1.1 $\|X - Y\| = \frac{1}{2}|X - Y|_1$

Proof: For any set $S \subset D$,

$$\begin{aligned}
&2|\Pr[X \in S] - \Pr[Y \in S]| \\
&= |\Pr[X \in S] - \Pr[Y \in S]| + |\Pr[X \notin S] - \Pr[Y \notin S]| \\
&= \left|\sum_{x \in S} (\Pr[X = x] - \Pr[Y = x])\right| + \left|\sum_{x \notin S} (\Pr[X = x] - \Pr[Y = x])\right| \\
&\leq \sum_{x \in S} |\Pr[X = x] - \Pr[Y = x]| + \sum_{x \notin S} |\Pr[X = x] - \Pr[Y = x]| \\
&= |X - Y|_1 .
\end{aligned}$$

Equality is achieved by taking $S = \{x : \Pr[X = x] > \Pr[Y = x]\}$. ■

It is immediate from this characterization of statistical difference that it is a metric (as long as we identify random variables that are identically distributed). In particular, it satisfies the Triangle Inequality.

Fact 2.1.2 (Triangle Inequality) *For any probability distributions X, Y, and Z,*

$$\|X - Y\| \leq \|X - Z\| + \|Z - Y\|$$

Recall that for any two vectors $v \in \mathbb{R}^m$ and $w \in \mathbb{R}^n$, their *tensor product* $v \otimes w$ is the vector in $\mathbb{R}^{nm}$, whose (i,j)'th component is $v_i w_j$. Now, if we have a pair of random variables (X, Y) (on the same probability space) taking values in $D \times E$, then X is independent from Y iff the corresponding mass functions satisfy $(X, Y) = X \otimes Y$, where we view the mass functions of X and Y as elements of $\mathbb{R}^D$ and $\mathbb{R}^E$, respectively. For this reason, if we have random variables X and Y taking values in sets D and E, respectively, we write $X \otimes Y$ for the random variable taking values in $D \times E$ which consists of independent samples of X and Y.

Now, for any two vectors v and w, $|v \otimes w|_1 = |v|_1 \cdot |w|_1$. In addition, for any mass function X, $|X|_1 = 1$. These facts enable one to show that the statistical difference behaves well with respect to independent random variables:

Fact 2.1.3 *Suppose X_1 and X_2 are independent random variables on one probability space and Y_1 and Y_2 are independent random variables on another probability space. Then,*

$$||(X_1, X_2) - (Y_1, Y_2)|| \leq ||X_1 - Y_1|| + ||X_2 - Y_2||$$

Proof:

$$\begin{aligned} ||(X_1, X_2) - (Y_1, Y_2)|| &\leq ||(X_1, X_2) - (Y_1, X_2)|| + ||(Y_1, X_2) - (Y_1, Y_2)|| \\ &= \frac{1}{2}|X_1 \otimes X_2 - Y_1 \otimes X_2|_1 + \frac{1}{2}|Y_1 \otimes X_2 - Y_1 \otimes Y_2|_1 \\ &= \frac{1}{2}|(X_1 - Y_1) \otimes X_2|_1 + \frac{1}{2}|Y_1 \otimes (X_2 - Y_2)|_1 \\ &= \frac{1}{2}|X_1 - Y_1|_1 \cdot |X_2|_1 + \frac{1}{2}|Y_1|_1 \cdot |X_2 - Y_2|_1 \\ &= ||X_1 - Y_1|| + ||X_2 - Y_2|| \end{aligned}$$

■

Another basic fact about statistical difference is that for any procedure A, even if it is randomized, the statistical difference between $A(X)$ and $A(Y)$ is no greater than the statistical difference betewen X and Y. Formally, if D is any set, a *randomized procedure* on D is a a pair $A = (f, R)$, where R is a probability distribution on some set E and f is a function from $D \times E$ to any set F. Think of the distribution R as providing a "random seed" to the procedure A. If X is a probability distribution on D, then $A(X)$ denotes the probability distribution on F obtained by sampling $X \otimes R$ and applying f to the result. Note that applying a *function* is a special case of applying a randomized procedure.

Fact 2.1.4 *If X and Y are random variables and A is any randomized procedure, then*

$$||A(X) - A(Y)|| \leq ||X - Y||$$

Proof: Let $A = (f, R)$. Then, for any set $S \subset F$,

$$\begin{aligned} &|\Pr[A(X) \in S] - \Pr[A(Y) \in S]| \\ &= |\Pr[f(X \otimes R) \in S] - \Pr[f(Y \otimes R) \in S]| \\ &= \left|\Pr\left[X \otimes R \in f^{-1}(S)\right] - \Pr\left[Y \otimes R \in f^{-1}(S)\right]\right| \\ &\leq ||X \otimes R - Y \otimes R|| \\ &\leq ||X - Y|| + ||R - R|| \\ &= ||X - Y||. \end{aligned}$$

Taking the maximum over all sets S completes the proof. ■

The next fact is useful when arguing that the statistical difference between two distributions is small.

Fact 2.1.5 *Suppose $X = (X_1, X_2)$ and $Y = (Y_1, Y_2)$ are probability distributions on a set $D \times E$ such that*

1. *X_1 and Y_1 are identically distributed, and*

2. *With probability greater than $(1 - \epsilon)$ over $x \leftarrow X_1$ (equivalently, $x \leftarrow Y_1$),*

$$\|X_2|_{X_1=x} - Y_2|_{Y_1=x}\| < \delta$$

(where $B|_{A=a}$ denotes the conditional distribution of B given that $A = a$ for jointly distributed random variables A and B).

Then $\|X - Y\| < \epsilon + \delta$.

Proof: Let $T \subset D$ be the set of x's for which $\|X_2|_{X_1=x} - Y_2|_{Y_1=x}\| < \delta$. Now, let S be an arbitrary subset of $D \times E$ and, for every $x \in D$, define $S_x = \{y \in E\colon (x, y) \in S\}$. Then,

$$\begin{aligned}
\Pr[X \in S] &\le \Pr[X_1 \notin T] + \sum_{x \in T} \Pr[X_2 \in S_x | X_1 = x] \Pr[X_1 = x] \\
&< \epsilon + \sum_{x \in T} (\Pr[Y_2 \in S_x | Y_1 = x] + \delta) \Pr[Y_1 = x] \\
&\le \epsilon + \delta + \Pr[Y \in S].
\end{aligned}$$

By symmetry, we also have $\Pr[Y \in S] < \epsilon + \delta + \Pr[X \in S]$. Since S was arbitrary, $\|X - Y\| < \epsilon + \delta$. ∎

The next fact formalizes the intuition that if two distributions have small statistical difference, then their mass functions must be close at most points.

Fact 2.1.6 *If X and Y are any two distributions such that $\|X - Y\| < \epsilon$, then with probability $> 1 - 2\sqrt{\epsilon}$ over $x \leftarrow X$,*

$$(1 - \sqrt{\epsilon}) \Pr[X = x] < \Pr[Y = x] < (1 + \sqrt{\epsilon}) \Pr[X = x]$$

Proof: Let $S = \{x\colon (1 - \sqrt{\epsilon}) \Pr[X = x] \ge \Pr[Y = x]\}$, *i.e.* the set of x's for which the left-hand inequality in Fact 2.1.6 is violated. Then,

$$\begin{aligned}
\Pr[Y \in S] &\le (1 - \sqrt{\epsilon}) \Pr[X \in S] \\
&= \Pr[X \in S] - \sqrt{\epsilon} \Pr[X \in S].
\end{aligned}$$

Thus, $\|X - Y\| \ge \sqrt{\epsilon} \Pr[X \in S]$,so we must have $\Pr[X \in S] < \sqrt{\epsilon}$. A similar argument show that the right-hand inequality in Fact 2.1.6 is violated with probability less than $\sqrt{\epsilon}$. ∎

2.1.6 Computational Indistinguishability

The notion of statistical distance gives a natural metric on distributions, which lets us compare how similar two distributions are in an absolute sense. In particular, recall that a function $\mu(n)$ is *negligible* if for all polynomials $p(n)$, $\mu(n) \le \frac{1}{p(n)}$ for all sufficiently large n. Suppose we had two ensembles of distributions $\{A_k\}_{k \in \mathbb{N}}$ and $\{B_k\}_{k \in \mathbb{N}}$. If we knew that $\|A_k - B_k\| \le \mu(k)$ for some negligible function μ, this would indicate that these two ensembles of distributions are nearly identical to each other (in an asymtotic sense).

We now consider a computational notion of two ensembles of distributions being nearly identical. We say that two ensembles of distributions $\{A_k\}_{k\in\mathbb{N}}$ and $\{B_k\}_{k\in\mathbb{N}}$ are *computationally indistinguishable* [GM84, Yao82] if for every non-uniform probabilistic polynomial-time machines D (called a "distinguisher"), there exists a negligible function $\mu(\cdot)$ such that for all k:

$$\left|\Pr\left[D(1^k, A_k) = 1\right] - \Pr\left[D(1^k, B_k) = 1\right]\right| \leq \mu(k).$$

Similarly, we say two ensembles of distributions indexed by strings $\{A_x\}_{x\in L}$ and $\{B_x\}_{x\in L}$ are *computationally indistinguishable* if for every non-uniform probabilistic polynomial-time distinguisher D there exists a negligible function $\mu(\cdot)$ such that for all $x \in L$:

$$|\Pr[D(x, A_x) = 1] - \Pr[D(x, B_x) = 1]| \leq \mu(|x|).$$

2.2 Zero-knowledge proofs

2.2.1 Interactive Protocols and Proofs

Before defining zero-knowledge, we need to first introduce the notion of a protocol between two interacting machines. We use the formalism given by Goldwasser and Sipser [GS89]:

Definition 2.2.1 *An* interactive protocol (P, V) *is a pair of probabilistic machines P and V. The* interaction *of P and V on common input x (with private inputs p_P for P and p_V for V, sometimes omitted) is a random process which proceeds as follows:*

1. *Choose random strings r_P and r_V uniformly among infinite strings over $\{0, 1\}$.*
2. *For $i = 1, 2, \ldots$ do:*
   (a) *If i is odd, let $m_i = P(p_A, x, m_1, \ldots, m_{i-1}; r_A)$.*
   (b) *If i is even, let $m_i = V(p_B, x, m_1, \ldots, m_{i-1}; r_B)$.*
   (c) *If $m_i \in \{\texttt{accept}, \texttt{reject}, \texttt{halt}\}$, then stop.*

If `accept` *is the last message, then we say that the party which output it* accepts *the interaction, and similarly for* `reject`. *Such a protocol is* polynomially bounded *if there is a polynomial $p(\cdot)$ such that the protocol halts within $p(|x|)$ exchanges of messages.*

Furthermore, we denote by $\mathrm{View}_{P,V}(x)$ *the random variable which represents the view of V in its interaction with P. Namely, this variable consists of the private input p_V to V together with its random coins r_V, along with all messages m_i exchanged between the two parties.*

We follow [GMR89] and [Gol95] in defining interactive proofs and zero-knowledge. The original definitions in [GMR89] were given for languages. We generalize these definitions to promise problems in the natural way, as previously done in [GK93]. That is, conditions previously required for inputs in the language are now required for YES instances of a promise problem and conditions previously required for inputs not in the language are now required for NO instances.

Informally, an interactive proof is a protocol in which a possibly computationally unbounded prover[1] attempts to convince a polynomial-time verifier V that an assertion is true, *i.e.* that a string x is a YES instance of a promise problem. More formally, we have:

Definition 2.2.2 *An interactive protocol (P, V) between a computationally unbounded prover P and a probabilistic polynomial-time verifier V is said to be an* interactive proof system *for a promise problem Π with* completeness error $c(n)$ *and* soundness error $s(n)$ *if*

1. *(*Completeness*): If $x \in \Pi_Y$, then* $\Pr[(P, V)(x) = \texttt{accept}] \geq 1 - c(|x|)$.
2. *(*Soundness*): If $x \in \Pi_N$, then for all P^*,* $\Pr[(P^*, V)(x) = \texttt{accept}] \leq s(|x|)$.

Note that as long as $1 - c(n) > s(n) + 1/\text{poly}(n)$ and that both $c(n)$ and $s(n)$ can be computed in time $\text{poly}(n)$, parallel repetition can be used to obtain a new interactive proof for Π with completeness error and soundness error 2^{-n^k}, for any constant k. We call the *number of rounds* in an interactive protocol to be the total number of messages exchanged between the prover and the verifier. An interactive proof system is said to be *public coin* (also known as *Arthur-Merlin*) if on every input, the verifier's random coins r can be written as a concatenation of strings $r_1 r_2 \cdots r_l$ such that the i'th message sent from the verifier to the prover is simply r_i. In this case, sometimes the verifier is called Arthur, and the prover is called Merlin [BM88]. When we present public-coin protocols, we often use the Arthur-Merlin naming convention to emphasize the public-coin nature of these protocols.

2.2.2 Honest-Verifier Zero-Knowledge Proofs

Roughly speaking, an interactive proof is said to be zero-knowledge if, when the input is a YES instance, the verifier can simulate its view of the interaction on its own. Intuitively, then, whatever knowledge the verifier can obtain from the interaction with the prover, it could in fact obtain completely on its own – by simply performing the simulation. A crucial point here concerns which verifiers' interaction we can simulate. If the interaction of the verifier V given in the protocol description with the prover P can be simulated, then we call the protocol an honest-verifier zero-knowledge proof – since in this case, we know that following the protocol honestly prevents gaining any knowledge one could not obtain on one's own. If on the other hand, we can simulate the interaction of any verifier with the prover, then we call the protocol a (general) zero-knowledge proof.

We will first discuss honest-verifier zero-knowledge proofs. There are three kinds of such proofs, depending on the quality of the simulation.

Definition 2.2.3 *Let (P, V) be an interactive proof system with negligible completeness and soundness errors, for a promise problem Π.*

- *(P, V) is said to be an* honest-verifier statistical zero-knowledge *proof system if there exists a probabilistic polynomial-time[2] simulator S and a negligible function α (called the* simulator deviation*) such that*

$$\text{For all } x \in \Pi_Y, \text{ we have } \|S(x) - \text{View}_{P,V}(x)\| \leq \alpha(|x|). \qquad (2.2)$$

[1] We will later discuss the notion of a *computationally-sound proof*, also known as an *argument*, where the prover is assumed to be limited to polynomial time.

[2] The definition of each type of zero knowledge can also be made allowing the simulator to run in expected polynomial time instead of strict polynomial time. See the remarks after the definition.

- *A* honest-verifier perfect zero-knowledge *proof system is defined in the same way, except that Condition (2.2) is replaced by* $\|S(x) - \text{View}_{P,V}(x)\| = 0$*, where* S *is allowed to output '*`fail`*' with probability at most* $1/2$ *and* $S(x)$ *denotes the conditional distribution of* S *given that the output is not* `fail`.[3]

- *A* honest-verifier computational zero-knowledge *proof system replaces Condition (2.2) with the requirement that the ensembles* $\{S(x)\}_{x \in \Pi_Y}$ *and* $\{\text{View}_{P,V}(x)\}_{x \in \Pi_Y}$ *are computationally instinguishable ensembles of distributions.*

We let HVSZK *(resp.* HVPZK, HVCZK*) denote the class of promise problems with statistical (resp. perfect, computational) zero-knowledge proof systems against the honest verifier.*

Remark 2.2.1 We make several remarks about this definition:

1. (Honest Verifiers) In this definition, we only require that the zero-knowledge condition to hold against the honest verifier, *i.e.* the verifier that follows the protocol as specified. The usual definition requires the zero-knowledge property to hold against any polynomial-time verifier strategy. However, we will show that any proof system which is statistical zero-knowledge against the honest verifier can be transformed into one that is zero-knowledge against any verifier. Via this transformation, many of our results about HVSZK directly translate to the class of promise problems possessing statistical zero-knowledge proofs against any verifier.

2. (Error Probabilities) Note that we have required the completeness and soundness errors to be negligible. However, it is easy to see that zero-knowledge *against an honest verifier* is preserved under parallel repetition. Therefore, starting with any completeness and soundess errors such that $1 - c(n) > s(n) + 1/poly(n)$, the completeness and soundness error probabilities can be made exponentially small without increasing the number of rounds, by parallel repetition (see for instance, [Gol95]).

3. (Strict vs. Expected Polynomial-time Simulation) We note that zero knowledge has been defined in the past in two ways. [Gol95] works with the variant of zero knowledge in which the simulator is required to run in *strict* polynomial time, with some probability of failure in the perfect case. The original definition in [GMR89] allows the simulator to run in expected polynomial time, but with zero probability of failure. Note that for honest-verifier statistical zero knowledge, the choice to restict to strict polynomial time is not very restrictive, because we are only discussing honest-verifier statistical zero-knowledge and we do not know of any proof systems which require an expected polynomial time simulator for the honest verifier. In addition, as shown in Section 3.3.5 of Chapter 3, our techniques can be used to prove that expected polynomial time simulators and strict polynomial time simulators are actually *equivalent* for public-coin statistical zero-knowledge proofs against an honest verifier.

4. (Promise Problems vs. Languages) Our definitions above generalize the original definitions of [GMR89] from languages to promise problems, and we will focus on the "promise class" HVSZK rather than the class of languages possessing statistical zero-knowledge proofs. A couple of justifications can be given for this extension. First, for essentially all of our results about statistical zero knowledge, the fact that we prove them for the promise class only makes them stronger, by virtue of the fact that the promise class contains the language class. Second,

[3]A failure probability can also be allowed in the definition of statistical zero-knowledge, but this can easily be reduced to an 2^{-n^k} for any constant k by repeated trials and absorbed in to the simulator deviation.

several of the most important natural problems known to be in HVSZK, such as those in [GK93, GG98], are not languages, but promise problems, so it may actually be preferable to study the promise class.

Our only result which requires new interpretation for the language class is the Completeness Theorem. As the complete problem is a promise problem, it is not complete for the language class in the usual sense. Nevertheless, it still gives a characterization of the language class, in that a language has a statistical zero-knowledge proof *if and only if* it reduces to the complete problem.

We note that one must be a bit more careful in a complexity-theoretic investigation of promise classes, particularly when discussing reductions that may violate the promise (cf., discussions in [ESY84, GG98]), and it may be the case that the language class has some different properties than the promise one.

2.2.3 General Zero-Knowledge Proofs

We now consider how to define proofs that remain zero knowledge against all verifier strategies. We start with the simplest extension of the definition above to handle the any-verifier case, following [GMR89].

Definition 2.2.4 *Let (P, V) be an interactive proof system with negligible completeness and soundness errors, for a promise problem* Π.

- (P, V) *is said to be a* (general) statistical zero-knowledge *proof system if for every non-uniform probabilistic polynomial-time verifier V^*, there exists a probabilistic polynomial-time*[4] *simulator S and a negligible function α (called the* simulator deviation*) such that*

$$\textit{For all } x \in \Pi_Y, \textit{ we have } \|S(x) - \mathrm{View}_{P,V^*}(x)\| \leq \alpha(|x|). \tag{2.3}$$

- *A* perfect zero-knowledge *proof system is defined in the same way, except that Condition (2.3) is replaced by $\|S(x) - \mathrm{View}_{P,V^*}(x)\| = 0$, where S is allowed to output '`fail`' with probability at most $1/2$ and $S(x)$ denotes the conditional distribution of S given that the output is not* `fail`.[5]

- *A* computational zero-knowledge *proof system replaces Condition (2.3) with the requirement that the ensembles $\{S(x)\}_{x\in\Pi_Y}$ and $\{\mathrm{View}_{P,V^*}(x)\}_{x\in\Pi_Y}$ are computationally instinguishable ensembles of distributions.*

It may seem that the definition above would be very difficult to achieve – how can we construct a simulator for each of the infinitely many possible verifier strategies? The way this has been done in all known general zero-knowledge proofs is to actually exhibit one universal simulator, which when given "black box" access to any verifier strategy, is able to produce a simulation of that verifier with the prover. This notion of "black box zero knowledge" was first formalized by Goldreich and Oren [GO94]. We state it formally below:

[4]As in the honest-verifier case, the definition of each type of zero knowledge can also be made allowing the simulator to run in expected polynomial time instead of strict polynomial time.

[5]As noted for the honest-verifier case, a failure probability can also be allowed in the definition of statistical zero-knowledge, but this can easily be reduced to an 2^{-n^k} for any constant k by repeated trials and absorbed in to the simulator deviation.

Definition 2.2.5 *Let (P, V) be an interactive proof system with negligible completeness and soundness errors, for a promise problem Π.*

- *(P, V) is said to be a* (general) black-box statistical zero-knowledge *proof system if there exists a probabilistic polynomial-time*[6] *(oracle machine) simulator S and a negligible function α (called the* simulator deviation*) such that for every non-uniform probabilistic polynomial-time verifier V^*, we have:*

$$\text{For all } x \in \Pi_Y, \text{ we have } \left\| S^{V^*}(x) - \text{View}_{P,V^*}(x) \right\| \leq \alpha(|x|). \qquad (2.4)$$

- *A* black-box perfect zero-knowledge *proof system is defined in the same way, except that Condition (2.4) is replaced by $\left\| S^{V^*}(x) - \text{View}_{P,V^*}(x) \right\| = 0$, where S is allowed to output '`fail`' with probability at most $1/2$ and $S(x)$ denotes the conditional distribution of S given that the output is not* `fail`.[7]

- *A* black-box computational zero-knowledge *proof system replaces Condition (2.4) with the requirement that the ensembles $\{S^{V^*}(x)\}_{x \in \Pi_Y}$ and $\{\text{View}_{P,V^*}(x)\}_{x \in \Pi_Y}$ are computationally instinguishable ensembles of distributions.*

We let SZK *(resp.* PZK, CZK*) denote the class of promise problems with (general) black-box statistical (resp. perfect, computational) zero-knowledge proof systems.*

Remark 2.2.2 Note that in this black-box model, we can also make sense of what it means to allow unlimited verifiers in the case of statistical or perfect zero-knowledge proofs. We can say that a proof system is black-box statistical (resp. perfect) zero knowledge *against arbitrary verifier strategies* if it satisfies Condition (2.4) for all possible verifier machines V^*.

We will omit other definitions until later chapters, where they will be needed.

[6] Again, the definition of each type of zero knowledge can also be made allowing the simulator to run in expected polynomial time instead of strict polynomial time.

[7] Again, a failure probability can also be allowed in the definition of statistical zero-knowledge, but this can easily be reduced to an 2^{-n^k} for any constant k by repeated trials and absorbed in to the simulator deviation.

Chapter 3

A Complete Problem for Statistical Zero Knowledge

A revolution in theoretical computer science occurred when it was discovered that NP has complete problems [Coo71, Lev73, Kar72]. Most often, this theorem and other completeness results are viewed as negative statements, as they provide evidence of a problem's intractability. These same results, viewed as positive statements, enable one to study an entire class of problems by focusing on a single problem. For example, all languages in NP were shown to have computational zero-knowledge proofs when such a proof was exhibited for GRAPH 3-COLORABILITY [GMW91]. Similarly, the result that IP = PSPACE was shown by giving an interactive proof for QUANTIFIED BOOLEAN FORMULA, which is complete for PSPACE [LFKN90, Sha90]. More recently, the celebrated PCP theorem characterizing NP was proven by designing efficient probabilistically checkable proofs for a specific NP-complete language [FGL$^+$96, AS92, ALM$^+$92].

In this chapter, we present a complete problem for HVSZK, the class of promise problems possessing statistical zero-knowledge proofs (against the honest verifier)[1]. This problem provides a new and simple characterization of HVSZK — one which makes no reference to interaction or zero knowledge. We propose the use of complete problems as a tool to unify and extend the study of statistical zero knowledge. To this end, we use our complete problem to both establish a number of new results about HVSZK and easily deduce nearly all previous results about HVSZK. The work we present in this chapter is based on two papers [SV97, SV99] authored jointly with Salil Vadhan.

3.1 Overview

The promise problem we show to be complete for HVSZK is STATISTICAL DIFFERENCE. An instance of STATISTICAL DIFFERENCE consists of a pair of probability distributions, specified by circuits which sample from them. Roughly speaking, the problem is to decide whether the distributions defined by the two circuits are statistically close or far apart. (The gap between 'close' and 'far apart' is what makes it a promise problem and not just a language.) Our main theorem is that STATISTICAL DIFFERENCE is complete for HVSZK. This Completeness Theorem gives a new characterization of HVSZK. Informally, it says that the assertions that can be proven in statistical zero knowledge are exactly those that can be cast as deciding whether a pair of efficiently sampleable distributions are statistically close or far apart.

[1]In this chapter, whenever we write "statistical zero-knowledge proof," we mean one that is zero knowledge against an honest verifier. In the next chapter, we will show that this restriction is without loss of generality.

The starting point for our proof of the Completeness Theorem is a powerful theorem of Okamoto [Oka96], which states that all languages in HVSZK have *public-coin* (also known as Arthur-Merlin [BM88]) statistical zero-knowledge proofs. Using the approach pioneered by Fortnow [For89], we analyze the simulator of such a proof system and show that statistical properties of the simulator's output distribution can be used to distinguish between YES and NO instances of the problem in consideration. Our key new observation is that, for a *public-coin* proof system, these statistical properties can be captured by the statistical difference between efficiently sampleable distributions. We thereby conclude that every problem in HVSZK reduces to STATISTICAL DIFFERENCE.

To show that STATISTICAL DIFFERENCE is in HVSZK, we exhibit a simple 2-message proof system for it, generalizing the well-known proof systems for QUADRATIC NONRESIDUOSITY [GMR89] and GRAPH NONISOMORPHISM [GMW91]. One ingredient in our proof system is a new "Polarization Lemma" for statistical difference, which may be of independent interest. Roughly speaking, this lemma gives an efficient transformation which takes as input a pair of probability distributions (specified by circuits which sample from them) and produces a new pair of distributions such that if the original pair is statistically close (resp., far apart), the new pair is statistically much closer (resp., much further apart).

3.1.1 Consequences

We propose using complete problems, such as STATISTICAL DIFFERENCE, to unify and extend the study of HVSZK. We also use the connection between HVSZK and statistical properties of samplable distributions to establish new techniques for manipulating such distributions. The results we obtain along these lines are summarized below.

The relationship between HVSZK and BPP. Our complete problem illustrates that statistical zero knowledge is a natural generalization of BPP. In the definition of STATISTICAL DIFFERENCE , the circuits can output strings of any length. If we restrict the circuits to have output of logarithmic length, the resulting problem is easily shown to be complete for BPP.

Efficient HVSZK proof systems. The zero-knowledge proof system we exhibit for STATISTICAL DIFFERENCE has many attractive properties (which we describe shortly); by the Completeness Theorem it follows that every problem in HVSZK also has a proof system with such properties. First, the protocol is very communication-efficient — only two messages are exchanged between the prover and verifier, and the prover only sends *one bit* to the verifier (to achieve soundness error $1/2$). In addition, we will show that when the input is a YES instance, the verifier's view of the interaction can be simulated by a polynomial-time simulator with *exponentially small* statistical deviation. Moreover, we will show that this simulator deviation can be made to shrink exponentially fast as a function of a separate "security parameter" which can be varied independently from the assertion being proven. This is in contrast to the definition of HVSZK, which only requires that the verifier be able simulate the interaction with statistical deviation $1/n^{\omega(1)}$, where n is the the length of the assertion being proven.

Closure properties. Using the complete problem, we demonstrate that HVSZK has some very strong closure properties. First, our proof of the completeness theorem contains a much simpler proof that HVSZK is closed under complement, first shown by Okamoto [Oka96]. We also immediately deduce the theorems showing that all problems in HVSZK must have relatively low computational complexity, first proven in [For89, AH91]. Then, in Section 3.3.2, we use the complete problem to prove much stronger boolean closure properties. These can be informally described as asserting the existence of statistical zero-knowledge proofs for complex assertions built out simpler assertions already known to be in HVSZK. These complex assertions take the form of arbitrary propositional formulae whose atoms are statements about membership in some problem in HVSZK, and the statistical zero-knowledge proofs we exhibit have complexity which is polynomial in the size of these formulae. These results generalize earlier ones of De Santis, Di Crescenzo, Persiano, and Yung [DDPY94] and Damgård and Cramer [DC96], which held for monotone formulae and various subclasses of HVSZK, such as random self-reducible problems.

By the Completeness Theorem, the closure properties we exhibit are equivalent to the existence of efficient transformations that manipulate the statistical difference between sampleable distributions in various ways. Indeed, it is by exhibiting such transformations that we prove these closure properties. The transformations we give (and their application to closure properties) are inspired by the techniques of De Santis, Di Crescenzo, Persiano, and Yung [DDPY94].

Knowledge complexity. In addition to introducing zero-knowledge proofs, the conference version of the paper of Goldwasser, Micali, and Rackoff [GMR89] proposed a more general idea of measuring the amount of knowledge leaked in an interactive proof. Goldreich and Petrank [GP91] suggested several definitions of *knowledge complexity* to accomplish this, and relationships between these various types of knowledge complexity were explored in [GP91, BP94, GOP98, ABV95, PT96]. Loosely speaking, the definitions of (statistical) knowledge complexity measure the "amount of help" a verifier needs to generate a distribution that is statistically close to its real interaction with the prover. There are several ways of formalizing the "amount of help" the verifier needs and each leads to a different notion of knowledge complexity.

Our work on HVSZK turns out to have consequences for (non-zero) knowledge complexity as well. First, we show that for the weakest of the various measures of knowledge complexity, namely statistical knowledge complexity in the "hint sense", the corresponding hierarchy collapses by logarithmic additive factors at all levels, and in particular, knowledge complexity $\log n$ equals statistical zero knowledge. No collapse was previously known for any of the variants of knowledge complexity suggested in [GP91]. Our results are obtained by combining our results on HVSZK with a general lemma relating knowledge complexity in the hint sense to zero knowledge *for promise problems*.

As with zero knowledge, *perfect* knowledge complexity can also be defined. This measures the number of bits of help the verifier needs to simulate the interaction *exactly*, rather than statistically closely. Using our complete problem for HVSZK, we are able to subsume the relevant previous results of [ABV95] to give tighter bounds on the perfect knowledge complexity of statistical zero knowledge.

Reversing statistical difference. One interesting result that follows from the completeness of STATISTICAL DIFFERENCE and the closure of HVSZK under complement is the existence of an efficient mapping which "reverses" statistical difference. That is, for every pair of efficiently samplable distributions, we can construct another pair of efficiently samplable distributions such that when the former are statistically close, the latter are statistically far apart, and when the former are far apart, the latter are close.

This motivated us to search for a more explicit description of such a transformation. By extracting ideas from the work of Okamoto [Oka96] and our proof of the Completeness Theorem, we have obtained such a description which we give in Section 3.3.4.

Weak HVSZK and expected polynomial-time simulators. The original definition of HVSZK in [GMR89] allows the simulator to run in *expected* polynomial time, whereas we insist on strict polynomial time, following [Gol95]. Actually, our proof of the Completeness Theorem shows that the two definitions are equivalent for *public-coin* proof systems. That is, if a problem possesses a public-coin HVSZK proof system with an expected polynomial-time simulator, then it also possesses an HVSZK proof system with a strict polynomial-time simulator (which can be made public coin by [Oka96]). In fact, the equivalence even extends to an even weaker definition of HVSZK, in which it is only required that for every polynomial $p(n)$, there exists a simulator achieving simulator deviation $1/p(n)$.

Perfect and computational zero knowledge. Our techniques can also be used to analyze public-coin perfect and computational zero-knowledge proofs. Although we do not obtain complete problems in these cases, we do obtain some novel insights into the corresponding complexity classes. Specifically, in Section 3.3.6 we show that every problem possessing a public-coin perfect zero-knowledge proof (essentially) reduces to a restricted version of STATISTICAL DIFFERENCE. We also show that for any problem possessing a public-coin computational zero-knowledge proof, there exist ensembles of sampleable distributions indexed by instances of the problem such that on YES instances, the distributions are computationally indistinguishable and on NO instances, the distributions are statistically far apart.

Hard-on-average languages and one-way functions. Ostrovsky [Ost91] showed that if any "hard-on-average" problem possesses a statistical zero-knowledge proof, then one-way functions exist. Combining the Completeness Theorem with a result of Goldreich [Gol90] on computational indistinguishability, we give a simpler proof of Ostrovsky's result.

3.2 The Completeness Theorem

3.2.1 The complete problem

The main aim of this chapter is to demonstrate that HVSZK consists exactly of the problems that involve deciding whether two efficiently samplable distributions are either far apart or close together. This can be formally described as the following promise problem STATISTICAL DIFFERENCE (abbreviated SD):

$$\begin{aligned} \mathrm{SD}_Y &= \left\{(C_0, C_1) : \|C_0 - C_1\| > \frac{2}{3}\right\} \\ \mathrm{SD}_N &= \left\{(C_0, C_1) : \|C_0 - C_1\| < \frac{1}{3}\right\} \end{aligned}$$

In the above definition, C_0 and C_1 are circuits; these define probability distributions as discussed in Chapter 2. The thresholds of $1/3$ and $2/3$ in this definition are not arbitrary; it is important for the Polarization Lemma of Section 3.2.2 that $(2/3)^2 > 1/3$.

We can now state the main theorem of this chapter.

Theorem 3.2.1 (Completeness Theorem) SD *is complete for* HVSZK.

The most striking thing about Theorem 3.2.1 is that it characterizes statistical zero-knowledge *with no reference to interaction.* Future investigation of the properties of HVSZK as a class can focus on the single problem SD, instead of dealing with complicated protocols and arbitrary languages.

We emphasize that the importance of this result lies in the specific complete problem we present and not simply the *existence* of a complete promise problem. It is fairly straightforward to construct a complete promise problem for HVPZK involving descriptions of Turing machines for the verifier and simulator (though not for HVSZK). However, in contrast to SD, a complete problem constructed in this manner is essentially restatement of the definition of the class and therefore does not simplify the study of the class at all.

The proof of Theorem 3.2.1 will come in Sections 3.2.3 and 3.2.4 via two lemmas and a theorem of Okamoto [Oka96]. But first, we observe that a statement analogous to Theorem 3.2.1 can be made for BPP, if we generalize BPP to promise problems in the obvious way.

Proposition 3.2.1 *If* SD' *is the promise problem obtained by modifying the definition of* SD *so that* C_0 *and* C_1 *only have 1 bit of output, then* SD' *is complete for* BPP.

Proof: To see that SD' is in BPP, first observe that for circuits C_0 and C_1 (or any random variables) that just output 0 or 1,

$$\|C_0 - C_1\| = |\Pr[C_0 = 1] - \Pr[C_1 = 1]|.$$

Thus, an estimate on $\|C_0 - C_1\|$ that is correct within an additive factor of 1/3 can be obtained by sampling C_0 and C_1 polynomially many times and counting the number of ones that occur for each. This is sufficient to decide SD'.

Now we show that every promise problem Π in BPP reduces to SD'. Let A be the PPT machine which outputs 1 with probability greater than $2/3$ when $x \in \Pi_Y$, but outputs 1 with probability less than $1/3$ when $x \in \Pi_N$. Let $p(n)$ be a polynomial bound on the running time of A. Given an input x, we can, by standard techniques,[2] produce in polynomial time a circuit C_x describing the computation of A on x for $p(|x|)$ steps. The input to C_x is the first $p(|x|)$ bits on the random tape of A the output is the first bit on the output tape. Let D be a circuit that always outputs 0. Then $\|C_x - D\| = \Pr[A(x) = 1]$, so $x \mapsto (C_x, D)$ is a polynomial-time reduction from Π to SD'. ■

Proposition 3.2.1 remains true even if we allow C_0 and C_1 to output strings of logarithmic length. Other classes such as P and co-RP can be obtained by modifying the definition of SD in a similar fashion (and changing the thresholds). This demonstrates that HVSZK is a natural generalization of these well-known classes.

3.2.2 A polarization lemma

In this section, we exhibit a transformation which "polarizes" the statistical relationship between two distributions. That is, pairs of distributions which are statistically close become much closer and pairs of distributions which are statistically far apart become much further apart.

Lemma 3.2.1 (Polarization Lemma)[3] *There is a polynomial-time computable function that takes a triple* $(C_0, C_1, 1^k)$, *where* C_0 *and* C_1 *are circuits, and outputs a pair of circuits* (D_0, D_1) *such that*

$$\begin{aligned} \|C_0 - C_1\| < 1/3 &\Rightarrow \|D_0 - D_1\| < 2^{-k} \\ \|C_0 - C_1\| > 2/3 &\Rightarrow \|D_0 - D_1\| > 1 - 2^{-k} \end{aligned}$$

[2]See, for example, [Pap94, Thms. 8.1 and 8.2].

[3]The Polarization Lemma stated here is called the Amplification Lemma in [SV97]. We change the name here to stress that the Polarization Lemma does not merely increase statistical difference.

The usefulness of the Polarization Lemma comes from the fact that the two distributions it produces can be treated almost as if they were identically distributed or disjoint (*i.e.* statistical difference 0 and 1, respectively). Indeed, it will be essential in proving that SD (with thresholds of $2/3$ and $1/3$, as we've defined it) is in HVSZK and we will make further us of it in deriving consequences of Theorem 3.2.1.

Superficially, it may seem that a Chernoff bound argument is all that is needed to prove Lemma 3.2.1. However, Chernoff bounds are primarily useful for distinguishing between two events. This corresponds to *increasing* statistical difference, as formalized in the following "direct product" lemma:

Lemma 3.2.2 (Direct Product Lemma) *Let X and Y be distributions such that $\|X - Y\| = \epsilon$. Then for all k,*

$$k\epsilon \geq \| \otimes^k X - \otimes^k Y\| \geq 1 - 2e^{-k\epsilon^2/2}$$

Proof: The upper bound of $k\epsilon$ follows immediately from Fact 2.1.3, so we proceed to the proof of the lower bound. Recall, from the definition of statistical difference, that there must exist a set S such that

$$\Pr[X \in S] - \Pr[Y \in S] = \epsilon.$$

Let $p = \Pr[Y \in S]$. Then, $\Pr[X \in S] = p + \epsilon$. Hence, in k independent samples of X, the expected number of samples that lie in S is $(p+\epsilon)k$, whereas in k independent samples of Y, the expected number of samples that lie in S is pk. The Chernoff bound[4] tells us that the probability that *at least* $(p + \frac{\epsilon}{2})k$ components of $\otimes^k Y$ lie in S is at most $\exp(-k\epsilon^2/2)$, whereas the probability that *at most* $(p + \frac{\epsilon}{2})k$ components of $\otimes^k X$ lie in S is at most $\exp(-k\epsilon^2/2)$. Let S' be the set of all k-tuples that contain more than $(p + \frac{\epsilon}{2})k$ components that lie in S. Then we have,

$$\| \otimes^k X - \otimes^k Y\| \geq \Pr\left[\otimes^k X \in S'\right] - \Pr\left[\otimes^k Y \in S'\right] \geq 1 - 2e^{-k\epsilon^2/2}.$$

■

At first glance, it may seem that the proof above is unnecessarily loose, and that one might be able to prove that the statistical difference increases even for small values of k. However, as pointed out to us by Madhu Sudan, one cannot hope for this: For any $p \in [0,1]$, there exist distributions X and Y such that $\|X \otimes X - Y \otimes Y\| = \|X - Y\| = p$. (Take X and Y to be the distributions on $\{0,1\}$ which are 1 with probabilities $(1 \pm p)/2$.)

Notice that the Direct Product Lemma 3.2.2 is *not* sufficient to prove the Polarization Lemma, because it always increases statistical difference, whereas we would like to increase statistical difference in some cases and decrease it in others. However, it does drive larger values of the statistical difference to 1 more quickly than it drives smaller values to 1, so it is a step in the right direction. The following lemma provides a complementary technique which decreases the statistical difference to 0, with small values going to 0 faster than large values.

Lemma 3.2.3 (XOR Lemma) *There is a polynomial-time computable function that maps a triple $(C_0, C_1, 1^k)$, where C_0 and C_1 are circuits, to a pair of circuits (D_0, D_1) such that $\|D_0 - D_1\| = \|C_0 - C_1\|^k$. Specifically, D_0 and D_1 are defined as follows:*

[4]For the formulation of the Chernoff bound we use, see, for example, the formulation of Hoeffding's inequality in [Hof95, Sec. 7.2.1].

D_0: *Uniformly select* $(b_1, \ldots, b_k) \in \{0,1\}^k$ *such that* $b_1 \oplus \cdots \oplus b_k = 0$, *and output a sample of* $C_{b_1} \otimes \cdots \otimes C_{b_k}$.
D_1: *Uniformly select* $(b_1, \ldots, b_k) \in \{0,1\}^k$ *such that* $b_1 \oplus \cdots \oplus b_k = 1$, *and output a sample of* $C_{b_1} \otimes \cdots \otimes C_{b_k}$.

In order to prove this lemma, we employ a generalization of the technique used in [DDPY94] to represent the logical AND of statements about GRAPH NONISOMORPHISM. This tool is described in the following Proposition.

Proposition 3.2.2 *Let* X_0, X_1, Y_0, Y_1 *be any random variables, and define the following pair of random variables:*

Z_0*: Choose* $a, b \in_R \{0,1\}$ *such that* $a \oplus b = 0$*. Output a sample of* $X_a \otimes Y_b$.
Z_1*: Choose* $a, b \in_R \{0,1\}$ *such that* $a \oplus b = 1$*. Output a sample of* $X_a \otimes Y_b$.

Then $\|Z_0 - Z_1\| = \|X_0 - X_1\| \cdot \|Y_0 - Y_1\|$.

The statistical difference between X_0 and X_1 (or Y_0 and Y_1) measures the advantage a computationally unbounded party has, over random guessing, of guessing b given a sample from X_b, where b is selected uniformly from $\{0,1\}$. Intuitively, the above Proposition says that the advantage one has in guessing the XOR of two independent bits is the product of the advantages one has for guessing each individual bit.

Proof:

$$\begin{aligned}
\|Z_0 - Z_1\| &= \frac{1}{2}|Z_0 - Z_1|_1 \\
&= \frac{1}{2}\left|\left(\frac{1}{2}X_0 \otimes Y_0 + \frac{1}{2}X_1 \otimes Y_1\right) - \left(\frac{1}{2}X_1 \otimes Y_0 + \frac{1}{2}X_0 \otimes Y_1\right)\right|_1 \\
&= \frac{1}{4}|(X_0 - X_1) \otimes (Y_0 - Y_1)|_1 \\
&= \left(\frac{1}{2}|X_0 - X_1|_1\right) \cdot \left(\frac{1}{2}|Y_0 - Y_1|_1\right) \\
&= \|X_0 - X_1\| \cdot \|Y_0 - Y_1\|
\end{aligned}$$

■

Proposition 3.2.2 and an induction argument establish Lemma 3.2.3. Yao's XOR Lemma [Yao82] (see also [GNW95]) can be seen as an analogue of Lemma 3.2.3 in the computational setting, where the analysis is much more difficult.

Now we combine the Direct Product and XOR constructions of Lemmas 3.2.2 and 3.2.3 to prove Lemma 3.2.1. The Direct Product construction gives a way to increase statistical difference with large values going to 1 faster than small values. Similarly, the XOR Lemma shows how to decrease statistical difference with small values going to 0 faster than large values. Intuitively, alternating these procedures should "polarize" large and small values of statistical difference, pushing them closer to 1 and 0, respectively. A similar alternation between procedures with complementary effects was used by Ajtai and Ben-Or [AB84] to amplify the success probability of randomized constant-depth circuits.

Proof: Let $\ell = \lceil \log_{4/3} 6k \rceil$. Apply Lemma 3.2.3 to the triple $(C_0, C_1, 1^\ell)$ to produce (C_0', C_1') such that if

$$\begin{aligned} \|C_0 - C_1\| < 1/3 &\Rightarrow \|C_0' - C_1'\| < (1/3)^\ell \\ \|C_0 - C_1\| > 2/3 &\Rightarrow \|C_0' - C_1'\| > (2/3)^\ell. \end{aligned}$$

Let $m = 3^{\ell-1}$. Let $C_0'' = \otimes^m C_0'$ and let $C_1'' = \otimes^m C_1'$. Then, by Fact 2.1.3 and the Direct Product Lemma,

$$\begin{aligned} \|C_0 - C_1\| < 1/3 &\Rightarrow \|C_0'' - C_1''\| < 1/3 \\ \|C_0 - C_1\| > 2/3 &\Rightarrow \|C_0'' - C_1''\| > 1 - 2\exp(-3^{\ell-1}(2/3)^{2\ell}/2) > 1 - 2e^{-k}. \end{aligned}$$

Finally, apply the transformation of Lemma 3.2.3 one more time to $(C_0'', C_1'', 1^k)$ to produce (D_0, D_1) such that

$$\begin{aligned} \|C_0 - C_1\| < 1/3 &\Rightarrow \|D_0 - D_1\| < 3^{-k} < 2^{-k} \\ \|C_0 - C_1\| > 2/3 &\Rightarrow \|D_0 - D_1\| > (1 - 2e^{-k})^k > 1 - 2ke^{-k} > 1 - 2^{-k}. \end{aligned}$$

■

Notice that the above analysis relies on the fact that $(2/3)^2 > (1/3)$, so it will not work if $1/3$ and $2/3$ are replaced by, say, .49 and .51. We do not know how to prove such a polarization lemma for arbitrary constant thresholds. We can however extend it to thresholds α and β, where $\beta^2 > \alpha$, and the running time will be polynomial in $\exp\left((1 - \log(\beta^2)/\log(\alpha))^{-1}\right)$ along with the input size. More precisely, we have:

Lemma 3.2.4 (General Polarization Lemma) *There is a function that takes as input a 5-tuple $(C_0, C_1, \alpha, \beta, 1^k)$, where $\beta^2 = \gamma \cdot \alpha$, with $\gamma > 1$, and C_0 and C_1 are circuits. The function is computable in time polynomial in the input size and $\alpha^{-1/\log(\gamma)}$, and outputs a pair of circuits (D_0, D_1) such that*

$$\begin{aligned} \|C_0 - C_1\| < \alpha &\Rightarrow \|D_0 - D_1\| < 2^{-k} \\ \|C_0 - C_1\| > \beta &\Rightarrow \|D_0 - D_1\| > 1 - 2^{-k} \end{aligned}$$

Proof: Let $\ell = \lceil \log_\gamma(6 \ln 6) \rceil$. Apply the XOR Lemma (Lemma 3.2.3) to the triple $(C_0, C_1, 1^\ell)$ to produce (C_0', C_1') such that if

$$\begin{aligned} \|C_0 - C_1\| < \alpha &\Rightarrow \|C_0' - C_1'\| < \alpha^\ell \\ \|C_0 - C_1\| > \beta &\Rightarrow \|C_0' - C_1'\| > \beta^\ell. \end{aligned}$$

Let $m = 1/3\alpha^\ell = \gamma^\ell/(3\beta^{2\ell})$. Then apply the direct product construction, letting $C_0'' = \otimes^m C_0'$ and let $C_1'' = \otimes^m C_1'$. Then, by Lemma 3.2.2,

$$\begin{aligned} \|C_0 - C_1\| < \alpha &\Rightarrow \|C_0'' - C_1''\| < 1/3 \\ \|C_0 - C_1\| > \beta &\Rightarrow \|C_0'' - C_1''\| > 1 - 2e^{-\frac{\gamma^\ell}{3\beta^{2\ell}} \cdot \frac{\beta^{2\ell}}{2}} \geq 2/3 \end{aligned}$$

An application of the standard Polarization Lemma finishes the proof. ■

3.2.3 A protocol for Statistical Difference

In this section, we show that SD has a simple two-message honest-verifier statistical zero-knowledge proof system, which is a generalization of the standard protocols for for Quadratic Nonresiduosity [GMR89] and Graph Nonisomorphism [GMW91]. Intuitively, if two distributions are statistically far apart, then, when given a random sample from one of the distributions, a computationally unbounded party should have a good chance of guessing which distribution it came from. However, if the two distributions are statistically very close, even a computationally unbounded party should not have much better than a 50% chance of guessing correctly. This suggests the following two-message (private-coin) protocol for SD:

Zero-knowledge proof system for SD

1. V, P: Compute $(D_0, D_1) = \textbf{Polarize}(C_0, C_1, 1^n)$, where $n = |(C_0, C_1)|$.
2. V: Flip one random coin $r \in \{0, 1\}$. Let z be a sample of D_r. Send z to P.
3. P: If $\Pr[D_0 = z] > \Pr[D_1 = z]$, answer 0, otherwise answer 1.
4. V: Accept if P's answer equals r, reject otherwise.

We establish the following lemma.

Lemma 3.2.5 *The above is an honest-verifier statistical zero-knowledge proof system for* SD, *with soundness error* $\frac{1}{2} + 2^{-n}$*, and completeness error and simulator deviation both* 2^{-n}*. Thus* SD $\in$ HVSZK.

Proof: We will argue that the prover strategy in the protocol we give is optimal — *i.e.* it maximizes the verifier's acceptance probability — and use this to bound both the soundness and completeness error. The simulator deviation will then follow easily.

Consider any prover P^*. Suppose for some z the prover P^* fails to follow the strategy we present. If $\Pr[D_0 = z] \neq \Pr[D_1 = z]$, this means that with nonzero probability, P^* choses the distribution in which z is less likely to occur. Then, conditioned on z, the success probability of P^* will certainly be lower than that of the prover in our protocol. If $\Pr[D_0 = z] = \Pr[D_1 = z]$, the prover has no information about r, so no matter what strategy it uses, it has exactly even odds on guessing correctly. Since these observations hold for all z, our prover is optimal.

We now analyze the probability of success of the optimal prover. Recall that $\|D_0 - D_1\| = \Pr[D_0 \in S] - \Pr[D_1 \in S]$ for $S = \{z : \Pr[D_0 = z] > \Pr[D_1 = z]\}$. The probability that the optimal prover guesses correctly is exactly

$$\begin{aligned} \frac{1}{2}\Pr[D_0 \in S] + \frac{1}{2}\Pr[D_1 \notin S] &= \frac{1}{2}(\Pr[D_0 \in S] + 1 - \Pr[D_1 \in S]) \\ &= \frac{1 + \|D_0 - D_1\|}{2}. \end{aligned}$$

By Lemma 3.2.1, $\|D_0 - D_1\| > 1 - 2^{-n}$ when (C_0, C_1) is a YES instance of SD, and $\|D_0 - D_1\| < 2^{-n}$ when (C_0, C_1) is a NO instance. Hence, the probability that the prover convinces the verifier to accept is greater than $(1 + 1 - 2^{-n})/2 > 1 - 2^{-n}$ for YES instances, and less than $(1 + 2^{-n})/2 < 1/2 + 2^{-n}$ for NO instances. This immediately gives the completeness error; the soundness error also follows because we considered the optimal prover strategy.

Now, notice that when the prover answers correctly, all the verifier receives from the prover is the value of r, which the verifier already knew. Thus, since we have shown that the prover is answering correcty with all but exponentially small probability, the verifier should be learning nothing. To turn this intuition into a proof of statistical zero-knowledge, we consider the following probabilistic polynomial-time simulator: On input (C_0, C_1), the simulator first computes $(D_0, D_1) = \textbf{Polarize}(C_0, C_1, 1^n)$, where $n = |(C_0, C_1)|$. The simulator then flips one random coin $r \in \{0,1\}$. If $r = 0$, it samples z from D_0, otherwise it samples z from D_1. The simulator then outputs a conversation in which the virtual verifier sends z to the virtual prover, and the virtual prover responds with r. The simulator also outputs the random coins it used to generate r and z as the coins of the virtual verifier. Thus, the simulator presented here always outputs conversations in which the prover responds correctly. Except for the prover's response, all other components of the simulator's output distribution are distributed identically to the verifier's view of the real interaction. Hence, the simulator deviation is bounded by the probability that the prover responds incorrectly in the real interaction, which we have already argued is at most 2^{-n} in the case of YES instances. ■

Observe that by using a security parameter k rather than n in the call to **Polarize**, both the completeness error and simulator deviation can be reduced to 2^{-k}. Thus, even very short assertions about SD can be proven with with very high security. Contrast this with the original definition of HVSZK [GMR89], which only requires that the simulator deviation vanish as an *negligible* function of the *input length*. This property has obvious cryptographic significance, so we formulate it more precisely in Section 3.3.1.

3.2.4 HVSZK-hardness of SD

The other major lemma we prove to show that SD is complete for HVSZK is the following:

Lemma 3.2.6 *Suppose promise problem Π has a public coin statistical zero-knowledge proof system. Then there exist PPT's A and B and a negligible function α such that*

$$\begin{aligned} x \in \Pi_Y &\Rightarrow \|A(x) - B(x)\| \leq \alpha(|x|), \text{ and} \\ x \in \Pi_N &\Rightarrow \|A(x) - B(x)\| \geq 1 - 2^{-\Omega(|x|)}. \end{aligned}$$

We defer the proof of this lemma to Section 3.2.5. We first observe how this lemma gives a reduction to SD for problems with public-coin statistical zero-knowledge proofs.

Corollary 3.2.1 *Suppose promise problem Π has a public-coin statistical zero-knowledge proof system. Then Π reduces to $\overline{\text{SD}}$. (Equivalently, $\overline{\Pi}$ reduces to* SD.*)*

Proof: First apply Lemma 3.2.6 to produce A and B, with $p(|x|)$ being a polynomial bound on the running times of $A(x)$ and $B(x)$. Given a string x, we can, by standard techniques,[5] produce in polynomial time circuits C_0 and C_1 describing the computation of A and B, respectively, on x for $p(|x|)$ steps. The inputs to C_0 and C_1 are the first $p(|x|)$ bits on the random tapes of A and B and the outputs are the first $p(|x|)$ positions on the output tapes. Then $\|C_0 - C_1\| = \|A(x) - B(x)\|$, which is at most $\alpha(|x|) < 1/3$ if $x \in \Pi$ and at least $1 - 2^{-|x|} > 2/3$ if $x \notin \Pi$ (for all sufficiently long x). So $x \mapsto (C_0, C_1)$ is a reduction from Π to $\overline{\text{SD}}$ which works for all but finitely many x. ■

[5]See, for example, [Pap94, Thms. 8.1 and 8.2].

The final ingredient in the proof of Theorem 3.2.1 is a theorem of Okamoto [Oka96], which we state in terms of promise problems.[6]

Theorem 3.2.2 ([Oka96, Thm. 1]) *If a promise problem Π has a statistical zero-knowledge proof system, then Π has a public coin statistical zero-knowledge proof system.*

Now it will be easy to show that SD is complete for HVSZK.

Proof of Theorem 3.2.1: Lemma 3.2.5 tells us that SD $\in$ HVSZK, so we only need to show that every problem in HVSZK reduces to SD. Corollary 3.2.1 and Theorem 3.2.2 imply that every problem $\Pi \in$ HVSZK reduces to $\overline{\text{SD}}$. In particular, SD reduces to $\overline{\text{SD}}$, or, equivalently, $\overline{\text{SD}}$ reduces to SD. Composing reductions, it follows that every problem $\Pi \in$ HVSZK reduces to SD. ∎

3.2.5 Proof of Lemma 3.2.6

The constructions in this lemma and the statistical zero-knowledge proof system for STATISTICAL DIFFERENCE are carried out for the specific example of GRAPH ISOMORPHISM in Section 3.2.6.

Intuition. Recall that we wish to construct a pair of probabilistic polynomial-time machines A and B such that if $x \in \Pi_Y$, the distributions $A(x)$ and $B(x)$ are statistically very close, but when $x \in \Pi_N$, $A(x)$ and $B(x)$ are far apart. We are given that Π has a *public-coin* statistical zero-knowledge proof system. A natural place to search for such distributions is in the output of the simulator for this proof system. We think of the simulator as describing the moves of a *virtual prover* and a *virtual verifier*.[7]. We wish to find properties of the simulator's output that (1) distinguish the case $x \in \Pi_Y$ from $x \in \Pi_N$, and (2) are captured by the statistical difference between samplable distributions. In the case that $x \in \Pi_Y$, we have strong guarantees on the simulator's output. Namely, it outputs accepting conversations with high probability and its output distribution is statistically very close to the real interaction. When $x \in \Pi_N$, there are two cases. If the simulator outputs accepting conversations with low probability, this easily distinguishes it from the simulator output when $x \in \Pi_Y$. However, it is possible that the simulator will output accepting conversations with high probability even when $x \in \Pi_N$. This means that the virtual prover is doing quite well in fooling the virtual verifier. This naturally suggest a strategy for a real prover — imitate the virtual prover's behavior. Such a prover, called a *simulation-based prover,* was introduced by Fortnow [For89] and is a crucial construct in our proof. The soundness of the proof system tells us that the simulation-based prover cannot hope to convince the real verifier with high probability. There must be a reason for this discrepancy between the success rates of the virtual prover and the simulation-based prover. One possibility is that the virtual verifier's coins in the simulator's output are *far from uniform,* so that the simulation only captures a small fraction of possible verifier states. However, this is not the only difficulty. A second difficulty is that the responses of the virtual prover may depend on future coins of the virtual verifier, which is impossible in a real public-coin interaction. Note that this is equivalent to the virtual verifier's coins being *dependent on previous messages* of the virtual prover. We will show that these are the only two obstacles the simulation-based prover faces in trying to fool the verifier, and thus they must be present when $x \in \Pi_N$. In the case that $x \in \Pi_Y$, however, these

[6]Okamoto stated his result in terms of languages, but the proof readily extends to promise problems (cf., [GV99]).

[7]This terminology is taken from [AH91]. The cases we consider are quite similar to those analyzed in [For89, AH91] Because we focus on public coin proofs, many complications that other researchers faced do not arise. This allows us to make some new observations and reach a novel conclusion.

difficulties cannot arise since we are guaranteed that the simulator output distribution is very close to that of the real interation. If we could measure the extent to which these anomalies are present by the statistical difference between samplable distributions, we would achieve our objective. This is precisely what we do.

Notation. Let (P, V) be a public-coin interactive proof system for a promise problem Π which is statistically zero-knowledge against V (the honest verifier) and let S be a simulator for this proof system. Without loss of generality, it may be assumed that the interaction of P and V on input x always has $2r(|x|)$ exchanged messages, with V sending the first message and each message consisting of exactly $q(|x|)$ bits, for some polynomials q and r. Moreover, it may be assumed that S's output always consists of $2r(|x|)$ strings of length $q(|x|)$. The output of S and the conversation between P and V on input x will be written in the form $S(x) = (c_1, p_1, \ldots, c_r, p_r)_S$ and $(P, V)(x) = (c_1, p_1, \ldots, c_r, p_r)_{(P,V)}$, respectively, where $c_1, \ldots, c_r$ represent the messages (or equivalenty, the coins) of V, while $p_1, \ldots, p_r$ represent the prover messages, and $r = r(|x|)$. (Dependence on x will often be omitted in this manner for notational convenience.) We use notation such as $(c_i)_S$ for the random variable obtained by running S once and taking the c_i-component of its output. More generally, partial conversation transcripts will be written like $(c_1, p_1, c_2, p_2)_S$. We call a conversation transcript $(c_1, p_1, \ldots, c_r, p_r)$ which would make V accept (resp., reject) an *accepting conversation* (resp., *rejecting conversation*). We denote by $U(n)$ the uniform distribution on strings of length n.

The proof. In order to formalize the above intuition, a definition of the simulation-based prover needs to be given. This is the prover P^* that imitates the virtual prover, *i.e.* P^* does the following to compute its next message when the current conversation transcript is $(c_1, p_1, \ldots, c_i)$:

If $S(x)$ outputs conversations that begin with $(c_1, p_1, \ldots, c_i)$ with probability 0, then output $0^{q(|x|)}$.
Else output $y \in \{0, 1\}^{q(|x|)}$ with probability

$$p_y = \Pr[S(x) \text{ begins with } (c_1, p_1, \ldots, c_i, y) | \\ S(x) \text{ begins with } (c_1, p_1, \ldots, c_i)].$$

In order to analyze the success probability of P^*, we first compare the output of S to the actual conversations between P^* and V. Let ϵ_i be the statistical difference between $(c_1, p_1 \ldots, c_{i-1}, p_{i-1}, c_i)_S$ and $(c_1, p_1 \ldots, c_{i-1}, p_{i-1})_S \otimes U(q(|x|))$. Thus ϵ_i measures how far from uniform the virtual verifier's i-th set of coins are and how far from independent they are from what comes before. The following claim formalizes our intuition that P^* can do as well as the virtual prover, as long as the virtual verifier's coins are near-uniform and near-independent from what preceeds them.

Claim 3.2.1 $\|S(x) - (P^*, V)(x)\| \leq \sum_{i=0}^{r} \epsilon_i$.

Proof: Let $C_i^S = (c_1, p_1, \ldots, c_i)_S$ be the random variable of partial simulator transcripts ending with the i-th coins of the virtual verifier. Let $P_i^S = (c_1, p_1, \ldots, c_i, p_i)_S$ be the random variable of partial transcripts ending with the i-th virtual prover response. Similarly define C_i^* and P_i^* as partial conversation transcripts of (P^*, V). The aim is to show that at round k, the statistical error grows

Algorithm A		Algorithm B	
$A_0(x)$	Run $S(x)$ for $\lvert x\rvert$ repetitions. Output '1' if the majority are accepting conversations; otherwise output '0'.	$B_0(x)$	Output 1.
$A_i(x)$	Run $S(x)$ to output $(c_1, p_1, \ldots, c_i)_{S(x)}$.	$B_i(x)$	Run $S(x)$ and flip $q(\lvert x\rvert)$ coins to output $(c_1, p_1, \ldots, c_{i-1}, p_{i-1})_{S(x)} \otimes U(q(\lvert x\rvert))$.

Table 3.1: The components of A and B

by at most ϵ_k. Formally, it will be shown by induction on k that

$$\|P_k^S - P_k^*\| \leq \sum_{i=0}^{k} \epsilon_i$$

The case $k = 0$ is trivial. For general k, first note that since P^* gives a response chosen according to the same distribution as the virtual prover, adding these responses to the conversations cannot increase the statistical difference (by Fact 2.1.4). That is,

$$\|P_{k+1}^S - P_{k+1}^*\| = \|C_{k+1}^S - C_{k+1}^*\|$$

The idea now is to extract the parts of $\|C_{k+1}^S - C_{k+1}^*\|$ corresponding to ϵ_{k+1} and observe that what is left is simply the error from the previous round. Note that $C_{k+1}^* = P_k^* \otimes U(q(|x|))$, since the real verifier's coins are always uniform and independent from what came before.

Then, applying Fact 2.1.3 and the Triangle Inequality,

$$\begin{aligned}
&\|C_{k+1}^S - C_{k+1}^*\| \\
&\quad \leq \|C_{k+1}^S - P_k^S \otimes U(q(|x|))\| \\
&\qquad + \|P_k^S \otimes U(q(|x|)) - P_k^* \otimes U(q(|x|))\| \\
&\quad \leq \epsilon_{k+1} + \|P_k^S - P_k^*\| + \|U(q(|x|)) - U(q(|x|))\| \\
&\quad \leq \epsilon_{k+1} + \sum_{i=0}^{k} \epsilon_i.
\end{aligned}$$

This completes the induction. Since $P_r^S = S(x)$ and $P_r^* = (P^*, V)(x)$, the Claim is proved. ■

We are now ready to construct the distributions we seek. The two distributions A and B each consist of $r + 1$ components, described in Table 3.1. A is the algorithm whose output on input x is $(A_0(x), A_1(x), \ldots, A_r(x))$, all run independently, and B is the algorithm whose output is $(B_0(x), B_1(x), \ldots, B_r(x))$, all run independently.

Here, A_i is a sampling of a partial conversation transcript from S up to the virtual verifier's i-th set of coins, while B_i is a sampling of a partial conversation transcript from S up to the virtual prover's $(i-1)$-th response followed by $q(|x|)$ independent random bits. So, for $i \geq 1$, the statistical difference between A_i and B_i is ϵ_i.

We will show that the statistical difference between A and B is negligible if $x \in \Pi_Y$ and is noticeable if $x \in \Pi_N$. Amplifying this gap by repetition will give us Lemma 3.2.6.

Claim 3.2.2 *There exists a negligible function α such that if $x \in \Pi_Y$, then $\|A(x) - B(x)\| \leq \alpha(|x|)$.*

Proof: By Fact 2.1.3, the statistical difference between $A(x)$ and $B(x)$ is bounded above by the sum of the statistical differences between $A_i(x)$ and $B_i(x)$ over $i = 1, \ldots, r(|x|)$. First, let's examine A_0 and B_0. Since $S(x)$ outputs a conversation which makes V accept with probability at least $2/3 - \text{neg}(|x|)$, the Chernoff bound implies that $\Pr[A_0(x) = 1] = 1 - 2^{-\Omega(|x|)}$, so the statistical difference between A_0 and B_0 is negligible. In the real conversations of P and V, the verifier's coins are truly uniform and independent from prior rounds, so for $i \geq 1$, $\|A_i(x) - B_i(x)\|$ should essentially be bounded by the statistical difference between the simulator's output and the real interaction. This is in fact true, as we have:

$$\begin{aligned}\|A_i(x) - B_i(x)\| &\leq \|A_i(x) - (c_1, p_1, \ldots, c_i)_{(P,V)(x)}\| + \|(c_1, p_1, \ldots, c_i)_{(P,V)(x)} - B_i(x)\| \\ &\leq \|S(x) - (P, V)(x)\| + \|S(x) - (P, V)(x)\|.\end{aligned}$$

(The last inequality is by Fact 2.1.4.) Since the statistical difference between S and (P, V) is negligible, $\|A(x) - B(x)\|$ is bounded by the sum of polynomially many negligible functions and is therefore negligible itself. ■

Claim 3.2.3 *If $x \in \Pi_N$ then $\|A(x) - B(x)\| \geq 1/12r(|x|)$.*

Proof: By Fact 2.1.4, it suffices to show that for some i, $\epsilon_i = \|A_i(x) - B_i(x)\| > 1/12r(|x|)$. We deal with two cases depending on the probability that S outputs an accepting conversation.

Case 1: $\Pr[S(x) \text{ accepts}] \leq 5/12$. Then, by the Chernoff bound, $\Pr[A_0(x) = 1] \leq 2^{-\Omega(|x|)}$, so the statistical difference between $A_0(x)$ and $B_0(x)$ is at least $1 - 2^{-\Omega(|x|)} > 1/12r(|x|)$.

Case 2: $\Pr[S(x) \text{ accepts}] > 5/12$. Then, since $\Pr[(P^*, V)(x) \text{ accepts}]$ is at most $1/3$, we must have

$$\sum_{i=0}^{r} \epsilon_i \geq \|S(x) - (P^*, V)(x)\| > 1/12.$$

Thus, at least one ϵ_i must be greater than $1/12r(|x|)$. ■

Now consider the samplable distributions $\hat{A}(x) = \otimes^{s(|x|)} A(x)$ and $\hat{B}(x) = \otimes^{s(|x|)} B(x)$, where $s(n) = n \cdot r(n)^2$. If $x \in \Pi_Y$, $\left\|\hat{A}(x) - \hat{B}(x)\right\| \leq s(|x|)\,\|A(x) - B(x)\|$, which is still negligible. If $x \in \Pi_N$, then by the Direct Product Lemma (Lemma 3.2.2), $\left\|\hat{A}(x) - \hat{B}(x)\right\| \geq 1 - 2^{-\Omega(|x|)}$.

This completes the proof of Lemma 3.2.6. ■

3.2.6 Example for Graph Isomorphism

For illustrative purposes, here we explicitly describe what happens when the transformations of Lemma 3.2.6 and the protocol from Section 3.2.3 are applied to the well-known public-coin perfect zero-knowledge protocol for Graph Isomorphism [GMW91]. The protocol proceeds as follows on input (G_0, G_1).

1. P sends V a random isomorphic copy H of G_0.

2. V picks $b \in \{0, 1\}$ at random and sends it to P.

3. P sends V a random isomorphism π between G_b and H, if one exists.

4. V checks that $\pi G_b = H$.

The simulator S for this protocol does the following on input (G_0, G_1):

1. Pick random $b \in \{0,1\}$ and a random permutation π.

2. Output $(\pi G_b, b, \pi)$.

Notice that the conversations output by S always make V accept.

If the reduction to SD from the proof of Lemma 3.2.6 is applied to the above protocol, the following distributions are obtained:

$A_0(G_0, G_1)$: Always output 1.
$B_0(G_0, G_1)$: Always output 1.
$A_1(G_0, G_1)$: Output $(\pi G_b, b)$ for a random permutation π and $b \in \{0,1\}$ chosen at random.
$B_1(G_0, G_1)$: Output $(\pi G_b, c)$ for a random permutation π and b and c chosen uniformly and independently from $\{0,1\}$.

Thus, $\|A_0(x) - B_0(x)\|$ always equals 0. $\|A_1(x) - B_1(x)\|$ is easily seen to be 0 if $G_0 \cong G_1$, and $1/2$ otherwise. For the rest of this section, we ignore A_0 and B_0 since they are irrelevant.

If we now apply the protocol for Statistical Difference from Section 3.2.3 to the distributions A_1 and B_1 (without first applying the Polarization Lemma) we obtain the following protocol (P', V') for Graph Nonisomorphism:

1. V' picks a random bit $d \in \{0,1\}$. If $d = 0$, V' chooses a random bit $b \in \{0,1\}$ and a random permutation π and sends $(\pi G_b, b)$ to P'. If $d = 1$, V' chooses random bits $b, c \in \{0,1\}$ and a random permutation π and sends $(\pi G_b, c)$ to P'.

2. P' receives message (H, b) from V'. If H is isomorphic to G_b, then P' guesses 0, else P' guesses 1.

3. V' accepts if the P' guesses d correctly.

Now, if G_0 is not isomorphic to G_1, then P' will guess correctly with probability 3/4. However, if G_0 is isomorphic to G_1, then no prover can guess correctly with probability greater than 1/2. The above protocol is of the same spirit as the standard Graph Nonisomorphism protocol [GMW91]. In both cases, the verifier randomly permutes one of the graphs to obtain a graph H and in order for the prover to succeed with probability greater than 1/2, the prover needs to be able to tell which graph H came from.

3.3 Applications

We now use the Completeness Theorem to establish a number of results. These include almost all previously known results about HVSZK, along with several new results.

3.3.1 Efficient statistical zero-knowledge proofs

The proof system for STATISTICAL DIFFERENCE given in Section 3.2.3 has a number of desirable features. It is very efficient in terms of communication and interaction, and the simulator deviation can be made exponentially small in a security parameter (that can be varied independently of the input length). By the Completeness Theorem, it follows that every problem in HVSZK also has a proof system with these properties.

We begin by formalizing one of the properties of the SD proof system that was informally discussed in Section 3.2.3.

Definition 3.3.1 *An interactive protocol (P, V) is called a* security-parametrized statistical zero-knowledge proof system *for a promise problem Π if there exists a PPT simulator S, a negligible function $\alpha(k)$ (called the* simulator deviation*), and completeness and soundness errors $c(k)$ and $s(k)$ such that for all strings x and all $k \in \mathbb{N}$,*

1. *If $x \in \Pi_Y$, then* $\Pr\left[(P, V)(x, 1^k) = \texttt{accept}\right] \geq 1 - c(k)$.
2. *If $x \in \Pi_N$, then for all P^*,* $\Pr\left[(P^*, V)(x, 1^k) = \texttt{accept}\right] \leq s(k)$.
3. *If $x \in \Pi_Y$, then* $\left\|S(x, 1^k) - \mathrm{View}_{P,V}(x, 1^k)\right\| \leq \alpha(k)$.

As usual, we require that $c(k)$ and $s(k)$ are computable in time $\mathrm{poly}(k)$ *and* $1 - c(k) > s(k) + 1/\mathrm{poly}(k)$

We now describe the efficient proof systems inherited by all of HVSZK.

Corollary 3.3.1 *Every problem in* HVSZK *possesses a security-parametrized statistical zero-knowledge proof system with the following properties:*

1. *Simulator deviation 2^{-k}, completeness error 2^{-k}, and soundness error $1/2 + 2^{-k}$.*
2. *The prover and verifier exchange only 2 messages.*
3. *The prover sends only 1 bit to the verifier.*
4. *The prover is deterministic.*

Proof: Let Π be any promise problem in HVSZK. Let f be the reduction from Π to SD guaranteed by the Completeness Theorem. A protocol with the desired properties for Π can be obtained as follows: on input $(x, 1^k)$, execute the proof system for SD, given in Section 3.2.3, on input $f(x)$ and using k rather than n in the call to **Polarize**. ■

3.3.2 Closure properties

In this section, we prove several closure properties of HVSZK. The first, closure under reductions, is a direct consequence of the "security parametrization" property shown to hold for HVSZK in the previous section.

Corollary 3.3.2 HVSZK *is closed under (Karp) reductions. That is, if* $\Pi \in$ HVSZK *and* Γ *reduces to* Π, *then* $\Gamma \in$ HVSZK.

Proof: By Corollary 3.3.1, Π has a security-parameterized statistical zero-knowledge proof. A statistical zero-knowledge proof for Γ can be obtained as follows: On input x, the prover, verifier, and simulator run the security-parametrized proof for Π on input $(f(x), 1^{|x|})$, where f is the reduction from Γ to Π. ■

The security-parametrization property is essential in the above proof, because an arbitrary reduction f could potentially shrink string lengths dramatically, and we want the simulator deviation to be negligible as a function of $|x|$, not $|f(x)|$.

Next, we show how Okamoto's result that HVSZK is closed under complement follows immediately from our proof of Completeness Theorem.

Corollary 3.3.3 ([Oka96, Thm. 2]) HVSZK *is closed under complement, even for promise problems.*

Proof: Let Π be any problem in HVSZK. By Theorem 3.2.2 and Corollary 3.2.1, $\overline{\Pi}$ reduces to SD, which is in HVSZK. By Corollary 3.3.2, $\overline{\Pi} \in$ HVSZK. ■

Before moving on to additional closure properties, we deduce the upper bounds of Fortnow [For89] and Aiello and Håstad [AH91] on the complexity of HVSZK.

Corollary 3.3.4 ([For89, AH91]) HVSZK $\subset$ AM $\cap$ co-AM, *where* AM *denotes the class of problems possessing constant-round interactive proofs.*

Proof: Immediate from Corollaries 3.3.1 and 3.3.3. ■

Above, we have shown that HVSZK satisfies a computational closure property (Corollary 3.3.2) and a boolean closure property (Corollary 3.3.3). Now we will exhibit a stronger closure property, which can be viewed as both a computational one and a boolean one: given an arbitrary boolean formula whose atoms are statements about membership in *any* problem in HVSZK, one can efficiently construct a statistical zero-knowledge interactive proof for its validity. Note that such a property does not follow immediately from the fact that a class is closed under intersection, union, and complementation, because applying these more than a constant number of times could incur a superpolynomial cost in efficiency, while we ask that the construction can be done efficiently with respect to the size of the formula. The procedure for doing this is based on work by De Santis, Di Crescenzo, Persiano, and Yung [DDPY94]. They show how to construct statistical zero-knowledge proofs for all monotone boolean formulae whose atoms are statements about a random self-reducible language. Their zero-knowledge proofs are constructed by producing two distributions which are either disjoint or identical, depending on whether or not the formula is true. Hence, their construction can be viewed as a reduction to an extreme case of SD, in which the thresholds are 1 and 0.

Using the direct product construction, the XOR Lemma, and the Polarization Lemma, we generalize their result to monotone formulae whose atoms are statements about membership in Statistical Difference. Then, using the completeness of SD (Theorem 3.2.1) and closure under complement (Corollary 3.3.3), we deduce the result to general (*i.e.* not necessarily monotone) formulae and every promise problem in HVSZK.

We begin with some definitions describing precisely what kind of boolean closure properties we will achieve. (Later, we will see how it can also be interpreted as closure under a certain class of polynomial-time reductions.) In order to deal with instances of promise problems that violate the promise, we adapt standard definitions in a straightforward way.

Definition 3.3.2 *Let Π be any promise problem. We define a tuple of boolean variables $(b_1, \ldots, b_k) \in \{0,1\}^k$ to be* Π-valid *with respect to a tuple of strings $(x_1, \ldots, x_k)$ if for all i:*

- *$x_i \in \Pi_Y$ implies $b_i = 1$.*
- *$x_i \in \Pi_N$ implies $b_i = 0$.*
- *b_i is unrestricted if x_i violates the promise of Π.*

Definition 3.3.3 *Let Π be any promise problem. Then we define a new promise problem $\Phi(\Pi)$ whose instances are $(\phi, x_1, \ldots, x_k)$ where $k \geq 0$ and ϕ is a k-ary propositional formula.*

- *The YES instances of $\Phi(\Pi)$ are those for which $\phi(b_1, \ldots, b_k) = 1$ for all Π-valid settings of $(b_1, \ldots, b_k)$ with respect to $(x_1, \ldots, x_k)$.*
- *The NO instances of $\Phi(\Pi)$ are those for which $\phi(b_1, \ldots, b_k) = 0$ for all Π-valid settings of $(b_1, \ldots, b_k)$ with respect to $(x_1, \ldots, x_k)$.*

$\mathrm{Mon}(\Pi)$ *is defined analogously, except that only monotone ϕ are considered.*[8]

In [DDPY94], it is shown that $\mathrm{Mon}(L) \in$ HVSZK for any language L which is random self-reducible, whose complement is self-reducible, or whose complement has a noninteractive statistical zero-knowledge proof. They also give statistical zero-knowledge proofs for some simple statements involving a random-self-reducible language and its complement. Damgård and Cramer [DC96] extend these results by showing that $\mathrm{Mon}(L) \in$ HVSZK as long as L or its complement has a 3-round public-coin statistical zero-knowledge proof, and also treat a larger class of monotone functions.

Our result holds for all of HVSZK and for all boolean formulae, not just monotone ones:

Theorem 3.3.1 *For any promise problem $\Pi \in$ HVSZK, $\Phi(\Pi) \in$ HVSZK.*

This theorem can be generalized to work for all boolean formulae whose atoms are statements about membership in any finite set of languages in HVSZK, but we omit the notationally cumbersome formal statement since it is immediate from the completeness of STATISTICAL DIFFERENCE.

The main step in proving Theorem 3.3.1 is the following Lemma, which is based on the construction of [DDPY94] for Mon(GRAPH NONISOMORPHISM):

Lemma 3.3.1 Mon(SD) $\in$ HVSZK.

Proof: For intuition, consider two instances of statistical difference (C_0, C_1) and (D_0, D_1), both of which have statistical difference very close to 1 or very close to 0 (which can be achieved by the Polarization Lemma). Then $(C_0 \otimes D_0, C_1 \otimes D_1)$ will have statistical difference very close to 1 if either of the original statistical differences is very close to 1 and will have statistical difference very close to 0 otherwise. Thus, this Direct Product operation represents OR. Similarly, the XOR operation in Proposition 3.2.2 represents AND. To obtain Lemma 3.3.1, we will recursively apply these constructions, taking care to keep the running time polynomial.

Let $w = (\phi, (C_0^1, C_1^1), \ldots, (C_0^k, C_1^k))$ be an instance of Mon(SD) and let $n = |w|$. By applying the Polarization Lemma (Lemma 3.2.1), we can constuct in polynomial time pairs of circuits $(D_0^1, D_1^1), \ldots, (D_0^k, D_1^k)$ such that the statistical difference between D_0^i and D_1^i is greater than $1 - 2^{-n}$ if $(C_0^i, C_1^i) \in \mathrm{SD}_Y$ and is less than 2^{-n} if $(C_0^i, C_1^i) \in \mathrm{SD}_N$.

```
Sample($\psi$, b)
    If $\psi = v_i$, sample $z \leftarrow D_b^i$.
    If $\psi = \tau \vee \mu$,
        Sample $z_1 \leftarrow$ Sample($\tau$, b);
        Sample $z_2 \leftarrow$ Sample($\mu$, b);
        Let $z = (z_1, z_2)$.
    If $\psi = \tau \wedge \mu$,
        Choose $c, d \in_R \{0,1\}$ subject to $c \oplus d = b$;
        Sample $z_1 \leftarrow$ Sample($\tau$, c);
        Sample $z_2 \leftarrow$ Sample($\tau$, d);
        Let $z = (z_1, z_2)$.
    Output $z$.
```

Figure 3-1:

Consider the randomized recursive procedure Sample(ψ, b) in Figure 3.3.2 which takes a subformula $\psi(v_{i_1}, \dots, v_{i_k})$ of ϕ and a bit $b \in \{0,1\}$ as input.

Executing Sample(ϕ, b) for $b \in \{0,1\}$ takes time polynomial in n, because the number of recursive calls is equal to the number of subformulas of ϕ. For a subformula τ of ϕ, let

$$\text{Dif}(\tau) = ||\text{Sample}(\tau, 0) - \text{Sample}(\tau, 1)||.$$

Then we can prove the following about Dif:

Claim 3.3.1 *For every subformula* $\psi = \psi(v_{i_1}, \dots, v_{i_j})$ *of* ϕ, $\text{Dif}(\psi) > 1 - m2^{-n}$ *if*

$$\psi((C_0^{i_1}, C_1^{i_1}) \in \text{SD}_Y, \dots, (C_0^{i_j}, C_1^{i_j}) \in \text{SD}_Y)$$

is true and $\text{Dif}(\psi) < m2^{-n}$ *if it is false, where* $m = |\psi|$.

Proof: The proof of the claim is by induction on subformulae ψ of ϕ. It holds for atomic subformulae (*i.e.* the variables v_i) by the properties of the D_b^i's.

- Consider the case when $\psi = \tau \vee \mu$. If ψ is true (with the appropriate arguments), either τ or μ must be true. Without loss of generality, say τ is true. Then, by Fact 2.1.4 and induction,

$$\text{Dif}(\psi) \geq \text{Dif}(\tau) > 1 - |\tau|2^{-n} > 1 - |\psi|2^{-n}.$$

  If ψ is false, then both τ and μ are false. By Fact 2.1.3 and induction,

$$\text{Dif}(\psi) \leq \text{Dif}(\tau) + \text{Dif}(\mu) < |\tau|2^{-n} + |\mu|2^{-n} \leq |\psi|2^{-n}.$$

- Now consider the case when $\psi = \tau \wedge \mu$. By Proposition 3.2.2, $\text{Dif}(\psi) = \text{Dif}(\tau) \cdot \text{Dif}(\mu)$. If ψ is true, then, by induction,

$$\text{Dif}(\psi) \geq (1 - |\tau|2^{-n})(1 - |\mu|2^{-n}) > 1 - (|\tau| + |\mu|)2^{-n} \geq 1 - |\psi|2^{-n}.$$

[8]In [DDPY94], only monotone formulae are treated. What they call $\Phi(L)$ is what we call $\text{Mon}(L)$.

If ψ is false, then, without loss of generality, say τ is false. By induction,

$$\text{Dif}(\psi) \leq \text{Dif}(\tau) < |\tau|2^{-n} \leq |\psi|2^{-n}.$$

■

Now, let A and B be circuits describing the computations of Sample$(\phi, 0)$ and Sample$(\phi, 1)$, respectively, (which take the random bits each procedure uses as input). By the above claim, $\|A-B\| > 1-n2^{-n} > 2/3$ if ϕ is true with the appropriate arguments, and $\|A-B\| < n2^{-n} < 1/3$ if ϕ is false. In other words, the construction of A and B from w describes a many-one reduction from Mon(SD) to SD. This reduction can be computed in polynomial time because Sample runs in polynomial time. Thus, by Corollary 3.3.2, Mon(SD) $\in$ HVSZK. ■

Now it is straightforward to deduce Theorem 3.3.1.

Proof: Let Π be any promise problem in HVSZK. By closure under complement (Corollary 3.3.3) and the completeness of SD (Theorem 3.2.1), both Π and $\overline{\Pi}$ reduce to SD. Let f and g be these reductions, respectively. Now, let $(\phi, x_1, \ldots, x_k)$ be any instance of $\Phi(\Pi)$, where $\phi = \phi(v_1, \ldots, v_k)$. Use De Morgan's laws to propagate all negations of ϕ to its variables. Now replace all occurrences of the literal $\neg v_i$ with a new variable w_i. Let $\psi(v_1, \ldots, v_k, w_1, \ldots, w_k)$ be the resulting (monotone) formula. It is clear that

$$(\phi, x_1, \ldots, x_k) \mapsto (\psi, f(x_1), \ldots, f(x_k), g(x_1), \ldots, g(x_k))$$

is a reduction from $\Phi(\Pi)$ to Mon(SD). Since SD is closed under reductions (Corollary 3.3.2), Theorem 3.3.1 follows from Lemma 3.3.1. ■

Theorem 3.3.1 can be also viewed as demonstrating that HVSZK is closed under a type of polynomial-time reducibility, which is formalized by the following two definitions.

Definition 3.3.4 (truth-table reduction [LLS75]): *We say a promise problem* Π truth-table reduces *to a promise problem* Γ *if there exists a (deterministic) polynomial-time computable function* f*, which on input* x *produces a tuple* $(x_1, \ldots, x_k)$ *and a circuit* C*, such that*

1. *If* $x \in \Pi_Y$ *then for all* Π*-valid settings of* $(b_1, \ldots, b_k)$ *with respect to* $(x_1, \ldots, x_k)$*,* $C(b_1, \ldots, b_k) = 1$*, and*
2. *If* $x \in \Pi_N$ *then for all* Π*-valid settings of* $(b_1, \ldots, b_k)$ *with respect to* $(x_1, \ldots, x_k)$*,,* $C(b_1, \ldots, b_k) = 0$.

In other words, a truth-table reduction for promise problems is a non-adaptive Cook reduction which is allowed to make queries which violate the promise, but must be able to tolerate both yes and no answers in response to queries that violate the promise. We further consider the case where we restrict the complexity of computing the output of the reduction from the queries:

Definition 3.3.5 (NC^1 truth-table reductions): *A truth-table reduction* f *between promise problems is an* NC^1 truth-table reduction *if the circuit* C *produced by the reduction on input* x *has depth bounded by* $c_f \log |x|$*, where* c_f *is a constant independent of* x.

With these definitions, we can restate Theorem 3.3.1 as follows:

Corollary 3.3.5 HVSZK *is closed under* NC^1 *truth-table reductions.*

Proof: Any circuit of size s and depth d can be efficiently "unrolled" into a formula of size $2^d \cdot s$. Hence, an NC^1 truth-table reduction from Γ to Π gives rise to a Karp reduction from Γ to $\Phi(\Pi)$. Since HVSZK is closed under $\Phi(\cdot)$ and Karp reductions, it is also closed under NC^1 truth-table reductions. ∎

It would be interesting to prove that HVSZK is closed under general truth-table reductions (or, even better, Cook reductions), or give evidence that this is not the case.

3.3.3 Knowledge complexity

Knowledge complexity [GMR89, GP91] is a generalization of zero knowledge which attempts to quantify how much a verifier learns from an interactive proof. A number of different measures have been proposed to accomplish this, most of which are based on the intuition that a verifier gains at most k bits of "knowledge" from an interaction if it can simulate the interaction with at most k bits of "help". Below we give terse definitions of the variants we consider. All of the following definitions come from [GP91], except for the last which comes from [ABV95]. Let (P, V) be an interactive proof system for a promise problem Π. Then the knowledge complexity of (P, V) in various senses is defined as follows:

- **Hint sense:** We say that (P, V) has perfect (resp., statistical) knowledge complexity $k(n)$ in the *hint* sense if there exists a PPT simulator S and a hint function $h : \Pi_Y \to \{0,1\}^*$ such that for all $x \in \Pi_Y$, we have $|h(x)| = k(|x|)$ and $\|S(x, h(x)) - \mathrm{View}_{P,V}(x)\|$ is 0 (resp., is bounded by a negligible function of $|x|$.)

- **Strict oracle sense:** (P, V) is said to have perfect (resp., statistical) knowledge complexity $k(n)$ in the *strict oracle* sense if there exists a PPT oracle-machine S and an oracle $\mathcal{O}$ such that on every input $x \in \Pi_Y$, S queries $\mathcal{O}$ at most $k(|x|)$ times and $\left\|S^{\mathcal{O}}(x) - \mathrm{View}_{P,V}(x)\right\|$ is 0 (resp., is bounded by a negligible function of $|x|$.)

- **Average oracle sense:** (P, V) has perfect (resp., statistical) knowledge complexity $k(n)$ in the *average oracle* sense if there exists a PPT oracle-machine S and an oracle $\mathcal{O}$ such that for every input $x \in \Pi_Y$, the average number of queries S makes to $\mathcal{O}$ is at most $k(|x|)$ and $\left\|S^{\mathcal{O}}(x) - \mathrm{View}_{P,V}(x)\right\|$ is 0 (resp., is bounded by a negligible function of $|x|$.)

- **Entropy sense:** (P, V) has perfect (resp., statistical) knowledge complexity $k(n)$ in the *entropy* sense if there exists a PPT oracle-machine S, an oracle $\mathcal{O}$, and a PPT oracle-simulator A such that for all $x \in \Pi_Y$, $\mathrm{E}_R[\log P_x(R)^{-1}] \le k(|x|)$, where $P_x(R) = \Pr_\rho[A(x, R; \rho) = S^{\mathcal{O}}(x; R)]$ and $\left\|S^{\mathcal{O}}(x) - \mathrm{View}_{P,V}(x)\right\|$ is 0 (resp., is bounded by a negligible function of $|x|$). Here, the notation $M(y; r)$ denotes the output of PPT M on input y and random coins r,

We say that the *knowledge complexity* (in some specified sense) of a promise problem Π is $k(n)$ if there exists an interactive proof system (P, V) for Π achieving negligible error probablity in both the completeness and soundness conditions such that the knowledge complexity of (P, V) is $k(n)$. The class of problems possessing perfect knowledge complexity $k(n)$ in the hint, strict oracle, average oracle, and entropy senses are denoted by $\mathsf{PKC}_{\mathsf{hint}}$, $\mathsf{PKC}_{\mathsf{strict}}$, $\mathsf{PKC}_{\mathsf{avg}}$, and $\mathsf{PKC}_{\mathsf{ent}}$, respectively. Statistical knowledge complexity is denoted by SKC with the appropriate subscript.

Our first result about knowledge complexity is that the $\mathsf{SKC}_{\mathsf{hint}}$ hierarchy collapses by logarithmic additive factors. Previously, Goldreich and Petrank [GP91] have shown that $\mathsf{SKC}_{\mathsf{hint}}(\mathrm{poly}(n)) \subset$

AM and $\mathsf{SKC_{hint}}(O(\log(n))) \subset$ co-AM; the second of these results can be derived immediately from our result and Fortnow's theorem [For89] that HVSZK $\subset$ co-AM.

Theorem 3.3.2 *For any polynomially bounded function $k(n)$,*

$$\mathsf{SKC_{hint}}(k(n) + \log n) = \mathsf{SKC_{hint}}(k(n)).$$

For intuition, consider the case that $k(n) = 0$. Loosely speaking, if the verifier is given the hint along with the input (with the "promise" that the hint is correct), then the original proof system becomes statistical *zero* knowledge, so we can apply the results of the previous section. By the boolean closure properties established in Theorem 3.3.1, we can take the "union over all possible hints" (there are only polynomially many of them) without leaving HVSZK. The result is easily seen to be the original problem.

In order to turn this intuition into a proof, we first show that knowledge complexity in the hint sense can be characterized in terms of zero-knowledge promise problems, so that questions about the $\mathsf{SKC_{hint}}$ hierarchy are reduced to questions about statistical zero knowledge. This is equivalence is obtained by providing the hint along with the input and "promising" that the hint is correct.

Lemma 3.3.2 *Let $k(n)$ be any polynomially bounded function. Then $\Pi \in \mathsf{SKC_{hint}}(k(n))$ (resp., $\mathsf{PKC_{hint}}(k(n))$) iff there exists a promise problem $\Gamma \in$ HVSZK (resp., HVPZK) such that*

1. *$x \in \Pi_Y \Rightarrow$ there exists a such that $|a| = k(|x|)$ and $(x, a) \in \Gamma_Y$, and*

2. *$x \in \Pi_N \Rightarrow$ for all a, $(x, a) \in \Gamma_N$.*

Proof: We only give the proof for statistical knowledge complexity and zero knowledge; the perfect case is identical.

$\Rightarrow$ Let Π be a problem in $\mathsf{SKC_{hint}}(k(n))$ and let $h : \Pi_Y \to \{0,1\}^*$. be a hint function corresponding to an appropriate interactive proof system and simulator for Π. Consider the following promise problem Γ:

$$\begin{aligned} \Gamma_Y &= \{(x, h(x)) : x \in \Pi_Y\} \\ \Gamma_N &= \{(x, a) : x \in \Pi_N\} \end{aligned}$$

By using the protocol and simulator for Π, we see that $\Gamma \in$ HVSZK (the verifier and prover for Γ should ignore the second component, whereas the simulator uses it as a hint.) It is clear that Γ satisfies the other conditions of Lemma 3.3.2.

$\Leftarrow$ Let $\Gamma \in$ HVSZK be the promise problem satisfying the stated conditions. Let $h : \Pi_Y \to \{0,1\}^*$ be any function such that for all $x \in \Pi_Y$,

1. $|h(x)| = k(|x|)$,

2. $(x, h(x)) \in \Gamma_Y$.

(Such a function is guaranteed by Condition 1.) We now give a proof system for Π of knowledge complexity $k(n)$. On input x, the prover gives the verifier $h(x)$ in the first step, and then they execute the protocol for Γ on $(x, h(x))$. The completeness and soundness of this protocol follow from the properties of the Γ proof system. This proof system is easily seen to have knowledge complexity

$k(n)$ in the hint sense, using the zero-knowledge simulator for Γ with hint $h(x)$. ■

We now prove Theorem 3.3.2.

Proof: Let Π be a problem in $\mathsf{SKC}_{\mathsf{hint}}(k(n) + \log n)$ and let Γ be the promise problem guaranteed by Lemma 3.3.2. By Theorem 3.3.1, $\Phi(\Gamma) \in \mathsf{HVSZK}$. Now consider a different, but related promise problem Γ', defined by

$$\begin{aligned} \Gamma'_Y &= \{(x,a)) \colon \text{there exists } b \text{ such that } |b| = \log|x| \text{ and } (x,ab) \in \Gamma_Y\} \\ \Gamma'_N &= \{(x,a) \colon \text{for all } b,\ (x,ab) \in \Gamma_N\} = \{(x,a) \colon x \in \Pi_N\}. \end{aligned}$$

For any string x, let $m = \log|x|$, let $b_1, \ldots, b_n$ be all strings of length m, and let ϕ be the formula $\phi(v_1, \ldots, v_n) = \bigvee_i v_i$. The relationship between Γ and Γ' above implies that

$$(x,a) \mapsto (\phi, (x,ab_1), \ldots, (x,ab_n))$$

is a reduction from Γ' to $\Phi(\Gamma)$. By closure under reductions (Corollary 3.3.2), we conclude that $\Gamma' \in \mathsf{HVSZK}$.

Now, if $x \in \Pi_Y$, then there exists an a of length $k(|x|) + \log(|x|)$ such that $(x,a) \in \Gamma_Y$. Taking a' to be the first $k(|x|)$ bits of a, we see that there exists an a' of length $k(|x|)$ such that $(x,a') \in \Gamma'_Y$. Moreover, if $x \in \Pi_N$, then for all a, $(x,a) \in \Gamma'_N$. Thus, by Lemma 3.3.2, we conclude that $\Pi \in \mathsf{SKC}_{\mathsf{hint}}(k(n))$. ■

The next theorem establishes tighter bounds on the perfect knowledge complexity of HVSZK. Aiello, Bellare, and Venkatesan [ABV95] have previously demonstrated that every language in HVSZK has perfect knowledge complexity $n^{-\omega(1)}$ (resp., $1 + n^{-\omega(1)}$) in the entropy (resp. average oracle) sense. Our results improve on these bounds, although the results of [ABV95] also apply to cheating-verifier classes and ours do not. Goldreich, Ostrovsky, and Petrank [GOP98] show that HVSZK has logarithmic perfect knowledge complexity in the *oracle sense* (which we do not consider), so our results are incomparable to theirs. Our result for the strict oracle sense is the first that we know of.

Theorem 3.3.3 [9]

1. *For every polynomial-time computable* $m(n) = \omega(\log n)$, $\mathsf{HVSZK} \subset \mathsf{PKC}_{\mathsf{strict}}(m(n))$.
2. $\mathsf{HVSZK} \subset \mathsf{PKC}_{\mathsf{avg}}(1 + 2^{-n})$.
3. $\mathsf{HVSZK} = \mathsf{PKC}_{\mathsf{ent}}(2^{-n})$.

Corollary 3.3.1 tells us that every problem in HVSZK has a simple two-message proof system like the SD proof system of Section 3.2.3. Thus, in order to measure the perfect knowledge complexity of HVSZK and prove Theorem 3.3.3, it suffices to analyze this protocol. Intuitively, since the prover is only sending the verifier one bit and this bit is almost always a value the verifier knows, the knowledge complexity of this protocol should be extremely small. However, this argument does not suffice, because the knowledge complexity of a problem Π is determined only by proof systems for Π which achieve *negligible* error probability in both the completeness and soundness conditions. We can overcome this difficulty by performing $\omega(\log n)$ parallel repetitions.

[9]The $2^{-\Omega(n)}$ in these results can be improved to $2^{-\Omega(n^k)}$ for any constant k by amplifying with security parameter n^k instead of $n+1$ in Protocol π of Section 3.2.3.

Proof: Let Π be any problem in HVSZK and let (P, V) be the proof system for Π constructed in Corollary 3.3.1 (from the SD proof system of Section 3.2.3) with the security parameter set to $k = 4n$ (so the completeness error is 2^{-4n}). Let $m = m(n)$ be any function computable in time poly(n) such that $\omega(\log n) \leq m \leq n$. Consider the proof system (P', V') obtained by m parallel repetitions of (P, V); this has negligible completeness and soundness errors. We now analyze its perfect knowledge complexity in various senses:

1. The prover sends at most m bits to the verifier on inputs of length n, so the perfect knowledge complexity of this protocol in the strict oracle sense is bounded by m.

2. A perfect simulator for (P', V') can be obtained as follows: On input x of length n, the simulator runs $V(x)$ for m times independently and queries the oracle *once* to find out if any of these runs would result in an incorrect prover response. If the oracle replies yes, the simulator queries the oracle m more times to find out which runs would result in an incorrect response. The simulator then outputs the random coins used for running V and the appropriate prover responses.

   In each subprotocol, the prover gives an incorrect response with probability at most 2^{-4n}. Thus, the simulator has to query the oracle for more than one bit with probability at most $n2^{-4n}$. Thus, on average, the simulator queries the oracle for at most $1 + mn2^{-4n} < 1 + 2^{-n}$ bits.

3. Let S be the simulator for (P', V') which simply simulates V' and queries the oracle $\mathcal{O}$ for all prover responses. One possible oracle simulator would assume that the prover is correct in all subprotocols. Unfortunately, this gives $1/P_x(R) = \infty$ for some R and yields infinite knowledge complexity. Thus, we instead have our oracle simulator A assume that the prover is right in each subprotocol independently with probability $1 - \delta$, where $\delta = 2^{-2n}$. Thus, $P_x(R) = (1-\delta)^k \delta^{m-k}$, if R is a set of random coins for V' (equivalently S, since S mimics V') which would elicit a correct prover response in exactly k of the subprotocols. Let ϵ be the probability that the prover is incorrect in an individual subprotocol. Then, $\epsilon \leq \delta^2$, and we have

$$\begin{aligned}
\mathrm{E}_R\left[\log \frac{1}{P_x(R)}\right] &= \sum_{k=0}^{m} \binom{m}{k} \epsilon^{m-k}(1-\epsilon)^k \log\left(\frac{1}{(1-\delta)^k \delta^{m-k}}\right) \\
&= \left(\log \frac{1}{\delta^m}\right) \sum_{k=0}^{m} \binom{m}{k} \epsilon^{m-k}(1-\epsilon)^k \\
&\quad + \left(\log \frac{\delta}{1-\delta}\right) \sum_{k=0}^{m} \binom{m}{k} \epsilon^{m-k}(1-\epsilon)^k k \\
&= \log \frac{1}{\delta^m} + m(1-\epsilon)\left(\log \frac{\delta}{1-\delta}\right) \\
&= m\left(\log \frac{1}{1-\delta} + \epsilon \log \frac{1-\delta}{\delta}\right) \\
&\leq m\left(\log \frac{1}{1-\delta} + \delta^2 \log\left(\frac{1}{\delta}\right)\right) \\
&\leq 2m\delta < 2^{-n}
\end{aligned}$$

   for sufficiently large n.

The opposite inclusion follows from the result of [ABV95] that $\mathsf{PKC}_{\mathsf{ent}}(\mathrm{neg}(n)) \subset \mathsf{HVSZK}$ for any negligible function $\mathrm{neg}(n)$.

■

3.3.4 Reversing statistical difference

By the completeness of SD (Theorem 3.2.1) and HVSZK's closure under complement (Corollary 3.3.3), we see that $\overline{\mathrm{SD}}$ reduces to SD. This is equivalent to the following surprising result:

Proposition 3.3.1 (Reversal Mapping) *There is a polynomial-time computable function that maps pairs of circuits (C_0, C_1) to pairs of circuits (D_0, D_1) such that*

$$\begin{aligned} \|C_0 - C_1\| < 1/3 \quad &\Rightarrow \quad \|D_0 - D_1\| > 2/3 \\ \|C_0 - C_1\| > 2/3 \quad &\Rightarrow \quad \|D_0 - D_1\| < 1/3. \end{aligned}$$

That is, SD *reduces to* $\overline{\mathrm{SD}}$.

By extracting ideas used in the transformations of statistical zero-knowledge proofs given in [Oka96] and in the proof of the Completeness Theorem, we obtain the description of this transformation given below.

The Construction. Let (C_0, C_1) be any pair of circuits and let $n = |(C_0, C_1)|$. By the Polarization Lemma (Lemma 3.2.1), we can produce in polynomial time a pair of circuits (C_0', C_1') such that

$$\begin{aligned} \|C_0 - C_1\| < 1/3 \quad &\Rightarrow \quad \|C_0' - C_1'\| > 1 - 2^{-n} \\ \|C_0 - C_1\| > 2/3 \quad &\Rightarrow \quad \|C_0' - C_1'\| < 2^{-n} \end{aligned}$$

Let $q = \mathrm{poly}(n)$ be the number of input gates of C_0' and C_1' (w.l.o.g. we may assume they have the same number) and let $\ell = \mathrm{poly}(n)$ be the number of output gates. For notational convenience, let $R = \{0,1\}^q$ and $L = \{0,1\}^\ell$. Let $m = n^3q^2$ and define a new distribution $\vec{C}$: $\{0,1\}^m \times R^m \to L^m$ as follows:

$$\vec{C}(\vec{b}, \vec{r}) = (C_{b_1}'(r_1), \ldots, C_{b_m}'(r_m)).$$

We use the notation $\vec{z} \leftarrow \vec{C}$ to denote $\vec{z}$ chosen according to $\vec{C}$, *i.e.* select $\vec{b}$ and $\vec{r}$ uniformly and let $\vec{z} = \vec{C}(\vec{b}, \vec{r})$.

Let $\mathcal{H}$ be a 2-universal family of hash functions from $\{0,1\}^m \times R^m \times L^m$ to $T = \{0,1\}^{(q+1)m-2\Delta-n}$, where $\Delta = \sqrt{nmq^2} = m/n$. We can now describe the new distributions:

D_0: Let $(\vec{b},\vec{r}) \in_R \{0,1\}^m \times R^m$, $\vec{y} \leftarrow \vec{C}$, and $h \in_R \mathcal{H}$. Output $(\vec{C}(\vec{b},\vec{r}), \vec{b}, h, h(\vec{b},\vec{r},\vec{y}))$.
D_1: Let $(\vec{b},\vec{r}) \in_R \{0,1\}^m \times R^m$, $h \in_R \mathcal{H}$, and $t \in_R T$. Output $(\vec{C}(\vec{b},\vec{r}), \vec{b}, h, t)$.

The important things to note about these distributions are that $\vec{b}$ is part of the output, and that D_0 and D_1 only differ in the last component, where D_0 has the value of the hash function and D_1 has a truly random element of T. Also note that the size of T is chosen to be $|\{0,1\}^m \times R^m|/2^{2\Delta+n}$, which is essentially $|\{0,1\}^m \times R^m|$, scaled down by a "slackness" factor of $2^{2\Delta+n}$. The introduction of the sample $\vec{y}$ in D_0 may at first seem superfluous; we explain below.

Intuition. For intuition, consider the case that $\vec{C}$ is distributed uniformly over some domain; that is, for every $\vec{z} \in \mathrm{range}(\vec{C})$, the size of the preimage set $|\{(\vec{b}, \vec{r}) \colon \vec{C}(\vec{b}, \vec{r}) = \vec{z}\}|$ is the same value N. (It turns out that $\vec{C}$ is actually "close enough" to uniform for these arguments to work.) Then the range of $\vec{C}$ has size $2^{(q+1)m}/N$. So, in D_0, conditioned on a value for $\vec{C}(\vec{b}, \vec{r})$, the triple $(\vec{b}, \vec{r}, \vec{y})$ is selected uniformly from a set of size $2^{(q+1)m}$. Since this is much greater than $|T|$, the Leftover Hash Lemma of [ILL89] implies that conditioned on any value for the first component of D_0, the last two components $(h, h(\vec{b}, \vec{r}, \vec{y}))$ are distributed close to the uniform distribution on $\mathcal{H} \times T$, which is the distribution that D_1 has in its last two components.[10] Thus, if their second components were missing, D_0 and D_1 would be statistically close. Now, consider the case that $\|C'_0 - C'_1\| \approx 1$. Then $\vec{b}$ is essentially "determined" by $\vec{C}(\vec{b}, \vec{r})$. So the presence of $\vec{b}$ can be ignored, and the above argument says that D_0 and D_1 are statistically very close. Now, consider the case that $\|C'_0 - C'_1\| \approx 0$. Then $\vec{b}$ is essentially "unrestricted" by $\vec{C}(\vec{b}, \vec{r})$. Since there are 2^m choices for $\vec{b}$, conditioning on $\vec{b}$ in addition to $\vec{C}(\vec{b}, \vec{r})$, cuts the number of triples $(\vec{b}, \vec{r}, \vec{y})$ down from $2^{m(q+1)}$ to roughly $2^{m(q+1)}/2^m$. Since $2^{m(q+1)}/2^m$ is much smaller than $|T|$, $h(\vec{b}, \vec{r}, \vec{y})$ will cover only a small fraction of $|T|$ and thus will be far from uniform (conditioned on values for $\vec{C}(\vec{b}, \vec{r})$, $\vec{b}$, and h).

Proof of Proposition 3.3.1. First we will argue that $\vec{C}$ is close to uniform, so that we can apply arguments like those given above. This is the case because $\vec{C}$ is composed of many independent, identically distributed random variables. For $\vec{z} \in L^m$, we say the weight of $\vec{z}$ is the logarithm of the size of the preimage set of $\vec{z}$. Formally, let $\mathrm{wt}(\vec{z}) = \log_2 |\{(\vec{b}, \vec{r}) \colon \vec{C}(\vec{b}, \vec{r}) = \vec{z}\}|$. Let w be the expected weight of an image, $w = \mathrm{E}_{\vec{z} \leftarrow \vec{C}}[\mathrm{wt}(\vec{z})]$. Then we can show the following:

Lemma 3.3.3 $\Pr_{\vec{z} \leftarrow \vec{C}}[|\mathrm{wt}(\vec{z}) - w| > \Delta] < 2^{-\Omega(n)}$.

Proof: For $z \in L$, let $\mathrm{wt}_0(z) = \log_2 |\{(b, r) \colon C_b(r) = z\}|$. Then, for $\vec{z} \in L^m$, $\mathrm{wt}(\vec{z}) = \mathrm{wt}_0(z_1) + \cdots + \mathrm{wt}_0(z_m)$, where $\vec{z} = (z_1, \ldots, z_m)$. Observe that when $\vec{z}$ is selected according to $\vec{C}$, $z_1, \ldots, z_m$ are independent and identically distributed. Moreover, for any $z \in L$, $0 \le \mathrm{wt}_0(z) \le q$. So, by the Hoeffding inequality [Hof95, Sec. 7.2.1], we have

$$\Pr_{\vec{z} \leftarrow \vec{C}}[|\mathrm{wt}(\vec{z}) - w| > \Delta] < 2e^{-2\Delta^2/mq^2} = 2e^{-2n}.$$

■

It will be convenient to eliminate those $\vec{z} \in L^m$ that have weight far above or below the mean. Let $G = \{(\vec{b}, \vec{r}) \colon |\mathrm{wt}(\vec{C}(\vec{b}, \vec{r})) - w| \le \Delta\}$ be the set of *good* pairs $(\vec{b}, \vec{r})$. The above Lemma says that $|G| \ge (1 - 2^{-\Omega(n)})|\{0,1\}^m \times R^m|$. Thus $\|G - \{0,1\}^m \times R^m\| \le 2^{-\Omega(n)}$, where for simplicity of notation, we let the name of a set also refer to the uniform distribution on the same set. Define $\vec{C}'$ to be the distribution obtained by selecting $(\vec{b}, \vec{r}) \leftarrow G$ and outputting $\vec{C}(\vec{b}, \vec{r})$. Then, since $\vec{C}$ is a function, Fact 2.1.4 tells us that $\|\vec{C} - \vec{C}'\| = 2^{-\Omega(n)}$. Similarly, we define variants of D_0 and D_1 that sample from G instead of $\{0,1\}^m \times R^m$:

[10] Here we see the importance of $\vec{y}$: Without $\vec{y}$, conditioned on some value of $\vec{C}(\vec{b}, \vec{r})$, the pair $(\vec{b}, \vec{r})$ would be selected uniformly from a space of size N. If we were only hashing this pair, for the distribution $h(\vec{b}, \vec{r})$ to be uniform by the Leftover Hash Lemma, T would have had to be chosen so that $|T| \ll N$. The value of N, however, depends on the inner workings of the circuit C, and is in general unknown. By including $\vec{y}$, which comes uniformly from a space of size $2^{(q+1)m}/N$, we balance the arguments to h so that they come from a space of size $2^{(q+1)m}$, a known quantity. This use of "dummy" samples to form a space whose size is known is the "complementary usage of messages" technique of Okamoto [Oka96].

D_0': Let $(\vec{b},\vec{r})\in_R G$, $\vec{y} \leftarrow \vec{C}'$, and $h\in_R\mathcal{H}$. Output $(\vec{C}'(\vec{b},\vec{r}),\vec{b},h,h(\vec{b},\vec{r},\vec{y}))$.

D_1': Let $(\vec{b},\vec{r})\in_R G$, $h\in_R\mathcal{H}$, and $t\in_R T$. Output $(\vec{C}'(\vec{b},\vec{r}),\vec{b},h,t)$.

Since D_0' (or D_1') is a randomized procedure applied to two (or one) independent samplings from G, Fact 2.1.4 tells us that $\|D_0 - D_0'\| = 2^{-\Omega(n)}$ (and $\|D_1 - D_1'\| = 2^{-\Omega(n)}$). Hence, it suffices to prove that these modified distributions have the properties we want in each case. For the case when C_0 and C_1 are statistically far, we prove the following claim:

Claim 3.3.2 *If* $\|C_0' - C_1'\| > 1 - 2^{-n}$, *then* $\|D_0' - D_1'\| < 2^{-\Omega(n)}$.

Proof: First we formalize the idea that $\vec{b}$ is "determined" by $\vec{C}$. Define $f\colon L \to \{0,1\}$ by

$$f(z) = \begin{cases} 0 & \text{if } \Pr[C_0' = z] > \Pr[C_1' = z] \\ 1 & \text{otherwise} \end{cases}$$

Then

$$\Pr_{b,r}[f(C_b'(r)) = b] = \frac{1}{2}\Pr_r[f(C_0'(r)) = 0] + \frac{1}{2}\Pr_r[f(C_1'(r)) = 1].$$

Now, by the definition of statistical difference, $\Pr_r[C_0'(r) \in f^{-1}(0)] \geq 1 - 2^{-n}$ and $\Pr_r[C_1'(r) \in f^{-1}(1)] \geq 1 - 2^{-n}$. Thus, $\Pr_{b,r}[f(C_b'(r)) = b] > 1 - 2^{-n}$. Now define $\vec{f}\colon L^m \to \{0,1\}^m$ by $\vec{f}(\vec{z}) = (f(z_1),\ldots,f(z_m))$. Then

$$\Pr_{\vec{b},\vec{r}}[\vec{f}(\vec{C}(\vec{b},\vec{r})) = \vec{b}] > (1-2^{-n})^m = 1 - 2^{-\Omega(n)}.$$

Since G is a $1 - 2^{-\Omega(n)}$ fraction of $\{0,1\}^m \times R^m$, the same is true when $(\vec{b},\vec{r})$ is selected uniformly from G. Thus, if we define:

D_0'': Let $(\vec{b},\vec{r})\in_R G$, $\vec{y} \leftarrow \vec{C}'$, and $h\in_R\mathcal{H}$. Output $(\vec{C}'(\vec{b},\vec{r}),\vec{f}(\vec{C}'(\vec{b},\vec{r})),h,h(\vec{b},\vec{r},\vec{y}))$.

D_1'': Let $(\vec{b},\vec{r})\in_R G$, $h\in_R\mathcal{H}$, and $t\in_R T$. Output $(\vec{C}'(\vec{b},\vec{r}),\vec{f}(\vec{C}'(\vec{b},\vec{r})),h,t)$.

Then, by Fact 2.1.5, $\|D_0' - D_0''\| = 2^{-\Omega(n)}$ and $\|D_1' - D_1''\| = 2^{-\Omega(n)}$. So it suffices to show that $\|D_0'' - D_1''\| = 2^{-\Omega(n)}$. Since the first components of D_0'' and D_1'' are identically distributed and the second components are determined by the first ones, it suffices to show (by Fact 2.1.5) that, conditioned on any value for the first coordinate, the third and fourth components have statistical difference $2^{-\Omega(n)}$. This will follow from the Leftover Hash Lemma [ILL89]:

Lemma 3.3.4 (Leftover Hash Lemma [ILL89]) *Let $\mathcal{H}$ be a family of 2-universal hash functions from D to T. Let X by a probability distribution on D such that for all $x \in D$,* $\Pr[X = x] \leq \epsilon/|T|$. *Then the following two distributions have statistical difference at most $\epsilon^{1/3}$.*

1. *Choose $x \leftarrow X$, $h\in_R\mathcal{H}$. Output $(h,h(x))$.*

2. *Choose $h\in_R\mathcal{H}$, $t\in_R T$. Output (h,t).*

By the above argument and the Leftover Hash Lemma, it suffices to show that conditioned on any value $\vec{z}$ for $\vec{C}'(\vec{b},\vec{r})$, no triple $(\vec{b},\vec{r},\vec{y})$ has probability more than $2^{-\Omega(n)}/|T|$. The pair $(\vec{b},\vec{r})$ comes uniformly from a set of size $2^{\mathrm{wt}(\vec{z})} \geq 2^{w-\Delta}$, and $\vec{y}$ is selected independently according to $\vec{C}'$, so the probability of any triple $(\vec{b},\vec{r},\vec{y})$ is at most

$$\left(\frac{1}{2^{w-\Delta}}\right)\left(\frac{2^{w+\Delta}}{|G|}\right) \leq \frac{2^{2\Delta}}{(1-2^{-\Omega(n)})2^{(q+1)m}} = \frac{2^{-\Omega(n)}}{|T|}.$$

Thus, $\|D_0'' - D_1''\| \leq 2^{-\Omega(n)}$, and the claim is established. ■

Now we treat the other case, when C_0 and C_1 are statistically close.

Claim 3.3.3 *If $\|C_0' - C_1'\| < 2^{-n}$, then $\|D_0' - D_1'\| > 1 - 2^{-\Omega(n)}$.*

Proof: First, we formalize the idea that $\vec{b}$ is almost completely "undetermined" by $\vec{C}(\vec{b},\vec{r})$. Since $\|C_0' - C_1'\| < 2^{-n}$, it follows from Fact 2.1.6 that with probability $1 - 2^{-\Omega(n)}$ over $z \leftarrow C_0'$,

$$(1-2^{-\Omega(n)})\Pr\left[C_1' = z\right] \leq \Pr\left[C_0' = z\right] \leq (1+2^{-\Omega(n)})\Pr\left[C_1' = z\right].$$

In other words,

$$1 - 2^{-\Omega(n)} \leq \frac{|\{r\colon C_0'(r) = z\}|}{|\{r\colon C_1'(r) = z\}|} \leq 1 + 2^{-\Omega(n)}.$$

The same is true with probability $1 - 2^{-\Omega(n)}$ when the roles of C_0' and C_1' are reversed. Thus, with probability $1 - m2^{-\Omega(n)} = 1 - 2^{-\Omega(n)}$ over $\vec{z} \leftarrow \vec{C}$, we have for *every* pair $\vec{b},\vec{c} \in \{0,1\}^m$,

$$1 - 2^{-\Omega(n)} = (1 - 2^{\Omega(n)})^m \leq \frac{\left|\{\vec{r}\colon \vec{C}(\vec{b},\vec{r}) = \vec{z}\}\right|}{\left|\{\vec{r}\colon \vec{C}(\vec{c},\vec{r}) = \vec{z}\}\right|} \leq (1 + 2^{-\Omega(n)})^m = 1 + 2^{-\Omega(n)}.$$

Since there are 2^m choices for $\vec{c}$, this, combined with Lemma 3.3.3, implies that, with probability $1 - 2^{-\Omega(n)}$ over $\vec{z} \leftarrow \vec{C}$, the following holds for *every* $\vec{b} \in \{0,1\}^m$:

$$\left|\{\vec{r}\colon \vec{C}(\vec{b},\vec{r}) = \vec{z}\}\right| \leq (1 + 2^{-\Omega(n)}) \cdot \frac{2^{\mathrm{wt}(\vec{z})}}{2^m} \leq (1 + 2^{-\Omega(n)}) \cdot 2^{w+\Delta-m}.$$

Since this is true with probability $1 - 2^{-\Omega(n)}$ for $\vec{z}$ selected according to $\vec{C}$, it is also true with probability $1 - 2^{-\Omega(n)}$ for $\vec{z}$ selected according to $\vec{C}'$. Fix any such $\vec{z}$ and fix any $\vec{b} \in \{0,1\}^m$ and $h \in \mathcal{H}$. Then, in D_0', conditioned on $\vec{C}'(\vec{b},\vec{r}) = \vec{z}$, $\vec{b}$, and h, there are at most

$$\begin{aligned}
(1 + 2^{-\Omega(n)}) \cdot 2^{w+\Delta-m}\left(\frac{|G|}{2^{w-\Delta}}\right) &\leq (1 + 2^{-\Omega(n)}) \cdot 2^{2\Delta-m}(2^{m+mq}) \\
&= (1 + 2^{-\Omega(n)}) \cdot 2^{4\Delta+n-m}|T| \\
&= 2^{-\Omega(m)}|T|
\end{aligned}$$

possible values for $(\vec{r},\vec{y})$. Thus, with probability $1 - 2^{-\Omega(n)}$, conditioned on values for the first three components of D_0', the fourth component $h(\vec{b},\vec{r},\vec{y})$ can cover at most a $2^{-\Omega(m)} \leq 2^{-\Omega(n)}$ fraction of T. In contrast, conditioned on values for the first three components of D_1', the fourth component is uniformly distributed on T. Therefore, $\|D_0' - D_1'\| \geq 1 - 2^{-\Omega(n)}$. ■

3.3.5 Weak-HVSZK and expected polynomial-time simulators

Recall that, in this thesis, we define statistical zero-knowledge with respect to *strict* polynomial-time simulators. As noted in Chapter 2, the original definition of statistical zero-knowledge permits *expected* polynomial-time simulators, but only allowing strict polynomial-time simulators is not very restrictive when discussing honest-verifier proofs, as we are.

However, our techniques do say something about expected polynomial-time simulators, and in particular show that expected polynomial-time simulators are no more powerful than strict ones for public-coin statistical zero-knowledge. This is the first general equivalence between strict and expected polynomial-time simulators for statistical zero knowledge that we know of.

Indeed, we are able to generalize further to an even weaker notion, which we call *weak* statistical zero knowledge:

Definition 3.3.6 *An interactive proof system (P, V) for a promise problem Π is* weak honest-verifier statistical zero knowledge *if for all polynomials p, there exists an efficient probabilistic (strict) polynomial-time algorithm S_p such that*

$$\|S_c(x) - (P, V)(x)\| \leq 1/p(|x|),$$

for all sufficiently long $x \in \Pi_Y$.

We denote by weak-HVSZK the class of promise problems admitting weak statistical zero-knowledge proofs, and by public-coin weak-HVSZK the class corresponding to public-coin such proofs. Note that any proof system admitting an expected polynomial-time simulator (in the usual sense) certainly also satisfies the requirements of weak statistical zero-knowledge. We show that in fact any public-coin weak statistical zero-knowledge proof system can be transformed into a statistical zero-knowledge proof system with a strict polynomial-time simulator achieving negligible (in fact, exponentially small) simulator deviation. In other words, public-coin weak-HVSZK = HVSZK.

Proposition 3.3.2 public-coin weak-HVSZK = HVSZK = public-coin HVSZK.

The only obstacle in generalizing Proposition 3.3.2 to all weak statistical zero-knowledge proofs (instead of just public-coin ones) is that Okamoto's private to public-coin transformation in [Oka96] is only given for strict polynomial-time simulators achieving negligible simulator deviation. In fact, this generalization was accomplished in work (subsequent to ours) by Goldreich and Vadhan [GV99].

In order to establish Proposition 3.3.2, it suffices to show that every problem in public-coin weak-HVSZK reduces to SD, as the proposition follows by closure under reductions (Corollary 3.3.2) and Okamoto's theorem that HVSZK = public-coin HVSZK (Theorem 3.2.2). Therefore, we need only establish the following generalization of Lemma 3.2.6:

Lemma 3.3.5 *Suppose promise problem Π has a public-coin weak statistical zero-knowledge proof. Then there exist probabilistic (strict) polynomial time machines A and B such that*

$$x \in \Pi_Y \quad \Rightarrow \quad \|A(x) - B(x)\| \leq \frac{1}{3}, \textit{ and}$$
$$x \in \Pi_N \quad \Rightarrow \quad \|A(x) - B(x)\| \geq \frac{2}{3}.$$

Proof: The proof is identical to the proof of Lemma 3.2.6, except that wherever the simulator S is used in that proof, we replace it with S_p, where the simulator deviation $1/p(n) = 1/(8n(12r(n))^2 \cdot r(n))$ (recall n will be $|x|$). Then we replace Claim 3.2.2 with the following:

Claim 3.3.4 *If $x \in \Pi_Y$, then $\|A(x) - B(x)\| \leq 1/3(n(12r(n))^2)$.*

Proof: The proof is identical to the proof of Claim 3.2.2, except that now, we have that

$$\|A_i(x) - B_i(x)\| \leq 2\|S_c(x) - (P,V)(x)\| \leq 1/(4n(12r(n))^2 \cdot r(n)).$$

As in the proof of Claim 3.2.2, it is still the case that $A_0(x)$ outputs 1 with probability $1 - 2^{-\Omega(n)}$. Thus, we have that

$$\|A(x) - B(x)\| \leq 1/4(n(12r(n))^2) + 2^{-\Omega(n)} \leq 1/3(n(12r(n))^2).$$

■

On the other hand, Claim 3.2.3 remains true, and so we have that $x \in \Pi_N$ implies $\|A(x) - B(x)\| \geq 1/12r(n)$.

Then, as in the original proof, we consider the samplable distributions $\hat{A}(x) = \otimes^{s(|x|)} A(x)$ and $\hat{B}(x) = \otimes^{s(|x|)} B(x)$, where $s(n) = n(12r(n))^2$. If $x \in \Pi_Y$, $\left\|\hat{A}(x) - \hat{B}(x)\right\| \leq s(|x|)\,\|A(x) - B(x)\| \leq 1/3$, as desired. If $x \in \Pi_N$, then by the Direct Product Lemma (Lemma 3.2.2), $\left\|\hat{A}(x) - \hat{B}(x)\right\| \geq 1 - 2^{-\Omega(|x|)}$. ■

3.3.6 Perfect and computational zero-knowledge

Although the focus of this chapter is (honest-verifier) statistical zero-knowledge, some of the techniques also apply to (honest-verifier) perfect and computational zero knowledge. In particular, for public-coin proof systems we obtain variants of Lemma 3.2.6 for both perfect and computational zero knowledge. In addition, a restricted version of STATISTICAL DIFFERENCE can be shown to have an (honest-verifier) perfect zero-knowledge proof.

First, we define some variants of SD. For any two constants α and β with $\alpha > \beta$, define:

$$\begin{aligned} \mathrm{SD}^{\alpha,\beta}_Y &= \{(C_0, C_1) : \|C_0 - C_1\| \geq \alpha\} \\ \mathrm{SD}^{\alpha,\beta}_N &= \{(C_0, C_1) : \|C_0 - C_1\| \leq \beta\} \end{aligned}$$

We can almost show that every problem which has a public-coin perfect zero-knowledge proof reduces to $\overline{\mathrm{SD}^{1/2,0}}$. The caveats are that either the original proof system must have perfect completeness, or we obtain distributions that are samplable in *expected* polynomial time rather than circuits.

Proposition 3.3.3 *Every promise problem having a public-coin perfect zero-knowledge proof with perfect completeness reduces to the complement of* $\mathrm{SD}^{1/2,0}$.

Proof: It suffices to show that the distributions $A(x)$ and $B(x)$ constructed in the proof of Lemma 3.2.6 have statistical difference 0 on YES instances, when the original proof system has perfect completeness and the simulator deviation is 0. Indeed, for $i \geq 1$, the distributions $A_i(x)$ and $B_i(x)$ are identical if the simulator deviation is 0, and the distributions $A_0(x)$ and $B_0(x)$ are identical under the additional assumption that the proof system has perfect completeness. ■

Proposition 3.3.4 *Suppose promise problem* Π *has a public-coin perfect zero-knowledge proof. Then there exist probabilistic* expected *polynomial time machines* A *and* B *such that*

$$\begin{aligned} x \in \Pi_Y &\Rightarrow \|A(x) - B(x)\| = 0, \text{ and} \\ x \in \Pi_N &\Rightarrow \|A(x) - B(x)\| \geq 1 - 2^{-\Omega(|x|)}. \end{aligned}$$

Proof: The proof is nearly identical to that of Proposition 3.3.3, except that we modify $A_0(x)$ and $B_0(x)$ to have statistical difference 0 (at the price of $B_0(x)$ becoming expected polynomial time). Let $c(n)$ be a polynomial bound on the number of random coins S uses on inputs of length n. Then we define A_0 and B_0 as follows (in both descriptions, $n = |x|$):

$A_0(x)$: Run $S(x)$ for $nc(n)$ repetitions. Output '1' if the majority are accepting conversations and '0' otherwise.

$B_0(x)$: With probability $1 - 2^{-c(n)}$, output '1'. Otherwise, calculate the probability σ that $S(x)$ outputs an accepting conversation (by exhaustive search over all $2^{c(n)}$ random seeds). Now calculate

$$\tau = \sum_{i=0}^{\lceil \frac{nc(n)}{2} \rceil} \binom{nc(n)}{i} \sigma^i (1-\sigma)^{nc(n)-i}.$$

If $\tau > 2^{-c(n)}$, output '1.' Otherwise, output '0' with probability $\tau/2^{-c(n)}$, and '1' otherwise.

Note that $B_0(x)$ runs in expected polynomial time, since with probability $2^{-c(n)}$ it runs in time $\mathrm{poly}(n)2^{c(n)}$ and otherwise it runs in time $\mathrm{poly}(n)$. Also observe that τ is the probability that $A_0(x)$ outputs '0'.

Now we argue that, when $x \in \Pi_Y$, $A_0(x)$ and $B_0(x)$ have statistical difference 0, i.e. output '1' with the same probability. Since $S(x)$ outputs a conversation which makes V accept with probability at least $2/3 - \mathrm{neg}(|x|)$, the Chernoff bound implies that $\Pr[A_0(x) = 1] = 1 - 2^{-\Omega(|x|c(|x|))}$. This means that τ will always be less than $2^{-c(|x|)}$ (for sufficiently large $|x|$), so B_0 will output '0' with probability $2^{-c(|x|)}(\tau/2^{-c(|x|)}) = \tau$, which is the probability that A_0 outputs '0'. ■

Now, if we could show that $\mathrm{SD}^{1/2,0}$ (or its complement) has a perfect zero-knowledge proof system, we would have something like a completeness result for HVPZK. Although we do not know how to do this, we can instead show that $\mathrm{SD}^{1,1/2} \in$ HVPZK. Indeed, consider the protocol of Section 3.2.3 with the modification that the two parties use the XOR Lemma (Lemma 3.2.3) instead of the Polarization Lemma. Then the proof of Lemma 3.2.5 tells us that this protocol, when used for $\mathrm{SD}^{1,1/2}$ has completeness error 0, simulator deviation 0, and soundness error $1/2 + 2^{-n}$. Thus we have:

Proposition 3.3.5 $\mathrm{SD}^{1,1/2} \in$ HVPZK.

For *computational* zero knowledge, the techniques of Lemma 3.2.6 give us the following:

Proposition 3.3.6 *Suppose promise problem* Π *has a public-coin computational zero-knowledge proof. Then there exist probabilistic polynomial time machines* A *and* B *such that*

1. $x \in \Pi_N \Rightarrow \|A(x) - B(x)\| \geq 1 - 2^{-\Omega(|x|)}$*, and*

2. $\{A(x)\}_{x\in\Pi_Y}$ *and* $\{B(x)\}_{x\in\Pi_Y}$ *are* computationally indistinguishable *ensembles of probability distributions.*

Note that, in contrast to perfect and statistical zero knowledge, we do not obtain a way of distinguishing YES and NO instances; it is possible for $A(x)$ and $B(x)$ to have statistical difference greater than $1 - 2^{-\Omega(|x|)}$ even for $x \in \Pi_Y$. We also remark that Proposition 3.3.6 holds even when the simulator for the proof system runs in expected polynomial-time, except that A and B will also run in expected polynomial-time.

Proof: The proof follows Lemma 3.2.6 exactly, except for Claim 3.2.2, which should be replaced with the following:

Claim 3.3.5 $\{A(x)\}_{x\in\Pi_Y}$ *and* $\{B(x)\}_{x\in\Pi_Y}$ *are computationally indistinguishable ensembles of probability distributions.*

The proof in Claim 3.2.2 that $A_0(x)$ and $B_0(x)$ have exponentially small statistical difference for $x \in \Pi_Y$ still holds. Thus it suffices to show that the ensembles $\{A'(x)\}_{x\in\Pi_Y}$ and $\{B'(x)\}_{x\in\Pi_Y}$ obtained by removing the 0'th components of $A(x)$ and $B(x)$, respectively, are computationally indistinguishable. To prove this, we first note that a hybrid argument shows that the ensembles $\{\otimes^{r(|x|)}(P,V)(x)\}_{x\in\Pi_Y}$ and $\{\otimes^{r(|x|)}S(x)\}_{x\in\Pi_Y}$ are computationally indistinguishable, since $\{(P,V)(x)\}_{x\in\Pi_Y}$ and $\{S(x)\}_{x\in\Pi_Y}$ are computationally indistinguishable.

Now we introduce a new ensemble $\{C(x)\}_{x\in\Pi_Y}$. Define $C_i(x) = (c_1, p_1, \ldots, c_i)_{(P,V)(x)}$ for $x \in \Pi_Y, 1 \le i \le r(|x|)$, and let $C(x) = C_1(x)\otimes\cdots\otimes C_r(x)$. Then $\{C(x)\}_{x\in\Pi_Y}$ and $\{A'(x)\}_{x\in\Pi_Y}$ are computationally indistinguishable since a distinguisher D between them could be used to make a distinguisher D' between $\{\otimes^r(P,V)(x)\}_{x\in\Pi_Y}$ and $\{\otimes^r S(x)\}_{x\in\Pi_Y}$: Given a sequence of $r(|x|)$ transcripts $(t_1,\ldots,t_r)$, D' truncates $t_i = (c_1,p_1,\ldots,c_r,p_r)$ to produce $t_i' = (c_1,p_1,\ldots,c_i)$ and feeds $(t_1',\ldots,t_r')$ to D. When fed with $\otimes^r S(x)$, D' gives D a sample of $A'(x)$, and when fed with $\otimes^r(P,V)(x)$, D' gives D a sample of $C(x)$.

Similarly, $\{C(x)\}_{x\in\Pi_Y}$ and $\{B'(x)\}_{x\in\Pi_Y}$ are also computationally indistinguishable because a distinguisher between them could be to make a distinguisher D' between $\{\otimes^r(P,V)(x)\}_{x\in\Pi_Y}$ and $\{\otimes^r S(x)\}_{x\in\Pi_Y}$: Given a sequence of $r(|x|)$ transcripts $(t_1,\ldots,t_r)$, D' truncates $t_i = (c_1,p_1,\ldots,c_r,p_r)$ and selects u_i according to the uniform distribution on strings of length $r(|x|)$ to produce $t_i' = (c_1,p_1,\ldots,p_{i-1},u)$ and feeds $(t_1',\ldots,t_r')$ to D. When fed with $\otimes^r S(x)$, D' gives D a sample of $B'(x)$, and when fed with $\otimes^r(P,V)(x)$, D' gives D a sample of $C(x)$.

Now, because both $\{A'(x)\}_{x\in\Pi_Y}$ and $\{B'(x)\}_{x\in\Pi_Y}$ are computationally indistinguishable from $\{C(x)\}_{x\in\Pi_Y}$, they must be computationally indistinguishable from each other, completing the proof. ■

3.3.7 Hard-on-average languages and one-way-functions

In this section, we show how to use techniques from this chapter to provide an alternative simpler proof of a known result relating zero-knowledge proofs for hard-on-average problems and one-way functions.

At the heart of most cryptographic constructions lie problems that we believe to be hard in the average case (e.g. QUADRATIC RESIDUOSITY [GM84]). However, not every such problem has proved useful in general cryptographic constructions. Surprisingly, Ostrovsky [Ost91] showed that statisitical zero knowledge provides one way of identifying the cryptographic usefulness of a problem in general applications: He proved that if any hard-on-average problem admits a statistical zero-knowledge proof, then one-way functions exist (which implies the existence of various cryptographic primitives). Later, Ostrovsky and Wigderson [OW93] generalized this result to computational zero knowledge as well.

We show how to use the techniques from this chapter to provide an alternative simpler proof of the theorem of Ostrovsky, and a restriction of the theorem of Ostrovsky and Wigderson where we consider only *public-coin* computational zero-knowledge proofs. We will give a proof only for the case of public-coin HVCZK. Note that this implies the same results for HVSZK, since HVSZK = public-coin HVSZK ⊂ HVCZK. We extend the results of [Ost91, OW93] to promise problems. We start with a definition of a hard-on-average problem:

Definition 3.3.7 *We say a promise problem* Π *is* hard-on-average *if there exists a PPT algorithm* S *such that for all positive integers* n *and all PPT algorithms* A*, there exists a negligible function* α *such that:*

$$\Pr_{x \leftarrow S(1^n)}[A(x) = \chi(x)] \leq \frac{1}{2} + \alpha(n)$$

where χ *is the characteristic function for the promise problem* Π *and we may assume the strings output by* $S(1^n)$ *are of length at least* n*.*

A key ingredient in our proof, also used in the proofs of [Ost91, OW93], will be the result of [Gol90] showing that the existence of a pair of efficiently samplable ensembles of distributions that are computationally indistinguishable but statistically far apart is equivalent to the existence of pseudorandom generators, and hence one-way functions. We first recall this result:

Theorem 3.3.4 ([Gol90]) *If there are two ensembles of efficiently samplable distributions* $\{X_n\}$ *and* $\{Y_n\}$ *such that:*

1. $\{X_n\}$ *and* $\{Y_n\}$ *are computationally indistinguishable.*
2. *There exists a polynomial* p *such that for all positive integers* n*, we have* $\|A(1^n) - B(1^n)\| \geq 1/p(n)$.

Then one-way functions exist.

We now establish:

Theorem 3.3.5 *If* public-coin HVCZK *contains a hard-on-average promise problem* Π*, then one-way functions exist.*

Proof: Suppose Π is hard-on-average with respect to the sampler S. We apply Proposition 3.3.6 to Π to construct distributions A and B such that if $x \in \Pi_N$, then $A(x)$ and $B(x)$ are statistically far apart, but for $x \in \Pi_Y$, we have that $A(x)$ and $B(x)$ are computationally indistinguishable. Now consider the distribution ensembles $X_n = (x, A(x))$ and $Y_n = (x, B(x))$ where x in both distributions is chosen according to $S(1^n)$.

Intuition. Note that because Π is hard-on-average with respect to S, for all n, the probability that the output of $S(1^n)$ is in Π_N must be negligibly different from $1/2$. Thus, the ensembles X_n and Y_n must have statistical difference at least $1/2 - \alpha(n)$ for some negligible function α. Thus, X_n and Y_n satisfy Condition 2 of Theorem 3.3.4. On the other hand, suppose there exists some distinguisher D that gets non-negligible advantage $\gamma(n)$ in distinguishing X_n and Y_n, *i.e.* $\Pr[D(x, A(x)) = 1] -$

$\Pr[D(x, B(x)) = 1] \geq \gamma(n)$. We know that $A(x)$ and $B(x)$ are computationally indistinguishable when $x \in \Pi_Y$, so D must be distinguishing $A(x)$ and $B(x)$ only when $x \in \Pi_N$. We can use this to decide $\chi(x)$, contradicting the assumption that Π is hard-on-average. This cannot be, and so the conditions of Theorem 3.3.4 must be satisfied, which implies the existence of one-way functions.

More formally, consider the following algorithm M for deciding Π:

1. Given x, compute $a = D(x, A(x))$ and $b = D(x, B(x))$.
2. If $a = 1$ and $b = 0$, output 0, by which we mean $x \in \Pi_N$.
3. If $a = 0$ and $b = 1$, output 1, by which we mean $x \in \Pi_Y$.
4. Otherwise, flip a fair coin to decide.

Let $p(x) = \Pr[D(x, A(x)) = 1]$, and $q(x) = \Pr[D(x, B(x)) = 1]$. Then note that for any given x, the probability that the algorithm decides that $x \in \Pi_N$ is exactly $p(x)(1 - q(x) + \frac{1}{2}(p(x)q(x) + (1 - p(x))(1 - q(x))) = \frac{1}{2} + (p(x) - q(x))/2$. Now, by Proposition 3.3.6, there exists a negligible function (which we can assume is also α) such that for all $x \in \Pi_Y$, we know that $|p(x) - q(x)| \leq \alpha(n)$.

Now, we consider

$$\begin{aligned}
\Pr_{x \leftarrow S(1^n)}[M(x) = \chi(x)] &= \Pr_{x \leftarrow S(1^n)}[x \in \Pi_Y] \cdot \Pr_{x \leftarrow S(1^n)}[M(x) = 1 \mid x \in \Pi_Y] + \\
&\quad \Pr_{x \leftarrow S(1^n)}[x \in \Pi_N] \cdot \Pr_{x \leftarrow S(1^n)}[M(x) = 0 \mid x \in \Pi_N] \\
&= \Pr_{x \leftarrow S(1^n)}[x \in \Pi_Y] \cdot \frac{1}{2}\left(1 - E_{x \leftarrow S(1^n)}[(p(x) - q(x)) \mid x \in \Pi_Y]\right) + \\
&\quad (1 - \Pr_{x \leftarrow S(1^n)}[x \in \Pi_Y]) \cdot \frac{1}{2}\left(1 + E_{x \leftarrow S(1^n)}[(p(x) - q(x)) \mid x \in \Pi_N]\right) \\
&\geq \frac{1}{2} - \alpha(n) + \frac{\gamma}{4}
\end{aligned}$$

where the last inequality follows since $E[p(x) - q(x)] = \gamma$, and yet for all $x \in \Pi_Y$, we have that $p(x) - q(x) \leq \alpha(n)$.

This contradicts the hardness assumption of Π. Hence, X_n and Y_n are computationally indistinguishable distribution ensembles that have non-negligible statistical distance, and hence by Theorem 3.3.4, one-way functions exist. ■

Chapter 4

Dealing with Cheating Verifiers in Statistical Zero-Knowledge Proofs

So far, all our results have dealt only with *honest-verifier* zero-knowledge proofs, where the guarantee that the verifier learns nothing only holds if the verifier follows the protocol exactly as specified. As we have seen, this notion already gives rise to interesting protocols and complexity-theoretic structure. From a cryptographic point of view, however, it is most often unrealistic to assume that parties will follow a protocol as specified. Indeed, one of the most important goals in the construction of a cryptographic protocol is to guarantee robustness against parties that deviate from the protocol – and it is precisely in guaranteeing this kind of security that zero-knowledge proofs have found their most compelling application [GMW91, GMW87]. Clearly, honest-verifier zero-knowledge proofs do not suffice for such applications.

This brings up the natural question: how much cheating can we tolerate from the verifier? In this chapter, we show that for statistical zero-knowledge proofs and public-coin computational zero-knowledge proofs, the answer is that we can tolerate *any* amount of cheating on behalf of the verifier. More precisely, we give an efficient transformation that converts any public-coin proof that is statistical zero knowledge only for the honest verifier into one that is zero knowledge for *any* verifier, no matter how much it deviates from the specified protocol. This transformation applies to all public-coin honest-verifier computational zero-knowledge proofs, as well. This transformation has two interesting consequences:

- Because Okamoto [Oka96] has given a transformation from honest-verifier statistical zero-knowledge proofs to public-coin such proofs, by composing this with our transformation, we obtain the result that $\mathsf{HVSZK} = \mathsf{SZK}$. Therefore, we can translate the results we obtained about HVSZK to hold for all of SZK.

- Our transformation also preserves the "power of the prover" – that is, if the prover of the original protocol can be implemented in probabilistic polynomial time, then the prover of the transformed protocol can be implemented in probabilistic polynomial time as well. Thus, our transformation also makes possible a useful design methodology for constructing cryptographically useful zero-knowledge proofs: First, construct a public-coin proof that is zero knowledge only for the honest verifier. Then, use our transformation to convert it into a proof that is zero knowledge against all verifiers.[1]

[1]Note that the transformation of Okamoto [Oka96] from private-coin to public-coin honest-verifier statistical zero knowledge does not have this feature, and therefore in order to use our transformation cryptographically, it is important to start with a public-coin protocol.

The work we present in this chapter is based on a paper [GSV98] authored jointly with Oded Goldreich and Salil Vadhan.

4.1 Overview

Recall that we defined a black-box any-verifier (statistical or computational) zero-knowledge proof system to be as follows:

Definition 4.1.1 *Let (P, V) be an interactive proof system (P, V) with negligible completeness and soundness errors, for a promise problem Π.*

- *(P, V) is said to be a* (general) black-box statistical zero-knowledge *proof system if there exists a probabilistic polynomial-time*[2] *(oracle machine) simulator S and a negligible function α (called the* simulator deviation*) such that for every non-uniform probabilistic polynomial-time verifier V^*, we have:*

$$\text{For all } x \in \Pi_Y, \text{ we have } \left\| S^{V^*}(x) - \mathrm{View}_{P,V^*}(x) \right\| \leq \alpha(|x|). \tag{4.1}$$

- *A* black-box computational zero-knowledge *proof system replaces Condition (4.1) with the requirement that the ensembles $\{S^{V^*}(x)\}_{x\in\Pi_Y}$ and $\{\mathrm{View}_{P,V^*}(x)\}_{x\in\Pi_Y}$ are computationally instinguishable ensembles of distributions.*

We let SZK *(resp.* CZK*) denote the class of promise problems with (general) black-box statistical (resp. computational) zero-knowledge proof systems.*

Recall that in this black-box formulation of zero knowledge, we can also define what it means to allow unlimited verifiers in the case of statistical zero-knowledge proofs. We can say that a proof system is black-box statistical zero knowledge *against arbitrary verifier strategies* if it satisfies Condition (4.1) for all possible verifier machines V^*, as opposed to only computationally bounded ones.

We first describe some previous work on transforming honest-verifier zero-knowledge proofs into general zero-knowledge proofs, and then describe the main result of this chapter.

Previous Transformations of Honest-Verifier to General Zero Knowledge

Conditional results for CZK: Assuming the existence of one-way functions, it is known that every problem with an interactive proof also has a (general) black-box public-coin computational zero-knowledge proof[GMW91, IY87, BGG$^+$88, HILL, Nao91]), which of course in particular implies HVCZK = CZK (in fact HVCZK = public-coin CZK).

Conditional results for SZK: The problem of relating HVSZK to SZK was first studied in [BMO90]. They showed that the two classes coincide, provided that the Discrete Logarithm Problem is hard. At the time, it seemed puzzling that computational assumptions can be used in the supposedly "information theoretic" context of statistical zero-knowledge. However, a careful examination reveals that the standard class SZK does refer to computational limitations: It is required to simulate only all probabilistic polynomial-time verifiers. The computational assumption

[2]As noted earlier, the definition of each type of zero knowledge can also be made allowing the simulator to run in expected polynomial time instead of strict polynomial time.

is thus used to restrict the behavior of cheating verifiers. This approach was carried to its climax in [Oka96] (cf., [DGOW95, Part 2]): Using any bit commitment scheme (and thus any one-way function [HILL, Nao91]) it was shown that public-coin HVSZK = public-coin SZK. Combined with the HVSZK = public-coin HVSZK result of Okamoto [Oka96], this implies that the existence of one-way functions implies HVSZK = SZK (and in fact HVSZK = public-coin SZK).

Unconditional results for constant-round zero-knowledge: The only unconditional transformations of honest-verifier SZK (resp., CZK) known before, referred to the class of *constant-round public-coin* proof systems (cf., [Dam94, DGW94]). It was shown that if Π has a HVSZK (resp., HVCZK) public-coin proof system of a constant number of rounds then $\Pi \in$ public-coin SZK (resp., $\Pi \in$ public-coin CZK).

Weak statistical zero-knowledge: In [DOY97] it is claimed that any language in HVSZK has an interactive proof, (P, V), with a *weak* statistical zero-knowledge property: For every positive polynomial p, and every non-uniform probabilistic polynomial-time verifier V^*, there exists a probabilistic polynomial-time simulator S^*_p (with running-time depending on p) so that the variation distance between the probability ensembles, $\{(P, V^*)(x) : x \in L\}$ and $\{S^*_p(x) : x \in L\}$, is at most $1/p(|x|)$.

4.1.1 Our results

We obtain the first unconditional general transformation of honest-verifier zero-knowledge to general zero-knowledge.

Theorem 4.1.1 (Transformation Theorem): *There exists an efficient transformation of honest-verifier statistical* (resp., computational) *zero-knowledge* public-coin *proof systems, into general black-box statistical* (resp., computational) *zero-knowledge* public-coin *proof systems. Furthermore,*

1. *The resulting proof systems has twice as many rounds as the original one.*
2. *The resulting prover strategy can be implemented in probabilistic polynomail-time given oracle access to the original prover strategy.*
3. *The completeness error of the resulting proof system is exponentially vanishing. In case the original proof system has perfect completeness, so does the resulting one.*
4. *The soundness error of the resulting proof system is bounded above by* $1/p(|x|)$, *where* p *is an arbitrary polynomial determined by the transformation.*
5. *In case of statistical zero-knowledge, the resulting proof system is black-box statistical zero knowledge against arbitrary verifier strategies, and its simulation error is at most* $\text{poly}(|x|) \cdot \epsilon(x) + 2^{-\Omega(|x|)}$, *where* $\epsilon(x)$ *is the simulation error of the original system.*

We obtain as immediate corollaries, our two main theorems:

Theorem 4.1.2 *Every promise problem having an honest-verifier computational zero-knowledge public-coin proof system, also has a general computational zero-knowledge public-coin proof system. And so,* public-coin HVCZK = public-coin CZK.

And, using Okamoto's result that HVSZK = public-coin HVSZK [Oka96, Thm. 1], we have:

Theorem 4.1.3 *Every language having an honest-verifier statistical zero-knowledge proof system, also has a general (public-coin) statistical zero-knowledge proof system. And so,* HVSZK = SZK.

Remark 4.1.1 Our transformation has several interesting features:

- **Arbitrary Verifier Strategies:** We stress that, in contrast to the previously mentioned conditional results, our result for *statistical* zero-knowledge is unconditional and guarantees (black-box) simulation of all possible verifier strategies (not only polynomial-time ones).

- **Computational Zero Knowledge:** Theorem 4.1.1 also provides a transformation for a wide class of *computational* zero-knowledge proof systems – that is, the class of public-coin proof systems. We view our result as a significant step towards showing that HVCZK = CZK without relying on any intractability assumptions.

- **Power of the prover:** The transformation of Theorem 4.1.1 preserves the power of the prover. In particular, if the prover strategy of the original public-coin honest-verifier proof system is implementable efficiently (*i.e.* in polynomial-time), then the prover strategy of the transformed protocol can also be implemented efficiently. Note, however, that the transformation of Okamoto from private-coin to public-coin honest-verifier statistical zero-knowledge proofs does *not* share this feature. Thus, the composed transformation of Theorem 4.1.3 does not preserve the power of the prover.

- **Soundness error and number of rounds:** The transformation of Theorem 4.1.1 increases the number of rounds of the original proof system only by a factor of 2. However, the resulting protocol has noticeable soundness error. That is, for any positive polynomial p, we can achieve a soundness error of $1/p(|x|)$. The soundness error may be further decreased, while preserving the zero-knowledge property, by *sequential* repetition of the proof system. In particular, to achieve negligible soundness error it suffices to use $\omega(1)$ sequential repetitions. This is unavoidable, unless one-way functions do not exist, since only BPP problems may have black-box simulation zero-knowledge *public-coin* proofs with constant number of rounds and negligible error probability [GK96].[3]

- **Completeness error:** By first applying the transformation of [FGM$^+$89], we may eliminate completeness error altogether (at the cost of at most one additional round and not preserving the complexity of the prover). (Recall that the transformation of [FGM$^+$89] increases the simulation error by at most an exponentially vanishing amount.)

- **Corollaries:** Many known results regarding the class HVSZK translate to the class SZK (and respectively results for public-coin HVCZK translate to public-coin CZK). For example, using known results regarding HVSZK, one obtains that SZK is closed under complement, has a complete promise problem, etc. We will elaborate on these in more detail in Section 4.8.

[3] Recall that if one-way functions exist then NP has constant-round public-coin proofs with negligible soundness error which are honest-verifier computational zero-knowledge [GMW91]. So, if Theorem 4.1.1 were to preserve all its features while resulting in a proof system with negligible soundness error then NP $\subseteq$ BPP would follow, which is impossible if one-way functions exist.

4.1.2 Techniques

Theorem 4.1.1 is proven by modifying the transformation presented in [DGW94]. Whereas the proof systems resulting from that transformation could be simulated only for a constant number of rounds, our modified transformation can be simulated for any (polynomial) number of rounds. Both transformations apply to honest-verifier Arthur-Merlin[4] zero-knowledge proofs (both statistical and computational).

The verifier (Arthur) is supposed to select its messages uniformly at random from the set $\{0,1\}^\ell$ of binary strings of a specified length. Thus, the simulator for the original proof system will also produce Arthur messages that come from a distribution that is indistinguishable (either in a statistical or computational sense) from uniform. When we remove the assumption that Arthur is honest, Arthur may now select its message strings in a manner which is not uniform, and thus the original simulator will not suffice.

To address this problem, in the transformation of [DGW94], each ℓ-bit long (random) message sent by Arthur is replaced by an invocation of a 3-round *Random Selection* protocol, for generating strings in $\{0,1\}^\ell$. In this protocol, Arthur and Merlin (the prover) interact to jointly produce a string which will be interpreted as "Arthur's message" in the original proof system. For any fixed positive polynomial p, a Random Selection protocol with the following two properties was presented [DGW94]:

1. As long as Arthur acts according to the protocol, Merlin may cause the outcome to deviate from uniform distribution over $\{0,1\}^\ell$ by at most $1/p(\ell)$. (That is, the statistical distance from the uniform distribution is at most $1/p(\ell)$.)

2. As long as Merlin plays according to the protocol, Arthur may not cause any ℓ-bit string to appear as the outcome with probability greater than $p(\ell)^4 \cdot 2^{-\ell}$. In particular, when Arthur applies any deterministic[5] cheating strategy, the outcome of the protocol is uniformly distributed over some set of at most $\frac{2^\ell}{p(\ell)^4}$ strings.

The proof system resulting from the above transformation is simulated in [DGW94] by running the honest-verifier simulator, and "hoping" that all Arthur-messages included in the transcript fall in the sets mentioned in Item (2) above. If this is the case, [DGW94] show that the simulation can be extended to fit the transformed proof system. Now, if the proof system uses only a constant number of invocations of the Random Selection protocol, then the "hope" will prevail with inverse polynomial probability. Thus, by repeating the experiment a polynomial number of times, we can be guaranteed success with extremely high confidence. This suffices for producing a black-box simulation with respect to any cheating Arthur-strategy. This approach fails when we have a non-constant number of rounds (Random Selection invocations), since in this case the "hope" will be dashed with all but negligible probability.

In our transformation, we modify the transformation of [DGW94] as follows: Rather than selecting a message, we use the [DGW94] Random Selection protocol to specify (in a succinct manner) a set of 2^n possible Arthur messages. Merlin then selects a message uniformly from this set. An immediate concern is that this allows Merlin to select a string which is advantageous for cheating. However, this only increases Merlin's cheating probability by a factor of 2^n per each round. Though

[4]Recall that public-coin proof systems are also called Arthur-Merlin proof systems, where the verifier is called Arthur and the prover is called Merlin. Since we want to stress the public-coin nature of many protocols in this chapter, we will often use the Arthur-Merlin naming convention when referring to the verifier and the prover.

[5]The restriction to deterministic strategies is without loss of generality here, as we will discuss briefly.

this may seem large, we can make the original proof system have an extremely tiny soundness error by parallel repetition, which effectively counteracts this threat.

The question now is what have we gained by making this change? Intuitively, we gain not having to simulate the Random Selection protocol for *any* possible outcome. Rather than having to simulate an execution which results in any specific ℓ-bit output, α, we only need to simulate an execution which results in a random set of strings containing α. The distinction is important since executions of the former type may exist only for a $1/\text{poly}(\ell)$ fraction of the possible α's, whereas – as we show – executions of the latter type exist and can be efficiently generated for all but a $2^{-\Omega(n)}$ fraction of the α's. Proving the last statement is a major technical undertaking of this chapter. It is reduced to proving the following lemma which may be of independent interest:

Lemma 4.1.1 (Hashing Lemma): *There exists a universal constant, $c > 0$, so that the following holds, for every $\epsilon, \delta > 0$. Let D and T be finite sets, H be a 2-universal family of hash functions from D to T, and $e \in T$. Let $S \subseteq H$ such that $|S| \geq \delta|H|$, and X be a random variable ranging over a finite set D having collision probability at most $\frac{\epsilon}{|T|}$* (i.e., $\sum_{x \in D} \Pr[X = x]^2 \leq \frac{\epsilon}{|T|}$). *Then the statistical difference between the following two random processes is at most $c \cdot \epsilon^{1/c}\delta^{-c}$.*

(A) *Select h uniformly in S, and let x be selected from X conditioned on $h(X) = e$. Output (h, x).*

(B) *Let $x \leftarrow X$, and h be selected uniformly among all $h \in H$ satisfying $h(x) = e$. Output (h, x).*

Actually, we need only establish a special case of this lemma, where X is uniform over D (and $|T| = \epsilon \cdot |D|$) for our proof of Theorem 4.1.1.

4.2 Notation

In this chapter, whenever we consider an interactive proof system, x will denote the common input and n will be the length of x. For notational convenience, we will often hide dependence on x or n when it is clear. For example, we write r instead of $r(n)$, when r is actually a function of n.

4.3 The starting proof system

Theorem 4.1.1 is proven by combining two transformations. The first transformation is obtained by simple parallel repetition, from which we immediately obtain the lemma below[6]. The protocols resulting from this transformation are the starting point for our main transformation, stated in the next section.

Lemma 4.3.1 *Let Π be a problem having a* honest-verifier *statistical (resp., computational) zero-knowledge* public-coin *proof systems of r rounds. Then Π has such a* (r-round honest-verifier) *zero-knowledge* (public-coin) *proof system in which*

1. *The prover strategy can be implemented in probabilistic polynomial-time given oracle access to the original prover stategy.*

2. *The completeness error is less than 2^{-n}, and in case the original proof system has perfect completeness so does the resulting one.*

[6] Recall that *honest-verifier* zero-knowledge properties are preserved under parallel repetition.

3. *Soundness error is less than* $2^{-n\cdot(r+1)}$.

4. *For* $\Pi \in$ HVSZK*: The simulator deviation is at most a polynomial factor greater than the original one.*

4.4 The transformation

Fix a problem Π in HVSZK or public-coin HVCZK and let (M, A) be the proof system guaranteed by Lemma 4.3.1. Let $r = r(n)$ be the number of rounds of (M, A) and let $\ell = \ell(n)$ be the length of A's messages. We assume an arbitrary polynomial $p(\cdot)$ has been chosen for the soundness error for the transformed proof system. Let $p = p(n)$. We may describe this proof system as follows:

Original Proof System (M, A), on input x:

1. In round $i \quad (i = 1, 2, \dots, r)$,
   (a) A chooses a message $\alpha_i \in_R \{0,1\}^\ell$ and sends it to M.
   (b) M sends a response $\beta_i \leftarrow M(\alpha_1, \beta_1, \dots, \alpha_{i-1}, \beta_{i-1}, \alpha_i)$ to A.
2. After round r, machine A deterministically decides whether to accept or reject.

The reason such a protocol could be zero-knowledge against the honest verifier but not against dishonest verifiers is that nothing prevents A from choosing the α_i's maliciously rather than uniformly. The idea of our transformation is to replace A's random choices with a Random Selection protocol (to be described in Section 4.5) which guarantees that the α_i's are statistically close to uniform, regardless of how A behaves. The new protocol, denoted $(\mathbf{M}, \mathbf{A})$, proceeds as follows.

Transformed Proof System $(\mathbf{M}, \mathbf{A})$, on input x:

1. In stage $i \quad (i = 1, 2, \dots, r)$,
   (a) $\mathbf{M}$ and $\mathbf{A}$ use the Random Selection protocol, $RS_{2pr,\ell}(n)$, to select $\alpha_i \in \{0,1\}^\ell$.
   (b) $\mathbf{M}$ sends the response $\beta_i \leftarrow M(\alpha_1, \beta_1, \dots, \alpha_{i-1}, \beta_{i-1}, \alpha_i)$ to $\mathbf{A}$.
2. After stage r, machine $\mathbf{A}$ accepts or rejects as A would on transcript $(\alpha_1, \beta_1, \dots, \alpha_r, \beta_r)$.

We will prove the following about the Transformed Proof System:

Lemma 4.4.1 *The Transformed Proof System* $(\mathbf{M}, \mathbf{A})$ *has the following properties:*

1. *The number of rounds is twice the number of rounds in* (M, A).
2. $\mathbf{M}$ *can be implemented in probabilistic polynomial time given oracle access to* M.
3. *The completeness error is exponentially vanishing. In case* (M, A) *has perfect completeness, so does* $(\mathbf{M}, \mathbf{A})$.
4. *Soundness error* $1/p$.

5. *When (M, A) is honest-verifier statistical (resp., computational) zero-knowledge,* $(\mathbf{M}, \mathbf{A})$ *is statistical (resp., computational) zero-knowledge, and this zero-knowledge property is exhibited by a black-box simulator. Furthermore, for honest-verifier statistical zero-knowledge proofs, this simulation holds against arbitrary verifier strategies.*

6. *In the case of Statistical Zero-Knowledge, the simulator deviation is at most* $2^{-\Omega(n)}$ *greater than that of* (M, A).

Theorem 4.1.1 follows immediately from Lemmas 4.3.1 and 4.4.1. Lemma 4.4.1 depends crucially on properties of the Random Selection protocol. We devote the next section to describing the Random Selection protocol and its crucial properties. In Section 4.6, we show how to derive Lemma 4.4.1 from the properties of the Random Selection protocol.

4.5 Random Selection

Let q and ℓ be any polynomials. In this section, we describe an Arthur-Merlin protocol $RS_{q,\ell}(n) = (M_{RS}, A_{RS})(n)$ for randomly selecting a string in $\{0,1\}^{\ell(n)}$. The protocol employs the Random Selection protocol $DGW_{q,\ell}(n) = (M_{DGW}, A_{DGW})$ of [DGW94] as a subprotocol, and the following presentation is adapted from that paper.

For notational convenience, we will write q to mean $q(n)$ and ℓ to mean $\ell(n)$. Let $\mathcal{H}$ be the space of affine linear functions from $\{0,1\}^{\ell}$ to $\{0,1\}^{\ell-n}$, i.e. $h \in \mathcal{H}$ is of the form $h(x) = Ax+b$ for some appropriately sized matrix A and vector b. For $\alpha \in \{0,1\}^{\ell}$, we write $\mathcal{H}_{\alpha}$ for $\{h \in \mathcal{H}\colon h(\alpha) = 0\}$. Let $s = \ell \cdot (\ell - n) + (\ell - n)$ and $t = s - 4\log_2(3qs)$. Note that elements of $\{0,1\}^s$ can be viewed as elements from $\mathcal{H}$. The protocol $DGW_{q,\ell}$ utilizes a space of functions $\mathcal{F}$ from $\{0,1\}^s$ to $\{0,1\}^t$ satisfying the following properties:

1. Each $f \in \mathcal{F}$ has a description of size $\mathrm{poly}(n)$.

2. There is a $\mathrm{poly}(n)$-time algorithm that, on input $f \in \mathcal{F}$ and $h \in \{0,1\}^s$, outputs $f(h)$.

3. There is a $\mathrm{poly}(n)$-time algorithm that, on input $f \in \mathcal{F}$, $y \in \{0,1\}^t$, lists all the elements of $f^{-1}(y)$. In particular, $|f^{-1}(y)| \leq p(n)$ for some polynomial p.

4. For every $y \in \{0,1\}^s$ and $f \in \mathcal{F}$, $f^{-1}(y)$ is nonempty.

5. $\mathcal{F}$ is a family of almost s-wise independent hashing functions in the following sense: For every s distinct points $h_1, \ldots, h_s \in (\{0,1\}^s \setminus \{0,1\}^t 0^{s-t})$, for a uniformly chosen $f \in \mathcal{F}$, the random variables $f(h_1), \ldots, f(h_s)$ are independently and uniformly distributed in $\{0,1\}^t$. (This property is used only for the proof of the soundness condition of the protocol, found in [DGW94].)

Such a family can be constructed essentially by associating the domain $\{0,1\}^s$ with the field $GF(2^s)$, and considering all polynomials of degree $s - 1$ over this field. Corresponding to each such polynomial p, there is a member f of the family which on input $h \in \{0,1\}^s$ outputs the t most significant digits of $p(h)$, unless h is of the form $h = x0^{s-t}$, in which case $f(h) = x$. For more details and to see why this family satisfies the conditions above, see [DGW94]. We can view each $f \in \mathcal{F}$ as defining a partition of $\{0,1\}^s$ into 2^t cells of the form $f^{-1}(y)$, each of size $\mathrm{poly}(n)$. For notational convenience, we will sometimes write *cell* y to refer to the cell $f^{-1}(y)$.

We now describe the protocol of [DGW94]:

The DGW Random Selection Protocol $DGW_{q,\ell} = (M_{DGW}, A_{DGW})(n)$:

1. A_{DGW} selects $f \in_R \mathcal{F}$, and sends it to M_{DGW} (i.e., A_{DGW} selects a random partition).
2. M_{DGW} selects $y \in_R \{0,1\}^t$, and sends it to A_{DGW} (i.e., M_{DGW} uniformly selects a cell).
3. A_{DGW} selects $h \in_R f^{-1}(y)$ (i.e. A_{DGW} uniformly selects an element of the cell).
4. Output h.

If, at any step, A_{DGW} or M_{DGW} do not select an object from the appropriate set, whatever message they send is interpreted as a canonical element of that set. In [DGW94], it was shown that the above protocol has the following properties (roughly speaking):

1. (Soundness) For any Merlin strategy M^*_{DGW}, the output distribution on $\mathcal{H} = \{0,1\}^s$ of (M^*_{DGW}, A_{DGW}) deviates from uniform by at most $1/q$ (in statistical difference).
2. (Simulability) Let A^*_{DGW} be any strategy for Arthur. At least a $1/\mathrm{poly}(n)$ fraction of the h's in $\{0,1\}^s$ occur as possible outputs of the interaction (M_{DGW}, A^*_{DGW}) and given such an h, one can simulate in $\mathrm{poly}(n)$-time A^*_{DGW}'s vview of an interaction resulting in h.

The main hindrance in applying the protocol as used by [DGW94] is that the simulator is only guaranteed to work for a $1/\mathrm{poly}(n)$ fraction of the h's. In our protocol, we interpret the output $h \in \mathcal{H}$ of the DGW protocol as a set of strings (namely $h^{-1}(0)$), from which a single string α is randomly selected by Merlin. It is this α, rather than h, that is the output of the Random Selection protocol. Thus, we only need to simulate the Random Selection protocol for a random α rather than a random h. For a given α, there are exponentially many hash functions h such that $h(\alpha) = 0$. Because this space of h's is so large and covers the α's near-uniformly, we are able to perform the simulation for a $1 - 2^{-\Omega(n)}$ fraction of the α's.

A full description of our Random Selection protocol follows.

Our Random Selection Protocol $RS_{q,\ell} = (M_{RS}, A_{RS})(n)$:

1. A_{RS} selects $f \in_R \mathcal{F}$, and sends it to M_{RS} (i.e., A_{RS} selects a random partition).
2. M_{RS} selects $y \in_R \{0,1\}^t$, and sends it to A_{RS} (i.e., M_{RS} uniformly selects a cell).
3. A_{RS} selects $h \in_R f^{-1}(y)$ (i.e. A_{RS} uniformly selects an element of the cell).
4. M_{RS} selects $\alpha \in_R h^{-1}(0)$. (If $h^{-1}(0) = \emptyset$ then α is defined to be 0^ℓ.)
5. Output α.

As with the DGW protocol, if A_{RS} or M_{RS} do not select an object from the appropriate set at any step, whatever message they send is interpreted as a canonical element of that set. The properties of this protocol are described in the following Proposition.

Proposition 4.5.1 *For any polynomials q and ℓ, the Random Selection protocol $RS_{q,\ell}$ is a 4-round protocol with the following properties:*

1. *(Efficiency) Both M_{RS} and A_{RS} can be implemented in time* $\mathrm{poly}(n)$ *and the protocol is public-coin for both parties.*

2. *(Soundness) For all Merlin strategies M^*_{RS} and all sets $S \subset \{0,1\}^\ell$, the probability that the output of $(M^*_{RS}, A_{RS})(n)$ lies in S is at most*

$$2^n \cdot \frac{|S|}{2^\ell} + \frac{1}{q}.$$

Also, the statistical distance between the distribution on h from executions of $(M^*_{RS}, A_{RS})(n)$ and the uniform distribution over $\mathcal{H}$ is at most $1/q$.

3. *(Strong Simulability) There exists a black-box simulator S_{RS} running in time $\mathrm{poly}(n)$, such that for all deterministic[7] Arthur strategies A^*_{RS}, the statistical difference between the following distributions is $2^{-\Omega(n)}$:*

   (I) *Execute $(A^*_{RS}, M_{RS})(n)$, let $\alpha \in \{0,1\}^\ell$ be the output of the protocol, and let v be A^*_{RS}'s view of the interaction (i.e., v is a transcript (f, y, h, α)).[8]*

   (II) *Choose α uniformly from $\{0,1\}^\ell$. Output $(S_{RS}^{A^*_{RS}}(\alpha), \alpha)$.*

Remark 4.5.1 The α's are included in the outputs of Distributions (I) and (II) above to force the simulator to produce a transcript for an *externally specified* α (rather than an α which it generates on its own while producing the transcript.)

Proof: Efficiency is immediate from the description of the protocol and the properties of the families $\mathcal{F}$ and $\mathcal{H}$.

Soundness. Let M^*_{RS} be any cheating Merlin strategy and consider an execution of the protocol (M^*_{RS}, A_{RS}). Notice that that the probability that the output α lies in some set S is bounded above by the probability that $h^{-1}(0)$ contains an element of S. Now, for h chosen *uniformly* from $\mathcal{H}$ (instead of by the protocol), the probability that $h^{-1}(0)$ contains an element of S is at most

$$\sum_{\alpha \in S} \Pr_{h \in_R \mathcal{H}}[h(\alpha) = 0] = \frac{|S|}{2^{\ell - n}}.$$

In our protocol, h is chosen using the DGW protocol. It shown in [DGW94, Prop. 1] that a cheating Merlin can cause at most a $1/q$ statistical difference from the uniform distribution on $\mathcal{H}$, and so the Soundness property follows.

Strong Simulability. We now describe the simulator which will be used to establish Strong Simulability. Recall that p is polynomial bound on the size of $f^{-1}(y)$ for any $f \in \mathcal{F}$, s is the description length for elements of $\mathcal{H}$, and functions in $\mathcal{F}$ map $\{0,1\}^s$ to $\{0,1\}^t$, where $t = s - 4\log_2(3qs)$.

[7]The restriction to deterministic Arthur strategies is only for ease of presentation, as a simulator for randomized Arthur strategies can uniformly select and fix Arthur's coins and then use the simulator for deterministic strategies. When we use the Random Selection simulator as a subroutine in the simulator for the Transformed Protocol in Section 4.6, the coins of Arthur will have already been fixed by the outer simulator.

[8]In Chapter 2, we defined the Verifier's view to consist of his random coins and the Prover's messages. Here, we do not include random coins, as they are irrelevant for deterministic strategies. We also include Arthur's messages — this is unnecessary as they are functions of Merlin's messages, but it will be convenient for our presentation.

The simulator $S_{RS}^{A^*_{RS}}$, on input $\alpha \in \{0,1\}^\ell$, proceeds as follows:

S1. Let $f \in \mathcal{F}$ be the first message sent by A^*_{RS}.

S2. Repeat the following up to $n \cdot 2(3sq)^4 \cdot p$ times:

(a) Choose h' uniformly from $\mathcal{H}_\alpha$ (Recall that $\mathcal{H}_\alpha = \{h\colon h(\alpha) = 0\}$.

(b) Let $y = f(h')$ (i.e., y is the cell containing h'). Compute $k \stackrel{\text{def}}{=} |f^{-1}(y) \cap \mathcal{H}_\alpha|$. With probability $1 - \frac{1}{k}$, proceed to next iteration of Step S2. (Otherwise continue.)

(c) Let $h = A^*_{RS}(y)$, that is, the element (hereafter called the *cell representative*) of cell y that A^*_{RS} gives in Step 3 after being sent y in Step 2.

(d) If $h(\alpha) = 0$, output $((f, y, h, \alpha), \alpha)$ and terminate the simulation. Otherwise, proceed to next iteration of Step S2.

S3. If the simulator failed to produce output so far, output `fail`.

From the various properties of the families $\mathcal{F}$ and $\mathcal{H}$, such as the fact that $f^{-1}(y)$ can be enumerated in time poly(n), and the fact that s, q, and p are all poly(n), we see immediately that the running time of $S_{RS}^{A^*_{RS}}$ is poly(n).

Let us now show that Distributions (I) and (II) in Proposition 4.5.1 have statistical difference $2^{-\Omega(n)}$. Each produces output of the form $((f, y, h, \alpha), \alpha)$. In both cases, f is the (deterministically chosen) first message of A^*_{RS} and $y = f(h)$, so it suffices to show that the distributions restricted to their (h, α) components are statistically close. We therefore define the Distributions (I') and (II') to be the Distributions (I) and (II) restricted to their (h, α) components. To analyze these distributions, we make use of the following Lemma, the proof of which is in Section 4.7.

Lemma 4.5.1 *There exists a universal constant $c > 0$, so that the following holds: Let $\mathcal{H}$ be the family of affine-linear maps from $D = \{0,1\}^\ell$ to $T = \{0,1\}^{\ell'}$, i.e. $h \in \mathcal{H}$ is of the form $h(x) = Ax + b$ for some matrix A and vector b. Let $S \subset \mathcal{H}$ be such that $|S| \geq \delta|\mathcal{H}|$. Let $\varepsilon = \frac{|T|}{|D|}$. Then*

Part 1: *The statistical difference between the following two distributions is at most $(c \cdot \varepsilon^{1/c}\delta^{-c})$:*

(A) *Choose $h \in_R S$. Let $x \in_R h^{-1}(0)$. Output (h, x).*

(B) *Choose $x \in_R D$. Let $h \in_R S \cap \mathcal{H}_x$. Output (h, x).*

Part 2: *For at least a $1 - (c \cdot \varepsilon^{1/c}\delta^{-c})$ fraction of $x \in D$,*

$$\frac{|S \cap \mathcal{H}_x|}{|\mathcal{H}_x|} \geq \delta/2.$$

When we apply the lemma, we take $\ell' = \ell - n$, $\varepsilon = 2^{-n}$, and $S = \{A^*_{RS}(y)\colon y \in \{0,1\}^t\}$. In other words, S is the set all possible *cell representatives* that A^*_{RS} can send in Step 3 of the protocol (M_{RS}, A^*_{RS}). Notice that

$$\delta \stackrel{\text{def}}{=} \frac{|S|}{|\mathcal{H}|} = \frac{2^t}{2^s} = 2^{-4\log_2(3sq)} = \frac{1}{(3sq)^4}.$$

and so, $c \cdot \varepsilon^{1/c}\delta^{-c} = 2^{-\Omega(n)}$. Now, observe that the protocol (M_{RS}, A^*_{RS}) selects h uniformly from S. (Recall that A^*_{RS} is deterministic.) Thus, Distribution (I') is exactly Distribution (A) of Lemma 4.5.1. Now we will show that the Distribution (II') is statistically close to Distribution (B).

Let us consider a single iteration of Step S2 in $S_{RS}^{A^*_{RS}}$. In such an iteration, h' is chosen uniformly from $\mathcal{H}_\alpha$, and $y = f(h')$. We write $f(\mathcal{H}_\alpha)$ to denote the set of images of elements of $\mathcal{H}_\alpha$ under f (i.e., $f(\mathcal{H}_\alpha) = \{f(h): h \in \mathcal{H}_\alpha\}$). In other words, $f(\mathcal{H}_\alpha)$ is the set of cells intersecting $\mathcal{H}_\alpha$. We want to establish that the distribution of h's produced by the simulator will be uniform in $S \cap \mathcal{H}_\alpha$. In order for this to happen, y must be uniformly selected from $f(\mathcal{H}_\alpha)$. If f was chosen honestly by A^*_{RS}, we would expect it to be one-to-one on the set $\mathcal{H}_\alpha$, since $\mathcal{H}_\alpha$ is a vanishingly small fraction of the domain. However, f is chosen adversarially, so we must do some work to ensure uniformity:

Notice that for any $y_0 \in f(\mathcal{H}_\alpha)$, the probability that $f(h') = y_0$ when uniformly selecting $h' \in \mathcal{H}_\alpha$ is exactly

$$\frac{|\mathcal{H}_\alpha \cap f^{-1}(y_0)|}{|\mathcal{H}_\alpha|}.$$

In Step S2b, any such choice is maintained with probability $1/|\mathcal{H}_\alpha \cap f^{-1}(y_0)|$. Thus the probability that $y = y_0$ after Steps S2a and S2b in S_{RS} is exactly

$$\frac{1}{|\mathcal{H}_\alpha|}.$$

This is independent of y_0, and therefore y is a uniformly chosen element of $f(\mathcal{H}_\alpha)$ — that is, a uniformly chosen cell intersecting $\mathcal{H}_\alpha$. (These probabilities sum up to $|f(\mathcal{H}_\alpha)|/|\mathcal{H}_\alpha|$, which may be less than 1; this is due to the possibility that the iteration ends prematurely in Step S2b.)

Now, since, in Step S2c, $h = A^*_{RS}(y)$ is taken to be the representative of cell y, the function h is uniformly distributed over the representatives of cells which intersect $\mathcal{H}_\alpha$. In Step S2d, we abandon any h not in $\mathcal{H}_\alpha$, so the resulting distribution on h is uniform over cell representatives in $\mathcal{H}_\alpha$, that is, uniform over $S \cap \mathcal{H}_\alpha$. Thus a single iteration of the loop produces an h uniformly chosen from $S \cap \mathcal{H}_\alpha$, if it manages to produce output at all. This is identical to how h is chosen in Distribution (B) of Lemma 4.5.1. So, to show that the Distribution (II') is statistically close to Distribution (B), we need only to show that the probability that the repeat loop fails to produce output in all its $\text{poly}(n)$ iterations is $2^{-\Omega(n)}$ for at least a $1 - 2^{-\Omega(n)}$ fraction of the α's in $\{0,1\}^\ell$. We do this by showing that each iteration produces output with probability at least n times the reciprocal of the number of iterations.

There are two places in which an iteration can be exited, causing it to fail to produce output — Steps S2b and S2d. Observe that the simulator never exits in Step S2d if h' chosen in Step S2a lies in S, because then h will equal h'. This occurs with probability

$$\frac{|S \cap \mathcal{H}_\alpha|}{|\mathcal{H}_\alpha|}.$$

By Lemma 4.5.1, for at least a $1 - 2^{-\Omega(n)}$ fraction of $\alpha \in \{0,1\}^\ell$, this quantity is at least $\delta/2 = 1/2(3sq)^4$.

Now suppose that h' has been chosen in S. The probability of not exiting in Step S2b is at least $1/|f^{-1}(y)|$, which is at least $1/p$ by the properties of the family $\mathcal{F}$. Thus, for a $1 - 2^{-\Omega(n)}$ fraction of the α's, a single iteration produces output with probability at least $1/(2(3sq)^4 \cdot p)$. Since there are $(2(3sq)^4 \cdot p) \cdot n$ iterations, output is produced with probability $1 - 2^{-\Omega(n)}$.

We have shown that Distribution (I') is identical to Distribution (A) in Lemma 4.5.1 and Distribution (II') has a statistical difference of $2^{-\Omega(n)}$ from Distribution (B). So, by Lemma 4.5.1, we conclude that Distributions (I) and (II) have statistical difference $2^{-\Omega(n)}$ and Strong Simulability is established. ■

4.6 Proofs for the Transformed Protocol

In this section, we prove Lemma 4.4.1, establishing the desired properties of the transformed protocol using the properties of the Random Selection protocol which we established in the previous section.

Number of Rounds. Since the Random Selection protocol consists of 4 message exchanges, with the last message from Merlin to Arthur (and thus β can be sent along with Merlin's choice of α), we have that the transformed protocol exchanges $4r$ messages, exactly twice the number which the original protocol did. This establishes Property 1 of Lemma 4.4.1.

Efficiency. Since the actions of both Arthur as well as Merlin in the Random Selection protocol can be implemented in probabilistic polynomial time, it is clear that $\mathbf{M}$ can be implemented in probabilistic polynomial time given oracle access to M. This establishes Property 2 of Lemma 4.4.1.

Completeness. It is immediate from the description of the transformed protocol (together with the fact that the Random Selection protocol always produces an output if both parties act according to the protocol) that if the original proof system has perfect completeness, then so does the transformed proof system. Otherwise, since the proof system (M, A) has completeness error at most 2^{-n} by Lemma 4.3.1, and the Random Selection protocol yields an output distribution on α which has statistical difference at most $2^{-\Omega(n)}$ from uniform, it follows that the completeness error of the transformed protocol is exponentially vanishing. This establishes Property 3 of Lemma 4.4.1.

Soundness. Let x be some string in Π_N. Assume the polynomial $p(n)$ (chosen for the Soundness error to be at most $1/p(n)$) is non-constant. In the protocol, when the Random Selection protocol is invoked for the ith time (in order to select α_i), we will denote the hash function selected (from $\mathcal{H}$) in that execution by h_i. First we observe that there is a strategy $\mathbf{M}^*$ which maximizes the probability that $\mathbf{A}$ accepts satisfying the following conditions:

1. $\mathbf{M}^*$ is deterministic.
2. $\mathbf{M}^*$ always chooses α_i such that $h_i(\alpha_i) = 0$ (as long as $h_i^{-1}(0) \neq \emptyset$) and this choice depends only on
   $\alpha_1, \ldots, \alpha_{i-1}, \beta_1, \ldots, \beta_{i-1}, h_i$ (i.e., not on the messages exchanged during the Random Selection protocols or the previous h_j's.)
3. $\mathbf{M}^*$'s choice of β_i only depends on $\alpha_1, \ldots, \alpha_i, \beta_1, \ldots, \beta_{i-1}$.

The first condition can always be met, for any proof system. The last two conditions can be achieved because $\mathbf{A}$'s behavior (in particular, his decision to accept) after h_i is selected only depends on the stated quantites.

Now consider a modified version of the protocol in which the h_i's are chosen uniformly from $\mathcal{H}$ by Arthur rather than through the first three steps of the Random Selection protocol. More precisely, let $\overline{\mathbf{A}}$ be the verifier strategy obtained by modifying $\mathbf{A}$ to choose h_i's uniformly from $\mathcal{H}$ rather than participate in the first three steps of the Random Selection protocol. By the properties above, $\mathbf{M}^*$ defines a strategy against $\overline{\mathbf{A}}$ and we can consider the probability that $\mathbf{M}^*$ convinces $\overline{\mathbf{A}}$ to accept.

Claim 4.6.1 $\mathbf{M}^*$ *makes* $\overline{\mathbf{A}}$ *accept with probability at most* $2^{-\Omega(n)}$.

Proof: We upper bound the probability that $\mathbf{M}^*$ gets $\overline{\mathbf{A}}$ to accept. Notice that, for every sequence $\vec{\alpha} = (\alpha_1, \ldots, \alpha_r) \in \{0,1\}^{r\ell}$, Property 3 of $\mathbf{M}^*$ uniquely determines a sequence $\vec{\beta}(\vec{\alpha}) = (\beta_1, \ldots, \beta_r)$ of $\mathbf{M}^*$ responses. Call a sequence $\vec{\alpha}$ *accepting* if A would accept on coins $\vec{\alpha}$ and prover responses $\vec{\beta}(\vec{\alpha})$. Notice that $\overline{\mathbf{A}}$ accepts iff the protocol chooses an accepting $\vec{\alpha}$. So we first bound the number of accepting $\vec{\alpha}$'s and then we bound the probability that any given $\vec{\alpha}$ occurs.

By Property 3, we may consider $\mathbf{M}^*$'s success probability against the original verifier A. In this case, A will accept iff his coins $\vec{\alpha}$ are accepting. The soundness of the original proof system says that at most $2^{\ell r} \cdot 2^{-(r+1)n}$ of the $\vec{\alpha}$'s are accepting.

Now, for any $\alpha \in \{0,1\}^{\ell}$, the probability that $h(\alpha) = 0$ for $h \in_R \mathcal{H}$ is exactly $2^{-(\ell-n)}$. Thus, for every sequence $\vec{\alpha} = (\alpha_1, \ldots, \alpha_r) \in \{0,1\}^{r\ell}$, the probability that this sequence of α_i's occurs in the interaction between $\mathbf{M}^*$ and $\overline{\mathbf{A}}$ is bounded above by $2^{-r\cdot(\ell-n)}$.

Thus, by a simple union bound, the probability that an accepting sequence occurs in $(\mathbf{M}^*, \overline{\mathbf{A}})$ is at most $(2^{\ell r} \cdot 2^{-(r+1)n}) \cdot 2^{-r(\ell-n)} = 2^{-n}$. ∎

Now we consider the success probability of $\mathbf{M}^*$ against $\mathbf{A}$ rather than $\overline{\mathbf{A}}$ — that is, when the h_i's are not selected uniformly, but according to the Random Selection protocol. By the Soundness property of the Random Selection protocol, $\mathbf{M}^*$ can induce a statistical difference of at most $1/2pr$ from uniform on the choice of each h_i. Thus the joint distribution of $(h_1, \ldots, h_r)$ can have a statistical difference of at most $1/2p$ from uniform. By Fact 2.1.4. this means that $\mathbf{A}$ will accept with probability at most $1/2p$ greater than $\overline{\mathbf{A}}$ does. Thus, $\mathbf{A}$ accepts with probability at most $2^{-\Omega(n)} + 1/2p(n) < 1/p(n)$, for sufficiently large n. This establishes Property 4 of Lemma 4.4.1.

Zero Knowledge. We now describe the simulator for the protocol $(\mathbf{M}, \mathbf{A})$ of Section 4.4. Let S be the simulator for the honest verifier in the original protocol (M, A). We will give a universal simulator $\mathbf{S}$ for $(\mathbf{M}, \mathbf{A})$ which uses any verifier strategy $\mathbf{A}^*$ as a black-box.

The simulator $\mathbf{S}^{\mathbf{A}^*}$, on input x:

1. Uniformly choose and fix random coins c for $\mathbf{A}^*$ to obtain a deterministic strategy $A^{(1)}$.
2. Run the original honest-verifier simulator to obtain a transcript $(\alpha_1, \beta_1, \ldots, \alpha_r, \beta_r) \leftarrow S(x)$.
3. For $i = 1$ to r, do the following:
   (a) Run the strong simulator for the Random Selection protocol, on input α_i with Arthur strategy $A^{(i)}$, to obtain a simulated transcript t_i of the Random Selection protocol (i.e., $t_i \leftarrow S_{RS}^{A^{(i)}}(\alpha_i)$).
   (b) Let $A^{(i+1)}$ be the state of $A^{(i)}$ after additional history t_i, α_i, β_i.
4. Output $(t_1, \alpha_1, \beta_1, \ldots, t_r, \alpha_r, \beta_r; c)$.

To prove that the above simulator has the desired properties, we first consider its output distribution in the case that the original honest-verifier simulator S is perfect: Let $\overline{\mathbf{S}}^{\mathbf{A}^*}$ be the output distribution of $\mathbf{S}^{\mathbf{A}^*}$ if the output of S in Step 2 is replaced with a true sample $(\alpha_1, \beta_1, \ldots, \alpha_r, \beta_r)$ of the protocol (M, A). By an induction argument using the strong simulability property of the Random Selection protocol, it is easy to show the following:

Claim 4.6.2 $\overline{\mathbf{S}}^{\mathbf{A}^*}(x)$ *and* $(\mathbf{M}, \mathbf{A}^*)(x)$ *have statistical difference at most* $2^{-\Omega(n)}$.

Proof: By induction on i, we show that the view of $\mathbf{A}^*$ up to β_i (which we will denote by $T_i^* = (t_1, \alpha_1, \beta_1, \ldots, t_i, \alpha_i, \beta_i; c)_{(\mathbf{M},\mathbf{A}^*)}$) has statistical difference at most $i2^{-\Omega(n)}$ from the output of $\overline{\mathbf{S}}^{\mathbf{A}^*}$ up to the same point (denoted $\overline{T}_i = (t_1, \alpha_1, \beta_1, \ldots, t_i, \alpha_i, \beta_i; c)_{\overline{\mathbf{S}}}$). For $i = 0$, this is clear.

Let us consider what happens in both the interaction between $\mathbf{M}$ and $\mathbf{A}^*$ and in the simulator $\overline{\mathbf{S}}^{\mathbf{A}^*}$ conditioned on a partial transcript T_i (i.e., conditioned on $T_i^* = T_i$ and $\overline{T}_i = T_i$, respectively.) Let $A^{(i+1)}$ be $\mathbf{A}^*$ with history T_i. By the specification of $\mathbf{M}$, the process by which t_{i+1} and α_{i+1} are obtained in the interaction between $\mathbf{M}$ and $\mathbf{A}^*$ is exactly Distribution (I) in the strong simulability condition of the Random Selection protocol (Prop. 4.5.1), taking Arthur to be $A^{(i+1)}$. Now, in $\overline{\mathbf{S}}^{\mathbf{A}^*}$, it is clear that each α_{i+1} is uniform and independent of $(\alpha_1, \beta_1, \ldots, \alpha_i, \beta_i)$ (and thus also of $\overline{T}_i$). Therefore, the process by which t_{i+1} and α_{i+1} are obtained in $\overline{\mathbf{S}}^{\mathbf{A}^*}$ is exactly Distribution (II) in the strong simulability condition of the Random Selection protocol. The strong simulability condition tells us that Distributions (I) and (II) have statistical difference $2^{-\Omega(n)}$. Moreover, β_{i+1} is chosen according to the same distribution (conditioned on T_i, t_{i+1} and α_{i+1}) in both $(\mathbf{M}, \mathbf{A}^*)$ and $\overline{\mathbf{S}}^{\mathbf{A}^*}$ — that is, according to the original M strategy. So the β_{i+1}'s cannot increase the statistical difference. We have argued that $(T_{i+1}^* | T_i^* = T_i)$ and $(\overline{T}_{i+1} | \overline{T}_i = T_i)$ have statistical difference at most $2^{-\Omega(n)}$ for every T_i. Thus, $\|\overline{T}_{i+1} - T_{i+1}^*\| \leq \|\overline{T}_i - T_i^*\| + 2^{-\Omega(n)} \leq (i+1)2^{-\Omega(n)}$, completing the induction.

We have shown that the total statistical difference is at most $r(n) \cdot 2^{-\Omega(n)} = 2^{-\Omega(n)}$. ■

Now we deduce Lemma 4.4.1, Parts 5 and 6, from Claim 4.6.2.

Statistical Zero-Knowledge. Using the output of S instead of a true sample from (M, A) can increase the simulator deviation by at most $\|S(x) - (M, A)(x)\|$, which is exactly the simulator deviation for the protocol (M, A).

Computational Zero-Knowledge. We claim that the ensembles of probability distributions $X_1 \stackrel{\text{def}}{=} \{(\mathbf{M}, \mathbf{A}^*)(x)\}_{x \in \Pi_Y}$ and $X_2 \stackrel{\text{def}}{=} \{\mathbf{S}^{\mathbf{A}^*}(x)\}_{x \in \Pi_Y}$ are computationally indistinguishable for any probabilistic polynomial-time $\mathbf{A}^*$. Consider the ensemble $X_3 \stackrel{\text{def}}{=} \{\overline{\mathbf{S}}^{\mathbf{A}^*}(x)\}_{x \in \Pi_Y}$. By Claim 4.6.2, X_1 and X_3 are statistically close and therefore computationally indistinguishable. We claim that X_2 and X_3 are computationally indistinguishable, for any probabilistic polynomial-time $\mathbf{A}^*$. This holds because any distinguisher D between X_2 and X_3 can be transformed into a distinguisher D' between $\{(M, A)(x)\}_{x \in \Pi_Y}$ and $\{S(x)\}_{x \in \Pi_Y}$, which are computationally indistinguishable by hypothesis. The new distinguisher D' operates as follows: Given a transcript T of either of the latter two ensembles, perform the procedure specified by $\mathbf{S}^{\mathbf{A}^*}$, replacing the execution in Step 2 with T, and feed the output of $\mathbf{S}^{\mathbf{A}^*}$ to D. When T is selected according to $\{(M, A)(x)\}_{x \in \Pi_Y}$, D is fed with ensemble X_3, whereas when T is selected according to $\{S(x)\}_{x \in \Pi_Y}$, D is fed with ensemble X_2.

Remark 4.6.1 The above proof actually shows that, for any (not just probabilistic polynomial-time) verifier $\mathbf{A}^*$, if $(\mathbf{M}, \mathbf{A}^*)$ and $\mathbf{S}^{\mathbf{A}^*}$ can be distinguished by algorithm D, then there is an algorithm no more powerful than $\mathbf{A}^*$ and D (i.e., a probabilistic polynomial time machine with oracle access to

$\mathbf{A}^*$ and D) that can distinguish the original honest-verifier proof system (M, A) from its simulator S. So, if the honest-verifier simulator produces transcripts indistinguishable from (M, A) by any machine running in, say, quasi-polynomial time, then the new protocol $(\mathbf{M}, \mathbf{A})$ is zero-knowledge against all quasi-polynomial time verifiers.

4.7 Proof of Hashing Lemma

Here we provide a proof of the Hashing Lemma used to establish the main result of this chapter. We restate the lemma here:

Lemma 4.7.1 (Hashing Lemma) *There exists a universal constant $c > 0$, so that the following holds: Let $\mathcal{H}$ be the family of affine-linear maps from $D = \{0,1\}^\ell$ to $T = \{0,1\}^{\ell'}$, i.e. $h \in \mathcal{H}$ is of the form $h(x) = Ax + b$ for some matrix A and vector b. Let $S \subset \mathcal{H}$ be such that $|S| \geq \delta|\mathcal{H}|$. Let $\varepsilon = \frac{|T|}{|D|}$. Then*

Part 1: *The statistical difference between the following two distributions is at most $c \cdot \varepsilon^{1/c}\delta^{-c}$:*

$A = (A_\mathcal{H}, A_X)$*: Let $h \in_R S$. Let $x \in_R h^{-1}(0)$. Output (h, x).*

$B = (B_\mathcal{H}, B_X)$*: Let $x \in_R D$. Let $h \in_R S \cap \mathcal{H}_x$. Output (h, x).*

Part 2: *For at least a $1 - (c \cdot \varepsilon^{1/c}\delta^{-c})$ fraction of $x \in D$,*

$$\frac{|S \cap \mathcal{H}_x|}{|\mathcal{H}_x|} \geq \frac{1}{2} \cdot \frac{|S|}{|H|} \geq \frac{\delta}{2}.$$

Proof: We define a *perfect* hash function $h \in \mathcal{H}$ to be one of the form $h(x) = Ax + b$, where the matrix A is full rank (and hence h is surjective). Note that a straightforward calculation shows that at most an ε fraction of the functions in $\mathcal{H}$ are *not* perfect.

We first establish Part 1 of the Hashing Lemma for the special case of perfect hash functions.

Sublemma 4.7.1 *Part 1 of the Hashing Lemma holds when S contains only perfect hash functions.*

Proof: First, we consider the relationship between distributions A_X and B_X.

Claim 4.7.1 $\|A_X - B_X\| \leq \frac{3\varepsilon^{1/3}}{\delta}$.

Proof: Note B_X is uniform over D. To establish the claim, it suffices to show that for all $C \subseteq D$,

$$\left|\Pr[A_X \in C] - \frac{|C|}{|D|}\right| \leq \frac{3\varepsilon^{1/3}}{\delta}.$$

Note $\left|\Pr[A_X \in C] - \frac{|C|}{|D|}\right| = \left|\Pr[A_X \in (D \setminus C)] - \frac{|D \setminus C|}{|D|}\right|$, so it suffices to consider sets C such that $\frac{|C|}{|D|} \geq \frac{1}{2}$. From the definition of A, we observe:

$$\Pr[A_X \in C] = \frac{1}{|S|} \sum_{h \in S} \frac{|h^{-1}(0) \cap C|}{|h^{-1}(0)|} = \frac{1}{|S|} \sum_{h \in S} \varepsilon \cdot |h^{-1}(0) \cap C|$$

where the last equality is due to our assumption that every $h \in S$ is perfect, and hence $|h^{-1}(0)| = 1/\varepsilon$.

To analyze the expression above, which refers to a sum over $h \in S$, we first consider the behaviour of the sum over all $h \in \mathcal{H}$. Here, we can use Chebyshev's inequality. Consider the probability space uniform over $\mathcal{H}$, and define, for every $x \in C$, an indicator random variable:

$$\chi_x(h) = \begin{cases} 1 \text{ if } h(x) = 0 \\ 0 \text{ otherwise} \end{cases}$$

Let $W_C(h) = \varepsilon \cdot |h^{-1}(0) \cap C| = \varepsilon \cdot \sum_{x \in C} \chi_x(h)$. Since $\mathcal{H}$ is a 2-universal family of hash functions, the χ_x's are pairwise independent with $\Pr_{h \in \mathcal{H}}[\chi_x(h) = 1] = \frac{1}{|T|} = \frac{1}{\varepsilon \cdot |D|}$. Thus, we have that:

$$\mathrm{E}_{h \in \mathcal{H}}[W_C(h)] = \varepsilon \cdot \sum_{x \in C} \mathrm{E}_{h \in \mathcal{H}}[\chi_x(h)] = \varepsilon \cdot \sum_{x \in C} \frac{1}{|T|} = \frac{|C|}{|D|}.$$

$$\mathrm{Var}_{h \in \mathcal{H}}[W_C(h)] = \varepsilon^2 \cdot \sum_{x \in C} \mathrm{Var}_{h \in \mathcal{H}}[\chi_x(h)] = \varepsilon^2 \cdot \sum_{x \in C} \frac{1}{|T|}\left(1 - \frac{1}{|T|}\right) < \varepsilon \cdot \frac{|C|}{|D|}.$$

By Chebyshev's inequality,

$$\begin{aligned} \Pr_{h \in \mathcal{H}}\left[\left|W_C(h) - \frac{|C|}{|D|}\right| > \varepsilon^{1/3} \cdot \frac{|C|}{|D|}\right] &< \frac{\mathrm{Var}[W_C]}{\left(\varepsilon^{1/3} \cdot \frac{|C|}{|D|}\right)^2} \\ &< \frac{\varepsilon^{1/3}|D|}{|C|} \leq 2\varepsilon^{1/3} \end{aligned}$$

where the last inequality is because $|C| \geq |D|/2$. Since $\frac{|S|}{|\mathcal{H}|} \geq \delta$, we can apply the above to the probability space uniform over S and conclude,

$$\Pr_{h \in S}\left[\left|W_C(h) - \frac{|C|}{|D|}\right| > \varepsilon^{1/3}\frac{|C|}{|D|}\right] < \frac{2\varepsilon^{1/3}}{\delta}.$$

Recall,

$$\Pr[A_X \in C] = \frac{1}{|S|}\sum_{h \in S} W_C(h).$$

Hence, for all but at most $\frac{2\varepsilon^{1/3}}{\delta} \cdot |S|$ terms in the sum, we have that $\left|W_C(h) - \frac{|C|}{|D|}\right| \leq \varepsilon^{1/3}\frac{|C|}{|D|}$. Since for every h it is true that $0 \leq W_C(h) \leq 1$, we have,

$$\left|\Pr[A_X \in C] - \frac{|C|}{|D|}\right| \leq \varepsilon^{1/3}\frac{|C|}{|D|} + \frac{2\varepsilon^{1/3}}{\delta} \leq \frac{3\varepsilon^{1/3}}{\delta}.$$

And the claim is proved. ■

We are now ready to complete the proof of this sublemma. For all $x \in D$ and all $h \in S$ such that $h(x) = 0$, we have, by Bayes' Law:

$$\begin{aligned} \Pr[A_{\mathcal{H}} = h | A_X = x] &= \frac{\Pr[A_X = x | A_{\mathcal{H}} = h] \cdot \Pr[A_{\mathcal{H}} = h]}{\Pr[A_X = x]} \\ &= \frac{|h^{-1}(0)|^{-1} \cdot |S|^{-1}}{\Pr[A_X = x]} = \frac{\varepsilon \cdot |S|^{-1}}{\Pr[A_X = x]} \end{aligned}$$

where the last step is because for all perfect h, $|h^{-1}(0)| = 1/\varepsilon$. Note that this value has no dependence on h. Hence, for every x, given $A_X = x$, the distribution $A_{\mathcal{H}}$ is uniform over $\{h \in S : h(x) = 0\}$. Note that for all x, given $B_X = x$, $B_{\mathcal{H}}$ is also uniform over the same set. Thus, conditioned on the value of x, the distributions $A_{\mathcal{H}}$ and $B_{\mathcal{H}}$ are identical.

Hence $\|A - B\| = \|A_X - B_X\| \leq \epsilon_1$, and the sublemma is established. ■

Before we argue Part 1 of the Hashing Lemma in general, we will show how Part 2 follows from Sublemma 4.7.1. In the sequel, it will be convenient to introduce the following notation: For any subset $I \subseteq \mathcal{H}$, we will write I_x to denote the set $\{h \in I : h(x) = 0\}$.

In order to apply Sublemma 4.7.1, we will consider the subset $S' \subset S$ of all perfect hash functions in S. Since less than an ε fraction of all hash functions are not perfect, $|S'| \geq (1 - \frac{\varepsilon}{\delta})|S| \geq (\delta - \varepsilon) \cdot |\mathcal{H}|$. Similarly, we define the following two modifications of the distributions A and B, using S' instead of S:

$A' = (A'_{\mathcal{H}}, A'_X)$: Let $h \in_R S'$. Let $x \in_R h^{-1}(0)$. Output (h, x).

$B' = (B'_{\mathcal{H}}, B'_X)$: Let $x \in_R D$. Let $h \in_R S' \cap \mathcal{H}_x$. Output (h, x).

The following claim establishes Part 2 of the Hashing Lemma:

Claim 4.7.2 *Let* $\epsilon_1 \stackrel{\text{def}}{=} \frac{3\varepsilon^{1/3}}{\delta - \varepsilon}$. *For at least a* $(1 - \sqrt{\epsilon_1})$ *fraction of* $x \in D$, $\frac{|S_x|}{|\mathcal{H}_x|} \geq \delta/2$.

Proof: By the definition of A'_X,

$$\Pr[A'_X = x] = \frac{1}{|S'|} \sum_{h \in S'_x} \frac{1}{|h^{-1}(0)|} = \varepsilon \frac{|S'_x|}{|S'|}$$

where the last equality follows because $|h^{-1}(0)| = 1/\varepsilon$ for all $h \in S'$. However, by the Sublemma, $\|A'_X - B'_X\| \leq \epsilon_1$. Note that B'_X is uniform over D, so for a $(1 - \sqrt{\epsilon_1})$ fraction of $x \in D$, it must be that

$$\varepsilon \frac{|S'_x|}{|S'|} = \Pr[A'_X = x] \geq (1 - \sqrt{\epsilon_1}) \cdot \frac{1}{|D|}.$$

Thus,

$$\frac{|S_x|}{|\mathcal{H}_x|} \geq \frac{|S'_x|}{|\mathcal{H}_x|} \geq (1 - \sqrt{\epsilon_1}) \cdot \frac{|S'|}{\varepsilon |D| \cdot |\mathcal{H}_x|} = (1 - \sqrt{\epsilon_1}) \cdot \frac{|S'|}{|\mathcal{H}|}$$

where the last equality follows from $\varepsilon \cdot |D| = |T|$ and $|T| \cdot |\mathcal{H}_x| = |\mathcal{H}|$. Using the fact that $\frac{|S'|}{|\mathcal{H}|} \geq (1 - \frac{\varepsilon}{\delta}) \cdot \frac{|S|}{|\mathcal{H}|}$, we have, for a $(1 - \sqrt{\epsilon_1})$ fraction of $x \in D$,

$$\frac{|S_x|}{|\mathcal{H}_x|} \geq (1 - \sqrt{\epsilon_1}) \cdot (1 - \frac{\varepsilon}{\delta}) \cdot \delta \geq \frac{\delta}{2}.$$

Note that the final inequality follows because we can safely assume that $\sqrt{\epsilon_1} + \frac{\varepsilon}{\delta} < \frac{1}{2}$. This is because we can freely assume that $c \cdot \varepsilon^{1/c} \delta^{-c} < 1$, since otherwise the statement of the Hashing Lemma becomes trivially satisfied. Since $\sqrt{\epsilon_1} + \frac{\varepsilon}{\delta}$ is upper bounded by $k \cdot \varepsilon^{1/k} \delta^{-k}$ for some constant k, our assumption can be made to imply that $\sqrt{\epsilon_1} + \frac{\varepsilon}{\delta} < \frac{1}{2}$ by choosing $c > 2k$. ■

Finally, we establish Part 1 of the Hashing Lemma in general by showing that the presence of imperfect hash functions will not disturb our computations. First, we see immediately that since $|S'| \geq (1 - \frac{\varepsilon}{\delta})|S|$, the statistical difference between A and A' can be at most $\frac{\varepsilon}{\delta}$. To see that the statistical difference between B' and B is sufficiently small, it suffices to show that for almost all x, the probability that $B_{\mathcal{H}}$ outputs an imperfect hash function, given that $B_X = x$, is small. First we argue:

Claim 4.7.3 *For every $x \in D$, $\Pr_{h \in \mathcal{H}_x}[h \text{ is imperfect}] \leq \varepsilon$.*

Proof: Observe that for any $x \in D$, $\mathcal{H}_x$ consists exactly of those functions $h(y) = Ay + b$ where $b = -Ax$. Thus, there is exactly one function in $\mathcal{H}_x$ for every matrix A. Hence, the fraction of imperfect functions in $\mathcal{H}_x$ is precisely the fraction of matrices A that do not have full rank, which is at most ε. ■

For any $x \in D$, the probability that $B_{\mathcal{H}}$ outputs an imperfect hash function given that $B_X = x$ is

$$\Pr_{h \in S_x}[h \text{ is imperfect}] \leq \Pr_{h \in \mathcal{H}_x}[h \text{ is imperfect}] \cdot \frac{|\mathcal{H}_x|}{|S_x|}.$$

Using Claim 4.7.2 and Claim 4.7.3 above, we have that for at least a $(1 - \sqrt{\epsilon_1})$ fraction of $x \in D$, this probability is at most $\epsilon_2 \stackrel{\text{def}}{=} \varepsilon \cdot (2/\delta)$. Thus, $\|B - B'\| \leq (1 - \sqrt{\epsilon_1}) \cdot \epsilon_2 + \sqrt{\epsilon_1} \leq \epsilon_2 + \sqrt{\epsilon_1}$. We have already observed that $\|A' - A\| \leq \frac{\varepsilon}{\delta}$, and Sublemma 4.7.1 showed that $\|B' - A'\| \leq \epsilon_1$. Hence $\|A - B\| \leq \epsilon_1 + \frac{\varepsilon}{\delta} + \epsilon_2 + \sqrt{\epsilon_1}$, and the Hashing Lemma is established. ■

4.8 Consequences for SZK

In the previous chapter, we studied HVSZK in depth, and obtained many interesting results. In this chapter, we saw how to convert any honest-verifier statistical zero-knowledge proof into a general statistical zero-knowledge proof. In this section, we will see how the results of the previous chapter translate to SZK.

In this chapter, we showed that:

Theorem 4.8.1 HVSZK = SZK.

Of course, since Theorem 4.8.1 gives an equality of classes, the Completeness Theorem extends to the general class:

Proposition 4.8.1 STATISTICAL DIFFERENCE *is complete for* SZK.

We now look at the applications of the Completeness Theorem, beginning with our results on efficient HVSZK proof systems in Corollary 3.3.1.

Simulator deviation and security parametrization. Both the transformation of [Oka96] and the transformation of this chapter can be made to preserve a simulator deviation of $2^{-\Omega(n)}$. Applying these transformations to Corollary 3.3.1, we see that every language in HVSZK has a general statistical zero-knowledge proof with simulator deviation $2^{-\Omega(n)}$.

We can also consider a security-parametrized variant of general zero-knowledge, analogous to the honest-verifier case (Definition 3.3.1): the protocol takes an extra parameter k (in unary) and the zero-knowledge condition demands that, for any verifier, the simulator deviation is less than $\alpha(k)$ for some negligible function α. The transformation of [Oka96] and the transformation of this chapter both preserve the security-parametrization property, so we obtain:

Proposition 4.8.2 *Any promise problem in* HVSZK *has a general security-parametrized statistical zero-knowledge proof against arbitrary verifiers with simulator deviation 2^{-k}.*

Message complexity. Corollary 3.3.1 shows that every promise problem in SZK has a 2-message honest-verifier statistical zero-knowledge proof. Although the transformation of this chapter only multiplies the number of messages by a factor of two when applied to public-coin proof systems, the private to public-coin transformation of [Oka96] increases the number of messages to polynomial even when applied to a constant-message protocol. However, if one is willing to make a computational assumption, then the transformation of [BMO90] applies and this transformation does preserve the message complexity up to a constant factor.

Proposition 4.8.3 *If the discrete logarithm problem is hard,*[9] *then every promise problem in* HVSZK *has a constant-message general statistical zero-knowledge proof system with soundness and completeness errors* 2^{-n}.

Proof: Let Π be any promise problem in HVSZK. From Corollary 3.3.1, we know that Π has a 2-message (honest-verifier) statistical zero-knowledge proof system. Repeating this protocol in parallel $2^{-O(n)}$ times gives a constant round proof system with soundness and completeness errors 2^{-n}. Note that parallel repetition preserves honest-verifier statistical zero knowledge. Now, apply the transformation of [BMO90], which yields a constant-message general statistical zero-knowledge proof system for Π, under the assumption that the discrete logarithm is hard. This transformation only increases the number of messages by a constant factor and preserves the completeness and soundness error. ■

It is still open whether one can *unconditionally* prove that all of SZK has constant round any-verifier proofs. We note that Goldreich and Krawcyzk [GK96] have shown some limitations on the message complexity of general zero-knowledge proofs (for problems outside BPP): If the proof system has *negligible soundness error* and is zero knowledge under black-box simulation, then it cannot consist of fewer than 4 messages. If, in addition, it is public-coin, then it cannot consist of any constant number of messages.

Communication. Corollary 3.3.1 shows that every promise problem in HVSZK has a very communication-efficient honest-verifier statistical zero-knowledge proof, in that the prover only sends one bit to achieve completeness error $1-2^{-n}$ and soundness error $1/2+2^{-n}$. Unfortunately, none of the known transformations to general statistical zero-knowledge preserve the amount of communication, so this result does not translate to general statistical zero-knowledge.

Deterministic Prover. We note that the fact that the prover is deterministic in Corollary 3.3.1 *cannot* extend to any-ver SZK (unless SZK = BPP) [GO94].

Closure properties. Since Theorem 4.8.1 gives an equality of classes, any closure properties of the honest-verifier class (namely, Corollaries 3.3.2, 3.3.3, and 3.3.5, and Theorem 3.3.1) also hold for the general classs. So we immediately obtain the following:

Proposition 4.8.4 any-ver SZK *is closed under Karp reductions, complement,* $\Phi(\cdot)$, *and* NC^1 *truth-table reductions.*

Hard-on-average languages and one-way functions. These results are stronger for the honest-verifier class, because the existence of a hard-on-average problem in the general class implies the existence of one in the honest-verifier class (even without Theorem 4.8.1).

[9]See [BMO90] for a precise formulation of their assumption.

Chapter 5

Non-interactive Statistical Zero Knowledge

As we have seen in the previous chapters, statistical zero-knowledge interactive proofs have a rich and interesting theory. A crucial feature in the definition of zero-knowledge proofs is *interaction*. Indeed, Goldreich and Oren [GO94] showed that if one removes interaction from the definition of zero-knowledge proofs (*i.e.* allows only unidirectional communication from the prover to the verifier), then zero knowledge becomes trivial (*i.e.* zero-knowledge proofs only exist for problems in BPP). However, for many cryptographic challenges where zero-knowledge proofs may appear to have promising applications, such as encryption, interaction may not be feasible.

Surprisingly, however, Blum, Feldman, and Micali [BFM88], showed that by changing the model slightly, it is possible to achieve zero knowledge in a non-interactive setting (i.e. where only unidirectional communication can occur). Specifically, they assume that both Prover and Verifier have access to a shared truly random string, called the *reference string*. Aside from this assumption, all communication consists of one message, the "proof," which is generated by the Prover (based on the assertion being proved and the reference string) and sent from the Prover to the Verifier.

Non-interactive zero-knowledge proofs, by virtue of their communication efficiency, have several applications not offered by ordinary interactive zero-knowledge proofs. They have been used, among other things, to build digital signature schemes secure against adaptive chosen message attack [BG89], public-key cryptosystems secure against chosen-ciphertext attack [NY90, DDN91, S99], and non-malleable cryptosystems [DDN91].

In this chapter, we focus on the class NISZK of promise problems that possess non-interactive statistical zero-knowledge proofs, and its relationship with the interactive class SZK that we have already examined. Continuing in the spirit of our work on SZK, we show that there are natural problems which are complete for NISZK. We use these problems to draw conclusions about the relationship between NISZK and SZK. The work we present in this chapter is based on a paper [GSV99] authored jointly with Oded Goldreich and Salil Vadhan.

5.1 Overview

5.1.1 The Non-Interactive Model

The main difference between the models in which interactive and non-interactive zero knowledge are defined is that, in the non-interactive setting, we assume that all parties have access to a shared *random reference string*. In this modified model, called the *shared random reference string* model, we first define the notion of a non-interactive proof system:

Definition 5.1.1 *A* non-interactive proof system *with* completeness error $c(n)$ *and* soundness error $s(n)$ *for a promise problem* Π *is a pair of machines* P *and* V*, where* V *is probabilistic polynomial-time* P *is computationally unbounded, together with a polynomial* $r(n)$ *(which will give the size of the random* reference string σ*), such that:*

1. *(*Completeness*): If* $x \in \Pi_{\text{YES}}$*, then the probability that* $V(x, \sigma, P(x, \sigma))$ *accepts is at least* $1 - c(|x|)$.
2. *(*Soundness*): If* $x \in \Pi_{\text{NO}}$*, then the probability that* $V(x, \sigma, P(x, \sigma))$ *accepts is at most* $s(|x|)$.

where the probabilities in Conditions 1 and 2 are taken over the random coins of V and P, and the choice of σ *uniformly from* $\{0,1\}^{r(n)}$.

Now that we have defined non-interactive proofs in the shared random reference string model, we can define non-interactive statistical zero-knowledge proofs. Here, we adapt the definition given in [BDMP91] to promise problems.[1]

Definition 5.1.2 *A non-interactive proof system* (P, V) *for a promise problem* Π *is a* non-interactive statistical zero-knowledge proof system *if:*

1. *(*Completeness and Soundness*): The completeness and soundness errors are negligible functions.*
2. *(*Zero Knowledge*): For all* $x \in \Pi_{\text{YES}}$*, the statistical distance between the following two distributions is at most* $\beta(|x|)$*:*

   (A) Choose σ *uniformly from* $\{0,1\}^{r(|x|)}$*, sample p from* $P(x, \sigma)$*, and output* (p, σ).

   (B) $S(x)$ *(where the random coins for* S *are chosen uniformly at random.)*

where $\beta(n)$ *is a negligible function termed the* simulator deviation.

We define the class of all promise problems possessing non-interactive statistical zero-knowledge proofs NISZK.

Remark 5.1.1 We make several remarks concerning this definition:

1. One can also define perfect and computational non-interactive zero-knowledge proofs in the straightforward analogous manner. Since all our results concern only NISZK, we omit formal definitions for these other classes.
2. Our definition captures what [BDMP91] call a *bounded proof system*, in that for any given shared random reference string, only one proof can be simulated. In contrast to non-interactive *computational* zero knowledge (cf., [BDMP91, FLS90]), it is unknown whether every problem that has such a (bounded) non-interactive *statistical* zero-knowledge proof system also has one in which the shared reference string can be used an unbounded (polynomial) number of times.

[1]Actually, only non-interactive *perfect* and computational zero-knowledge proofs were defined in [BDMP91]. The definition we are using, previously given in [BR90, DDPY98], is the natural non-interactive analogue of (interactive) statistical zero knowledge [GMR89].

3. Recall that in the context of interactive zero-knowledge proofs, another issue that arises in the zero-knowledge condition is the behavior of the verifier – whether it behaves honestly or not. Note that in the case of non-interactive zero knowledge, the issue of honest verifiers does not arise since the verifier does not interact with the prover at all. Also, note that we can always transform a non-interactive zero-knowledge proof into an honest verifier zero-knowledge proof, since we could have the honest verifier supply a random string which can replace the common reference string required for non-interactive zero knowledge. This implies that NISZK $\subset$ SZK (recalling the equivalence of SZK with HVSZK).

We also define a weaker notion of zero knowledge, known as a *weak non-interactive statistical zero-knowledge proof system,* analogous to the notion of weak interactive statistical zero-knowledge that we defined in Definition 3.3.6 in Chapter 4.

Definition 5.1.3 *A non-interactive proof system* (P, V) *for a promise problem* Π *is a* weak non-interactive statistical zero-knowledge proof system *if:*

1. *(*Completeness and Soundness*): The completeness and soundness errors are negligible functions.*
2. *(*Zero Knowledge*): For all polynomials* $p(\cdot)$*, there exists an efficient probabilistic (strict) polynomial-time algorithm* S_p *such that for all sufficiently long* $x \in \Pi_Y$*, we have that the statistical distance between the following two distributions is at most* $1/p(|x|)$*:*

   (A) Choose σ *uniformly from* $\{0,1\}^{r(|x|)}$*, sample p from* $P(x,\sigma)$*, and output* (p,σ)*.*

   (B) $S_p(x)$ *(where the random coins for* S_p *are chosen uniformly at random.)*

We define the class of all promise problems possessing weak non-interactive statistical zero-knowledge proofs weak-NISZK.

Remark 5.1.2 Clearly, NISZK $\subset$ weak-NISZK.

5.1.2 Previous work

Since its introduction in [BFM88], most work on non-interactive zero knowledge has focused on the computational type (cf., [BFM88, DMP87, DMP88, BDMP91, FLS90, KP98]). With non-interactive statistical zero knowledge, the main objects of investigation have been the specific proof system for QUADRATIC NONRESIDUOSITY and variants [BDMP91, DDP94, DDP97].[2] Recently, De Santis *et. al.* [DDPY98] opened the door to a general study of non-interactive statistical zero-knowledge by showing that it contains a complete (promise) problem[3]. In this chapter, we build on this work, and seek a better understanding of non-interactive statistical zero-knowledge proofs, and their relationship to their interactive counterparts.

[2]The only exception is an unpublished manuscript of Bellare and Rogaway [BR90] who proved some basic results about non-interactive perfect zero-knowledge and showed a non-interactive perfect zero-knowledge proof for the language of graphs with trivial automorphism group.

[3]This was done subsequent to our discovery of complete problems for interactive statistical zero knowledge, which appeared in [SV97].

5.1.3 Our Contribution

In this chapter, we seek to understand what, if any, additional power interaction gives in the context of statistical zero knowledge. Thus, we continue the investigation of NISZK, focusing on its relationship with SZK. We obtain two main results relating NISZK and SZK:

1. We show that the non-triviality of SZK is equivalent to the non-triviality of NISZK, where by non-trivial we mean that a class includes problems which are *not* solvable in probabilistic polynomial-time. In other words $\mathsf{SZK} \neq \mathsf{BPP} \iff \mathsf{NISZK} \neq \mathsf{BPP}$. The hypothesis that $SZK \neq \mathsf{BPP}$ holds under various assumptions, such as the intractability of the Discrete Logarithm Problem [GK93] or approximate versions of the Shortest and Closest Vector Problems in a lattice [GG98]. It was not previously known that NISZK must be non-trivial if these assumptions hold.

2. Furthermore, we show that if NISZK is closed under complement, then in fact $\mathsf{SZK} = \mathsf{NISZK}$ — *i.e.* , all statistical zero-knowledge proofs can be made non-interactive.[4]

We also show that $\mathsf{NISZK} = \text{weak-}\mathsf{NISZK}$.

Complete Problems. Central to our methodology is the continued use of simple and natural complete problems to understand classes, such as SZK and NISZK, whose definitions are rather complicated. In particular, we exhibit two natural promise problems and prove that they are complete for NISZK. The two problems refer to the "distance" (in two different senses) of a given distribution from the uniform one. These two problems are natural restrictions of two promise problems known to be complete for SZK – the problem StatisticalDifference introduced in Chapter 3 and another similar problem, called EntropyDifference[5]. Indeed, our results about the relationship between SZK and NISZK come from showing reductions between the corresponding complete problems. Thus we continue the general theme of using completeness to simplify the study of a class.

The richness of NISZK. Since we obtain our results comparing NISZK and SZK by exhibiting special reductions from problems complete for SZK to NISZK, we obtain a very interesting additional consequence: For *every* problem in SZK, there is a corresponding problem in NISZK of comparable computational complexity. Since we know SZK to be a rich class with infinite families of problems believed to be intractable (see Section 3.3.2 of Chapter 3), this implies that NISZK enjoys similar richness.

5.2 The Complete Problems

Continuing in the spirit of Chapter 3, the primary tools we use in our investigation are promise problems that are complete for SZK or NISZK. In Chapter 3, we showed that the promise problem Statistical Difference(SD) is complete for SZK, providing the first completeness result for SZK. After the intial publication of our result [SV97], it was shown in [GV99] that another natural

[4]We note that [DDPY98] had claimed that NISZK is closed under complement (and OR), but these claims have been retracted [DDPY99].

[5]EntropyDifference was shown to be complete for SZK by [GV99] after the publication of our work showing the completeness of StatisticalDifference [SV97].

problem, called ENTROPY DIFFERENCE (ED), is complete for SZK as well. In this work, we show that "one-sided" versions of these problems, which we call STATISTICAL DIFFERENCE FROM UNIFORM (SDU) and ENTROPY APPROXIMATION (EA), are complete for NISZK.

Definition 5.2.1 (Problems involving statistical difference): *Recall that the promise problem* STATISTICAL DIFFERENCE, *denoted* $\mathrm{SD} = (\mathrm{SD}_{\mathrm{YES}}, \mathrm{SD}_{\mathrm{NO}})$, *consists of*

$$\begin{aligned}\mathrm{SD}_{\mathrm{YES}} &\stackrel{\mathrm{def}}{=} \{(X,Y) : \|X - Y\| < 1/3\}\\ \mathrm{SD}_{\mathrm{NO}} &\stackrel{\mathrm{def}}{=} \{(X,Y) : \|X - Y\| > 2/3\}\end{aligned}$$

where X and Y are distributions encoded as circuits which sample from them. The promise problem STATISTICAL DIFFERENCE FROM UNIFORM, *denoted* $\mathrm{SDU} = (\mathrm{SDU}_{\mathrm{YES}}, \mathrm{SDU}_{\mathrm{NO}})$, *consists of*

$$\begin{aligned}\mathrm{SDU}_{\mathrm{YES}} &\stackrel{\mathrm{def}}{=} \{X : \|X - U\| < 1/n\}\\ \mathrm{SDU}_{\mathrm{NO}} &\stackrel{\mathrm{def}}{=} \{X : \|X - U\| > 1 - 1/n\}\end{aligned}$$

where X is a distribution encoded as a circuit outputting n bits, and U is the uniform distribution on n bits.

For the two problems related to entropy, we recall that the (Shannon) entropy of a random variable X, denoted $\mathrm{H}(X)$, is defined as

$$\mathrm{H}(X) \stackrel{\mathrm{def}}{=} \sum_{\alpha} \Pr[X = \alpha] \cdot \log_2(1/\Pr[X = \alpha])$$

Definition 5.2.2 (Problems involving entropy): *The promise problem* ENTROPY DIFFERENCE, *denoted* $\mathrm{ED} = (\mathrm{ED}_{\mathrm{YES}}, \mathrm{ED}_{\mathrm{NO}})$, *consists of*

$$\begin{aligned}\mathrm{ED}_{\mathrm{YES}} &\stackrel{\mathrm{def}}{=} \{(X,Y) : \mathrm{H}(X) > \mathrm{H}(Y) + 1\}\\ \mathrm{ED}_{\mathrm{NO}} &\stackrel{\mathrm{def}}{=} \{(X,Y) : \mathrm{H}(Y) > \mathrm{H}(X) + 1\}\end{aligned}$$

The promise problem ENTROPY APPROXIMATION, *denoted* $\mathrm{EA} = (\mathrm{EA}_{\mathrm{YES}}, \mathrm{EA}_{\mathrm{NO}})$, *consists of*

$$\begin{aligned}\mathrm{EA}_{\mathrm{YES}} &\stackrel{\mathrm{def}}{=} \{(X,k) : \mathrm{H}(X) > k + 1\}\\ \mathrm{EA}_{\mathrm{NO}} &\stackrel{\mathrm{def}}{=} \{(X,k) : \mathrm{H}(X) < k - 1\}\end{aligned}$$

In these problems, k is a positive integer and X and Y are distributions encoded as circuits which sample from them.

5.3 Formal Statement of Results

Our first theorem, which is the starting point for our other results, is:

Theorem 5.3.1 (EA and SDU are NISZK-complete) *The promise problems* EA *and* SDU *are complete for* NISZK. *That is,* EA, SDU $\in$ NISZK *and for every promise problem* $\Pi \in$ NISZK, *there is a polynomial-time Karp (many-one) reduction from* Π *to* EA *and another from* Π *to* SDU.

From the proof of this theorem, we also obtain a method for transforming weak non-interactive statistical zero knowledge proofs into standard ones.

Theorem 5.3.2 weak-NISZK = NISZK.

Armed with our complete problems, we then begin the work of comparing SZK and NISZK. First we show that the non-triviality of NISZK is equivalant to the non-triviality of SZK. This is shown by giving a Cook (or Turing) reduction from ED to EA.

Theorem 5.3.3 (non-triviality of NISZK) SZK $\neq$ BPP $\iff$ NISZK $\neq$ BPP.

In fact, it turns out that the type of Cook reduction we use is a special one, and by examining it further, we are able to shed more light on the SZK vs. NISZK question. Specifically, we observe that the reduction we give from ED to EA is an AC^0 *truth-table reduction.* That is, it is a non-adaptive Cook reduction in which the postprocessing is done in AC^0. (Formal definitions are given in Section 5.6.2.) Further, we can prove that if NISZK is closed under complement, then NISZK is closed under AC^0 truth-table reductions. Thus we deduce that NISZK being closed under complement implies that NISZK = SZK. In fact, we can show that closure under complement and a number of other natural conditions are equivalent to SZK = NISZK:

Theorem 5.3.4 (conditions for SZK = NISZK) *The following are equivalent:*

1. SZK = NISZK.
2. NISZK *is closed under complement.*
3. NISZK *is closed under* NC^1 *truth-table reductions.*
4. ED *(resp.,* SD*) Karp-reduces to* EA *(resp.,* SDU*). ("general versions reduce to one-sided ones")*
5. EA *(resp.,* SDU*) Karp-reduces to its complement. ("one-sided versions reduce to their complements")*

Theorem 5.3.4 can be interpreted as saying that if NISZK has a relatively weak closure property (closure under complement), then the class is surprisingly rich (equals SZK) and has a much stronger closure property (closure under NC^1 truth-table reductions.) At first, it might seem implausible that a class like NISZK with such an assymetric definition would be closed under complement. But SZK, which has a similarly assymetric definition, is known to be closed under complement [Oka96]. In light of this, the closure of NISZK under complement would not be quite as unexpected, and Theorem 5.3.4 illustrates that proving it would have wider consequences.

The last two conditions in Theorem 5.3.4 show that these questions about non-interactive versus interactive statistical zero-knowledge proofs are actually equivalent to basic questions about relationships between natural computational problems whose definitions have no *a priori* relationship to zero-knowledge proofs.

The equality of SZK and NISZK has interesting consequences not just for NISZK, but also for SZK. Currently, as we saw in Chapter 4 the best known generic protocol for SZK (against cheating verifiers, making no computational assumptions) requires a polynomial number of rounds.[6] For NISZK, however, by [DGW94], it is known that every problem in NISZK has a *constant round* statistical zero-knowledge proof system (against general, cheating verifiers) with inverse polynomial soundness error. Whether every problem in SZK has such a proof system is still an open question, which would be resolved in the positive if SZK = NISZK.

[6]Under the assumption that the Discrete Logarithm is hard, however, there is a constant round, cheating verifier SZK proof system with inverse polynomial soundness error for all of SZK [Oka96, BMO90].

5.3.1 A wider perspective

The study of non-interactive *statistical* (rather than *computational*) zero-knowledge proofs may be of interest for two reasons. Firstly, *statistical* zero-knowledge proofs provide an almost absolute level of security, whereas *computational* zero-knowledge proofs only provide security relative to computational abilities (and typically under complexity theoretic assumptions). Secondly, by analogy from the study of zero-knowledge *interactive* proofs, we believe that techniques developed for the "cleaner" statistical model can be applied or augmented to yield results for computational zero-knowledge: The proof that one-way functions are necessary for SZK to be non-trivial [Ost91] (a simpler proof was given in Chapter 3) was later generalized to CZK [OW93] (in Chapter 3 we saw a proof for public-coin computational zero knowledge). More recently, the transformations of honest-verifier zero knowledge to general zero knowledge, presented in [Dam94, DGW94, DGOW95] and Chapter 4 apply both to statistical and computational zero knowledge (whereas the original motivation was the study of statistical zero knowledge). It is our hope that the current study of NISZK will eventually lead to a better understanding of NICZK, where there are still important open questions such as the minimal conditions under which NP has NICZK proofs.

5.4 EA is in NISZK

In this section, we show that EA has a non-interactive statistical zero-knowledge proof system. The proof is given in Subsection 5.4.1, assuming a certain lemma (Lemma 5.4.1). Subsections 5.4.2 to 5.4.4 are devoted to the proof Lemma 5.4.1, which is technically somehwat involved. Therefore, the reader is encouraged to read only Subsection 5.4.1 from this section on first reading.

5.4.1 The proof system

Our aim in this section is the prove the following:

Lemma 5.4.1 EA $\in$ NISZK. *Moreover, there is a non-interactive statistical zero-knowledge proof system for* EA *in which the completeness error, soundness error, and simulator deviation are all exponentially vanishing (specifically* 2^{-s}*, where* s *is the length of the input).*

The transformation given by the following lemma (proved in Subsection 5.4.4) will be applied at the start of the proof system:

Lemma 5.4.1 *There is a polynomial-time computable function that takes an instance* (X, k) *of* EA *and a parameter* s *(in unary) and produces a distribution* Z *on* $\{0,1\}^{\ell}$ *(encoded by a circuit which samples from it) such that*

1. *If* $\mathrm{H}(X) > k + 1$*, then* Z *has statistical difference at most* 2^{-s} *from the uniform distribution on* $\{0,1\}^{\ell}$*, and*
2. *If* $\mathrm{H}(X) < k - 1$*, then the support of* Z *is at most a* 2^{-s} *fraction of* $\{0,1\}^{\ell}$.

Lemma 5.4.1 essentially transforms an instance of ENTROPY APPROXIMATION into an instance of IMAGE DENSITY, the complete problem of [DDPY98]. Given this transformation, it is straightforward to give a noninteractive statistical zero-knowledge proof system for EA:

Protocol 5.4.1 *Non-interactive proof system for* EA*, on input* (X, k)

1. *Let Z be the distribution on $\{0,1\}^\ell$ obtained from (X,k) as in Lemma 5.4.1 taking s to be the total description length of (X,k) in bits. Let $\sigma \in \{0,1\}^\ell$ be the reference string.*
2. *P selects r uniformly among $\{r' \colon Z(r') = \sigma\}$ and sends r to V.*
3. *V accept if $Z(r) = \sigma$ and rejects otherwise.*

It is immediate from Lemma 5.4.1 that the completeness error and soundness error of this proof system are 2^{-s}. For zero-knowledgeness, we consider the following probabilistic polynomial-time simulator:

Simulator for EA **proof system, on input** (X,k)

1. Let Z be obtained from (X,k) as in the proof system.
2. Select an input r to Z uniformly at random and let $\sigma = Z(r)$.
3. Output (σ, r).

It follows from Part 1 of Lemma 5.4.1 that this simulator has statistical difference at most 2^{-s} from the distribution of transcripts of (P,V). Thus, assuming Lemma 5.4.1, we have established Lemma 5.4.1. In fact, we need not require that s be the length of (X,k). Instead, s can be taken to be an arbitrary security parameter, and the completeness, soundness, and simulation error will be exponentially small in s, while the running time of the protocol only depends polynomially on s. We can use this to prove the following, which will be useful to us later.

Proposition 5.4.1 *If any promise problem* Π *reduces to* EA *by a Karp (i.e. many-one) reduction (even if it is length-reducing), then* $\Pi \in \mathsf{NISZK}$.

Proof: A noninteractive statistical zero-knowledge proof system for Π can be given as follows: On an instance x of Π, both parties compute the image (X,k) of x under the reduction $\Pi \le_{\mathtt{Karp}} \mathrm{EA}$ and execute the proof system for EA on (X,k), except that we take s to be the length of x. Hence, the completeness and soundness errors and simulator deviation of this proof system are exponentially small in $|x|$ (rather than $|(X,k)|$ which could be shorter than x). ■

5.4.2 Flat distributions and the Leftover Hash Lemma

Here we discuss some standard notions and techniques that will be useful in the proof of Lemma 5.4.1. We use the clean formulations of these tools given in [GV99]. A distribution X is called *flat* if all strings in the support of X have the same probability. Notice that if X is flat, then by the definition of entropy, $\Pr[X = x] = 2^{-\mathrm{H}(X)}$ for every x in the support of X. We quantify deviation from flatness as follows:

Definition 5.4.1 (heavy, light and typical elements): *Let X be a distribution, x an element possibly in its support, and Δ a positive real number. We say that x is* Δ-heavy (resp., Δ-light) *if* $\Pr[X = x] \ge 2^{\Delta} \cdot 2^{-\mathrm{H}(X)}$ (resp., $\Pr[X = x] \le 2^{-\Delta} \cdot 2^{-\mathrm{H}(X)}$). *Otherwise, we say that x is* Δ-typical.

A natural relaxed definition of flatness follows. The definition links the amount of slackness allowed in "typical" elements with the probability mass assigned to non-typical elements.

Definition 5.4.2 (flat distributions): *A distribution X is called Δ-*flat *if for every $t > 0$, the probability that an element chosen from X is $t \cdot \Delta$-typical is at least $1 - 2^{-t^2+1}$.*

By straightforward application of Hoefding Inequality, we have:

Lemma 5.4.2 (flattening lemma): *Let X be a distribution, k a positive integer, and $\otimes^k X$ denote the distribution composed of k independent copies of X. Suppose that for all x in the support of X it holds that $\Pr[X = x] \geq 2^{-m}$. Then $\otimes^k X$ is $\sqrt{k} \cdot m$-flat.*

Proof: For every x in the support of X, we let $w(x) = -\log \Pr[X = x]$. Then w maps the support of X, denoted D, to $[0, m]$. Let $X_1, ..., X_k$ be identical and independent copies of X. The lemma asserts that for every t

$$\Pr\left[\left|\sum_{i=1}^{k} w(X_i) - k \cdot \mathrm{H}(X)\right| > t \cdot m\sqrt{k}\right] < 2^{-t^2+1}$$

Observe that $E(w(X_i)) = \sum_x \Pr[X = x]\, w(x) = \mathrm{H}(X)$, for every i. Thus, the lemma follows by a straightforward application of Hoefding Inequality: Specifically, define random variables $\xi_i = w(X_i)$, let $\mu = E(\xi_i)$ and $\delta = tm/\sqrt{k}$, and use

$$\begin{aligned}\Pr\left[\left|\frac{\sum_{i=1}^{k} \xi_i}{k} - \mu\right| > \delta\right] &< 2 \cdot \exp\left(-\frac{2\delta^2}{m^2} \cdot k\right) \\ &= 2 \cdot \exp\left(-2t^2\right)\end{aligned}$$

The lemma follows. ■

The key point is that the entropy of $\otimes^k X$ grows linearly with k, whereas its deviation from flatness grows significantly slower (i.e., linear in $\sqrt{k}$) as a function of k. Note that if X is a distribution defined by a circuit with ℓ input gates, then $\Pr[X = x] \geq 2^{-\ell}$ for all x in the support of X, so the conclusion of Lemma 5.4.2 holds with $m = \ell$. The other main tool we will use is:

Lemma 5.4.2 (Leftover Hash Lemma [ILL89]) *Let $\mathcal{H}$ be a 2-universal family of hash functions mapping a domain D to a range R. Suppose X is a distribution on D such that with probability at least $1 - \delta$ over x selected from X, $\Pr[X = x] \leq \varepsilon/|R|$. Then the statistical difference between the following two distributions is at most $O(\delta + \varepsilon^{1/3})$:*

(A) Choose h uniformly from $\mathcal{H}$ and x according to X. Output $(h, h(x))$.

(B) Choose h uniformly from $\mathcal{H}$ and y uniformly from R. Output (h, y).

In particular, notice that if X is a Δ-flat distribution, then for any parameters $s, t > 0$, X satisfies the hypothesis of the Leftover Hash Lemma with $|R| = 2^{\mathrm{H}(X)-t\Delta-s}$, $\delta = 2^{-t^2+1}$, and $\varepsilon = 2^{-s}$. As we will be applying Lemma 5.4.2 to sets of strings, we define, for any pair of positive integers a and b, $\mathcal{H}_{a,b}$ to be one of the standard 2-universal families of hash functions mapping $\{0,1\}^a$ to $\{0,1\}^b$ (e.g., affine GF(2)-linear transformations).

5.4.3 Overview of Lemma 5.4.1

The transformation proceeds in four stages, which are roughly described below:

1. Let X' consist of many copies of X so that the entropy gap between YES and NO instances increases, and the distribution becomes quite flat relative to its entropy.

2. Hash X' so that YES instances become close to the uniform distribution while NO instances have much smaller entropy than the uniform distribution. That is, let Y be of the form $(h, h(X'))$, where h is uniformly distributed in a 2-universal family with appropriate parameters.

3. Let Y' consist of many copies of Y so that for NO instances, the entropy deficiency (as compared to the uniform distribution) becomes large and yet Y' becomes quite flat relative to its entropy; while YES instances remain close to uniform.

4. Hash the *inputs* to Y' so that NO instances have small support (rather than just small entropy), while keeping YES instances close to uniform. That is, let Z be of the form $(Y'(r), h, h(r))$ where h is uniformly distributed in a 2-universal family with appropriate parameters.

5.4.4 Proof of Lemma 5.4.1

Let (X, k) be an instance of EA, let m (resp., n) denote the number of input and output gates to X, and let s be the extra parameter in the transformation. By increasing s if necessary, we may assume that s is greater than the total description length of (X, k). Thus, all the intermediate circuits we build will be of size $\mathrm{poly}(s)$. Also note that it suffices for the transformation to achieve error parameters just $2^{-\Omega(s)}$ rather than 2^{-s}, as this can be compensated for by first increasing s by a linear factor.

Many copies I. The first step is to take many copies of each distribution; this has the effect of increasing the entropy gap between YES and NO instances relative to X's deviation from flatness. Namely, let $q = 4sm^2$ and let $X' = \otimes^q X$ (i.e., X' consists of q independent copies of X). Then $\mathrm{H}(X') = q \cdot \mathrm{H}(X)$ and, by Lemma 5.4.2, X' is Δ-flat for $\Delta = \sqrt{4sm^2} \cdot m = 2\sqrt{s} \cdot m^2$. In particular, we have established

Claim 5.4.1

1. *If* $\mathrm{H}(X) > k + 1$, *then* $\mathrm{H}(X') > qk + q \geq qk + \sqrt{s}\Delta + s$.
2. *If* $\mathrm{H}(X) < k - 1$, *then* $\mathrm{H}(X') < qk - q < qk$.

Hashing I. Now consider the distribution Y on pairs $(h, h(x))$ induced by choosing h uniformly from $\mathcal{H}_{qn,qk+1}$ and x according to X'. Say that elements of $\mathcal{H}_{qn,qk+1}$ take $u \leq \mathrm{poly}(qn, qk) \leq \mathrm{poly}(s)$ bits to represent. Then Y is represented by a circuit with inputs (resp., outputs) of length $m' = u + qm$ (resp., $n' = u + qk + 1$). Y has the following properties:

Claim 5.4.2

1. *If* $\mathrm{H}(X) > k+1$, *then* Y *has statistical difference at most* $2^{-\Omega(s)}$ *from the uniform distribution on* $\{0,1\}^{n'}$.
2. *If* $\mathrm{H}(X) < k - 1$, *then the entropy of* Y *is less than* $n' - 1$.

Proof: Part 1 follows from the Δ-flatness of X and the Leftover Hash Lemma. Part 2 follows from the fact that the entropy of Y is at most the entropy of X' (which is less than qk) plus the entropy of the uniform distribution on $\mathcal{H}_{qn,qk+1}$ (which is u). ■

Many copies II. We now take many copies of Y, so that the entropy deficiency of NO instances becomes large relative to the flatness while YES instances remain close to uniform. Specifically, let $q' = 4s \cdot (m')^2$ and let $Y' = \otimes^{q'} Y$, so that Y' has $M = m'q'$ input gates, $N = n'q'$ output gates, and Y' is Δ'-flat for $\Delta' = \sqrt{4s(m')^2} \cdot m' = 2\sqrt{s} \cdot (m')^2$. Then we immediately have:

Claim 5.4.3

1. *If* $\mathrm{H}(X) > k+1$, *then* Y' *has statistical difference at most* $q' \cdot 2^{-\Omega(s)} = 2^{-\Omega(s)}$ *from the uniform distribution on* $\{0,1\}^N$.

2. *If* $\mathrm{H}(X) < k-1$, *then* $\mathrm{H}(Y') < N - q' \leq N - \sqrt{3s} \cdot \Delta' - s$.

Hashing II. The final step is to make a distribution which, for NO instances, has small support (rather than just low entropy) in the case of NO instances, while YES instances remain close to uniform. Consider a circuit Z which takes as input $r \in \{0,1\}^M$ and a hash function $h \in \mathcal{H}_{M,M-N-s}$ and outputs $(Y'(r), h, h(r))$. Then,

Claim 5.4.4 *Z satisfies the requirements of Lemma 5.4.1 (with error parameters $2^{-\Omega(s)}$ rather than 2^{-s}). That is,*

1. *If* $\mathrm{H}(X) > k+1$, *then Z has statistical difference at most $2^{-\Omega(s)}$ from the uniform distribution on* $\{0,1\}^N \times \mathcal{H}_{M,M-N-s} \times \{0,1\}^{M-N-s}$, *and*

2. *If* $\mathrm{H}(X) < k-1$, *then the support of Z is at most a $2^{-\Omega(s)}$ fraction of* $\{0,1\}^N \times \mathcal{H}_{M,M-N-s} \times \{0,1\}^{M-N-s}$.

The intuition for this is the following: In the case of YES instances, Y' is close to the uniform distribution on $\{0,1\}^N$, so for almost all $y \in \{0,1\}^N$, there will be about 2^{M-N} values of r such that $Y'(r) = y$. Thus, hashing r down to $M - N - s$ bits will still result in a nearly uniform distribution.

In the case of a NO instance, Y' has large entropy deficiency and is nearly flat. From this, we can deduce that Y' lands in some small subset T of $\{0,1\}^N$ with very high probability. Thus, points $y \notin T$ must have very low probability under Y', i.e. there are very few inputs r such that $Y'(r) = y$. So, for each $y \notin T$, the pairs $(h, h(r))$ will only hit a small subset of the possible values. Therefore, $(Y'(r), h, h(r))$ has small support, because either the first component lands in a small set (namely T) or the last two components land in a small set.

Proof: Suppose $\mathrm{H}(X) > k+1$. From the fact that Y' has statistical difference at most $2^{-\Omega(s)}$ from uniform it follows that with probability at least $1 - 2^{-\Omega(s)}$ over y selected according to Y',

$$\Pr\left[Y' = y\right] \geq \frac{1}{2} \cdot \frac{1}{2^N}. \tag{5.1}$$

Fix any y satisfying Inequality 5.1. Conditioned on $Y'(r) = y$, r is selected uniformly from $\{r : Y'(r) = y\}$, which by Equation 5.1 is a set of size at least 2^{M-N-1}. Thus, by the Leftover Hash Lemma, conditioned on $Y'(r) = y$, the distribution of $(h, h(r))$ has statistical difference at most $2^{-\Omega(s)}$ from uniform. Therefore the total statistical difference of Z from uniform is $2^{-\Omega(s)}$.

Now suppose $\mathrm{H}(X) < k-1$. We want to show that the support S of Z is a small fraction of $D = \{0,1\}^N \times \mathcal{H}_{M,M-N-s} \times \{0,1\}^{M-N-s}$. To do this, we divide S into three parts, depending on the probability mass given to the y component by Y'. Recall that a "typical" y for Y' has probability mass $\approx 2^{-\mathrm{H}(Y')} \geq 2^{-N+\sqrt{3s}\cdot\Delta'+s}$.

$$\begin{array}{lcll} S_1 & = & \{(y,h,z) \in S\colon \Pr[Y'=y] \leq 2^{-N-2s}\} & \text{("much too light")} \\ S_2 & = & \{(y,h,z) \in S\colon 2^{-N-2s} < \Pr[Y'=y] \leq 2^{-N+s}\} & \text{("too light, but not much too light")} \\ S_3 & = & \{(y,h,z) \in S\colon 2^{-N+s} < \Pr[Y'=y]\} & \text{("not too light")} \end{array}$$

Clearly, $S = S_1 \cup S_2 \cup S_3$. We will show that $|S_i|/|D| \leq 2^{-s}$ for $i = 1,2,3$, and so $|S|/|D| \leq 3 \cdot 2^{-s} = 2^{-\Omega(s)}$.

First we bound $|S_1|$. For any y such that $\Pr[Y'=y] \leq 2^{-N-2s}$, there are at most 2^{M-N-2s} values of r such that $Y'(r) = y$. Thus, for any such y and any h, the set of z such that $(y,h,z) \in S_1$ is of size at most 2^{M-N-2s} (because each such z must be of the form $h(r)$ for some r such that $Y'(r) = y$). This implies that S_1 is at most a $2^{M-N-2s}/2^{M-N-s} = 2^{-s}$ fraction of D.

Now we bound $|S_2|$. We show that the set A of y such that $2^{-N-2s} < \Pr[Y'=y] \leq 2^{-N+s}$ is at most a 2^{-s} fraction of $\{0,1\}^N$. From this, it follows that S_2 is at most a 2^{-s} fraction of D. Every $y \in A$ is $\sqrt{3s}\cdot\Delta'$-light (since Y' has entropy at most $N - s - \sqrt{3s}\cdot\Delta'$). By the Δ'-flatness of Y', $\Pr[Y' \in A]$ is at most 2^{-3s+1}. Since every y in A has probability mass at least 2^{-N-2s} under Y', $|A|$ is at most $2^{-3s+1}/2^{-N-2s} < 2^{N-s}$, as desired.

Finally, we bound $|S_3|$. Clearly, there can be at most 2^{N-s} values of y such that $\Pr[Y'=y] \geq 2^{-N+s}$. From this it follows that $|S_3|/|D| \leq 2^{N-s}/2^N = 2^{-s}$. ■

5.5 EA and SDU are NISZK-complete

In this section, we complete the proof of Theorem 5.3.1. First, we establish that SDU $\in$ NISZK by showing:

Lemma 5.5.1 SDU$\leq_{\text{Karp}}$EA. *In particular,* SDU $\in$ NISZK.

Proof: Let X be an instance of SDU. We assume that $\log(n) > 5$, where n is the output length of the circuit X (otherwise, once can decide in probabilistic polynomial time whether X is a YES or NO instance of SDU by random sampling). Let U denote the uniform distribution on n bits. We claim the map $X \mapsto (X, n-3)$ is the reduction required by the lemma.

If $X \in \text{SDU}_{\text{YES}}$, then $\delta = \|X - U\| < 1/n$. Now we use the following fact:

Fact 5.5.1 *For any two random variables, A and B, ranging over a domain D it holds that*

$$|\mathrm{H}(A) - \mathrm{H}(B)| \;\leq\; \log(|D|-1)\cdot\delta \;+\; \mathrm{H}_2(\delta)$$

where $\delta \stackrel{\text{def}}{=} \|A - B\|$, and where $\mathrm{H}_2(\theta)$ denotes the entropy of a 0–1 random variable with mean θ.

This fact can be inferred from Fano's Inequality (cf., [CT91, Thm. 2.11.1]), and a slightly weaker bound can be found in [CT91, Thm. 16.3.2]. A more direct proof follows.

Proof: Assume $\delta > 0$ or else the claim is obvious. Let $p(x) \stackrel{\text{def}}{=} \Pr[A = x]$ and $q(x) \stackrel{\text{def}}{=} \Pr[B = x]$. Define $m(x) \stackrel{\text{def}}{=} \min\{p(x), q(x)\}$. Then $\sum_{x\in D} m(x) = 1 - \delta$. Define random variables Z', X' and Y' so that

$$\Pr[Z' = x] \;=\; m'(x) \stackrel{\text{def}}{=} \frac{1}{1-\delta}\cdot m(x)$$

$$\Pr[X' = x] = p'(x) \stackrel{\text{def}}{=} \frac{1}{\delta} \cdot (p(x) - m(x))$$
$$\Pr[Y' = x] = q'(x) \stackrel{\text{def}}{=} \frac{1}{\delta} \cdot (q(x) - m(x))$$

Think of A (resp., B) as being generated by picking Z' with probability $1 - \delta$ and X' (resp., Y') otherwise. Then

$$\mathrm{H}(A) \leq (1-\delta) \cdot \mathrm{H}(Z') + \delta \cdot \mathrm{H}(X') + \mathrm{H}_2(\delta)$$
$$\mathrm{H}(B) \geq (1-\delta) \cdot \mathrm{H}(Z')$$

Observing that $\Pr[X' = x] = 0$ on at least one $x \in D$, it follows that $\mathrm{H}(X') \leq \log(|D| - 1)$, and the fact follows. ■

Comment: The above bound is tight. Let $e \in D$ and consider A which is identically e, and B which with probability $1 - \delta$ equals e and otherwise is uniform over $D \setminus \{e\}$. Clearly, $\|A - B\| = \delta$ and $\mathrm{H}(A) - \mathrm{H}(B) = \delta \log(|D| - 1) + \mathrm{H}_2(\delta) - 0$.

Returning to our proof, if we apply this fact with $A = U$ and $B = X$, we have

$$n - \mathrm{H}(X) < n \cdot 1/n + \mathrm{H}_2(1/n) < 2.$$

Hence $(X, n-3) \in \mathrm{EA}_{\mathrm{YES}}$, as desired.

If $X \in \mathrm{SDU}_{\mathrm{NO}}$, then $\|X - U\| \geq 1 - 1/n$. By the definiton of statistical difference, this implies the existence of a set $S \subset \{0,1\}^n$ such that $\Pr[X \in S] - \Pr[U \in S] > 1 - 1/n$. This implies that

$$\Pr[X \in S] > 1 - 1/n \quad \text{and} \quad \Pr[U \in S] < 1/n.$$

Thus, $\mathrm{H}(X) \leq \Pr[X \in S] \cdot \log(|S|) + \Pr[X \notin S] \cdot n < 1 \cdot (n - \log n) + (1/n) \cdot n < n - 4$, and we have that $(X, n-3) \in \mathrm{EA}_{\mathrm{NO}}$.

The "in particular" part of Lemma 5.5.1 follows immediately from Proposition 5.4.1. ■

Now, we establish both Theorem 5.3.1 and Theorem 5.3.2 by showing that all promise problems in weak-NISZK (and hence all promise problems in NISZK) are reducible to SDU (and hence by the previous lemma to EA).

Lemma 5.5.2 *Every promise problem in* weak-NISZK *Karp-reduces to* SDU.

Proof: Let Π be any promise problem in weak-NISZK. As weak-NISZK is preserved under parallel repetition, we may assume that Π has a weak-NISZK proof system (P, V) with completeness and soundness errors at most 2^{-n} on inputs of length n. Let $r(n) = \mathrm{poly}(n)$ be the length of the random reference string in (P, V), and let S be a randomized polynomial-time simulator S such that the statistical difference between the output distribution of S and the distribution of true transcripts of P is at most $1/(3r(n))$. (Such an S is guaranteed by the weak-NISZK property.) Let U denote the uniform distribution on $r(n)$ bits.

Let x be an instance of Π. Define M_x to be a circuit which does the following on input s:
$M_x(s)$: Simulate $S(x)$ with randomness s to obtain a transcript (σ, p). If $V(x, \sigma, p)$ accepts, then output σ, else output $0^{r(n)}$.

We claim that the map $x \mapsto M_x$ is the reduction required by the lemma. Suppose $x \in \Pi_{\text{YES}}$. In this case, we know that the random reference string σ in the output of S has statistical difference less than $1/3r(n)$ from U. In addition, since the completeness error of protocol P is at most 2^{-n}, $S(x)$ can output rejecting transcripts with probability at most $1/(3r(n)) + 2^{-n} \leq 2/(3r(n))$. Hence, $\|M_x - U\| < 2/(3r(n)) + 1/(3r(n)) \leq 1/r(n)$, and $M_x \in \text{SDU}_{\text{YES}}$.

Suppose $x \in \Pi_{\text{NO}}$. Since the soundness error of protocol P is bounded by 2^{-n}, for at most a 2^{-n} fraction of reference strings σ does there exist an accepting transcript (σ, p). Since M_x only outputs reference strings corresponding to accepting transcripts or $0^{r(n)}$, $\|M_x - U\| \geq 1-(2^{-n}+2^{-r(n)}) > 1 - 1/r(n)$. Thus, $M_x \in \text{SDU}_{\text{NO}}$. ■

Clearly, Lemmas 5.4.1, 5.5.1, and 5.5.2 combine to prove Theorem 5.3.1. Lemmas 5.5.2 and 5.5.1 show that any promise problem Π in weak-NISZK reduces to EA; by Proposition 5.4.1, this implies that $\Pi \in$ NISZK and establishes Theorem 5.3.2.

5.6 Comparing NISZK and SZK

Armed with NISZK-complete promise problems so closely related to problems known to be complete for SZK, we can quickly begin relating the two classes.

5.6.1 Nontriviality of NISZK

First, we establish Theorem 5.3.3 by giving a Cook reduction from ENTROPY DIFFERENCE (ED), complete for SZK, to ENTROPY APPROXIMATION (EA), complete for NISZK.

Lemma 5.6.1 *Suppose (X,Y) is an instance of* ED. *Let $X' = \otimes^4 X$ (resp., $Y' = \otimes^4 Y$) consist of 4 independent copies of X (resp., Y), and let n denote the maximum of the output sizes of X' and Y'. Then,*

$$(X,Y) \in \text{ED}_{\text{YES}} \implies \bigvee_{k=1}^{n} \left[\left((X',k) \in \text{EA}_{\text{YES}}\right) \wedge \left((Y',k) \in \text{EA}_{\text{NO}}\right)\right]$$

$$(X,Y) \in \text{ED}_{\text{NO}} \implies \bigwedge_{k=1}^{n} \left[\left((X',k) \in \text{EA}_{\text{NO}}\right) \vee \left((Y',k) \in \text{EA}_{\text{YES}}\right)\right]$$

Proof: Suppose $(X,Y) \in \text{ED}_{\text{YES}}$, so that $H(X') > H(Y') + 4$. Let $k = \lfloor H(X') \rfloor - 2$. Then $H(X') > k + 1$. On the other hand, $k + 3 > H(X') > H(Y') + 4$, and hence $H(Y') < k - 1$. Suppose instead $(X,Y) \in \text{ED}_{\text{NO}}$, so that $H(Y') > H(X') + 4$. Then for all $k > \lceil H(X') \rceil + 1$, we have $H(X') < k - 1$. So, for all $k \leq \lceil H(X') \rceil + 1$, we have $k + 1 < H(X') + 3 < H(Y')$. ■

From this reduction, we conclude that SZK $\neq$ BPP $\iff$ NISZK $\neq$ BPP, which is Theorem 5.3.3. Again, by BPP we mean the class of *promise problems* solvable in probabilistic polynomial time.

Proof of Theorem 5.3.3. By definition, NISZK $\subset$ SZK (recall that SZK equals honest-verifier SZK [GSV98]). Hence if SZK = BPP, then NISZK = BPP.

Now suppose $\mathsf{NISZK} = \mathsf{BPP}$, so in particular there is a probabilistic polynomial-time machine M which decides EA (with exponentially small error probability). To show $\mathsf{SZK} = \mathsf{BPP}$, it suffices to show that ED $\in$ BPP since ED is SZK-complete. We now describe how to decide instances of ED: Let (X, Y) be an instance of ED. Letting X' and Y' be as stated in Lemma 5.6.1, we run $M(X', k)$ and $M(Y', k)$ for all $k \in [1, n]$. If for some k, we see that $M(X', k) = 1$ and $M(Y', k) = 0$, we output 1. Otherwise, we output 0. By Lemma 5.6.1, this is a correct BPP algorithm for deciding ED. ■.

5.6.2 Conditions under which NISZK = SZK

The reduction given by Lemma 5.6.1 is a very special type of Cook reduction, which we call an AC^0 truth-table reduction. In this section, we use the special properties of this reduction to show that if NISZK is closed under complement, then in fact $\mathsf{NISZK} = \mathsf{SZK}$. We now recall how to precisely define the types of reductions we are using, taking care how they are defined for promise problems.

Definition 5.6.1 (truth-table reduction [LLS75]): *We say a promise problem* Π truth-table reduces *to a promise problem* Γ*, written* $\Pi \leq_{\mathrm{tt}} \Gamma$*, if there exists a (deterministic) polynomial-time computable function* f*, which on input* x *produces a tuple* $(x_1, x_2, \ldots, x_k)$ *and a circuit* C*, such that*

1. *If* $x \in \Pi_{\mathrm{YES}}$ *then for all valid settings of* $b_1, b_2, \ldots, b_k$, $C(b_1, b_2, \ldots, b_k) = 1$, *and*
2. *If* $x \in \Pi_{\mathrm{NO}}$ *then for all valid settings of* $b_1, b_2, \ldots, b_k$, $C(b_1, b_2, \ldots, b_k) = 0$.

where a setting for b_i *is considered valid when* $b_i = 1$ *if* $x_i \in \Gamma_{\mathrm{YES}}$ *and* $b_i = 0$ *if* $x_i \in \Gamma_{\mathrm{NO}}$ *(and* b_i *is unrestricted when* x_i *violates the promise).*

In other words, a truth-table reduction for promise problems is a non-adaptive Cook reduction which is allowed to make queries which violate the promise, but must be able to tolerate both yes and no answers in response to queries that violate the promise. We further consider the case where we restrict the complexity of computing the output of the reduction from the queries:

Definition 5.6.2 (AC^0 and NC^1 truth-table reductions): *A truth-table reduction* f *between promise problems is an* AC^0 (resp., NC^1) truth-table reduction *if the circuit* C *produced by the reduction on input* x *has depth bounded by a constant* c_f *independent of* x *(resp., has depth bounded by* $c_f \log |x|$*). If there is an* AC^0 *(resp.,* NC^1*) truth-table reduction from* Π *to* Γ*, we write* $\Pi \leq_{\mathsf{AC}^0\text{-}\mathrm{tt}} \Gamma$ *(resp.,* $\Pi \leq_{\mathsf{NC}^1\text{-}\mathrm{tt}} \Gamma$*).*

With this definition, we observe that Lemma 5.6.1 in fact gives an AC^0 truth-table reduction, since the formula given in the lemma can be expressed as an AC^0 circuit, and the statement of the lemma shows that the reduction has the robustness properties against promise violations that are required in Definition 5.6.2. Thus, we have:

Proposition 5.6.1 $\mathrm{ED} \leq_{\mathsf{AC}^0\text{-}\mathrm{tt}} \mathrm{EA}$.

We say that a class $\mathcal{C}$ of promise problems is closed under a class of reductions $\leq_*$ if $\Pi \leq_* \Gamma$ and $\Gamma \in \mathcal{C}$ implies that $\Pi \in \mathcal{C}$. By the above, if NISZK is closed under AC^0 truth-table reductions, then ED $\in$ NISZK and hence $\mathsf{NISZK} = \mathsf{SZK}$. Thus, we would like to capture the minimal conditions necessary for a promise class to be closed under AC^0 truth-table reductions. Here, care must be taken to because of the possibility of promise violations. Keeping this in mind, we define the following operator on promise problems to capture the notion of an unbounded fan-in AND gate for promise problems:

Definition 5.6.3 *(unbounded AND): For any promise problem* Π, *we define* $\mathrm{AND}(\Pi)$ *to be the promise problem:*

$$\begin{aligned}\mathrm{AND}_{\mathrm{YES}}(\Pi) &\stackrel{\mathrm{def}}{=} \{(x_1, x_2, \ldots, x_k) : k \geq 0, \forall i \in [1,k] x_i \in \Pi_{\mathrm{YES}}\} \\ \mathrm{AND}_{\mathrm{NO}}(\Pi) &\stackrel{\mathrm{def}}{=} \{(x_1, x_2, \ldots, x_k) : k \geq 0, \exists i \in [1,k] x_i \in \Pi_{\mathrm{NO}}\}\end{aligned}$$

We say a class of promise problems C *is* closed under unbounded AND *if* $\Pi \in C$ *implies that* $\mathrm{AND}(\Pi) \in C$.

We have defined AND so that it has the weakest promise condition possible to remain well-defined. In particular, we see that $\mathrm{AND}_{\mathrm{NO}}(\Pi)$ is defined to include x_i's that violate Π's promise, as long as just *one* of them is in Π_{NO}. We also need a way of combining two promise problems:

Definition 5.6.4 *(disjoint union): For any pair of promise problems* Π *and* Γ, *we define the* disjoint union *of* Π *and* Γ *to be the promise problem* $\textsc{DisjointUnion}(\Pi, \Gamma)$ *defined as follows:*

$$\begin{aligned}\textsc{DisjointUnion}_{\mathrm{YES}}(\Pi, \Gamma) &\stackrel{\mathrm{def}}{=} \{0\} \times \Pi_{\mathrm{YES}} \cup \{1\} \times \Gamma_{\mathrm{YES}} \\ \textsc{DisjointUnion}_{\mathrm{NO}}(\Pi, \Gamma) &\stackrel{\mathrm{def}}{=} \{0\} \times \Pi_{\mathrm{NO}} \cup \{1\} \times \Gamma_{\mathrm{NO}}\end{aligned}$$

We say a class of promise problems C *is* closed under disjoint union *if* $\Pi, \Gamma \in C$ *implies that* $\textsc{DisjointUnion}(\Pi, \Gamma) \in C$.

With these definitions, we can give the following lemma which gives some conditions sufficient to give closure under AC^0 truth-table reductions.

Lemma 5.6.2 *A promise class* C *is closed under* AC^0 *truth-table reductions if the following conditions hold:*

1. *C is closed under Karp (i.e., many-one) reductions.*
2. *C is closed under unbounded AND.*
3. *C is closed under disjoint union.*
4. *C is closed under complementation.*

Proof: First note that any unbounded fan-in circuit can be efficiently converted into a circuit with only unbounded fan-in NAND gates (allowing also unary NAND gates), with only a constant factor blowup in depth. So, as a first step, we observe that C is closed under unbounded NAND: for any promise problem Π, $\mathrm{NAND}(\Pi) \stackrel{\mathrm{def}}{=} \overline{\mathrm{AND}(\Pi)} \in C$, by closure under unbounded AND and complementation. To generalize this to constant depth circuits with unbounded fan-in NAND gates, we first need a definition.

Definition 5.6.5 *For any promise problem* Π, *and for all natural numbers* $d \geq 0$ *we define* $\textsc{Depth}^d(\Pi)$ *to be the promise problem whose instances are tuples* $(C, (x_1, x_2, \ldots, x_k))$, *where* C *is a circuit of depth at most* d *(using unbounded fan-in NAND gates only). The* YES *instances are those such that for all valid settings of* $b_1, b_2, \ldots, b_k$, $C(b_1, b_2, \ldots, b_m) = 1$; *whereas the* NO *instances are those tuples such that for all valid settings of* $b_1, b_2, \ldots, b_k$, $C(b_1, b_2, \ldots, b_k) = 0$. *Here, a setting for* b_i *is considered valid when* $b_i = 1$ *if* $x_i \in \Pi_{\mathrm{YES}}$ *and* $b_i = 0$ *if* $x_i \in \Pi_{\mathrm{NO}}$ *(and* b_i *is unrestricted when* x_i *violates the promise).*

Using the fact that every AC^0 circuit can be efficiently transformed into one with only NAND gates, we see that $\Pi \leq_{\mathsf{AC}^0-\mathrm{tt}} \Gamma$ means that there exists some d such that $\Pi \leq_{\mathrm{Karp}} \textsc{Depth}^d(\Gamma)$ under a Karp reduction. Hence if we can show that for all $d \geq 0$ and promise problems Π, $\textsc{Depth}^d(\Pi) \in \mathcal{C}$, the lemma will be established. We will prove this by induction on d.

First, observe that a depth 0 circuit is simply a variable (negations of variables are achieved with one unary NAND gate, so count as depth 1). Hence, $\textsc{Depth}^0(\Pi) \leq_{\mathrm{Karp}} \Pi \in \mathcal{C}$. Now assume that $\textsc{Depth}^d(\Pi) \in \mathcal{C}$. Observe that a depth $d+1$ circuit is simply a NAND of some number of depth d circuits. Using this observation, we will argue that that

$$\textsc{Depth}^{d+1}(\Pi) \leq_{\mathrm{Karp}} \textsc{DisjointUnion}(\textsc{Depth}^d(\Pi), \mathrm{NAND}(\textsc{Depth}^d(\Pi))).$$

By the hypothesized closure properties of $\mathcal{C}$, this implies that $\textsc{Depth}^{d+1}(\Pi) \in \mathcal{C}$. The reduction works as follows. The input to the reduction is a tuple $(C, \vec{x})$ where $\vec{x} = (x_1, x_2, \ldots x_k)$. If C is actually a depth d circuit, then it simply outputs $(0, (C, \vec{x}))$. If not, then it extracts from C the circuits $C_1, C_2, \ldots, C_s$ that provide input to the topmost NAND gate. Then the reduction outputs $(1, ((C_1, \vec{x}), (C_2, \vec{x}), \ldots, (C_s, \vec{x})))$. It is clear that map gives a Karp reduction from $\textsc{Depth}^{d+1}(\Pi)$ to $\textsc{DisjointUnion}(\textsc{Depth}^d(\Pi), \mathrm{NAND}(\textsc{Depth}^d(\Pi)))$, completing the induction step and the proof. ■

Which of the conditions of Lemma 5.6.2 does NISZK satisfy? We argue that Conditions 1, 2, and 3 are satisfied by NISZK:

Lemma 5.6.3 NISZK *is closed under Karp reductions.*

Proof: Suppose $\Gamma \in \mathsf{NISZK}$, and $\Pi \leq_{\mathrm{Karp}} \Gamma$. Since EA is complete for NISZK, we have $\Gamma \leq_{\mathrm{Karp}} \mathrm{EA}$. By composing reductions, we see that $\Pi \leq_{\mathrm{Karp}} \mathrm{EA}$. By Proposition 5.4.1, $\Pi \in \mathsf{NISZK}$. ■

Lemma 5.6.4 NISZK *is closed under unbounded AND.*

Proof: First, we argue that $\mathrm{AND}(\mathrm{EA}) \in \mathsf{NISZK}$ by describing a NISZK proof system for $\mathrm{AND}(\mathrm{EA})$: Let $((X_1, k_1), \ldots, (X_m, k_m))$ be an instance of $\mathrm{AND}(\mathrm{EA})$, and say ℓ is the total length of the instance. Artificially pad each circuit X_i to be of description size ℓ (by adding unused gates) and let Y_i be the resulting circuit. Now execute the NISZK proof system for EA given by Lemma 5.4.1 on each pair (Y_i, k_i) in parallel, and have the $\mathrm{AND}(\mathrm{EA})$-verifier accept if the EA-verifier would have accepted on each pair.

If every pair (X_i, k_i) is a YES instance of EA, the $\mathrm{AND}(\mathrm{EA})$ verifier will accept with probability at least $1 - m \cdot 2^{-\ell} = 1 - 2^{-\Omega(\ell)}$, as the completeness error of the EA proof system is at most $2^{-\ell}$. Similarly, running the simulator for the EA proof system m times independently will give a simulation for the $\mathrm{AND}(\mathrm{EA})$ proof system with simulator deviation at most $m \cdot 2^{-\ell} = 2^{-\Omega(\ell)}$. Finally, if just one pair (X_i, k_i) is a NO instance of EA (even if the others violate the promise), the verifier will accept with probability at most $2^{-\ell}$ in the i'th execution of the EA protocol, and so the $\mathrm{AND}(\mathrm{EA})$ verifier will accept with probability at most $2^{-\ell}$.

This shows that $\mathrm{AND}(\mathrm{EA}) \in \mathsf{NISZK}$. Now let Π be any promise problem in NISZK. Since EA is complete for NISZK, there is a Karp reduction f from Π to EA. This induces a Karp reduction from $\mathrm{AND}(\Pi)$ to $\mathrm{AND}(\mathrm{EA})$ in the obvious way (i.e. $(x_1, \ldots, x_k) \mapsto (f(x_1), \ldots, f(x_k))$). As $\mathrm{AND}(\mathrm{EA})$ is in NISZK and NISZK is closed under Karp reductions, $\mathrm{AND}(\Pi) \in \mathsf{NISZK}$. ■

Lemma 5.6.5 NISZK *is closed under disjoint union.*

Proof: For any two promise problem Π and Γ in NISZK, the Karp reductions f_0 from Π to EA and f_1 from Γ to EA induce a Karp reduction from DISJOINTUNION(Π, Γ) to EA given by $(\sigma, x) \mapsto f_\sigma(x)$. By Proposition 5.4.1, DISJOINTUNION$(\Pi, \Gamma) \in$ NISZK. ■

Combining everything, we can give a condition under which SZK = NISZK.

Proposition 5.6.2 *If* NISZK *is closed under complementation, then* SZK = NISZK.

Proof: Suppose NISZK is closed under complementation. Combining this with Lemmas 5.6.2, 5.6.3, 5.6.4, and 5.6.5, it follows that NISZK is closed under AC^0 truth-table reductions. Applying Proposition 5.6.1 ($\mathrm{ED} \leq_{\mathsf{AC}^0\text{-}\mathtt{tt}} \mathrm{EA}$) and Lemma 5.4.1 (EA $\in$ NISZK), we conclude that ED $\in$ NISZK. Since ED is complete for SZK [GV99] and NISZK is closed under Karp reductions (Lemma 5.6.3), we have SZK $\subset$ NISZK. As NISZK $\subset$ SZK is true from the definition of NISZK, we conclude that NISZK = SZK. ■

Finally, we deduce Theorem 5.3.4, which gives a number of conditions equivalent to NISZK = SZK.

Proof of Theorem 5.3.4:
1 $\Rightarrow$ 3. This follows from the result of Chapter 3 that SZK is closed under NC^1 truth-table reductions (Corollary 3.3.5).
3 $\Rightarrow$ 2 $\Rightarrow$ 1. The first is trivial and the second is Proposition 5.6.2.
1 $\Leftrightarrow$ 4. This follows from Theorem 5.3.1 (which asserts that that EA and SDU are complete for NISZK), the fact that ED and SD are complete for SZK (shown in Chapter 3 (the Completeness Theorem) and [GV99]) and Lemma 5.6.3 (that NISZK is closed under Karp reductions).
2 $\Leftrightarrow$ 5. This follows from Theorem 5.3.1 (that EA and SDU are complete for NISZK) and Lemma 5.6.3 (that NISZK is closed under Karp reductions).

Chapter 6

Concurrent Zero Knowledge

In this chapter, we turn our attention to extending the cryptographic uses of zero-knowledge proofs in general. Zero-knowledge proofs were designed and defined to provide provable security for a single pair of interacting parties. In general multi-user environments, however, where many interactions can take place concurrently, one must face the challenge of coordinated multi-party attacks. The standard 2-party definition of zero knowledge does not necessarily guarantee security in this scenario. In this chapter, we define and consider the notion of *concurrent zero knowledge*, where zero knowledge is guaranteed even when faced with a coordinated attack by many verifiers all acting concurrently. We show how to build concurrent zero-knowledge protocols in which honest parties need only act locally, i.e., an honest prover and verifier need not even be aware of other parties in order to be guaranteed security.

A critical novel component of our approach is an explicit use of certain local timing constraints in our protocols. The correctness of our protocols relies on a weak synchronization assumption on the behavior of the local clocks of honest parties. This assumption holds in particular if we assume that clocks of honest parties run at rates within constant factors of each other. The work we present in this chapter is based on a paper [DNS98] authored jointly with Cynthia Dwork and Moni Naor.

6.1 Overview

In a distributed computing aggregate, a collection of physically separated parties communicate via a heterogeneous network. To date, research applications of cryptographic techniques to distributed systems have overwhelmingly concentrated on the paradigm in which the system consists of n mutually aware parties trying to cooperatively compute a function of their respective inputs (see *e.g.* [BGW88, GMW87]). In contrast with this traditional paradigm, parties in an aggregate in general do not know of all the other members, nor do they generally know the topology of the network. The parties are typically *not* all acting cooperatively to compute one function or perform a specific set of tasks; in general no coordination is assumed. A prime example of an aggregate is the Internet.

Electronic interactions over an aggregate, such as economic transactions, transmission of medical data, data storage, and telecommuting, pose security risks inadequately addressed in computer science research. In particular, the issue of the security of *concurrent* executions is often ignored. In this chapter we address the problem of maintaining zero knowledge under concurrency, continuing research initiated in [DDN91] on zero-knowledge interactions in an aggregate.

6.1.1 Zero-Knowledge and Concurrency: A Description of the Problem

A zero-knowledge protocol is supposed to ensure that no information is leaked during its execution. In this section, we try to understand what goes wrong when we try to prove that standard zero-knowledge proofs remain zero knowledge in the concurrent scenario.

For purposes of illustration, we will focus on a well-known NP-complete [Kar72] problem called DIRECTED HAMILTONIAN CYCLE:

DIRECTED HAMILTONIAN CYCLE (DHC):
Instance: A directed graph G.
Question: Does there exist a Hamiltonian cycle in G, *i.e.* a cycle containing every node in G exactly once?

We will first consider a relatively simple 3-round zero-knowledge interactive proof for DHC, due to Blum [Bl]. This zero-knowledge proof, like all zero-knowledge proofs, can be composed *sequentially* as many times as one likes and remain zero knowledge. Another type of composition is *parallel* composition, where a protocol is repeated independently many times, but all of them in parallel with each other. Note that parallel composition is a special case of concurrent composition. We will see that the basic protocol for DHC does not remain zero knowledge under parallel composition. The problem of parallel composition, however, has been studied extensively. We will next give a modified protocol due to [GMW91, GKa96] which does indeed remain zero-knowledge under parallel composition. Nevertheless, we will show that a straightforward extension of the simulation strategy for this protocol to the scenario with concurrency fails for a particular interleaving of concurrent interactions. This will illustrate the difficulties involved in achieving concurrent zero knowledge.

String Commitment The notion of a *commitment scheme* will be a crucial ingredient in the protocol we present for DHC, and in all of our protocols as well. We will give formal definitions of various types of commitment schemes in Section 6.2. Informally speaking, a commitment scheme is used to enable a party to commit itself to a value while keeping it secret. Later on, a commitment may be "opened" to reveal the value – and it should be guaranteed that a given commitment can only be "opened" to a single value.

In other words, loosely speaking, a bit commitment scheme is a protocol between two parties – a sender S and a receiver R – that has two phases: a "commitment phase," in which S commits to a value; and an "opening phase," where S reveals to R the value that it committed to. This protocol must enjoy two basic properties:

- *Secrecy:* After the "commitment phase," the receiver R should not gain any knowledge about the committed value, even if R tries to cheat.
- *Binding:* After a successful "commitment phase," there can be at most one value that S can successfully open the commitment as in the "opening phase," even if S tries to cheat.

We will explore many variations on the notion of a commitment scheme when we define them formally, but for now this intuition will suffice.

6.1.2 Basic protocol for DHC.

Manuel Blum [Blu86] exhibited an elegant computational zero-knowledge proof system for DHC. It consists of many sequential repetitions of the following basic protocol:

Protocol 6.1.1 *A directed graph G is the common input to both parties. The prover is assumed to know a Hamiltonian cycle w in G.*

1. *The prover picks a random permutation π on the nodes of G. It then sends to the verifier commitments to each of the entries of the adjacency matrix of the permuted graph $\pi(G)$.*
2. *The verifier responds with a single bit b, chosen uniformly from $\{0, 1\}$.*
3. *The prover does the following:*
   - *If $b = 0$, the prover opens its commitments from Step 1 to all the entries of the adjacency matrix of $\pi(G)$, and also sends a description of the permutation π. The verifier, upon receipt, verifies the correctness of the openings, and also verifies that the revealed adjacency matrix actually is the adjacency matrix of $\pi(G)$. If either of these conditions fail to hold, the verifier rejects the proof.*
   - *If $b = 1$, the prover opens its commitments only to those entries in the adjacency matrix of $\pi(G)$ that correspond to edges in the Hamiltonian cycle $\pi(w)$. The verifier, upon receipt, verifies the correctness of the openings, and also verifies that the revealed entries of $\pi(G)$ do indeed form a Hamiltonian cycle. If either of these conditions fail to hold, the verifier rejects the proof.*

Completeness: Clearly, if the graph G is indeed Hamiltonian, then the prover by following the protocol will always convince the verifier to accept.

Soundness: On the other hand, if G has no Hamiltonian cycle, we claim that the verifier will reject with probability at least $1/2$ no matter what the prover tries to do (*i.e.* the soundness error is $1/2$).

In Step 1, the prover commits to the adjacency matrix of some graph H (or one or more commitments are invalid). There are two cases: either H is isomorphic to G, or it is not (here we include the case that one or more commitments are invalid):
Case 1: H is isomorphic to G. Then H cannot contain a Hamiltonian cycle. Therefore, if $b = 1$, the prover will not be able to respond in Step 3 in any way that will make the verifier accept.
Case 2: H is not isomorphic to G, or one or more of the commitments are invalid. In this case, if $b = 0$, the prover will not be able to respond in Step 3 in any way that will make the verifier accept.

Thus, this protocol has soundness error $1/2$.

Zero Knowledge: To argue that this protocol is computational zero knowledge (in the ordinary 2-party sense), we must simulate the interaction of the prover with any given verifier, without knowing the location of a Hamiltonian cycle in the graph. We will make the simplifying assumption that the verifier always completes the protocol (*i.e.* , it never refuses to provide a valid message in Step 2). The key observation here is that *if the simulator knew what the verifier's query bit would be in advance, then it would be easy to simulate the protocol:*

Suppose the simulator "knew" that the verifier would set $b = 0$ (and therefore in Step 3, the verifier would check that the prover reveals a permutation of the graph). In that case, the simulator can simply follow the prover's protocol: namely, it can choose a random permutation π and commit to the entries of the adjacency matrix of $\pi(G)$ in Step 1, and then reveal everything in Step 3, finishing the simulation.

Suppose instead that the simulator "knew" that the verifier would set $b = 1$ (and therefore in Step 3, the verifier would check that the prover reveals a random n-node cycle). In this case, the simulator can, in Step 1, simply commit to a matrix of all 1's, corresponding to the adjacency matrix of the graph that has a directed edge in both directions between every pair of nodes (which we will call the *complete directed graph*). Then in Step 3, the simulator can simply pick a random Hamiltonian cycle in the complete directed graph and reveal it, completing the simulation. Because the commitments to the other entries of the matrix are never opened, the (computationally bounded) verifier cannot distinguish them from the commitments it would have encountered in a real interaction with the prover.

Of course, the simulator does not actually know the verifier's query in advance. But in this simple protocol, the simulator can simply guess what the verifier's query bit will be, and be correct with probability at least $1/2$. If the simulator is wrong, it can simply try again until it guesses correctly, to produce a good simulation.

6.1.3 A step closer to concurrency – parallelization.

We know that the simple protocol above remains zero knowledge if it is executed many times sequentially. To move one step closer to a scenario where there are many verifiers interacting concurrently with a prover, we now consider the problem of repeating this protocol in parallel, rather than sequentially.

In the naïve parallelization of this basic protocol, the prover commits to n adjacency matrices, receives a vector of n bits, and, according to the ith bit, reveals either the ith permutation and the ith adjacency matrix, or the cycle in the the ith adjacency matrix. However, the proof that the basic block is zero-knowledge fails for the naïve parallelization. Indeed, there is no 3-round (black-box) zero-knowledge proof for any language not in BPP that achieves negligible soundness error [GK96].

This already indicates that the standard 2-party definition of zero knowledge does not imply the more stringent requirement of simulability of concurrent interactions with many verifiers, as it does not imply closure under parallel composition. However, the question of designing "parallelized" versions of zero-knowledge proofs has been addressed in the past – from the point of view of trying to design constant-round (*i.e.* low levels of interaction) zero-knowledge proofs with negligible soundness error. In order to "parallelize" the basic block for DIRECTED HAMILTONIAN CYCLE it suffices for the verifier to commit to the vector of queries in advance using a commitment scheme with strong (information-theoretic) secrecy. This was suggested in the paper of Goldreich, Micali, and Wigderson [GMW91], and formally carried out by Goldreich and Kahan [GKa96] for a related NP-complete problem. We adapt this approach to DIRECTED HAMILTONIAN CYCLE:

Protocol 6.1.2 *A directed graph G is the common input to both parties. The prover is assumed to know a Hamiltonian cycle w in G.*

1. *The verifier picks n random* query *bits $q_1, \ldots, q_n$. The verifier sends to the prover commitments to these bits $q_1, \ldots, q_n$.*

2. *The prover picks n random permutations $\pi_1, \ldots, \pi_n$ on the nodes of G. It then sends to the verifier commitments to all the entries of the adjacency matrices of the permuted graphs $\pi_1(G), \ldots, \pi_n(G)$.*

3. *The verifier then opens the commitments it sent in Step 1 to reveal the query bits $q_1, \ldots, q_n$. The prover, upon receipt, verifies correctness of the openings. (If any opening is invalid, the prover halts.)*

4. *The prover does the following: For $i = 1, \ldots, n$:*

   - *If $q_i = 0$, the prover opens its commitments from Step 2 to all the entries of the adjacency matrix of $\pi_i(G)$, and also sends a description of the permutation π_i. The verifier, upon receipt, verifies the correctness of the openings, and also verifies that the revealed adjacency matrix actually is the adjacency matrix of $\pi_i(G)$. If either of these conditions fail to hold, the verifier rejects the proof.*
   - *If $q_i = 1$, the prover opens its commitments only to those entries in the adjacency matrix of $\pi_i(G)$ that correspond to edges in the Hamiltonian cycle $\pi_i(w)$. The verifier, upon receipt, verifies the correctness of the openings, and also verifies that the revealed entries of $\pi_i(G)$ do indeed form a Hamiltonian cycle. If either of these conditions fail to hold, the verifier rejects the proof.*

To see why this protocol is zero knowledge, we must exhibit a simulator.[1] Again, the intuition remains that if the simulator somehow knew in advance what the verifier's queries would be, then the simulation would be easy (as outlined earlier).

Again, of course, the simulator does not actually know the verifier's queries in advance. However, we have not yet made use of the fact that the verifier commits to its queries in advance, in Step 1. We can exploit this to create a situation where the simulator discovers the verifier's queries before it must produce the simulated Step 2 commitments that must be output.

The simulator begins by initializing the verifier, and receiving the verifier's Step 1 commitments to its queries. At this point, the simulator saves the state of the verifier for later use.

The simulator continues by simulating the Step 2 commitments of the prover by simply following the prover's protocol (note that the prover does not require any special information until Step 4, when it must make use of the fact that it knows a Hamiltonian cycle). In Step 3, the verifier reveals her query bits $q_1, \ldots, q_n$. At this point, the simulator *restores* the state of the verifier to just after Step 1, when the verifier committed itself to the query bits. Now, the simulator has achieved the situation we desired – it knows what the query bits of the verifier will be. When the verifier submits its Step 3 openings, it must open the queries to the same values $q_1, \ldots, q_n$, since it was committed to them by its Step 1 message. Thus, the simulator can complete the simulation using the strategy discussed earlier.

The crucial point here is that the simulator was able to "extract" the verifier's query bits, then "rewind" the simulation to an earlier point just after the verifier had committed itself, and then complete the simulation using knowledge of the verifier's queries.

Sketched out, the steps of the simulation are as follows (here, the "prover to verifier" steps are provided by the simulator). We will denote the verifier's commitment function by K, and the prover's commitment function by C:

Simulation:
1. $V \longrightarrow P$: K(query bits)
2. $P \longrightarrow V$: C(permutations of G)
3. $V \longrightarrow P$: open query bits

Reset state of verifier to state after Step 1.

2'. $P \longrightarrow V$: Commit to graphs suitable for query bits
3'=3. $V \longrightarrow P$: open query bits
4. $P \longrightarrow V$: reply to queries

[1]Note again that we assume that the verifier will always produce valid messages when it is supposed to. The analysis for when this is not the case can get quite messy [GKa96].

Thus, this protocol is zero-knowledge and yet can have arbitrarily low soundness error.

Reality strikes – the full concurrent model. Unfortunately, when this parallelized protocol for DHC is run concurrently, say, by a single prover interacting with multiple verifiers, the natural extension of this simulation strategy fails. Consider the following *nested* interleaving, shown in Diagram A below, of n colluding verifiers $V_1, \ldots, V_n$ concurrently executing the parallelized protocol for DHC with a single prover.

V_1	V_2	$\ldots$	V_n
Step 1			
Step 2			
	Step 1		
	Step 2		
		$\vdots$	
			Step 1
			Step 2
			Step 3
			Step 4
		$\vdots$	
	Step 3		
	Step 4		
Step 3			
Step 4			

Diagram A. A troublesome interleaving.

An adversary controlling the verifiers can arrange that the Step 1 commitments to queries made by verifiers $V_{i+1}, \ldots, V_n$ can depend on messages sent by the prover in Step 2 of its interaction with V_i. The difficulty with the straightforward generalization of our simulation strategy is that once the queries in the interaction with V_i are opened (in Step 3), it becomes necessary to re-simulate Step 2 of the interaction with V_i, and therefore all of the interactions with verifiers $V_{i+1}, \ldots, V_n$ must be re-simulated (since the Step 1 messages of each of these verifiers will have changed). The most deeply nested interaction, with V_n, must be simulated 2^n times. After the initial appearance of this work [DNS98], it was shown that in fact the interleaving we present above cannot be (black-box) simulated in probabilistic polynomial time unless DHC is in BPP [KPR98].

6.1.4 Our Approach: Adding Timing Constraints.

This problem of building protocols that are simulatable in the concurrent setting is indeed quite tricky. To overcome the problem of simulation for parallelism, it was possible to make use of an initial commitment by the verifier. A natural analogue of this that can easily be made to work in the concurrent scenario is to require that all verifiers engage in some kind of initial commitment before the interleaved protocols begin. Such a solution, however, is very unsatisfying as it requires a *global* requirement across all the participants in the protocols, very much against the spirit of a distributed aggregate in which parties are not necessarily even aware of each other's existence. Instead, one would like to be able to impose *local* constraints that affect only the pair of users involved in a single protocol. We achieve this goal by formally introducing into our protocols a notion which is always present in reality: the notion of time. We employ time in our protocols by imposing reasonable local timing contraints on the parties in our protocols. We are then able to build several interesting constant-round protocols that are zero knowledge in the concurrent setting.

In this work, we consider timing constraints of only the following two types:

1. **Delays**: One party must delay the sending of some message until *at least* some specified time β has elapsed on its local clock since some specified point earlier in the protocol. Note that no special assumptions (other than the ability for the party to keep time) are needed to implement such a constraint, although delays might be somewhat inconvenient.

2. **Time Limits**: One party, the Receiver, requires that the Sender deliver its next message before some specified time α has elapsed on the Receiver's local clock since some specified point earlier in the protocol. With each such constraint, there is an assumption that an honest Sender will be able to produce and deliver its message within the alloted time. (Thus α must be chosen with this in mind.) We remark that such time limit constraints are already found in almost any secure implementation of any cryptographic protocol.

In all our protocols, we use only the above types of timing constraints, which we believe to be quite reasonable. Furthermore, we assume, as we must, some kind of synchronization of clocks of honest parties. Fortunately, since our timing constraints are so undemanding, the synchronization assumption we need is quite weak: We make the assumption that all good parties have clocks that satisfy what we call an (α, β)-*constraint* (for some $\alpha \leq \beta$):

Definition 6.1.1 *We say a system of honest parties satisfy the* weak synchronization assumption *corresponding to an (α, β)-constraint (for some $\alpha \leq \beta$) if for any two (possibly the same) non-faulty parties P_1 and P_2, the following is always true: Suppose P_1 measures how long it takes for α time to elapse on its local clock, while P_2 measures how long it takes for β time to elapse on its local clock. Then, if P_2 begins its measurement after P_1 does (in real time), it must be the case that P_2 finishes its measurement after P_1 does, as well.*

Remark 6.1.1 We make some remarks concerning the weak synchronization assumption:

1. An (α, β) constraint is implied by most reasonable assumptions on the behavior of clocks in a system (*e.g.* the linear drift assumption).

2. In the special case in which $P_1 = P_2$, the constraint holds trivially for any $\alpha \leq \beta$.

Concurrency and timing are complicated issues; we make several remarks regarding the assumptions and limitations of our model in Section 6.2.4 after defining the model more precisely.

6.1.5 Related previous work

The study of zero-knowledge in a setting with concurrent interactions was initiated in [DDN91], who considered the soundness of concurrent executions of zero-knowledge *proofs of knowledge*. In a proof of knowledge, the prover does not merely prove that some assertion is true (such as that a graph G is Hamiltonian). Instead, the prover proves (in some sense) that it *knows* some information (such as knowing the actual Hamiltonian cycle in a graph G). Here, the verifier faces the possibility that the prover with which it is interacting is actually using some concurrently running second interaction as an "oracle" to help answer the verifier's queries – this is the classic chess master's problem. In this scenario, a cheating prover may be able to use such an oracle to seemingly prove knowledge of information that it actually does not have. The problem here is the possible *malleability* of the interactive proof of knowledge, formalized and addressed in [DDN91]. In this thesis, we concentrate on standard proofs of validity rather than proofs of knowledge.

Our use of timing considerations to ensure zero knowledge appears to be new. The only work of which we are aware that uses timing in zero-knowledge protocols is due to Brands and Chaum [BC94], in which very accurate timing is needed in order to prevent person-in-the-middle attacks by distant parties (see also the work of Beth and Desmedt [BD91]). Note that timing has been suggested as a cryptanalytic tool – the best example is Kocher's timing attack [Koc96] – so it follows that any implementation of a cryptographic protocol must be time-aware in some sense.

6.1.6 Summary of Results

We introduce timing in order to construct protocols that are zero knowledge against many colluding verifiers acting in concurrent interactions.

In a first attempt, we add timing constraints to the standard parallelization of the Blum DIRECTED HAMILTONIAN CYCLE protocol given above. Although we are not able to prove that this achieves our goal, we show that this yields a five-round *weak* concurrent computational zero-knowledge interactive proof for an NP-complete language. Here, by "*weak* concurrent zero knowledge," we are referring to an analog of weak-SZK that we saw earlier in Chapter 3, defined more formally below. By shifting to the context of computationally-sound proofs (also called arguments) where the prover is also assumed to be computationally bounded to polynomial time, we are able to obtain stronger results. In this context, we obtain a six-round *perfect* concurrent zero-knowledge argument in the timing model for DIRECTED HAMILTONIAN CYCLE under the Discrete Log Assumption. We also obtain a four-round concurrent computational zero-knowledge argument for DIRECTED HAMILTONIAN CYCLE in the timing model (thereby showing that the impossibility result of [KPR98] fails to hold in the timing model) under the assumption of the existence of one-to-one one-way functions, which becomes a five-round protocol assuming only ordinary one-way functions.

6.1.7 Subsequent Work on Concurrent Zero Knowledge

After the initial publication of the results presented in this chapter [DNS98], a number of other researchers have investigated concurrent zero knowledge.

In joint work with Cynthia Dwork [DS98], we presented a constant-round preprocessing protocol which uses timing constraints like the ones used in this chapter. This protocol need only be executed once by any pair of prover and verifier to establish a kind of public and private key. After this preprocessing phase, a rich class of constant-round protocols can be executed by the parties with no need for further timing constraints in the protocols. In a future version of this thesis, this material will be presented as well.

Kilian, Petrank, and Rackoff [KPR98] presented the first lower bound on concurrent zero knowledge. They showed that any problem that has a 4 or 5 round concurrent (black-box) zero knowledge interactive proof or argument must be in BPP. Thus, a large class of known zero-knowledge interactive proofs and arguments in the standard setting for problems believed to be outside of BPP do not remain provably zero-knowledge in the setting with concurrency.

Richardson and Kilian [RK99] then presented the first concurrent zero-knowledge argument for an NP-complete language in the standard model with no timing assumptions or constraints. Their protocol, however, required a polynomial number of rounds. This has been improved by Kilian and Petrank [KP00] to a polylogarithmic number of rounds.

Other work has focused on achieving constant-round protocols but imposing different changes to the standard model. Di Crescenzo and Ostrovsky [DO99] presented a non-local approach using a different kind of preprocessing-based protocol, where the requirement is that all verifiers must complete all preprocessing protocols prior to any verifier beginning any further interaction. Canetti, Goldreich, Goldwasser, and Micali [CGGM00] consider a model with much weaker non-locality, described in the next paragraph.

Motivated in part by our notion of concurrent zero knowledge, Canetti, Goldreich, Goldwasser, and Micali [CGGM00] introduce a yet further strengthening of zero knowledge called *resettable* zero knowledge – where the protocol must remain zero knowledge even against a verifier that is given the ability to reset the state of the prover arbitrarily. This requirement implies concurrent zero knowledge, and also provides additional security in settings where a potentially resettable physical device, such as a smart card, may be acting as the prover. [CGGM00] give two main results. First, they show how to translate the result of Richardson and Kilian [RK99] to achieve resettable zero knowledge arguments for NP with a polynomial number of rounds, without timing or non-locality. Second, they also consider a model with a limited form of non-locality, where all verifiers publish a public key (which may be invalid) prior to the start of any interactions[2]. In this model, though making use of potentially stronger intractability assumptions (requiring subexponential hardness), they are able to exhibit constant-round resettable zero-knowledge arguments for an NP-complete language.

6.2 The Model and Preliminaries

In this section, we will define our model, concurrent zero knowledge, and various other notions that we will need in the rest of the chapter. We begin with some preliminaries before we define the model:

6.2.1 Computationally-Sound Proofs or Arguments

Recall our definition of an interactive proof:

Definition 6.2.1 *An interactive protocol (P, V) between a computationally unbounded prover P and a probabilistic polynomial-time verifier V is said to be an* interactive proof system *for a promise problem Π with* completeness error $c(n)$ *and* soundness error $s(n)$ *if*

1. *(*Completeness*): If $x \in \Pi_Y$, then* $\Pr[(P, V)(x) = \texttt{accept}] \geq 1 - c(|x|)$.
2. *(*Soundness*): If $x \in \Pi_N$, then for all P^*,* $\Pr[(P^*, V)(x) = \texttt{accept}] \leq s(|x|)$.

In this definition, we demand that the soundness guarantee hold for *all P^**, even machines that take longer than polynomial time. This definition has advantages in its ease of application; however, for most cryptographic applications, we assume that all parties are polynomially bounded. Therefore, it makes sense to define a notion of proof where the prover is assumed to be computationally bounded as well. This leads to the notion of a computationally-sound proof, also called an argument, first introduced by Brassard, Chaum, and Crepeau [BCC88]:

Definition 6.2.2 *An interactive protocol (P, V) between a probabilistic polynomial-time prover P and a probabilistic polynomial-time verifier V is said to be a* computationally-sound interactive

[2]The actual requirement is considerably weaker. For details, see [CGGM00].

proof system, *or* argument, *for a promise problem* Π *with* completeness error $c(n)$ *and* soundness error $s(n)$ *if*

1. *(*Completeness*): If* $x \in \Pi_Y$, *then* $\Pr[(P, V)(x) = \texttt{accept}] \geq 1 - c(|x|)$.
2. *(*Soundness*): If* $x \in \Pi_N$, *then for all nonuniform probabilistic polynomial-time* P^*, *we have that* $\Pr[(P^*, V)(x) = \texttt{accept}] \leq s(|x|)$.

The standard notions of perfect, statistical, and computational zero knowledge then carry over to arguments as well.

6.2.2 Weak Synchronization Assumption

We recall the weak synchronization assumption we will need in most of our protocols:

Definition 6.2.3 *We say a system of honest parties satisfy the* weak synchronization assumption *given by the* (α, β)*-constraint (for some* $\alpha \leq \beta$*) if for any two (possibly the same) non-faulty parties* P_1 *and* P_2*, the following is always true: Suppose* P_1 *measures how long it takes for* α *time to elapse on its local clock, while* P_2 *measures how long it takes for* β *time to elapse on its local clock. Then, if* P_2 *begins its measurement after* P_1 *does (in real time), it must be the case that* P_2 *finishes its measurement after* P_1 *does, as well.*

6.2.3 Our Model and Concurrent Zero Knowledge

We now consider what it means for an interactive proof system or argument to be zero knowledge against many colluding verifiers acting concurrently. We need a way of modeling the actions of many colluding verifiers. We do so by considering the worst case – that all verifiers are under the direct control of a central controlling adversary. In fact, we even give the adversary the power to create and destroy provers and verifiers at will, and total adaptive control over the timings of all messages exchanged between parties.

Definition 6.2.4 *Let* (P, V) *be an interactive protocol. Let the adversary* A *be a non-uniform probabilistic polynomial-time machine which has a time bound of the polynomial* $t(n)$*. The* concurrent interaction *of* P *and* A *on common input* x *(with private inputs* p_P *for* P *and* p_V *for* A*, sometimes omitted) is a random process which proceeds as follows:*

1. *Choose random strings* r_P *and* r_A *uniformly among infinite strings over* $\{0, 1\}$.
2. *For* $i = 1, 2, \ldots, t(|x|)^3$ *do:*[3]
   (a) *Let* request $= A(p_A, x, m_1, \ldots, m_{i-1}; r_A)$.
   (b) *If* request $=$ (receive, P_a, V_b, m, τ)*, this indicates that prover* P_a *receives from verifier* V_b *the message* m *at prover* P_a*'s local time* τ*. If the adversary has never output* P_a *before, this indicates that the new prover* P_a *is being created. Similarly, if the adversary has never output* V_b *before, this indicates that the new verifier* V_b *is being created. If* V_b *is sending its first message to* P_a*, then a new "copy" of the prover* P *is created (in our experiment) to interact with* V_b*. This copy of the prover is unaware of what*

[3]Here, we are establishing a convention that the number of rounds of communication that A can engage in is bounded by its running time.

other communication P_a might be engaging in. In our experiment, we keep track of the progress of the interactions of every pair of interacting verifiers and provers. Note that the adversary must output a local time τ which is later than all previous local times output for P_a. If this is not the case, the adversary's message is ignored. Let $m_i = \text{request}$.

(c) *If* $\text{request} = (\text{send}, P_a, V_b, \tau)$, *this indicates that prover P_a must send its next message to V_b at the prover's local time τ. (Again, if P_a or V_b have not appeared before, then a new prover or verifier is created in our experiment.) In this case, we obtain the message m that P_a would send next to V_b in its interaction. Note that it could be that P_a would send no message – either because it is not P_a's turn in the interaction, or because P_a refuses to do so because of some rule in the protocol. Again, the adversary must output a local time τ which is later than all previous local times output for P_a. Let* $m_i = (\text{request}, m)$.

(d) *If* $\text{request} = \texttt{halt}$, *then stop.*

Furthermore, we denote by $(P \leftrightarrow A)(x)$ the random variable which represents the view of A in its interaction with P. Namely, this variable consists of the private input p_A to A together with its random coins r_A, along with all m_i.

For an adversary A to satisfy a weak synchronization (α, β)-constraint, it must always output local times in an order that respects the constraint. For example, such an adversary could *not* output the following sequence of requests:

1. $(\text{receive}, P_1, V_1, m, \tau = 0)$
2. $(\text{receive}, P_2, V_2, m, \tau = 0)$
3. $(\text{send}, P_2, V_2, \tau = \beta)$
4. $(\text{send}, P_1, V_1, \tau = \alpha)$

This would violate the constraint, since it implies that local time β elapsed on P_2's clock while less than α local time elapsed on P_1's clock.

We are now ready to define concurrent zero-knowledge proofs and arguments:

Definition 6.2.5 *Let (P, V) be an interactive proof system (resp. argument) with negligible completeness and soundness errors, for a promise problem* Π.

- *(P, V) is said to be a* concurrent (black-box) statistical zero-knowledge *proof system (resp. argument) if there exists a probabilistic expected polynomial-time*[4] *(oracle machine) simulator S and a negligible function α (called the* simulator deviation*) such that for every non-uniform probabilistic polynomial-time adversary A,*

$$\textit{For all } x \in \Pi_Y, \textit{ we have } \left\|S^A(x) - (P \leftrightarrow A)(x)\right\| \leq \alpha(|x|). \tag{6.1}$$

- *A* concurrent perfect zero-knowledge *proof system (resp. argument) is defined in the same way, except that Condition (6.1) is replaced by* $\left\|S^A(x) - (P \leftrightarrow A)(x)\right\| = 0$.

[4]For concurrent zero knowledge, we take the slightly more liberal definition and allow the simulation to be expected polynomial time instead of strict polynomial time. Unlike for standard 2-party statistical zero-knowledge proofs, we do not know if the definitions with expected and strict polynomial-time simulators are equivalent.

- *A* concurrent computational zero-knowledge *proof system (resp. argument) replaces Condition (6.1) with the requirement that the ensembles* $\{S^A(x)\}_{x\in\Pi_Y}$ *and* $(P \leftrightarrow A)(x)\}_{x\in\Pi_Y}$ *are computationally indistinguishable ensembles of distributions.*

Note that here, for an oracle machine S to be probabilistic polynomial-time (resp. probabilistic expected polynomial-time), we mean that for every non-uniform probabilistic polynomial-time machine A, there exists a polynomial $t(\cdot)$ such that for all x, we have that $S^A(x)$ halts in time $t(|x|)$ (resp. expected time $t(|x|)$). The point here is that the polynomial time bound for S can depend on the polynomial bound of the machine A that S makes oracle queries to. This is necessary for the manner in which we have defined the adversary's interactions, since the number of total interactions that the adversary engages in depends on the adversary's running time. Therefore, the running time of S^A must depend on the running time of A.

We can also define a notion of *weak concurrent zero knowledge*, similar to how we defined weak zero knowledge for the standard 2-party case in Chapter 3:

Definition 6.2.6 *Let* (P, V) *be an interactive proof system (resp. argument) with negligible completeness and soundness errors, for a promise problem* Π.

- (P, V) *is said to be a* concurrent weak (black-box) statistical zero-knowledge *proof system (resp. argument) if for every polynomial* p*, there exists a probabilistic polynomial-time*[5] *(oracle machine) simulator* S *such that for every non-uniform probabilistic polynomial-time adversary* A,

$$\text{For all sufficiently long } x \in \Pi_Y, \text{ we have } \left\|S^A(x) - (P \leftrightarrow A)(x)\right\| \leq 1/p(|x|). \tag{6.2}$$

- *A* concurrent weak computational zero-knowledge *proof system (resp. argument) if for every polynomial* p*, there exists a probabilistic polynomial-time (oracle machine) simulation* S *such that for every non-uniform probabilistic polynomial-time adversary* A*, and every non-uniform distinguisher* D*, for all sufficiently long* $x \in \Pi_Y$*, we have*

$$\left|\Pr\left[D\left(S^A(x)\right) = 1\right] - \Pr\left[D\left((P \leftrightarrow A)(x)\right) = 1\right]\right| \leq 1/p(|x|) \tag{6.3}$$

6.2.4 Remarks On Our Model

In this section, we make a series of remarks concerning our model:

- **"Rewinding" and Concurrency.** In the definition of black-box zero knowledge, we assume that the simulator has oracle access to the adversary, which allows it to reset the state of the adversary (which has been called "rewinding the verifier" in the literature) and control the randomness used by the adversary. In a distributed setting with concurrency, however, this assumption that the state of the adversary can be reset to earlier states may be difficult to interpret. For example, consider an adversary that makes use of a concurrent interaction with a third party to select its responses. In this situation, while it seems reasonable to expect that the simulator can reset the adversary machine, it may not be reasonable to expect that the simulator can reset the state of the interacting third party. We make several observations here:

[5]For concurrent weak zero knowledge, we can always restrict to strict polynomial time since the running time can depend on the threshold polynomial p.

1. In some sense, difficulty here is unavoidable, since the adversary could indeed be obtaining knowledge from the third party, depending on the protocol being carried out by the third party and the adversary. If the third party is engaging in the concurrent zero knowledge protocol as a prover and is assumed to be honest, then it fits in the model as another prover.
2. On the other hand, we may consider the entire complex of parties that are concurrently interacting with the provers as being one large probabilistic polynomial-time system (assuming there are only a polynomial number of concurrently interacting parties). Since this entire system can be taken together as a machine whose state can be reset, proving that a protocol is concurrent zero knowledge implies that it "leaks no knowledge to the outside world", since it is zero knowledge with respect to the entire complex of external parties.
3. This issue does cause problems in situations where black-box simulability is used to establish deniability – that is, where one tries to argue that because one can generate simulated transcripts of the protocol, one should not trust transcripts of purported executions of the protocol as evidence that the protocol took place. If however, a user uses a concurrent interaction with a trusted third party to select its responses in the protocol that is supposed to be deniable, then the argument of deniability breaks down, since the user cannot reset the state of the third party as may be needed to produce a simulation.
4. This issue arises with clocks, as well, as discussed below.

- **Timing As Information.** In our adversary model, we give the adversary total adaptive control over all local clocks in the system, subject only to a weak synchronization constraint. This choice allows us to guarantee security against the widest possible array of interleavings, which an adversary could potentially force upon the system[6]. However, because we give control of the prover's message timings to the adversary, our model fails to capture the possibility that the timing of a prover's message itself could yield potentially damaging information to the adversary. Indeed, the problem of using timing information to attack systems has been discussed in the work of Kocher [Koc96], along with methods to minimize the risk of information loss through timing information. However, because of the various engineering issues involved in protecting against timing-based knowledge, an investigation of this important issue is beyond the scope of this thesis. For more information, see [Koc96].

- **Clocks.** Another byproduct of the adversary's total control over local clocks is the implicit assumption that local clocks are indeed *local* – that a system's measurement of time is determined as a result of local computations, as opposed to external inputs. As discussed above, the need for resettability in simulation gives rise to difficulties if there are interactions with third parties. For example, if the clock of a system is provided by signed messages by a trusted third party, this will cause problems in simulation, since the third party cannot be reset to earlier times. This situation is ruled out in our model since we assume that the local clock times are dictated by the adversary machine. Note that in our model, the simulator does not have direct control over clocks in the system, but only indirect control through the ability to reset the state of the adversary, which controls the clocks.

- **Soundness.** In this thesis, we only consider soundness in the sense of validity – we insist that no prover should be able to prove false assertions. We note here that this is in contrast to

[6]For example, an adversary may find ways to manipulate the network or loads on the prover's machine in order to manipulate timings.

proofs of knowledge, where the prover tries to prove that it *knows* some information, rather than simply proving that some string x is a YES instance of a promise problem. Because we deal only with validity, concurrency issues do not arise in proving soundness, since we may consider all interacting parties together as one probabilistic polynomial-time machine (here we assume that there are at most a polynomial number of parties interacting that can effect any single prover). Thus, if we prove that no probabilistic polynomial-time machine acting as prover can prove a false assertion (which is the standard 2-party definition of soundness for computationally-sound proofs), this implies that no false assertions can be proved even in a distributed environment with concurrency. Again, here we do *not* deal with issues of impersonation or misrepresenting one's knowledge. Such issues are addressed in the work of Dolev, Dwork, and Naor [DDN91].

- **Completeness.** The addition of timing constraints adds some potential threats to the completeness guarantee in protocols. In particular, a time limit may expire due to no fault of the party required to produce a timely message. This introduces some unavoidable non-locality depending on the overall network, and also potentially creates a vulnerability to certain kinds of denial of service attacks, where an adversary slows down the network to deny service. Nevertheless, time limits have proven useful and important in secure implementations of many cryptographic protocols (see for instance, [Sch96]) for other reasons, and are widely used. One must take care in selecting the time limits to maintain reasonable expectations on the computation of the parties involved as well as the status of the network.

- **Hijacked Sessions.** Another issue that can arise in the context of distributed environments is the problem of hijacked sessions, where an adversary disrupts one user's communication and assumes that user's identity. We do not address this problem in any detail. We note that many such abuses can be guarded against without assuming any infrastructure by having parties engage in secure key agreement protocols before communicating, and authenticating all messages with the agreed key. Further protection may be possible assuming a public key infrastructure and authenticated channels.

- **Timing Constraints and Composition of Protocols.** While the timing constraints that we propose (delays and time limits) allow for considerable flexibility, one must nevertheless be careful when designing protocols to ensure that the timing constraints remain reasonable. Although in this thesis we only discuss using these timing constraints to achieve zero knowledge, it may be possible to use such timing constraints to provide other types of security. Thus, when building protocols out of many components that use timing constraints, one must ensure that all the timing constraints taken together do not conflict with each other.

6.2.5 Commitment Schemes

Commitment schemes are a crucial ingredient in many of our protocols. They are used to enable a party to commit itself to a value while keeping it secret. Later on, a commitment may be "opened" to reveal the value – and it should be guaranteed that a given commitment can only be "opened" to a single value.

In other words, loosely speaking, a bit commitment scheme is a protocol between two parties – a sender S and a receiver R – that has two phases: a "commitment phase," in which S commits to a value; and an "opening phase," where S reveals to R the value that it committed to. This protocol must enjoy two basic properties:

- *Secrecy:* After the "commitment phase," the receiver R should not gain any knowledge about the committed value, even if R tries to cheat.

- *Binding:* After a successful "commitment phase," there can be at most one value that S can successfully open the commitment as in the "opening phase," even if S tries to cheat.

We will explore many variations on the notion of a commitment scheme, but they will all fall into the general framework outlined above. We now discuss specific types of commitment schemes.

When formalizing either the secrecy or the binding properties mentioned above, we can choose to impose either a computational or statistical[7] standard of security, much as we did when defining zero knowledge. It is easy to show that no commitment protocol can exist that has statistical guarantees for both secrecy and binding, so we must consider a tradeoff between these two requirements. Thus, we consider both *Computationally Secret Statistically Binding (CSSB)* commitment schemes as well as *Statistically Secret Computationally Binding (SSCB)* commitment schemes.

6.2.6 Computationally Secret Statistically Binding (CSSB) Commitment

This is the more commonly considered type of commitment scheme, where the committer is committed to exactly one value, but secrecy is only guaranteed in a computational sense. A formal definition follows, following [Gol95][8].

Definition 6.2.7 *(*Computationally Secret Statistically Binding (CSSB) bit commitment scheme*):* *A* CSSB bit commitment scheme *is a tuple of probabilistic polynomial-time machines, denoted* (S_I, C, O, R_I, V)*, where* (S_I, R_I) *are a pair of interactive machines. Informally,* S_I *and* R_I *correspond to an initialization protocol between sender and receiver,* C *is the machine which produces commitment strings,* O *is the machine which produces "openings," and* V *is the machine which verifies that an opening is valid. These machines together describe a CSSB bit commitment scheme if:*

- Input/Output Specification: *The common input to the initialization protocol* (S_I, R_I) *is an integer* k *presented in unary, serving the role of a security parameter. This interaction produces a private output for the sender* p_S *and a private output for the receiver* p_R*. We will denote by* $\text{View}(S_I, R_I)$ *the random coins used by* R_I *in the interaction together with all messages received by* R_I *from* S_I*. The commitment machine* C *takes as input* p_S *together with a bit* $b \in \{0,1\}$ *to be committed to, and outputs the commitment string* c *to be sent to the receiver. The opening machine* O *takes as input* p_S *and the random string* r_C *used by the commitment machine* C*, along with a bit* $b' \in \{0,1\}$*, and produces an opening string* o*. Finally, the verification machine* V *takes as input the initial private output for the receiver* p_R *together with a commitment string* c*, a bit* b''*, and an opening string* o*, and outputs* 1 *if the verifier accepts the opening as the bit* b''*, and* 0 *otherwise.*

- Completeness: *For all* $b \in \{0,1\}$*, the probability that* $V(p_R, c, b, o) = 1$ *is* 1*, where* p_R *and* p_S *are produced by* (S_I, R_I)*, while* $c = C(p_S, b)$ *using random coins* r_C*, and* $o = O(p_S, r_C, b)$.

[7] For simplicity in application, here when considering a statistical security standard for secrecy, we will insist on *perfect* statistical security. See below for formal definitions.

[8] We give a somewhat less general definition, where we insist that after some initializing interaction, the commitment and opening process are both non-interactive.

- Secrecy: *The receiver (even when deviating from the protocol) cannot distinguish a commitment to* 0 *from a commitment to* 1 *in a computational sense: Namely, for every probabilistic polynomial-time machine* R_I^* *interacting with* S_I*, the ensembles* $(\text{View}(S_I, R^*)(1^k), C(p_S, 0))$ *and* $(\text{View}(S_I, R^*)(1^k), C(p_S, 1))$ *(indexed by k) are computationally indistinguishable.*

- Binding: *The sender (even when deviating from the protocol) cannot produce a commitment that can be opened as both a* 0 *and a* 1*, even by an infinitely powerful party. Namely, for every (arbitrarily powerful) cheating sender machines* S_I^* *and* C^**, there exists a negligible function* $\alpha(k)$ *such that the probability that there exist two openings* o^0 *and* o^1 *such that* $V(p_R, c, 0, o^0) = 1$ *and* $V(p_R, c, 1, o^1) = 1$ *is at most* $\alpha(k)$*. Here,* p_R *and* p_S *are produced by* (S_I^*, R_I)*, while* $c = C^*(p_S)$*.*

There are two simple example of CSSB bit commitment schemes that we will use. Full proofs of security for both can be found in [Gol95].

The first assumes the existence of a 1-1 one-way function f together with a hard-core predicate for it, denoted B. (We assume that these functions are secure against non-uniform probabilistic expected polynomial-time machines.)

Construction 6.2.1 *(*simple bit commitment*): Let* $f : \{0,1\}^* \to \{0,1\}^*$ *be a 1-1 one-way function, and* $B : \{0,1\}^* \to \{0,1\}$ *be a hard-core predicate for* f*. Then we consider the following construction of a CSSB bit commitment scheme:*

1. *The initialization phase is trivial.*
2. $C(b) = (f(s), B(s) \oplus b)$ *where* s *is chosen uniformly from* $\{0,1\}^k$*. Note* s *denotes the random coins used by* C*.*
3. $O(s, b) = s$*. To open, the committer simply reveals its coins.*
4. $V(c = (x, \tau), b, o) = 1$ *iff* $f(o) = x$ *and* $B(o) \oplus b = \tau$*. The verification simply checks to see if the commitment was created according to the rules of the protocol.*

This construction in fact achieves perfect *binding, since by the injectivity of* f*, no commitment string can be a commitment to both* 0 *and* 1.

We also recall another construction, due to [Nao91], which can be based on any one-way function (a proof of security for this scheme is also given in [Gol95]). Using the results of [HILL], we know that one-way functions imply the existence of pseudo-random generators. We use such generators in the construction:

Construction 6.2.2 *(*bit commitment from any one-way function*): Let* $G : \{0,1\}^* \to \{0,1\}^*$ *be a pseudo-random generator where* $|G(s)| = 3 \cdot |s|$ *for all* $s \in \{0,1\}^*$*. Then consider the following construction of a CSSB bit commitment scheme:*

1. *In the initialization phase,* R_I *simply uniformly selects* $r \in \{0,1\}^{3k}$ *and sends it to* S_I*. The private outputs of both parties is this random string* r*.*
2. $C(r, b)$ *chooses* $s \in \{0,1\}^k$*, and outputs* $G(s)$ *if* $b = 0$ *and* $G(s) \oplus r$ *if* $b = 1$*. Note that the random coins of* C *are this seed* s*.*
3. $O(r, s, b) = s$*. To open the commitment, the committer reveals his seed* s*.*
4. $V(r, c, b, o)$ *accepts if* $b = 0$ *and* $G(o) = c$ *or if* $b = 1$ *and* $G(o) \oplus r = c$*).*

Note that, in this construction, there is a probability of at most 2^{-k} that r might be chosen by R_I to be equal to $G(s) \oplus G(s')$ for some $s, s' \in \{0,1\}^k$. In this case, note that by choosing the commitment string $c = G(s)$, the sender could open as both a 0 (by sending the opening $o = s$) as well as a 1 (by sending the opening $o = s'$).

We note that to commit to a string, one can simply commit bit-by-bit to each bit in the string. Note however, that one needs to obtain a new random string r for each bit to be committed. This can be done initially all at once and used in subsequent commitments.

6.2.7 Statistically Secret Computationally Binding (SSCB) Commitment

We now discuss commitment schemes in which the secrecy requirement is made perfectly secure – the commitment contains absolutely *no* information about what is being committed to – while the binding nature of the commitment only holds in a computational sense. Note that, while for CSSB commitment schemes, we gave a definition only for commitment to a single bit, for SSCB commitment schemes, we will give a definition for arbitrarily long strings. We will also exhibit a SSCB construction which is more efficient for long strings than a bit-by-bit commitment would be.

Definition 6.2.8 *(*Statistically Secret Computationally Binding (SSCB) string commitment scheme*): A* SSCB string commitment scheme *is a tuple of probabilistic polynomial-time machines, denoted (S_I, K, O, R_I, V), where (S_I, R_I) are a pair of interactive machines. Informally, S_I and R_I correspond to an initialization protocol between sender and receiver, K is the machine which produces commitment strings, O is the machine which produces "openings," and V is the machine which verifies that an opening is valid. These machines together describe a SSCB string commitment scheme if:*

- Input/Output Specification: *The common input to the initialization protocol (S_I, R_I) is an integer k presented in unary, serving the role of a security parameter. This interaction produces a private output for the sender p_S and a private output for the receiver p_R. We will denote by* $\mathrm{View}(S_I, R_I)$ *the random coins used by R_I in the interaction together with all messages received by R_I from S_I. The commitment machine K takes as input p_S together with a string $s \in \{0,1\}^k$ to be committed to, and outputs the commitment string c to be sent to the receiver. The opening machine O takes as input p_S and the random string r_K used by the commitment machine K, along with a string $s' \in \{0,1\}^k$, and produces an opening string o. Finally, the verification machine V takes as input the initial private output for the receiver p_R together with a commitment string c, a string s'', and an opening string o, and outputs 1 if the verifier accepts the opening as the string s'', and 0 otherwise.*

- Completeness: *For all $s \in \{0,1\}^k$, the probability that $V(p_R, c, s, o) = 1$ is 1, where p_R and p_S are produced by (S_I, R_I), while $c = K(p_S, s)$ using random coins r_K, and $o = O(p_S, r_K, s)$.*

- Secrecy: *The commitments contain no information whatsoever about the string being committed to: Namely, for every probabilistic polynomial-time machine R_I^* interacting with S_I, for every valid run of (S_I, R_I^*) producing private inputs p_R and p_S, we have that the distribution of outputs produced by $K(p_S, s)$ is completely independent of s. In other words, for every $s, s' \in \{0,1\}^k$, the distribution produced by $K(p_S, s)$ is identical to the distribution produced by $K(p_S, s')$.*

- Binding: *The sender (even when deviating from the protocol) cannot produce a commitment that it can later open as both a* 0 *and a* 1*, assuming the sender is computationally limited. Namely, for every probabilistic polynomial-time cheating sender machines* S_I^* *and* M^**, there exists a negligible function* $\alpha(k)$ *such that the probability that* $M(p_S) = (c, s^0, o^0, s^1, o^1)$ *such that* $s^0 \neq s^1$*,* $V(p_R, c, s^0, o^0) = 1$ *and* $V(p_R, c, s^1, o^1) = 1$ *is at most* $\alpha(k)$*. Here,* p_R *and* p_S *are produced by* (S_I^*, R_I)*.*

We now give an example of an SSCB string commitment scheme based on the Discrete Logarithm Problem (DLP) over prime finite fields, due to [CvHP91]. First, let us identify the specific assumption we will make:

Assumption 6.2.1 *(*Discrete Logarithm Assumption (DLA)*): Consider a generation algorithm* G_{DL} *which does the following: On input* 1^k*, it uniformly selects a prime* q *between* $2^k + 1$ *and* 2^{k+1} *(by picking random numbers and testing for primality*[9] *until it finds a prime). It then tests if* $p = 2q + 1$ *is also prime; if not then this process is repeated until one is found. We make the number-theoretic assumption that primes of this form have sufficient density to ensure that this process will terminate in time that is expected to be polynomial in* k*. Note that the multiplicative group* $\mathbb{Z}_p^*$ *contains exactly* $p - 1 = 2q$ *elements. Therefore it has a unique subgroup* Q *of size* q*. Random elements* $g \neq 1$ *are chosen in* $\mathbb{Z}_p^*$ *until one is found that satisfies* $g^q = 1 \bmod p$ *(note that this holds for at least* $q - 1$ *choices of* g*). Then* g *must generate the subgroup* Q*. A random number* $x \in \mathbb{Z}_q$ *is chosen, and* $h = g^x \bmod p$ *is thus set to be a random element of* Q*. Finally* $G_{DL}(1^k)$ *outputs the tuple* (p, g, h, x)*.*

Then we assume that for every non-uniform probabilistic expected polynomial-time machine M*, there exists a negligible function* $\alpha(\cdot)$ *such that:*

$$\Pr\left[h = g^{M(p,g,h)} \bmod p\right] < \alpha(k)$$

where $(p, g, h, x) \leftarrow G_{DL}(1^k)$*.*

Remark 6.2.1 Note that we need not insist that p have the form $2q + 1$, although it is generally believed that the DLP is hardest for primes of this form. For further discussion, see [Gol95].

We now return to the construction of a SSCB string commitment scheme based on the DLP:

Construction 6.2.3 *(*SSCB string commitment from DLP*):*

1. *In the initialization phase,* R_I *runs* G_{DL} *to obtain* (p, g, h, x)*. The values* p*,* g*, and* h *are sent to* S_I*. The private output of* S_I *are these values, while the private output of* R_I *are all four values.*

2. $K(p, g, h, s) = g^r \cdot h^s \bmod p$*, where* r *is chosen uniformly from* $\mathbb{Z}_q$*, and* s *is interpreted as a number between* 0 *and* $2^k < q$*. Note that* r *is the randomness used by* K*.*

3. $O(p, g, h, r, s) = (r, s)$*. To open the commitment, the committer reveals the exponent* r *and the string* s*.*

4. *The verification procedure, given the commitment* c *and the opening* (r, s) *accepts if indeed* $c = g^r \cdot h^s \bmod p$*.*

[9]For more information on Primality testing algorithms, see [GB99].

Since this scheme is not very well known and crucial to several of our constructions, we will prove that it is indeed a SSCB commitment scheme, assuming the Discrete Logarithm Assumption above:

Proof: Clearly, the completeness condition is satisfied.

To see why Perfect Statistical Secrecy holds, notice that for any value of s, the distribution $K(p, g, h, s) = g^{(\cdot)}h^s$ is uniform over Q. So indeed, the commitment string reveals no information whatsoever about the string s being committed to.

To see why the sender is nevertheless bound computationally, consider any probabilistic polynomial-time machine M which violated the Binding constraint. Note that the input to M is (p, g, h) from the output of $G_{DL}(1^k)$. Suppose M outputs (c, s^0, r^0, s^1, r^1) such that $s^0 \neq s^1$, and yet $c = g^{r^0}h^{s^0} = g^{r^1}h^{s^1} \bmod p$. Now, $s^0 \neq s^1$ implies $r^0 \neq r^1 \bmod q$. Then, in fact, we have $g^{r^0-r^1} = h^{s^1-s^0} \bmod p$, and hence $h = g^y \bmod p$, where we compute $y = \frac{r^0-r^1}{s^1-s^0} \bmod q$. Thus, from M's output we can efficiently compute a $y \in \mathbb{Z}_q$ such that $h = g^y$. By the DLA, we know this can happen with at most negligible probability, which completes our proof. ■

Remark 6.2.2 In fact, the DLP-based construction given above has an addition "trapdoor" property that will be crucial to some of our protocols: This is the property that, if the committer learns x, then in fact it would be able to open any commitment to any desired value. Note again that the distribution of commitments is totally independent of the string s being committed to. Suppose a commitment sent was $c = g^r h^s$, where $r \in_R \mathbb{Z}_q$. Now, for any s', we know $c = g^{(r-(s'-s)x)}h^{s'}$. Furthermore, for any fixed s, s' we have that $(r - (s' - s)x) \bmod q$ is distributed uniformly in $\mathbb{Z}_q$. Thus, if the sender knows x, the sender can decommit any commitment to an arbitrarily selected value s' in a manner perfectly indistinguishable from a honest commitment to s'.

6.3 Weak Concurrent Zero-Knowledge Proofs for NP

In this section, we show how adding timing constraints to the 5-round computational zero-knowledge proof system for NP first suggested by [GMW91], and presented and analyzed rigorously by [GKa96], allow us to achieve weak concurrent zero knowledge (though not zero knowledge). The intuition provided in this section will be useful in the next section, where we will see how to achieve full concurrent zero knowledge in the context of computationally-sound proofs. For concreteness and consistency with the rest of this thesis, we will base the protocol on the Discrete Logarithm Assumption, although in fact any claw-free permutation family suffices (for further information, see [Gol95]). The protocol given by [GKa96] was based on the NP-complete problem GRAPH 3-COLORABILITY. Here we present this protocol adapted to the NP-complete problem DIRECTED GRAPH HAMILTONICITY, based on the basic zero-knowledge proof due to Blum [Bl]. We recall the problem:

DIRECTED GRAPH HAMILTONICITY:
Instance: A directed graph G.
Question: Does there exist a Hamiltonian cycle in G, *i.e.* a cycle containing every node in G exactly once?

The protocol makes use of both a Statistically Secret Computationally Binding (SSCB) commitment scheme, as well as a Computationally Secret Statistically Binding (CSSB) commitment scheme. We now describe the protocol:

Protocol 6.3.1 Weak concurrent zero-knowledge proof system for DIRECTED HAMILTONIAN CYCLE **with timing.**
A directed graph G is the common input to both parties. The prover is assumed to know a Hamiltonian cycle w in G.

1. *The prover and verifier interact to set up a SSCB commitment scheme (K) for use by the verifier, as well as a CSSB commitment scheme (C) for use by the prover. Note that this step may be interleaved with the steps described below, as long as the commitment schemes are ready before first used.*
2. *The verifier picks n random* query *bits $q_1, \ldots, q_n$. The verifier sends to the prover a commitment using the SSCB (K) commitment scheme to these bits $q_1, \ldots, q_n$.*
3. *The prover picks n random permutations $\pi_1, \ldots, \pi_n$ on the nodes of G. It then sends to the verifier commitments using the CSSB (C) commitment scheme to all the entries of the adjacency matrices of the permuted graphs $\pi_1(G), \ldots, \pi_n(G)$.*
4. *The verifier then opens the commitments it sent in Step 2 to reveal the query bits $q_1, \ldots, q_n$. The prover, upon receipt, verifies correctness of the openings. (If any opening is invalid, the prover halts.)*
5. *In the final step, the prover does the following: For $i = 1, \ldots, n$:*
   - *If $q_i = 0$, the prover opens its commitments from Step 3 to all the entries of the adjacency matrix of $\pi_i(G)$, and also sends a description of the permutation π_i. The verifier, upon receipt, verifies the correctness of the openings, and also verifies that the revealed adjacency matrix actually is the adjacency matrix of $\pi_i(G)$. If either of these conditions fail to hold, the verifier rejects the proof.*
   - *If $q_i = 1$, the prover opens its commitments only to those entries in the adjacency matrix of $\pi_i(G)$ that correspond to edges in the Hamiltonian cycle $\pi_i(w)$. The verifier, upon receipt, verifies the correctness of the openings, and also verifies that the revealed entries of $\pi_i(G)$ do indeed form a Hamiltonian cycle. If either of these conditions fail to hold, the verifier rejects the proof.*

Timing Constraints:

1. *The prover must receive the verifier's Step 4 message within α (local) time since the receipt of the verifier's Step 2 message. Otherwise the prover aborts.*
2. *The prover delays sending the Step 5 message until β (local) time has elapsed since the receipt of the verifier's Step 4 message.*

Note again that we make the weak synchronization assumption for the (arbitrary) pair of times (α, β) used in the protocol. For completeness of the protocol, we must further assume that all necessary communication (Steps 3-4) can occur in time less than α. Recall that under the Discrete Logarithm Assumption, there exists an SSCB commitment scheme (described in Construction 6.2.3) requiring a single initialization message from the receiver to the sender (in this case, prover to verifier). Since the Discrete Logarithm Assumption also gives a one-to-one one-way function, this also implies that the simple CSSB commitment scheme described in Construction 6.2.1 exists, which requires no initialization. Thus using these commitment schemes, the above protocol requires 5 rounds of communication.

We can now establish:

Theorem 6.3.1 *Protocol 6.3.1 is a weak computational concurrent zero-knowledge proof system for* DIRECTED GRAPH HAMILTONICITY, *with soundness error* $2^{-n} + \alpha(n)$, *where* $\alpha(\cdot)$ *is a negligible function*[10], *assuming the existence of CSSB and SSCB commitment schemes and the weak synchronization assumption for* (α, β).

Proof: We must argue completeness, soundness, and weak concurrent zero knowledge. Completeness follows by inspection.

Soundness. For soundness, we assume that G does not have a Hamiltonian cycle, and we will show that the probability that any prover P^* can successfully complete the protocol is at most $2^{-n} + \alpha(n)$, for some negligible function $\alpha(\cdot)$.

Now, conditioned on any particular Step 2 commitments by the verifier, because of perfect secrecy of the verifier's SSCB commitment scheme, the distribution of the query bits $q_1, \ldots, q_n$ remains uniform. Then we consider the prover's Step 3 commitment. By the statistical binding property of the CSSB commitment scheme, we know there is at most probability $\alpha(n)$, where $\alpha(\cdot)$ is some negligible function, that *any* of the prover's commitments are not absolutely binding. Thus we may condition on this fact and lose only an $\alpha(n)$ probability mass. Hence, we may speak of the prover's commitments defining the adjacency matrices of graphs $H_1, \ldots, H_n$ (where it can also be the case that some of these graphs are *invalid*, by which we mean that the prover's commitment to some entry of the adjacency matrix could not be opened as either a 0 or a 1).

Now, for each $i = 1, \ldots, n$, consider the following: Either H_i is isomorphic to G, or it is not (here we include the case that H_i is invalid):
Case 1: H_i is isomorphic to G. Then H_i cannot contain a Hamiltonian cycle. Therefore, if $q_i = 1$, the prover will not be able to respond in Step 5 in any way that will make the verifier accept.
Case 2: H_i is not isomorphic to G. In this case, if $q_i = 0$, the prover will not be able to respond in Step 5 in any way that will make the verifier accept.

Since conditioned on all previous messages, the each bit q_i revealed in Step 4 is uniform, the prover can fail to satisfy the verifier with independent probability $1/2$ for each i. Therefore, the probability that the prover will cause the verifier to accept is at most 2^{-n}. Thus, factoring the probability the prover's commitments are not binding, the overall probability of the prover convincing the verifier to accept is as most $\alpha(n) + 2^{-n}$, as claimed.

Weak concurrent zero-knowledge. Before we argue that the protocol is weak concurrent zero-knowledge, we will first informally argue that the protocol is zero knowledge in the 2-party case, in the special case in which we are assured that the verifier will always provide valid commitments in Step 2 and always open them in a valid and timely manner in Step 4. This argument proceeds along the lines of the one given in [GKa96], although it is much simpler because of the assumption we make on the verifier's behavior.

Intuition – ordinary zero knowledge in a special case. [11] We must simulate the interaction of the prover with any given verifier, without knowing the location of a Hamiltonian cycle in the

[10]The negligible α contribution to the soundness comes only from the probability that the CSSB (C) commitment used by the Prover turns out not to be binding. If the Prover's commitment scheme has perfect binding (such as Construction 6.2.1), then $\alpha(\cdot) = 0$)

[11]This discussion is largely repeated from the introduction to this chapter.

graph. The key observation here is that *if the simulator knew what the verifier's query bits would be in advance, then it would be easy to simulate the protocol:*

Suppose the simulator "knew" the verifier would set $q_i = 0$ (and therefore check that the prover reveals a permutation of the graph in Step 5). In that case, the simulator can simply follow the prover's protocol, namely choose a random permutation π_i and commit to the entries of the adjacency matrix of $\pi_i(G)$ in Step 3, and then reveal everything in Step 5.

Suppose instead that the simulator "knew" the verifier would set $q_i = 1$ (and therefore check that the prover reveals a random n-node cycle in Step 5). In this case, the simulator can, in Step 3, simply commit to a matrix of all 1's, corresponding to the adjacency matrix of the graph that has a directed edge in both directions between every pair of nodes (which we will call the *complete directed graph*). Then in Step 5, the simulator can simply pick a random Hamiltonian cycle in the complete directed graph and reveal it. Because the commitments to the other entries of the matrix are never opened, the (computationally bounded) verifier cannot distinguish them from the commitments it would have encountered in a real interaction with the prover.

Of course, the simulator does not actually know the verifier's queries in advance. However, also note that the discussion above did not make use of the fact that the verifier commits to its queries in advance, in Step 2. We can exploit this to create a situation where the simulator discovers the verifier's queries before it must produce the simulated Step 3 commitments that must be output.

The simulator begins by initializing the verifier, engaging in the Step 1 setup of the commitment schemes, and receiving the verifier's Step 2 commitments to its queries. At this point, the simulator saves the state of the verifier for later use.

The simulator continues by simulating the Step 3 commitments of the prover by simply following the prover's protocol (note that the prover does not require any special information until Step 5). In Step 4, the verifier reveals her query bits $q_1, \ldots, q_n$. At this point, the simulator *restores* the state of the verifier to just after Step 2, when the verifier committed itself to the query bits. Now, the simulator has achieved the situation we desired – it knows what the query bits of the verifier will be. When the verifier submits its Step 4 openings, it must open the queries to the same values $q_1, \ldots, q_n$, since it was committed to them by its Step 2 message. (If the computationally limited verifier were able to open its commitments differently with noticeable probability, this would lead to a contradiction of the binding requirements of the SSCB commitment scheme the verifier uses.) Thus, the simulator can complete the simulation using the strategy discussed earlier.

The crucial point here is that the simulator was able to "extract" the verifier's query bits, then "rewind" the simulation to an earlier point just after the verifier had committed itself, and then complete the simulation using knowledge of the verifier's queries.

Intuition – the concurrent scenario. This simulation strategy for the protocol, without timing constraints, unfortunately fails in the concurrent scenario, where we must simulate the interaction of a set of provers with many colluding verifiers acting concurrently. This failure occurs essentially because while the simulator is waiting to extract the query bits of one verifier, another verifier may appear and insist on completing the entire interaction. In order to do this, the simulator will need to extract the new verifier's query bits, *which could depend on the particular Step 3 message that the simulator produced for the first verifier.* In this case, when the simulator finishes extracting the query bits of the first verifier and then *changes* the Step 3 message sent to it, the query bits of the second verifier could change, which would mean that the entire simulation of the second verifier would have to be redone.[12]

[12]For more discussion on this, see the introduction to this chapter.

We solve this problem using timing constraints. The timing constraints of our protocol, together with the weak synchronization assumption, imply the following extremely useful *interleaving constraint* on the adversary:

Interleaving Constraint:

For any two verifiers V_1 and V_2, if V_2 provides its Step 4 message after (in real time) V_1 provides its Step 2 commitments, then it must be that the Step 5 response to V_2 will not need to be sent before V_1 provides its Step 4 message.

This is crucial to us, since any interaction can be simulated perfectly up until Step 5 – and we need the Step 4 message of the verifier to discover its queries. The simulation proceeds by simulating each interaction perfectly until it reaches Step 4, at which point it resets back and changes its Step 3 commitments. Because of the interleaving constraint, as we continue this process one verifier at a time, we will never encounter the situation where we must provide a Step 5 message for any verifier whose queries we have not already discovered and used to prepare Step 3 commitments.

This would suffice to establish full concurrent zero knowledge, were it not for the simplifying assumption we made regarding the verifiers. In reality, the verifiers controlled by the adversary need not provide Step 4 messages in a timely or correct fashion. Unfortunately, this causes a dilemma – if a verifier V refuses to cooperate at Step 4, what can we do? If we simply ignore V and move on, later we may need to reset back to an earlier time in the simulation because of another verifier, and this time V may decide to provide a timely and correct Step 4 message. Then we must reset again, and we wind up in a very similar situation to the one we started in. In later sections, we will see how to design protocols in a different manner to deal with this problem (in computationally-sound proofs). For this protocol, however, we can prove only weak concurrent zero knowledge. This is achieved by trying to get a valid Step 4 message from each verifier repeatedly, but only a fixed (polynomial) number of times. If we succeed, then we proceed with the simulation. If the verifier never cooperates, then we assume that the verifier will *never* provide a valid Step 4 message. If the verifier later does provide such a message, we abort. Unfortunately, this can happen with inverse polynomial probability, which is why we are only able to prove *weak* zero knowledge.

Proof sketch of weak concurrent zero knowledge – the simulator. Let the adversary A be bounded by time $t(|x|)$. Let $1/q(|x|)$ be the simulator deviation allowed (recall in weak zero knowledge, one constructs for every polynomial $q(\cdot)$ a simulator that achieves simulator deviation $1/q(|x|)$). For notational convenience, we will omit the dependence on $|x|$ and simply write t for the time bound of A, and q for the inverse of the simulator deviation we are aiming for.

The simulator's process will be very similar to the simulator for the ordinary parallel DHC protocol. We will call a verifier V's queries *extracted* once we discover V's Step 4 message revealing its queries. After we reset that verifier and are ready to change the Step 3 commitments as specified above to be appropriate for the queries we have found, we call the verifier *neutralized.* Once a verifier is neutralized, as noted above, all future messages exchanged with it (in its current interaction) are easy to simulate. As noted above, the simulator will be making judgements about whether a verifier will ever provide a valid and timely Step 4 message. If the simulator decides that a verifier will never provide a valid and timely Step 4 message, then we call the verifier *lazy.* The simulation will also have a notion of a *current* or *active* verifier, which will be the verifier that the simulator is trying to neutralize or declare lazy.

In the simulation, we will maintain two important invariants:

- The simulator will never reset the interaction with A back to a point before the Step 2 message of the current verifier.

- All verifiers that provided a Step 2 message before the current verifier's Step 2 message will be either declared lazy or be neutralized.

Clearly these invariants hold at the start of the simulation. The simulator then proceeds in the following manner. As in the ordinary parallel protocol, for each interaction of prover and verifier before the verifier is neutralized, until and including Step 4 the simulator simply follows the prover's protocol exactly with that verifier. For each new verifier V that appears and provides a valid Step 2 message, the simulator makes that verifier the active verifier, saves the state of the adversary, and executes $t^4 \cdot q$ *trials*, resetting back to this saved state at the beginning of each trial:

In each trial, the simulator proceeds following the prover's protocol exactly with V. For all other verifiers that came after V, it does the same. For all verifiers that have been neutralized, the simulator proceeds in the manner specified in our discussions above. Note that if a neutralized verifier ever opens its queries in a way that is not the same as the queries extracted from it, we abort the simulation. By the properties of the SSCB commitment scheme, we know this can occur only with negligible probability. For all verifiers that have been declared lazy, the simulator follows the prover's protocol. But if such a verifier ever provides a valid Step 4 message, the trial is declared a *failed* trial, is not counted among the $t^4 \cdot q$ trials, and a new trial begins. If, however, over $3t^4 \cdot q$ failed trials occur, then the entire simulation is aborted. This is called a *trial failure abort.* We will of course show that the probability of this occurring is very small. If V does provide a timely and valid Step 4 message which reveals its queries, the simulator resets back to just after V's Step 2 message, and thus V is neutralized. If V provides an invalid Step 4 message, or the time limit for V's message runs out, we stop the trial and begin a new one.

If all $t^4 \cdot q$ trials end in failure, then V is declared lazy, we reset back to the saved state, and proceed to the next verifier.

If, once we have declared a verifier lazy or neutralized it, before the next new verifier arrives, a lazy verifier provides a valid and timely Step 4 message, then we abort the entire simulation. This is called an *output abort,* since the simulation that occurs after one verifier has been neutralized or declared lazy, but before a new verifier (if one is coming) provides a Step 2 message, is included in the output of the simulation.

The output of the simulator is the combined set of messages from the portions of the simulation from the very beginning, before the first active verifier was found, and that occurs after one verifier has been neutralized or declared lazy, but before a new verifier (if one is coming) provides a Step 2 message.

Proof of weak concurrent zero knowledge – simulator analysis. Clearly, the simulator described above runs in time polynomial in $|x|$, t, and q.

We now discuss the quality of the simulation. First, consider a simulator S' that acts as above, but instead of performing $t^4 \cdot q$ trials, it conducts as many trials as necessary to either extract the verifier's queries, or conclude that there is no input for which the verifier will provide a valid Step 4 message. This simulator may not be polynomial-time, but by the discussions above, the output of this simulator is certainly computationally indistinguishable from $(P \leftrightarrow A)(x)$[13].

To analyze our simulator, we compute the probability that it behaves in a manner that is observably different than S'. Note that if our simulator has failed trials fewer than $3t^4 \cdot q$, this does not affect the output and is therefore irrelevant. Also, if our simulator and S' differ in terms of the contents of commitments that are never opened, this is by assumption not a difference that can be

[13]This follows formally from hybrid arguments that are standard in cryptography. For examples, see [Gol95].

detected, except with negligible probability. Thus, the only differences then are trial failure aborts and output aborts (which do not occur for S', but can in our simulator).

Consider the behavior of any verifier V, conditioned on the state of the adversary before the trials for V. Let p_V be the probability that V will provide a timely and valid Step 4 message. If $p_V > 1/(4t^3 \cdot q)$, then the probability that the simulator does not extract the verifier's queries is:

$$(1 - p_V)^{(t^4 \cdot q)} = 2^{-\Omega(t)}$$

On the other hand, suppose $p_V \leq 1/(4t^3 \cdot q)$. Suppose, in the worst case, the simulator does not extract V's queries. In this case, the probability that V will later provide a valid and timely Step 4 message is at most $1/(4t^3 \cdot q) + \mu(|x|)$, where $\mu(|x|)$ denotes the amount by which the probability of V providing a timely and valid Step 4 message can increase in the case where the adversary A sees some CSSB commitments to matrices of all 1's. We know from the security of the CSSB scheme that $\mu(\cdot)$ is a negligible function. Therefore, without loss of generality, we can bound the probability of V providing a valid and timely Step 4 message in the future by $1/(2t^3 \cdot q)$.

Thus, the probability of an output abort is bounded by $t \cdot 1/(2t^3 \cdot q) \leq 1/(4 \cdot q)$ (since there are at most t different verifiers). The probability that more than $3t^4 \cdot q$ failed trials will occur at any particular verifier's trials is bounded above by the probability that any particular V produces at least $3t^3 \cdot q$ valid and timely Step 4 messages out of at most $4t^4 \cdot q$. But the expected number of times V responds is bounded by $1/(2t^3 \cdot q) \cdot (4t^4 \cdot q) = 2t$. The multiplicative Chernoff bound states that if the random variable X is the sum of IID 0/1 variables, then for any $\gamma > 0$ we have:

$$\Pr[X \geq (1+\gamma)E[X]] \leq e^{-E[X]\cdot(\gamma^2)/2}.$$

Thus, the probability of a trial failure abort occurring is $2^{-\Omega(t)}$. Thus, adding up all the probabilities, we find that the probability of difference in observable behavior of the simulator and the "idealized" simulator S' is less than $1/q$, as desired. ■

6.4 Concurrent Zero-Knowledge Arguments

In this section, we show how to overcome the difficulty we saw in the previous section by switching to the context of computationally-sound proofs. We present concurrent zero-knowledge arguments (computationally-sound proofs) for DHC. We achieve perfect zero knowledge under the Discrete Log Assumption, and computational zero knowledge assuming only one-way functions.

We were only able to prove *weak* zero knowledge for the protocol from the previous section because, in the simulation, P's Step 3 message had to be changed according to V's future response in Step 4. This creates a dilemma for the simulator in the case where a verifier V fails to provide a valid Step 4 message: in this case, should the simulator ignore the verifier and simply go on with the simulation for the other verifiers, or rather should the simulator "rewind" the simulation and give the verifier V another chance to respond?

- If the simulator chooses the latter option, to rewind and give V another chance to respond, then we may encounter the same situation again if V still fails to provide a valid message. How many chances should we give V? There is a clear trade-off between the running time of the simulation and how certain we can be that V will never respond. In the previous section, we chose to give V a fixed polynomial number of chances to respond with a valid message. But, as we saw, this still left an inverse polynomial probability for certain verifiers that our simulation would fail to see a valid Step 4 message even though the verifier would give a valid message later on in the simulation.

- If the simulator instead chooses the first option, to ignore the verifier V and move on, then it may happen that later, another verifier V' provides a valid Step 4 message, which forces the simulator to "rewind" the simulation in order to change the Step 3 message to V'. At this point, the original verifier V may decide to produce a valid Step 4 message! However, in this case the simulator will not be prepared to answer V's challenge. Note that this is problematic exactly because the simulator had to "travel backwards in time" to "fix" an earlier message sent to V', at which point it became susceptible again to the whims of V.

In this section, we design new protocols to avoid this problem. In these protocols, the simulator only needs to change *future* messages based on the verifier's messages, thus avoiding the problem outlined above. For instance, in one protocol, the simulator chooses P's Step 6 response based on the verifier's Step 5 message. Of course, if the simulator were able to determine the Step 6 response based on a single Step 5 message of the verifier, then a cheating prover could do the same to fool the verifier. Thus, we design the protocol such that the simulator needs to obtain two distinct Step 5 messages from the verifier in order to be able to generate a Step 6 response that the verifier will accept.

It may seem that this simulator is no better, since it has to "travel backwards in time," as well. But the crucial difference between this new simulation strategy and the one from the previous section is that the simulator only needs to "rewind" the simulation in order to extract information, rather than to *change* earlier messages in the simulation. Once the information is extracted, the simulator discards the "rewinded" simulation and returns to the original simulation.

Thus, in the simulation, if a verifier V fails to provide a valid Step 5 message, the simulator can safely ignore it and move on. If later, when dealing with another verifier V', the simulator "rewinds" the simulation and V decides to provide a valid Step 5 message, the simulator can safely disregard this message, since the simulator will eventually return to the original simulation in which V had not given a valid Step 5 message. We use timing constraints to ensure that during an extraction, the simulator will not have to provide a Step 6 response to any verifier V that provides a valid Step 5 message during the extraction procedure. As a result, the simulator can build a transcript step-by-step, never needing to go back to change some part of the transcript constructed up to that point.

In the protocols we have seen up to this point, the simulations were based on the following basic observation about Blum's original protocol described in Section 6.1.1 (Protocol 6.1.1): *if the simulator knew what the verifier's query bits would be in advance, then it would be easy to simulate the protocol.* For the approach we will take in this section, we will make use of a different, even more basic, observation about Blum's protocol: *if the simulator could open the prover's commitments to arbitrary values, then it would be easy to simulate the protocol.* This observation is obvious but may also seem useless, since we know that commitments are supposed to be binding – one is not supposed to be able to open commitments to any value other than the one committed to. However, as we saw in Remark 6.2.2, there exist "trapdoor" commitment schemes, where given a trapdoor that is ordinarily kept secret, one can open commitments to arbitrary values. Such trapdoor commitment schemes are critical to the protocols of this section. Whereas in the previous section, the simulator tried to "extract" the verifier's queries, in this section, the simulators will try to "extract" the trapdoor of a commitment scheme.

6.4.1 Concurrent Perfect Zero-Knowledge Argument based on DLP

In the first protocol given below, we will use the SSCB (trapdoor) commitment protocol based on the discrete logarithm problem described in Construction 6.2.3.

Protocol 6.4.1 Concurrent perfect zero-knowledge argument for DHC **with timing**
A directed graph G is the common input to both parties. The prover is assumed to know a Hamiltonian cycle w in G.

1. *The verifier does the following: Choose random prime p of the form $p = 2q + 1$ (as described in Construction 6.2.3). Choose random g of order q in $\mathbb{Z}_p^*$. Choose random $a, a' \in \mathbb{Z}_q$. Set $h = g^a \bmod p_V$, $h' = g^{a'} \bmod p_V$. Set $K = (p, g, h)$. Send K and h' to prover.*
2. *The prover picks n random permutations $\pi_1, \ldots, \pi_n$ on the nodes of G. It then sends to the verifier commitments using the K commitment scheme to all the entries of the adjacency matrices of the permuted graphs $\pi_1(G), \ldots, \pi_n(G)$. The prover also sends a commitment $K(r_P)$, where $r_P \in_R \mathbb{Z}_q$.*
3. *The verifier sends $r_V \in_R \mathbb{Z}_q$.*
4. *The prover opens its commitment to r_P (which the verifier confirms).*
5. *The verifier does the following: It sets $r = r_V + r_P \bmod q$, and sends $c \overset{\text{def}}{=} (ra + a') \bmod q$ and a vector $\overrightarrow{\mathsf{query}} \in_R \{0,1\}^n$ of queries to prover.*
6. *In the final step, the prover checks that $h^r \cdot h' = g^c \bmod p$. If this is correct, for $i = 1, \ldots, n$:*
   - *If $\mathsf{query}_i = 0$, the prover opens its commitments from Step 3 to all the entries of the adjacency matrix of $\pi_i(G)$, and also sends a description of the permutation π_i. The verifier, upon receipt, verifies the correctness of the openings, and also verifies that the revealed adjacency matrix actually is the adjacency matrix of $\pi_i(G)$. If either of these conditions fail to hold, the verifier rejects the proof.*
   - *If $\mathsf{query}_i = 1$, the prover opens its commitments only to those entries in the adjacency matrix of $\pi_i(G)$ that correspond to edges in the Hamiltonian cycle $\pi_i(w)$. The verifier, upon receipt, verifies the correctness of the openings, and also verifies that the revealed entries of $\pi_i(G)$ do indeed form a Hamiltonian cycle. If either of these conditions fail to hold, the verifier rejects the proof.*

Timing Constraints:

1. *P requires the Step 5 message be received within α local time from receipt of the Step 1 message;*
2. *P delays the Step 6 message until at least β local time has elapsed since the receipt of the verifier's Step 5 message.*

For the zero knowledge guarantee of this protocol, we require that the adversary be constrained by the (α, β)-constraint.

Theorem 6.4.1 *Protocol 6.4.1 is a perfect concurrent zero-knowledge argument for* DHC, *assuming the weak synchronization assumption for (α, β) and the Discrete Logarithm Assumption.*

Proof: As usual, completeness follows by inspection.

Soundness. The proof relies on the fact that, if some cheating prover P^* convinces V of a false statement with non-negligible probability, then by exploring the answers to two different query vectors (Steps 5 and 6), it is possible to obtain two different openings of a "committed" value.

Assume for the sake of contradiction that some non-uniform probabilistic polynomial time bounded cheating P^* can convince V of a false theorem with non-negligible probability ρ. Note that we can assume that P^* is deterministic without loss of generality, because we allow it to be non-uniform[14]. We argue that there exists a probabilistic polynomial time bounded M that, by interacting with P^*, can break the DLA with probability $\Omega(\rho^6)$, which is also non-negligible.

On input $p, q, g, h = g^a$, where g generates an order q subgroup of $\mathbb{Z}_p^*$, the machine M (which is trying to find a) interacts with P^* as follows.

Step 1. M chooses $c, r \in_R \mathbb{Z}_q$. M sends $K = (p, g, h)$ and $h' = g^c h^{-r}$.

Step 2. P^* sends commitments using K to certain adjacency matrices, and sends a commitment $K(r_P)$ to r_P. At this point, M saves the state of P^* for later use.

Step 3. M chooses $r_V \in_R \mathbb{Z}_q$ and sends it to P^*.

Step 4. P^* opens its commitment to r_P. At this point M resets the state of P^* to just after Step 2, sets $r_V = r - r_P \bmod q$, and M re-sends its original Step 3 message, modified substituting the new value of r_V:

Step 3'. M sends $r_V = r - r_P \bmod q$. Note that since r was chosen uniformly at random, conditioned on r_P, this new value of r_V is still distributed uniformly.

Assuming that P^* responds to the Step 3' message, one of two things can happen: either P^* changes the way it opens its commitment to r_P or not. If P^* changes its opening of r_P, then from these two openings of the Step 2 commitment, M can immediately derive the discrete logarithm a.

Otherwise, the following occurs:

Step 4'. P^* opens r_P as before, such that $r = r_V + r_P \bmod q$. At this point, M saves the state of P^* for later use.

Step 5. M sends c (and note that we have arranged so that $h^r \cdot h' = g^c$). and sends a query vector $\overrightarrow{\text{query}} \in_R \{0,1\}^n$.

Step 6. The check that P^* is supposed to do would succeed by definition of h'. P^* replies to the queries. M resets the state of P^* to just after Step 4'.

Step 5'. M sends c as before. Then, M chooses a new $\overrightarrow{\text{query}}' \in_R \{0,1\}^n$ and sends $\overrightarrow{\text{query}}'$ to P^*. Note that with overwhelming probability, $\overrightarrow{\text{query}}' \neq \overrightarrow{\text{query}}$.

Step 6. P^*'s check succeeds by definition of h'. P^* replies to the new queries. M now has the reply to two different queries. If both queries were answered by P^* in a manner that convinces V to accept, and yet G does not contain a Hamiltonian cycle, it must be that P^* has provided the opening of some commitment under K (made in Step 2) to two different values. From this M can extract a.

We now analyze the probability that M succeeds in obtaining the discrete logarithm a. The (P^*, V) interaction can be viewed as a tree with probability ρ of success ("success" is when P^* provides messages which cause the verifier to accept). Recall that we may assume that P^* is deterministic. Note that M's choices of p, g, h, and h' are chosen randomly but remain fixed throughout the interaction. Now, with probability at least $\rho/2$ over these choices (which are sent in Step 1), P^* will still have a probability of at least $\rho/2$ of convincing the verifier to accept[15]. Let us assume that

[14]Thus, we can include as part of P^*'s description the "best" setting for its random coins. For more discussion and examples of this type of standard argument, see for example [Gol95].

[15]Note that if this were not the case, it would mean that with probability $(1 - \rho/2)$, there would be a probability of at most $\rho/2$ of success, while with probability $\rho/2$, there is probability at most 1. But $(1 - \rho/2) \cdot (\rho/2) + (\rho/2) \cdot 1 < \rho$, which is a contradiction of our assumption that P^* has probability ρ of convincing the verifier to accept. We will use this type of argument often in this proof.

we are in the good case, where P^* has a probability $\rho/2$ of causing the verifier to accept conditioned on these choices.

Then, with probability at least $\rho/4$ (over the choices of r_V sent in Step 3), it must be that P^* will respond to the Step 3 message of M in a way such that P^* still has a probability of $\rho/4$ of convincing the verifier to accept conditioned on its Step 3 response (which in particular implies that P^*'s Step 4 response is well-formed and valid). Thus, the probability that M chooses two Step 3 messages (once before and once after resetting the state of P^*) which lead to a state where P^* still has a $\rho/4$ chance of success is at least $(\rho/4)^2$, conditioned on being in a probability $\rho/2$ state after the choices of p, g, h, and h'.

Assume that this is the case. At this point, M may have already succeeded in extracting a. Let us assume that we are in the bad case, where M has not yet succeeded in learning a, *i.e.* that P^*'s opening of r_P does not change in Step 4'. Again, note that r and c, though randomly selected, are not changed once selected. We know that with probability $\rho/8$ over these choices, P^* will have a probability at least $\rho/8$ of still convincing the verifier to accept conditioned on these choices. Thus, finally, the probability that M chooses two Step 5 messages to which P^* responds correctly is at least $(\rho/8)^2$. In this, case M succeeds in obtaining the discrete logarithm a. Thus, the overall probability of extracting a is at least

$$\frac{\rho}{2} \cdot \frac{\rho^2}{16} \cdot \frac{\rho}{8} \cdot \frac{\rho^2}{64} = \Omega(\rho^6)$$

minus a negligible probability of 2^{-n} that $\overrightarrow{\mathsf{query}} = \overrightarrow{\mathsf{query}}'$ during the (P^*, M) interaction.

Ordinary Zero Knowledge. Before proving concurrent zero knowledge we outline the simulation technique for proving ordinary zero knowledge in the standard model, without timing. Note that here, unlike in the protocol of the previous section, *we make no simplifying assumptions on the behavior of the verifier.* Suppose we have an arbitrary (polynomial-time) verifier V^*. The (traditional) black-box simulation for V^* works as follows.

Step 1. The simulator receives the first message from V^*, in which it sends the paramenters for the commitment scheme K with trapdoor a (unknown to the simulator), along with h'. If the verifier fails to provide a message, the simulator outputs the empty transcript and terminates. The simulator saves the state S of V^* at this point for later use.

Step 2. The simulator performs the second step exactly as an honest prover would, sending commitments to permutations of the graph along with a commitment under K to r_P, chosen randomly.

Step 3. The simulator receives r_V from the verifier. If the verifier fails to provide a message, the simulator outputs the transcript so far and terminates.

Step 4. Again, the simulator behaves just as the honest prover would, revealing its commitment to r_P. Note that by the fact that the Step 2 commitment to r_P was truly independent of r_P, and the fact that r_P was chosen randomly, $r = r_V + r_P \bmod q$ is exactly uniformly distributed in $\mathbb{Z}_q$ regardless of the verifier's actions so far.

Step 5. The verifier sends $c = (r \cdot a + a') \bmod q$, where $r = r_V + r_P \bmod q$, with exactly the same conditional probability that it does in real executions (the conditioning is on the state S of the verifier after Step 1). A *valid* Step 5 message is a syntactically correct message that passes the test in Step 6, that $h^r \cdot h' = g^c$. If the verifier sends no message or sends an invalid message, then the simulator outputs the transcript (including the invalid Step 4 message, if any) and halts. If the verifier does send a valid message, then the simulator saves the state S' of the verifier and the transcript T so far, and then executes the following extraction procedure:

Extraction: Interleave the following two procedures in parallel until the first one halts.

1. Repeat until the simulator discovers the trapdoor a:
   (a) Reset the state of the verifier to just after Step 1.
   (b) Repeat the simulation strategy above, selecting a new $r'_P \in_R \mathbb{Z}_q$ each time. Note again that by the statistical secrecy property of K, regardless of the verifier's choice of r'_V in Step 3, after Step 4, $r' = r'_V + r'_P$ will be uniformly distributed in $\mathbb{Z}_q$. Thus, the probability that $r = r'$ is exactly $1/q$.
   (c) If the verifier provides a valid Step 5 response (c') to a choice of $r' \neq r$, then the simulator, by solving the equations $c = ra + a'$ and $c' = r'a + a'$ (both modulo q) for a and a', can find the trapdoor a.
   (d) If the verifier fails to provide a valid Step 5 response, or if $r' = r$, then the simulator repeats this process.
2. Find a by brute force, trying all q different possibilities in the equation $g^{(\cdot)} = h$.

Step 6: Now that the simulator knows the trapdoor a, it restores the verifier to state S' and continues building on the transcript T: The simulator now replies to the Step 5 queries of V^* as necessary using the trapdoor in order to perfectly simulate the distribution that the prover would provide in the protocol.

This completes the description of the simulator. By construction, we see that the simulator perfectly simulates the protocol with no error. We now compute the running time of the simulator. Let η be the probability, conditioned on the state S of the verifier after Step 1, that the verifier will send a valid Step 5 message in an execution of the protocol — that is, that the Extraction procedure is executed. Let ξ be the probability that the verifier responds with a valid Step 5 message during extraction such that $r' \not\equiv r \bmod q$.

The probability of choosing an $r' = r \bmod q$ is exactly $\frac{1}{q}$, thus $\xi \geq \eta - \frac{1}{q}$. We consider two cases:

- $\eta > \frac{2}{q}$: In this case $\eta - \frac{1}{q} > \frac{\eta}{2}$ and so $\xi \geq \frac{\eta}{2}$ and the expected number of trials during extraction is at most $\frac{2}{\eta}$. Recall that the simulation only enters the extraction phase with probability η. Thus the overall expected number of trials is at most $\eta\frac{3}{\eta} = 3$, leading to an expected $O(1)$ trials in the extractions overall.
- $\eta \leq \frac{2}{q}$: In this case the brute force search succeeds in $O(q)$ steps, for a total of an expected $O(\eta q) = O(1)$ steps.

Thus, we have that the simulation is expected polynomial time.

Concurrent Zero Knowledge We have arranged so that the proof of concurrent zero knowledge is almost exactly the same as the proof for ordinary zero knowledge. We redefine a *valid* Step 5 message to specify that the message must also satisfy the timing constraint. For a verifier V_i, we redefine η to be the probability, conditioned on the state S of the adversary through the Step 1 message of V_i, that the extraction procedure will be executed. Finally, note that the timing constraints lead to an Interleaving Constraint analogous to the one from Protocol 6.3.1. This interleaving constraint ensures that during the interaction with any verifier V_i the simulator never needs to provide a Step 6 message to any V_j that sent its Step 5 message after V_i sent its Step 1 message. Thus, during the extraction of V_i, the simulator will only need to provide Step 6 answers[16] for verifiers whose

[16]Recall that Step 6 is the only step that requires special information to simulate perfectly.

trapdoor information has *already* been extracted. Thus the extraction will never get stuck because of another verifier.

The simulation is constructed message by message. If in some (P, V_i) interaction the adversary sends a valid Step 5 message for V_i, then the extraction procedure is executed to obtain V_i's trapdoor information a_i. As noted above, the timing constraints ensure that the extraction procedure can be carried out. This completes the proof of concurrent zero knowledge. ■

6.4.2 Concurrent Zero-Knowledge Under General Assumptions

In this section we present a protocol that relies only on the existence of one-way functions. We observe that the critical element of the protocol given in the previous section is the trapdoor commitment scheme, with a protocol in which the trapdoor can be extracted by the simulator. This idea had been used before to construct ordinary zero-knowledge arguments. We look at the protocol of Feige and Shamir [FS89], and show that by introducing timing requirements and using a particular simulator, this protocol can be proven to be concurrent zero-knowledge.

We first present the protocol assuming that one-to-one one-way functions exist, in which case the protocol has four moves. Note that this gives a kind of "converse" to the result of [KPR98], showing that four-round concurrent zero-knowledge arguments are possible in the timing model. We later indicate how to transorm this protocol into a five-move protocol, for which we need only assume that one-way functions exist (that are not necessarily one-to-one).

The Feige-Shamir trapdoor commitment scheme. The central tool used in the protocol will be a trapdoor commitment scheme based on the basic zero-knowledge proof for DIRECTED HAMILTONIAN CYCLE of Blum [Blu86], introduced in [FS89]. This commitment scheme will use an CSSB commitment scheme C as a subprocedure. The trapdoor commitment scheme will be based on a graph I (say with q nodes) that has a Hamiltonian cycle which cannot be determined in polynomial time. This graph will be chosen using a reduction from a one-way function, so that finding a Hamiltonian cycle in the graph is as difficult as inverting the one-way function (described in detail below). The main property of the commitment scheme is that one can create commitment strings that are indistinguishable from normal commitments, such that given a Hamiltonian cycle for I, one can decommit to both a 0 and a 1. The commitment scheme C_I works as follows:

- To commit to a 0, the sender picks a random permutation π of the nodes of I, and commits to the entries of the adjacency matrix of the permuted graph one by one, using C. To decommit, the sender reveals π and opens every entry of the adjacency matrix.
- To commit to a 1, the sender commits to the entries of an all-ones adjacency matrix (corresponding to the complete graph on q nodes) using C. To decommit, the sender reveals a random Hamiltonian cycle in the adjacency matrix.

This commitment scheme certainly has the property of being computationally secret, *i.e.* the distributions $C_I(0)$ and $C_I(1)$ are computationally indistinguishable for any graph I. Note also that given a Hamiltonian cycle in I, one can open commitments to 0 as both 0 and 1. Note that the converse is also true, namely that given the opening of any commitment as both a 0 and a 1, one can extract a Hamiltonian cycle of I.

Picking a suitable graph for the commitment scheme. For the commitment scheme above to be binding, it must be the case that given the graph I, it is difficult to find a Hamiltonian cycle in I. To ensure that this is the case, we use a reduction from a one-way function to produce the graph. Let $f : \{0,1\}^* \to \{0,1\}^*$ be any length-preserving (strong) one-way function. Consider the following language:

$$L = \{(y_1, y_2) : \text{there exists } x \in \{0,1\}^* \text{ such that either } f(x) = y_1 \text{ or } f(x) = y_2\}$$

Clearly, $L \in \mathsf{NP}$. By the theory of NP-completeness [Coo71, Lev73], because DIRECTED HAMILTONIAN CYCLE is NP-complete [Kar72], there is a polynomial-time reduction mapping instances of L to instances of DHC that also gives a witness mapping in the other direction. More precisely, there is a polynomial-time computable function g which takes as input an instance (y_1, y_2) of L, and produces as output a graph I with the property that $I \in \text{DHC} \iff (y_1, y_2) \in L$, together with a circuit C (called the *witness mapper*). C takes as input the description w of a cycle in I, and if w is a Hamiltonian cycle in I, it outputs an x such that either $f(x) = y_1$ or $f(x) = y_2$. Furthermore, g does not reduce the number of witnesses – if for some instance (y_1, y_2), there are two values x_1 and x_2 such that $f(x_1) = y_1$ and $f(x_2) = y_2$, then there are at least two distinct Hamiltonian cycles in G.

Thus, to choose a suitable graph for the commitment scheme, for a given n, we choose two random values[17] $x_1, x_2 \in_R \{0,1\}^n$, and let $y_1 = f(x_1)$, and $y_2 = f(x_2)$. We apply the reduction g on (y_1, y_2) to obtain a graph I on poly(n) vertices, together with the circuit C. Note again, that by the properties of g, if one finds a Hamiltonian cycle in I, using C this would yield a preimage x of either y_1 or y_2 under the one-way function f. By assumption, any non-uniform polynomial-time machine could accomplish this goal with only negligible probability over the choices of x_1 and x_2.

We are now ready to give the protocol. In this protocol, we assume that C is the one-round CSSB commitment scheme given in Construction 6.2.1.

Protocol 6.4.2 Concurrent Computational ZK Argument for DHC **- General Assumptions**
A directed graph G is the common input to both parties. The prover is assumed to know a Hamiltonian cycle w in G.

1. $V \longrightarrow P$: *Use the method described above to construct graph I on $q = poly(n)$ nodes containing two distinct Hamiltonian cycles w_1 and w_2. Choose $m = q^2$ random permutations $\phi_1, \ldots, \phi_m$ on q elements. Send I, and entry-by-entry commitments using C to the adjacency matrices of $\phi_1(I), \ldots, \phi_m(I)$.*

2. $P \longrightarrow V$: *Prover must perform the following two actions:*

    (a) *Send $r = (r_1, \ldots, r_m) \in_R \{0,1\}^m$.*

    (b) *Choose m random permutations $\psi_1, \ldots, \psi_m$ on n elements. Send entry-by-entry commitments using C_I (as described above) to the adjacency matrices of $\psi_1(G), \ldots, \psi_m(G)$.*

3. $V \longrightarrow P$: *Verifier must perform the following two actions:*

    (a) *For $i = 1, \ldots, m$:*

    - *If $r_i = 0$, open commitments to entire adjacency matrix of $\phi_i(I)$ and reveal ϕ_i. (Prover confirms consistency.)*

[17] It may seem strange that we insist on having two possible witnesses. This is important in the proof of soundness given by Feige and Shamir [FS89] for the protocol we build upon. For details, see [FS89]. This is not important to our proof of concurrent zero knowledge.

- *If $r_i = 1$, open commitments only to entries in adjacency matrix of $\phi_i(I)$ that correspond to the Hamiltonian cycle $\phi_i(w_1)$. (Prover confirms that entries revealed are all* 1 *and form a Hamiltonian cycle.)*

(b) Send $R = (R_1, \ldots, R_m) \in_R \{0,1\}^m$

4. $P \longrightarrow V$ *: For* $i = 1, \ldots, m$*:*

- *If $R_i = 0$, open commitments (as described above) to entire adjacency matrix of $\psi_i(G)$ and reveal ψ_i. (Verifier confirms consistency.)*
- *If $R_i = 1$, open commitments (as described above) only to entries in adjacency matrix of $\psi_i(G)$ that correspond to the Hamiltonian cycle $\psi_i(w)$. (Verifier confirms that entries revealed are all* 1 *and form a Hamiltonian cycle.)*

Timing Constraints*:*

1. *The Step 3 messages must be received within time α of receipt of the Step 1 message.*
2. *The Prover waits until at least time β has elapsed since receipt of the Step 3 messages before sending the Step 4 message.*

Theorem 6.4.2 *Protocol 6.4.2 is a concurrent computational zero-knowledge argument for* DHC, *assuming the weak synchronization assumption for (α, β) and the existence of one-to-one one-way functions.*

Proof: As usual, completeness is clear. Soundness follows from the soundness of the protocol without timing constraints as proved in [FS89].

Ordinary Zero Knowledge. The proof we present here follows the argument given by [FS89], and is analogous to the proof given in the previous subsection. We will then show how the timing constraints allow the simulation to be extended to the concurrent scenario, much as in Protocol 6.4.1. Suppose we have an arbitrary (polynomial-time) verifier V^*. The (traditional) black-box simulation proceeds as follows:
Step 1. The simulator obtains the Step 1 message of the verifier, obtaining the graph I together with entry-by-entry commitments to m adjacency matrices. The simulation saves the state S of the verifier after its first message has been sent.
Step 2. For part (a), the simulator chooses $r \in_R \{0,1\}^m$, just as the real prover would. For part (b), the simulator commits (using C_I) to all 0's for each of the m adjacency matrices. Note that so far, the only difference between simulated and real interactions is that the simulator commits to all 0's, while the real prover may not.
Step 3. We define a *valid* Step 3 message to be a syntactically well-formed message such that the prover's verification checks pass for part (a). If the verifier fails to send a Step 3 message in time, or if it sends an invalid message, the simulation ends and the partial transcript so far (including the invalid Step 3 message, if any) is output. Otherwise, the simulator stores the part (a) openings of commitments to adjacency matrices (and possibly associated permutations) for r, along with R from part (b). The simulator then saves the state S' of the verifier and the transcript T so far. It then executes the following procedure in order to extract a Hamiltonian cycle in I:
Extraction: Interleave the following two procedures in parallel until the first one halts.

1. Repeat until verifier obtains a Hamiltonian cycle in I:

   (a) Reset the state of verifier to just after Step 1.
   (b) Repeat Step 2 exactly as above (picking new random $r' \in_R \{0,1\}^m$).
   (c) If verifier replies for $r \neq r'$, there is some index i such that $r_i \neq r'_i$. Thus, by examining the openings of the commitments to the i'th adjacency matrix, we obtain a Hamiltonian cycle in I, and stop.

2. Find a Hamiltonian cycle in I by brute force. Note that this procedure takes $q! = O(2^m)$ time.

There are two possibilities: First, it is possible the brute force search completes and discovers there is no Hamiltonian cycle in I. Note that this event can only occur with probability 2^{-m}, since if I has no Hamiltonian cycle, then there is only at most one value for r which could admit a valid verifier Step 3 response. In this case, the simulation aborts. The other case is that one of the two procedures uncovers a Hamiltonian cycle c in I. This allows the simulator to open its commitments from Step 2b to any desired value in a manner (computationally) indistinguishable from what occurs in actual protocols.

Step 4: The simulator resets the state of the verifier to S' (after the first valid Step 3 message was received) and adds to the transcript T as follows: For each $i = 1, \ldots, m$, the simulator picks a random permutation ψ_i on n elements. If $R_i = 0$, the simulator uses c to open the i'th adjacency matrix as $\psi_i(G)$ and reveals ψ_i. If $R_i = 1$, the simulator uses c to open a random circuit as all 1's. The simulation ends, and the transcript prepared is output.

This completes the description of the simulator. First, we observe that because $C_I(0)$ and $C_I(1)$ are computationally indistinguishable (and the simulation aborts with only negligible probability), the output of the simulation is computationally indistinguishable from transcripts from actual interactions.

We now compute the running time of the simulation. Certainly all time spent outside of the extraction procedure is polynomially bounded. We now show that the expected running time of the extraction is also polynomially bounded: Let η be the probability conditioned on state S of the verifier (after Step 1), that the verifier will send a valid Step 3 message after the simulated Step 2. (Note that we are considering *simulated* Step 2, not actual Step 2, where the commitments would differ.) Thus, conditioned on S, η is the probability that the Extraction procedure is executed. Let ξ be the probability that the verifier responds with a valid Step 3 message during extraction where $r' \neq r$. Thus, $\xi \geq \eta - 2^{-m}$. We now consider two cases:

- Case $\eta \leq 2 \cdot 2^{-m}$: In this case, since the brute force search will end in at most 2^m steps, the expected number of steps spent in the extraction procedure overall is bounded by $\eta \cdot 2^m \leq 2$.

- Case $\eta > 2 \cdot 2^{-m}$: Then, $\xi > \eta/2$. Hence, the expected number of trials during extraction is at most $\eta \cdot 1/\xi \leq 2$.

Concurrent Zero Knowledge The extension of the proof to concurrent zero knowledge is almost exactly the same as in Protocol 6.4.1. We note that again, the timing constraints ensure that during the interaction with any verifier V_i the prover never needs to provide a Step 4 message to any V_j that sent its Step 3 message after V_i sent its Step 1 message. Thus, during the extraction of V_i, the simulator will only need to provide Step 4 answers[18] for verifiers whose trapdoor information has *already* been extracted. Thus the extraction procedure will never get stuck because of another verifier.

[18]Recall that Step 4 is the only step that requires special information to simulate.

The simulated transcript is constructed message by message. If in some (P, V_i) interaction the adversary sends a valid Step 3 message for V_i, then the extraction procedure is executed to obtain the Hamiltonian cycle for the graph that V_i sent. As noted above, the timing constraints ensure that the extraction procedure can be carried out. This completes the proof of concurrent zero knowledge. ■

If we assume only that one-way functions exist, then we can substitute the basic commitment scheme C with the 2-round interactive commitment scheme of Naor [Nao91] (this commitment scheme is described in Construction 6.2.2). This requires one additional setup round for the commitment scheme. The proof of concurrent zero-knowledge remains the same, except for one caveat: In the scheme of Naor, there is an exponentially small probability, over the setup round, that a committer could produce a commitment that can be opened both as a 0 and as a 1. If the simulator ever observes the verifier open a commitment as both a 0 and a 1, it simply aborts. Since this can only occur so rarely, it does not affect the quality of the simulation.

Chapter 7

Conclusions

In this thesis, we have explored two frontiers of research on zero knowledge:

Statistical Zero Knowledge. In the first part of this thesis (Chapters 3, 4, and 5), we took a complexity-theoretic perspective and studied statistical zero-knowledge proofs in detail. In Chapter 3, we introduced the problem STATISTICAL DIFFERENCE, and showed that it is complete for the complexity class HVSZK of all problems that admit honest-verifier statistical zero-knowledge proofs. This provided a new characterization of the class which made no reference to interaction or zero knowledge. This complete problem gave us a unifying framework under which to study HVSZK. We were able to give simplified proofs for nearly all previously known results about HVSZK, as well as establish several new interesting properties.

In Chapter 4, we gave a transformation that allowed us to transform any proof system that is statistical zero knowledge just for the honest verifier, into one that is statistical zero knowledge for every verifier. This in particular showed that the class of problems that admit honest-verifier statistical zero-knowledge proofs is no bigger than the class of problems admitting (general) statistical zero-knowledge proofs, *i.e.* HVSZK = SZK. Thus, the many properties we had established about HVSZK in Chapter 3 in fact carry over to the general class SZK. In particular, this showed that the problem STATISTICAL DIFFERENCE characterizes all problems admitting statistical zero-knowledge proofs.

Finally, in Chapter 5, we extended our theory to cover the non-interactive form of statistical zero-knowledge proofs, as well. We showed that the class NISZK of problems admitting non-interactive statistical zero-knowledge proofs has complete problems that are very closely related to STATISTICAL DIFFERENCE, the complete problem for SZK. Using this fact, we were able to obtain results relating the interactive and non-interactive forms of statistical zero-knowledge proofs.

Although our investigation has shed some light on the nature of statistical zero-knowledge proofs, many open problems remain:

1. **Other complete problems for SZK.** We showed that STATISTICAL DIFFERENCE is complete for SZK, and [GV99] have shown that the related problem ENTROPY DIFFERENCE is also complete for SZK. Are there other problems, graph-theoretic or number-theoretic in nature, that are complete for SZK?

2. **Number of rounds necessary for general statistical zero-knowledge proofs.** In Chapter 3, we showed that every problem in HVSZK has a 2-message honest-verifier statistical zero-knowledge proof. While the transformation we gave in Chapter 4 from honest-verifier to

general statistical zero-knowledge proofs only increases the number of rounds in the protocol by a constant factor, it only applies to public-coin protocols, whereas our protocol from Chapter 3 was a private-coin protocol. The transformation of Okamoto [Oka96] from private-coin to public-coin protocols increases the number of rounds by a polynomial factor. Is this necessary? Is it possible that every problem in SZK has a constant-round general statistical zero-knowledge proof system? A constant-round public-coin such proof system?

3. **Relationship of NISZK and SZK.** We showed in Chapter 5 that if NISZK is closed under complement, then NISZK = SZK. Is this the case? Recall that if this is the case, then it would imply that the answers to the open problems from the previous item would be positive.

4. **Other types of zero knowledge.** Is it possible to further extend the techniques we have used in examining statistical zero-knowledge proofs to other types of zero-knowledge protocols, such as computational zero-knowledge proofs, or zero-knowledge computationally-sound proofs (also called arguments)? In particular, can one exhibit a problem that is unconditionally complete for the class of problems admitting computational zero-knowledge proofs, or even honest-verifier computational zero-knowledge proofs? We showed in Chapter 4 how to transform public-coin honest-verifier computational zero-knowledge proofs into general computational zero-knowledge proofs. Can one do this for all honest-verifier computational zero-knowledge proofs unconditionally?

5. SZK **vs.** PZK. We saw in Chapter 3 that the "perfect knowledge complexity" of problems in SZK is extremely low – in fact, when interpreted in the entropy sense, we saw that every problem in SZK has a proof system for which there is a simulator which only needs 2^{-n^c} bits of information from an oracle in order to be able to simulate the protocol *perfectly*. Thus, the "gap" between perfect zero-knowledge proofs and statistical zero-knowledge proofs is very small. Is there actually no gap, *i.e.* , is it the case that PZK = SZK? If one could exhibit a perfect zero-knowledge proof for STATISTICAL DIFFERENCE, this would settle this question in the affirmative.

6. **Power of the Prover.** The transformation we gave in Chapter 4 from public-coin honest-verifier to general zero-knowledge proofs had the interesting feature of preserving the power of the prover. Unfortunately, the transformation of [Oka96] from private-coin to public-coin honest-verifier statistical zero-knowledge proofs does not have this property, and in fact even the transformation of Goldwasser and Sipser [GS89] from private-coin to public-coin proofs does not. Must all general transformations from private-coin to public-coin proofs require an increase in the power of the prover? Recently, Vadhan [Vad00] gave some evidence that this may be true, by analyzing a special case of the complete problem STATISTICAL DIFFERENCE presented in Chapter 3 to rule out a wide class of transformations. Is there an unconditional transformation from private-coin honest-verifier to general zero-knowledge proofs which preserves the power of the prover? On a related note, how much computational power is required by the prover in statistical zero-knowledge proofs?

Concurrent Zero Knowledge. In the second part of this thesis (Chapter 6), we took a cryptographic perspective, and considered the problem of maintaining zero knowledge in a distributed environment where many parties may act concurrently. In this setting, we introduced the notion of a *concurrent zero-knowledge* proof — namely, a protocol that can be carried out by a single pair of parties oblivious to other parties in the system, which nevertheless remains zero knowledge even when faced with a coordinated attack by many verifiers. We introduced a notion of local timing constraints and gave general techniques for using them to achieve constant-round concurrent zero-knowledge computationally-sound proofs, which we illustrated with such proofs for an NP-complete problem under various assumptions.

The work in this thesis was the first to consider the problem of concurrent zero knowledge, and much work has followed. However, research on concurrency and cryptographic protocols remains in its infancy, and many fundamental problems remain open:

1. **Constant-round concurrent zero knowledge without timing.** Is it possible to obtain constant-round concurrent zero-knowledge proofs or arguments without using timing constraints? We have shown how to construct 4-round arguments based on standard intractability assumptions with timing, and [KPR98] have given an impossibility result for up to 5-round protocols without timing.

2. **Addressing all concerns of concurrency together.** In this thesis, we considered only the zero knowledge requirement in a distributed setting, but there are many other security notions that require attention in a distributed setting (see the remarks of Section 6.2.4). [DDN91], for example, considered the problem of malleability for zero-knowledge proofs in a distributed setting. Can one build efficient protocols that address all these concerns simultaneously?

3. **Other uses of timing constraints.** In this thesis, we examined using local timing constraints to aid us in achieving concurrent zero knowledge. Are there other cryptographic uses for such local timing constraints, especially in distributed settings?

Bibliography

[ABV95] William Aiello, Mihir Bellare, and Ramarathnam Venkatesan. Knowledge on the average—perfect, statistical, and logarithmic. In *Proceedings of the Twenty Seventh Annual ACM Symposium on the Theory of Computing*, 1995.

[AH91] William Aiello and Johan Hastad. Statistical zero-knowledge languages can be recognized in two rounds. *Journal of Computer and System Sciences*, 42(3):327–345, June 1991.

[AB84] Miklos Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth computations. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 471–474, Washington, D.C., 1984.

[ALM$^+$92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the Thirty Third Annual Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs. In *Proceedings of the Thirty Third Annual Symposium on Foundations of Computer Science*, pages 2–13, 1992.

[BM88] Lászlő Babai and Shlomo Moran. Arthur-Merlin games: A randomized proof system and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[Bel97] Mihir Bellare. A note on negligible functions. Technical Report CS97-529, Department of Computer Science and Engineering, University of California at San Diego, March 1997. Also available from the Theory of Cryptography Library (`http://theory.lcs.mit.edu/~tcryptol`).

[BGS97] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Personal communication, June 1997.

[BG89] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 194–211. Springer-Verlag, 1990, 20–24 August 1989.

[BMO90] Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. Perfect zero-knowledge in constant rounds. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 482–493, 1990.

[BMO90] Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. The (true) complexity of statistical zero-knowledge. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 494–502, 1990.

[BP94] Mihir Bellare and Erez Petrank. Making zero-knowledge provers efficient. In *Proceedings of the Twenty Sixth Annual ACM Symposium on the Theory of Computing*, 1994.

[BR90] Mihir Bellare and Phillip Rogaway. Non-interactive perfect zero-knowledge. Unpublished manuscript, June 1990.

[BGG$^+$88] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56. Springer-Verlag, 1990, 21–25 August 1988.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 1-10, 1988.

[BD91] T. Beth and E. Desmedt. Identification Tokens - or: Solving the Chess Grandmaster Problem. In *Advances in Cryptology–CRYPTO '90*, Lecture Notes in Computer Science, Vol. 537, Springer-Verlag, 1991, pp. 169–177.

[Blu86] Manuel Blum. How to Prove a Theorem So No One Else Can Claim It. *Proceedings of the International Congress of Mathematicians*, Berkeley, California, USA, 1986, pp. 1444-1451.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, December 1991.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, Illinois, 2–4 May 1988.

[BHZ87] Ravi B. Boppana, Johan Hastad, and Stathis Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25:127–132, 1987.

[BC94] S. Brands and D. Chaum. Distance-Bounding Protocols. In *Advances in Cryptology - EUROCRYPT'93*, 1993, Lecture Notes in Computer Science, Vol. 765, Springer Verlag, 1994, pp. 344–359.

[BCC88] Gilles Brassard, David Chaum, and Claude Crepeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156-189, October 1988.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable Zero-Knowledge. ECCC Report TR99-042, revised June 2000. Available from `http://www.eccc.uni-trier.de/eccc/`.

[CvHP91] D. Chaum, E. van Heijst and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In *Advances in Cryptology - CRYPTO'91, Lecture Notes in Computer Science* 576, Springer Verlag, pages 470–484, 1992.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Conference Record of Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley Series in Telecommunications. John Wiley & Sons, Inc., 2nd edition, 1991.

[Dam94] Ivan Damgård. Interactive hashing can simplify zero-knowledge protocol design. In *Proceedings of Crypto '95, Lecture Notes in Computer Science*, volume 403, pages 100–109. Springer-Verlag, 1994.

[DC96] Ivan Damgård and Ronald Cramer. On monotone function closure of perfect and statistical zero-knowledge. Theory of Cryptography Library: Record 96-03, 1996. `http://theory.lcs.mit.edu/~tcryptol`.

[DGOW95] Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. Honest verifier vs. dishonest verifier in public coin zero-knowledge proofs. In *Proceedings of Crypto '95, Lecture Notes in Computer Science*, volume 403. Springer-Verlag, 1995.

[DGW94] Ivan Damgård, Oded Goldreich, and Avi Wigderson. Hashing functions can simplify zero-knowledge protocol design (too). Technical Report RS-94–39, BRICS, November 1994. See Part 1 of [DGOW95].

[DDP94] Alfredo De Santis, Giovanni Di Crescenzo, and Guiseppe Persiano. The knowledge complexity of quadratic residuosity languages. *Theoretical Computer Science*, 132(1–2):291–317, 26 September 1994.

[DDP97] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Randomness-efficient non-interactive zero-knowledge (extended abstract). In Pierpaolo Degano, Robert Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium*, volume 1256 of *Lecture Notes in Computer Science*, pages 716–726, Bologna, Italy, 7–11 July 1997. Springer-Verlag.

[DDPY94] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. On monotone formula closure of SZK. In *Proceedings of the Thirty Fifth Annual Symposium on Foundations of Computer Science*, pages 454–465, 1994.

[DDPY98] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. Image Density is complete for non-interactive-SZK. In *Automata, Languages and Programming, 25th International Colloquium*, Lecture Notes in Computer Science, pages 784–795, Aalborg, Denmark, 13–17 July 1998. Springer-Verlag. See also [DDPY99].

[DDPY99] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. Image Density is complete for non-interactive-SZK, May 1999. Preliminary draft of full version.

[DMP87] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *Advances in Cryptology—CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 52–72. Springer-Verlag, 1988, 16–20 August 1987.

[DMP88] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge with preprocessing. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 269–282. Springer-Verlag, 1990, 21–25 August 1988.

[DO99] Giovanni Di Crescenzo and Rafail Ostrovsky. On Concurrent Zero-Knowledge with Pre-processing. In *Advances in Cryptology — CRYPTO'99*, Lecture Notes in Computer Science, pages 485–502. Springer-Verlag, 1999.

[DOY97] Giovanni Di Crescenzo, Tatsuaki Okamoto, and Moti Yung. Keeping the SZK-verifier honest unconditionally. In Burton S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 31–45. Springer-Verlag, 17–21 August 1997.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, Louisiana, 6–8 May 1991.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 409–418, 1998.

[DS98] Cynthia Dwork and Amit Sahai. Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. In *Advances in Cryptology — CRYPTO '98*, Lecture Notes in Computer Science, pages 442–457. Springer-Verlag, 1998.

[ESY84] Shimon Even, Alan L. Selman, and Yacov Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, May 1984.

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77-94, 1988.

[FGL$^+$96] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. In *Journal of the Association for Computing Machinery*, 43(2):268–292, 1996.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science*, volume I, pages 308–317, St. Louis, Missouri, 22–24 October 1990. IEEE.

[FS89] Uriel Feige and Adi Shamir. Zero Knowledge Proofs of Knowledge in Two Rounds. In *Advances in Cryptology - CRYPTO'89*, Lecture Notes in Computer Science 435, Springer-Verlag, 1989, pp. 526–544.

[For89] Lance Fortnow. The complexity of perfect zero-knowledge. In Silvio Micali, editor, *Advances in Computing Research*, volume 5, pages 327–343. JAC Press, Inc., 1989.

[FGM$^+$89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. In Silvio Micali, editor, *Advances in Computing Research*, volume 5, pages 429–442. JAC Press, Inc., 1989.

[Gol90] Oded Goldreich. A note on computational indistinguishability. *Information Processing Letters*, 34:277–281, 1990.

[Gol95] Oded Goldreich. *Foundations of Cryptography (Fragments of a Book).* Weizmann Institute of Science, February 1995.

[GG98] Oded Goldreich and Shafi Goldwasser. On the limits of non-approximability of lattice problems. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, Dallas, TX, May 1998. ACM.

[GKa96] Oded Goldreich and Ariel Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, 9(3): 167-190, 1996.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[GK93] Oded Goldreich and Eyal Kushilevitz. A perfect zero-knowledge proof system for a problem equivalent to the discrete logarithm. *Journal of Cryptology*, 6:97–116, 1993.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposiumm on Theorey of Computing*, pages 218-229, New York City, 25-27 May 1987.

[GMW91] Oded Goldreich, Silvio Micali, and Avi 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*, 38(1):691–729, 1991.

[GNW95] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR-lemma. ECCC Report TR95-050, March 1995. Available from `http://www.eccc.uni-trier.de/eccc/`.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, Winter 1994.

[GOP98] Oded Goldreich, Rafail Ostrovsky, and Erez Petrank. Computational complexity and knowledge complexity. *SIAM Journal on Computing*, 27(4):1116–1141, August 1998.

[GP91] Oded Goldreich and Erez Petrank. Quantifying knowledge complexity. In *Proceedings of the Thirty Second Annual Symposium on Foundations of Computer Science*, pages 59–68, 1991.

[GSV98] Oded Goldreich, Amit Sahai, and Salil Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 399–408, Dallas, TX, May 1998. ACM.

[GSV99] Oded Goldreich, Amit Sahai, and Salil Vadhan. Can statistical zero-knowledge be made non-interactive?, or On the relationship of SZK and NISZK. In *Advances in Cryptology—CRYPTO '99*, Lecture Notes in Computer Science, pages 467–484. Springer-Verlag, 1999, 15–19 August 1999.

[GV99] Oded Goldreich and Salil Vadhan. Comparing entropies in statistical zero-knowledge with applications to the structure of SZK. In *Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity*, pages 54–73, Atlanta, GA, May 1999. IEEE Computer Society Press.

[GB99] Shafi Goldwasser and Mihir Bellare. Lecture Notes on Cryptography, 1999. Available from homepage of Mihir Bellare (`http://www-cse.ucsd.edu/users/mihir`).

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, February 1989.

[GS89] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In Silvio Micali, editor, *Advances in Computing Research*, volume 5, pages 73–90. JAC Press, Inc., 1989.

[HILL] Johan Håstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. Construction of pseudorandom generator from any one-way function. To appear in *SICOMP*. Preliminary versions by Impagliazzo et. al. in *21st STOC* (1989) and Håstad in *22nd STOC* (1990).

[Hof95] Micha Hofri. *Analysis of Algorithms: Computational Methods and Mathematical Tools*. Oxford University Press, 1995.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstract). In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, Washington, 15–17 May 1989.

[IY87] Russell Impagliazzo and Moti Yung. Direct minimum-knowledge computations (extended abstract). In Carl Pomerance, editor, *Advances in Cryptology—CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 40–51. Springer-Verlag, 1988, 16–20 August 1987.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In J. W. Thatcher and R. E. Miller, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, Inc., 1972.

[KP98] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for NP with general assumptions. *Journal of Cryptology*, 11(1):1–27, Winter 1998.

[KP00] Joe Kilian and Erez Petrank. Concurrent Zero-Knowledge in Poly-logarithmic Rounds. Manuscript, April 2000.

[KPR98] Joe Kilian, Erez Petrank and Charlie Rackoff. Lower Bounds for Zero Knowledge on the Internet. *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, 1998, pp. 484–492.

[Koc96] P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. In *Advances in Cryptology - CRYPTO '96*, Lecture Notes in Computer Science, Vol. 1109, Springer-Verlag, 1996, pp. 104–113.

[LLS75] R. E. Ladner, N. A. Lynch, and A. L. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1:103–123, 1975.

[Lev73] Leonid A. Levin. Universal'nyĭe perebornyĭe zadachi (Universal search problems : in Russian). *Problemy Peredachi Informatsii*, 9(3):265–266, 1973.

[LFKN90] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proofs. In *Proceedings of the Thirty First Annual Symposium on Foundations of Computer Science*, pages 1–10, 1990.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 427–437, Baltimore, Maryland, 14–16 May 1990.

[Oka96] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. In *Proceedings of the Twenty Eighth Annual ACM Symposium on the Theory of Computing*, 1996. See also preprint of full version, August 1997.

[OVY93] R. Ostrovsky, R. Venkatesan, and M. Yung. Interactive hashing simplifies zero-knowledge protocol design. In *Proceedings of Eurocrypt '93, Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[Ost91] Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proceedings of the Thirty Second Annual Symposium on Foundations of Computer Science*, pages 133–138, 1991.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Proceedings of the Second Israel Symposium on Theory of Computing and Systems*, 1993.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[PT96] Erez Petrank and Gábor Tardos. On the knowledge complexity of NP. In *Proceedings of the Thirty Seventh Annual Symposium on Foundations of Computer Science*, pages 494–502, 1996.

[RK99] Ransom Richardson and Joe Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *Advances in Cryptology – EUROCRYPT'99*, 1999, Lecture Notes in Computer Science, Springer Verlag, 1999, pp. 415–431.

[S99] Amit Sahai. Non-malleable non-interactive zero knowledge and chosen ciphertext security. In *Proceedings of the Fortieth Annual IEEE Symposium on Foundations of Computer Science*, 1999.

[SV97] Amit Sahai and Salil Vadhan. A complete promise problem for statistical zero-knowledge. In *Proceedings of the 38th Annual Symposium on the Foundations of Computer Science*, pages 448–457. IEEE, October 1997.

[SV99] Amit Sahai and Salil Vadhan. Manipulating statistical difference. In Panos Pardalos, Sanguthevar Rajasekaran, and José Rolim, editors, *Randomization Methods in Algorithm Design (DIMACS Workshop, December 1997)*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 251–270. American Mathematical Society, 1999.

[Sch96] Bruce Schneier. *Applied cryptography: protocols, algorithms, and source code in C, 2nd edition.* New York : Wiley, 1996.

[Sha90] Adi Shamir. IP=PSPACE. In *Proceedings of the Thirty First Annual Symposium on Foundations of Computer Science*, pages 11–15, 1990.

[Vad00] Salil P. Vadhan. On transformations of interactive proofs that preserve the prover's complexity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, Portland, OR, May 2000. ACM. To appear.

[Vad99] Salil Pravin Vadhan. *A Study of Statistical Zero-Knowledge Proofs*. PhD thesis, Massachusetts Institute of Technology, August 1999.

[Yao82] Andrew C. Yao. Theory and application of trapdoor functions. In *Proceedings of the Twenty Third Annual Symposium on Foundations of Computer Science*, pages 80–91, 1982.