A Study of Statistical Zero-Knowledge Proofs

by

Salil Pravin Vadhan

A.B., Mathematics and Computer Science, Harvard University (1995)
Certificate of Advanced Study in Mathematics, Cambridge University (1996)

Submitted to the Department of Mathematics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1999

Author ....
Department of Mathematics
August 6, 1999

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

Accepted by ....
Michael Sipser
Chairman, Applied Mathematics Committee

Accepted by ....
Tomasz Mrowka
Chairman, Departmental Committee on Graduate Students

A Study of Statistical Zero-Knowledge Proofs
by
Salil Pravin Vadhan

Submitted to the Department of Mathematics
on August 6, 1999, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Zero-knowledge interactive proofs, introduced by Goldwasser, Micali, and Rackoff, are fascinating constructs which enable one party (the "prover") to convince another party (the "verifier") of an assertion, with the property that the verifier learns nothing other than the fact that the assertion being proven is true. In addition to being powerful tools for constructing secure cryptographic protocols, zero-knowledge proofs yield rich classes of computational problems that are of both complexity-theoretic and cryptographic interest.

This thesis is a detailed investigation of *statistical* zero-knowledge proofs, which are zero-knowledge proofs in which the condition that the verifier "learns nothing" is interpreted in a strong statistical sense. We begin by showing that the class **SZK** of problems possessing such proofs has two natural *complete problems*. These problems essentially amount to approximating the statistical difference or the difference in entropies between two "efficiently samplable" distributions. Thus, they give a new characterization of **SZK** which makes no reference to interaction or zero knowledge. They also simplify the study of statistical zero knowledge, as questions about the entire class **SZK** can be reduced to examining these two particular complete problems.

Using these complete problems as tools, we proceed to answer a number of fundamental questions about zero-knowledge proofs, including:

- Transforming any statistical zero-knowledge proof against an honest verifier (*i.e.,* a verifier that follows the specified protocol) into one which is zero knowledge even against cheating verifiers that deviate arbitrarily from the specified protocol. This transformation applies to public-coin computational zero-knowledge proofs as well.

- Constructing statistical zero-knowledge proofs for complex assertions built out of simpler assertions already shown to be in **SZK**. Via the complete problems, these closure properties translate to new methods for manipulating "efficiently samplable" distributions, which may be of independent interest.

- Obtaining simpler proofs of most previously known results about statistical zero knowledge, such as: Okamoto's result that **SZK** is closed under complement; the Fortnow and Aiello–Håstad upper bounds on the complexity of **SZK**; and Okamoto's result that every statistical zero-knowledge proof can be transformed into a public-coin one.

Thesis Supervisor: Shafi Goldwasser
Title: RSA Professor of Computer Science

Acknowledgments

The most daunting task facing a new graduate student is to find a research topic which is interesting yet approachable. I was rescued from this very early on by my advisor, Shafi Goldwasser. One month into graduate school, Shafi started me on a research problem that led to all the work in this thesis. (Alas, I still do not know the answer to the question she asked.) For that and her endless supply of other exciting research problems; for the way she always helped me think about research from a wider perspective; and for all her down-to-earth advice and encouragement, I will be forever grateful.

I was very lucky to start my Ph.D. when Oded Goldreich was visiting MIT. Much of the work in this thesis is joint work with Oded, and from him I learned a tremendous amount about both doing research and presenting it. Oded's trust in me has given me confidence as a researcher, and I am deeply indebted to him.

I also thank Amit Sahai, with whom I shared my first research experiences in graduate school and collaborated on much of the work in this thesis.

In the course of this research, I discussed these topics with many different people. These conversations not only affected my perspective on the area, but also helped refocus my research efforts. In this regard, I thank Sanjeev Arora, Mihir Bellare, Danny Gutfreund, Moni Naor, Erez Petrank, Madhu Sudan, Luca Trevisan, and Avi Wigderson, though undoubtedly I have forgotten a few.

There are many other people who have made my educational experience to this point so exciting and pleasurable, and who have helped shape my view of mathematics and computer science. There are too many to name individually, so I will be content to send my thanks to a few important groups. First, to my fellow students, the faculty, and the postdocs in the MIT Theory of Computation group for making it such a fun and inspiring place to be a graduate student. Second, to all my collaborators on research endeavors not described in this thesis, for all they have taught me. Third, to some wonderful teachers and engaging fellow students at Oceanside High School and Harvard College, for sparking my interest in mathematics and computer science well before I came to MIT.

My most important acknowledgment is to my family and friends, who have filled my non-academic life with happiness. Most significantly, I refer to my parents, who have always lovingly encouraged me to pursue my passions; my brother, Nehal, who always makes sure that I am having fun along the way; and my wife, Jen, who has given me a lifetime to look forward to.

My graduate studies were supported primarily by a Department of Defense National Defense Science and Engineering Graduate Fellowship and partially by DARPA grant DABT63-96-C-0018.

To Jen

Contents

1 Introduction **11**
1.1 The magic of zero-knowledge proofs 11
1.2 Informal definitions 12
1.3 Motivation for our study 14
1.4 Results & structure of this thesis 16

2 Definitions **19**
2.1 An example 19
2.2 Notation and preliminaries 22
2.3 Zero-knowledge proofs 24
2.4 Contrast with the GMR definition 28
2.5 Complexity aspects of interactive proofs 30

3 Complete Problems **33**
3.1 Statistical Difference 34
3.2 Analyzing public-coin **HVSZK** proofs 43
3.3 Analyzing general **HVSZK** proofs 52
3.4 Entropy Difference reduces to Statistical Difference 60
3.5 The Completeness Theorem 66

4 Applications of the Complete Problems **67**
4.1 Efficient **HVSZK** proof systems 68
4.2 The complexity of **SZK** 69
4.3 Expected polynomial-time simulators 70
4.4 Reversing statistical difference 71
4.5 Closure properties 73
4.6 Knowledge complexity 79
4.7 Perfect and computational zero knowledge 89
4.8 Zero-knowledge proofs for hard problems imply one-way functions 92

5 Private Coins vs. Public Coins **97**
5.1 Motivation and results 97
5.2 Overview 99
5.3 Subprotocol specifications 106
5.4 The transformed proof system 108

5.5 Okamoto's subprotocols 115

6 Coping with Cheating Verifiers **125**
6.1 Definitions 126
6.2 A cheating-verifier **SZK** proof system for $\overline{\mathrm{SD}^{1,1/2}}$ 128
6.3 Transforming honest-verifier proofs to cheating-verifier ones 132
6.4 Random selection 141
6.5 Corollaries and open problems 146

7 Noninteractive SZK **149**
7.1 The noninteractive model 150
7.2 The Completeness Theorem 154
7.3 Entropy Approximation is in **NISZK** 155
7.4 Proof of the Completeness Theorem 159
7.5 Comparing **SZK** and **NISZK** 161
7.6 Other applications of the Completeness Theorem 166

8 Conclusion **171**

A Chernoff Bounds **173**

B Hashing Lemmas **175**
B.1 Proof of Lemma 5.4.10 175
B.2 Proof of Lemma 6.4.5 177

Chapter 1

Introduction

1.1 The magic of zero-knowledge proofs

The notion of a *proof* plays a central role in mathematics and computer science. Proofs are the main fruits of a mathematician's labor; the goal of modern mathematics is not just to determine which mathematical statements are true but to prove that they are so. In theoretical computer science, the fundamental "**P** vs. **NP**" question essentially amounts to asking whether proofs are computationally harder to find than they are to verify.

Given their importance, it is natural to ask "What does one learn from a proof?" By definition, upon verifying a proof, one should be convinced that the assertion being proven is true. But a proof can actually reveal much more than that. Indeed, proofs in mathematics are valued for providing a great deal of insight in addition to validating a particular theorem. And, at a minimum, it seems inherent in the notion of a proof that after verifying a proof, one leaves not just with confidence that the assertion is true, but also with the ability to present the same proof to others and convince them of the assertion.

Zero-knowledge proofs, introduced by Goldwasser, Micali, and Rackoff [GMR89], are fascinating constructs which somehow escape the confines of this intuition — they are proofs which are convincing but reveal nothing other than the validity of the assertion being proven. In particular, after verifying a zero-knowledge proof, one does not gain the ability to convince someone else of the same statement!

In order to achieve this seemingly impossible goal, Goldwasser, Micali, and Rackoff introduce two new elements to the notion of a proof — interaction and randomization. Whereas classically a proof is a static object that can be written down and later verified, now a proof is viewed as interactive protocol between two communicating parties, a prover and a verifier. Both parties can be randomized (*i.e.,* they can "flip coins"), so the verifier can present the prover with "random challenges" and the prover can give "random responses." At the end, the verifier should be convinced (with high statistical confidence) that the assertion is true. Amazingly, it is possible to guarantee that the verifier learns essentially *nothing else* from the interaction.

Goldwasser, Micali, and Rackoff gave several possible interpretations of the condition that the verifier "learns nothing." This thesis is a detailed investigation of *statistical zero-knowledge proofs*, which are zero-knowledge proofs in which "learns nothing" is interpreted in a strong statistical sense. In the course of this investigation, we will address and answer several fundamental questions about statistical zero-knowledge proofs and completely characterize the types of "assertions" that possess statistical zero-knowledge proofs. Some of our results also apply to *perfect* and *computational* zero-knowledge proofs, which are those obtained by different interpretations of the condition that the verifier "learns nothing."

1.2 Informal definitions

It is remarkable that zero-knowledge proofs can even be defined in a meaningful and realizable manner. In this section, we give a high-level sketch of the notions needed to formalize them, beginning with what we mean by "assertion."

In order to facilitate their complexity-theoretic study, "assertions" are thought of as strings written in some fixed alphabet, and their interpretations are given by a *language* L identifying the "valid assertions." For example, assertions about 3-colorability of graphs can be formalized by interpreting every string x as a graph G_x and taking the language L to be the set of x for which G_x is 3-colorable. So, the string x represents the assertion "$x \in L$," which translates to the statement "G_x is 3-colorable." We also think of the language L as defining the following decision problem: given a string x, decide whether it is in L or not. Therefore, we will use the terms "assertion," "language," and "problem" interchangeably in our informal discussion.

The complexity class **NP** consists of those languages possessing efficiently verifiable "classical" proofs. That is, a language L is in **NP** if there is an efficient proof-verification algorithm (called a *verifier*) satisfying the following two conditions:

- **Completeness:** For every valid assertion (*i.e.,* every string in L), there exists a proof that the verifier will accept.

- **Soundness:** For every invalid assertion (*i.e.,* every string not in L), no "proof" can make the verifier accept.

By "efficient," we mean that the verifier should run in time polynomial in the length of the assertion (written as a string). We consider such proofs "classical" because the proof is a fixed, written string given in its entirety to the verification algorithm which checks it deterministically.

Interactive proofs, introduced by Goldwasser, Micali, and Rackoff [GMR89] serve the same purpose as classical proofs — to convince a verifier with limited computational power that some assertion is true. However, as mentioned above, this is no longer accomplished by giving the verifier a fixed, written proof, but rather by having the verifier to interact with a *prover* that has unbounded computational power. After the parties exchange messages for some number of rounds, the verifier decides whether to accept or reject. We still require that the verifier's computation time be polynomial in the length of the assertion, but now both the prover and verifier may be randomized. The following two relaxations of the classical notions of completeness and soundness guarantee that an interactive proof is "convincing":

- **Completeness:** For every valid assertion, there is a prover strategy that will make the verifier accept with high probability.

- **Soundness:** For every invalid assertion, the verifier will reject with high probability, no matter what strategy the prover follows.

The complexity class **IP** is the class of languages possessing interactive proofs. Clearly, every language that possesses a classical proof also possesses an interactive proof (in which the prover simply sends the verifier the classical proof). But the converse is not clear; interactive proofs are potentially much more expressive than classical ones. In fact, it has been shown that many more languages possess interactive proofs than classical ones [LFKN92, Sha92]. That is, **IP** is much larger than **NP** (given widely believed complexity-theoretic assumptions).

A *zero-knowledge proof* is an interactive proof in which the verifier learns nothing from the interaction with the prover, other than the fact that the assertion being proven is true. This is guaranteed by requiring that whatever the verifier sees in the interaction with the prover is something it could have efficiently generated on its own. That is, there should be a polynomial-time algorithm, called a *simulator*, that "simulates" the verifier's view of the interaction with the prover (*e.g.,* all the messages exchanged between the two parties). Recall that the interaction between the prover and verifier is probabilistic. Thus, the simulator is also probabilistic, and we require that it generates an output distribution that is "close" to what the verifier sees when interacting with the prover (when the assertion being proven is true). Intuitively, this means that the verifier learns nothing because it can run the simulator instead of interacting with the prover.

Three different interpretations of "close" were suggested in [GMR89] and these lead to the three forms of zero knowledge commonly considered in the literature:

- **Perfect zero knowledge:** Requires that the distributions are identical.
- **Statistical zero knowledge:** Requires that the distributions are statistically close.
- **Computational zero knowledge:** Requires that the distributions cannot be distinguished by any polynomial-time algorithm.

PZK, **SZK**, and **CZK** are the classes of languages possessing perfect, statistical, and computational zero-knowledge proofs, respectively. Perfect and statistical zero knowledge capture much stronger requirements than computational zero-knowledge, in that the zero-knowledge condition is meaningful regardless of the computational power of the verifier.[1]

Amazingly, every problem having a classical proof also has a computational zero-knowledge proof; that is $\mathbf{NP} \subset \mathbf{CZK}$ [GMW87]. In fact, so does every problem with an interactive proof; that is, $\mathbf{IP} = \mathbf{CZK}$ [IY87, BGG$^+$88].[2]

In contrast, it is unlikely that every problem in **NP** possesses a perfect or statistical zero-knowledge proof [For89, AH91, BHZ87]. This is the price paid for the strong security guarantee offered by these types of zero-knowledge proofs. Still, as we will see, a number of important, nontrivial problems possess statistical zero-knowledge proofs, and these are sufficient for some cryptographic applications.

[1] Although the verifier need only run in polynomial time to verify an interactive proof, a "cheating" verifier may be willing to invest additional computation to gain knowledge from the proof. Perfect and statistical zero-knowledge proofs guarantee that this will not help.

[2] Both of these results require the standard assumption that "one-way functions" exist.

1.3 Motivation for our study

1.3.1 Complexity Theory

Statistical zero knowledge, though defined with cryptography in mind, is a rich domain for complexity-theoretic investigations. The first indication of this comes from the fact that statistical zero-knowledge proofs have been given for a number of important computational problems: Quadratic Residuosity and Nonresiduosity [GMR89], Graph Isomorphism and Nonisomorphism [GMW91], a problem equivalent to Discrete Logarithm [GK93], and approximate versions of the Shortest Vector and Closest Vector problems for lattices [GG98a]. These problems have attracted a great deal of attention in the theoretical computer science and cryptography literature, and statistical zero knowledge captures a nontrivial property shared by all of them. Moreover, no efficient (*i.e.,* polynomial-time) algorithms are known for solving these problems and they are widely believed to be computationally hard. On the other hand, it is unlikely that any problem possessing a statistical zero-knowledge proof is **NP**-hard [For89, AH91, BHZ87]. Thus, the class **SZK**, of problems possessing statistical zero-knowledge proofs, holds an intriguing position in complexity theory, lying somewhere between the tractable problems and the **NP**-hard problems.

In this thesis, we will show that **SZK** possesses two natural "complete problems," called Statistical Difference and Entropy Difference. Both of these problems involve comparing probability distributions given by efficient sampling procedures. The fact that they are "complete" means that the computational complexity of these problems is equivalent to that of entire class **SZK**. As a consequence, many results about statistical zero knowledge directly translate to methods for manipulating efficiently samplable distributions and conversely. Indeed, in this thesis we will make use of this correspondence in both directions. These complete problems are of independent interest, so the fact that they are complete for **SZK** gives further evidence that statistical zero knowledge captures a rich and natural class of computational problems.

1.3.2 Cryptography

As one might imagine, zero-knowledge proofs have vast applicability in cryptography. One of the first examples of their utility was the construction of *Identification Schemes* by Feige, Fiat, and Shamir [FFS88]. The premise is that one party, Alice, should be able to identify herself repeatedly to a second party, Bob. For example, Bob can be thought of as an internet service provider or a remote computer network on which Alice has an account. The most common solution for this problem is for Alice to choose a password that Bob keeps stored in a secure password file. When Alice wishes to identify herself to Bob, she simply sends her password to Bob, who checks it against the file. The difficulty with this solution is that an adversary can, by impersonating Bob, obtain Alice's password and later use this to misrepresent himself as Alice.

Zero-knowledge proofs provide an elegant solution to this problem. Instead of choosing a password, Alice generates a true mathematical statement S for which only she knows the proof (and such that it is difficult for an adversary to come up with a proof for S). Bob stores this statement. When Alice wishes to identify herself to Bob, she gives Bob a *zero-knowledge* *proof* that S is true. This identifies Alice as the one who knows a proof for S, while Bob (or an adversary impersonating Bob) does not learn the proof for S and hence cannot later misrepresent himself as Alice. There are some subtleties in making this approach work, but these can be handled, and the example illustrates the potential cryptographic power of zero-knowledge proofs. The advantage of using *statistical* zero-knowledge proofs in a cryptographic protocol is that they provide an extremely strong security guarantee, in that they remain convincing and reveal nothing even when the parties involved have unlimited computational power.

More generally, zero-knowledge proofs are a tool for forcing parties to behave "honestly" in cryptographic protocols. Participants can prove to each other that their actions are consistent with a specified protocol without revealing any of the secret information they possess (such as cryptographic keys). To exploit this idea in its full generality, Goldreich, Micali, and Wigderson [GMW91, GMW87] and Yao [Yao86] use the fact that all **NP** statements can be proven in computational zero knowledge (*i.e.,* $\mathbf{NP} \subset \mathbf{CZK}$), as shown in [GMW91]. As mentioned earlier, the strong security guarantee of statistical zero-knowledge proofs makes it unlikely that $\mathbf{NP} \subset \mathbf{SZK}$ [For89, AH91, BHZ87], but statistical zero-knowledge proofs can and have been used in specific cryptographic protocols, such as the identification schemes of Feige, Fiat, and Shamir [FFS88] mentioned above.

This issue of parties deviating from the protocol already arises within zero-knowledge proofs themselves. For the first few chapters of this thesis, we will focus on *honest-verifier* zero-knowledge proofs, which are those in which the verifier is only guaranteed to learn nothing *if it follows the specified protocol.* Focusing on honest-verifier proofs will greatly facilitate our investigation and will be essential in our proofs of several results, such as the completeness theorems mentioned earlier. But clearly such proofs are unsuitable for most cryptographic applications. One of the main results of this thesis is a method for transforming every honest-verifier statistical zero-knowledge proof into one robust even against verifiers that deviate arbitrarily from the specified protocol. By this transformation (given in Chapter 6), our results about honest-verifier zero knowledge proofs automatically translate to general zero-knowledge proofs. Moreover, it suggests a methodology for constructing general zero-knowledge proofs: first construct an honest-verifier proof (which is often an easier task) and then use our transformation to make it robust against cheating verifiers. Our transformation also applies to wide class of computational zero-knowledge proofs (namely, "public coin" proofs).

Another important role statistical zero knowledge can play from the perspective of cryptography is that it provides the cleanest model for the study of zero-knowledge proofs. Statistical zero-knowledge proofs tend to be easier to analyze and manipulate than other forms of zero-knowledge proofs, and general theorems about them can be proven without making any complexity-theoretic assumptions. In contrast, other forms of zero-knowledge proofs, such as computational zero-knowledge proofs and zero-knowledge "arguments"[3] are usually constructed based on intractability assumptions such as the existence of "one-way functions" (*e.g.,* the hardness of factoring). Thus, a natural methodology is to first understand

[3]Zero-knowledge arguments, introduced by Brassard, Chaum, and Crépeau [BCC88], are a variant of zero-knowledge proofs in which the soundness requirement is weakened to only require that it is computationally hard to convince the verifier of a false statement. We will not discuss these further in this thesis.

a phenomenon with respect to statistical zero knowledge and then attempt to translate the techniques and results to other forms of zero knowledge. This approach has seen success in the past (*e.g.,* [Ost91] leading to [OW93]) and our transformation of honest-verifier zero-knowledge proofs into general ones is another example.

It should be noted that, if one assumes the existence of one-way functions, essentially all questions about computational zero knowledge have been resolved. However, we regard it as important to understand which aspects of computational zero knowledge rely inherently on intractability assumptions and which do not. Moreover, minimizing the use of hard problems in constructing zero-knowledge proofs tends to lead to more efficient constructions and higher levels of security. Our approach of first proving results about statistical zero knowledge and then attempting to translate them to computational zero knowledge is compelling in these respects.

1.4 Results & structure of this thesis

Chapter 2 — Definitions. We give an introduction to statistical zero knowledge. After an informal example, we formally define statistical zero-knowledge proofs and identify some of the issues that arise with the definitions. In particular, we note that until Chapter 6, we focus on honest-verifier statistical zero knowledge. As mentioned above, this restriction will be convenient for the first few chapters, but will be removed completely in Chapter 6.

Chapter 3 — Complete Problems. We introduce the problems STATISTICAL DIFFERENCE and ENTROPY DIFFERENCE and prove that they are complete for (honest-verifier) **SZK**. These complete problems will be our main tools in obtaining further results about statistical zero knowledge. When proving general theorems about statistical zero knowledge, we will be able to focus on these specific complete problems, and largely avoid working with the rather unwieldy general definition of statistical zero-knowledge proofs. STATISTICAL DIFFERENCE was shown to be complete for statistical zero knowledge in joint work with Amit Sahai [SV97] and ENTROPY DIFFERENCE in joint work with Oded Goldreich [GV99]. The material in Chapter 3 is a combination of techniques and results from the corresponding two papers.

Chapter 4 — Applications of the Complete Problems. We present a number of immediate applications of the complete problems and the techniques used in their proof. For example, we show that every problem possessing an (honest-verifier) statistical zero-knowledge proof also has a very communication-efficient one, in which only two messages are exchanged and the error parameters are exponentially small. We also exhibit some strong closure properties of statistical zero-knowledge, obtain efficient algorithms for manipulating the statistical properties of samplable distributions, and prove some results about "knowledge complexity." In addition, the complete problems yield simpler proofs of most previously known results about the complexity of statistical zero knowledge. For example, in Section 4.2, we show how Okamoto's result from [Oka96] that (honest-verifier) **SZK** is closed under complement follows immediately from the completeness theorems. We also apply some of the same techniques to obtain results about perfect and computational zero-knowledge proofs. Most of the material in this chapter was obtained in joint work with Amit Sahai [SV97, SV99].

Chapter 5 — Private Coins vs. Public Coins. We show that every problem possessing an (honest-verifier) statistical zero-knowledge proof also possesses a *public-coin* one — that is, a statistical zero-knowledge proof in which the verifier's messages consist merely of random coin flips. This was originally proven by Okamoto [Oka96]. However, we give a markedly simpler proof. The result of this chapter is useful because public-coin interactive proofs are much easier to analyze and manipulate than general "private-coin" interactive proofs. Indeed, this result provides an essential starting point for the following chapter. We also give the first transformation from private coins to public coins which applies to a wide class of computational zero-knowledge proofs. Namely, we show how to transform 3-message (honest-verifier) *computational* zero-knowledge proofs into public-coin ones.

The transformations from private coins to public coins presented in this chapter are based on joint work with Oded Goldreich [GV99] and discussions with Amit Sahai.

Chapter 6 — Coping with Cheating Verifiers. We show how to transform any honest-verifier statistical zero-knowledge proof into one which remains statistical zero-knowledge even against cheating verifier strategies. The same transformation applies to public-coin computational zero-knowledge proofs. The transformation is obtained by augmenting any public-coin honest-verifier proof with a new protocol for two mutually distrustful parties to select a random string. This Random Selection Protocol may be of independent interest. The material in this chapter is joint work with Oded Goldreich and Amit Sahai [GSV98].

Chapter 7 — Noninteractive SZK. We examine "noninteractive" statistical zero-knowledge proofs, which are ones in which the need for interaction is removed via an augmentation to the model. We exhibit two natural complete problems for **NISZK**, the class of problems possessing noninteractive statistical zero knowledge proofs. These complete problems are closely related to those for **SZK**. We then use these problems to relate the complexities of **NISZK** and **SZK**, and explore the possibility that every statistical zero-knowledge proof can be transformed into a noninteractive one. This chapter consists of results obtained with Oded Goldreich and Amit Sahai [GSV99].

Chapter 8 — Conclusions. We summarize what has been achieved in the thesis, and discuss possible avenues for further research.

Historical remark. The results in this thesis are not presented in chronological order. We have shuffled the historical order to yield what seems to be the most natural presentation, given the benefits of hindsight. In reality, Okamoto's transformation from private coins to public coins [Oka96] preceeded all the results in this thesis, and indeed sparked much of this work. The completeness of STATISTICAL DIFFERENCE [SV97], its applications given in Chapter 4 [SV97, SV99], and the honest-verifier to cheating-verifier transformation of Chapter 6 [GSV98] all originally used Okamoto's theorem as a starting point. We later introduced ENTROPY DIFFERENCE in [GV99] in order to give a simpler proof of Okamoto's theorem.

Chapter 2

Definitions

2.1 An example

Before giving the formal definitions, we illustrate the notion of a zero-knowledge proof with an elegant example: the (honest-verifier) perfect zero-knowledge proof for Graph Nonisomorphism. The proof system is due to Goldreich, Micali, and Wigderson [GMW91], and uses ideas from an earlier proof system for Quadratic Nonresiduosity, due to Goldwasser, Micali, and Rackoff [GMR89].

Definition 2.1.1 *If* $G = (V, E)$ *is an undirected graph and* π *is a permutation on* V*, then* $\pi(G)$ *denotes the graph obtained by permuting the vertices of* G *according to* π*. That is,* $\pi(G) = (V, E')$*, where* $E' = \{(\pi(u), \pi(v)) : (u, v) \in E\}$*. If* G *and* H *are graphs on the same vertex set, and there exists a* π *such that* $\pi(G) = H$*, we say that* G *and* H *are* isomorphic *and write* $G \cong H$*.* π *is called an* isomorphism *between* G *and* H*, and* H *is said to be an* isomorphic copy *of* G*.* Graph Isomorphism *is the language* $\mathrm{GI} = \{(G, H) : G \cong H\}$. Graph Nonisomorphism *(*GNI*) is the complement of* GI.[1]

It is easy to see that Graph Isomorphism is in **NP**; an easily verifiable proof that two graphs are isomorphic is an isomorphism between them. In contrast, no classical proofs are known for Graph Nonisomorphism. Nevertheless, Graph Nonisomorphism does possess a very efficient interactive proof.[2] The interactive proof is based on two observations. First, if two graphs are nonisomorphic, then their sets of isomorphic copies are disjoint. Second, if two graphs are isomorphic, then a uniformly selected isomorphic copy of one graph is indistinguishable from a uniformly selected isomorphic copy of the other. Thus, the interactive proof, given in Protocol 2.1.2, tests whether the prover can distinguish uniformly selected isomorphic copies of the two graphs.

[1] To formally define GI and GNI as sets of strings, one must specify how graphs are encoded as strings, but any reasonable encoding will work for our purposes. Typically, encoding issues are easily managed and hence we will usually ignore them in this thesis. Also note that if two graphs are on different vertex sets of the same size, we have implicitly defined them to be nonisomorphic. This convention is inessential.

[2] There has been some recent evidence that Graph Nonisomorphism is in **NP**, in fact based on the existence of an efficient interactive proof for Graph Nonisomorphism [KvM99].

Protocol 2.1.2: Interactive proof (P, V) for Graph Nonisomorphism

Input: Graphs $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$

1. V: Uniformly select $b \in \{0, 1\}$. Uniformly select a permutation π on V_b. Let $H = \pi(G_b)$. Send H to P.
2. P: If $G_0 \cong H$, let $c = 0$. Else let $c = 1$. Send c to V.
3. V: If $c = b$, accept. Otherwise, reject.

Proposition 2.1.3 ([GMW91]) *Protocol 2.1.2 is an interactive proof system for* Graph Nonisomorphism.

Proof Sketch: If G_0 and G_1 are nonisomorphic, then $G_0 \cong H$ if and only if $b = 0$. So the prover strategy specified above will make the verifier accept with probability 1. Thus, completeness is satisfied.

On the other hand, if G_0 and G_1 are isomorphic, then H has the same distribution when $b = 0$ as it does when $b = 1$. Thus, b is independent of H and the prover has at most probability at most $1/2$ of guessing b correctly *no matter what strategy it follows.* This shows that the protocol is sound. □

A few remarks about the proof system are in order. The first is that the verifier's confidence that the graphs are nonisomorphic after one execution of the protocol is not very high, as the prover can suceed with probability $1/2$ even when the graphs are isomorphic. However, this error probability can be made reduced to $1/2^k$ by repeating the protocol k times (sequentially or in parallel) and requiring that the prover succeeds in all k repetitions. Second, the proof system is very communication efficient; only two messages are exchanged and the prover sends only one bit to the verifier (more generally, k bits to achieve soundness $1/2^k$). Finally, note that it is crucial for soundness that the verifier's random coin flips are kept "private." If the bit b is made public and revealed to the prover, soundness will no longer hold. Surprisingly, every *private-coin* interactive proof (like the one above) can be transformed into a *public-coin* one; that is, one in which the verifier's coin flips are completely visible to the prover [GS89]. We will present an analogous transformation for statistical zero-knowledge proofs in Chapter 5.

We now informally argue that, when the graphs are nonisomorphic, the verifier learns nothing else from the above protocol. The only message sent from the prover to the verifier is the guess c. We have already shown that, when the graphs are nonisomorphic, the prover guesses correctly with probability 1. That means that, with probability 1, c is simply equal to b, which is a value the verifier already knows (since it chooses b itself)! Note that this intuition only refers to a verifier that follows the specified protocol. There is nothing to force a cheating verifier to select H by first picking one of the two input graphs and then permuting its vertices. So we have no reason to believe that a cheating verifier "already knows" whether H is isomorphic to G_0 or G_1, and thus we will only prove that the proof system is honest-verifier zero knowledge.

To formalize this intuition, we must exhibit a *simulator*, as required by the definition of zero knowledge. The simulator must be an efficient probabilistic algorithm whose output is similar to the verifier's view of the interaction, when given a pair of nonisomorphic graphs as input. The verifier's view of the interaction includes not just the messages exchanged between the verifier and prover (H and c), but also includes the verifier's random coin tosses (the permutation π and the bit b). By convention, the output of the simulator is of the form $(m_1, m_2, \ldots, m_k; r)$, where the m_i's are the simulated messages, and r is the simulation for the verifier's random coins. In light of the above discussion, the simulator, given in Algorithm 2.1.4, simply mimics the verifier's protocol and assumes that the prover guesses correctly.

Algorithm 2.1.4: Simulator for GRAPH NONISOMORPHISM **Proof System**

Input: Graphs $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$

1. Uniformly select $b \in \{0, 1\}$. Uniformly select a permutation π on V_b. Let $H = \pi(G_b)$.
2. Let $c = b$.
3. Output $(H, c; b, \pi)$

From the fact that the prover guesses correctly with probability 1 in the protocol, it follows immediately that the output distribution of the simulator is *identical* to the verifier's view of the interaction (when the input graphs are nonisomorphic and the verifier follows the protocol). Thus, we have:

Proposition 2.1.5 ([GMW91]) *Protocol 2.1.2 is an honest-verifier perfect zero-knowledge proof system for* GRAPH NONISOMORPHISM.

As mentioned before, the probability that the prover can convince the verifier to accept when the graphs are isomorphic can be reduced by repeating the proof system many times. Luckily, both forms repetition (parallel and sequential) preserve honest-verifier perfect zero knowledge; a simulator for the repeated proof system can be obtained by running the original simulator many times. With *cheating* verifiers, however, things are more subtle. Sequential repetition preserves zero knowledge against cheating verifiers, but parallel repetition does not [GK96b].

Although we have only exhibited an honest-verifier zero-knowledge proof system for GRAPH NONISOMORPHISM, Goldreich, Micali, and Wigderson [GMW91] show how to augment this particular protocol to make it perfect zero-knowledge even against cheating verifiers. Later in this thesis, we will present general method for making zero-knowledge proofs robust against cheating verifiers which could be used instead (though the result will be statistical, rather than perfect, zero knowledge).

Having seen just this one beautiful example of a zero-knowledge proof, one might wonder whether the same ideas can be used to construct zero-knowledge proofs for other problems. As mentioned earlier, the GRAPH NONISOMORPHISM proof system is based on ideas drawn from an earlier proof system for QUADRATIC NONRESIDUOSITY [GMR89]. Although both of these proof systems seem to be exploiting algebraic properties of permutation groups or quadratic residues modulo a composite, actually at the heart of correctness is simply the relationship between two probability distributions. In the case of GRAPH NONISOMORPHISM, these distributions are those obtained by taking a random isomorphic copy of G_0 or G_1, respectively. In Chapter 3, we use this observation to abstract and generalize Protocol 2.1.2. We then prove that the resulting proof system is "universal" for statistical zero knowledge, in the sense that *every* statistical zero-knowledge proof can be transformed into one "of the same form".

2.2 Notation and preliminaries

Strings and promise problems. Throughout this thesis, all strings are over the binary alphabet $\{0,1\}$. Often we will discuss non-binary strings or tuples of strings, but it is easy to encode such objects as binary strings, and we implicitly assume that such an encoding has been fixed. A unary string of length k is denoted 1^k.

We will consider a wider class of decision problems than languages. Specifically, we will allow some inputs to be "excluded." This is formalized by the notion of a *promise problem* [ESY84]. A promise problem Π is a pair (Π_Y, Π_N) of disjoint sets of strings, corresponding to YES *instances* and NO *instances*, respectively. This naturally yields the following computational problem: Given a string x which is "promised" to be in $\Pi_Y \cup \Pi_N$, decide whether $x \in \Pi_Y$ or $x \in \Pi_N$. Strings in $\Pi_Y \cup \Pi_N$ are called *instances* of Π, and strings not in $\Pi_Y \cup \Pi_N$ are said to *violate the promise*. The *complement* of a promise problem Π is the promise problem $\overline{\Pi}$, where $\overline{\Pi}_Y = \Pi_N$ and $\overline{\Pi}_N = \Pi_Y$. If $\mathbf{C}$ is a class of promise problems, then $\mathbf{co\text{-}C} \stackrel{\text{def}}{=} \{\overline{\boldsymbol{\Pi}} \,:\, \boldsymbol{\Pi} \in \mathbf{C}\}$.

Algorithms. As we will only be doing complexity analysis at a fairly coarse-grained level, the particular model of computation used is not crucial. Any standard model, such as the multitape Turing machine, will do, and the reader is referred to any standard text on complexity theory (e.g., [Sip97, Pap94]) for a more detailed discussion. We will describe algorithms at a high level, ignoring implementation details such as encodings of inputs. We measure the running time of a deterministic algorithm as a function of input length; algorithm A runs in time $t(\cdot)$, if A takes at most time $t(|x|)$ on every input x. The complexity class $\mathbf{P}$ is the class of *promise problems* solvable in polynomial time.

Randomized algorithms are obtained by allowing our algorithms the ability to flip an unbiased "coin" upon request. To avoid assuming an a priori bound on the number of coin flips an algorithm will make, we model this by giving the algorithm access to an infinite string $r \in \{0,1\}^*$ in which every bit is selected uniformly and independently. Since the number of bits in this string that are accessed is bounded by the algorithm's running time, we will often truncate r to be just a finite string containing only the random bits that are used in a particular execution. If A is a randomized algorithm, we write $A(x;r)$ for the output of A on input x, using random coins r. A is said to have *(strict) running time* $t(\cdot)$ if for every x and r, A takes time at most $t(|x|)$. A is said to have *expected running time* $t(\cdot)$ if for every x, the number of steps taken by A on input (x,r) has expectation at most $t(|x|)$, taken over the choice of r. For a fixed input x, we write either $A(x)$ or A_x for the probability distribution induced on the output $A(x;r)$ obtained by choosing r uniformly at random.

We say that a randomized algorithm A *solves* a promise problem Π *with 2-sided error* if for every instance x of Π, A correctly decides whether x is a YES or NO instance with probability at least $2/3$ over the choice of r. **BPP** is the class of promise problems that can be solved in (worst-case) polynomial time with 2-sided error.

We will also consider *nonuniform* polynomial-time algorithms, which are polynomial-time algorithms that are given an extra "advice" string of length polynomial in its input. More formally, a *nonuniform polynomial-time algorithm* A is specified by a polynomial-time algorithm B together with strings $\{\alpha_n\}_{n\in\mathbb{N}}$ such that $A(x) = B(x, \alpha_{|x|})$ for all x and $|\alpha_n|$ is bounded by some polynomial in n. *Nonuniform probabilistic polynomial-time algorithms* are defined analogously, by taking B to be probabilistic polynomial time. Nonuniform polynomial-time algorithms are equivalent to polynomial-sized families of circuits.

Reductions and completeness. Reductions are our means for comparing the complexities of problems. A *(Karp) reduction* from a promise problem Π to a promise problem Γ 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 such a reduction exists, we say that Π *(Karp-)reduces* to Γ and write $\Pi \leq_{\mathtt{Karp}} \Gamma$ (or just $\Pi \leq \Gamma$).

A *Cook reduction* from Π to Γ is a polynomial-time algorithm that solves Π when given access to an oracle which solves Γ. That is, on input x, the oracle returns Y if $x \in \Gamma_Y$, N if $x \in \Gamma_N$, and can respond either Y or N if x violates the promise. The reduction should work regardless of how the oracle responds on inputs that violate the promise. If such a reduction exists, we say that Π *Cook reduces* to Γ and write $\Pi \leq_{\mathtt{Cook}} \Gamma$. Informally, the existence of a (Karp or Cook) reduction from Π to Γ means that Π is computationally no harder than Γ.

Let $\mathbf{C}$ be a class of promise problems. We say that $\mathbf{C}$ is *closed under (Karp) reductions* (resp., Cook reductions) if $\Pi \leq \Gamma$ (resp., $\Pi \leq_{\mathtt{Cook}} \Gamma$) and $\Gamma \in \mathbf{C}$ implies that $\Pi \in \mathbf{C}$. A promise problem Γ is $\mathbf{C}$-hard (with respect to a given type of reduction) if every promise problem in $\mathbf{C}$ reduces to Γ via that type of reduction. Γ is *complete* for $\mathbf{C}$ (or $\mathbf{C}$-complete) if (1) $\Gamma \in \mathbf{C}$, and (2) Γ is $\mathbf{C}$-hard with respect to Karp reductions.

Probability distributions. If X is a probability distribution (or random variable) on a universe $\mathcal{U}$, then the *support* of X is $\mathrm{Supp}(X) \stackrel{\text{def}}{=} \{x \in \mathcal{U} : \Pr[X = x] > 0\}$. We write $x \leftarrow X$ to denote the process of randomly choosing x according to the distribution X. If

S is a set, then the uniform distribution is also written S, so $x \leftarrow S$ denotes choosing x uniformly in S.

The definition of statistical zero knowledge makes use of a standard measure of similarity between probability distributions.

Definition 2.2.1 (statistical difference) *If X and Y are probability distributions (or random variables) on a discrete universe $\mathcal{U}$, then the* statistical difference *(or* variation distance*) between X and Y is defined to be*

$$\mathrm{StatDiff}\,(X,Y) \stackrel{\text{def}}{=} \max_{S \subset \mathcal{U}} \left|\Pr\left[X \in S\right] - \Pr\left[Y \in S\right]\right|.$$

Various properties of this distance measure will play a major role in our investigation, but, for now, we just list some basic facts about $\mathrm{StatDiff}\,(\cdot,\cdot)$ that show that it conforms to an intuitive notion of the distance between probability distribution.

Fact 2.2.2 *Let X, Y, and Z be any three probability distributions (on a common universe $\mathcal{U}$). Then*

1. $\mathrm{StatDiff}\,(X,Y) \geq 0$*, with equality iff X and Y are identically distributed.*
2. $\mathrm{StatDiff}\,(X,Y) \leq 1$*, with equality iff X and Y have disjoint supports.*
3. $\mathrm{StatDiff}\,(X,Y) = \mathrm{StatDiff}\,(Y,X)$.
4. $\mathrm{StatDiff}\,(X,Z) \leq \mathrm{StatDiff}\,(X,Y) + \mathrm{StatDiff}\,(Y,Z)$.
5. *For any function f,* $\mathrm{StatDiff}\,(f(X),f(Y)) \leq \mathrm{StatDiff}\,(X,Y)$.

2.3 Zero-knowledge proofs

In this section, we give formal definitions for the notions of classical proofs (**NP**), interactive proofs (**IP**), and honest-verifier zero-knowledge proofs.

Definition 2.3.1 (classical proofs — NP) *A* classical proof system *for a promise problem Π is given by a* verification algorithm *V and a polynomial $p(\cdot)$ such that*

1. *(Efficiency) V runs in (deterministic) polynomial time.*
2. *(Completeness) If $x \in \Pi_Y$, then there exists a y of length at most $p(|x|)$ such that $V(x,y)$ accepts. y is called a* proof *(or* witness*) for x.*
3. *(Soundness) If $x \in \Pi_N$, then for every y, $V(x,y)$ rejects.*

NP *is the class of promise problems possessing classical proofs.*

NP was originally defined in terms of nondeterministic Turing machines, but it is well known that the above definition is equivalent. The purpose of the polynomial $p(\cdot)$ in the above definition is to guarantee that the time for verifying a proof is polynomial in the length of the assertion x.

Recall that interactive proofs are obtained by replacing proofs with a "prover" that "interacts" with a probabilistic "verifier". In order to make this precise, we must first formalize the notion of an interactive protocol between two parties A and B. We do this by viewing each party as a function, taking the history of the protocol (all the messages previously exchanged) and the party's random coins, to the party's next message. Either party can decide to halt the interaction (possibly accepting or rejecting at the same time), and the other party is given an opportunity to compute one more message at that time.

Definition 2.3.2 (interactive protocols) *An* interactive protocol (A, B) *is any pair of functions. The* interaction *between A and B on common input x is the following random process (denoted $(A, B)(x)$):*

1. *Uniformly choose random coins r_A and r_B (infinite binary strings) for A and B, respectively.*
2. *Repeat the following for $i = 1, 2, \ldots$:*
   (a) *If i is odd, let $m_i = A(x, m_1, \ldots, m_{i-1}; r_A)$.*
   (b) *If i is even, let $m_i = B(x, m_1, \ldots, m_{i-1}; r_B)$.*
   (c) *If $m_{i-1} \in \{\texttt{accept}, \texttt{reject}, \texttt{halt}\}$, then exit loop.*

If the last message computed by A is `accept` *(resp.,* `reject`*), we say that A* accepts *(resp.,* rejects*), and similarly for B. We call such a protocol* polynomially bounded *if there is a polynomial $p(\cdot)$ such that, on common input x, at most $p(|x|)$ messages are exchanged, and each is of length at most $p(|x|)$ (with probability 1 over the choice of r_A and r_B).*

In [GMR89], interactive protocols were defined in terms "interactive Turing machines," but that approach is too tied to a particular model of computation for our tastes. This equivalent formulation in terms of functions was noted by Goldwasser and Sipser [GS89].

Now interactive proofs can be defined as a type of interactive protocol between a prover (with no computational limitations) and a polynomial-time verifier. The completeness and soundness conditions of classical proofs are replaced with probabilistic ones that guarantee that the verifier gains statistical confidence that the assertion being proven is true. The amount of confidence gained by the verifier is quantified by two quantities, called the *completeness* and *soundness errors*, which in turn are functions of a *security parameter* k.

Definition 2.3.3 (interactive proofs — IP) *Let (P, V) be an interactive protocol and let Π be a promise problem. (P, V) is said to be an* interactive proof system *for Π with* completeness error $c : \mathbb{N} \to [0, 1]$ *and* soundness error $s : \mathbb{N} \to [0, 1]$ *if the following conditions hold:*

1. *(Efficiency) (P, V) is polynomially bounded and V is polynomial-time computable.*
2. *(Completeness) If $x \in \Pi_Y$, then V accepts with probability at least $1 - c(k)$ in $(P, V)(x, 1^k)$.*
3. *(Soundness) If $x \notin \Pi_Y$, then for any P^*, V rejects with probability at least $1 - s(k)$ in $(P^*, V)(x, 1^k)$.*

We require that $c(k)$ and $s(k)$ be computable in time poly(k) *and that* $1 - c(k) > s(k) + 1/\text{poly}(k)$. *If $c \equiv 0$, then we say that the proof system has* perfect completeness. **IP** *is class of promise problems possessing interactive proofs.*

Note that the completeness and soundness errors of an interactive proof system can both be reduced to 2^{-k} by repeating the proof system poly(k) times (sequentially or in parallel) and having the new verifier accept according to majority/threshold rule.

Recall that the definition of zero knowledge is based on the notion of a simulator, which is an algorithm that simulates the verifier's view of the interaction with the prover.

Definition 2.3.4 (view of an interactive protocol) *Let (A, B) be an interactive protocol. B's* view *of (A, B) on common input x is the random variable $\langle A, B\rangle(x) = (m_1, \ldots, m_t; r)$ consisting of all the messages $m_1, \ldots, m_t$ exchanged between A and B together with the substring r of r_B containing all the random bits that B has read during the interaction.*[3]

Statistical zero knowledge requires that the statistical difference between the simulator's output distribution and the verifier's view is so small that it does not become noticeable even after polynomially many repetitions of the protocol. This is achieved by requiring that the statistical difference is *negligible*. A function $\mu : \mathbb{N} \to [0, 1]$ is *negligible* if for every polynomial $p : \mathbb{N} \to \mathbb{N}$, $\mu(k) < 1/p(k)$ for sufficiently k.

We will allow our simulators to occasionally fail by outputting a string `fail`, and we only measure the quality of the simulation conditioned on non-failure. Thus, we call a probabilistic algorithm A *useful* if $\Pr[A(x) = \texttt{fail}] \leq 1/2$ for all x and we define $\widetilde{A}(x)$ to be the output distribution of A on input x, conditioned on $A(x) \neq \texttt{fail}$.

Definition 2.3.5 (honest-verifier zero knowledge — HVSZK, HVPZK) *An interactive proof system (P, V) for a promise problem Π is said to be* honest-verifier statistical zero knowledge *if there is a useful probabilistic polynomial-time algorithm S and a negligible function $\mu(\cdot)$ such that for all $x \in \Pi_Y$ and all $k > 0$,*

$$\text{StatDiff}\left(\widetilde{S}(x, 1^k), \langle P, V\rangle(x, 1^k)\right) \leq \mu(k).$$

The negligible function μ is called the simulator deviation. *If $\mu \equiv 0$, then (P, V) is said to be* honest-verifier perfect zero knowledge. **HVSZK** *(resp.,* **HVPZK***) denotes the class of promise problems possessing honest-verifier statistical (resp., perfect) zero-knowledge proofs.*

Note that the simulation is only required to be accurate on YES instances of the promise problem; that is, when the statement being proven is true. We wanted the definition to capture the fact that the verifier should learn nothing from the "proof" (which is now actually the strategy for P). For NO instances, there is no "correct" proof (as guaranteed by soundness), so it would be somewhat strange to require that the verifier learns nothing in this case. From a cryptographic point of view, this assymetry corresponds to the idea

[3] It may seem unnatural that our notation is assymetric in that it does not allow for indicating A's view of the protocol. However, in this thesis, we will only be interested in B's view (as B corresponds to the verifier in an interactive proof), and thus we have opted for a simpler notation at the expense of generality.

that we only wish to protect parties that are behaving honestly; a prover that is trying to prove a false statement is certainly not.

Computational zero-knowledge proofs are defined by requiring that simulator's output and the verifier's view are merely indistinguishable by any polynomial-time algorithm, rather than being statistically close. This is the natural computationally bounded analogue of the definition of statistical difference.

Definition 2.3.6 (computational indistinguishability [GM84, Yao82]) *Let $X = \{X_{x,k}\}_{x \in L, k \in \mathbb{N}}$ and $Y = \{Y_{x,k}\}_{x \in L, k \in \mathbb{N}}$ be ensembles of probability distributions indexed by strings x in a set L and natural numbers k (the* security parameter*). X and Y are said to be* computationally indistinguishable *if for every nonuniform probabilistic polynomial-time algorithm ("distinguisher") D, there is a negligible function $\mu(\cdot)$ such that*

$$\left| \Pr\left[D(x, 1^k, X_{x,k}) = 1\right] - \Pr\left[D(x, 1^k, Y_{x,k}) = 1\right] \right| \le \mu(k) \qquad \forall x \in L.$$

Definition 2.3.7 (honest-verifier zero knowledge — HVCZK) *An interactive proof system (P, V) for a promise problem Π is said to be* honest-verifier computational zero knowledge *if there is a useful probabilistic polynomial-time algorithm S such that*

$$\left\{\widetilde{S}(x, 1^k)\right\}_{x \in \Pi_Y, k \in \mathbb{N}} \qquad \textit{and} \qquad \left\{\langle P, V\rangle(x, 1^k)\right\}_{x \in \Pi_Y, k \in \mathbb{N}}$$

are computationally indistinguishable. **HVCZK** *denotes the class of promise problems possessing honest-verifier computational zero-knowledge proofs.*

A remark on nonuniformity. Note that we have allowed the distinguisher to be nonuniform in the definition of computational indistinguishability. While the definitions can be made in the uniform setting, the theory of computational zero knowledge is "cleaner" with a nonuniform definition. Already, some sort of nonuniformity is implicit in the notion of zero knowledge, because the verifier and distinguisher are given the input x, which can be regarded as nonuniform "advice". In addition, several researchers [FS89, GMR89, GO94, Ore87, TW87] have observed that that allowing some sort of nonuniformity (or "auxiliary input") is important in proving some some basic results about zero knowledge.

For example, suppose we repeat an **HVCZK** proof several times, either sequentially or in parallel. Intuitively, since one run of the simulator is computationally indistinguishable from one execution of the proof system, t independent runs of the simulator should be computationally indistinguishable from t independent executions of the proof system, and hence the repeated proof system should still be **HVCZK**. Unfortunately, this "fact" that computational indistinguishability is preserved under taking many independent samples only is guaranteed when either the distinguishers are permitted to be nonuniform or both distributions are polynomial-time samplable (see, *e.g.,* [GS98]). Since the executions of the proof system are not necessarily polynomial-time samplable, we must take nonuniform distinguishers. Having adopted a nonuniform definition, it can be proven using the standard "hybrid" argument of [GM84] that honest-verifier computational zero knowledge is preserved under both sequential and parallel repetition.

As discussed in Chapter 6, nonuniformity becomes even more important when we discuss cheating verifiers. It is possible to develop the theory of zero knowledge in the uniform setting, as done by Goldreich [Gol93]; there, the definitions are modified to require that it is *infeasible* to find YES instances x on which the prover leaks knowledge (rather than requiring this for *all* YES instances x). Most of the results we prove about computational zero knowledge also hold in that setting, but for simplicity, we only discuss the nonuniform versions.

2.4 Contrast with the GMR definition

The definitions we have given above differ from the original definitions given by Goldwasser, Micali, and Rackoff [GMR89] (henceforth called *the GMR definitions*) in several ways, which we outline below.

Honest verifiers. The most important difference is that we have only defined honest-verifier zero knowledge, which formalizes the requirement that the verifier should learn nothing from the interaction *if it follows the specified protocol.* The general definition of zero knowledge, which is important in cryptographic applications, requires that even cheating verifiers which deviate from the protocol should learn nothing. Roughly speaking, this is formalized by requiring that, for every (polynomial-time) verifier strategy, there exists a corresponding simulator. However, there are a number of subtle issues in the definition. For this reason, together with the fact that we will be focusing on honest-verifier zero knowledge for the first few chapters of this thesis, we postpone the definitions of zero-knowledge proofs for cheating verifers to Chapter 6. However, in informal discussions, we still will refer to the classes of problems possessing proof systems that are zero-knowledge against cheating verifiers, which we denote by **SZK**, **PZK**, and **CZK**. In that Chapter 6, we will show how to transform any honest-verifier statistical zero-knowledge proof (and any public-coin honest-verifier computational zero-knowledge proof) into one which are zero knowledge even against cheating verifiers. That is, **HVSZK** = **SZK**. Fortnow [For89] was the first to formally define and investigate honest-verifier zero-knowledge proofs. (His terminology was "trusted verifier").

The security parameter. Another difference between our definition and the GMR definition is our use of a security parameter to control the error parameters (completeness, soundness, and simulator deviation). The original definition measures these as a function of the input length $|x|$, and in particular only requires that the simulator deviation be negligible as a function of $|x|$. We feel that it is unnatural to tie the error parameters to the input length in this manner, as one may wish to prove even short statements with very high "security". The use of a separate security parameter to control the errors has appeared in various places in the literature such as [BP89, KMO89] and has become standard in the literature on "noninteractive" zero-knowledge proofs (*e.g.,* [FLS99, Kil94, KP98]).

With completeness and soundness errors, this definitional choice is mainly a philosophical one, as it does not change the class of problems possessing interactive proofs (since even a constant error probability can be made exponentially small in k by repeating the proof system $O(k)$ times.) With the simulator deviation, it is not *a priori* clear that, given a proof system with simulator deviation that is a function of $|x|$, one can obtain one whose simulator deviation is a function of a security parameter k (though we will prove it later in this thesis).

One nice property of our security-parameter based definition is that it allows one to prove that **HVSZK** is closed under reductions.

Proposition 2.4.1 *If Π has an honest-verifier statistical zero-knowledge proof with simulator deviation $\mu(\cdot)$, and Γ (Karp-)reduces to Π, then Γ has an honest-verifier statistical zero-knowledge proof with simulator deviation $\mu(\cdot)$. Thus,* **HVSZK** *and* **HVPZK** *are closed under (Karp) reductions.*

Proof: Let (P, V) be the statistical zero-knowledge proof for Π and f be the reduction from Γ to Π. A statistical zero-knowledge proof (P', V') for Γ can be obtained as follows: On common input $(x, 1^k)$, P' and V' execute the protocol (P, V) on common input $(f(x), 1^k)$. A simulator for (P', V') with deviation $\mu(\cdot)$ can be obtained by running the simulator for (P, V) on input $(f(x), 1^k)$. ■

The reason such a proposition cannot be proved so easily for the GMR definition is that $f(x)$ might be much shorter than x, which means $\mu(|f(x)|)$, which is the simulator deviation achieved by executing (P, V) and S on input $f(x)$ (according to the GMR definition), might not be negligible as a function of $|x|$. However, there are occasions when the security parameter is irrelevant (*e.g.,* perfect zero-knowledge proofs with constant completeness and soundness errors), and, in those cases, we will often omit the security parameter from the notation for sake of clarity.

Expected polynomial-time simulators. Two more differences between our definition and GMR's is that they allow expected polynomial-time simulators, but do not allow the simulator to fail. Following Goldreich [Gol95], we require strict polynomial-time simulators, but do allow the simulator to fail. The reason for this modification is that strict polynomial-time is better behaved and less controversial as formalization of "efficient computation" than expected polynomial time. Our requirement is more stringent, because a strict polynomial-time simulator which may fail can be converted into an expected polynomial-time one which never fails (by running the simulator many times independently until it succeeds). In fact, for statistical zero knowledge, one can remove the need for failure without passing to expected polynomial time: running the simulator polynomially many times makes the failure probability exponentially small, and this can be absorbed into the simulator deviation. In contrast, it is not clear how to convert an expected polynomial-time simulator into a strict polynomial-time simulator without incurring a nonnegligible increase in simulator deviation.

Weak statistical zero knowledge. A notion that captures all the ways in which the GMR definition is weaker than ours (and more) is that of **weak-HVSZK** (analogous to **weak-SZK** considered in [DOY97]):

Definition 2.4.2 (weak statistical zero knowledge — weak-HVSZK)
An interactive proof system (P, V) for a promise problem Π is weak honest-verifier statistical zero knowledge *if for every $c > 0$, there is a useful probabilistic polynomial-time*

algorithm S_c such that, for all but finitely many $x \in \Pi_Y$,

$$\mathrm{StatDiff}\left(S_c(x), \langle P, V\rangle(x, 1^{|x|})\right) \leq \frac{1}{|x|^c}.$$

weak-HVSZK *denotes the class of promise problems possessing weak honest-verifier statistical zero-knowledge proofs.*

Thus, the simulator deviation can be made smaller than any inverse polynomial, but the simulator itself (and, in particular, its running time) can depend on the particular polynomial. Any interactive proof with an expected polynomial-time simulator of negligible simulator deviation (*i.e.,* meeting the GMR definition for honest verifiers) also satisfies the above definition: truncating simulator executions after $|x|^c$ times the expected number of steps increases the simulator deviation by at most $1/|x|^c$. In Section 4.3, we will show how to convert weak honest-verifier statistical zero-knowledge proofs into ones meeting the more stringent Definition 2.3.5; that is, **weak-HVSZK** = **HVSZK**.

2.5 Complexity aspects of interactive proofs

There are a number of complexity issues that arise with interactive proofs and statistical zero knowledge which we will address in this thesis. One issue that arose in the interactive proof for Graph Nonisomorphism presented in Section 2.1 was that it was essential that the verifier's random coins were kept hidden from the prover. Proof systems in which the random coins used by the verifier at each round are revealed to the prover at the same time are called *public-coin* proof systems. Since the verifier's messages are a deterministic function of the input x and the coin flips, the prover can be given just the coin flips themselves at each round without loss of generality.

Definition 2.5.1 (public-coin protocols [BM88]) *An interactive protocol (A, B) is* public coin *for B if in every execution of the protocol, the string of random coins accessed by B can be written $r_1 r_2 \cdots r_t \in \{0,1\}^*$, so that B's i'th message m_{2i} equals $r_i \in \{0,1\}^{\ell_i}$, ℓ_i is solely a function of $(x, m_1, m_2, \ldots, m_{2i-1})$, and m_{2t+2} is the last message computed by B. An interactive proof (P, V) is* public coin *if, for every P^*, (P^*, V) is public coin for V.*

Public-coin proofs are also known as *Arthur–Merlin games*, so we often denote the prover in such proof systems by M (for "Merlin") and the verifier by A (for "Arthur"). Sometimes we will refer to general interactive proofs as *private-coin* proofs to emphasize the difference with public-coin ones. Public-coin proof systems are extremely computation efficient for the verifier, as the only computation the verifier needs to do is to compute its last message m_{2t+2} (`accept` or `reject`) and possibly the number of coins to send at each round (which is usually a simple function of $|x|$). Amazingly, every problem possessing an interactive proof also possesses a public-coin interactive proof [GS89]. In Chapter 5, we will prove an analogous theorem for statistical zero-knowledge proofs, a result first obtained by Okamoto [Oka96].

Some other important complexity measures for interactive proofs are the amount of interaction, as measured by the number of messages exchanged,[4] and the number of bits of communication.

Definition 2.5.2 (number of messages & communication) *We say that an interactive protocol (A, B) exchanges m* messages *on input x, if for every choice of the random coins for A and B, the number of messages computed before the first* `accept/reject/halt` *message is at most m (or $m+1$, if the first message m_1 of A is always the empty string). The class of promise problems possessing interactive proofs which exchange a* constant *number of messages is denoted* **AM**.

We say that an interactive protocol (A, B) has A-to-B (resp., B-to-A) communication *c on input x, if for every choice of the random coins for A and B, the sum of the lengths of the messages computed by A (resp., B) (excluding an* `accept/reject/halt` *message) is at most c. (A, B) has* total communication *c on input x, if the sum of the lengths of all messages computed (by both A and B) is at most c (again, excluding* `accept/reject/halt` *messages).*

In Section 4.1, we will show that every problem in **HVSZK** has an extremely efficient honest-verifier statistical zero-knowledge proof, namely, a 2-message proof system with 1 bit of prover-to-verifier communication.

[4]In the literature, sometimes the term "rounds" is used to measure the amount of interaction. However, its usage is not consistent — some authors count each message as one round, while others refer to a pair of A/B messages as a round. To avoid ambiguity, we speak only of the number of messages exchanged.

Chapter 3

Complete Problems

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** [LFKN92, Sha92]. More recently, the celebrated **PCP** theorem characterizing **NP** was proven by designing efficient probabilistically checkable proofs for a specific **NP**-complete language [ALM$^+$98, AS98].

In this chapter, we present two complete problems for **HVSZK**, the class of problems possessing statistical zero-knowledge proofs against an honest verifier. For traditional complexity classes, such as **NP** and **PSPACE**, the construction of natural complete problems has become a routine task. However, it may come as a surprise that **HVSZK**, which is defined in terms of two interacting machines and a simulator, has complete problems which makes *no reference to interaction or zero-knowledge.* In subsequent chapters, we use these complete problems problems not as a negative tool, but as a positive tool to derive general results about the entire class **HVSZK**.

Organization. Recall that proving that a problem Π is *complete* for **HVSZK** involves both proving that $\Pi \in$ **HVSZK** and that every problem in **HVSZK** reduces to Π. In this chapter, we do this for two problems, called STATISTICAL DIFFERENCE (SD) and ENTROPY DIFFERENCE (ED). These two problems will be simultaneously proven complete for **HVSZK** via a "circle of reductions" composed of the following three results.

1. SD $\in$ **HVSZK** (Section 3.1).

2. Every problem in **HVSZK** reduces to ED (Section 3.3).

3. ED reduces to SD (Section 3.4).

The combination of Steps 2 and 3 imply that every problem in **HVSZK** reduces to SD, and the combination of Steps 1 and 3 imply that ED $\in$ **HVSZK**, so it follows that both SD and ED are complete for **HVSZK**.

Section 3.2 contains a motivating warm-up to Step 2. Namely, we show that every problem possessing a *public-coin* honest-verifier statististical zero-knowledge proof reduces to SD.

3.1 Statistical Difference

The first problem we show to be complete for **HVSZK** is called Statistical Difference. Roughly speaking, it is the problem of deciding whether a pair of "efficiently samplable" distributions are statistically close or statistically far apart, as measured by the statistical difference metric. In order to define the problem formally, we must make precise the notion of an efficiently samplable distribution. To do this, we view Boolean circuits as sampling algorithms, whose inputs are random bits.

Definition 3.1.1 (distributions encoded by circuits) *Let X be a Boolean circuit (with AND, OR, and NOT gates, unbounded fan-in and fan-out) with m input gates and n output gates. The* distribution encoded by X *is the distribution induced on $\{0,1\}^n$ by evaluating X on a uniformly selected string from $\{0,1\}^m$. By abuse of notation, we also write X for the distribution defined by X.*

Since circuits can be evaluated in time polynomial in their size, yet are rich enough to encode general (*e.g.,* Turing machine) computations, they effectively capture the notion of an "efficiently samplable distribution." Now we can define the promise problem Statistical Difference.

Definition 3.1.2 Statistical Difference *is the promise problem* $\mathrm{SD} = (\mathrm{SD}_Y, \mathrm{SD}_N)$, *where*

$$\begin{aligned}\mathrm{SD}_Y &= \{(X,Y) \,:\, \mathrm{StatDiff}\,(X,Y) \geq 2/3\} \\ \mathrm{SD}_N &= \{(X,Y) \,:\, \mathrm{StatDiff}\,(X,Y) \leq 1/3\}\,.\end{aligned}$$

Above, X and Y are circuits encoding probability distributions, as in Definition 3.1.1.

In order to show that SD is complete for **HVSZK**, we need to prove two things: that SD $\in$ **HVSZK**, and that every problem in in **HVSZK** reduces to SD. This section is devoted to the former task. To do this, we generalize the Graph Nonisomorphism proof system given in Section 2.1. Recall that the analysis of that proof system is based on the observation that two probability distributions (obtained by taking a random isomorphic copy of one graph or the other) either have disjoint supports or are identical, depending on whether the input is a yes or no instances, respectively. This motivates considering a restriction of SD in which the distributions are either disjoint or identical (*as distributions, not as circuits*). We call this problem $\mathrm{SD}^{1,0}$ because it can obtained by replacing the thresholds of $2/3$ and $1/3$ in the definition of SD with 1 and 0, respectively. Actually, we consider a number of variants of SD, parametrized by the thresholds.

Definition 3.1.3 (variants of SD) *For any constants* $0 \leq \beta < \alpha \leq 1$, *the promise problem* $\mathrm{SD}^{\alpha,\beta} = (\mathrm{SD}_Y^{\alpha,\beta}, \mathrm{SD}_N^{\alpha,\beta})$ *is given by*

$$\begin{aligned} \mathrm{SD}_Y^{\alpha,\beta} &= \{(X,Y) \,:\, \mathrm{StatDiff}\,(X,Y) \geq \alpha\} \\ \mathrm{SD}_N^{\alpha,\beta} &= \{(X,Y) \,:\, \mathrm{StatDiff}\,(X,Y) \leq \beta\}. \end{aligned}$$

Above, X *and* Y *are circuits encoding probability distributions, as in Definition 3.1.1.*

3.1.1 A basic proof system

Following the intuition from the Graph Nonisomorphism proof system, a natural way to construct a proof system for any of these variants of SD is to test whether the prover can distinguish random sample from the first distribution from a random sample from the second distribution. The prover's best strategy is to simply guess that the sample came from the distribution which assigns it more probability mass. This intuition motivates the basic proof system given in Protocol 3.1.4.

Protocol 3.1.4: Basic proof system (P, V) for variants of SD

Input: Circuits X_0 and X_1 (each with m input gates and n output gates)

1. V: Select $b \leftarrow \{0,1\}$. Obtain a sample $x \leftarrow X_b$ (by choosing $r \leftarrow \{0,1\}^m$ and letting $x = X_b(r)$). Send x to P.
2. P: If $\Pr[X_0 = x] > \Pr[X_1 = x]$, let $c = 0$. Else let $c = 1$. Send c to V.
3. V: If $c = b$, accept. Otherwise, reject.

We first analyze this protocol for $\mathrm{SD}^{1,0}$. It is clear that if X_0 and X_1 have disjoint supports, then the prover strategy given in Protocol 3.1.4 will succeed with probability 1. On the other hand, if X_0 and X_1 are identical as distributions, then b is independent of x, so the prover can guess b from x with probability at most $1/2$, no matter what strategy it follows. Thus, we have

Claim 3.1.5 *Protocol 3.1.4 is an interactive proof system for* $\mathrm{SD}^{1,0}$ *with perfect completeness and soundness error* $1/2$.

When the distributions are disjoint, all the verifier sees is the prover's (correct) guess c for b, which is a value the verifier already "knows." This suggests that the proof system is zero knowledge, and thus we consider a simulator (given in Algorithm 3.1.6), analogous to Algorithm 2.1.4.

It follows readily from the fact that the prover guesses correctly with probability 1 that the output distribution of Algorithm 3.1.6 and the verifier's view of Protocol 3.1.4 are identical when X_0 and X_1 have disjoint supports. Thus, we have:

Algorithm 3.1.6: Simulator for basic SD **proof system**

Input: Circuits X_0 and X_1 (each with m input gates and n output gates)

1. Select $b \leftarrow \{0,1\}$. Choose $r \leftarrow \{0,1\}^m$ and let $x = X_b(r)$.
2. Let $c = b$.
3. Output $(x, c; b, r)$

Proposition 3.1.7 *Protocol 3.1.4 is an (honest-verifier) perfect zero-knowledge proof system for* $\mathrm{SD}^{1,0}$.

Intuitively, it seems that our analysis of this proof system should hold "approximately" when the distributions are either statistically very far apart or statistically very close instead of being disjoint or identical, respectively. This is indeed the case, and using statistical difference as a measure of closeness, we get exact expressions for the error parameters.

Lemma 3.1.8 *When X_0 and X_1 have statistical difference δ, the prover strategy given in Protocol 3.1.4 makes the verifier accept with probability exactly $(1+\delta)/2$, and no prover strategy succeeds with greater probability. Moreover, the output of Algorithm 3.1.6 has statistical difference exactly $(1-\delta)/2$ from the verifier's view of $(P,V)(X_0,X_1)$.*

In order to prove Lemma 3.1.8, we first need to get a slightly better understanding of the statistical difference metric.

Fact 3.1.9 *Let X and Y be probability distributions (or random variables) on a discrete universe $\mathcal{U}$, let $S_X = \{x \in \mathcal{U} \ : \ \Pr[X = x] > \Pr[Y = y]\}$, and define S_Y analogously. Then*

$$\mathrm{StatDiff}(X,Y) = \Pr[X \in S_X] - \Pr[Y \in S_X] = \Pr[Y \in S_Y] - \Pr[X \in S_Y].$$

Proof: For any set S, $\Pr[X \in S] = \sum_{x \in S} \Pr[X = x]$ and similarly for Y. So $\delta(S) \stackrel{\text{def}}{=} \Pr[X \in S] - \Pr[Y \in S]$ is increased by adding elements of S_X to S, decreased by adding elements of S_Y to S, and is unchanged by adding points on which X and Y have the same mass. Thus, the maximum (positive) value $\delta(S)$ can take on is achieved by $S = S_X$ and the minimum (negative) value is achieved by $S = S_Y$. The maximum positive value and the minimum negative value of $\delta(S)$ must have the same magnitude, since $\delta\left(\overline{S}\right) = -\delta(S)$. Hence,

$$\begin{aligned} \mathrm{StatDiff}(X,Y) &= \max_S |\delta(S)| \\ &= \Pr[X \in S_X] - \Pr[Y \in S_X] \\ &= \Pr[Y \in S_Y] - \Pr[X \in S_Y]. \qquad \blacksquare \end{aligned}$$

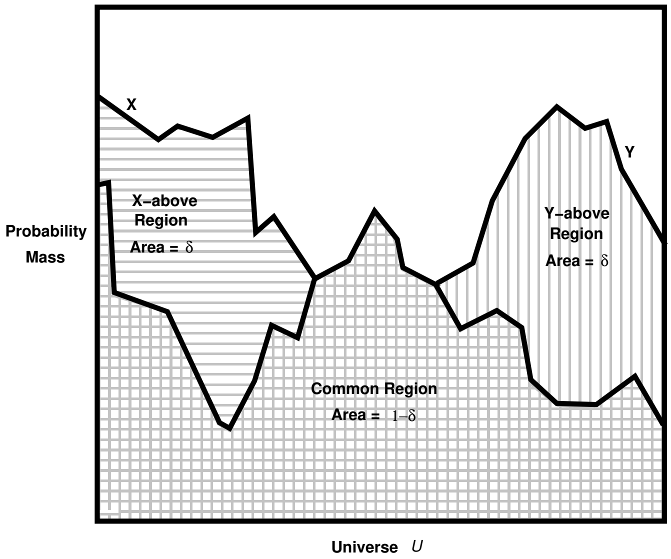

Figure 3-1: Statistical difference as area

Fact 3.1.9 gives us another way of viewing statistical difference — as area between curves. Suppose we graph the mass functions of two distributions X and Y (so the area under each of these curves is 1). Then, Fact 3.1.9 says that the region that is above Y and below X has area δ, the region that is above X and below Y has area δ, and the region that is below both has area $1 - \delta$. We call these regions the *X-above region*, the *Y-above region* and the *common region*, respectively. See Figure 3-1.

Proof of Lemma 3.1.8: From the description of statistical difference as area, we can give an alternative process that induces the same distribution on (b, x) as the verifier's strategy in Protocol 3.1.4:

1. Flip a biased coin d that is 0 with probability $1 - \delta$ and 1 with probability δ.
2. If $d = 0$:
   (a) Uniformly select a point in the common region, and let x be corresponding element of $\{0, 1\}^n$
   (b) Uniformly select $b \in \{0, 1\}$.
3. If $d = 1$:
   (a) Uniformly select $b \in \{0, 1\}$.
   (b) If $b = 0$, uniformly select a point from the X_0-above region, and let x be the corresponding element of $\{0, 1\}^n$.
   (c) If $b = 1$, uniformly select a point from the X_1-above region, and let x be the corresponding element of $\{0, 1\}^n$.

4. Output (b, x).

From this description, it is clear that, when $d = 0$, b is independent of x and the prover's success probability is exactly $1/2$ no matter what strategy is used. In addition, the prover strategy specified in Protocol 3.1.4 perfectly distinguishes the X_0-above and X_1-above regions and therefore succeeds with probability 1 when $d = 1$. Hence, the specified prover strategy is optimal, and its success probability is exactly

$$\frac{1}{2} \cdot \Pr[d = 0] + 1 \cdot \Pr[d = 1] = \frac{1}{2} \cdot (1 - \delta) + 1 \cdot \delta = \frac{1 + \delta}{2}.$$

To analyze the simulator deviation, notice that the only transcripts that occur with greater probability in the verifier's view than in the simulator's output are those in which verifier rejects. Since these occur with probability zero in the simulator, the statistical difference is exactly the prover's failure probability, which is $1 - (1 + \delta)/2 = (1 - \delta)/2$. ■

From Lemma 3.1.8, we immediately obtain:

Proposition 3.1.10 *For any constants $0 \leq \beta < \alpha \leq 1$, Protocol 3.1.4 is an interactive proof for $\mathrm{SD}^{\alpha,\beta}$ with completeness error $(1 - \alpha)/2$ and soundness error $(1 + \beta)/2$.*

Proposition 3.1.11 *For every constant $0 \leq \beta < 1$, Protocol 3.1.4 is an honest-verifier perfect zero-knowledge proof for $\mathrm{SD}^{1,\beta}$ with perfect completeness and soundness error $(1 + \beta)/2$.*

However, Lemma 3.1.8 does not yet give a zero-knowledge proof for $\mathrm{SD} = \mathrm{SD}^{2/3,1/3}$ as desired, because the simulator deviation would be a constant $(1/6)$, rather than a negligible function. One way to obtain a negligible simulator deviation would be to give a transformation which maps a pair of circuits with statistical difference at least $2/3$ to a pair with statistical difference extremely close to 1 (while keeping an initial statistical difference of at most $1/3$ bounded away from 1). In the next section, we show how to achieve this.

3.1.2 A polarization lemma

Lemma 3.1.12 (Polarization Lemma)[1] *Let $\alpha, \beta \in [0, 1]$ be any two constants such that $\alpha^2 > \beta$ (e.g., $\alpha = 2/3$, $\beta = 1/3$). There is a polynomial-time computable function $\mathrm{Polarize}_{\alpha,\beta}$ that takes a triple $(X_0, X_1, 1^k)$, where X_0 and X_1 are distributions encoded by circuits, and outputs a pair of circuits (Y_0, Y_1) such that*

$$\begin{aligned} \mathrm{StatDiff}(X_0, X_1) \geq \alpha \quad &\Rightarrow \quad \mathrm{StatDiff}(Y_0, Y_1) \geq 1 - 2^{-k} \\ \mathrm{StatDiff}(X_0, X_1) \leq \beta \quad &\Rightarrow \quad \mathrm{StatDiff}(Y_0, Y_1) \leq 2^{-k} \end{aligned}$$

The usefulness of the Polarization Lemma comes from the fact that the two distributions it produces can be treated almost as if they were disjoint or identically distributed, respectively (*i.e.*, statistical difference 0 and 1, respectively). Indeed, in the next section, we show

[1] The Polarization Lemma stated here is called the Amplification Lemma in [SV97]. The name was changed in [SV99] to stress that the Polarization Lemma does not merely increase statistical difference.

how this Polarization Lemma can be used to augment Protocol 3.1.4 and obtain a statistical zero-knowledge proof for SD. This section is devoted to the proof of the Polarization Lemma. The challenge in proving the lemma is that we need to increase statistical difference in some cases and decrease statistical difference in other cases. We will obtain such a transformation by combining two complementary transformations — one which increases statistical difference and one which decreases statistical difference. The analysis of both of these transformations will make use of yet another formulation of statistical difference, this time in terms of probability mass vectors.

If X is a probability distribution on a discrete universe $\mathcal{U}$, then we can view its mass function as a vector in $\mathbb{R}^{\mathcal{U}}$, which we denote by $\vec{X}$. Fact 3.1.9 says that the area between the graphs of the mass functions of distributions X and Y is exactly twice their statistical difference (see Figure 3-1). The area between the graphs is exactly the ℓ_1-distance between the vectors $\vec{X}$ and $\vec{Y}$, where the ℓ_1-norm of a vector $\vec{v} \in R^S$ is $|\vec{v}|_1 \stackrel{\text{def}}{=} \sum_{i \in S} |v_i|$. Thus, we have:

Fact 3.1.13 $\text{StatDiff}(X, Y) = \frac{1}{2} \left| \vec{X} - \vec{Y} \right|_1$.

We now focus on increasing statistical difference. Intuitively, taking many independent copies of two distributions should increase their distinguishability and drive the statistical difference to 1. Thus, we now analyze the behavior of statistical difference with respect to independence. In order to do so, we express independence in terms of probability mass vectors.

Recall that the *tensor product* of vectors $\vec{v} \in \mathbb{R}^S$ and $\vec{w} \in \mathbb{R}^T$ is the vector $\vec{v} \otimes \vec{w} \in \mathbb{R}^{S \times T}$ with $(\vec{v} \otimes \vec{w})_{i,j} = v_i \cdot w_j$. Note that, any vectors $\vec{v}$ and $\vec{w}$, $|\vec{v} \otimes \vec{w}|_1 = |\vec{v}|_1 \cdot |\vec{w}|_1$. Now, observe that a pair of jointly distributed random variables (X, Y) are independent iff $\overrightarrow{(X, Y)} = \vec{X} \otimes \vec{Y}$. For this reason, for any two distributions X and Y, we write $X \otimes Y$ for the distribution obtained by taking a sample of X followed by an independent sample of Y, and $\otimes^k X$ for the distribution consisting of k independent samples of X.

The above observations enable us to bound the effect of independence on statistical difference.

Fact 3.1.14 *Let $X = (X_0, X_1)$ be a distribution in which X_0 and X_1 are independent and $Y = (Y_0, Y_1)$ be one in which Y_0 and Y_1 are independent. Then*

$$\text{StatDiff}(X, Y) \leq \text{StatDiff}(X_0, X_1) + \text{StatDiff}(Y_0, Y_1).$$

Proof:

$$\begin{aligned}
\text{StatDiff}(X, Y) &= \frac{1}{2}\left|\vec{X_0} \otimes \vec{X_1} - \vec{Y_0} \otimes \vec{Y_1}\right|_1 \\
&\leq \frac{1}{2}\left|\vec{X_0} \otimes \vec{X_1} - \vec{Y_0} \otimes \vec{X_1}\right|_1 + \frac{1}{2}\left|\vec{Y_0} \otimes \vec{X_1} - \vec{Y_0} \otimes \vec{Y_1}\right|_1 \\
&= \frac{1}{2}\left|(\vec{X_0} - \vec{Y_0}) \otimes \vec{X_1}\right|_1 + \frac{1}{2}\left|\vec{Y_0} \otimes (\vec{X_1} - \vec{Y_1})\right|_1 \\
&= \frac{1}{2}\left|\vec{X_0} - \vec{Y_0}\right|_1 \cdot \left|\vec{X_1}\right|_1 + \frac{1}{2}\left|\vec{Y_0}\right|_1 \cdot \left|\vec{X_1} - \vec{Y_1}\right|_1 \\
&= \text{StatDiff}(X_0, Y_0) + \text{StatDiff}(X_1, Y_1). \qquad \blacksquare
\end{aligned}$$

Of course, Fact 3.1.14 does not accomplish our goal; it only gives an upper bound on the effect of independent copies on statistical difference, whereas want a lower bound. The following Direct Product Lemma shows that statistical difference goes to 1 exponentially fast when we take independent copies. The lemma is reminiscent of a Chernoff bound, and indeed, that is how the proof will proceed.

Lemma 3.1.15 (Direct Product Lemma) *Let X and Y be distributions such that* $\mathrm{StatDiff}(X, Y) = \delta$. *Then for all $k \in \mathbb{N}$,*

$$1 - 2e^{-k\delta^2/2} \leq \mathrm{StatDiff}\left(\otimes^k X, \otimes^k Y\right) \leq k\delta$$

Proof: The upper bound of $k\delta$ follows immediately from Fact 3.1.14, so we proceed to the proof of the lower bound. Recall, from the definition of statistical difference, that there exists a set S such that

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

Let $p = \Pr[Y \in S]$. Then, $\Pr[X \in S] = p + \delta$. Hence, in k independent samples of X, the expected number of samples that lie in S is $(p + \delta)k$, whereas in k independent samples of Y, the expected number of samples that lie in S is pk.

The Chernoff Bound (Theorem A.1) tells us that the probability that *at least* $(p + \frac{\delta}{2})k$ components of $\otimes^k Y$ lie in S is at most $\exp(-k\delta^2/2)$, whereas the probability that *at most* $(p + \frac{\delta}{2})k$ components of $\otimes^k X$ lie in S is at most $\exp(-k\delta^2/2)$. Let S' be the set of all k-tuples that contain more than $(p + \frac{\delta}{2})k$ components that lie in S. Then we have,

$$\mathrm{StatDiff}\left(\otimes^k X, \otimes^k Y\right) \geq \Pr\left[\otimes^k X \in S'\right] - \Pr\left[\otimes^k Y \in S'\right] \geq 1 - 2e^{-k\delta^2/2}. \qquad \blacksquare$$

Given the Direct Product Lemma, a first attempt at making Protocol 3.1.4 statistical zero knowledge for SD would be to replace each distribution with many independent copies of itself. If the original pair of distributions was YES instance (*i.e.,* with statistical difference at least 2/3), their statistical difference will now be exponentially close to 1, and hence the simulation will be statistically close by Lemma 3.1.8. Unfortunately, this will also drive the statistical difference of some NO instances (like those with statistical difference 1/3) towards 1 and this will destroy the soundness of the proof system.

However, the Direct Product Lemma does drive larger values of statistical difference to 1 more quickly than it drives smaller values to 1 (as illustrated by the upper bound of $k\delta$), so it is a step in the right direction. Thus, we will seek a complementary technique which decreases the statistical difference to 0, with small values going to 0 faster than large values. By alternating the two procedures, we will manage to increase the statistical difference for YES instances and decrease it for NO instances.

To figure out how one might decrease the statistical difference between two distributions in a controlled manner, we consider how one might decrease the prover's success probability in Protocol 3.1.4. One natural idea would be to repeat the protocol many times independently and see if the prover guesses the correctly in all executions. That is, the verifier would choose $b_1, \ldots, b_k \in \{0, 1\}$ uniformly and independently at random, obtain samples $z_1, \ldots, z_k$ independently from $X_{b_1}, \ldots, X_{b_k}$, respectively, send these samples to the prover, and see if the prover can guess all the b_i's. Alternatively, one might instead ask the prover to guess the exclusive-OR of all the b_i's. Looking at the proof of Lemma 3.1.8, one sees that the prover will have no information about b_i if the "common region" is hit in the i'th execution. If the prover has no information about even just one b_i, then it also has no information about the exclusive-OR, and hence the success probability will be exactly 1/2. The probability that the common region is hit in the i'th execution is $1-\delta$, where $\delta = \text{StatDiff}(X_0, X_1)$, so the probability that it is hit in at least one execution is $1-\delta^k$. Thus, the prover's success probability goes to 1/2 exponentially fast with k. This suggests that the two distributions on k-tuples obtained by conditioning on the exclusive-OR being 0 and 1, respectively, in this repeated protocol have statistical difference δ^k. This is formalized by the following XOR Lemma.

Lemma 3.1.16 (XOR Lemma) *There is a polynomial-time computable function that maps a triple $(X_0, X_1, 1^k)$, where X_0 and X_1 are circuits, to a pair of circuits (Y_0, Y_1) such that* $\text{StatDiff}(Y_0, Y_1) = \text{StatDiff}(X_0, X_1)^k$. *Specifically, Y_0 and Y_1 are defined as follows:*

Y_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 $X_{b_1} \otimes \cdots \otimes X_{b_k}$.*
Y_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 $X_{b_1} \otimes \cdots \otimes X_{b_k}$.*

The motivation given above actually is sufficient to prove the lemma, but instead, we will do a calculation using the ℓ_1 description of statistical difference to see what happens when we combine just two pairs of distributions in this fashion. This construction is a generalization of the technique used by De Santis *et. al.* [DDPY94] to represent the logical AND of statements about Graph Nonisomorphism.

Proposition 3.1.17 *Let X_0, X_1, Y_0, Y_1 be any random variables, and define the following pair of random variables:*

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

Then $\text{StatDiff}(Z_0, Z_1) = \text{StatDiff}(X_0, X_1) \cdot \text{StatDiff}(Y_0, Y_1)$.

Proof:

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

Proposition 3.1.17 and an induction argument establish Lemma 3.1.16. Yao's XOR Lemma [Yao82] (see also [GNW95]) can be seen as an analogue of Lemma 3.1.16 in the computational setting, where the analysis is much more difficult.

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. Alternating these procedures should "polarize" large and small values of statistical difference, pushing them closer to 1 and 0, respectively, and yield Lemma 3.1.12. This following proof confirms this intuition.

Proof of Lemma 3.1.12: Let $\lambda = \min\{\alpha^2/\beta, 2\} > 1$, and let $\ell = \lceil \log_\lambda 4k \rceil = O(\log k)$. Apply the XOR Lemma (Lemma 3.1.16) to the triple $(X_0, X_1, 1^\ell)$ to produce (X_0', X_1') such that

$$\begin{aligned}
\mathrm{StatDiff}\,(X_0, X_1) \geq \alpha \quad &\Rightarrow \quad \mathrm{StatDiff}\left(X_0', X_1'\right) \geq \alpha^\ell \\
\mathrm{StatDiff}\,(X_0, X_1) \leq \beta \quad &\Rightarrow \quad \mathrm{StatDiff}\left(X_0', X_1'\right) \leq \beta^\ell
\end{aligned}$$

Let $m = \lambda^\ell/(2\alpha^{2\ell}) \leq 1/(2\beta^\ell)$. Notice that $m \leq \mathrm{poly}(k)$, since $\ell = O(\log k)$, $\lambda \leq 2$, and α is a constant. Now apply the Direct Product construction, defining $X_0'' = \otimes^m X_0'$ and $X_1'' = \otimes^m X_1'$. Then, by Lemma 3.1.15,

$$\begin{aligned}
\mathrm{StatDiff}\,(X_0, X_1) \geq \alpha \quad &\Rightarrow \quad \mathrm{StatDiff}\left(X_0'', X_1''\right) \geq 1 - 2\exp\left(\left(\frac{\lambda^\ell}{2\alpha^{2\ell}}\right)\cdot\frac{(\alpha^\ell)^2}{2}\right) \geq 1 - 2e^{-k} \\
\mathrm{StatDiff}\,(X_0, X_1) \leq \beta \quad &\Rightarrow \quad \mathrm{StatDiff}\left(X_0'', X_1''\right) \leq (1/2\beta^\ell)\cdot\beta^\ell = 1/2
\end{aligned}$$

Finally, apply the XOR Lemma (Lemma 3.1.16) one more time to $(X_0'', X_1'', 1^k)$ to produce (Y_0, Y_1) such that

$$\begin{aligned}
\mathrm{StatDiff}\,(X_0, X_1) \geq \alpha \quad &\Rightarrow \quad \mathrm{StatDiff}\,(Y_0, Y_1) \geq (1 - 2e^{-k})^k \geq 1 - 2ke^{-k} > 1 - 2^{-k} \\
\mathrm{StatDiff}\,(X_0, X_1) \leq \beta \quad &\Rightarrow \quad \mathrm{StatDiff}\,(Y_0, Y_1) \leq 1/2^k
\end{aligned}$$

(as long as k is sufficiently large, which we may assume by artificially increasing it at the start). ■

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. Interestingly, we do not know how to remove the condition that $\alpha^2 > \beta$ in Lemma 3.1.12. Perhaps this constraint is inherent for any transformation like ours, in which the new distributions are obtained by concatenating random samples taken obliviously from the original distributions.

Open Problem 3.1.18 *Is there a Polarization Lemma for arbitrary constant thresholds $0 \leq \beta < \alpha \leq 1$? Or even for any specific thresholds such that $\alpha^2 \leq \beta$ (e.g., $\alpha = 5/9$, $\beta = 4/9$)? Is the constraint $\alpha^2 > \beta$ inherent for a wide class of transformations?*

3.1.3 Statistical Difference is in HVSZK

With the Polarization Lemma, it is easy to give a statistical zero-knowledge proof for SD. The proof system is given in Protocol 3.1.19 and the simulator in Algorithm 3.1.20.

Protocol 3.1.19: Statistical zero-knowledge proof (P, V) for SD

Input: Circuits X_0 and X_1, and security parameter 1^k

1. P, V: Both parties compute $(Y_0, Y_1) = \text{Polarize}_{2/3,1/3}(X_0, X_1, 1^{k-1})$.
2. P, V: Both parties execute Protocol 3.1.4 on common input (Y_0, Y_1). V accepts or rejects as in that protocol.

Algorithm 3.1.20: Simulator for SD proof system

Input: Circuits X_0 and X_1 (each with m input gates and n output gates), and security parameter 1^k

1. Compute $(Y_0, Y_1) = \text{Polarize}_{2/3,1/3}(X_0, X_1, 1^{k-1})$.
2. Run Algorithm 3.1.6 on input (Y_0, Y_1) and output whatever it outputs.

It follows immediately from Lemma 3.1.8 that the above protocol and simulator yield a statistical zero-knowledge proof for SD.

Theorem 3.1.21 *Protocol 3.1.19 is an honest-verifier statistical zero-knowledge proof for* SD *with completeness error* 2^{-k}*, soundness error* $1/2 + 2^{-k}$*, and simulator deviation* 2^{-k}*. In particular,* $\text{SD} \in \mathbf{HVSZK}$.

From Lemma 3.1.12, we see this protocol and theorem can be generalized to place $\text{SD}^{\alpha,\beta}$ in **HVSZK** as long as $\alpha^2 > \beta$.

3.2 Analyzing public-coin HVSZK proofs

To complete the proof that Statistical Difference is complete for **HVSZK**, it remains to show that every problem possessing an honest-verifier statistical zero-knowledge proof reduces to SD. As is typical with completeness theorems, this is the more challenging part of the proof. In this section, we focus on the easier task of showing that every problem with a *public-coin* statistical zero-knowledge proof reduces to SD (actually its complement).

Although this result will be subsumed by the general reduction for all of **HVSZK** given in subsequent sections, it will provide a good motivating warm-up. Both reductions are refinements of the general approach to analyzing statistical zero-knowledge proofs pioneered by Fortnow [For89].

3.2.1 The Fortnow methodology

In both classical and interactive proofs, the *verifier* is what distinguishes between YES and NO instances; on YES instances, there is a proof (or prover strategy) that makes the verifier accept, whereas on NO instances there is not. The crucial observation of Fortnow [For89] was that in zero-knowledge proofs, the *simulator* also provides information that can distinguish between YES and NO instances. Specifically, he showed that, for statistical zero-knowledge proofs, YES and NO instances can be (almost) completely distinguished based on statistical properties of the simulator's output distribution. By then showing that these statistical properties can be decided in low complexity, he was able to give a strong upper bound on the complexity of statistical zero knowledge, namely **HVSZK** $\subset$ **co-AM**.[2] Aiello and Håstad [AH91] subsequently used the same general approach to show that **HVSZK** $\subset$ **AM**. These are fairly strong upper bounds on the complexity of **HVSZK**, as they imply that **HVSZK** cannot contain **NP**-hard problems unless the Polynomial Time Hierarchy[3] collapses [BHZ87]. However, in this thesis, we will not be satisfied with upper bounds on the complexity of statistical zero knowledge. Rather, we seek a tight characterization of statistical zero knowledge, in the form of complete problems. In order to obtain such characterizations, we will refine Fortnow's methodology, and thus we begin by describing his approach at an intuitive level.

Recall that Fortnow's aim was to find properties of the simulator's output distribution that distinguish between YES and NO instances of the promise problem whose statistical zero-knowledge proof we are considering. Using terminology taken from [AH91], we think of the simulator as describing an "interaction" between a *virtual prover* and a *virtual verifier*. For YES instances, the definition of statistical zero knowledge gives very strong guarantees on the output distribution of the simulator. Namely, the simulator's output must be very close to the interaction between the real prover and verifier. In particular, the following two conditions must hold.

Conditions for YES instances. Both of the following must hold:

1. The simulator outputs accepting conversations (*i.e.*, ones in which the virtual verifier accepts) with high probability.

2. The virtual verifier "behaves like" the real verifier.

[2] Actually, there was an error in Fortnow's proof, pointed out in [GOP98], but his general approach was sound and influenced many later works. Aiello and Håstad [AH91] gave a correct proof of the result (and we will see another one in this thesis).

[3] See any standard textbook on complexity theory (e.g., [Sip97, Pap94]) for a definition of the Polynomial Time Hierarchy, which is widely conjectured to be infinite.

For NO instances, however, the definition of zero knowledge does not explicitly give any guarantees on the simulator's behavior. Despite this, one can prove something about the simulator's behavior in this case. Specifically, the above two conditions cannot simultaneously hold for NO instances. Suppose both conditions did hold for a NO instance. The first condition says the virtual prover is convincing the virtual verifier to accept with high probability. The second condition then implies that if we allow the virtual prover to interact with the *real* verifier instead of the virtual verifier, it should not change things significantly. Therefore, the virtual prover will convince the real verifier to accept with high probability. But this cannot happen for a NO instance, by the soundness of the proof system. Actually, since there is large gap between the verifier's acceptance probability on YES instances and NO instances, we obtain the following "strong complement" to the conditions for YES instances:

Conditions for NO instances. At least one of the following must hold:

1. The simulator outputs accepting conversations with low probability.
2. The virtual verifier "behaves very differently" from the real verifier.

Now, to distinguish between YES and NO instances, one need only show how to separate the conditions for YES instances from the conditions for NO instances. In [For89, AH91], it was shown that short interactive proofs can separate the two cases, and thereby **HVSZK** was placed in **AM** $\cap$ **co-AM**. Here, we will show that the conditions can be embedded into instances of STATISTICAL DIFFERENCE, and this will show that every problem in **HVSZK** reduces to SD.

There are a number of aspects of the above intuition that are nontrivial to formalize or quantify. First, one must make precise this idea of allowing the virtual prover to interact with the real verifier. Fortnow gave a natural solution to this, by introducing the notion of a *simulation-based prover* P_S, which is a (real) prover strategy that determines its messages according to the same distribution as the virtual prover, when conditioned on past messages. Thus, the interaction (P_S, V) exactly captures the idea of the virtual prover interacting with the real verifier.

A second important challenge is to quantify what it means for the virtual verifier to "behave like" the real verifier. This is the crucial point which determines the tightness of the characterization obtained at the end, for the other condition is easily determined (by running the simulator many times to estimate the probability of an accepting conversation).

To summarize, the main steps in analyzing the simulator of a statistical zero knowledge proof are the following:

1. Quantify what it means for the virtual verifier to "behave like" the real verifier.
2. Confirm that in the case of a YES instance, the virtual verifier does indeed behave like the real verifier according to the chosen quantification.
3. Show that if the virtual verifier behaves like the real verifier, then the interaction between the simulation-based prover and the real verifier is "close" to the output distribution of the simulator.

4. Conclude that if the virtual verifier behaves like the real verifier on a NO instance, then the simulator must output accepting conversations with low probability.

In the next section, we will carry out this approach for *public-coin* statistical zero-knowledge proofs. In this case, it is particularly easy to quantify what it means for the virtual verifier to "behave like" the real verifier, because all the real verifier's behavior consists of is sending uniformly distributed strings that are independent of the conversation history. Thus, the virtual verifier behaves like the real verifier if and only if the virtual verifier's messages are nearly uniform and nearly independent of the conversation history. We will show how to capture this condition by the statistical difference between samplable distributions, thereby obtaining a reduction to STATISTICAL DIFFERENCE.

3.2.2 Simulator analysis

Notation and conventions. Let (P, V) be an interactive proof system for a promise problem Π, and let S be a simulator for (P, V). (At this point, S is an arbitrary algorithm, since we have not yet specified the quality of the simulation.) Throughout this section and Section 3.2.3, we will fix the security parameter $k = |x|$ and omit it from the notation. When we apply our simulator analysis, we will only require *weak* statistical zero knowledge and *constant* completeness and soundness errors, settings in which the security parameter is irrelevant. Since the proof system is polynomially bounded, there is a polynomial $v(\cdot)$ such that $v(|x|)$ bounds the total number of messages sent from the verifier to the prover on input x (not including the verifier's final `accept`/`reject` message). By convention (see Definition 2.3.2), the prover's messages are those with odd index, and the verifier's messages are those with even index. We are interested in the random variables $S(x)$ and $\langle P, V\rangle(x)$, describing the simulation and (the verifier's view of) the real interaction, respectively. We also consider prefixes of these random variables, where $S(x)_i$ and $\langle P, V\rangle(x)_i$ denote the prefixes consisting of the first i messages exchanged. At times, we may drop x from these notations.

For $j \leq 2v(|x|) + 2$, we refer to a tuple of strings $\gamma = (m_1, m_2, \ldots, m_j; r)$ as a *(partial) conversation transcript* if the even-numbered messages in γ (including an `accept`/`reject` message) correspond to what V would have sent given random coins r and the odd-numbered prover messages specified in γ. Without loss of generality, we may assume that the output of the simulator always consists of conversation transcripts that are consistent with V in this sense. This can be achieved by having the simulator, before giving its output, always use the verifier algorithm to recalculate the verifier messages based on the simulated prover messages and the simulated verifier coins. This modification does not affect any of the error parameters or complexity parameters of the proof system. We say that a transcript γ is *accepting* if the verifier accepts on it.

Simulation-based prover. Recall that the simulation-based prover P_S is the prover strategy that "mimics" the virtual prover described by S. More formally, given an input (x) and a conversation history γ (consisting of $2i$ previous messages exchanged), P_S responds as follows:

- If $S(x)$ outputs conversations that begin with γ with probability 0, then P_S replies with a dummy message, say `fail`.

- Otherwise, P_S replies according with the same conditional probability as the prover in the output of the simulator. That is, it replies β with probability

$$p_\beta = \Pr[S(x)_{2i+1} = (\gamma, \beta) | S(x)_{2i} = \gamma]$$

Following our previous notation, we denote the verifier's view of the interaction between P_S and V by $\langle P_S, V\rangle(x)$ and its prefixes by $\langle P_S, V\rangle(x)_i$

Public-coin proofs. For the remainder of this section, we consider only public-coin interactive proofs (P, V). Recall that this means that in every execution of the protocol, the string of random coins accessed by V can be written $r_1 r_2 \cdots r_v \in \{0,1\}^*$, so that the verifier's i'th message m_{2i} equals $r_i \in \{0,1\}^{\ell_i}$, where $\ell_i = \ell_i(x, \gamma)$ is solely a function of the input x and the history $\gamma = (m_1, m_2, \ldots, m_{2i-1})$. Since V runs in polynomial time, ℓ_i is polynomial-time computable from the input and history. Without loss of generality, we may assume that the simulated random coins output by the simulator do not contain any coins other than those that would actually be accessed by the verifier in the interaction; removing these "irrelevant" coins from the output can only decrease the simulator deviation.

The simulator analysis. Now we need to quantify what it means for the virtual verifier to "behave like" the real verifier. As noted in the previous section, for public-coin proofs, this amounts to measuring how close to uniform and independent of history the virtual verifier's messages are. Thus, for $i = 1, \ldots, v(|x|)$, we compare the following two distributions $X_i = X_i(x)$ and $Y_i = Y_i(x)$:

$X_i(x)$**:** Run $S(x)$ to obtain a transcript γ and let γ_{2i} denote the first $2i$ messages exchanged. Output γ_{2i}.

$Y_i(x)$**:** Run $S(x)$ to obtain a transcript γ and let γ_{2i-1} denote the first $2i-1$ messages exchanged. Compute $\ell_i = \ell_i(x, \gamma_{2i-1})$. Choose $r \leftarrow \{0,1\}^{\ell_i}$. Output (γ_{2i-1}, r).

In X_i, the i'th verifier message is computed according to the virtual verifier strategy, and in Y_i, it is chosen uniformly and independently of the history (of the appropriate length). Thus, the statistical difference between these two distributions measures exactly how much the virtual verifier behaves like the real verifier in computing its i'th message. So we define $\delta_i = \delta_i(x)$ by

$$\delta_i \stackrel{\text{def}}{=} \mathrm{StatDiff}\left(X_i, Y_i\right).$$

The following lemma confirms that, when the simulation is good (*e.g.*, for YES instances), the virtual verifier does indeed behave like the real verifier according to this measure.

Lemma 3.2.1 *For every* $i = 1, \ldots, v(|x|)$,

$$\delta_i(x) \le 2 \cdot \mathrm{StatDiff}\left(S(x), \langle P, V\rangle(x)\right).$$

Proof: Dropping x from the notation, we have:

$$\begin{aligned}\delta_i &= \text{StatDiff}\left(X_i, Y_i\right) \\ &\leq \text{StatDiff}\left(X_i, \langle P, V\rangle_{2i}\right) + \text{StatDiff}\left(\langle P, V\rangle_{2i}, Y_i\right)\end{aligned}$$

Note that X_i is the same distribution as S_{2i}, so

$$\text{StatDiff}\left(X_i, \langle P, V\rangle_{2i}\right) = \text{StatDiff}\left(S_{2i}, \langle P, V\rangle_{2i}\right) \leq \text{StatDiff}\left(S, \langle P, V\rangle\right).$$

On the other hand, Y_i is obtained from a sample of S_{2i-1} by applying a certain randomized procedure: namely computing ℓ_i and then concatenating ℓ_i random bits to Y_i. Applying the same randomized procedure to $\langle P, V\rangle_{2i-1}$ yields $\langle P, V\rangle_{2i}$, by the definition of V. Thus, since applying the same randomized procedure to two distributions cannot increase their statistical difference (to be justified after this proof), we have

$$\text{StatDiff}\left(\langle P, V\rangle_{2i}, Y_i\right) \leq \text{StatDiff}\left(\langle P, V\rangle_{2i-1}, S_{2i-1}\right) \leq \text{StatDiff}\left(\langle P, V\rangle, S\right). \qquad \blacksquare$$

The claim that randomized procedures cannot increase statistical difference used in the above proof can be formalized as follows: A *randomized procedure* on a set $\mathcal{U}$ is a probability distribution F on functions from a $\mathcal{U}$ to some set $\mathcal{V}$. The distribution obtained by *applying the randomized procedure* F to a distribution X on $\mathcal{U}$ is defined to be the probability distribution $F(X)$ on $\mathcal{V}$ obtained by independently sampling $f \leftarrow F$ and $x \leftarrow X$ and evaluating $f(x)$. The following fact follows immediately from and Facts 2.2.2 and 3.1.14.

Fact 3.2.2 *For any two probability distributions X and Y on $\mathcal{U}$, and any randomized procedure F on $\mathcal{U}$,* $\text{StatDiff}\left(F(X), F(Y)\right) \leq \text{StatDiff}\left(X, Y\right)$.

The next step in analyzing the simulator is to show that, if the virtual verifier is behaving like the real verifier (*i.e.*, all the δ_i's are small), then the interaction between the simulation-based prover and the real verifier is close to the simulator's output distribution.

Lemma 3.2.3

$$\text{StatDiff}\left(\langle P_S, V\rangle(x), S(x)\right) \leq \sum_{i=0}^{v(|x|)} \delta_i(x)$$

Proof: We will prove by induction on j that for $j = 1, \ldots, v(|x|)$,

$$\text{StatDiff}\left(\langle P_S, V\rangle_{2j}, S_{2j}\right) \leq \sum_{i=0}^{j} \delta_i.$$

The case $j = 0$ is trivial. For general j, note that the definition of the simulation-based prover implies that $\langle P_S, V\rangle_{2j+1}$ is generated by applying the same randomized procedure to $\langle P_S, V\rangle_{2j}$ as the one used to obtain S_{2j+1} from S_{2j}. Thus, by Fact 3.2.2,

$$\text{StatDiff}\left(\langle P_S, V\rangle_{2j+1}, S_{2j+1}\right) = \text{StatDiff}\left(\langle P_S, V\rangle_{2j}, S_{2j}\right). \tag{3.1}$$

Recalling that $X_{j+1} = S_{2j+2}$, we have

$$\begin{aligned}\mathrm{StatDiff}\left(\langle P_S, V\rangle_{2j+2}, S_{2j+2}\right) &= \mathrm{StatDiff}\left(\langle P_S, V\rangle_{2j+2}, X_{j+1}\right)\\ &\leq \mathrm{StatDiff}\left(\langle P_S, V\rangle_{2j+2}, Y_{j+1}\right) + \mathrm{StatDiff}\left(Y_{j+1}, X_{j+1}\right).\end{aligned}$$

Now $\langle P_S, V\rangle_{2j+2}$ is obtained from $\langle P_S, V\rangle_{2j+1}$ via the same randomized procedure used to obtain Y_{j+1} from S_{2j+1}. Thus,

$$\begin{aligned}\mathrm{StatDiff}\left(\langle P_S, V\rangle_{2j+2}, S_{2j+2}\right) &\leq \mathrm{StatDiff}\left(\langle P_S, V\rangle_{2j+1}, S_{2j+1}\right) + \mathrm{StatDiff}\left(Y_{j+1}, X_{j+1}\right).\\ &\leq \mathrm{StatDiff}\left(\langle P_S, V\rangle_{2j}, S_{2j}\right) + \mathrm{StatDiff}\left(Y_{j+1}, X_{j+1}\right).\\ &\leq \left(\sum_{i=0}^{j} \delta_i\right) + \delta_j,\end{aligned}$$

where the last inequality is by induction. This completes the induction.

Taking $j = v$ in Inequality 3.1 almost gives the lemma, except that the transcripts coming from $\langle P_S, V\rangle$ and S contain a few additional strings: the prover messages m_{2v+1} and m_{2v+3}, the verifier's `accept/reject` message m_{2v+2}, and the simulated verifier coins. By our assumptions that the simulator's output is consistent with the verifier algorithm and does not contain any "irrelevant" simulated coins, and the definition of the simulation-based prover, these strings are determined in both $\langle P_S, V\rangle$ and S by applying the same randomized procedure to the history $(m_1, m_2, \ldots, m_{2v})$. Thus, including these components does not increase the statistical difference. ■

The final lemma needed to complete the analysis simply says that if the simulator outputs accepting conversations with high probability in the case of a NO instance, then the simulator's output and the interaction between the simulation-based prover and the real verifier cannot be close.

Lemma 3.2.4 *Let p denote the probability that $S(x)$ outputs an accepting transcript, and let q be the maximum, taken over all provers P^*, that V accepts in $(P^*, V)(x)$. Then,*

$$\mathrm{StatDiff}\left(\langle P_S, V\rangle(x), S(x)\right) \geq p - q.$$

Proof: This follows immediately from the definition of statistical difference — the set of transcripts in which the verifier accepts occurs with probability p in the simulator and with probability at most q in $\langle P_S, V\rangle(x)$. ■

3.2.3 Reducing to STATISTICAL DIFFERENCE

We now use the simulator analysis given above to show that every problem possessing an public-coin statistical zero-knowledge proof reduces $\overline{\mathrm{SD}}$. In fact, the reduction will even work for weak public-coin statistical zero-knowledge proofs.

Theorem 3.2.5 *Every promise problem possessing a weak public-coin honest-verifier statistical zero-knowledge proof reduces to $\overline{\mathrm{SD}}$.*

Proof: Let Π be a promise problem with a weak public-coin honest-verifier statistical zero-knowledge proof (P, V). We maintain the notation and conventions from the Section 3.2.2, in particular fixing $k = |x|$ and dropping it from the notation. We also hide the dependency of the various parameters and distributions on x from the notation throughout this proof. Without loss of generality, we assume that Π has completeness and soundness errors $c = s = 1/3$. Let S be a simulator for (P, V) achieving simulator deviation $\mu \leq 1/\left[4 \cdot (12v)^3\right]$

The reduction should map an input x to a pair of distributions (X, Y), which are statistically close or far, depending on whether x is YES instance or NO instance, respectively. X (resp., Y) will essentially consist of the concatentation of all the X_i's (resp., Y_i's). Lemma 3.2.1 immediately implies that all the X_i's and Y_i's have small statistical difference when x is a YES instance. Lemmas 3.2.3 and 3.2.4 imply that they cannot all have small statistical difference when x is a NO instance *and* the simulator outputs accepting transcripts with too much probability. Thus, we still need to define distributions that will handle the case that the simulator outputs accepting conversations with low probability. Therefore, we define distributions X_0 and Y_0 as follows:

X_0**:** Output 1.

Y_0**:** Run S for $216 \ln 12v$ independent executions, and output 1 if verifier accepts in the majority of the transcripts obtained.

Now we consider the distributions $X' = X_0 \otimes X_1 \otimes \cdots \otimes X_v$ and $Y' = Y_0 \otimes Y_1 \otimes \cdots \otimes Y_v$ (not yet our final distributions).

Claim 3.2.6 *If x is a* YES *instance, then* $\mathrm{StatDiff}\left(X', Y'\right) \leq 1/\left[12 \cdot (12v)^2\right]$

Proof of claim: By Fact 3.1.14, the statistical difference between X' and Y' is at most the sum of the statistical differences between the X_i's and Y_i's. By Lemma 3.2.1,

$$\mathrm{StatDiff}\left(X_i, Y_i\right) \leq 2\mu \leq \frac{1}{2 \cdot (12v)^3}$$

when x is a YES instance.

To bound the difference betwen X_0 and Y_0, observe that, on YES instances x, S must output accepting conversations with probability at least $2/3 - \mu \geq 7/12$. By the Chernoff bound (Theorem A.1), Y_0 outputs 1 with probability at least

$$1 - \exp\left(-2 \cdot (216 \ln 12v) \cdot (1/12)^2\right) \geq 1 - \frac{1}{(12v)^3}.$$

Thus, the statistical difference between X_0 and Y_0 is at most $1/(12v)^3$, and the total statistical difference between X' and Y' is at most

$$v \cdot \frac{1}{2 \cdot (12v)^3} + \frac{1}{(12v)^3} \leq \frac{1}{12 \cdot (12v)^2}.$$

$\square$

Claim 3.2.7 *If x is a* NO *instance, then* $\mathrm{StatDiff}\left(X', Y'\right) \geq 1/12v$.

Proof of claim: It suffices to show that for at least one i, the statistical difference between X_i and Y_i is at least $1/12v$, as the statistical difference between X' and Y' is only greater.

First suppose the simulator outputs accepting conversations with probability at most $5/12$. Then, by the Chernoff bound (Theorem A.1), Y_0 outputs 1 with probability at most

$$\exp\left(-2 \cdot (216 \ln 12v) \cdot (1/12)^2\right) < \frac{1}{2},$$

so the statistical difference between X_0 and Y_0 is at least $1/2 \geq 1/12v$.

Now suppose that the simulator outputs accepting conversations with probability at least $5/12$. By Lemma 3.2.4, this implies that the statistical difference between $\langle P_S, V\rangle$ and S is at least $5/12 - 1/3 \geq 1/12$. Lemma 3.2.3 in turn implies that, for some i, $\delta_i \geq 1/12v$. □

So now let $s = 4 \cdot (12v)^2$, consider $X = \otimes^s X'$, $Y = \otimes^s Y'$. By the above two claims and Lemma 3.1.15, we conclude:

$$\begin{aligned} x \in \Pi_Y \quad &\Rightarrow \quad \mathrm{StatDiff}\,(X,Y) \leq \left(4 \cdot (12v)^2\right) \cdot \frac{1}{12 \cdot (12v)^2} \leq \frac{1}{3} \\ x \in \Pi_N \quad &\Rightarrow \quad \mathrm{StatDiff}\,(X,Y) \geq 1 - \exp\left(-\frac{4 \cdot (12v)^2 \cdot (1/12v)^2}{2}\right) > 2/3 \end{aligned}$$

Thus, X and Y are the desired distributions, and the $x \mapsto (X,Y)$ is a Karp reduction from Π to $\overline{\mathrm{SD}}$. Strictly speaking, the distributions X and Y, which are defined in terms of the simulator need to be encoded by circuits mapping random coins to the output. This can be done by the standard technique of encoding general (*e.g.,* Turing machine) computations as circuits. (See, *e.g.,* the proof of Cook's theorem in [Pap94].) ■

Note that the above proof only requires a simulator with deviation $O(1/v^3)$, where v is the number of messages sent from the verifier to the prover in the proof system. Hence, for constant-message proof systems, the reduction even works when the simulator deviation is a (sufficiently small) constant!

Theorem 3.2.8 (Thm. 3.2.5, generalized) *There is a constant C such that the following holds. Suppose a promise problem Π possesses a public-coin interactive proof system (P,V) with completeness and soundness errors $1/3$ which exchanges at most $m(n)$ messages on inputs of length n. Suppose further that (P,V) has a simulator that achieves deviation $\mu(n) \leq 1/(C \cdot m(n)^3)$. Then, Π reduces to $\overline{\mathrm{SD}}$. In particular, $\overline{\Pi} \in$ **HVSZK**.*

In addition, we need not assume completeness and soundness errors of $1/3$, because parallel repetitions can be used to the reduce the error of the proof system. Note, however, that ℓ parallel repetitions increases the simulator deviation by a factor of ℓ (though it does not increase the number of messages exchanged). Thus the bound on the simulator deviation required to generalize Theorem 3.2.8 to arbitrary completeness and soundness errors will involve the completeness and soundness errors.

3.3 Analyzing general HVSZK proofs

In this section, we generalize the approach outlined in the previous section to handle general, private-coin proof systems. In doing so, we will actually reduce not to Statistical Difference, but to a different promise problem, called Entropy Difference. The reduction is based on the simulator analysis of Aiello and Håstad [AH91].

In Section 3.3.1, we introduce the promise problem Entropy Difference, and also mention some basic notions from information theory that we will use. Section 3.3.2 contains the Aiello–Håstad simulator analysis, formulated in terms of entropy, following Petrank and Tardos [PT96]. In Section 3.3.3, we use this simulator analysis to prove that every problem in **HVSZK** reduces to Entropy Difference.

3.3.1 Entropy Difference

We recall Shannon's notion of *entropy*.

Definition 3.3.1 (entropy) *If X is a discrete probability distribution, then the* entropy *of X, denoted* $\mathrm{H}(X)$, *is defined as*

$$\mathrm{H}(X) \stackrel{\text{def}}{=} \sum_x \Pr[X = x] \cdot \log \frac{1}{\Pr[X = x]} = \mathop{\mathrm{E}}_{x \leftarrow X}\left[\log \frac{1}{\Pr[X = x]}\right].$$

The binary entropy function $\mathrm{H}_2 : [0,1] \to [0,1]$ *is defined to be the entropy of a 0–1 random variable with expectation p, i.e.,*

$$\mathrm{H}_2(p) \stackrel{\text{def}}{=} p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$$

The entropy of a distribution is a measure of how many "bits of randomness" the distribution contains. Some basic facts about entropy that illustrate its naturalness as a measure of randomness are given below. (Proofs can be found in any standard text on information theory, such as [CT91].)

Fact 3.3.2 *For any distribution X (or joint distribution (X,Y)) on a universe $\mathcal{U}$,*

1. $\mathrm{H}(X) \geq 0$, *with equality iff X is constant.*
2. $\mathrm{H}(X) \leq \log |\mathcal{U}|$, *with equality iff X is uniform on $\mathcal{U}$.*
3. *For any function f,* $\mathrm{H}(f(X)) \leq \mathrm{H}(X)$.
4. $\mathrm{H}(X,Y) \leq \mathrm{H}(X) + \mathrm{H}(Y)$, *with equality iff X and Y are independent.*

The second problem we will prove to be complete for **HVSZK** is essentially the problem of determining which of two given samplable distributions has significantly higher entropy.

Definition 3.3.3 Entropy Difference *is the promise problem* ED $= (\mathrm{ED}_Y, \mathrm{ED}_N)$, *where*

$$\begin{aligned} \mathrm{ED}_Y &= \{(X,Y) : \mathrm{H}(X) \geq \mathrm{H}(Y)+1\} \\ \mathrm{ED}_N &= \{(X,Y) : \mathrm{H}(Y) \geq \mathrm{H}(X)+1\}. \end{aligned}$$

Above, X and Y are circuits encoding probability distributions, as in Definition 3.1.1.

Requiring a gap of 1 bit of entropy in the definition of ED is inessential, as any noticeable gap can be easily amplified by replacing each distribution with many independent copies of itself. This contrasts with the definition SD, in which the thresholds of 2/3 and 1/3 are not arbitrary (*cf.,* Open Problem 3.1.18).

In the subsequent sections, we will show that every problem possessing a (private-coin) **HVSZK** proof reduces to Entropy Difference. In doing so, we will make use of a more sophisticated (albeit less intuitive) measure of distance between probability distributions than statistical difference.

Definition 3.3.4 *Let X and Y be two discrete probability distributions. The* relative entropy *(or* Kullback–Leibler distance*) between X and Y is defined as*

$$\mathrm{RelEnt}\,(X,Y) \stackrel{\mathrm{def}}{=} \mathop{\mathrm{E}}_{\alpha \leftarrow X}\left[\log \frac{\Pr[X=\alpha]}{\Pr[Y=\alpha]}\right].$$

We also define the binary relative entropy *for $p, q \in [0,1]$ by*

$$\mathrm{RelEnt}_2\,(p,q) \stackrel{\mathrm{def}}{=} p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}.$$

Note that if X and Y are 0–1 random variables with expections p and q respectively, then $\mathrm{RelEnt}\,(X,Y) = \mathrm{RelEnt}_2\,(p,q)$.

Although $\mathrm{RelEnt}\,(\cdot,\cdot)$ is not symmetric and does not satisfy the triangle inequality, it is useful to think of it as a distance between probability distributions. It does have some of the other properties we would expect such a distance measure to have.

Fact 3.3.5 *For any two distributions X and Y,*

1. $\mathrm{RelEnt}\,(X,Y) \geq 0$, *with equality iff X and Y are identically distributed.*
2. *For any function f,* $\mathrm{RelEnt}\,(f(X), f(Y)) \leq \mathrm{RelEnt}\,(X,Y)$.
3. *For any $0 \leq q' \leq q \leq p \leq p' \leq 1$,* $\mathrm{RelEnt}_2\,(p',q') \geq \mathrm{RelEnt}_2\,(p,q)$.

Proofs for these facts can be found in any standard text on information theory, such as [CT91]. Item 2 is equivalent to the Log Sum Inequality [CT91, Thm. 2.7.1], and Item 3 follows from the convexity of $\mathrm{RelEnt}\,(\cdot,\cdot)$ [CT91, Thm. 2.7.2].

One other notion from information theory that will prove useful to us is that of conditional entropy, which, for a joint distribution (X,Y), measures how much randomness is left in X after Y is revealed.

Definition 3.3.6 (conditional entropy) *If* (X, Y) *is a joint probability distribution, then the* conditional entropy *of* X *given* Y*, denoted* $\mathrm{H}(X|Y)$*, is defined as*

$$\mathrm{H}(X|Y) \stackrel{\text{def}}{=} \mathop{\mathrm{E}}_{y \leftarrow Y}\left[\mathrm{H}(X|_{Y=y})\right].$$

Some basic facts about conditional entropy, whose proofs can be found in [CT91], follow.

Fact 3.3.7 *For every joint distribution* (X, Y)*,*

1. $\mathrm{H}(X|Y) \le \mathrm{H}(X)$.
2. $\mathrm{H}(X, Y) = \mathrm{H}(Y) + \mathrm{H}(X|Y)$.

3.3.2 The Aiello–Håstad simulator analysis

In this section, we present the simulator analysis for private-coin statistical zero-knowledge proofs, due to Aiello and Håstad [AH91]. Following, Petrank and Tardos [PT96], we formulate the analysis in terms of entropy and relative entropy, rather than in terms of set sizes as done in [AH91]. This simulator analysis will be used in Section 3.3.3 to show that every problem in **HVSZK** reduces to Entropy Difference.

Notation and conventions. Let (P, V) be an interactive proof system for a promise problem Π and let S be a simulator for (P, V). We follow the notation and conventions given in Section 3.2.2. In particular, $v(|x|)$ is a polynomial bound on the number of messages sent from the verifier to the prover on input x. In addition, we let $t(|x|)$ and $r(|x|)$ be polynomial bounds on the total communication in the proof system (as measured in bits) and the number of random bits accessed by the verifier, respectively. We now modify the proof system so that the verifier sends its random coins to the prover in an additional message just before the end of the protocol. S can be modified to simulate this without increasing the simulator deviation (since S was supposed to simulate the verifier's coins, too), and this does not increase the completeness or soundness errors. The total communication and the number of messages sent from the verifier to prover now increase to $t'(|x|) = t(|x|) + r(|x|)$ and $v'(|x|) = v(|x|) + 1$, respectively. The purpose of this modification is so that we may simultaneously analyze the simulation of the messages exchanged and the simulation of the verifier's random coins, rather treating them separately.

The simulator analysis. Recall that, according to the approach outlined in Section 3.2.1, the first step in analyzing the simulator is to quantify what it means for the virtual verifier to "behave like" the real verifier. In the case of public-coin proofs, it was easy to see that this amounts to measuring how close to uniform and independent of history the virtual verifier's messages are. For private-coin proofs, however, the analogous condition is less obvious. Intuitively, it should somehow capture the requirement that the verifier's messages are distributed almost correctly, given the history, but it is unclear how to quantify this. For a clue, we skip to the second step, and compare the output of the simulator to the interaction $\langle P_S, V\rangle$ between the simulation-based prover (as defined in Section 3.2.2) and the real verifier. The only difference between these distributions is that, in the simulator, the real verifier is replaced with the virtual verifier, so comparing $\langle P_S, V\rangle$ and S is tantamount to comparing the real verifier and virtual verifer. Amazingly, the relative entropy between these distributions can be rewritten *exactly* as an expression just involving entropies of prefixes of the simulator's output.

Lemma 3.3.8 (implicit in [AH91], explicit in [PT96])

$$\mathrm{RelEnt}\left(S(x), \langle P_S, V\rangle(x)\right) = r(|x|) - \sum_{i=1}^{v'(|x|)} \left[\mathrm{H}(S(x)_{2i}) - \mathrm{H}(S(x)_{2i-1})\right]$$

The term $\mathrm{H}(S(x)_{2i}) - \mathrm{H}(S(x)_{2i-1})$ equals the conditional entropy $\mathrm{H}(S(x)_{2i}|S(x)_{2i-1})$. Intuitively, this measures how many bits of randomness the i'th virtual verifier message contributes to the output distribution of the simulator. Since, over the course of the entire interaction, the real verifier exposes all of its r random coins, the sum of these terms should be close to r when the simulation is good. The converse is also plausible. If this sum is close to r, then it means that the virtual verifier's randomness has been fully spread out over its messages. Since we have required that the simulator's output is consistent with the verifier algorithm, this should mean that the virtual verifier is indeed behaving like the real verifier. Therefore, we use the same quantity to compare how much the virtual verifier behaves like the real verifier and to measure the similarity between the distributions $S(x)$ and $\langle P_S, V\rangle(x)$, in contrast to the public-coin case, in which we used different measures for these two purposes and related them via Lemma 3.2.3.

Proof: For a transcript γ, we we let γ_i denote the prefix of γ consisting of the first i messages exchanged. Then, by definition,

$$\begin{aligned}
\mathrm{RelEnt}\left(S, \langle P_S, V\rangle\right) &= \sum_\gamma \Pr\left[S = \gamma\right] \cdot \log \frac{\Pr\left[S = \gamma\right]}{\Pr\left[\langle P_S, V\rangle = \gamma\right]} \\
&= \sum_\gamma \Pr\left[S = \gamma\right] \cdot \log \frac{\prod_{i=1}^{2v} \Pr\left[S_i = \gamma_i | S_{i-1} = \gamma_{i-1}\right]}{\prod_{i=1}^{2v} \Pr\left[\langle P_S, V\rangle_i = \gamma_i | \langle P_S, V\rangle_{i-1} = \gamma_{i-1}\right]} \\
&= \sum_\gamma \Pr\left[S = \gamma\right] \cdot \log \frac{\prod_{j=1}^{v} \Pr\left[S_{2j} = \gamma_{2j} | S_{2j-1} = \gamma_{2j-1}\right]}{\prod_{j=1}^{v} \Pr\left[\langle P_S, V\rangle_{2j} = \gamma_{2j} | \langle P_S, V\rangle_{2j-1} = \gamma_{2j-1}\right]}
\end{aligned}$$

where the last equality is due to the definition of P_S, by which

$$\Pr\left[\langle P_S, V\rangle_{2j-1} = \gamma_{2j-1} | \langle P_S, V\rangle_{2j-2} = \gamma_{2j-2}\right] = \Pr\left[S_{2j-1} = \gamma_{2j-1} | S_{2j-2} = \gamma_{2j-2}\right].$$

A key observation is that, for any transcript γ, the denominator in the above fraction equals the reciprocal of the number of possible outcomes of the verifier coins (i.e., 2^{-r}), since even-indexed messages of $\langle P_S, V\rangle$ are generated by V exactly as in $\langle P, V\rangle$. Multiplying both the numerator and denominator in the above fraction by $\prod_{j=1}^{v} \Pr[S_{2j-1} = \gamma_{2j-1}]$, we obtain

$$\begin{aligned}
\mathrm{RelEnt}(S, \langle P_S, V\rangle) &= \sum_{\gamma} \Pr[S=\gamma] \cdot \log \frac{\prod_{j=1}^{v} \Pr[S_{2j} = \gamma_{2j}]}{2^{-r} \cdot \prod_{j=1}^{v} \Pr[S_{2j-1} = \gamma_{2j-1}]} \\
&= \sum_{j=1}^{v} \sum_{\gamma} \Pr[S=\gamma] \cdot \log \Pr[S_{2j} = \gamma_{2j}] \\
&\quad + r + \sum_{j=1}^{v} \sum_{\gamma} \Pr[S=\gamma] \cdot \log \frac{1}{\Pr[S_{2j-1} = \gamma_{2j-1}]} \\
&= -\sum_{j=1}^{v} \mathrm{H}(S_{2j}) + r + \sum_{j=1}^{v} \mathrm{H}(S_{2j-1})
\end{aligned}$$

The lemma follows. ■

We will now confirm that, when the simulation is good (*e.g.,* for YES instances), the virtual verifier does indeed behave like the real verifier, as measured by the expression in Lemma 3.3.8. In order to do this, we will observe that the expression is zero if $S(x)$ is replaced by $\langle P, V\rangle(x)$. When the simulation is good, it follows that $\mathrm{H}(S(x)_i)$ is approximately equal to $\mathrm{H}(\langle P, V\rangle(x)_i)$, so replacing former by the latter does not affect the value significantly. Since the quality of the simulation is given in terms of statistical difference, we need a bound on entropy difference in terms of statistical difference.

Fact 3.3.9 *For any two random variables, X and Y, ranging over a universe $\mathcal{U}$ it holds that*

$$|\mathrm{H}(X) - \mathrm{H}(Y)| \;\leq\; \log(|\mathcal{U}| - 1) \cdot \delta \;+\; \mathrm{H}_2(\delta)$$

where $\delta \stackrel{\mathrm{def}}{=} \mathrm{StatDiff}(X, Y)$.

This fact can be inferred from Fano's Inequality (cf., [CT91, Thm. 2.11.1]). A more direct proof follows.

Proof: Assume $\delta > 0$ or else the claim is obvious. Consider the description of statistical difference in terms of area (Fact 3.1.9 and Figure 3-1). Let C, X^+, and Y^+ denote the distributions on $\mathcal{U}$ induced by choosing a point uniformly in the common region, X-above region, and Y-above region, respectively.

Think of X (resp., Y) as being generated by flipping a biased coin R which is 1 with probability $1-\delta$, and then outputting a sample of C if $R = 1$ and a sample of X^+ (resp., Y^+) otherwise. Then, by Facts 3.3.2 and 3.3.7,

$$\begin{aligned}
\mathrm{H}(X) &\leq \mathrm{H}(X, R) \\
&= \mathrm{H}(R) + \mathrm{H}(X|R) \\
&= \mathrm{H}_2(\delta) + (1-\delta) \cdot \mathrm{H}(C) + \delta \cdot \mathrm{H}(X^+),
\end{aligned}$$

and

$$\mathrm{H}(Y) \geq \mathrm{H}(Y|R) \geq (1-\delta)\cdot \mathrm{H}(C).$$

Observing that $\Pr[X^+ = x] = 0$ on at least one $x \in \mathcal{U}$, it follows that $\mathrm{H}(X^+) \leq \log(|\mathcal{U}|-1)$, and the fact follows. ■

Remark 3.3.10 The above bound is tight. Let $e \in \mathcal{U}$ and consider X which is identically e, and Y which with probability $1-\delta$ equals e and otherwise is uniform over $\mathcal{U}\setminus\{e\}$. Clearly, $\mathrm{StatDiff}\,(X,Y) = \delta$ and $\mathrm{H}(Y) - \mathrm{H}(X) = \delta\log(|\mathcal{U}|-1) + \mathrm{H}_2(\delta) - 0$.

Thus, we have the following lemma, analogous to Lemma 3.2.1 in the public-coin case.

Lemma 3.3.11 (implicit in [AH91, PT96]) *Let $\delta(x) = \mathrm{StatDiff}\,(S(x), \langle P,V\rangle(x))$. Then*

$$r(|x|) - \sum_{i=1}^{v'(|x|)} [\mathrm{H}(S(x)_{2i}) - \mathrm{H}(S(x)_{2i-1})] \leq 2v'(x)\cdot\left[t'(x)\cdot\delta(x) + \mathrm{H}_2(\delta(x))\right].$$

Proof: Consider a perfect simulator (*i.e.*, of zero deviation), denoted $\overline{S}$, for $\langle P,V\rangle$. Note that the simulator-based-prover with respect to $\overline{S}$ is P itself. Thus, by Lemma 3.3.8,

$$\begin{aligned} r + \sum_{i=1}^{2v'}(-1)^{i+1}\cdot \mathrm{H}(\langle P,V\rangle_i) &= r + \sum_{i=1}^{2v'}(-1)^{i+1}\cdot \mathrm{H}(\overline{S}_i) \\ &= \mathrm{RelEnt}\left(\overline{S}, \langle P,V\rangle\right) \ = \ 0 \end{aligned}$$

Now we have

$$\begin{aligned} r + \sum_{i=1}^{2v'}(-1)^{i+1}\cdot \mathrm{H}(S_i) &\leq r + \sum_{i=1}^{2v'}(-1)^{i+1}\cdot \mathrm{H}(\langle P,V\rangle_i) + \sum_{i=1}^{2v'}|\mathrm{H}(S_i) - \mathrm{H}(\langle P,V\rangle_i)| \\ &= 0 + \sum_{i=1}^{2v'}|\mathrm{H}(S_i) - \mathrm{H}(\langle P,V\rangle_i)| \\ &\leq 2v'\cdot(\delta\cdot t' + \mathrm{H}_2(\delta)), \end{aligned}$$

where the last inequality is by Fact 3.3.9. ■

Finally, we observe that a lemma analogous to Lemma 3.2.4 holds for the relative entropy measure.

Lemma 3.3.12 (implicit in [AH91, PT96]) *Let p denote the probability that $S(x)$ outputs an accepting transcript, let q be the maximum, taken over all provers P^*, that V accepts in $(P^*,V)(x)$, and assume that $p \geq q$. Then,*

$$\mathrm{RelEnt}\,(S(x), \langle P_S, V\rangle(x)) \geq \mathrm{RelEnt}_2\,(p,q).$$

Proof: Define a Boolean function on transcripts by $f(\gamma) = 1$ if γ is accepting and $f(\gamma) = 0$ otherwise. By Fact 3.3.5, Items 2 and 3, we have

$$\mathrm{RelEnt}\,(S, \langle P_S, V\rangle) \geq \mathrm{RelEnt}\,(f(S), f(\langle P_S, V\rangle)) = \mathrm{RelEnt}_2\,(p, q') \geq \mathrm{RelEnt}_2\,(p, q),$$

where $q' \leq q$ equals the probability that $\langle P_S, V\rangle$ is accepting. ■

3.3.3 Reducing to Entropy Difference

In analogy with Section 3.2.3, we now use the simulator analysis of the previous section to reduce every problem in **HVSZK** to Entropy Difference.

Theorem 3.3.13 *Every promise problem possessing a weak public-coin honest-verifier statistical zero-knowledge proof reduces to* ED*.*

Proof: Let Π be a promise problem with a weak honest-verifier statistical zero-knowledge proof (P, V). We maintain the notation and conventions from the Section 3.3.2, in particular fixing $k = |x|$ and dropping it from the notation. We also hide the dependency of the various parameters and distributions on x from the notation throughout this proof. Without loss of generality, we assume that Π has completeness and soundness errors $c = s = 2^{-40}$. Let S be a simulator for (P, V) achieving simulator deviation $\mu \leq \min\{1/v't', \epsilon\}$, where ϵ is a small constant to be determined from the proof.

The reduction should map an input x to a pair of distributions (X, Y) such that X or Y has larger entropy, depending on whether x is YES instance or NO instance, respectively. X is defined as $X = S_2 \otimes S_4 \otimes \cdots \otimes S_{2v'}$, and the Y will be closely related to the distribution $Y_1 = S_1 \otimes S_3 \otimes \cdots \otimes S_{2v'-1}$. Note that $\mathrm{H}(X) = \sum_i \mathrm{H}(S_{2i})$ and $\mathrm{H}(Y_1) = \sum_i \mathrm{H}(S_{2i-1})$. Lemma 3.3.11 implies that, in the case of YES instances, $\mathrm{H}(X) \approx \mathrm{H}(Y_1) + r$. Lemmas 3.3.8 and 3.3.12 imply that $\mathrm{H}(X) \ll \mathrm{H}(Y_1) + r$ for NO instances on which the simulator outputs accepting transcripts with too much probability. To compensate for the r in these expressions, we define Y_2 to be the uniform distribution on $r - 7$ bits. We still need to handle the case that the simulator outputs accepting conversations with low probability. Therefore, we define a distribution Y_3 that we will use to artificially increase the entropy of Y in this case.

Y_3: Run S $8\ln(t'v' + 2)$ times independently. If the verifier rejects in the majority of the transcripts obtained, output $t'v' + 2$ random bits. Otherwise, output the empty string.

We define $Y = Y_1 \otimes Y_2 \otimes Y_3$.

Claim 3.3.14 *If x is a* YES *instance, then* $\mathrm{H}(X) \geq \mathrm{H}(Y) + 1$.

Proof of claim: By Lemma 3.3.11,

$$\mathrm{H}(Y_1) + r - \mathrm{H}(X) \leq 2v' \cdot \left[t' \cdot \mu + \mathrm{H}_2(\mu)\right].$$

Standard Taylor estimates show that $\mathrm{H}_2(\delta) = \delta \cdot \log(1/\delta) + O(\delta)$ for small δ, so we may assume that $\mathrm{H}_2(\mu) \leq \sqrt{\mu}$. Also noting that $t' \geq v'$, we have

$$\mathrm{H}(Y_1) + r - \mathrm{H}(X) \leq 2v' \left[t' \cdot \left(\frac{1}{v't'}\right) + \sqrt{\frac{1}{v't'}}\right] \leq 4.$$

To bound the entropy of Y_3, observe that, on YES instances x, S must output rejecting conversations with probability at most $2^{-40}+\mu \le 1/4$. By the Chernoff bound (Theorem A.1), the probability p that the majority of the conversations sampled from S are rejecting satisfies

$$p \le \exp\left[-2 \cdot (8\ln(t'v'+2)) \cdot (1/4)^2\right] \le \frac{1}{t'v'+2}.$$

Thus,

$$\begin{aligned} \mathrm{H}(Y_3) &\le p \cdot \left(t'v'+2\right) + (1-p)\cdot 0 + \mathrm{H}_2(p) \\ &\le 1+0+1 = 2. \end{aligned}$$

Putting the above together, we have

$$\begin{aligned} \mathrm{H}(Y) &= \mathrm{H}(Y_1) + \mathrm{H}(Y_2) + \mathrm{H}(Y_3) \\ &\le (\mathrm{H}(X) + 4 - r) + (r-7) + 2 \\ &\le \mathrm{H}(X) - 1. \quad \blacksquare \end{aligned}$$

Claim 3.3.15 *If x is a* NO *instance, then* $\mathrm{H}(Y) \ge \mathrm{H}(X) + 1$.

Proof of claim: It suffices to show that either $\mathrm{H}(Y_1) + \mathrm{H}(Y_2) \ge \mathrm{H}(X) + 1$ or $\mathrm{H}(Y_3) \ge \mathrm{H}(X) + 1$. First, suppose the simulator outputs accepting conversations with probability at most $1/4$. By the Chernoff bound (Theorem A.1), the probability p that the majority of the conversations independently sampled from S are accepting is

$$p \le \exp\left[-2 \cdot (8\ln(t'v'+2)) \cdot (1/4)^2\right] \le \frac{1}{t'v'+2}.$$

Thus,

$$\mathrm{H}(Y_3) \ge (1-p)\cdot(t'v'+2) \ge t'v'+1 \ge \mathrm{H}(X)+1,$$

where the last inequality is because X outputs at most $t'v'$ bits.

Now, suppose that the simulator outputs accepting conversations with probability at least $1/4$. By Lemma 3.2.4, the relative entropy between S and $\langle P_S, V\rangle$ is at least $\mathrm{RelEnt}_2\left(1/4, 2^{-40}\right) > 8$. By Lemma 3.3.8, this implies that $r - \mathrm{H}(X) + H(Y_1) \ge 8$, and therefore

$$\mathrm{H}(Y_1) + \mathrm{H}(Y_2) \ge (r-7) + (8 + \mathrm{H}(X) - r) = \mathrm{H}(X) + 1.$$

$\square$

These claims show that the map $x \mapsto (X, Y)$ is a Karp reduction from Π to ED. $\blacksquare$

Note that the above proof only requires a simulator with deviation $\mu = O(1/t'v') = O(1/[(t+r)\cdot v])$, where t is a bound on the total communication, r is the number of random coins used by the verifier, and v is the number of messages sent from the verifier to the prover.

Theorem 3.3.16 (Thm. 3.3.13, generalized) *There is a constant C such that the following holds. Suppose a promise problem Π possesses an interactive proof system (P, V) with completeness and soundness errors* $1/3$*, in which the number of messages exchanged is $m(n)$, the total communication is $t(n)$, and the verifier uses $r(n)$ random coins on inputs of length n. Suppose further that (P, V) has a simulator that achieves deviation $\mu(n) \leq 1/\left[C \cdot m(n) \cdot (t(n) + r(n))\right]$. Then, Π reduces to* ED.

As with Theorem 3.2.8, this result also applies to proof systems with completeness and soundness errors other than $1/3$, as the error can be reduced using parallel repetitions. (Indeed, this is why we may we state Theorem 3.3.16 for error $1/3$, when we assumed error 2^{-40} in the proof.) Note that the parallel repetitions increase t and r in addition to μ.

3.4 Entropy Difference reduces to Statistical Difference

In this section, we complete the circle of reductions, by showing that Entropy Difference reduces to Statistical Difference. It will then follow that both problems are complete for **HVSZK**. The main technical tool in the reduction is 2-universal hash functions, so we begin by describing those in Section 3.4.1. Then, in Section 3.4.2, we explain the main ideas in the reduction, by treating the special case of "flat" distributions, which are distributions which are uniform over some subset of their range. In Section 3.4.3, we formalize the notion of a "nearly flat" distribution and present some standard techniques for "flattening" distributions. Finally, we combine all these ideas to give the general reduction in Section 3.4.4.

3.4.1 Universal hashing

Universal hash functions, introduced by Carter and Wegman [CW79], are families of functions whose values are pairwise independent. They have a wide variety of applications in computer science, and we will use them many times throughout this thesis.

Definition 3.4.1 (universal hash functions [CW79]) *A family $\mathcal{H}$ of functions mapping a domain $\mathcal{D}$ to a range $\mathcal{R}$ is* 2-universal *if for every $x \neq y \in \mathcal{D}$ and $a, b \in \mathcal{R}$,*

$$\Pr_{h \leftarrow \mathcal{H}}\left[h(x) = a \ \& \ h(y) = b\right] = \frac{1}{|R|^2}.$$

There exist very efficient families of 2-universal hash functions. For example, if we identify the set $\{0,1\}$ with GF(2), the set of affine-linear functions $\mathcal{H}_{m,n}$ from $\mathrm{GF}(2)^m$ to $\mathrm{GF}(2)^n$ is a 2-universal family of hash functions from $\{0,1\}^m$ to $\{0,1\}^n$. Every function h in this family can be uniquely written in the form $h(x) = Ax + b$, where A is an $n \times m$ matrix over GF(2) and b is a vector in $\mathrm{GF}(2)^n$. Throughout this thesis, we write $\mathcal{H}_{m,n}$ for this particular family of 2-universal hash functions (with this represenation).

3.4.2 A special case — flat distributions

In order to motivate our reduction from Entropy Difference to Statistical Difference, we first limit ourselves to a simpler class of distributions. A distribution X is called *flat* if all elements in the support of X have the same probability mass. That is, X is the uniform distribution on $\mathrm{Supp}(X)$. The simplifying assumptions we make is that we are given an instance (X, Y) of ED such that

1. X and Y are both flat.
2. $|\mathrm{H}(X) - \mathrm{H}(Y)| \geq k$, where k is "the security parameter".

Now we want to construct from (X, Y) a new pair of distributions (A, B) such that if $\mathrm{H}(X) \geq \mathrm{H}(Y) + k$, then A and B are statistically far apart, and if $\mathrm{H}(Y) \geq \mathrm{H}(X) + k$, then A and B are statistically close. Let S_X and S_Y be the supports of X and Y, respectively. By the definition of entropy, $|S_X| = 2^{\mathrm{H}(X)}$ and $|S_Y| = 2^{\mathrm{H}(Y)}$, so the condition $\mathrm{H}(X) \gg \mathrm{H}(Y)$ is equivalent to the condition $|S_X| \gg |S_Y|$ and similarly for $\mathrm{H}(Y) \gg \mathrm{H}(X)$.

The following special case of the "Leftover Hash Lemma" shows how to convert flat distributions with high entropy into uniform ones.

Lemma 3.4.2 (Leftover Hash Lemma for flat distributions [ILL89]) *Let $\mathcal{H}$ be a 2-universal family of hash functions mapping a domain $\mathcal{D}$ to a range $\mathcal{R}$. Let Z be a flat distribution on $\mathcal{D}$ such that $|\mathcal{R}| \leq \varepsilon \cdot 2^{\mathrm{H}(Z)}$ Then, the following distribution has statistical difference at most $\varepsilon^{\Omega(1)}$ from the uniform distribution on $\mathcal{H} \times \mathcal{R}$.*

- *Choose $h \leftarrow \mathcal{H}$ and $x \leftarrow Z$. Output $(h, h(x))$.*

It is also easy to see that the same process gives a distribution that is far from uniform if Z has small entropy: For any h, the number of values $h(x)$ can take on is at most $|\mathrm{Supp}(Z)| = 2^{\mathrm{H}(Z)}$, so if this is much smaller than $|\mathcal{R}|$, $(h, h(x))$ will be very far from uniform on $\mathcal{H} \times \mathcal{R}$.

Lemma 3.4.3 *Let $\mathcal{H}$ be any family of functions mapping a domain $\mathcal{D}$ to a range $\mathcal{R}$. Let Z be a flat distribution on $\mathcal{D}$ such that $2^{\mathrm{H}(Z)} \leq \varepsilon \cdot |\mathcal{R}|$. Then, the following distribution has statistical difference at least $1 - \varepsilon$ from the uniform distribution on $\mathcal{H} \times \mathcal{R}$.*

- *Choose $h \leftarrow \mathcal{H}$ and $x \leftarrow Z$. Output $(h, h(x))$.*

These two lemmas seem to be a step in the right direction, because they convert a condition about entropy into a condition about statistical difference: distributions with large entropy are transformed into ones having small statistical difference from uniform, whereas distributions with small entropy are transformed into ones with large statistical difference from uniform. So, one approach would be to take $Z = Y$ and choose $\mathcal{R}$ such that $|\mathcal{R}| = 2^{|\mathrm{H}(X)|}$. Then, the distribution described in the above lemmas and the uniform distribution on $\mathcal{H} \times R$ will have large or small statistical difference according to whether $\mathrm{H}(X) \gg \mathrm{H}(Y)$ or $\mathrm{H}(Y) \gg \mathrm{H}(X)$, as desired. Unfortunately, constructing a set $\mathcal{R}$ for which $|\mathcal{R}| = 2^{\mathrm{H}(X)}$ requires that we know the entropy of X. If computing (or even approximating) the entropy of a samplable distribution could be done in polynomial time, then Entropy

DIFFERENCE would be in **BPP** and there would be nothing to prove! To overcome this difficulty, we adopt a technique of Okamoto [Oka96] (which he calls "complementary usage of messages").

Recall that we are given a circuit (which we also denote X) which samples from X, and let m denote the length of the input to this circuit. So, for any x, we let $\Omega_X(x) \subset \{0,1\}^m$ denote the set of inputs to the circuit which yield output x. Then, $\Pr[X = x] = 2^{-m} \cdot |\Omega_X(x)|$. Since X is flat, we have

$$|\Omega_X(x)| = 2^m \cdot \Pr[X = x] = \begin{cases} 2^m \cdot 2^{-\mathrm{H}(X)} & \text{if } x \in S_X. \\ 0 & \text{otherwise.} \end{cases}$$

The key observation is that for any $x \in S_X$, $|S_Y \times \Omega_X(x)| = 2^{\mathrm{H}(Y)+m-\mathrm{H}(X)}$. Whether $\mathrm{H}(X) \gg \mathrm{H}(Y)$ or $\mathrm{H}(Y) \gg \mathrm{H}(X)$ now translates to whether $|S_Y \times \Omega_X(x)|$ is $\gg 2^m$ or $\ll 2^m$, where m is a value that we can compute just by looking at the circuit for X! So, instead of hashing Y down to a set of size $2^{\mathrm{H}(X)}$ bits, we will hash the uniform distribution on $S_Y \times \Omega_X(x)$ down to $\{0,1\}^m$ (for some $x \in S_X$). However, we are not explicitly given a sampling algorithm for $\Omega_X(x)$. This can be "simulated" by having each of our new distributions choose $r \in \{0,1\}^m$ and reveal $x = X(r)$. Then, conditioned on x, r is uniformly distributed in $\Omega_X(x)$, as desired. That is, letting $\mathcal{H} = \mathcal{H}_{m+n,m}$, where n is the number of output gates of Y, we define A and B as follows.

A: Choose $r \leftarrow \{0,1\}^m$ and let $x = X(r)$. Choose $h \leftarrow \mathcal{H}$, $y \leftarrow Y$. Output $(x, h, h(r, y))$.

B: Choose $x \leftarrow X$, $h \leftarrow \mathcal{H}$, $z \leftarrow \{0,1\}^m$. Output (x, h, z).

It follows from our discussion above and Lemmas 3.4.2 and 3.4.3 that this reduction is correct, under our assumptions about X and Y. That is, for flat X and Y, we have:

1. If $\mathrm{H}(X) > \mathrm{H}(Y) + k$, then $\mathrm{StatDiff}(A, B) \geq 1 - 2^{-\Omega(k)}$.
2. If $\mathrm{H}(Y) > \mathrm{H}(X) + k$, then $\mathrm{StatDiff}(A, B) \leq 2^{-\Omega(k)}$.

To deal with general instances of ED, we need to remove both of our simplifying assumptions. The assumption that $|\mathrm{H}(X) - \mathrm{H}(Y)| \geq k$ is easy to achieve. If let X' (resp., Y') consist of k independent copies of X (resp., Y), then $\mathrm{H}(X') = k \cdot \mathrm{H}(X)$ and $\mathrm{H}(Y') = k \cdot \mathrm{H}(Y)$. So, the difference in entropies is multiplied by k. The same construction also helps deal with the fact that X and Y are not flat. As we shall see in the next section, taking many independent copies of each distribution yields distributions that are "nearly flat" (in a sense to be made precise later). Our final construction is therefore the same as the construction described above, merely augmented by replacing X and Y with many independent copies of each at the start.

3.4.3 Flattening distributions

As a preliminary step towards treating general instances of ENTROPY DIFFERENCE, we formulate the process of "flattening" distributions (*i.e.*, making them "nearly flat" by taking many independent copies).

Definition 3.4.4 (heavy, light and typical elements) *Let X be a distribution on a universe $\mathcal{U}$, x an element of $\mathcal{U}$, and Δ a positive real number. We say that x is* Δ-heavy *(resp.,* Δ-light*) if* $\Pr[X = x] \geq 2^{\Delta} \cdot 2^{-\mathrm{H}(X)}$ (resp., $\Pr[X = x] \leq 2^{-\Delta} \cdot 2^{-\mathrm{H}(X)}$). *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 3.4.5 (nearly 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 Hoeffding Inequality, we have

Lemma 3.4.6 (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 define the *weight of x* to be $\mathrm{wt}(x) = -\log \Pr[X = x]$. Then $\mathrm{wt}(\cdot)$ maps the support of X to $[0, m]$. For every $x_1, \ldots, x_k$, we have

$$\log \frac{1}{\Pr[\otimes^k X = (x_1, \ldots, x_k)]} = \sum_{i=1}^{k} \mathrm{wt}(x_i).$$

Thus, if we let $X_1, \ldots, X_k$ be independent, identically distributed copies of X, we have:

$$\Pr\left[\otimes^k X \text{ is not } t\Delta\text{-typical}\right] = \Pr\left[\left|\sum_{i=1}^{k} \mathrm{wt}(X_i) - \mathrm{H}(\otimes^k X)\right| \geq t\Delta\right].$$

For every i, $\mathrm{E}[\mathrm{wt}(X_i)] = \mathrm{H}(X)$ and $\mathrm{H}(\otimes^k X) = k \cdot \mathrm{H}(X)$, so we are bounding the probability that the average of k independent, identically distributed random variables taking values in $[0, m]$ deviates from its expectation by $t\Delta/k$. By the Hoeffding Inequality (Theorem A.2), this probability is at most

$$2 \cdot \exp\left(\frac{-2 \cdot k \cdot (t\Delta/k)^2}{m^2}\right).$$

For $\Delta = \sqrt{k} \cdot m$ this bound becomes $2\exp(-2t^2) \leq 2^{-t^2+1}$, establishing the lemma. ■

The key point is that the entropy of $\otimes^k X$ grows linearly with k, whereas its deviation from flatness grows significantly more slowly (*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 3.4.6 holds with $m = \ell$. The other main tool we will use is the following more general form of the Leftover Hash Lemma.:

Lemma 3.4.7 (Leftover Hash Lemma [ILL89]) *Let $\mathcal{H}$ be a 2-universal family of hash functions mapping a domain $\mathcal{D}$ to a range $\mathcal{R}$. Suppose Z is a distribution on $\mathcal{D}$ such that with probability at least $1 - \delta$ over z selected from Z,* $\Pr[Z = z] \leq \varepsilon/|\mathcal{R}|$. *Then the following distribution has statistical difference at most $O(\delta + \varepsilon^{1/3})$ from the uniform distribution on $\mathcal{H} \times \mathcal{R}$.*

- *Choose* $h \leftarrow \mathcal{H}$ *and* $z \leftarrow Z$*. Output* $(h, h(z))$.

In particular, notice that if Z is a Δ-flat distribution, then for any parameters $s, t > 0$, Z satisfies the hypothesis of the Leftover Hash Lemma with $|\mathcal{R}| = 2^{\mathrm{H}(X) - t\Delta - s}$, $\delta = 2^{-t^2+1}$, and $\varepsilon = 2^{-s}$.

3.4.4 The general reduction

Now, we combine the ideas of Section 3.4.2 with the tools in Section 3.4.3 to prove our desired result.

Theorem 3.4.8 ENTROPY DIFFERENCE *reduces to* STATISTICAL DIFFERENCE.

Proof: Given an instance (X, Y) of ENTROPY DIFFERENCE, we describe how to efficiently produce an instance (A, B) of STATISTICAL DIFFERENCE such that the latter is a YES or NO instance according to whether the former is. By artificially adding gates if necessary, we may assume that both X and Y have m input gates and n output gates. Let k be a large constant (to be determined from the proof). Set $q = 9km^2$ and define $X' = \otimes^q X$, $Y' = \otimes^q Y$. X' and Y' have input (resp., output) length $m' = qm$ (resp., $n' = qn$). Let $\mathcal{H} = \mathcal{H}_{m'+n', m'}$. The distributions A and B are defined just as in Section 3.4.2, except that we use X' and Y' instead of X and Y:

A: Choose $r \leftarrow \{0,1\}^{m'}$ and let $x = X'(r)$. Choose $h \leftarrow \mathcal{H}$ and $y \leftarrow Y'$. Output $(x, h, h(r, y))$.

B: Choose $x \leftarrow X'$, $h \leftarrow \mathcal{H}$, and $z \leftarrow \{0,1\}^{m'}$. Output (x, h, z).

Now we analyze this construction. We denote the components of the distributions by $A = (A_1, A_2, A_3)$ and $B = (B_1, B_2, B_3)$. By Lemma 3.4.6, X' and Y' are Δ-flat for $\Delta = \sqrt{9km^2} \cdot m = 3\sqrt{k} \cdot m^2$. Noting that $q > 2\sqrt{k}\Delta + k$, we have:

Claim 3.4.9

$$(X, Y) \in \mathrm{ED}_Y \;\Rightarrow\; \mathrm{H}(X') > \mathrm{H}(Y') + 2\sqrt{k}\Delta + k.$$
$$(X, Y) \in \mathrm{ED}_N \;\Rightarrow\; \mathrm{H}(Y') > \mathrm{H}(X') + 2\sqrt{k}\Delta + k.$$

Now we show that A and B are statistically far or close according whether X or Y has larger entropy.

Claim 3.4.10 *If* $(X, Y) \in \mathrm{ED}_Y$*, then* $\mathrm{StatDiff}\,(A, B) \geq 1 - O(2^{-k})$.

> **Proof of claim:** (A_1, A_2) and (B_1, B_2) are both distributed according to $X' \otimes \mathcal{H}$. Thus, to show that A and B are statistically far, it suffices to show that conditioned on most values $(x, h) \leftarrow X' \otimes \mathcal{H}$, the marginal distribution on A_3 and B_3 are statistically far. Since X' is Δ flat, $x \leftarrow X'$ is $\sqrt{k}\Delta$-typical with probability at least $1 - 2^{-k+1}$. Fix any such $\sqrt{k}\Delta$-typical x and fix any $h \in \mathcal{H}$, and we compare the distributions $A_{x,h} = A_3|_{A_1 = x, A_2 = h}$ and $B_{x,h} =$

$B_3|_{B_1=x,B_2=h}$. $B_{x,h}$ is simply the uniform distribution on $\{0,1\}^{m'}$. A_x is the distribution obtained by selecting $(r,y) \leftarrow \Omega_{X'}(x) \otimes Y'$ and outputting $h(r,y)$. Since Y' is Δ-flat, $y \leftarrow Y'$ is $\sqrt{k}\Delta$-typical with probability at least $1 - 2^{-k+1}$. Let

$$T_{x,h} = \left\{ h(r,y) \ : \ r \in \Omega_{X'}(x) \text{ and } y \text{ is } \sqrt{k}\Delta\text{-typical} \right\}.$$

So, $A_{x,h}$ lies in $T_{x,h}$ with high probability. We will argue that $|T_{x,h}|$ is much smaller than $2^{m'}$. $|T_{x,h}|$ is certainly at most $|\Omega_{X'}(x)|$ times the number of $\sqrt{k}\Delta$-typical y's. $|\Omega_{X'}(x)| \leq 2^{m'-\mathrm{H}(X')+\sqrt{k}\Delta}$, because x is $\sqrt{k}\Delta$-typical. The number of $\sqrt{k}\Delta$-typical y's is at most $2^{\mathrm{H}(Y')+\sqrt{k}\Delta}$, since they each have mass at least $2^{-\mathrm{H}(Y')-\sqrt{k}\Delta}$. Thus,

$$|T_{x,h}| \leq 2^{m'-\mathrm{H}(X')+\sqrt{k}\Delta} \cdot 2^{\mathrm{H}(Y')+\sqrt{k}\Delta} \leq 2^{m'-k},$$

where the second inequality is by Claim 3.4.9. So,

$$\begin{aligned} \mathrm{StatDiff}\,(A_{x,h}, B_{x,h}) &\geq \Pr[A_{x,h} \in T_{x,h}] - \Pr[B_{x,h} \in T_{x,h}] \\ &\geq \left(1 - \frac{1}{2^{k-1}}\right) + \frac{2^{m'-k}}{2^{m'}} \\ &= 1 - O(2^{-k}). \end{aligned}$$

This holds for any h and any $\sqrt{k}\Delta$-typical x, so to lower-bound the statistical difference between A and B, we should subtract the probability that x is not typical, which is also $O(2^{-k})$. □

Claim 3.4.11 *If* $(X,Y) \in \mathrm{ED}_N$, *then* $\mathrm{StatDiff}\,(A,B) \leq 2^{-\Omega(k)}$.

Proof of claim: As in the proof for YES instances, fix any $\sqrt{k}\Delta$-typical x. We consider the distributions $A_x = (A_2, A_3)|_{A_1=x}$ and $B_x = (B_2, B_3)|_{B_1=x}$. B_x is simply the uniform distribution on $\mathcal{H} \times \{0,1\}^{m'}$. A_x is the distribution obtained by selecting $(r,y) \leftarrow \Omega_{X'}(x) \otimes Y'$, $h \leftarrow \mathcal{H}$ and outputting $(h, h(r,y))$. Since $\Omega_{X'}(x)$ is a flat distribution and Y' is Δ-flat, $\Omega_{X'}(x) \otimes Y'$ is also Δ-flat. The entropy of this distribution can also be bounded by Claim 3.4.9 and the $\sqrt{k}\Delta$-typicality of x as follows.

$$\begin{aligned} \mathrm{H}\left(\Omega_{X'}(x) \otimes Y'\right) &= \log|\Omega_{X'}(x)| + \mathrm{H}(Y') \\ &\geq \left(m' - \mathrm{H}(X') - \sqrt{k}\Delta\right) + (\mathrm{H}(X') + 2\sqrt{k}\Delta + k) \\ &\geq m' + k + \sqrt{k}\Delta. \end{aligned}$$

Thus, taking $\mathcal{R} = \{0,1\}^{m'}$, $\varepsilon = 2^{-k}$, and $\delta = 2^{-k+1}$ in the Leftover Hash Lemma (Lemma 3.4.7), it follows that A_x has statistical difference at most $2^{-\Omega(k)}$ from B_x for any $\sqrt{k}\Delta$-typical x. Since x is $\sqrt{k}\Delta$-typical with probability at least $1 - O(2^k)$, A and B have statistical difference at most $2^{-\Omega(k)}$. □

The theorem follows from Claims 3.4.10 and 3.4.11, taking k to be a sufficiently large constant. ■

3.5 The Completeness Theorem

Putting everything together, we obtain the main theorem of this chapter.

Theorem 3.5.1 (Completeness Theorem) Statistical Difference *and* Entropy Difference *are both complete for* **HVSZK**.

Proof: SD is in **HVSZK** by Theorem 3.1.21. Since ED reduces to SD (Theorem 3.4.8) and **HVSZK** is closed under Karp reductions (Proposition 2.4.1), it follows that ED $\in$ **HVSZK**. Every problem in **HVSZK** reduces to ED by Theorem 3.3.13. Composing these reductions with the reduction from ED to SD (Theorem 3.4.8), it follows that every problem in **HVSZK** reduces to SD. ■

This theorem has a number of immediate consequences. The first is that it gives us a very clear picture of expressiveness of statistical zero-knowledge proofs. Specifically, Theorem 3.5.1 has the following informal interpretation:

> *The assertions that can be proven in statistical zero knowledge are exactly those that can be cast as comparing two samplable distributions, with respect to either their entropies or their statistical difference.*

The term "statistical zero knowledge" coined by Goldwasser, Micali, and Rackoff [GMR89] seems almost prophetic of this characterization of statistical zero knowledge as the class of (approximate) statistical properties.

A second consequence of this theorem is that questions about **HVSZK** can now be translated to questions about these two specific complete problems, and conversely. For example, if we wish to show that every problem in **HVSZK** has a proof system with a certain properties (such as being constant round, public coin, zero knowledge against cheating verifiers, or perfect zero knowledge), we need only exhibit such a proof system for one of the complete problems. Or a question such as whether **HVSZK** is closed under complementation now translates to asking if one of the complete problems reduces to its complement (which is easily seen for Entropy Difference). Indeed, in the remainder of this thesis, these complete problems will be used to prove many new results about **HVSZK** and also obtain much simpler proofs of previously known results.

This correspondence is also fruitful in the reverse direction; that is, from examining **HVSZK**, we obtain new results about efficiently samplable distributions, and how their entropies and statistical differences can be manipulated. Already, we have seen a few examples of this. The XOR Lemma (Lemma 3.1.16), the Polarization Lemma (Lemma 3.1.12), and the reduction from Entropy Difference to Statistical Difference (Theorem 3.4.8) are all results solely about manipulating efficiently samplable distributions, which are of interest independent of their significance for zero-knowledge proofs. Yet, we obtained each of these transformations by extracting ideas from works on statistical zero knowledge. We will see additional examples of this in the next chapter.

Chapter 4

Applications of the Complete Problems

In this chapter, we give a number of applications of the Completeness Theorem. We briefly describe these results by section:

Section 4.1 — Efficient HVSZK proof systems. Using the completeness of STATISTICAL DIFFERENCE, we prove that every problem in **HVSZK** has a very communication-efficient honest-verifier statistical zero-knowledge proof — namely, a two-message proof system with one bit of prover-to-verifier communication (to achieve soundness error 1/2).

Section 4.2 — The complexity of SZK. We deduce some of the important results on the complexity of **HVSZK** as immediate corollaries of the Completeness Theorem. Specifically, Okamoto's result that **HVSZK** is closed under complement [Oka96] and the upper bounds of Fortnow [For89] and Aiello and Håstad [AH91] on the complexity of **HVSZK** all follow immdeiately.

Section 4.3 — Expected polynomial-time simulators. We show how our proof of the Completeness Theorem implies that our definition of **HVSZK** (using strict polynomial-time simulators and a security parameter) is actually equivalent to the weaker GMR definition, and in fact equivalent to **weak-HVSZK**. That is, we show that every problem possessing a **weak-HVSZK** proof system also possesses an **HVSZK** proof system in our sense.

Section 4.4 — Reversing statistical difference. From the Completeness Theorem and the closure of **HVSZK** under complement, we deduce a novel result about manipulating the statistical difference between efficiently samplable distributions. Specifically, we give a polynomial-time computable transformation which maps pairs of distributions that are statistically close (resp., far apart) to pairs that are statistically far apart (resp., close). We also extract a more explicit description of such a "Reversal Mapping" (that does not pass through statistical zero-knowledge proofs).

Section 4.5 — Closure properties. We prove strong Boolean closure properties of **HVSZK** using the complete problem STATISTICAL DIFFERENCE together with our results about manipulating efficiently samplable distributions (the XOR Lemma, Direct Product Lemma, Polarization Lemma, and Reversal Mapping). These closure properties can be interpreted as giving honest-verifier statistical zero-knowledge proofs for complex assertions built out simpler assertions already known to be in **HVSZK** (*e.g.,* proving that at least half of $(x_1, \ldots, x_m)$ are YES instances of some problem in $\Pi \in$ **HVSZK**). Alternatively, these closure properties can be viewed as asserting the closure of **HVSZK** under nonadaptive Cook reductions whose postcomputation is done by a log-depth circuit.

Section 4.6 — Knowledge complexity. We consider the notions of knowledge complexity defined in [GMR89, GP91], which aim to measure the *amount* of knowledge that is leaked in an interactive proof. We show how (statistical) knowledge complexity in the "hint sense" [GP91] can be understood in terms of statistical zero knowledge, and thereby use our results about **HVSZK** to obtain new results about this form of knowledge complexity. In particular, we obtain the first collapse in any of the knowledge complexity hierarchies defined by Goldreich and Petrank [GP91]. In addition, we obtain some tighter bounds on the perfect knowledge complexity of **HVSZK**.

Section 4.7 — Perfect and computational zero knowledge. We apply the simulator analyses of Sections 3.2 and 3.3 to perfect and computational zero-knowledge proofs. We obtain reductions to restricted versions of STATISTICAL DIFFERENCE and ENTROPY DIFFERENCE for **HVPZK**, and nontrivial results for public-coin **HVCZK**, though they do not seem to yield complete problems.

Section 4.8 — Zero-knowledge proofs for hard problems imply one-way functions. Using the completeness of STATISTICAL DIFFERENCE, we obtain a simpler proof of a theorem of Ostrovsky [Ost91], which asserts that if **HVSZK** contains a hard-on-average language, then one-way functions exist. We also consider the generalization of Ostrovsky's result to computational zero knowledge, due to Ostrovsky and Wigderson [OW93]. Using the simulator analysis from Section 4.7, we also obtain a simpler proof of the Ostrovsky–Wigderson theorem in the case of public-coin computational zero-knowledge proofs.

4.1 Efficient HVSZK proof systems

One immediate consequence of the Completeness Theorem is that every problem in **HVSZK** inherits a proof system with the nice properties possessed by the one for STATISTICAL DIFFERENCE (Protocol 3.1.19).

Corollary 4.1.1 *Every problem in* **HVSZK** *has an honest-verifier statistical zero-knowledge proof system with the following properties:*

1. *The proof system exchanges only 2 messages.*

2. *The prover-to-verifier communication is only 1 bit.*

3. *The completeness error and simulator deviation are both* 2^{-k}.

4. *The soundness error is* $1/2 + 2^{-k}$.

5. *The prover is deterministic.*

Okamoto [Oka96] has previously shown that every problem in **HVSZK** has a 2-message **HVSZK** proof, but the other properties listed in Corollary 4.1.1 are new.

The soundness error above can actually be reduced to exactly 1/2 using a simple trick [Gol99]. Specifically, set $p = 1/(1 + 2^{-k+1})$ and modify the proof system as follows. At the start the verifier automatically rejects with probability $1 - p$, and otherwise proceeds as in the original proof system. The soundness error becomes $p \cdot (1/2 + 2^{-k}) = 1/2$, the completeness error and simulator deviation become at most $1 - p + 2^{-k} = O(2^{-k})$, and the other properties listed remain the same.

It is easy to see that soundness error $\approx 1/2$ is the best achievable in nontrivial proof systems where the prover sends one bit and the completeness error is small.

Proposition 4.1.2 *Suppose promise problem* Π *has an interactive proof in which the prover-to-verifier communication is one bit and the completeness error* c *and soundness error* s *are constants satisfying* $1 - c > 2s$. *Then* $\Pi \in$ **BPP**

Proof: The following randomized algorithm decides Π: Simulate the verifier algorithm for both possible prover responses. If either response makes the verifier accept, then accept.

On YES instances, this algorithm will accept with probability at least $1 - c$, since completeness tells us that there is a good response with at least that probability. On NO instances, this algorithm will accept with probability at most $2s$, for otherwise a prover strategy that chooses its response uniformly at random will make the verifier accept with probability greater than s. Since $1 - c > 2s$ and both quantities are constants, the error probability of this algorithm can be reduced via the usual method. ∎

4.2 The complexity of SZK

From the complete problems, we obtain as immediate corollaries some of the most important results known about the complexity of **HVSZK**. First note that the complete problem ENTROPY DIFFERENCE has a trivial reduction to its complement — the map $(X, Y) \mapsto (Y, X)$. From this (and the closure of **HVSZK** under reductions), we obtain a trivial proof of Okamoto's result that **HVSZK** is closed under complement.

Corollary 4.2.1 ([Oka96]) **HVSZK** *is closed under complement.*

From the efficient proof systems given by Corollary 4.1.1 and closure under complement, the main results of Fortnow [For89] and Aiello and Håstad [AH91] immediately follow.

Corollary 4.2.2 ([For89, AH91]) **HVSZK** $\subset$ **AM** $\cap$ **co-AM**.

This is a strong upper bound on the complexity of **HVSZK**, as demonstrated by the following result of Boppana, Håstad, and Zachos [BHZ87].

Proposition 4.2.3 ([BHZ87]) *If* **NP** $\subset$ **co-AM**, *then the Polynomial Hierarchy (***PH***) collapses.*

From these two results, it immediately follows that neither **NP** nor **co-NP** are contained in **HVSZK** unless the **PH** collapses. Moreover, **HVSZK** cannot contain any problem that is **NP**-hard under any type of reduction that **AM** $\cap$ **co-AM** is closed under. As noted in [ESY84, GG98a], some care must be taken here, since we are dealing with classes of promise problems. As a class of promise problems, **AM** $\cap$ **co-AM** is actually unlikely to be closed under the most general form of Cook reductions. It is, however, closed under Cook reductions which are either *nonadaptive* (*i.e.,* the oracle queries are made all at once, prior to receiving any answers) or *smart* (*i.e.,* the queries do not violate the promise) [ESY84, GG98a].[1]

The completeness of Statistical Difference also illustrates a closer connection between **HVSZK** and **BPP** than might be evident from their definitions.

Proposition 4.2.4 *Let* 1-SD *be the promise problem obtained by restricting the circuits in the definition of* SD *to have only one bit of output. Then* 1-SD *is complete for* **BPP**.

Proof: To see that 1-SD is in **BPP**, first observe that for any distributions X and Y on $\{0,1\}$,

$$\mathrm{StatDiff}(X,Y) = |\Pr[X=1] - \Pr[Y=1]|.$$

Thus, an estimate on $\mathrm{StatDiff}(X,Y)$ that is correct within an additive factor of, say, $1/6$, can be obtained by sampling X and Y a constant number of times and counting the number of ones that occur. This is sufficient to decide 1-SD.

Now we show that every promise problem $\Pi \in$ **BPP** reduces to 1-SD. Let A be the probabilistic polynomial time machine that decides Π with two-sided error at most $1/3$. Given an input x, it is possible to compute in polynomial time a circuit X_x describing the computation of A on input x (see, *e.g.,* the proof of Cook's theorem in [Pap94]). The inputs to X_x are the random bits used by A's computation on x and the output is 1 (resp., 0) if A accepts (resp., rejects). Let Y be a circuit which always outputs 0. Then $\mathrm{StatDiff}(X_x, Y) = \Pr[A(x) \text{ accepts}]$, so $x \mapsto (X_x, Y)$ is a reduction from Π to 1-SD. ■

Proposition 4.2.4 remains true even if we allow the circuits to have output length logarithmic in their size. Analogously restricting $\mathrm{SD}^{1,1/2}$ to have one output bit yields a complete problem for **co-RP**, and restricting the *input* length of SD to be one bit or logarithmically many bits yields a complete problem for **P** (under logarithmic-space reductions).

4.3 Expected polynomial-time simulators

Recall that our definition of statistical zero knowledge differs from the definition of Goldwasser, Micali, and Rackoff [GMR89] in several ways. The GMR definition is a weaker

[1] It should be noted that these problems disappear if one considers only *languages*. When restricted to languages, **AM** $\cap$ **co-AM** is closed under general Cook reductions, and no language in **AM** $\cap$ **co-AM** can be **NP**-hard with respect to even more general forms of reducibility (unless **PH** collapses) [Sch88].

requirement in that it allows *expected* polynomial time simulators whose deviation is a function of the *input length* rather than a separate security parameter. Definition 2.4.2 introduces **weak-HVSZK** as an even weaker notion, in which, for every polynomial $p(\cdot)$, there can be a different simulator to achieve simulator deviation $1/p(|x|)$. From the definitions, **HVSZK** $\subset$ **weak-HVSZK**, and the class satisfying the GMR definition (for honest verifiers) lies between these two classes. Our proof of the completeness theorem actually demonstrates that all three of these classes are equal.

Corollary 4.3.1 weak-HVSZK = HVSZK.

Proof: Theorem 3.3.13 shows that every problem in **weak-HVSZK** reduces to ED. Since ED $\in$ **HVSZK** and **HVSZK** is closed under reductions, **weak-HVSZK** $\subset$ **HVSZK**. ■

From Corollary 4.1.1, this implies that every problem in **weak-HVSZK** in fact has an honest-verifier statistical zero-knowledge proof with *exponentially small* simulator deviation as a function of a security parameter. Thus, the relatively weak inverse-polynomial simulation condition of **weak-HVSZK** can always be bootstrapped into this very strong one.

4.4 Reversing statistical difference

The completeness of Statistical Difference together with the closure of **HVSZK** under complementation imply that SD reduces to $\overline{\text{SD}}$. This is equivalent to the following surprising result about manipulating statistical difference.

Corollary 4.4.1 (Reversal Mapping) *There is a polynomial-time computable function that maps pairs of circuits (X, Y) to pairs of circuits (X', Y') such that*

$$\begin{aligned} \text{StatDiff}(X,Y) \geq 2/3 \quad &\Rightarrow \quad \text{StatDiff}\left(X',Y'\right) \leq 1/3 \\ \text{StatDiff}(X,Y) \leq 1/3 \quad &\Rightarrow \quad \text{StatDiff}\left(X',Y'\right) \geq 2/3 \end{aligned}$$

Although the statement of this result does not involve zero-knowledge proofs, the proof of it given above and (and its original discovery in [SV97]) both involve transformations and analysis of statistical zero-knowledge proofs. In this section, we give a more direct construction of such a mapping, that does not use zero-knowledge proofs (though it is based on ideas extracted from works on statistical zero knowledge [Oka96, SV97, GV99]).

Recall that the other complete problem for **HVSZK**, Entropy Difference, has a trivial reduction to its complement, namely the map $(X, Y) \mapsto (Y, X)$. Also recall that, in Section 3.4, we gave a direct reduction from ED to SD. Thus, to reduce SD to its complement, it suffices to give a direct reduction from SD to ED. That is what we proceed to do.

Let (X_0, X_1) be an instance of SD and consider the following joint distribution $Y = (X, B)$:

$Y = (X, B)$**:** Choose $b \leftarrow \{0, 1\}$. Sample $x \leftarrow X_b$. Output (x, b).

Intuitively, if X_0 and X_1 are statistically very far apart, then b is essentially determined by x, and therefore $\mathrm{H}(Y) \approx \mathrm{H}(X)$. On the other hand, if X_0 and X_1 are statistically very close, then b is essentially independent of x and therefore $\mathrm{H}(Y) \approx \mathrm{H}(X)+1$. Thus, the statistical closeness of X_0 and X_1 is converted into entropy; this same construction was used by Goldreich [Gol90] in the computational setting to convert computational indistinguishability into "false entropy."

We now make this intuition quantitative by estimating $\mathrm{H}(Y)$ as a function of $\mathrm{H}(X)$ and the statistical difference between X_0 and X_1.

Claim 4.4.2 *Let* $\delta = \mathrm{StatDiff}(X_0, X_1)$. *Then* $1-\delta \le \mathrm{H}(Y) - \mathrm{H}(X) \le \mathrm{H}_2((1+\delta)/2)$.

Proof of claim: By Fact 3.3.7, $\mathrm{H}(Y) = \mathrm{H}(X) + \mathrm{H}(B|X)$, so our task is to bound $\mathrm{H}(B|X)$ above and below. For the lower bound, consider the description of statistical difference in terms of area (Fact 3.1.9 and Figure 3-1). Without loss of generality, assume that both X_0 and X_1 have n output gates and thereby define distributions on universe $\mathcal{U} = \{0,1\}^n$. Let C, X_0^+, and X_1^+ denote the distributions on $\mathcal{U}$ induced by choosing a point uniformly in the common region, X_0-above region, and X_1-above region, respectively. We can think of Y as being generated as in the proof of Lemma 3.1.8: A biased coin D is flipped such that D is 0 with probability $1-\delta$. If $D=0$, then X is chosen according to C and B is selected uniformly in $\{0,1\}$. If $D=1$, then B is selected uniformly in $\{0,1\}$ and X is chosen according to X_B^+. Now, conditioned on $D=0$, B is uniform in $\{0,1\}$ and independent of X, so

$$\mathrm{H}(B|X) \ge \mathrm{H}(B|X,D) \ge 1-\delta,$$

which gives the lower bound in the claim.

For the upper bound, note that the distribution of (X,B) is the same as the distribution of (x,b) in Protocol 3.1.4. Let $P(x)$ denote the specified prover's guess for b when given x in the protocol (*i.e.*, $P(x)=0$ iff $\Pr[X_0 = x] > \Pr[X_1 = x]$). In Lemma 3.1.8, we showed that $P(X) = B$ with probability $(1+\delta)/2$. Let E be the indicator for the event that $P(X) = B$. Then,

$$\mathrm{H}(B|X) \le \mathrm{H}(E) + \mathrm{H}(B|X,E) = \mathrm{H}_2\left(\frac{1+\delta}{2}\right) + 0,$$

since B is determined by X and E. □

Plugging in $\delta = 1/3, 2/3$ into this claim, we see:

$$\begin{aligned} (X_0,X_1) \in \mathrm{SD}_Y &\;\Rightarrow\; \mathrm{H}(Y) - \mathrm{H}(X) \le \mathrm{H}_2(5/6) < .651 \\ (X_0,X_1) \in \mathrm{SD}_N &\;\Rightarrow\; \mathrm{H}(Y) - \mathrm{H}(X) \ge 1 - 1/3 > .666 \end{aligned}$$

If we let $X' = \otimes^{200} X \otimes U_{132}$, where U_{132} is the uniform distribution on 132 bits, and $Y' = \otimes^{200} Y$, then

$$\begin{aligned} (X_0,X_1) \in \mathrm{SD}_Y &\;\Rightarrow\; \mathrm{H}(Y') - \mathrm{H}(X') < .651 \cdot 200 - 132 < -1 \\ (X_0,X_1) \in \mathrm{SD}_N &\;\Rightarrow\; \mathrm{H}(Y') - \mathrm{H}(X') > .666 \cdot 200 - 132 > 1. \end{aligned}$$

Thus, $(X_0, X_1) \mapsto (X', Y')$ is a reduction from SD to ED, as desired.

Remark 4.4.3 Clearly, the above technique can be used to reduce $\mathrm{SD}^{\alpha,\beta}$ to ED for any constants α and β such that $\mathrm{H}_2((1+\alpha)/2) < 1-\beta$. After reducing to ED, one can apply the reduction from ED to SD (Theorem 3.4.8) and the Polarization Lemma for SD (Lemma 3.1.12), to obtain a Polarization Lemma for $\mathrm{SD}^{\alpha,\beta}$. Unfortunately, it turns out that $\mathrm{H}_2((1+\alpha)/2) \geq 1-\alpha^2$ for all $\alpha \in [0,1]$, so we must have $\alpha^2 > \beta$ for this to work, in which case Lemma 3.1.12 applies directly. Hence, this does not answer Open Problem 3.1.18.

In addition, both the upper and lower bounds in Claim 4.4.2 are tight. To match the lower bound, consider a universe of three points $\mathcal{U} = \{0,1,2\}$, define X_0 to be 0 with probability δ and 2 otherwise, and define X_1 to be 1 with probability δ and 2 otherwise. Then $\mathrm{StatDiff}(X_0, X_1) = \delta$ and $\mathrm{H}(B|X) = 1-\delta$. To match the upper bound, consider a universe of two points $\mathcal{U} = \{0,1\}$, define X_0 to be 0 with probability $(1+\delta)/2$ and 1 otherwise, and X_1 to be 1 with probability $(1+\delta)/2$ and 0 otherwise. Then $\mathrm{StatDiff}(X_0, X_1) = \delta$ and $\mathrm{H}(B|X) = \mathrm{H}_2((1+\delta)/2)$.

4.5 Closure properties

We have already shown that **HVSZK** has two closure properties. Closure under Karp reductions (Proposition 2.4.1), which is a *computational* closure property, follows immediately from our security-parameter based definition. Closure under complementation (Corollary 4.2.1), which is a *Boolean* closure property, follows from the symmetry of the complete problem ENTROPY DIFFERENCE. In this section, we will prove that **HVSZK** satisfies a stronger closure property that is both computational and Boolean in nature.

In order to motivate the closure property, we first describe the consequence it will have for statistical zero-knowledge proofs. Suppose Π is a problem in **HVSZK** and a prover wishes to convince a verifier not just that a string is a YES instance of Π, but also that some complex expressions built out of Π are true. For example, they might be given m instances $x_1, \ldots, x_m$ of Π, and the prover wishes to convince the verifier that exactly half of these are YES instances. Or more generally, they are given m instances of Π together with an m-ary Boolean formula $\phi(v_1, \ldots, v_m)$, and the prover wants to convince the verifier that ϕ is true when $v_1, \ldots, v_m$ are set to 0 or 1 according to whether x_i is a YES or NO instance of Π. In this section, we demonstrate that such Boolean expressions over Π can be proven in (honest-verifier) statistical zero knowledge as long as $\Pi \in$ **HVSZK**. Moreover, the interaction is polynomially-bounded not just in $|x_1|, \ldots, |x_m|$ and the security parameter, but also in m and $|\phi|$.

In order to deal with instances of promise problems that violate the promise, we will work with an extension of Boolean algebra that includes an additional "ambiguous" value $\star$.

Definition 4.5.1 *A* partial assignment *to variables $v_1, \ldots, v_k$ is a k-tuple $\overline{a} = (a_1, \ldots, a_k) \in \{0,1,\star\}^k$. For a propositional formula (or circuit) ϕ on variables $v_1, \ldots, v_k$, the evaluation*

$\phi(\overline{a})$ is recursively defined as follows:

$$v_i(\overline{a}) = a_i \qquad (\phi \wedge \psi)(\overline{a}) = \begin{cases} 1 & \text{if } \phi(\overline{a}) = 1 \text{ and } \psi(\overline{a}) = 1 \\ 0 & \text{if } \phi(\overline{a}) = 0 \text{ or } \psi(\overline{a}) = 0 \\ \star & \text{otherwise} \end{cases}$$

$$(\neg\phi)(\overline{a}) = \begin{cases} 1 & \text{if } \phi(\overline{a}) = 0 \\ 0 & \text{if } \phi(\overline{a}) = 1 \\ \star & \text{if } \phi(\overline{a}) = \star \end{cases} \qquad (\phi \vee \psi)(\overline{a}) = \begin{cases} 1 & \text{if } \phi(\overline{a}) = 1 \text{ or } \psi(\overline{a}) = 1 \\ 0 & \text{if } \phi(\overline{a}) = 0 \text{ and } \psi(\overline{a}) = 0 \\ \star & \text{otherwise} \end{cases}$$

Note that $\phi(\overline{a})$ equals 1 (resp., 0) for some partial assignment $\overline{a}$, then $\phi(\overline{a}')$ also equals 1 (resp., 0) for every *Boolean* $\overline{a}'$ obtained by replacing every $\star$ in $\overline{a}$ with either a 0 or 1. The converse, however, is not true: The formula $\phi = v \vee \neg v$ evaluates to 1 on every Boolean assignment, yet is not 1 when evaluated at $\star$. Thus, the "law of excluded middle" $\phi \vee \neg\phi \equiv 1$ no longer holds in this setting. However, other identities in boolean algebra such as De Morgan's laws (e.g. $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$) do remain true.

Definition 4.5.2 *For a promise problem Π, the* characteristic function *of Π is the map $\chi_\Pi : \{0,1\}^* \to \{0,1,\star\}$ given by*

$$\chi_\Pi(x) = \begin{cases} 1 & \text{if } x \in \Pi_Y \\ 0 & \text{if } x \in \Pi_N \\ \star & \text{otherwise} \end{cases}$$

The following definition describes 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.)

Definition 4.5.3 *For any promise problem Π, we define a new promise problem $\Phi(\Pi)$ as follows:*

$$\begin{aligned} \Phi(\Pi)_Y &= \{(\phi, x_1, \ldots, x_m) : \phi(\chi_\Pi(x_1), \ldots, \chi_\Pi(x_m)) = 1\} \\ \Phi(\Pi)_N &= \{(\phi, x_1, \ldots, x_k) : \phi(\chi_\Pi(x_1), \ldots, \chi_\Pi(x_m)) = 0\}, \end{aligned}$$

where ϕ is a m-ary propositional formula. Mon(Π) *is defined analogously, except that only monotone ϕ (i.e., without negations) are considered.*

De Santis *et. al.* [DDPY94] show that Mon$(L) \in$ **PZK** for any language L which is "random self-reducible" or whose complement is self-reducible. They also show Mon$(L) \in$ **HVSZK** for any language whose complement has a "noninteractive" statistical zero-knowledge proof. In addition, they give statistical zero-knowledge proofs for some simple statements involving a random-self-reducible language and its complement and for threshold formulae over random-self-reducible languages. Damgård and Cramer [DC96] extend some of these results by showing that for any language L which has a 3-message honest-verifier public-coin statistical (resp., perfect) zero-knowledge proof, Mon$(L) \in$ **SZK** (resp., Mon$(L) \in$ **PZK**) and Mon$(\overline{L}) \in$ **HVSZK**. The results of [DC96] actually apply to all monotone functions which have an "efficient secret-sharing scheme with completion," not just monotone formulae. In this section, we prove a result which holds for all of **HVSZK** and for all Boolean formulae, not just monotone ones:

Theorem 4.5.4 *For any promise problem* $\Pi \in \mathbf{HVSZK}$, $\Phi(\Pi) \in \mathbf{HVSZK}$.

Although some of the results of [DDPY94, DC96] yield proof systems that are zero knowledge even for cheating verifiers, this theorem only yields honest-verifier proof systems. However, this weakness will be removed in Chapter 6, when we prove that $\mathbf{HVSZK} = \mathbf{SZK}$.

First, let us see how strong a result follows directly from what we have already shown. Note that it is trivial to give a proof system when we restrict to formulae which are *conjunctions, i.e.,* $\phi(v_1, \ldots, v_m) = \bigwedge_{i=1}^m v_i$. The prover and verifier execute the **HVSZK** proof system for Π (with error reduced to $\ll 1/m$) on each of the inputs x_i and the verifier accepts iff it would accept in all m executions. This can be simulated by running the simulator for Π's proof system m times, on each of the inputs. If all the x_i's are YES instances (*i.e.,* the formula is true when all the v_i's are set appropriately), then this simulation will be good, with deviation increased just by a factor of m.

Combining this observation with the fact that **HVSZK** is closed under complementation, it follows that we can also handle disjunctions. By an inductive argument, one can then construct proof systems for arbitrary formulae. This, however, does not yield our desired result, because the proof system may not be polynomially bounded in the size of the formula. Suppose, for example, that transforming the proof system for a problem Π into one for its complement or one for disjunctions over Π squares the running time of the proof system. Then one can only afford to apply the inductive step a constant number of times while keeping the time polynomial. Therefore, one obtains a closure result for *constant-depth* formulae (with unbounded fan-in), which are provably weaker than general formulae and do not contain simple functions like parity and majority [FSS84].

To obtain a more efficient construction, we first observe that it suffices to focus on the complete problem STATISTICAL DIFFERENCE. We then note that the techniques developed by De Santis *et. al.* to show that $\mathrm{Mon}(\overline{L}) \in \mathbf{HVSZK}$ for any random self-reducible language L directly generalize to SD. Specifically, our Direct Product Lemma (Lemma 3.1.15) and XOR Lemma (Lemma 3.1.16) are generalizations of the methods they use to represent AND and OR over random self-reducible languages. Combining these with our Reversal Mapping (Corollary 4.4.1) and Polarization Lemma (Lemma 3.1.12), we are able to efficiently handle any Boolean formula over SD. The resulting construction can be viewed as purely an efficient procedure for manipulating statistical difference, which may be of interest independent of statistical zero knowledge:

Lemma 4.5.5 $\Phi(\mathrm{SD})$ *reduces to* SD.

Proof: For intuition, consider two instances of statistical difference (X_0, X_1) and (Y_0, Y_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 $(X_0 \otimes Y_0, X_1 \otimes Y_1)$ 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 operation represents OR. Similarly, the XOR operation in Proposition 3.1.17 represents AND. To obtain Lemma 4.5.5, we will recursively apply these constructions, taking care to keep the running time polynomial.

Let $w = (\phi, (X_0^1, X_1^1), \ldots, (X_0^m, X_1^m))$ be an instance of $\Phi(\mathrm{SD})$. By applying De Morgan's Laws, we may assume that the only negations in ϕ are applied directly to the variables. By the Polarization Lemma (Lemma 3.1.12) and the Reversal Mapping (Corollary 4.4.1), we can construct in polynomial time pairs of circuits $(Y_0^1, Y_1^1), \ldots, (Y_0^m, Y_1^m)$ and $(Z_0^1, Z_1^1), \ldots, (Z_0^m, Z_1^m)$ such that

$$(X_0^i, X_1^i) \in \mathrm{SD}_Y \Rightarrow \mathrm{StatDiff}\left(Y_0^i, Y_1^i\right) \geq 1 - \frac{1}{3|\phi|} \text{ and } \mathrm{StatDiff}\left(Z_0^i, Z_1^i\right) \leq \frac{1}{3|\phi|}$$
$$(X_0^i, X_1^i) \in \mathrm{SD}_N \Rightarrow \mathrm{StatDiff}\left(Y_0^i, Y_1^i\right) \leq \frac{1}{3|\phi|} \text{ and } \mathrm{StatDiff}\left(Z_0^i, Z_1^i\right) \geq 1 - \frac{1}{3|\phi|}.$$

Consider the randomized recursive procedure $\mathrm{Sample}_w(\psi, b)$, given in Algorithm 4.5.6, which takes a subformula ψ of ϕ and a bit b as input.

Algorithm 4.5.6: $\mathrm{Sample}_w(\psi, b)$

Input: A subformula ψ of ϕ and a bit $b \in \{0, 1\}$

1. If $\psi = v_i$, sample $z \leftarrow Y_b^i$.
2. If $\psi = \neg v_i$, sample $z \leftarrow Z_b^i$.
3. If $\psi = \tau \vee \mu$,
   (a) Sample $z_1 \leftarrow \mathrm{Sample}_w(\tau, b)$;
   (b) Sample $z_2 \leftarrow \mathrm{Sample}_w(\mu, b)$;
   (c) Let $z = (z_1, z_2)$.
4. If $\psi = \tau \wedge \mu$,
   (a) Choose $c, d \leftarrow \{0, 1\}$ subject to $c \oplus d = b$;
   (b) Sample $z_1 \leftarrow \mathrm{Sample}_w(\tau, c)$;
   (c) Sample $z_2 \leftarrow \mathrm{Sample}_w(\mu, d)$;
   (d) Let $z = (z_1, z_2)$.
5. Output z.

Executing $\mathrm{Sample}_w(\phi, b)$ for $b \in \{0, 1\}$ takes time polynomial in $|w|$, because the number of recursive calls is equal to the number of subformulae of ϕ. For a subformula τ of ϕ, let $\Delta(\tau) = \mathrm{StatDiff}\left(\mathrm{Sample}_w(\tau, 0), \mathrm{Sample}_w(\tau, 1)\right)$. Then we can prove the following about Δ:

Claim 4.5.7 *Let* $\overline{a} = (\chi_{\mathrm{SD}}(X_0^1, X_1^1), \ldots, \chi_{\mathrm{SD}}(X_0^m, X_1^m))$. *For every subformula* ψ *of* ϕ*, we have:*

$$\psi(\overline{a}) = 1 \Rightarrow \Delta(\psi) \geq 1 - \frac{|\psi|}{3|\phi|}$$

$$\psi(\overline{a}) = 0 \quad \Rightarrow \quad \Delta(\psi) \leq \frac{|\psi|}{3|\phi|}.$$

Note that nothing is claimed when $\psi(\overline{a}) = \star$.

Proof of claim: By induction on subformulae of ϕ. It holds for atomic subformulae (*i.e.,* the variables v_i and their negations $\neg v_i$) by the properties of the Y_b^i's and Z_b^i's.

Consider the case when $\psi = \tau \vee \mu$. By construction, $\text{Sample}_w(\psi, b) = \text{Sample}_w(\tau, b) \otimes \text{Sample}_w(\mu, b)$. If $\psi(\overline{a}) = 1$, then either $\tau(\overline{a}) = 1$ or $\mu(\overline{a}) = 1$. Without loss of generality, say $\tau(\overline{a}) = 1$. Then, by Fact 2.2.2 (Item 5) and induction,

$$\Delta(\psi) \geq \Delta(\tau) \geq 1 - \frac{|\tau|}{3|\phi|} \geq 1 - \frac{|\psi|}{3|\phi|}.$$

If $\psi(\overline{a}) = 0$, then both $\tau(\overline{a}) = \mu(\overline{a}) = 0$. By Fact 3.1.14 and induction,

$$\Delta(\psi) \leq \Delta(\tau) + \Delta(\mu) \leq \frac{|\tau|}{3|\phi|} + \frac{|\mu|}{3|\phi|} \leq \frac{|\psi|}{3|\phi|}.$$

Now consider the case when $\psi = \tau \wedge \mu$. By construction, the distributions for ψ are those obtained by applying the XOR construction (Proposition 3.1.17) to the distributions for τ and μ. Hence, $\Delta(\psi) = \Delta(\tau) \cdot \Delta(\mu)$. If $\psi(\overline{a}) = 1$, then, by induction,

$$\Delta(\psi) \geq \left(1 - \frac{|\tau|}{3|\phi|}\right) \cdot \left(1 - \frac{|\mu|}{3|\phi|}\right) > 1 - \frac{|\tau| + |\mu|}{3|\phi|} \geq 1 - \frac{|\psi|}{3|\phi|}.$$

If $\psi(\overline{a}) = 0$, then, without loss of generality, say $\tau(\overline{a}) = 0$. By induction,

$$\Delta(\psi) \leq \Delta(\tau) \leq \frac{|\tau|}{3|\phi|} \leq \frac{|\psi|}{3|\phi|}.$$

□

Let A and B be circuits describing the computations of $\text{Sample}_w(\phi, 0)$ and $\text{Sample}_w(\phi, 1)$, respectively, (which take the random bits each procedure uses as input). By the above claim, $\text{StatDiff}\,(A, B) \geq 2/3$ if $w \in \Phi(\text{SD})_Y$ and $\text{StatDiff}\,(A, B) \leq 1/3$ if $w \in \Phi(\text{SD})_N$. In other words, the construction of A and B from w describes a Karp reduction from $\Phi(\text{SD})$ to SD. This reduction can be computed in polynomial time because Sample runs in polynomial time. ■

Theorem 4.5.4 follows readily from Lemma 4.5.5:

Proof of Theorem 4.5.4: Let Π be any promise problem in **HVSZK**. By the completeness of SD, there is a reduction from Π to SD. This induces a reduction from $\Phi(\Pi)$ to $\Phi(\text{SD})$. Composing this with the reduction in Lemma 4.5.5, we see that $\Phi(\Pi)$ reduces to SD $\in$ **HVSZK**. Since **HVSZK** is closed under reductions, $\Phi(\Pi) \in$ **HVSZK**. ■

Examining the proof of Lemma 4.5.5 more closely, it is easy to see that all parts of the construction, except for the Reversal Mapping, preserve the extreme cases of SD (*i.e.,* statistical difference 0 or 1, respectively). Thus, we obtain the following additional reductions.

Proposition 4.5.8 $\mathrm{Mon}(\Pi)$ *reduces to* Π *for* $\Pi \in \left\{\mathrm{SD}^{1,0}, \mathrm{SD}^{1,1/2}, \mathrm{SD}^{1/2,0}\right\}$. *In particular,* $\mathrm{Mon}(\mathrm{SD}^{1,1/2}) \in \mathbf{HVPZK}$.

Theorem 4.5.4, though stated as a sort of Boolean closure property, can also be viewed from a more computational point of view, as asserting the closure of **HVSZK** under a certain class of reductions.

Definition 4.5.9 (truth-table reduction [LLS75]) *We say a promise problem* Γ truth-table reduces *to a promise problem* Π, *written* $\Gamma \leq_{\mathtt{tt}} \Pi$, *if there exists a (deterministic) polynomial-time computable function* f, *which on input* x *produces a tuple* $(y_1, y_2, \ldots, y_m)$ *and a circuit* C *(with* m *input gates), such that*

$$\begin{aligned} x \in \Gamma_Y &\Rightarrow C(\chi_\Pi(y_1), \ldots, \chi_\Pi(y_m)) = 1 \\ x \in \Gamma_N &\Rightarrow C(\chi_\Pi(y_1), \ldots, \chi_\Pi(y_m)) = 0 \end{aligned}$$

Truth-table reductions are easily seen to be equivalent to *nonadaptive* Cook reductions. We further consider the case where we restrict the complexity of computing the output of the reduction from the answers to the queries:

Definition 4.5.10 ($\mathbf{NC}_1$ truth-table reductions) *A truth-table reduction* f *between promise problems is an* $\mathbf{NC}_1$ truth-table reduction[2] *if the circuit* C *produced by the reduction on input* x *has fan-in 2 and depth bounded by* $c_f \log|x|$, *where* c_f *is a constant independent of* x. *If there is an* $\mathbf{NC}_1$ *truth-table reduction from* Γ *to* Π, *we write* $\Gamma \leq_{\mathbf{NC}_1-\mathtt{tt}} \Pi$.

It is easy to see that closure under $\mathbf{NC}_1$ truth-table reductions is equivalent to closure under the $\Phi(\cdot)$ operator (together with closure under Karp reductions).

Proposition 4.5.11 *A class* $\mathbf{C}$ *of promise problems is closed under* $\mathbf{NC}_1$ *truth-table reductions iff the following two conditions hold:*

1. $\Pi \in \mathbf{C} \Rightarrow \Phi(\Pi) \in \mathbf{C}$.
2. $\mathbf{C}$ *is closed under Karp reductions.*

Proof: $\Rightarrow$. Suppose $\mathbf{C}$ is closed under $\mathbf{NC}_1$ truth-table reductions. It is well-known that every formula ϕ can be transformed (in polynomial time) to an equivalent "balanced" formula ϕ' of depth $O(\log|\phi|)$ [Spi71]. Thus the map $(\phi, x_1, \ldots, x_m) \mapsto (\phi', x_1, \ldots, x_m)$ is an $\mathbf{NC}_1$ truth-table reduction from $\Phi(\Pi)$ to Π for any promise problem Π. Thus, if $\Pi \in \mathbf{C}$, then $\Phi(\Pi) \in \mathbf{C}$. For closure under Karp reductions, we simply note that every

[2] This terminology is inherited from the $\mathbf{NC}$ hierarchy of languages, where $\mathbf{NC}_i$ denotes the class of languages decided by (uniform) families of circuits of depth $O(\log^i n)$. See, *e.g.,* [Sip97, Pap94].

Karp reduction is also an $\mathbf{NC}_1$ truth-table reduction (take C to be the identity function on one variable).
$\Leftarrow$. Suppose that $\mathbf{C}$ is closed under $\Phi(\cdot)$ and Karp reductions. Any circuit C of depth d can be transformed in time $\mathrm{poly}(|C|, 2^d)$ into a formula ϕ_C. Let f be an $\mathbf{NC}_1$ truth-table reduction from a promise problem Γ to a promise problem $\Pi \in \mathbf{C}$. Composing f with the map $(C, x_1, \ldots, x_m) \mapsto (\phi_C, x_1, \ldots, x_m)$ gives a Karp reduction from Γ to $\Phi(\Pi)$ (computable in time $\mathrm{poly}(|x|, 2^{c_f \log |x|}) = \mathrm{poly}(|x|))$. By closure under $\Phi(\cdot)$ and Karp reductions, it follow that $\Gamma \in \mathbf{C}$. ■

Thus, we have:

Corollary 4.5.12 **HVSZK** *is closed under* $\mathbf{NC}_1$ *truth-table reductions.*

This shows that **HVSZK** has a substantial amount of robustness and richness as a class of computational problems. It would be very interesting to strengthen this result to more general forms of Cook or truth-table reductions, or give evidence that it is not possible to do so.

Open Problem 4.5.13 *Is* **HVSZK** *closed under general truth-table reductions? or (adaptive) Cook reductions?*

4.6 Knowledge complexity

The various definitions of zero-knowledge proofs beautifully capture what it means to learn "nothing" from an interactive proof. For interactive proofs which are not zero knowledge, it is natural to try to measure *how much* the verifier learns from the proof. Indeed, the conference version of the paper of Goldwasser, Micali, and Rackoff [GMR89] which introduced zero-knowledge proofs also suggested a more general notion of *knowledge complexity* to accomplish this task. However, the formalization of (non-zero) knowledge complexity suggested there does not seem to coincide with an intuitive notion of how much knowledge is leaked in a protocol (*cf.,* [GP91]). Goldreich and Petrank [GP91] presented a number of alternative definitions of knowledge complexity, which have been studied further by a number of researchers [GP91, GOP98, ABV95, PT96]. In this section, we use the results we have obtained on statistical zero knowledge to obtain new results about (non-zero) knowledge complexity.

4.6.1 Definitions

Recall that zero-knowledge proofs are defined by requiring that there is an efficient simulator whose output is "close" to the verifier's view of the interaction with the prover. Most of the definitions given by Goldreich and Petrank measure knowledge complexity by how much "help" a simulator needs to produce a good simulation. The various formulations arise from allowing the help to be given in different forms (*e.g.,* as a string or from an oracle) and from different methods of measuring the amount of help (*e.g.,* averaging over the simulator's coin tosses or taking the maximum).

There is one subtlety in our versions of the definitions that does not arise in the literature. In our definitions of interactive proofs and zero-knowledge proofs, the error parameters are controlled by a security parameter independent of the input length. It is not clear whether the knowledge complexity should be measured as a function of the security parameter in addition to the input length. It is not even clear whether for "natural" problems, the knowledge complexity would be increasing or decreasing as a function of the security parameter. On one hand, the prover may have to reveal more knowledge in order to reduce the error parameters. On the other hand, an increased security parameter gives the simulator more running time and hence the simulator may need less help.

Because of this unclear situation, we allow the knowledge complexity of *proof systems* to be a function of both the input length and the security parameter, but, in order to maintain consistency with the literature, we set the security parameter equal to the input length when defining the knowledge complexity of *promise problems* and the associated hierarchies.[3] In all our definitions, the knowledge complexity of a proof system (resp., promise problem) will be a function $\kappa : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ (resp., $\kappa : \mathbb{N} \to \mathbb{R}$). Throughout our discussion on knowledge complexity, we require that $\kappa(n,k)$ (resp., $\kappa(n)$) is computable in time $\mathrm{poly}(n,k)$ (resp., $\mathrm{poly}(n)$).

We now give definitions for several forms of knowledge complexity. Modulo some minor modifications for consistency with the rest of this thesis, all of these definitions are due to to Goldreich and Petrank [GP91], except for knowledge complexity in the "entropy sense" which is due to Aiello, Bellare, and Venkatesan [ABV95]. We only give definitions for perfect and statistical knowledge complexity, as all languages in **IP** have computational knowledge complexity zero (*i.e.,* are in **CZK**) if (nonuniformly) one-way functions exist [GMW91, IY87, BGG$^+$88]. We also only give definitions for honest verifiers, since we have not yet even given the definition of zero knowledge for cheating verifiers.

The first definition of knowledge complexity provides "help" to the simulator in the most direct manner — as a "hint" string given as an additional input.

Definition 4.6.1 (hint sense) *Let (P,V) be an interactive proof system for a promise problem Π. The* statistical knowledge complexity of (P,V) in the hint sense *is said to be $\kappa : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ if there is a function $h : \Pi_Y \times \mathbb{N} \to \{0,1\}^*$, a useful*[4] *probabilistic polynomial-time algorithm S, and a negligible function $\mu : \mathbb{N} \to [0,1]$ such that for all $x \in \Pi_Y$ and all $k \in \mathbb{N}$,*

1. $|h(x,k)| = \kappa(|x|,k)$.
2. $\mathrm{StatDiff}\left(\widetilde{S}(x,1^k,h(x,k)), \langle P,V\rangle(x,1^k)\right) \leq \mu(k)$.

[3] This is an admittedly ad-hoc solution to the problem, and will hopefully be remedied as our understanding of knowledge complexity improves. We still would prefer to have the error parameters (completeness, soundness, simulator deviation) controlled by a security parameter independent of the input length, whereas it seems that knowledge complexity should be primarily a function of the input length. But, as shown by several researchers [GOP98, PT96], the knowledge complexity and error parameters are not free to vary independently for certain languages.

[4] Recall that a probabilistic algorithm A is called *useful* if $\Pr[A(x) = \texttt{fail}] \leq 1/2$ for all x and $\widetilde{A}(x)$ denotes the output distribution of A on input x, conditioned on $A(x) \neq \texttt{fail}$.

$h(x,k)$ is called the hint*, S is called a* simulator*, and μ is called the* simulator deviation*. If $\mu \equiv 0$, then (P,V) is said to have* perfect knowledge complexity κ in the hint sense*.*

As with zero knowledge, allowing the simulator to fail with probability 1/2 is inessential for statistical knowledge complexity in the hint sense, as the failure probability can be made exponentially small and absorbed into the simulator deviation.

In the remaining definitions, help is provided to the simulator by means of an oracle which it can query. If $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$ is any function and M is an algorithm, then we write $M^{\mathcal{O}}$ to indicate that M is being given oracle access to M. The first oracle-based definition measures the amount of help provided by the oracle by the *maximum* number of bits it sends to the simulator and does not allow the simulator any failure probability.

Definition 4.6.2 (strict oracle sense) *Let (P,V) be an interactive proof system for a promise problem Π. The* statistical knowledge complexity of (P,V) in the strict oracle sense *is said to be $\kappa : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ if there is a function $\mathcal{O} : \{0,1\}^* \to \{0,1\}$, a probabilistic polynomial-time algorithm S, and a negligible function $\mu : \mathbb{N} \to [0,1]$ such that for all $x \in \Pi_Y$ and all $k \in \mathbb{N}$,*

1. *$S^{\mathcal{O}}(x,1^k)$ receives at most $\kappa(|x|,k)$ bits from $\mathcal{O}$, with probability 1 over the coins of S.*
2. $\mathrm{StatDiff}\left(S^{\mathcal{O}}(x,1^k), \langle P,V\rangle(x,1^k)\right) \le \mu(k)$.

S is called a simulator *and μ is called the* simulator deviation*. If $\mu \equiv 0$, then (P,V) is said to have* perfect knowledge complexity κ in the strict oracle sense*.*

The next definition allows the simulator to fail with probability 1/2.

Definition 4.6.3 (oracle sense) *Let (P,V) be an interactive proof system for a promise problem Π. The* statistical knowledge complexity of (P,V) in the oracle sense *is said to be $\kappa : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ if there is a function $\mathcal{O} : \{0,1\}^* \to \{0,1\}$, a useful probabilistic polynomial-time algorithm S, and a negligible function $\mu : \mathbb{N} \to [0,1]$ such that for all $x \in \Pi_Y$ and all $k \in \mathbb{N}$,*

1. *$S^{\mathcal{O}}(x,1^k)$ receives at most $\kappa(|x|,k)$ bits from $\mathcal{O}$, with probability 1 over the coins of S.*
2. $\mathrm{StatDiff}\left(\widetilde{S}^{\mathcal{O}}(x,1^k), \langle P,V\rangle(x,1^k)\right) \le \mu(k)$.

S is called a simulator *and μ is called the* simulator deviation*. If $\mu \equiv 0$, then (P,V) is said to have* perfect knowledge complexity κ in the oracle sense*.*

In contrast to statistical knowledge complexity in the hint sense, it is not apparent that allowing the simulator to fail in the oracle sense is inessential, as reducing the failure probability of the simulator to negligible may involve making more oracle queries. However, Goldreich and Petrank [GP91] show that the statistical knowledge complexities of any proof system in the strict oracle and oracle senses differ by at most $\log\log k + g(k)$ for any unbounded function $g(\cdot)$ (and this cannot be improved).

The next definition measures the average number of bits of help the simulator gets from the oracle.

Definition 4.6.4 (average oracle sense) *Let (P, V) be an interactive proof system for a promise problem Π. The* statistical knowledge complexity of (P, V) in the average oracle sense *is said to be $\kappa : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ if there is a function $\mathcal{O} : \{0,1\}^* \to \{0,1\}$, a useful probabilistic polynomial-time algorithm S, and a negligible function $\mu : \mathbb{N} \to [0,1]$ such that for all $x \in \Pi_Y$ and all $k \in \mathbb{N}$,*

1. *The expected number of bits that $S^{\mathcal{O}}(x, 1^k)$ receives from $\mathcal{O}$ is at most $\kappa(|x|, k)$, with the expectation being taken over the coins of S.*
2. $\mathrm{StatDiff}\left(\widetilde{S}^{\mathcal{O}}(x, 1^k), \langle P, V\rangle(x, 1^k)\right) \le \mu(k)$.

S is called a simulator *and μ is called the* simulator deviation*. If $\mu \equiv 0$, then (P, V) is said to have* perfect knowledge complexity κ in the average oracle sense.

In contrast to Goldreich and Petrank [GP91], we allow the simulator fail in defining the average oracle sense (as in [ABV95]). This affects the knowledge complexity by at most an additive constant, as the simulator can use the oracle to "hone in" on a non-failing set of random coins in an expected constant number of queries, analogous to the proof of [GP91, Prop. 4.6]. Goldreich and Petrank [GP91] have shown that the knowledge complexity of a proof system in the average oracle sense can be much smaller than its knowledge complexity in the oracle sense.

To address a feeling that the above measures slightly "overcount" the knowledge complexity, particularly for knowledge complexities close to 0, Aiello, Bellare, and Venkatesan [ABV95] introduced yet another measure of knowledge complexity. Essentially, this measure avoids counting the bits sent from the oracle to the simulator S to the extent that those bits can be guessed without access to the oracle $\mathcal{O}$. That is, another machine S', called an oracle simulator, is considered. S' is given the input x, the security parameter k, and the random coins R of the simulator, and attempts to guess the output of the simulator *without having access to the oracle $\mathcal{O}$*. The unpredictability of this output by S', as measured in an entropy-like fashion, is taken to be an upper bound on the amount of "useful information" obtained from the oracle.

Definition 4.6.5 (entropy sense) *Let (P, V) be an interactive proof system for a promise problem Π. The* statistical knowledge complexity of (P, V) in the entropy sense *is said to be $\kappa : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ if there is a function $\mathcal{O} : \{0,1\}^* \to \{0,1\}$, a useful probabilistic polynomial-time algorithm S, a probabilistic polynomial-time algorithm S', and a negligible function $\mu : \mathbb{N} \to [0,1]$ such that for all $x \in \Pi_Y$ and all $k \in \mathbb{N}$,*

1. $\mathrm{E}_R\left[\log\left(1/P_{x,k}(R)\right)\right] \le \kappa(|x|, k)$*, where* $P_{x,k}(R) = \Pr_\rho\left[S'(x, 1^k, R; \rho) = S^{\mathcal{O}}(x, 1^k; R)\right]$.
2. $\mathrm{StatDiff}\left(\widetilde{S}^{\mathcal{O}}(x, 1^k), \langle P, V\rangle(x, 1^k)\right) \le \mu(k)$.

S is called a simulator *and μ is called the* simulator deviation*. If $\mu \equiv 0$, then (P, V) is said to have* perfect knowledge complexity κ in the entropy sense.

Aiello, Bellare, and Venkatesan [ABV95] have shown that knowledge complexity in the entropy sense is always at most the knowledge complexity in the average oracle sense, and can be smaller by at most an additive constant.

It is clear that the prover-to-verifier communication of any interactive proof upper-bounds its perfect knowledge complexity in each of the above senses, *except the hint sense.* For this reason, the hint sense is viewed as an inadequate measure of knowledge complexity [GP91].

Each of the above forms of knowledge complexity gives rise to a hierarchy of promise problems.

Definition 4.6.6 (knowledge complexity hierarchies) *A promise problem* Π *has* knowledge complexity $\kappa : \mathbb{N} \to \mathbb{R}$ *in one of the above senses if there exists an interactive proof for* Π *with* negligible *completeness and soundness errors whose knowledge complexity* $\kappa' : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ *(in the given sense) satisfies* $\kappa'(n,n) \leq \kappa(n)$ *for all* n.

The classes of promise problems with statistical knowledge complexity $\kappa(n)$ *in the hint, strict oracle, oracle, average oracle, and entropy senses are denoted by* $\mathbf{SKC_{hint}}(\kappa(n))$, $\mathbf{SKC_{strict}}(\kappa(n))$ $\mathbf{SKC_{oracle}}(\kappa(n))$, $\mathbf{SKC_{avg}}(\kappa(n))$, *and* $\mathbf{SKC_{ent}}(\kappa(n))$, *respectively. The classes of promise problems with perfect knowledge complexity are similarly denoted by* **PKC** *with the appropriate subscript.*

Since the above definition only refers to case when the security parameter is equal to the input length, we will often omit the security parameter from the notation in what follows. It is clear that the bottom level (*i.e.,* $\kappa(n) \equiv 0$) of each of the statistical (resp., perfect) knowledge complexity hierarchies is exactly **HVSZK** (resp., **HVPZK**). An important question about these hierarchies is whether they are strict or not, and previously no collapses were known for any of them.

4.6.2 A Collapse for the Hint Hierarchy

The first thing we will prove in this section is a lemma showing that statistical knowledge complexity in the hint sense can be expressed in terms of statistical *zero* knowledge. This lemma will enable us to immediately deduce a number of results about the $\mathbf{SKC_{hint}}$ hierarchy from our results on **HVSZK**. Most significantly, the Boolean closure properties of **HVSZK** demonstrated in the previous section will easily imply that the statistical knowledge complexity hierarchy for the hint sense collapses by logarithmic additive terms at all levels. As mentioned earlier, the hint sense has some deficiencies as a measure of knowledge complexity, so it would be of greater interest to obtain such results for the other forms of knowledge complexity. Our results are best viewed as a first step in this direction.

Lemma 4.6.7 *Let* $\kappa : \mathbb{N} \to \mathbb{N}$ *be any polynomially bounded function. Then* $\Pi \in \mathbf{SKC_{hint}}(\kappa(n))$ *(resp.,* $\mathbf{PKC_{hint}}(\kappa(n))$*) iff there exists a promise problem* $\Gamma \in$ **HVSZK** *(resp.,* **HVPZK***) such that*

1. $x \in \Pi_Y \Rightarrow$ *there exists* $a \in \{0,1\}^{\kappa(|x|)}$ *such that* $(x,a) \in \Gamma_Y$, *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 similar.

$\Rightarrow$ Let Π be a promise problem in $\mathbf{SKC_{hint}}(\kappa(n))$ 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:

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

The protocol and simulator for Π *almost* immediately yield an honest-verifier statistical zero-knowledge proof for Γ (the verifier and prover for Γ should ignore the second component of the input and the simulator should use it as a hint). There is one small technicality, however. Since our definition of $\mathbf{SKC_{hint}}(\kappa(n))$ only refers to the knowledge complexity when the security parameter is set equal to the input length, the simulator deviation of the resulting proof system for Γ is only negligible as a function of the input length, not the security parameter. So, we only obtain $\Gamma \in$ **weak-HVSZK**. But now we can apply our result that **weak-HVSZK** = **HVSZK** (Corollary 4.3.1). It is clear that Γ satisfies the other conditions of Lemma 4.6.7.

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

1. For all x, $|h(x)| = \kappa(|x|)$,
2. $x \in \Pi_Y \Rightarrow (x, h(x)) \in \Gamma_Y$.

(Such a function is guaranteed by Condition 1.) We now give a proof system for Π of knowledge complexity $\kappa(n)$. 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 Γ's proof system. This proof system is easily seen to have knowledge complexity $\kappa(n)$ in the hint sense, using h as the hint function and the same simulator as for Γ's proof system. ■

From this lemma and its proof, we can immediately apply some of our results about **HVSZK** to $\mathbf{SKC_{hint}}$. First, looking at the proof of the "if" ($\Leftarrow$) direction of Lemma 4.6.7, we see that any problem in $\mathbf{SKC_{hint}}(\kappa(n))$ has a proof system which is simply an **HVSZK** proof system augmented by the prover sending $\kappa(n)$ bits in the first message. Combining this with the efficient **HVSZK** proof systems of Corollary 4.1.1 (possibly repeated in parallel), we obtain the following:

Corollary 4.6.8 *Let $\kappa, t : \mathbb{N} \to \mathbb{N}$ be any two polynomially bounded functions which are both computable in time polynomial in their arguments. Every problem in $\mathbf{SKC_{hint}}(\kappa(n))$ has an interactive proof system with the following properties (on input $(x, 1^k)$):*

1. *The statistical knowledge complexity is $\kappa(|x|)$.*
2. *The proof system exchanges only 3 messages.*
3. *The prover-to-verifier communication is $\kappa(|x|) + t(k)$ bits.*

4. *The completeness error and simulator deviation are both* 2^{-k}.

5. *The soundness error is* $1/2^{t(k)}$.

6. *The prover is deterministic.*

In particular, we have:

Corollary 4.6.9 ([GP91]) *For any polynomially bounded function* $\kappa : \mathbb{N} \to \mathbb{N}$,

$$\mathbf{SKC_{hint}}(\kappa(n)) \subset \mathbf{AM}.$$

Actually, this corollary does not neeed the full power of Corollary 4.1.1. The result of Aiello and Håstad [AH91] that **HVSZK** $\subset$ **AM** (cf., Corollary 4.2.2) together with Lemma 4.6.7 suffices.

By Proposition 4.2.3, **co-NP** does not have polynomial statistical knowledge complexity in the hint sense unless the Polynomial Hierarchy collapses. On the other hand, **co-NP** does possess interactive proofs in which the prover sends only polynomially many bits to the verifier (by definition) [LFKN92]; this intuitively should imply that the verifier is only gaining polynomially many bits of knowledge. This discrepancy is one of the deficiencies of the hint sense as a measure of knowledge complexity pointed out by Goldreich and Petrank [GP91].

Another result of ours about **HVSZK** that can be directly applied to $\mathbf{SKC_{hint}}$ is Corollary 4.3.1, which states that **weak-HVSZK** = **HVSZK**. In analogy with **weak-HVSZK**, once can define weak forms of the **SKC** hierarchies, in which for every polynomial $p : \mathbb{N} \to \mathbb{N}$ there should be a simulator that achieves simulator deviation $1/p(|x|)$ on input x using $\kappa(|x|)$ bits of help.

Corollary 4.6.10 *For any polynomially bounded function* $\kappa : \mathbb{N} \to \mathbb{N}$

$$\textbf{weak-}\mathbf{SKC_{hint}}(\kappa(n)) = \mathbf{SKC_{hint}}(\kappa(n)).$$

Proof: Note that the proof of the "only if" ($\Rightarrow$) direction of Lemma 4.6.7 yields $\Gamma \in$ **weak-HVSZK** = **HVSZK** satisfying the properties listed in lemma even if Π is only in **weak-**$\mathbf{SKC_{hint}}(\kappa(n))$. Now, applying the "if" ($\Leftarrow$) direction of the lemma to Γ, we see that Π is actually in (non-weak) $\mathbf{SKC_{hint}}(\kappa(n))$. ■

Finally, we use the Boolean closure properties we have proven about **HVSZK** to show a collapse in the $\mathbf{SKC_{hint}}$ hierarchy.

Theorem 4.6.11 *For any polynomially bounded function* $\kappa : \mathbb{N} \to \mathbb{R}$,

$$\mathbf{SKC_{hint}}(\kappa(n) + \log n) = \mathbf{SKC_{hint}}(\kappa(n)).$$

Proof: For intuition, consider the case $\kappa \equiv 0$, and let Π be any promise problem in $\mathbf{SKC_{hint}}(\log n)$. By Lemma 4.6.7, there is a promise problem $\Gamma \in$ **HVSZK** such that proving that x is a YES instance of Π amounts to proving that for at least one string a of length $\log |x|$, (x, a) is a YES instance of Γ. This is an OR of polynomially many statements about membership in Γ. In Section 4.5, we showed that any such Boolean formula over a problem in **HVSZK** can also be proven in **HVSZK**, so it follows that $\Pi \in \mathbf{HVSZK} = \mathbf{SKC_{hint}}(0)$, as desired. To deal with $\kappa(n) > 0$, we only take the OR over the last $\log n$ bits of the hint, and use the "if" direction of Lemma 4.6.7 to pass back to $\mathbf{SKC_{hint}}$.

We now proceed with the formal proof. Let Π be a language in $\mathbf{SKC_{hint}}(\kappa(n) + \log n)$ and let Γ be the related promise problem guaranteed by the "only if" ($\Rightarrow$) direction of Lemma 4.6.7. Now consider a different promise problem Γ', defined by

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

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 definition of Γ' implies that

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

is an $\mathbf{NC}_1$ truth-table reduction from Γ' to Γ. Since **HVSZK** is closed under such reductions (Corollary 4.5.12), $\Gamma' \in \mathbf{HVSZK}$.

Now, if $x \in \Pi_Y$, then there exists an a of length $\kappa(|x|) + \log(|x|)$ such that $(x,a) \in \Gamma_Y$. Taking a' to be the first $\kappa(|x|)$ bits of a, we see that there exists an a' of length $\kappa(|x|)$ such that $(x,a') \in \Gamma'_Y$. On the other hand, if $x \in \Pi_N$, then for all a, $(x,a) \in \Gamma'_N$. Thus, by the "if" ($\Leftarrow$) direction of Lemma 4.6.7, we conclude that $\Pi \in \mathbf{SKC_{hint}}(\kappa(n))$. ∎

It would be unexpected to strengthen Theorem 4.6.11 by increasing the $\log n$ to any function $f(n) = \omega(\log n)$, because $\mathbf{SKC_{hint}}(f(n))$ contains every problem solvable in nondeterministic time $f(n)$ (actually even polynomial time with $f(n)$ bits of nondeterminism), and it seems unlikely that all such problems would be contained in $\mathbf{HVSZK} \subset \mathbf{co\text{-}AM}$.

As stated above, it would be more significant to obtain a similar collapse for one of the other forms of knowledge complexity, or give evidence that such a collapse does not occur. Perhaps our work on statistical zero knowledge can also help in such a task. In particular, in Section 3.3, we used the Aiello–Håstad simulator analysis to show that every problem in **HVSZK** reduces to Entropy Difference. Petrank and Tardos [PT96] have used ideas from the the Aiello–Håstad simulator analysis to study interactive proofs with logarithmic statistical knowledge complexity in the oracle sense, and thereby showed that $\mathbf{SKC_{oracle}}(\log n) \subset \mathbf{AM} \cap \mathbf{co\text{-}AM}$. Perhaps all these ideas can be combined to obtain stronger results about knowledge complexity in the oracle sense. Specifically, showing that every problem in $\mathbf{SKC_{oracle}}(\log n)$ reduces to Entropy Difference would imply that $\mathbf{SKC_{oracle}}(\log n) = \mathbf{HVSZK}$.

Open Problem 4.6.12 *Do any of the other knowledge complexity hierarchies defined by Goldreich and Petrank [GP91] collapse?*

Our results about **HVSZK** in the next two chapters will also immediately imply similar results about $\mathbf{SKC_{hint}}$ via Lemma 4.6.7. Specifically, they will imply that, for any polynomially bounded function $\kappa : \mathbb{N} \to \mathbb{N}$, $\mathbf{SKC_{hint}}(\kappa(n))$ proofs can always be transformed into ones which use public coins, and into ones that have knowledge complexity $\kappa(n)$ even against cheating verifiers.

4.6.3 The Relationship between Perfect and Statistical Knowledge Complexity

One of the major questions about zero-knowledge proofs that has been open since the defining paper of Goldwasser, Micali, and Rackoff [GMR89] is whether perfect and statistical zero knowledge coincide. That is, does **PZK** = **SZK** (or even **HVPZK** = **HVSZK**)? This question motivates a more general study of the relationship between perfect and statistical knowledge complexity. Goldreich, Ostrovsky, and Petrank [GOP98] have shown that, in the oracle sense, the perfect and statistical knowledge complexity are not too far apart. Specifically, they have shown that $\mathbf{SKC_{oracle}}(\kappa(n)) \subset \mathbf{PKC_{oracle}}(\kappa(n) + O(\log n))$ for every function $\kappa : \mathbb{N} \to \mathbb{R}$. In particular, $\mathbf{HVSZK} \subset \mathbf{PKC_{oracle}}(O(\log n))$.

Aiello, Bellare, and Venkatesan [ABV95] have shown that the relationship is even tighter for the average oracle and entropy senses. Specifically, they proved that, for every polynomially bounded function $\kappa : \mathbb{N} \to \mathbb{R}$, $\mathbf{SKC_{avg}}(\kappa(n)) \subset \mathbf{PKC_{avg}}(\kappa(n) + 1 + n^{-\omega(1)})$ and $\mathbf{SKC_{ent}}(\kappa(n)) \subset \mathbf{PKC_{ent}}(\kappa(n) + n^{-\omega(1)})$, and the latter inclusion becomes an equality for $\kappa \equiv 0$. In addition, they give analogous results for the cheating-verifier versions of these classes.

Our completeness theorems give another way to obtain such results when $\kappa \equiv 0$, *i.e.*, when we want bounds on the perfect knowledge complexity of **HVSZK**. Namely, these theorems reduce the question to measuring the perfect knowledge complexity of *specific* proof systems for the complete problems. By a straightforward analysis of (variants of) Protocol 3.1.19, we obtain:

Theorem 4.6.13

1. *For any function* $\kappa(n) = \omega(\log n)$, $\mathbf{HVSZK} \subset \mathbf{PKC_{strict}}(\kappa(n))$.
2. *For any* $c > 0$, $\mathbf{HVSZK} \subset \mathbf{PKC_{avg}}(1 + 2^{-n^c})$.
3. *For any* $c > 0$, $\mathbf{HVSZK} = \mathbf{PKC_{ent}}(2^{-n^c})$.

The last two items improve on the bounds of [ABV95] for knowledge complexity 0 (though their results also apply to the cheating-verifier classes). These two items could also be obtained by applying their proofs to our result that every problem in **HVSZK** has a proof with simulator deviation 2^{-k}.

Proof:

1. Let Π be any problem in **HVSZK** and let $\kappa(n) = \omega(\log n)$. By Corollary 4.1.1, there is a proof system for Π with negligible completeness error, constant soundness error, and 1 bit of prover-to-verifier communication. Executing this protocol $\kappa(n)$ times in parallel results in a proof system with negligible error probabilities and prover-to-verifier communication $\kappa(n)$. The prover-to-verifier communication is an upper bound on the perfect knowledge complexity in the strict oracle sense.

2. Let Π be any promise problem in **HVSZK**. Consider the proof system for Π in which proceeds as follows on an input x of length n: Both parties apply the reduction to SD to obtain an instance (X, Y) of SD. They execute Protocol 3.1.19 n times (sequentially or in parallel) on input (X, Y) with the security parameter set to $k = 4n^c$, and the verifier accepts if the prover is correct in all subprotocols. This proof system has negligible completeness and soundness errors.

   A perfect simulator for the proof system can be obtained as follows: The simulator simulates the verifier strategy and queries the oracle once to find out if the prover would give an incorrect response in any of the executions of Protocol 3.1.19. Of the oracle replies yes, then the simulator queries the oracle n more times to find out which prover answers would be incorrect. The simulator then outputs the random coins used for running the verifier strategy together with the appropriate prover responses.

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

3. We consider the same protocol used in the proof of Part 2 above and show that it has perfect knowledge complexity 2^{-n^c} in the entropy sense. Let S be the simulator which simply simulates the verifier and queries the oracle 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 any R which corresponds to a transcript in which the prover would make an error. Thus, we instead have our oracle simulator S' assume that the prover is right in each subprotocol independently with probability $1 - \delta$, where $\delta = 2^{-2n^c}$. Thus, $P_x(R) = (1-\delta)^k \delta^{n-k}$, if R is a set of random coins for the verifier (equivalently S, since S mimics the verifier) 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

$$
\begin{aligned}
&\mathop{\mathrm{E}}_{R}\left[\log \frac{1}{P_x(R)}\right] \\
&= \sum_{k=0}^{n} \binom{n}{k} \epsilon^{n-k}(1-\epsilon)^k \log\left(\frac{1}{(1-\delta)^k \delta^{n-k}}\right) \\
&= \left(\log \frac{1}{\delta^n}\right) \cdot \left[\sum_{k=0}^{n} \binom{n}{k} \epsilon^{n-k}(1-\epsilon)^k\right] + \left(\log \frac{\delta}{1-\delta}\right) \cdot \left[\sum_{k=0}^{n} \binom{n}{k} \epsilon^{n-k}(1-\epsilon)^k k\right] \\
&= \left(\log \frac{1}{\delta^n}\right) \cdot 1 + \left(\log \frac{\delta}{1-\delta}\right) \cdot n \cdot (1-\epsilon) \cdot \left[\sum_{k=1}^{n} \binom{n-1}{k-1} \epsilon^{n-k}(1-\epsilon)^{k-1}\right] \\
&= \log \frac{1}{\delta^n} + n \cdot (1-\epsilon) \cdot \left(\log \frac{\delta}{1-\delta}\right) \cdot 1 \\
&= n\left(\log \frac{1}{1-\delta} + \epsilon \log \frac{1-\delta}{\delta}\right)
\end{aligned}
$$

$$\leq n\left(\log\frac{1}{1-\delta}+\delta^2\log\frac{1}{\delta}\right)$$
$$\leq 2n\delta < 2^{-n^c}$$

for sufficiently large n.

The opposite inclusion follows from the result of [ABV95] that $\mathbf{PKC_{ent}}(\mu(n)) \subset \mathbf{HVSZK}$ for any negligible function μ.

■

Despite these slightly improved bounds, the basic question about the relationship between perfect and statistical zero knowledge remains open.

Open Problem 4.6.14 *Does* **HVSZK** = **HVPZK**? *Does* **SZK** = **PZK**?

The Completeness Theorem may help in addressing this question, for now it becomes equivalent to asking whether STATISTICAL DIFFERENCE or ENTROPY DIFFERENCE has a perfect zero-knowledge proof.

4.7 Perfect and computational zero knowledge

The simulator analyses we used to prove the Completeness Theorems can also be applied to perfect and computational zero-knowledge proofs. Although we do not know how to obtain complete problems for **HVPZK** and **HVCZK** using these techniques, they do yield some additional insight into these classes.

We begin with the simulator analysis for public-coin proofs from Section 3.3. For perfect zero knowledge, we obtain a reduction to a variant of SD. Since the security parameter plays a less central role in perfect zero-knowledge proofs (there is no simulator deviation to control), we omit it in the statement and proof of this proposition.

Proposition 4.7.1 *Suppose a promise problem* Π *has an honest-verifier public-coin perfect zero-knowledge proof with perfect completeness. Then* Π *reduces to* $\overline{\mathrm{SD}}^{1/2,0}$.

More generally, the condition that the proof system has perfect completeness can be relaxed to requiring that the probability that verifier accepts an input $x \in \Pi_Y$ *be computable in polynomial time from* x.[5]

Proof: Let (P,V) be a perfect zero-knowledge proof for with perfect completeness and soundness error $s = 1/3$, with simulator S. The reduction constructs an instance (X,Y) of $\overline{\mathrm{SD}}^{1/2,0}$ from an instance x of Π. The distributions X and Y are constructed based on S (and V) exactly as in the proof of Theorem 3.2.5. The only change needed in the analysis is that Claim 3.2.6 should be replaced with one that states that $\mathrm{StatDiff}(X',Y') = 0$ when x is a YES instance. To see that this is the case, first note that Lemma 3.2.1 gives

[5] That is, we require that there is a (deterministic) polynomial-time algorithm A such that when given $x \in \Pi_Y$, A outputs the probability that V accepts on input x. A's behavior on NO instances or inputs that violate the promise can be arbitrary.

$\mathrm{StatDiff}(X_i, Y_i) = 2 \cdot 0 = 0$ for $i > 0$ when S is a perfect simulator. In addition, by perfect completeness and perfect simulation, Y_0 will always output 1 in the case of a YES instance, so $\mathrm{StatDiff}(X_0, Y_0) = 0$. Therefore, $\mathrm{StatDiff}(X', Y') = 0$. Since X and Y consist just of many independent copies of X' and Y', respectively, it follows that their statistical difference is also 0 in the case of a YES instance.

Now we treat the more general version in which the acceptance probability on YES instances is only assumed to be efficiently computable from the input. By repeating the proof system sequentially or in parallel and ruling by majority/threshold, we may assume that the completeness error is at most 1/3 (as is assumed in the proof of Theorem 3.2.5); note that majority/threshold rule preserves the property that the acceptance probability is efficiently computable. Now we construct X and Y just as before, with one small change. It is no longer the case that Y_0 always outputs 1, so we redefine X_0 as follows to compensate:

X_0: Calculate the probability p that the verifier accepts on input x (as if x were a YES instance). If $p < 2/3$, output 1. Otherwise let $t = 216 \ln 12v$ and calculate

$$q = \sum_{t/2 < j \leq t} \binom{t}{j} p^j (1-p)^{t-j}.$$

Output 1 with probability q, and 0 otherwise.

Note that q is exactly the probability that Y_0 outputs 1 on a YES instance, so we have $\mathrm{StatDiff}(X_0, Y_0) = 0$ for YES instances as desired. Now we must also check that this modification in X_0 does not hurt the analysis for NO instances. The only time X_0 plays a role is in the case that the simulator outputs accepting conversations with probability at most 5/12. There, a Chernoff bound is used to show that Y_0 outputs 1 with probability at most 1/2 in this case. A Chernoff bound similarly implies the modified X_0 given above always outputs 1 with high probability, certainly at least 3/4. This gives $\mathrm{StatDiff}(X_0, Y_0) \geq 1/4 > 1/12v$, so Claim 3.2.7 still holds. ■

If we completely remove the conditions on the acceptance probability, we can compensate by allowing the distributions to be samplable in *expected* polynomial time.

Proposition 4.7.2 *Suppose a promise problem Π has an honest-verifier public-coin perfect zero-knowledge proof. Then, there exist expected polynomial-time algorithms X and Y such that:*

1. $x \in \Pi_Y \Rightarrow \mathrm{StatDiff}(X(x), Y(x)) = 0$.
2. $x \in \Pi_N \Rightarrow \mathrm{StatDiff}(X(x), Y(x)) \geq 1/2$.

Proof Sketch: Again, the only difficulty is constructing the distributions X_0 and Y_0. Let r be the number of random coins used by the simulator S. Y_0 is modified so that it runs the simulator $t = \Theta(r + \ln v)$ times rather than just $\Theta(\ln v)$ times; this guarantees that Y_0 will output 1 with probability at least $1 - 2^{-r}$ in the case of a YES instance. X_0 is modified as follows:

X_0: With probability $1 - 2^{-r}$, just output 1. Otherwise, calculate the probability p that S outputs an accepting conversation (by exhaustive search over the random coins) and use that to calculate the probability q that Y_0 outputs 1 (a simple sum involving some binomial coefficients and p). If $q \geq 1 - 2^{-r}$, output 1 or 0 with exactly the right bias to guarantee that the overall probability of outputting 1 is q.

This definition of X_0 guarantees that $\mathrm{StatDiff}\,(X_0, Y_0) = 0$ for YES instances. Note that X_0 runs in expected polynomial time; with probability 2^{-r}, it does a calculation that takes time 2^r (times a polynomial factor). The tedious details are omitted. □

If we could show that $\mathrm{SD}^{1/2,0}$ (or its complement) has an (honest-verifier) public-coin perfect zero-knowledge proof, we would essentially have a completeness theorem for public-coin perfect zero knowledge. Interestingly, it is the "opposite" extreme case of SD — namely, $\mathrm{SD}^{1,1/2}$ — that we placed in (private-coin) **HVPZK** with Proposition 3.1.11. In Chapter 6, we shall see that $\overline{\mathrm{SD}^{1,0}}$ actually has a *public-coin* perfect zero-knowledge proof.

Adapting the simulator analysis of Section 3.2 to computational zero-knowledge proofs, we obtain the following.

Proposition 4.7.3 *Suppose a promise problem Π has an honest-verifier public-coin computational zero-knowledge proof system. Then there exist probabilistic polynomial-time algorithms X and Y such that*

1. $\{X(x, 1^k)\}_{x \in \Pi_Y, k \in \mathbb{N}}$ *and* $\{Y(x, 1^k)\}_{x \in \Pi_Y, k \in \mathbb{N}}$ *are computationally indistinguishable.*
2. $x \in \Pi_N \Rightarrow \mathrm{StatDiff}\left(X(x, 1^k), Y(x, 1^k)\right) \geq 1/2$.

Proof Sketch: Again, the construction of $X(x, 1^k)$ and $Y(x, 1^k)$ from x are just as in the proof of Theorem 3.2.5. The proof that they are statistically far for NO instances remains unchanged. To see that they are computationally indistinguishable for YES instances, one need only replace the arguments about statistical closeness with analogous ones referring to computationally indistinguishability (*e.g.,* Claim 3.2.6, Lemma 3.2.1). □

Clearly, an analogous proposition also holds for the traditional definition of **HVCZK** in which the indistinguishability in the zero-knowledge property is with respect to the input length and not a separate security parameter. (The indistinguishability of the resulting distributions $X(x)$ and $Y(x)$ will then hold with respect to the $|x|$ and not a separate security parameter.)

Note that the properties of the distributions produced by Proposition 4.7.3 cannot be used to distinguish between YES and NO instances in general, because there can be distributions which are both computationally indistinguishable and statistically far apart. Nevertheless, Proposition 4.7.3 will enable us to prove a nontrivial result about **HVCZK** in the next section.

Our simulator analysis for private-coin proofs in Section 3.3 can also be applied to perfect zero-knowledge proofs, and yields a reduction to the following variant of ENTROPY DIFFERENCE, denoted ED':

$$\begin{aligned} \mathrm{ED}'_Y &= \{(X, Y) : \mathrm{H}(X) = \mathrm{H}(Y)\} \\ \mathrm{ED}'_N &= \{(X, Y) : \mathrm{H}(Y) \geq \mathrm{H}(X) + 1\} \end{aligned}$$

Proposition 4.7.4 *Suppose a promise problem* Π *has an honest-verifier perfect zero-knowledge proof with perfect completeness. Then* Π *reduces to* ED'.

More generally, the condition that the proof system has perfect completeness can be relaxed to requiring that the probability that verifier accepts an input $x \in \Pi_Y$ *be computable in polynomial time from* x.

The proof of Proposition 4.7.4 consists of similar modifications to the proof of Theorem 3.3.13 as was needed to obtain Proposition 4.7.1 from the proof of Theorem 3.2.5. The details are omitted. As in Proposition 4.7.2, the computability condition on the acceptance probability can be removed, at the cost of yielding distributions that are samplable in expected polynomial time.

The simulator analysis of private-coin proofs can also be applied to computational zero knowledge, but unfortunately, it appears to yield something trivial. Specifically, it yields two probabilistic polynomial time algorithms X and Y such that (omitting the security parameter):

1. If $x \in \Pi_N$, $\mathrm{H}(Y(x)) \geq \mathrm{H}(X(x)) + 1$.
2. If $x \in \Pi_Y$, then there exist distributions $X'(x)$ and $Y'(x)$ such that $\mathrm{H}(X'(x)) \geq \mathrm{H}(Y'(x)) + 1$, $X'(x)$ is computationally indistinguishable from $X(x)$, and $Y'(x)$ is computationally indistinguishable from $Y(x)$.

These conditions are trivial in the sense that such algorithms X and Y can be shown to exist for any promise problem Π, regardless of whether it possesses a zero-knowledge proof: Fix $X(x)$ to be the uniform distribution on $|x|$ bits and $Y(x)$ the uniform distribution on $|x|+1$ bits, so the condition for NO instances certainly holds. To meet the condition for YES instances, take $X'(x)$ to equal $X(x)$ and $Y'(x)$ to be the uniform distribution on a subset of $\{0,1\}^{|x|+1}$ of size $2^{|x|-1}$. (Note that X' and Y' are not required to be samplable.)

The main question remaining here is whether a tighter characterization of **HVPZK** or **HVCZK** can be given.

Open Problem 4.7.5 *Exhibit natural complete problems for* **HVPZK** *or* **HVCZK**.

The question for **HVCZK** is only interesting if one does not assume that one-way functions exist, for under that assumption, **HVCZK** = **PSPACE** [GMW91, IY87, BGG$^+$88, LFKN92, Sha92], so any **PSPACE**-complete problem would do.

4.8 Zero-knowledge proofs for hard problems imply one-way functions

Looking at the array problems known to be in **HVSZK** — such as QUADRATIC RESIDUOSITY and NONRESIDUOSITY [GMR89], a problem equivalent to DISCRETE LOGARITHM [GK93], and approximate versions of the SHORTEST VECTOR and CLOSEST VECTOR problems for lattices [GG98a] — it strikes one that many of them are related to problems underlying various cryptosystems [DH76, GM84, ElG84, GGH97, AD97]. Ostrovsky [Ost91] showed that this is not a coincidence. Informally, he proved that if **HVSZK** contains *any* hard problem, then one-way functions exist and hence many cryptographic tasks can be accomplished. This may be surprising at first, because typically one-way functions are not explicitly used in constructing statistical zero-knowledge proofs. Rather, Ostrovsky's result should be understood as saying that the types of problems possessing statistical zero-knowledge proofs are the kind that would yield one-way functions if they were hard. Subsequently, Ostrovsky and Wigderson [OW93] generalized this result to computational zero knowledge — if **HVCZK** contains any hard problem, then one-way functions exist. From a very high level, their analysis of computational zero-knowledge proofs can be separated into two cases: if a hard language possesses an honest-verifier computational zero-knowledge proof, then one of the following two cases must hold: (a) the proof is really a statistical zero-knowledge proof, in which case Ostrovsky's result applies, or (b) one-way functions are implicitly being used in constructing a proof and simulation which are computationally indistinguishable but not statistically close.

In this section, we show how Ostrovsky's theorem follows readily from our Completeness Theorem and a result of Goldreich [Gol90] on computational indistinguishability. Using our analysis of public-coin computational zero-knowledge proofs (Proposition 4.7.3), we also obtain a simpler proof of the Ostrovsky–Wigderson theorem for the special case of public-coin proofs.

In order to state these theorems precisely, we need to define what we mean for a problem Π to be "hard." Informally, we require that membership in Π is (very) hard to decide under some samplable distribution of instances.

Definition 4.8.1 (samplable distributions) *An ensemble of distributions $\{D_n\}_{n\in\mathbb{N}}$ is said to be* samplable *if there is a probabilistic polynomial-time algorithm that, on input 1^n outputs a string distributed according to D_n.*

Definition 4.8.2 (hard-on-average problems) *A promise problem Π is* hard-on-average *if there exists a samplable ensemble of distributions $\{D_n\}_{n\in\mathbb{N}}$ such that the following holds: For every nonuniform probabilistic polynomial-time algorithm A, there exists a negligible function $\mu : \mathbb{N} \to [0,1]$ such that*

$$\Pr\left[A(x) \textit{ correctly decides whether } x \textit{ is a } \text{YES} \textit{ or } \text{NO} \textit{ instance of } \Pi\right] \le \frac{1}{2} + \mu(n) \qquad \forall n \in \mathbb{N},$$

where the probability is taken over $x \leftarrow D_n$ and the coins of A. (If x violates the promise, then A is considered to be correct no matter what it outputs.)

For completeness, we also define one-way functions.

Definition 4.8.3 (one-way functions) *A function $f : \{0,1\}^* \to \{0,1\}^*$ is* one way *if*

1. *f can be evaluated in polynomial time.*

2. *For every nonuniform probabilistic polynomial-time algorithm A, there is a negligible function $\mu : \mathbb{N} \to [0,1]$ such that*

$$\Pr_{x\leftarrow\{0,1\}^n}\left[A(f(x)) \in f^{-1}(f(x))\right] \le \mu(n) \qquad \forall n \in \mathbb{N},$$

where the probability is taken over $x \leftarrow \{0,1\}^n$ and the coins of A.

One-way functions are known to be necessary and sufficient for many cryptographic tasks, such as private-key encryption, digital signatures, pseudorandom generation, and bit commitment [GGM86, HILL99, IL89, Nao91, Rom90].

A formal statement of the result of Ostrovsky that we will prove in this section follows.

Theorem 4.8.4 ([Ost91]) *If there is a hard-on-average promise problem in* **HVSZK**, *then one-way functions exist.*

Our proof will make use of the following result of Goldreich [Gol90]:

Proposition 4.8.5 ([Gol90]) *Suppose there exist two samplable ensembles of distributions, $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$, such that*

1. *$\{X_n\}$ and $\{Y_n\}$ are computationally indistinguishable.*
2. *There is a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for all n, $\mathrm{StatDiff}(X_n, Y_n) \geq 1/p(n)$.*

Then one-way functions exist.

Proof of Theorem 4.8.4: Suppose Π is a hard-on-average problem in **HVSZK**, and let $\{D_n\}$ be the ensemble of distributions under which Π is hard. By the Completeness Theorem (Theorem 3.5.1) and the Polarization Lemma (Lemma 3.1.12), there is a polynomial-time computable function that maps instances x of Π to pairs $(X(x), Y(x))$ of distributions such that

1. $x \in \Pi_Y \Rightarrow \mathrm{StatDiff}(X(x), Y(x)) \geq 1/2$.
2. $x \in \Pi_N \Rightarrow \mathrm{StatDiff}(X(x), Y(x)) \leq \mathrm{neg}(|x|)$,

where here and throughout this proof, we write $\mathrm{neg}(n)$ to denote negligible functions.

We will show that the following ensembles $\{X_n\}$ and $\{Y_n\}$ meet the requirements of Proposition 4.8.5:

X_n: Sample x according to D_n. Sample z from $X(x)$. Output (x, z).

Y_n: Sample x according to D_n. Sample z from $Y(x)$. Output (x, z).

The statistical farness of these ensembles will follow from the farness of $X(x)$ and $Y(x)$ on YES instances. The computational indistinguishability will follow from the statistical closeness of $X(x)$ and $Y(x)$ on NO instances, together with the fact that it is hard to distinguish YES instances of Π from NO instances.

To formalize this intuition, we make some observations which follow from the hypothesis that Π is hard-on-average:

1. $\Pr[D_n \notin \Pi_Y \cup \Pi_N] = \mathrm{neg}(n)$.
2. $\left|\Pr[D_n \in \Pi_Y] - \frac{1}{2}\right| = \mathrm{neg}(n)$ and $\left|\Pr[D_n \in \Pi_Y] - \frac{1}{2}\right| = \mathrm{neg}(n)$.

3. The ensembles $\{D_n^Y\}_{n\in\mathbb{N}}$ and $\{D_n^N\}_{n\in\mathbb{N}}$ obtained by conditioning D_n on being a YES or NO instance, respectively, are computationally indistinguishable.

Items 1 and 2 hold because otherwise the trivial algorithm that always outputs YES or the one that always outputs NO would decide Π correctly with nonnegligible advantage. Item 3 holds because a distinguisher between $\{D_n^Y\}$ and $\{D_n^N\}$ could be used to decide Π with nonnegligible advantage.

Claim 4.8.6 $\mathrm{StatDiff}\,(X_n, Y_n) \geq 1/4 - \mathrm{neg}(n)$.

> **Proof of claim:** Since D_n must produce a YES instance of Π with probability at least $1/2 - \mathrm{neg}(n)$, $\mathrm{StatDiff}\,(X_n, Y_n) \geq (1/2 - \mathrm{neg}(n)) \cdot (1/2) = 1/4 - \mathrm{neg}(n)$. □

Claim 4.8.7 *$\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable.*

> **Proof of claim:** Let A be any probabilistic polynomial-time algorithm. From the fact that $X(x)$ and $Y(x)$ are statistically close for NO instances, it follows that
>
> $$|\Pr[A(x, X(x)) = 1 | x \in \Pi_N] - \Pr[A(x, Y(x)) = 1 | x \in \Pi_N]| = \mathrm{neg}(n), \quad (4.1)$$
>
> where these probabilities (and all those to follow) are taken over $x \leftarrow D_n$ and the coins of all algorithms (A, X, and Y). By the computational indistinguishability of $\{D_n^Y\}$ and $\{D_n^N\}$, we also have
>
> $$\begin{aligned} |\Pr[A(x, X(x)) = 1 | x \in \Pi_Y] - \Pr[A(x, X(x)) = 1 | x \in \Pi_N]| &= \mathrm{neg}(n) \\ |\Pr[A(x, Y(x)) = 1 | x \in \Pi_Y] - \Pr[A(x, Y(x)) = 1 | x \in \Pi_N]| &= \mathrm{neg}(n). \end{aligned}$$
>
> Combining these with Equation 4.1, we see that all four conditional probabilities differ only by negligible amounts. Therefore,
>
> $$\begin{aligned} &\Pr[A(x, X(x)) = 1] - \Pr[A(x, Y(x)) = 1] \\ &\quad\leq |\Pr[A(x, X(x)) = 1 | x \in \Pi_Y] - \Pr[A(x, Y(x)) = 1 | x \in \Pi_Y]| \\ &\quad\quad + |\Pr[A(x, X(x)) = 1 | x \in \Pi_N] - \Pr[A(x, Y(x)) = 1 | x \in \Pi_N]| \\ &\quad\quad + 2\Pr[x \notin \Pi_Y \cup \Pi_N] \\ &\quad= \mathrm{neg}(n). \end{aligned}$$
>
> This establishes the computational indistinguishability of $\{X_n\}$ and $\{Y_n\}$. □

Given these claims, the result now follows from Proposition 4.8.5. ■

Essentially the same proof applies to public-coin computational zero-knowledge proofs via Proposition 4.7.3.

Theorem 4.8.8 ([OW93] for public-coin proofs) *If a hard-on-average promise problem possesses a public-coin* **HVCZK** *proof system, then one-way functions exist.*

Proof: The one point in the proof of Theorem 4.8.4 where we used the statistical closeness of $X(x)$ and $Y(x)$ for $x \in \Pi_Y$ instances was Equation 4.1; it is clear that computational indistinguishability would actually suffice. Thus, if we replace the ensembles $\{X(x)\}$ and $\{Y(x)\}$ with the ones given by Proposition 4.7.3 (setting $k = |x|$), the proof will still work. (YES and NO instances play the opposite role, but that is okay.) ■

In both the theorems of Ostrovsky and Ostrovsky–Wigderson, one can relax the average-case assumption to a worst-case assumption at the price of a weaker conclusion. Specifically, if one only assumes that **HVSZK** or **HVCZK** contains a problem outside of **BPP**, then one can show the existence of a weak form of one-way functions, which are given an extra auxiliary input and are hard to invert only for infinitely many values of the auxiliary input. Such versions of Theorems 4.8.4 and 4.8.8 can also be proven using our techniques. In addition, these theorems also have uniform versions, in which the hard-on-average problems, one-way functions, and computational zero-knowledge are all with respect to uniform adversaries. Our proofs work essentially unchanged in that setting.

It would be interesting to obtain a simpler proof of the full version of the Ostrovsky–Wigderson theorem (*i.e.,* with no restriction to public coins) using our techniques. One approach would be to make use of the simulator analysis for private-coin proofs given in Section 3.3. Indeed, one can use that simulator analysis to show that if **HVSZK** contains a hard-on-average problem, then a "false entropy generator" (in the sense of [HILL99]) exists, which in turn implies the existence of one-way functions by [HILL99]. This gives yet another proof of Ostrovsky's theorem for statistical zero knowledge. Unfortunately, as discussed in Section 4.7, that simulator analysis appears to yield something trivial for computational zero knowledge.

Open Problem 4.8.9 *Can one refine the private-coin simulator analysis of Section 3.3 and use it to give a simpler proof of the full Ostrovsky–Wigderson [OW93] theorem?*

The Ostrovsky–Wigderson theorem also has a converse. If (nonuniformly) one-way functions exist, then it is known that **HVCZK** = **PSPACE** [GMW91, IY87, BGG+88, LFKN92, Sha92] and it can be shown that **PSPACE** contains hard-on-average promise problems. However, no such converse is known for Ostrovsky's theorem.

Open Problem 4.8.10 *Does the existence of one-way functions (or some other general intractability assumption) imply that* **HVSZK** *contains a hard-on-average problem? or even just that* **HVSZK** $\neq$ **BPP***?*

A positive answer to this question would show that the complexity of statistical zero knowledge is intimately tied with the feasibility of complexity-based cryptography [IL89].

Chapter 5

Private Coins vs. Public Coins

5.1 Motivation and results

In the two interactive proofs we have seen so far — the ones for Statistical Difference and Graph Nonisomorphism (Protocols 2.1.2 and 3.1.19) — it is essential that the verifier keeps its random coins hidden from the prover. From such examples, one might guess that allowing such deceptiveness on the verifier's part makes interactive proofs strictly more powerful. Surprisingly, this conjecture is false. Goldwasser and Sipser [GS89] showed that every interactive proof can be transformed into a public-coin one.[1] In this chapter, we will prove an analogous result for statistical zero-knowledge, originally due to Okamoto [Oka96]:

Theorem 5.1.1 ([Oka96]) *Every problem in* **HVSZK** *possesses a public-coin honest-verifier statistical zero-knowledge proof.*

Although Theorem 5.1.1 played a central role in later work, Okamoto's proof of it in [Oka96] was very complicated and understood by very few researchers. The proof we give here is much simpler. This simplification stems from the Completeness Theorem and Corollary 4.1.1 in particular, which says that every problem in **HVSZK** has a *2-message* **HVSZK** proof. Thus, to obtain our result, we need only give a transformation that applies to 2-message proof systems. This is a much simpler special case of Okamoto's transformation, and enables us to use Okamoto's innovative techniques in a clean form, unhampered by the complications arising from many rounds of interaction.

Transformations from private coins to public coins, like the one given by Theorem 5.1.1, are very useful as public-coin proofs are much easier to analyze and manipulate than general private-coin proofs. Indeed, the result of Goldwasser and Sipser for interactive proofs found many applications (*e.g.,* [BHZ87, FGM$^+$89, BGG$^+$88]), and the same is true for statistical zero knowledge. For example, we have already seen that the simulator analysis for public-coin proofs, given in Section 3.2, is much simpler than the simulator analysis for general private-coin proofs, given in Section 3.3. We will see another example in the following chapter: our transformation from honest-verifier zero-knowledge proofs to ones

[1] Recall that public-coin (a.k.a. Arthur–Merlin) proofs [BM88] are interactive proofs in which the verifier's messages consist solely of random coin flips, and the only computation the verifier does is to decide whether to accept or reject at the end of the interaction.

robust against cheating verifiers will only be given for public-coin proofs, and hence we will rely on Theorem 5.1.1 to deduce that **HVSZK** = **SZK**. Moreover, the closure of **HVSZK** under complement and the completeness of Statistical Difference (together with all its consequences) were both originally proven in [Oka96, SV97] using Theorem 5.1.1 as a starting point.

We make one additional modification to Okamoto's transformation (described later), which enables us to prove that the transformation also works for certain *computational* zero-knowledge proofs.

Theorem 5.1.2 *Every promise problem possessing a 3-message honest-verifier computational zero-knowledge proof also possesses a public-coin honest-verifier computational zero-knowledge proof.*

We view this as a first step towards exhibiting a general (*i.e.,* with no restriction on the message complexity) transformation from private coins to public coins for computational zero knowledge.[2]

An alternative approach to proving Theorem 5.1.1 would be to exhibit a public-coin statistical zero-knowledge proof for one of the complete problems. That is the approach we took in [GV99] and it gives the most direct proof of Okamoto's theorem, since all one needs is the reduction from **HVSZK** to Entropy Difference from Section 3.3 and a public-coin proof system for ED. However, given that we have already proven the Completeness Theorem, the transformation given in this chapter is not much more complex than the proof system for ED, and has the advantage of also applying to computational zero knowledge.

Organization. We begin by giving an overview of the transformation from 2-message zero-knowledge proofs to public-coin zero-knowledge proofs in Section 5.2. Like we did when reducing Entropy Difference to Statistical Difference in Section 3.4, for motivation we start by treating a special case of 2-message proof systems in which the verifier's message distribution is flat.[3] For this special case, we describe why the Goldwasser–Sipser transformation fails to be preserve zero knowledge and describe Okamoto's method for overcoming this problem. We conclude the overview by discussion of the ideas underlying the extension of this special case to the general one. In particular, two subprotocols due to Okamoto [Oka96] are crucial in treating the general case. In Section 5.3, we state the properties of these subprotocols that we need in the transformation. In Section 5.4, we give the transformation from private coins to public coins systems and prove Theorems 5.1.1 and 5.1.2, assuming the existence of Okamoto's subprotocols. Section 5.5 contains a self-contained presentation of the two subprotocols and their proofs of correctness.

[2]As usual, it is only interesting to exhibit such a transformation for computational zero knowledge *unconditionally*, for if one assumes that (nonuniformly) one-way functions exist, public-coin computational zero-knowledge proofs can be constructed for all of **IP** = **PSPACE** "from scratch" [GMW91, IY87, BGG$^+$88, LFKN92, Sha92].

[3]Recall that a *flat* distribution is one that is uniform over a subset of its range.

5.2 Overview

We begin with an exposition of the standard protocol for proving lower bounds on set sizes, which is the starting point for the Goldwasser–Sipser proof system. We stress that all protocols described in this section are public-coin protocols.

5.2.1 The standard lower bound protocol

Suppose T is some subset of $\{0,1\}^n$ and a prover M ("Merlin") wants to convince a verifier A ("Arthur") that $|T| \gg 2^m$. Assuming A has oracle access to a procedure which tests membership in T, Protocol 5.2.1 gives a way to accomplish this task using 2-universal hash functions.[4] This public-coin protocol was first described in [Bab85, GS89] and orginates with a lemma of Sipser [Sip83].

Protocol 5.2.1: Lower bound protocol (M, A)

Input: Integers m and n (in unary) and a membership oracle for $T \subset \{0,1\}^n$

1. A: Select h uniformly from $\mathcal{H}_{n,m}$ and send h to M.
2. M: Select y uniformly from $T \cap h^{-1}(0)$ (if this intersection is nonempty) and send y to A.[a] If the intersection is empty, send `fail` to A.
3. A: If both $h(y) = 0$ and $y \in T$, accept. Otherwise, reject.

[a] Here 0 is a canonically fixed element of $\{0,1\}^m$.

The best analysis of the Protocol 5.2.1 was provided in [AH91]:

Lemma 5.2.2 *Protocol 5.2.1 has the following properties:*

1. *(Completeness) If* $|T| \geq 2^k \cdot 2^m$*, then* A *accepts with probability at least* $1 - 2^{-k}$*.*
2. *(Soundness) If* $|T| \leq 2^{-k} \cdot 2^m$*, then no matter what strategy* M *uses,* A *accepts with probability at most* 2^{-k}*.*

In fact, this protocol also has a sort of statistical zero-knowledge property. The property holds with respect to the inputs n and m, provided that $|T| \gg 2^m$ and that one is given a uniformly selected element of T.

Lemma 5.2.3 (implicit in [Oka96]) *Let* $\mathcal{H}$ *be a 2-universal family of hash functions mapping a domain* $\mathcal{D}$ *to a range* $\mathcal{R}$*, and let* 0 *be any fixed element of* $\mathcal{R}$*. Let* T *be a subset of* $\mathcal{D}$ *such that* $|\mathcal{R}| \leq \varepsilon \cdot |T|$*. Then the following two distributions have statistical difference* $\varepsilon^{\Omega(1)}$*:*

[4] Recall that for every pair of integers k and ℓ, $\mathcal{H}_{k,\ell}$ denotes a family of 2-universal hash functions mapping $\{0,1\}^k$ to $\{0,1\}^\ell$ (affine-linear functions over GF(2)).

(A) *Choose h uniformly in $\mathcal{H}$, and y uniformly in $T \cap h^{-1}(0)$. Output (h, y).*[5]

(B) *Choose y uniformly in T, and h uniformly in $\{h' \in \mathcal{H} : h'(y) = 0\}$.*[6] *Output (h, y).*

Think of $\mathcal{D} = \{0,1\}^n$, $\mathcal{R} = \{0,1\}^m$, and $\epsilon = 2^m/|T|$. Then, Distribution (A) corresponds to A's view of the execution of the protocol and Distribution (B) provides a simulation with deviation (at most) $(2^m/|T|)^{\Omega(1)}$ for it.

5.2.2 The simplifying assumptions — flat distributions

Let (P, V) be a 2-message interactive proof system for a promise problem Π which is statistical zero knowledge for the honest verifier. We aim to construct a public-coin proof system (M, A) for Π. Without loss of generality, we may assume that, in (P, V), V sends its message first, since any verifier messages after the last prover message are irrelevant in an interactive proof. We write $V_{x,k}$ to denote the distribution of V's message on input x and security parameter k. The two simplifying assumptions we make about (P, V) are

1. The protocol has perfect completeness and soundness error 2^{-4k}.

2. $V_{x,k}$ is a flat distribution.

5.2.3 The Goldwasser–Sipser transformation for flat distributions

With these assumptions, we now describe the Goldwasser–Sipser transformation from 2-message proof systems to public-coin proof systems. Let $v = \mathrm{H}(V_{x,k})$, so that, by flatness, $|\mathrm{Supp}(V_{x,k})| = 2^v$. On YES instances x, the perfect completeness guarantees that for all verifier messages $y \in \mathrm{Supp}(V_{x,k})$, the specified prover response $P((x, 1^k), y)$ makes V accept with probability 1 over V's random coins (conditioned on y). The soundness error 2^{-4k} provides a strong negation of this for NO instances — for all but a 2^{-2k} fraction of the y's in $\mathrm{Supp}(V_{x,k})$, V accepts with probability at most 2^{-2k} conditioned on y, no matter what the prover response is. Thus, an idea for converting such a proof system into a public-coin one would be to use a lower bound protocol to show that there are "many" (*i.e.,* 2^v) y's for which V's marginal acceptance probability is high. But the last step of the lower bound protocol requires A to test membership in the set; that is, A must test that for the y given, V's marginal acceptance probability is high. It does not seem possible for A to accomplish this on his own.

To see how Goldwasser and Sipser overcome this obstacle, note that conditioned on y, V's random coins r are distributed uniformly in the set

$$\Omega(y) \stackrel{\text{def}}{=} \{r : V_{x,k}(r) = y\}.$$

Thus, V's marginal acceptance probability given y and a prover response z is exactly the fraction of $r \in \Omega(y)$ for which $V(x, y, z; r) = \texttt{accept}$. This fraction can be proven to be

[5] In case $T \cap h^{-1}(0) = \emptyset$ the output is defined to be a special failure symbol.

[6] Note that this task — choosing a hash function uniformly among those that map a given point to 0 — can be easily done in polynomial time for our particular hash families $\mathcal{H}_{n,m}$.

large via another lower bound protocol! Thus, the Goldwasser–Sipser transformation (for two-message proof systems) consists of two lower bound protocols; together, these show that there are "many" y's for which there are "many" r's making the verifier accept. For the formal description of the protocol, let m be the number of random coins used by V and let n be the length of V's messages. By flatness, $|\Omega(y)| = 2^{m-v}$ for any $y \in \mathrm{Supp}(V_{x,k})$, so 2^{m-v} is the set size for which the second lower bound should be proven. Actually, to guarantee a small completeness error via Lemma 5.2.1, we need some slackness in the set sizes for which the lower bound protocol is executed, so we use set sizes 2^{v-k} and 2^{m-v-k} rather than 2^v and 2^{m-v}. The resulting proof system is Protocol 5.2.4.

Protocol 5.2.4: Goldwasser–Sipser [GS89] transformed protocol (M, A) for flat distributions

Input: Instance x of Π and a security parameter k (in unary)

1. M: Calculate $v = \mathrm{H}(V_{x,k})$. Send v to A.
2. A: Choose h_1 uniformly from $\mathcal{H}_{n,v-k}$. Send h_1 to M.
3. M: Choose y uniformly in $\mathrm{Supp}(V_{x,k}) \cap h_1^{-1}(0)$. Send y to A.
4. A: Check that $h_1(y) = 0$. If not, reject immediately.
5. M: Let $z \leftarrow P(x, 1^k, y)$. Send z to A.
6. A: Choose h_2 uniformly from $\mathcal{H}_{m,m-v-k}$. Send h_2 to M.
7. M: Choose r uniformly in $\Omega(y) \cap h_2^{-1}(0)$. Send r to A.
8. A: Check that $V_{x,k}(r) = y$, $h_2(r) = 0$, and $V(x, 1^k, y, z; r) = \mathtt{accept}$. Accept if all three conditions hold and reject otherwise.

We now show that Protocol 5.2.4 is complete. Fix a YES instance x and a security parameter k. By the completeness of the lower bound protocol, M will succeed to find a $y \in \mathrm{Supp}(V_{x,k})$ (respectively, an $r \in \Omega(y)$) satisfying the appropriate hashing condition with probability at least $1 - 2^{-k}$ in each of the two lower bound protocols. By the perfect completeness of (P, V), the condition that $V(x, 1^k, y, z; r)$ will always be satisfied as long as $r \in \Omega(y)$. Thus, (M, A) has completeness error at most $2 \cdot 2^{-k}$.

The soundness of this proof system can be deduced from the soundness of both (P, V) and the lower bound protocol as follows: Fix a NO instance x and a security parameter k. Consider the optimal prover strategy M^*, which we may assume is deterministic without loss of generality. Let v^* be M^*'s first message. For any y and z, define

$$R_{y,z} \stackrel{\text{def}}{=} \{r \in \Omega(y) \ : \ V(x, 1^k, y, z; r) = \mathtt{accept}\}.$$

Also set

$$T = \{y \ : \ \exists\, z \ \ |R_{y,z}| > 2^{m-v^*-2k}\}.$$

From the soundness of (P, V), it follows that $|T| \leq 2^{v^*-2k}$ (for otherwise, there would exist a P^* which makes V accept with probability greater than $(2^{v^*-2k} \cdot 2^{m-v^*-2k})/2^m = 2^{-4k}$). By the soundness of the lower bound protocol, M^* will be able to select a $y \in T$ in Step 3 with probability at most $|T|/2^{v^*-k} \leq 2^{-k}$. Also by the soundness of the lower bound protocol, given any $y \notin T$ and any z, the probability that M^* will be able to select an r (in Step 7) that will make A accept is at most $|R_{y,z}|/2^{m-v^*-k} \leq 2^{-k}$. Thus, the total soundness error is at most $2 \cdot 2^{-k}$.

5.2.4 Preserving zero knowledge for flat distributions

Since (M, A) consists essentially of two lower bound protocols, Lemma 5.2.3 suggests that (M, A) might satisfy some sort of zero-knowledge property. That lemma implies that when both parties follow the protocol, y is distributed almost uniformly in $\mathrm{Supp}(V_{x,k})$, and given y and z, r is distributed almost uniformly in $\Omega(y)$. Thus the distribution of (y, z, r) in (M, A) is statistically close to the interaction between P and V (the statistical difference is $2^{-\Omega(k)}$). This suggests a way to simulate the (M, A) proof system: Run the simulator for (P, V) to obtain a transcript $(y, z; r)$ and then uniformly choose hash functions h_1 and h_2 subject to the conditions that $h_1(y) = 0$ and $h_2(r) = 0$. However, to select these hash functions, one needs to know v, the entropy of $V_{x,k}$. This appears difficult to compute in polynomial time; indeed, that is why we have the prover calculate it and send it to the verifier in the first message. This is essentially the *only* reason that (M, A) can fail to be zero-knowledge even when (P, V) is.

To get around this difficulty, Okamoto [Oka96] uses a technique which he calls "complementary usage of messages" (which we also used in the reduction from ED to SD). In Protocol 5.2.4, M proves two lower bound; one for set size 2^{v-k} (for the set of "good" y's), the other for set size 2^{m-v-k} (for the set of "good" r's). We aim to give a method by which M can prove such lower bounds *without revealing* v. We begin with the second lower bound (for set size 2^{m-v-k}). Recall that $|\mathrm{Supp}(V_{x,k})| = 2^v$. So, proving that some set T is of size at least 2^{m-v-k} is *equivalent* to proving that $T \times \mathrm{Supp}(V_{x,k})$ is of size at least 2^{m-k}. Note that v has disappeared, and all that is left is m (the number of coins that V uses) and k (the security parameter), which are trivial to compute! Thus, in the second lower bound in Protocol 5.2.4, we can replace $\Omega(y)$ with $\Omega(y) \times \mathrm{Supp}(V_{x,k})$ and have h_2 map to $m - k$ bits instead of $m - v - k$.

The first lower bound (for set size 2^{v-k}) is only slightly trickier. Recall that for any $y' \in T$, $\Omega(y') = 2^{m-v}$. Thus, proving a lower bound of 2^{v-k} on the size of a set T is equivalent to proving a lower bound of 2^{m-k} on the size of $T \times \Omega(y')$, for some $y' \in \mathrm{Supp}(V_{x,k})$. Note that in order to implement this idea, some y' must be fixed in advance; we can simply have M choose one at random and send it to A prior to the lower bound protocol. Incorporating these ideas into Protocol 5.2.4, we obtain Protocol 5.2.5, which gives a public-coin zero-knowledge proof system, assuming that $V_{x,k}$ is flat.

The last two steps of Protocol 5.2.5 are for M to prove that y'' is in fact in the support of $V_{x,k}$. The completeness and soundness errors of this protocol can be shown to be $2 \cdot 2^{-k}$ in the same manner done for Protocol 5.2.4, together with our observations above. In addition,

Protocol 5.2.5: Zero-knowledge transformed protocol (M, A) for flat distributions

Input: Instance x of Π and a security parameter k (in unary)

1. M: Select $y' \leftarrow V_{x,k}$. Send y' to A.
2. A: Choose h_1 uniformly from $\mathcal{H}_{n+m,m-k}$. Send h_1 to M.
3. M: Choose (y, r') uniformly in $(\mathrm{Supp}(V_{x,k}) \times \Omega(y')) \cap h_1^{-1}(0)$. Send (y, r') to A.
4. A: Check that $h_1(y, r') = 0$ and $V_{x,k}(r') = y'$. If either does not hold, reject immediately.
5. M: Let $z \leftarrow P(x, 1^k, y)$. Send z to A.
6. A: Choose h_2 uniformly from $\mathcal{H}_{m+n,m-k}$. Send h_2 to M.
7. M: Choose (r, y'') uniformly in $(\Omega(y) \times \mathrm{Supp}(V_{x,k})) \cap h^{-1}(0)$. Send (r, y'') to A.
8. A: Check that $V_{x,k}(r) = y$, $h_2(r, y'') = 0$, and $V(x, 1^k, y, z; r) = \texttt{accept}$. If any of these conditions does not hold, reject immediately.
9. M: Choose r'' uniformly in $\Omega(y'')$. Send r'' to A.
10. A: Check that $V_{x,k}(r'') = y''$ and accept if this holds and reject otherwise.

having eliminated the use of $v = \mathrm{H}(V_{x,k})$ in the protocol, we can now argue that the proof system is statistical zero-knowledge, assuming that (P, V) is. Consider the simulator given in Algorithm 5.2.6.

Algorithm 5.2.6: Simulator for Protocol 5.2.5

Input: Instance x of Π and a security parameter k (in unary)

1. Run the simulator for (P, V) on x to obtain a transcript $(y, z; r)$.
2. Choose r' and r'' uniformly from $\{0,1\}^m$. Let $y' = V_{x,k}(r')$ and $y'' = V_{x,k}(r'')$.
3. Choose h_1 uniformly from $\{h \in \mathcal{H}_{n+m,m-k} \ : \ h(y, r') = 0\}$.
4. Choose h_2 uniformly from $\{h \in \mathcal{H}_{m+n,m-k} \ : \ h(r, y'') = 0\}$.
5. Output $(y', h_1, (y, r'), z, h_2, (r, y''), r'')$.[a]

[a] In an honest-verifier public-coin proof, the verifier's coins need not be separately simulated since they are the same as the verifier's messages.

The deviation of this simulator can be analyzed as follows: First assume that the simulator for (P, V) is actually a perfect simulator, *i.e.,* has deviation 0; using a statistical zero-knowledge simulator would only increases the deviation by a negligible amount. Now, the distribution of y' is the same (uniform in $\mathrm{Supp}(V_{x,k})$) in both (M, A) and the simulator, so we analyze both distributions conditioned on any fixed y'. In (M, A), h_1 is chosen uniformly from $\mathcal{H}_{n+m,m-k}$, and then (y, r') is chosen uniformly from $(\mathrm{Supp}(V_{x,k}) \times \Omega(y')) \cap h^{-1}(0)$. In the simulator, (y, r') is distributed uniformly in $\mathrm{Supp}(V_{x,k}) \times \Omega(y')$ and h_1 is chosen uniformly subject to $h_1(y, r') = 0$. These correspond to distributions (A) and (B) in Lemma 5.2.3, respectively. By that lemma, these two distributions have statistical difference at most $\left(2^{m-k}/|\mathrm{Supp}(V_{x,k}) \times \Omega(y')|\right)^{\Omega(1)} = (2^{-k})^{\Omega(1)}$. So, now fix any $(y', h_1, (y, r'))$ (such that $h_1(y, r') = 0$ and $V_{x,k}(r') = y'$) and let us analyze the remaining components conditioned on those. In both (M, A) and the simulator, z is chosen according to P's strategy, so it does not increase the statistical difference. Another application of Lemma 5.2.3, with respect to the set $\Omega(y) \times \mathrm{Supp}(V_{x,k})$, shows that the components $(h_2, (r, y''))$ increase the statistical difference by at most $2^{-\Omega(k)}$. Finally, r'' is distributed uniformly in $\Omega(y'')$ in both (M, A) and the simulator. Thus, the total simulator deviation is at most $2^{-\Omega(k)}$ plus the deviation of the simulator for (P, V).

Remark 5.2.7 Okamoto [Oka96] also treats the special case of 2-message proof systems in which the verifier message distribution is flat for motivation. Protocol 5.2.5 (and its generalization to non-flat distributions below) differ from Okamoto's protocols in one respect. Okamoto uses the *simulated verifer* (as defined by the simulator for (P, V)) instead of the real verifier V in constructing the proof system (M, A). Because of this, in Okamoto's transformation, the fact that the simulated verifier is statistically close to the real verifier is used in proving the completeness of (M, A); hence, the transformation is restricted to *statistical* zero knowledge proofs. In our case, the simulator for (P, V) is only used in constructing the simulator for (M, A); this enables us to prove that the transformation also works for computational zero knowledge. On the other hand, the fact that Okamoto uses the simulator S rather than the verifier V in constructing (M, A) appears to be crucial in his transformation for *many-message* proof systems (which we have avoided via Corollary 4.1.1). To extend the result to *3-message* computational zero-knowledge proofs, we simply note that an extra prover message at the start does not harm the analysis.

5.2.5 Removing the assumptions — general distributions

There are several problems in generalizing the transformation of Protocol 5.2.5 to arbitrary two-message zero-knowledge proofs (P, V). The assumption about the completeness and soundness errors is not very problematic. Essentially the same analysis as given above works even when the proof system does not have perfect completeness, but completeness error, say, 2^{-k}. And exponentially small completeness and soundness errors can be achieved by straightforward parallel repetitions.

The assumption that the verifier message distribution $V_{x,k}$ is flat presents more serious difficulties. Recall the Flattening Lemma (Lemma 3.4.6), which says that if we take many independent copies of a distribution, the distribution gets "flattened" in the sense that, with high probability, a random sample from the distribution X will have probability mass within a factor of $2^{O(\Delta)}$ of $2^{-\mathrm{H}(X)}$, where Δ grows sublinearly with the number of copies taken. Note that taking parallel repetitions of the proof system has exactly the effect of replacing the message distribution $V_{x,k}$ with many independent copies of itself. A key point is that the soundness error decreases like a true exponential $2^{-\Omega(t)}$ with the number t of parallel repetitions. Thus, with sufficiently many parallel repetitions, we can make the deviation Δ from flatness small relative to the soundness error, in the sense that the extra slackness factors of $2^{O(\Delta)}$ needed in the lower bound protocols to deal with the deviation from flatness will not affect the soundness of the resulting proof system (M, A).

Unfortunately, the protocol still needs further modifications to work with "nearly flat" rather than truly flat distributions. The problems arise from the fact that, although Δ-flatness guarantees that, with high probability, a *random* sample will have a nearly typical probability mass, some very heavy and very light samples can still exist. So, M may select y' to be "too heavy", allowing him many choices for r' and leading to a violation of the soundness requirement. Similarly, although there are only about $2^{\mathrm{H}(V_{x,k})}$ choices for y'' that have probability mass near $2^{-\mathrm{H}(V_{x,k})}$, if $V_{x,k}$ is only nearly flat there may be many more choices for y'' (alas, these are "too light" — *i.e.,* have probability mass much smaller than $2^{-\mathrm{H}(V_{x,k})}$). This gives M too much freedom (in the choice of y'') and may also lead to violation of the soundness requirement.

In order to solve these problems, Okamoto [Oka96] introduces two subprotocols: The first is a "sample generation" protocol, which is a protocol for M and A to select a sample from a nearly flat distribution such that no matter what strategy M uses, the sample will not be too heavy. This will replace Step 1 in Protocol 5.2.5, and guarantee that M does not have too much freedom in its choice of r' (in Step 3). The second protocol is a "sample test" protocol, which is a way for M to prove that a sample y'' taken from a nearly flat distribution is not too light. This will replace Steps 9 and 10 in Protocol 5.2.5 and guarantee that M does not have too much freedom in its choice of y'' (in Step 7).

We stress that both of these subprotocols will be public-coin and will possess appropriate simulability properties to ensure that the resulting protocol for Π is a public-coin **HVSZK** proof system. Below, we will specify the properties of these subprotocols, and formulate and analyze the transformed proof system assuming that these subprotocols exist. In Section 5.5, we present these subprotocols and prove that they have the asserted properties.

5.3 Subprotocol specifications

Below, all distributions are given in the form of a circuit which generate them. The input to these protocols will consist of a distribution, denoted X. We will denote by m (resp., n) the length of the input to (resp., output of) the circuit generating the distribution X. In order to define the notion of a sample generation protocol, we must formalize what it means for an interactive protocol to have output.

Definition 5.3.1 *Let f be any (deterministic) polynomial-time computable function and let (A, B) be an interactive protocol. The f-*output *of (A, B) on input x is the random variable obtained by applying f to x and all the messages exchanged between A and B (but not to the random coins of A and B).*

Usually, for any given protocol, we will only be interested in one particular output function f (given at the same time as the protocol), so we will usually omit f from the notation when referring to the protocol.

Definition 5.3.2 (sample generation protocol) *A protocol (M, A) is called a* sample generation protocol *if on common input a distribution X and parameters Δ, t, such that X is Δ-flat and $t \leq \Delta$,*[7] *the following holds:*

1. *(Efficiency) (M, A) is polynomially bounded and A is polynomial-time computable.*
2. *("Completeness") If both parties are honest, then the output of the protocol will be $t \cdot \Delta$-typical with probability at least $1 - m \cdot 2^{-\Omega(t^2)}$.*
3. *("Soundness") If A is honest then, no matter how M plays, the output will be $2\sqrt{t\Delta} \cdot \Delta$-heavy with probability at most $m \cdot 2^{-\Omega(t^2)}$.*
4. *(Strong "Zero Knowledge") There exists a probabilistic polynomial-time* simulator *S so that for every (X, Δ, t) as above, the following two distributions have statistical difference at most $m \cdot 2^{-\Omega(t^2)}$:*

[7] The condition $t \leq \Delta$ is to simplify the error expressions and will always be satisfied in our applications. Moreover, the particular error expressions we give are artifacts of our construction and a protocol achieving slightly different expressions might suffice. What is important is that the error probabilities $(m \cdot 2^{-\Omega(t^2)})$ are negligible as a function of t and that the heaviness expression $2\sqrt{t\Delta} \cdot \Delta$ is subquadratic in Δ.

(A) *Execute (M, A) on common input (X, Δ, t) and output the view of A, appended by the output.*

(B) *Choose $x \leftarrow X$ and output $(S(X, \Delta, t, x), x)$.*

A sample generation protocol is said to be public coin *if it is public coin for A.*

The above zero-knowledge property is referred to as *strong* since the simulator cannot produce a view-output pair by first generating the view and then computing the corresponding output. Instead, the simulator is forced (by the explicit inclusion of x in Distribution (B)) to generate a consistent random view for a given random output (of the protocol). We comment that the trivial protocol in which A uniformly selects an input r to the circuit X and reveals both r and the output $x = X(r)$ cannot be used since the simulator is only given x and it may be difficult to find an r yielding x in general. Still, a sample generation protocol is implicit in Okamoto's work [Oka96] (where it is called a "pre-test").

Theorem 5.3.3 (implicit in [Oka96]) *There exists a public-coin sample generation protocol. Furthermore, the number of messages exchanged in the protocol is linear in m.*

A proof of Theorem 5.3.3 is presented in Section 5.5.

Definition 5.3.4 (sample test protocol) *A protocol (M, A) is called a* sample test protocol *if on common input a distribution X, a string $x \in \{0,1\}^n$ and parameters Δ, t, such that X is Δ-flat and $t \leq \Delta$, the following holds:*

1. *(Efficiency) (M, A) is polynomially bounded and A is polynomial-time computable.*
2. *("Completeness") If both parties are honest and x is $t \cdot \Delta$-typical then A accepts with probability at least $1 - m \cdot 2^{-\Omega(t^2)}$.*
3. *("Soundness") If x is $6\sqrt{t\Delta} \cdot \Delta$-light and A is honest then, no matter how M plays, A accepts with probability at most $m \cdot 2^{-\Omega(t^2)}$.*
4. *(Weak "Zero Knowledge") There exists a probabilistic polynomial-time* simulator *S so that for every (X, Δ, t) as above and for every $t \cdot \Delta$-typical x, the following two distributions have statistical difference at most $m \cdot 2^{-\Omega(t^2)}$:*

   (A) *Execute (M, A) on common input (X, x, Δ, t) and output the view of A, prepended by x.*

   (B) *Choose r uniformly in $\Omega_X(x) \stackrel{\text{def}}{=} \{r' : X(r') = x\}$, and output $(x, S(X, x, \Delta, t, r))$.*

A sample test protocol is said to be public coin *if it is public coin for A.*

The above zero-knowledge property is referred to as *weak* since the simulator gets a random r giving rise to x (*i.e.*, $x = X(r)$) as an auxiliary input (whereas A is only given x). We comment that a simple public-coin testing protocol exists in case one can approximate the size of $\Omega_X(x)$ and uniformly sample from it. However, this may not be the case in general. Still, a sample test protocol is implicit in Okamoto's work [Oka96] (where it is called a "post-test").

Theorem 5.3.5 (implicit in [Oka96]) *There exists a public-coin sample testing protocol. Furthermore, the number of messages exchanged in the protocol is linear in* m.

A proof of Theorem 5.3.5 is presented in Section 5.5.

5.4 The transformed proof system

We now present the transformation from 2-message zero-knowledge proofs to public-coin zero-knowledge proofs. (The case of 3-message computational zero-knowledge proofs is similar, but complicates the notation, so we just sketch the changes necessary at the end.) Let (P_0, V_0) be a 2-message interactive proof system which is honest-verifier zero-knowledge (either computational or statistical). Without loss of generality, we assume that on security parameter k, the completeness error is at most 2^{-k} and the soundness error is at most $1/2$. Throughout what follows, we will always assume that the security parameter k is at least the input length $|x|$; this can be achieved by artifically increasing k if necessary. Let $m_0(k) = \mathrm{poly}(k)$ be a bound on the number of random coins used by V_0 on inputs $(x, 1^k)$, when $k \geq |x|$. Let (P, V) denote the interactive proof system for Π, which does the following on input x and security parameter $k \geq |x|$: Both parties set $m_0 = m_0(k)$, and $\ell = 2^{16} \cdot m_0^6 \cdot k$; and execute $(P_0, V_0)(x, 1^k)$ ℓ times in parallel, with V accepting iff V_0 accepts in all ℓ executions.

Let $V_{x,k}$ denote the message distribution of V on input $(x, 1^k)$. Let n be the length of messages produced by this distribution, and $m = \ell \cdot m_0$ the number of random coins used to generate the distribution. We can immediately make the following observations about (P, V):

Claim 5.4.1

1. *The completeness error is at most* $\ell \cdot 2^{-k} = 2^{-\Omega(k)}$.
2. *The soundness error is at most* $2^{-\ell}$.
3. $V_{x,k}$ *is* Δ*-flat, for* $\Delta = \sqrt{\ell} \cdot m_0 = 2^8 m_0^4 \sqrt{k}$.

The last item follows from the Flattening Lemma (Lemma 3.4.6), as $V_{x,k}$ consist of ℓ independent copies of V_0's message distribution.

Protocol 5.4.2 gives the transformed proof system (M, A) obtained by generalizing Protocol 5.2.5 to "nearly flat" distributions and augmenting it with sample generation and sample test protocols. In the protocol, the parameters m_0, m, n, ℓ, and Δ have the same values as above (as functions of k), and, for any y, $\Omega(y) \stackrel{\text{def}}{=} \{r \in \{0,1\}^m : V_{x,k}(r) = y\}$.

We now prove that (M, A) is the protocol we want. Clearly, (M, A) is public-coin, assuming that the sample generation and test protocols are (as we may by Theorems 5.3.3 and 5.3.5). The completeness property will follow from the zero knowledge one, so we start by establishing soundness.

Lemma 5.4.3 (soundness) *Suppose that* $x \in \Pi_N$. *Then, for any* M^*, A *accepts in* $(M^*, A)(x, 1^k)$ *with probability at most* $\exp(-\Omega(k))$.

Protocol 5.4.2: Zero-knowledge transformed protocol (M, A)

Input: Instance x of Π and a security parameter $k \geq |x|$ (in unary)

1. M, A: Execute a sample generation protocol, with inputs $(V_{x,k}, \Delta, \sqrt{k})$, to obtain an output y'.
2. A: Choose h_1 uniformly from $\mathcal{H}_{n+m,m-3\sqrt{k}\Delta}$. Send h_1 to M.
3. M: Choose (y, r') according to $V_{x,k} \otimes \Omega(y')$,[a] conditioned on $h_1(y, r') = 0$. Send (y, r') to A.
4. A: Check that $h_1(y, r') = 0$ and $V_{x,k}(r') = y'$. If either does not hold, reject immediately.
5. M: Let $z \leftarrow P(x, 1^k, y)$. Send z to A.
6. A: Choose h_2 uniformly from $\mathcal{H}_{m+n,m-3\sqrt{k}\Delta}$. Send h_2 to M.
7. M: Choose (r, y'') according to $\Omega(y) \otimes V_{x,k}$, conditioned on $h_2(r, y'') = 0$. Send (r, y'') to A.
8. A: Check that $V_{x,k}(r) = y$, $h_2(r, y'') = 0$, and $V(x, 1^k, y, z; r) = \texttt{accept}$. If any of these conditions does not hold, reject immediately.
9. M, A: Execute a sample test protocol, with input $(V_{x,k}, y'', \Delta, \sqrt{k})$, and A accepts iff the test is concluded satisfactorily.

[a] Recall that we use the same notation for a set (*e.g.*, $\Omega(y')$) and the uniform distribution on that set.

Proof: Observe that the sample generation and test protocols are invoking with parameters $t = \sqrt{k}$ and $\Delta = 2^8 m_0^4 \sqrt{k}$, and $V_{x,k}$ is in fact Δ-flat. Thus, the soundness of the sample generation protocol implies that with probability at most $m \cdot \exp(-\Omega(t^2)) = \exp(-\Omega(k))$, the outcome y' is $2\sqrt{t\Delta} \cdot \Delta$-heavy. Thus, we have:

Claim 5.4.4 *y' is $2\sqrt{t\Delta} \cdot \Delta$-heavy with probability at most* $\exp(-\Omega(k))$.

Suppose that y' is not $2\sqrt{t\Delta} \cdot \Delta$-heavy. We will show that M^* will be forced to select a y which has very few accepting r's. As in the analysis of Protocol 5.2.4, for any y and z, let

$$R_{y,z} \stackrel{\text{def}}{=} \{r \in \Omega(y) \ : \ V(x, 1^k, y, z; r) = \texttt{accept}\},$$

and define

$$T \stackrel{\text{def}}{=} \{y \ : \ \exists\, z \ \ |R_{y,z}| > 2^{m-\mathrm{H}(V_{x,k})-7\sqrt{t\Delta}\cdot\Delta}\}.$$

Claim 5.4.5 $|T| \le 2^{\mathrm{H}(V_{x,k})-3\sqrt{t\Delta}\cdot\Delta}$.

> **Proof of claim:** Suppose not. Then there would be a prover strategy P^* which convinces V to accept with probability at least
>
> $$\frac{|T| \cdot 2^{m-\mathrm{H}(V_{x,k})-7\sqrt{t\Delta}\cdot\Delta}}{2^m} > 2^{-10\sqrt{t\Delta}\cdot\Delta}.$$
>
> However, by our setting of parameters,
>
> $$10\sqrt{t\Delta} \cdot \Delta = 10 \cdot 2^{12} \cdot m_0^6 \cdot k < \ell,$$
>
> so we have contradicted the fact that the soundness error of (P, V) is $2^{-\ell}$. □

Claim 5.4.6 *If y' is not $2\sqrt{t\Delta} \cdot \Delta$-heavy, then, with probability at least $1 - 2^{-k}$ (over Steps 2–4), $y \notin T$ (or A rejects).*

> **Proof of claim:** By the soundness of the standard lower bound protocol (Lemma 5.2.2), it suffices to show that the number of pairs (y, r') such that $V_{x,k}(r') = y'$ and $y \in T$ is at most $2^{-k} \cdot 2^{m-3\sqrt{k}\Delta}$. Since y' is not $2\sqrt{t\Delta} \cdot \Delta$-heavy, there are at most $2^{m-\mathrm{H}(V_{x,k})+2\sqrt{t\Delta}\cdot\Delta}$ values of r' such that $V_{x,k}(r') = y'$. Thus the total number of pairs (y, r') such that $V_{x,k}(r') = y'$ and $y \in T$ is at most
>
> $$2^{m-\mathrm{H}(V_{x,k})+2\sqrt{t\Delta}\cdot\Delta} \cdot 2^{\mathrm{H}(V_{x,k})-3\sqrt{t\Delta}\cdot\Delta} = 2^{m-\sqrt{t\Delta}\cdot\Delta}.$$
>
> So we need to show that $\sqrt{t\Delta} \cdot \Delta > 3\sqrt{k}\Delta + k$. This follows from our choice of parameters:
>
> $$\sqrt{t\Delta} \cdot \Delta = 2^4 m_0^2 \sqrt{k} \cdot \Delta > 3\sqrt{k} \cdot \Delta + k.$$
>
> □

Claim 5.4.7 *If $y \notin T$, then with probability at least $1 - 2^{-k}$ (over Steps 6–8), y'' is $6\sqrt{t\Delta} \cdot \Delta$-light (or A rejects).*

Proof of claim: In Step 7, M^* must choose r from $R_{y,z}$, or else A will reject. Thus, by the soundness of the lower bound protocol, it suffices to show that the number of pairs (r, y'') such that $r \in R_{y,z}$ and y'' is not $6\sqrt{t\Delta} \cdot \Delta$-light is at most $2^{-k} \cdot 2^{m-3\sqrt{k}\Delta}$. $|R_{y,z}| \leq 2^{m-\mathrm{H}(V_{x,k})-7\sqrt{t\Delta}\cdot\Delta}$, because $y \notin T$. The number non-$6\sqrt{t}\Delta \cdot \Delta$-light choices for y'' is at most $2^{\mathrm{H}(V_{x,k})+6\sqrt{t\Delta}\cdot\Delta}$ (as each such y'' has probability mass at least $2^{-\mathrm{H}(V_{x,k})-6\sqrt{t\Delta}\cdot\Delta}$ under $V_{x,k}$). Thus, the total number of pairs (r, y'') such that $r \in R_{y,z}$ and y'' is not $6\sqrt{t}\Delta \cdot \Delta$-light is at most

$$2^{m-\mathrm{H}(V_{x,k})-7\sqrt{t\Delta}\cdot\Delta} \cdot 2^{\mathrm{H}(V_{x,k})+6\sqrt{t\Delta}\cdot\Delta} = 2^{m-\sqrt{t\Delta}\cdot\Delta},$$

which is smaller than $2^{-k} \cdot 2^{m-3\sqrt{k}\Delta}$, as shown in the proof of Claim 5.4.6. □

By the soundness of the sample test protocol, we have:

Claim 5.4.8 *If y'' is $6\sqrt{t\Delta}\cdot\Delta$-light, A will reject in the sample test protocol with probability at least $1 - 2^{-\Omega(k)}$.*

Putting together all these claims, it follows that A will reject on a NO instance with probability at least $1 - 2^{-\Omega(k)}$. ■

We now show that (M, A) retains the zero-knowledge properties of (P, V). Let S be a (**HVSZK** or **HVCZK**) simulator for (P, V). The simulator for (M, A), given in Algorithm 5.4.9, is similar to Algorithm 5.2.6, but is augmented by the simulators for the sample generation and test protocols.

The correctness of this simulator will rely on the following generalization of Lemma 5.2.3 to non-flat distributions, proved in Appendix B.

Lemma 5.4.10 (implicit in [Oka96]) *Let $\mathcal{H}$ be a 2-universal family of hash functions mapping a domain $\mathcal{D}$ to a range $\mathcal{R}$ and let 0 be any fixed element of $\mathcal{R}$. Let Z be a distribution on $\mathcal{D}$ such that with probability $1-\delta$ over z selected according to Z, $\Pr[Z = z] \leq \varepsilon/|\mathcal{R}|$. Then the following two distributions have statistical difference at most $3(\delta + \varepsilon^{1/3})$:*

(A) *Choose h uniformly in $\mathcal{H}$. Select z according to Z conditioned on $h(z) = 0$. Output (h, z).*

(B) *Choose z according to Z. Select h uniformly in $\{h' \in \mathcal{H} : h'(z) = 0\}$. Output (h, z).*

We first analyze the simulator when transcript obtained from S is replaced with a true sample of $\langle P, V\rangle$.

Lemma 5.4.11 *Let $\overline{S}$ denote the output distribution of Algorithm 5.4.9, when the transcript $(y, z; r)$ obtained from $S(x, 1^k)$ is replaced with a sample of $\langle P, V\rangle(x, 1^k)$. Then, $\overline{S}$ has statistical difference at most $\exp(-\Omega(k))$ from $\langle M, A\rangle(x, 1^k)$.*

Proof: By the strong zero-knowledge property of the sample generation protocol, the pair (α, y') in an execution of $\overline{S}$ has statistical difference at most $m \cdot 2^{-\Omega(k)} = 2^{-\Omega(k)}$

Algorithm 5.4.9: Simulator for Protocol 5.4.2

Input: Instance x of Π and a security parameter k (in unary)

1. Run the simulator S for (P, V) on input $(x, 1^k)$ to obtain a transcript $(y, z; r)$.
2. Choose r' and r'' uniformly from $\{0,1\}^m$. Let $y' = V_{x,k}(r')$ and $y'' = V_{x,k}(r'')$.
3. Run the simulator for the sample generation protocol on input $(V_{x,k}, \Delta, \sqrt{k}, y')$ to obtain a transcript α corresponding to output y'.
4. Choose h_1 uniformly from $\{h \in \mathcal{H}_{n+m,m-k} \ : \ h(y, r') = 0\}$.
5. Choose h_2 uniformly from $\{h \in \mathcal{H}_{m+n,m-k} \ : \ h(r, y'') = 0\}$.
6. Simulate an execution of the sample test protocol on input $(V_{x,k}, y'', \Delta, \sqrt{k})$ and auxiliary input r'', obtaining a transcript, denoted β.
7. Output $(\alpha, h_1, (y, r'), z, h_2, (r, y''), \beta)$.

from a real execution of that protocol.[8] Since $V_{x,k}$ is Δ-flat, the string y' is $\sqrt{k}\Delta$-light with probability at most 2^{-k+1} in the simulator. Thus, we consider the distributions on $(h_1, (y, r'))$ conditioned on any pair (α, y') such that y' is not $\sqrt{k}\Delta$-light. To analyze this, we apply Lemma 5.4.10 with $Z = V_{x,k} \otimes \Omega(y')$, $\mathcal{D} = \{0,1\}^{n+m}$, and $\mathcal{R} = \{0,1\}^{m-3\sqrt{k}\Delta}$. Distribution (A) (resp., (B)) in Lemma 5.4.10 corresponds to the distribution of $(h_1, (y, r'))$ in the proof system (resp., $\overline{S}$). Since $V_{x,k}$ is Δ-flat and y' is not $\sqrt{k}\Delta$-light, the following holds with probability $\geq 1 - 2^{-k+1}$ over (y, r') selected according to $V_{x,k} \otimes \Omega(y')$:

$$\begin{aligned}
\Pr\left[V_{x,k} \otimes \Omega(y') = (y, r')\right] &= \Pr\left[V_{x,k} = y\right] \cdot \frac{1}{|\Omega(y')|} \\
&\leq 2^{-\mathrm{H}(V_{x,k})+\sqrt{k}\Delta} \cdot \frac{1}{2^{m-\mathrm{H}(V_{x,k})-\sqrt{k}\Delta}} \\
&= \frac{2^{-\sqrt{k}\Delta}}{|\mathcal{R}|} \\
&< \frac{2^{-k}}{|\mathcal{R}|}
\end{aligned}$$

[8] y' is not actually part of the transcripts, since it is not a message exchanged. Rather, it is computed by applying a polynomial-time computable function to α (see Definition 5.3.1). But, for the purposes of this proof, it is convenient to treat it on its own.

Thus, we can take $\delta = 2^{-k+1}$ and $\varepsilon = 2^{-k}$ in Lemma 5.4.10, and see that the two distributions on $(h_1, (y, r'))$ have statistical difference $2^{-\Omega(k)}$ (conditioned on history α).

In both $\overline{S}$ and $\langle M, A\rangle$, z is generated by applying the same randomized procedure to the history $(\alpha, h_1, (y, r'))$ (namely, the strategy for P), so including z does not increase the statistical difference. Another application of Lemma 5.4.10, similar to the one above, shows that the distributions on $(h_2, (r, y''))$ have statistical difference at most $2^{-\Omega(k)}$, conditioned on any history $(\alpha, h_1, (y, r'), z)$ in which y is not $\sqrt{k}\Delta$-light. Since y is distributed according to the Δ-flat distribution $V_{x,k}$ in the simulator, it is $\sqrt{k}\Delta$-light with probability at most 2^{-k+1}.

Finally, including β only increases the statistical difference by $2^{-\Omega(k)}$ by the weak zero-knowledge property of the sample test protocol (noting that in the simulator, y'' is $\sqrt{k}\Delta$-light with probability at most 2^{-k+1} and r'' is distributed uniformly in $\Omega(y'')$). ■

Lemma 5.4.11 immediately implies the completeness of (M, A):

Lemma 5.4.12 (completeness) *(M, A) has completeness error $2^{-\Omega(k)}$.*

Proof: The transcript generated by $\overline{S}$ is accepting whenever the transcript $(y, z; r)$ it receives from $\langle P, V\rangle(x, 1^k)$ is accepting. Since (P, V) has completeness error at most $\exp(-\Omega(k))$ and the statistical difference between $\overline{S}$ and $\langle M, A\rangle(x, 1^k)$ is at most $\exp(-\Omega(k))$, it follows that (M, A) has completeness error $\exp(-\Omega(k))$. ■

Statistical zero knowledge also follows readily from Lemma 5.4.11; using a simulator of deviation μ instead of sample of $\langle P, V\rangle$ can only affect the statistical difference by μ.

Lemma 5.4.13 (statistical zero knowledge) *If S simulates (P, V) with deviation deviation $\mu(k)$, then Algorithm 5.4.9 simulates (M, A) with deviation $\mu(k) + 2^{-\Omega(k)}$. Thus, if (P, V) is honest-verifier statistical zero knowledge, then so is (M, A).*

Computational zero knowledge follows from the additional observation that Algorithm 5.4.9 performs an *efficient* computation on the transcript of (P, V) received.

Lemma 5.4.14 (computational zero knowledge) *If (P, V) is honest-verifier computational zero knowledge, then so is (P, V).*

Proof: If $\langle P, V\rangle(x, 1^k)$ and $S(x, 1^k)$ are computationally indistinguishable, then so are $\overline{S}$ and the output of Algorithm 5.4.9, because they are obtained by applying the same polynomial-time procedures to $\langle P, V\rangle(x, 1^k)$ and $S(x, 1^k)$, respectively. Since $\langle M, A\rangle(x, 1^k)$ has negligible statistical difference from $\overline{S}$, it follows that it too is computationally indistinguishable from the output of Algorithm 5.4.9. ■

This completes the proof of Theorems 5.1.1, and the proof of Theorem 5.1.2 for 2-message proof systems. To extend the result to 3-message **HVCZK**, we simply note that including an extra prover message w at the start of the proof system does not cause any problems. Throughout the construction and analysis, the verifier message distribution $V_{x,k}$ should be replaced with $V_{x,k,w}$, which is the verifier's message distribution when the input is x, the security parameter is k, and the prover's first message is w.

We can strengthen the statements of the theorems somewhat. First, recall that we showed that every problem in **HVSZK** has a 2-message statistical zero-knowledge proof with simulator deviation 2^{-k} (Corollary 4.1.1). Using such a proof system as the starting point for the construction, Lemma 5.4.13 says that the resulting simulator deviation will be $2^{-\Omega(k)}$. (Actually, we did poly(k) parallel repetitions to obtain the (P, V) used in the transformation, which increases the simulator deviation by a poly(k) factor, but poly$(k) \cdot 2^{-\Omega(k)} = 2^{-\Omega(k)}$.) Renaming k, the simulator deviation becomes simply 2^{-k}. Second, Fürer *et. al.* [FGM$^+$89] have shown how to transform public-coin proofs into ones with perfect completeness; their transformation preserves honest-verifier statistical and computational zero-knowledge, and in fact preserves an exponentially small simulator deviation.[9] Thus, we obtain:

Theorem 5.4.15 (Thm. 5.1.1, strengthened) *Every problem in* **HVSZK** *has a public-coin honest-verifier statistical zero-knowledge proof with perfect completeness and simulator deviation* 2^{-k}.

Theorem 5.4.16 (Thm. 5.1.2, strengthened) *Every problem possessing a 3-message honest-verifier computational zero-knowledge proof also possesses a public-coin honest-verifier computational zero-knowledge proof with perfect completeness.*

By Lemma 4.6.7, we can immediately deduce an analogous result for knowledge complexity in the hint sense.

Corollary 5.4.17 *Let* $\kappa : \mathbb{N} \to \mathbb{N}$ *be any polynomially bounded function. Then every problem* $\Pi \in \mathbf{SKC_{hint}}(\kappa(n))$ *has a public-coin proof system of statistical knowledge complexity* $\kappa(n)$.

A corollary of the result for computational zero knowledge, is that we can now also prove the Ostrovsky–Wigderson for 3-message computational zero-knowledge proofs. That is, combining Theorems 4.8.8 and 5.1.2, we get:

Theorem 5.4.18 ([OW93] for 3-message proofs) *If a hard-on-average promise problem possesses a 3-message computational zero-knowledge proof, then one-way functions exist.*

The most important open problem left by these results is to remove the 3-message restriction for computational zero knowledge (without making any computational assumptions).

Open Problem 5.4.19 *Does every problem in* **HVCZK** *possess a public-coin* **HVCZK** *proof system?*

---

[9] Fürer *et. al.* [FGM$^+$89] actually claim to convert any honest-verifier statistical zero-knowledge proof (even a private-coin one) into one with perfect completeness, but actually, their transformation only preserves zero knowledge when starting with a public-coin proof system.

Another interesting problem is to improve the message complexity of the transformation. Neither of these theorems give any guarantee on the message complexity of the public-coin proof systems produced, despite the fact that they are obtained by starting with constant-message proof systems. This is a sharp contrast with the Goldwasser–Sipser transformation which only increases the message complexity by 2 (and this can be actually reduced to zero using the collapse theorems of [BM88]). Thus, the following question is still unanswered:

Open Problem 5.4.20 *Does every problem in* **HVSZK** *have a public-coin* **HVSZK** *proof system which exchanges a constant number of messages?*

For a positive answer to this question, it would suffice to exhibit constant-message (public-coin) sample generation and sample test protocols, as the rest of Protocol 5.4.2 only uses a constant number of messages. (The proof system for Entropy Difference in [GV99] uses even fewer messages beyond the sample generation and test protocols.)

5.5 Okamoto's subprotocols

In this section, we present Okamoto's protocols for generating and testing samples from a nearly flat distribution. Recall that these protocols must be public coin and furthermore must satisfy certain "zero-knowledge" properties.

5.5.1 Overview

Sample generation. Here the input to the protocol (M, A) is a Δ-flat distribution X (encoded by a circuit) and the output should be a sample x from this distribution. We require that, no matter what strategy M follows, x will not be too heavy. If, however, both parties play honestly, then x should be nearly typical with high probability, and should be simulatable for an *externally specified* x. In particular, the protocol should not reveal an input to the circuit X that yields x, as the simulator is only given x and it may be difficult to find an input yielding x in general. If we remove this condition, the problem becomes trivial: A could just sample x according to X and reveal both x and the input used to produce it. Since X is nearly flat, x will be nearly typical with high probability.

Okamoto's solution to this problem has the following general structure: M proposes a sample x (which is supposed to be distributed according to X) and sends it to A. (Of course, if M is dishonest, he can choose x to be too heavy.) Then M and A engage in a short "game" which ends by M proposing another sample x'. Roughly speaking, this game has the following properties:

1. If x is too heavy, then no matter what strategy M follows, he will be forced to select x' which is noticeably lighter than x.
2. If x is not too heavy, then no matter what strategy M follows, he will be forced to choose x' that is also not too heavy.
3. If x is nearly typical and M plays honestly, then x' will also be nearly typical.

4. If M plays honestly, then A's view of the game is simulatable for an externally specified x'.

Clearly, repeating this game many times to obtain a sequence of samples $x_0, \ldots, x_m$ (where x_0 is proposed by M and $x_{i+1} = x'_i$) will have the effect of pushing a heavy proposal for x_0 closer and closer to the nearly typical set. Taking m sufficiently large (but still polynomial in the appropriate parameters), x_m will be guaranteed to be not too heavy, no matter how M plays. On the other hand, if M plays honestly, all the samples will be nearly typical. Finally, the simulability property of the game enables the entire sample generation protocol to be simulated "backwards" for an externally specified x_m.

Sample test. Here the input to the protcol (M, A) is a Δ-flat distribution X (encoded by a circuit) together with a string x from the domain of X. At the end of the protocol, A accepts or rejects. We require that if x is too light, A should reject with high probabability. If, however, x is nearly typical and both parties play honestly, then A should accept with high probability, and, moreover, A's view of the interaction should be simulatable (given additionally a random input for X which yields x).

The general structure of this protocol is very similar to that of the sample generation protocol. Given x, M and A engage in a short game which ends by M proposing another sample x'. Roughly speaking, this game has the following properties:

1. If x is too light, then no matter what strategy M follows, he will be forced to select x' which is noticeably lighter than x.

2. If x is nearly typical and M plays honestly, then x' will also be nearly typical.

3. If x is nearly typical and M plays honestly, then A's view of the game is simulatable (given a random input to X which yields x).

Clearly, repeating this game many times to obtain a sequence $x_0, \ldots, x_m$ (where $x_0 = x$ and $x_{i+1} = x'_i$) will have the effect of making a light input sample lighter and lighter. Taking m sufficiently large, x_{m-1} will be so light that it has zero probability, so there is no x_m lighter than x_{m-1} and A will reject! Notice that we do not care what happens in the game if x_i is not too light and M plays dishonestly; if the original input is too light (which is the the only time we worry about a dishonest M), all the subsequent x_i's will also be too light with high probability. On the other hand, if the original input x is nearly typical and M plays honestly, all the samples will be nearly typical. Finally, the simulability property of the game enables the entire sample test protocol to be simulated "forwards" given coins for x. Amazingly, the game used for the sample test protocol is identical to the game used for the sample generation protocol. We describe this "pushing" game in the next section, and subsequently give formal descriptions of the two protocols.

5.5.2 The pushing game

Throughout the remainder of Section 5.5, X is a Δ-flat distribution encoded by a circuit and m (resp., n) denotes the length of the input (resp., output) of the circuit generating X.

Recall that for positive integers k and ℓ, $\mathcal{H}_{k,\ell}$ denotes a 2-universal family of hash functions mapping $\{0,1\}^k$ to $\{0,1\}^\ell$.

The basic game underlying the sample generation and sample test protocols is the 2-message protocol given in Protocol 5.5.1 (called "sequentially recursive hashing" in [Oka96]).

Protocol 5.5.1: Pushing game (M, A)

Input: (X, x, Δ, t), where $x \in \{0,1\}^n$ and $t \leq \Delta$

1. A: Choose h uniformly from $\mathcal{H}_{m+n,m-3t\Delta}$ and send h to M.
2. M: Choose (r, x') from the distribution $\Omega_X(x) \times X$, conditioned on $h(r, x') = 0$, and send (r, x') to A. (If there is no such pair (r, x'), then M sends `fail` to A.)
3. A: Check that $X(r) = x$ and $h(r, x') = 0$. If either condition fails, reject.

Output: x'

Observe that if $|\Omega_X(x)| = \emptyset$, then A rejects with probability 1. In order to describe remaining the properties of the pushing game, we define the *weight* of a string x relative to a circuit X by $\mathrm{wt}_X(x) = \log(\Pr[X = x] \cdot 2^{\mathrm{H}(X)})$. So, x is γ-heavy iff $\mathrm{wt}_X(x) \geq \gamma$ and x is γ-light iff $\mathrm{wt}_X(x) \leq -\gamma$. Also note that for x in the support of X, $|\mathrm{wt}_X(x)| \leq m$. When the distribution X is clear from the context, we will often write $\mathrm{wt}(x)$ instead of $\mathrm{wt}_X(x)$. The following lemma asserts that no matter how M plays, if the input to the game is atypical, then the output is noticeably lighter. (The behavior on typical inputs is analyzed later — in Lemma 5.5.3.)

Lemma 5.5.2 *If A follows the prescribed strategy in the pushing game, then no matter what strategy M uses, the following hold:*

1. *("heavy gets lighter") With probability $\geq 1 - 2^{-\Omega(t^2)}$, either* $\mathrm{wt}(x') < \max(\mathrm{wt}(x) - 1, 2\sqrt{t\Delta} \cdot \Delta)$ *or A rejects.*
2. *("light gets lighter") If* $\mathrm{wt}(x) \leq -6\sqrt{t\Delta} \cdot \Delta$, *then with probability* $\geq 1 - 2^{-\Omega(t^2)}$, *either* $\mathrm{wt}(x') < \mathrm{wt}(x) - 1$ *or A rejects.*

Proof: 1. Let S be the set of x' such that $\mathrm{wt}(x') \geq \max(\mathrm{wt}(x) - 1, 2\sqrt{t\Delta} \cdot \Delta)$. We need to show that with probability at most $2^{-\Omega(t^2)}$ over the choice of h from $\mathcal{H}_{m+n,m-3t\Delta}$, there exists a pair $(r, x') \in \Omega_X(x) \times S$ such that $h(x, r') = 0$. By the soundness of the standard lower-bound protocol (Lemma 5.2.3), it suffices to prove that

$$|\Omega_X(x) \times S| \leq 2^{-\Omega(t^2)} \cdot 2^{m-3t\Delta}.$$

The intuition is that the number of x' that are heavier than $\max(\mathrm{wt}(x)-1, 2\sqrt{t\Delta}\cdot\Delta)$ is so small that not even the size of $\Omega_X(x)$ can compensate.

By definition of $\mathrm{wt}(x)$, $|\Omega_X(x)| = 2^{m-\mathrm{H}(X)+\mathrm{wt}(x)}$. We now bound $|S|$. First, since X is Δ-flat, we have

$$\begin{aligned} 2^{-4t\Delta+1} &\geq \Pr_{x'\leftarrow X}\left[\mathrm{wt}(x') \geq 2\sqrt{t\Delta}\cdot\Delta\right] \\ &\geq \Pr\left[X \in S\right] \\ &= \sum_{x'\in S} \Pr\left[X = x'\right] \end{aligned}$$

On the other hand, every $x' \in S$ is $(\mathrm{wt}(x)-1)$-heavy, which means that $\Pr[X = x'] \geq 2^{-\mathrm{H}(X)+\mathrm{wt}(x)-1}$. Thus,

$$2^{-4t\Delta+1} \geq |S| \cdot 2^{-\mathrm{H}(X)+\mathrm{wt}(x)-1}.$$

Putting everything together, we have

$$\begin{aligned} |\Omega_X(x) \times S| &\leq 2^{m-\mathrm{H}(X)+\mathrm{wt}(x)} \cdot \left(\frac{2^{-4t\Delta+1}}{2^{-\mathrm{H}(X)+\mathrm{wt}(x)-1}}\right) \\ &= 2^{m-4t\Delta+2} \\ &\leq 2^{-t^2+2} \cdot 2^{m-3t\Delta}, \end{aligned}$$

as desired. (In the last inequality, we used the fact that $t \leq \Delta$.)

2. Let $S = \{x' : \mathrm{wt}(x') \geq \mathrm{wt}(x) - 1\}$. Again, it suffices to show that $|\Omega_X(x) \times S| \leq 2^{-\Omega(t^2)} \cdot 2^{m-3t\Delta}$. Here the intuition is that $|\Omega_X(x)|$ is so small (since x is so light) that the only way for M to succeed is to choose x' even lighter than x (since there cannot be too many strings of noticeable probability mass). This time we bound $|S|$ by dividing S into two parts. Define

$$\begin{aligned} S_1 &= \{x' : \mathrm{wt}(x) - 1 \leq \mathrm{wt}(x') \leq -2\sqrt{t\Delta}\cdot\Delta\} \\ S_2 &= \{x' : -2\sqrt{t\Delta}\cdot\Delta < \mathrm{wt}(x')\}, \end{aligned}$$

so that $S = S_1 \cup S_2$. Since every $x' \in S_2$ has probability mass greater than $2^{-\mathrm{H}(X)-2\sqrt{t\Delta}\cdot\Delta}$, we must have

$$\begin{aligned} |S_2| &< 2^{\mathrm{H}(X)+2\sqrt{t\Delta}\cdot\Delta} \\ &\leq 2^{\mathrm{H}(X)-\mathrm{wt}(x)-4t\Delta}, \end{aligned}$$

where the last inequality follows from $\mathrm{wt}(x) \leq -6\sqrt{t\Delta}\cdot\Delta$ and $\Delta \geq t$. We now bound $|S_1|$. Since X is Δ-flat, we have

$$\begin{aligned} 2^{-4t\Delta+1} &\geq \Pr\left[X' \in S_1\right] \\ &\geq |S_1| \cdot 2^{-\mathrm{H}(X)+\mathrm{wt}(x)-1}. \end{aligned}$$

Thus, $|S_1| \le 2^{\mathrm{H}(X)-\mathrm{wt}(x)-4t\Delta+2}$, and so

$$|S| = |S_1| + |S_2| < 2^{\mathrm{H}(X)-\mathrm{wt}(x)-4t\Delta+3},$$

and

$$\begin{aligned} |\Omega_X(x) \times S| &< 2^{m-\mathrm{H}(X)+\mathrm{wt}(x)} \cdot 2^{\mathrm{H}(X)-\mathrm{wt}(x)-4t\Delta+3} \\ &= 2^{m-4t\Delta+3} \\ &\le 2^{-t^2+3} \cdot 2^{m-3t\Delta}, \end{aligned}$$

as desired. ■

The pushing game has the following simulability and "completeness" properties when both parties are honest:

Lemma 5.5.3 *If both parties follow the protocol in the pushing game and x is $t\Delta$-typical, then the following two distributions have statistical difference at most $2^{-\Omega(t^2)}$:*

(A) *Execute the pushing game on input (X, x, Δ, t) to obtain (h, r, x'). Output (h, r, x').*

(B) *Let x' be distributed according to X and let r be selected uniformly from $\Omega_X(x)$. Choose h uniformly in $\mathcal{H}_{m+n,m-3t\Delta}$ subject to $h(r, x') = 0$. Output (h, r, x').*

Proof: We apply Lemma 5.4.10 with $Z = \Omega_X(x) \times X$, $\mathcal{D} = \{0,1\}^{m+n}$ and $\mathcal{R} = \{0,1\}^{m-3t\Delta}$. Distribution (A) (resp., (B)) in Lemma 5.4.10 corresponds to Distribution (A) (resp., (B)) above. Since X is Δ-flat, the following holds with probability $\ge 1 - 2^{-t^2+1}$ over (r, x') selected according to $\Omega_X(x) \times X$:

$$\begin{aligned} \Pr\left[\Omega_X(x) = (r, x')\right] &= \Pr\left[X = x'\right] \cdot \frac{1}{|\Omega_X(x)|} \\ &< 2^{-\mathrm{H}(X)+t\Delta} \cdot \frac{1}{2^{m-\mathrm{H}(X)-t\Delta}} \\ &= \frac{2^{-t\Delta}}{|\mathcal{R}|} \end{aligned}$$

Thus, we can take $\delta = 2^{-t^2+1}$ and $\varepsilon = 2^{-t\Delta} \le 2^{-t^2}$ in Lemma 5.4.10, and see that the two distributions have statistical difference $2^{-\Omega(t^2)}$. ■

5.5.3 The protocols

The sample generation and test protocols are given in Protocols 5.5.4 and 5.5.5, respectively, They simply consist of many repetitions of the basic pushing game.

5.5.4 Correctness of sample generation protocol

Using the properties of the pushing game, we now prove that the sample generation protocol satisfies Definition 5.3.2 and thus Theorem 5.3.3 holds.

Protocol 5.5.4: Sample generation protocol (M, A)

Input: (X, Δ, t), where $t \leq \Delta$

1. M: Select $x_0 \in \{0,1\}^n$ according to X and send x_0 to A.
2. M, A: Repeat for i from 1 to m: Execute the pushing game on input (X, x_{i-1}, Δ, t) and let x_i be the output.

Output: x_m, unless A rejects in one of the pushing games, in which case output any canonical string outside the range of X (*e.g.*, 0^{n+1}).

Protocol 5.5.5: Sample test protocol (M, A)

Input: (X, x, Δ, t), where $x \in \{0,1\}^n$ and $t \leq \Delta$

1. M, A: Let $x_0 = x$.
2. M, A: Repeat for i from 1 to $m + 1$: Execute the pushing game on input (X, x_{i-1}, Δ, t) and let x_i be the output.
3. A: Reject if A rejected in any of the pushing games, else accept.

Soundness. By Lemma 5.5.2 (Part 1) and induction, we see that for every $0 \le i \le m$, with probability at least $1 - i \cdot 2^{-\Omega(t^2)}$, either $\mathrm{wt}(x_i) < \max(\mathrm{wt}(x_0) - i, 2\sqrt{t\Delta} \cdot \Delta)$ or A rejects. In particular, since $\mathrm{wt}(x_0) \le m$, with probability at least $1 - m \cdot 2^{-\Omega(t^2)}$, we have

$$\mathrm{wt}(x_m) < \max(\mathrm{wt}(x_0) - m, 2\sqrt{t\Delta} \cdot \Delta) = 2\sqrt{t\Delta} \cdot \Delta$$

unless A rejects. In addition, if A rejects in any of the pushing games, then the output has weight 0 (since it is chosen to be outside the support of X). Therefore, soundness is satisfied.

Completeness and zero knowledge. First we observe that the completeness condition follows from the strong zero-knowledge condition: In Distribution (B) of Definition 5.3.2, x is distributed according to X, and hence is $t\Delta$-typical with probability $\ge 1 - 2^{-t^2+1}$ by the Δ-flatness of X. Since x corresponds to the output of the sample generation protocol in Distribution (A) and Distributions (A) and (B) have statistical difference at most $2^{-\Omega(t^2)}$, the output of the sample generation protocol must be $t\Delta$-typical with probability at least $1 - 2^{-t^2+1} - 2^{-\Omega(t^2)} = 1 - 2^{-\Omega(t^2)}$.

Now we prove the zero-knowledge condition. Consider the probabilistic polynomial-time simulator given in Algorithm 5.5.6.

Algorithm 5.5.6: Simulator for sample generation protocol

Input: (X, Δ, t, x)

1. Let $x_m = x$.
2. For i from m down to 1 repeat:
   (a) Choose r_{i-1} uniformly from $\{0,1\}^m$ and let $x_{i-1} = X(r_{i-1})$.
   (b) Choose h_i uniformly from $\mathcal{H}_{m+n,m-3t\Delta}$ subject to $h_i(r_{i-1}, x_i) = 0$.
3. Output $(x_0, h_1, (r_0, x_1), h_2, (r_1, x_2), \ldots, h_m, (r_{m-1}, x_m))$.

We prove by induction on i that the distribution on $\gamma_i = (x_0, h_1, (r_0, x_1), \ldots, h_i, (r_{i-1}, x_i))$ in the output of the simulator (when x is chosen according to X) has statistical difference at most $i \cdot 2^{-\Omega(t^2)}$ from the verifier's view of the sample generation protocol up to the end of the i'th execution of the pushing game. Clearly this is true for $i = 0$, as in both cases x_0 is distributed according to X. Now suppose it is true for i; we will prove it for $i + 1$. From the following two observations it follows that the statistical difference only increases by $2^{-t^2+1} + 2^{-\Omega(t^2)} = 2^{-\Omega(t^2)}$ when going from i to $i + 1$:

1. In the simulator, x_i is $t\Delta$-typical with probability at least $1 - 2^{-t^2+1}$.
2. For any history $\gamma_i = (x_0, h_1, (r_0, x_1), \ldots, h_i, (r_{i-1}, x_i))$ in which x_i is $t\Delta$-typical, the

following two distributions have statistical difference $2^{-\Omega(t^2)}$:

(A) A's view of the $(i+1)$'st pushing game conditioned on history γ_i.

(B) The distribution of $(h_{i+1}, (r_i, x_{i+1}))$ conditioned on history γ_i in the output of the simulator.

Observation 1 is immediate from the fact that x_i is distributed according to X in the simulator and X is Δ-flat. Observation 2 follows from Lemma 5.5.3, observing that conditioned on history γ_i, the triple $(h_{i+1}, (r_i, x_{i+1}))$ in the output of the simulator is selected exactly according to the Distribution (B) in Lemma 5.5.3. That is, conditioned on history γ_i, r_i is selected uniformly from $\Omega_X(x_i)$, x_{i+1} is distributed according to X, and h_{i+1} is selected uniformly in $\mathcal{H}_{m+n,m-3t\Delta}$ subject to $h_{i+1}(r_i, x_{i+1}) = 0$.

5.5.5 Correctness of sample test protocol

Finally, we prove that the sample test protocol satisfies Definition 5.3.4 and thus Theorem 5.3.5 holds.

Soundness. By Lemma 5.5.2 (Part 2) and induction, we see that if $\mathrm{wt}(x) \le -6\sqrt{t\Delta}\cdot\Delta$, then with probability at least $1 - i \cdot 2^{-\Omega(t^2)}$, for every $0 \le i \le m+1$, $\mathrm{wt}(x_i) < \mathrm{wt}(x_0) - i$ (or A rejects). In particular, since $\mathrm{wt}(x_0) < \mathrm{H}(X)$, with probability at least $1 - m \cdot 2^{-\Omega(t^2)}$, we have $\mathrm{wt}(x_m) < \mathrm{H}(X) - m$ unless A rejects at some iteration. Since $m - \mathrm{H}(X) + \mathrm{wt}(x_m) = \log|\Omega_X(x_m)|$ cannot be negative unless $|\Omega_X(x_m)| = \emptyset$, it follows that with probability at least $1 - m \cdot 2^{-\Omega(t^2)}$, A must reject in one of the iterations.

Completeness and zero knowledge. First we prove the zero-knowledge condition. Consider the probabilistic polynomial-time simulator given in Algorithm 5.5.7.

Algorithm 5.5.7: Simulator for sample test protocol

Input: (X, x, Δ, t, r)

1. Let $x_0 = x$ and $r_0 = r$.
2. For i from 1 to m repeat:
   (a) Choose r_i uniformly from $\{0,1\}^m$ and let $x_i = X(r_i)$.
   (b) Choose h_i uniformly from $\mathcal{H}_{m+n,m-3t\Delta}$ subject to $h_i(r_{i-1}, x_i) = 0$.
3. Output $(x_0, h_1, (r_0, x_1), h_2, (r_1, x_2), \ldots, h_{m+1}, (r_m, x_{m+1}))$.

We prove by induction on i that the distribution on $\gamma_i = (x_0, h_1, (r_0, x_1), \ldots, h_i, (r_{i-1}, x_i))$ in the output of the simulator (when r is selected uniformly from $\Omega_X(x)$ and x is $t\Delta$-typical) has statistical difference at most $i \cdot 2^{-\Omega(t^2)}$ from the verifier's view of the sample test protocol up to the end of the i'th execution of the pushing game. Clearly this is true for $i = 0$. The induction step is proved analogously to the argument used for the sample generation protocol, using the same two observations and noting that, although the simulator works in reverse order, the selection of r_i and h_i is as before.

Now we observe that the completeness condition follows from the weak zero-knowledge condition and the particular simulator we have given above. Specifically, the above simulator always outputs transcripts which would make A accept. Since it has statistical difference at most $m \cdot 2^{-\Omega(t^2)}$ from the sample test protocol, A must accept in the sample test protocol with probability at least $1 - m \cdot 2^{-\Omega(t^2)}$.

Chapter 6

Coping with Cheating Verifiers

Up to this point, the focus of our investigation has been *honest-verifier* zero-knowledge proofs, which only guarantee that the verifier learns nothing if it follows the specified protocol. The existence of such honest-verifier proofs is already interesting from both a mathematical and philosophical point of view, and, as we have seen, one can develop a rich theory about their complexity. However, from a cryptographic point of view, the applicability of honest-verifier proofs is quite limited, since it is usually unreasonable to assume that mutually distrustful parties will follow a given protocol. Indeed, one of the most dramatic applications of zero-knowledge proofs is as a general tool for limiting the amount of "cheating" in cryptographic protocols [GMW91, Yao86, GMW87]. Clearly, honest-verifer proofs are unsuitable for such purposes.

The main contribution of this chapter is a general method for converting honest-verifier zero-knowledge proofs into proofs which remain zero knowledge even against cheating verifiers. The transformation applies to all honest-verifier statistical zero-knowledge proofs, ansd thus we conclude that **HVSZK** = **SZK**. It also applies to all public-coin honest-verifier computational zero-knowledge proofs. Such a result is useful in several ways. First, the transformation allows us to immediately translate the results we have obtained about honest-verifier zero knowledge (such as the complete problems and closure properties) to the cheating-verifier zero knowledge. Second, the transformation suggests a useful methodology for constructing zero-knowledge proofs: First construct an honest-verifier zero-knowledge proof for the problem at hand (which is often an easier task), and then use our general transformation to convert it into one robust against cheating verifiers.

Our transformation relies on a new "random selection protocol," which may be useful in other settings. It is a protocol for two mutually distrustful parties to select a "random" string of a given length, with certain (assymmetric) guarantees on how much each party can affect the output distributions and an additional simulability property for one of the parties. The random selection protocol in turn relies on a new lemma about 2-universal hash functions.

We begin, in Section 6.1, by defining the various forms of zero-knowledge proofs against cheating verifiers, and discussing some issues that arise in the definitions. In order to illustrate the definitions, in Section 6.2 we present such a (cheating-verifier) statistical zero-knowledge proof for a variant of STATISTICAL DIFFERENCE (namely, $\overline{\mathrm{SD}^{1,1/2}}$).[1] In Section 6.3, we give our transformation from honest-verifier proofs to cheating-verifier ones and prove its correctness, assuming the existence of a random selection protocol with certain properties. We exhibit such a random selection protocol in Section 6.4, completing the proof of the main results of this chapter. We conclude, in Section 6.5, by listing some corollaries and open problems. In particular, we describe the results about **SZK** obtained by translating things we have proven about **HVSZK**.

6.1 Definitions

The basic approach of Goldwasser, Micali, and Rackoff [GMR89] in defining zero-knowledge proofs against cheating verifiers, is to require that, for every polynomial-time verifier strategy V^*, there exists a simulator whose output distribution is close to V^*'s view of the interaction. As with honest-verifier zero knowledge, different interpretations of "close" yield perfect, statistical, and computational variants of the definition. Also like the honest-verifier versions, our definitions differ from the original definitions of Goldwasser, Micali, and Rackoff [GMR89] in that we use a security parameter to control the error parameters and we require the simulator to run in strict polynomial time (but allow a failure probability). As discussed in more detail later, we allow the verifier to be nonuniform in these definitions, and hence provide the simulator with the same "advice" as is given to the verifier. For a polynomial-time algorithm V and a string a, let $V_{[a]}$ denote V with "advice" string a. (We adopt the conventation that the running time of V is is independent of a, so if a is too long, V will not be able to access it in its entirety.)

Definition 6.1.1 (cheating-verifier zero knowledge — PZK, SZK)
An interactive proof system (P,V) for a promise problem Π is said to be statistical zero knowledge *if for every probabilistic polynomial-time V^*, there exists a useful*[2] *probabilistic polynomial-time S and a negligible function $\mu(\cdot)$ such that*

$$\mathrm{StatDiff}\left(\widetilde{S}_{[a]}(x,1^k), \langle P, V^*_{[a]}\rangle(x,1^k)\right) \le \mu(k) \qquad \forall x \in \Pi_Y, k \in \mathbb{N}, a \in \{0,1\}^*.$$

The negligible function μ is called the simulator deviation *for V^*. If, for every V^*, $\mu \equiv 0$, then (P,V) is said to be* perfect zero knowledge. **SZK** *(resp.,* **PZK***) denotes the class of promise problems possessing statistical (resp., perfect) zero-knowledge proofs.*

Definition 6.1.2 (cheating-verifier zero knowledge — CZK)
An interactive proof system (P,V) for a promise problem Π is said to be computational zero knowledge *if for every probabilistic polynomial-time V^*, there exists a useful probabilistic polynomial-time S such that*

$$\left\{\widetilde{S}_{[a]}(x,1^k)\right\}_{x\in\Pi_Y,k\in\mathbb{N}} \qquad \textit{and} \qquad \left\{\langle P, V^*_{[a]}\rangle(x,1^k)\right\}_{x\in\Pi_Y,k\in\mathbb{N},a\in\{0,1\}^*}$$

[1] We have not shown this variant of SD to be complete for **HVSZK**, so this does not prove that **HVSZK** = **SZK**.

[2] Recall that a probabilistic algorithm A is called *useful* if $\Pr[A(x) = \mathtt{fail}] \le 1/2$ for all x and $\widetilde{A}(x)$ denotes the output distribution of A on input x, conditioned on $A(x) \ne \mathtt{fail}$.

are computationally indistinguishable. **CZK** *denotes the class of promise problems possessing computational zero-knowledge proofs.*

Note that we have allowed the verifier strategy V^* to be nonuniform even in the case of **SZK** and **PZK**. As it did with **HVCZK**, allowing nonuniformity allows us to prove that zero knowledge is preserved under sequential repetition, and this augmentation becomes even more important in the setting of cheating verifiers. In fact, it has been proven that cheating-verifier computational zero knowledge fails to be preserved under the uniform versions of the definitions [GK96b]. Allowing nonuniformity is also important when zero-knowledge proofs are used as components of larger cryptographic protocols, as in [Yao86, GMW87]. Intuitively, in these settings, the verifier can use information it has obtained prior to the start of the zero-knowledge proof (*e.g.,* from an earlier execution of the zero-knowledge proof, in the case of sequential composition) in trying to extract knowledge from the prover. A definition that is robust against nonuniform verifiers implies that such extra information does not help, as it can be regarded as "nonuniform advice". Even with a nonuniform definition, however, cheating-verifier zero knowledge fails to be closed under parallel composition [GK96b, FS90], so the only direct method for doing error reduction is sequential composition.

At first, Definitions 6.1.1 and 6.1.2 seem very hard to meet. How can one construct a simulator for each of the infinitely many possible verifier strategies? The way this task is handled in all known constructions of zero-knowledge proofs is that actually only one "universal" simulator is constructed. The way this one algorithm simulates the infinitely many possible verifiers is that, in order to simulate the interaction between P and some particular verifier V^*, the simulator is given *oracle access* to V^*. By observing the verifier's behavior when it is fed various partial transcripts, the simulator is able to construct a "good" simulation for that particular verifier. This sort of "universal" simulation in which the verifier is used as a "black box" was formalized by Goldreich and Oren [GO94]. One advantage of adopting such a definition is that it allows one to make sense of a proof being zero knowledge not just against polynomial-time verifiers, but *all* verifier strategies (even uncomputable ones).

Definition 6.1.3 (black-box simulation SZK) *An interactive proof system (P, V) for a promise problem Π is said to be* black-box simulation statistical zero knowledge *if there is a useful probabilistic polynomial-time algorithm S such that for every nonuniform probabilistic polynomial-time V^*,*

$$\mathrm{StatDiff}\left(\widetilde{S^{V^*}}(x,1^k), \langle P, V^*\rangle(x,1^k)\right) \leq \mu(k) \qquad \forall x \in \Pi_Y, k \in \mathbb{N},^{3} \tag{6.1}$$

for some negligible function μ (which may depend on V^*). The negligible function μ is called the simulator deviation *for V^*. If, for every V^*, $\mu \equiv 0$, then (P, V) is said to be* black-box simulation perfect zero knowledge. *If (6.1) holds for* all *verifier strategies V^* (not just polynomial-time ones), then the proof system is said to black-box simulation zero knowledge* against all verifiers

[3] Recall the notation $M^{\mathcal{O}}$ is used to indicate algorithm M being given oracle access to function $\mathcal{O}$.

Definition 6.1.4 (black-box simulation CZK) *An interactive proof system (P, V) for a promise problem Π is said to be* black-box simulation computational zero knowledge *if there is a useful probabilistic polynomial-time algorithm S such that for every nonuniform probabilistic polynomial-time V^*,*

$$\left\{\widetilde{S^{V^*}}(x, 1^k)\right\}_{x\in\Pi_Y, k\in\mathbb{N}} \qquad \textit{and} \qquad \left\{\langle P, V^*\rangle(x, 1^k)\right\}_{x\in\Pi_Y, k\in\mathbb{N}}$$

are computationally indistinguishable.

There is one subtlety in these definitions of black-box simulation, pointed out in [BMO90b]. S is required to run in time that is a fixed polynomial in its input length, yet it is required to simulate verifiers V^* whose running time can be an arbitrary polynomial in the input length, and hence even the messages and random coins of V^* can be too long for S to read. To deal with this, we give S some additional power:

1. S has random access to its communications with the oracle V^*, and may copy strings received from the oracle directly to the output (in one time step).

2. S can uniformly select and fix the random coins of V^* in one time step. It may also automatically copy them to the output in a single time step.

6.2 A cheating-verifier SZK proof system for $\overline{\mathrm{SD}^{1,1/2}}$

In this section, we illustrate the above definitions by giving a cheating-verifier zero-knowledge proof system for $\overline{\mathrm{SD}^{1,1/2}}$. The proof system is based on the perfect zero-knowledge proofs for Quadratic Residuosity [GMR89] and Graph Isomorphism [GMW91]. For motivation, we first observe that **NP** proofs can be given for membership in $\overline{\mathrm{SD}^{1,1/2}}$. A "proof" that two distributions X_0 and X_1 are not disjoint is simply a triple (x, r_0, r_1) such that $X_0(r_0) = x = X_1(r_1)$. In order to obtain a *zero-knowledge* proof, the prover sends just x (randomly sampled from one of the distributions) and the verifier asks for either a proof r_0 that $x \in \mathrm{Supp}(X_0)$ or a proof r_1 that $x \in \mathrm{Supp}(X_1)$. A formal description of this proof system is given in Protocol 6.2.1 (which is not yet our final proof system).

The completeness and soundness of this protocol are easy to check.

Lemma 6.2.2 *For any $\alpha < 1$, Protocol 6.2.1 is an interactive proof for $\overline{\mathrm{SD}^{1,\alpha}}$ with completeness error $\alpha/2$ and soundness error $1/2$.*

Proof: (Completeness) When both parties follow the protocol, A rejects iff $b = 1$ and x is not in the support of X_1. By the definition of statistical difference, a random sample from X_0 will fail to be in the support of X_1 with probability at most $\mathrm{StatDiff}(X_0, X_1) \leq \alpha$. Since $b = 1$ with probability $1/2$, A rejects with probability at most $\alpha/2$.

(Soundness) If X_0 and X_1 have disjoint supports, then no string x can be in the support of both distributions. So, with probability at least $1/2$, A will choose b so that $x \notin \mathrm{Supp}(X_b)$ and the prover will fail. ∎

Protocol 6.2.1: Basic proof system (M, A) for $\overline{\mathrm{SD}^{1,\alpha}}$

Input: Circuits X_0 and X_1 (each with m input gates and n output gates)

1. M: Sample $x \leftarrow X_0$. Send x to A.
2. A: Choose $b \leftarrow \{0,1\}$. Send b to M.
3. M: Choose r uniformly from $\Omega_b(x) \stackrel{\text{def}}{=} \{r' : X_b(r') = x\}$. Send r to A. (If $\Omega_b(x) = \emptyset$, then send `fail` to A.)
4. A: If $X_b(r) = x$, then accept. Otherwise, reject.

For zero-knowledgeness, we first consider the special case when X_0 and X_1 have statistical difference 0 and the verifier is honest. Note that the verifier's view consists of triples (x, b, r) where x is distributed according to X_0 (equivalently, X_1), b is uniform in $\{0,1\}$, and r is a random input to X_b yielding x. It is easy to generate such triples: choose b and r uniformly and let $x = X_b(r)$. The difficulty when extending this approach to cheating verifiers is that a cheating verifier may select b as a function of x. This can be handled by having the simulator "guess" b in advance and use its oracle access to A^* to check the guess at the end. Thus we consider the simulator given in Algorithm 6.2.3.

Algorithm 6.2.3: Black-box simulator for Protocol 6.2.1

Input: Circuits X_0 and X_1 (each with m input gates and n output gates) and oracle access to verifier A^*.

1. Select and fix the random coins R of A^*.
2. Choose $b \leftarrow \{0,1\}$ and $r \leftarrow \{0,1\}^m$.
3. Let $x = X_b(r)$.
4. Let $b' = A^*(x)$.[a]
5. If $b' = b$, output $(x, b, r; R)$. Otherwise, output `fail`.

[a] Here, and often in this chapter, we omit the input (X_0, X_1) and the random coins R of A^* to simplify the notation.

Lemma 6.2.4 *For any pair of circuits X_0, X_1 with statistical difference α and any verifier strategy A^* (even computationally unbounded), Algorithm 6.2.3 outputs* `fail` *with probability at most $(1+\alpha)/2$ and, conditioned on non-failure, has statistical difference at most α from $\langle M, A^* \rangle$.*

Proof: To compare the simulator distribution S^{A^*} to the real interaction $\langle M, A^* \rangle$, we consider the following intermediate distribution D:

D: Select and fix the random coins R of A^*. Choose $b \leftarrow \{0,1\}$. Sample $x \leftarrow X_0$. Let $b' = A^*(x)$. Choose $r \leftarrow \Omega_{b'}(x)$. Output $(x, b, r; R)$ if $b = b'$ and `fail` otherwise.

Note that, since b is independent of (x, b', r), D outputs `fail` with probability exactly $1/2$, and conditioned on non-failure, is distributed identically to $\langle M, A^* \rangle$. In addition, if the $x \leftarrow X_0$ in D is replaced with $x \leftarrow X_b$, we obtain exactly the output distribution of S^{A^*}. Since $b = 1$ with probability $1/2$ in D, the statistical difference between D and S^{A^*} is at most $(1/2) \cdot \mathrm{StatDiff}(X_0, X_1) = \alpha/2$. If we now condition both of these distributions on non-failure, the statistical difference increases by a factor of at most $1/\Pr[D = \texttt{fail}] = 2$ (as justified below). ■

The fact about the behavior of statistical difference with respect to conditioning used in the above proof is the following:

Lemma 6.2.5 *Let X and Y be any two distributions on a universe $\mathcal{U}$ and let $T \subset \mathcal{U}$ be any set. Let X' (respectively, Y') denote the distribution of X (resp., Y) conditioned on $X \in T$ (resp., $Y \in T$). Then,* $\mathrm{StatDiff}(X', Y') \leq \mathrm{StatDiff}(X, Y) / \Pr[X \in T]$.

Proof: We may assume that $\Pr[X \in T] \geq \Pr[Y \in T]$, for otherwise swapping the two distributions gives a stronger bound. Let T' be any subset of T. Then,

$$\begin{aligned}
\Pr[X' \in T'] - \Pr[Y' \in T'] &= \frac{\Pr[X \in T']}{\Pr[X \in T]} - \frac{\Pr[Y \in T']}{\Pr[Y \in T]} \\
&= \frac{\Pr[X \in T'] \cdot \Pr[Y \in T] - \Pr[Y \in T'] \cdot \Pr[X \in T]}{\Pr[X \in T] \cdot \Pr[Y \in T]} \\
&\leq \frac{\Pr[X \in T'] \cdot \Pr[Y \in T] - \Pr[Y \in T'] \cdot \Pr[Y \in T]}{\Pr[X \in T] \cdot \Pr[Y \in T]} \\
&= \frac{\Pr[X \in T'] - \Pr[Y \in T']}{\Pr[X \in T]} \leq \frac{\mathrm{StatDiff}(X, Y)}{\Pr[X \in T]}.
\end{aligned}$$

Maximizing over $T' \subset T$ completes the proof. ■

Setting $\alpha = 0$ in Lemmas 6.2.2 and 6.2.4, we have:

Proposition 6.2.6 $\overline{\mathrm{SD}^{1,0}} \in$ **PZK**. *Moreover, it has a perfect zero-knowledge proof with the following properties:*

1. *The proof system is public coin.*

2. *Perfect completeness and soundness error* $1/2$.

3. *1 bit of verifier-to-prover communication.*

4. *Exchanges only three messages.*

5. *Black-box simulation perfect zero knowledge against* all verifiers.

In order to reduce the error, one cannot do an arbitrary number of parallel repetitions, since **PZK** is not closed under parallel repetition [FS90]. However, up to $O(\log k)$ parallel repetitions *of this particular proof system* can be shown to remain perfect zero knowledge, by generalizing the simulator in the natural way. Thus, the error can be reduced to $1/k$ without increasing the number of rounds. It is unlikely that this can be improved further, as only problems in **BPP** have constant-round zero-knowledge proofs with negligible soundness error [GK96b].

Since Graph Isomorphism reduces to $\overline{\mathrm{SD}^{1,0}}$, the proof system for it from [GMW91] is a special case.

Corollary 6.2.7 ([GMW91]) Graph Isomorphism *is in* **PZK**. *Moreover, it has a proof system with all the properties listed in Proposition 6.2.6.*

As discussed above, this proof system has nonnegligible soundness error. However, Graph Isomorphism does have a constant-message (private-coin) perfect zero-knowledge proof system with negligible soundness error, as shown by Bellare, Micali, and Ostrovsky [BMO90a].

In order to get a zero-knowledge proof for $\overline{\mathrm{SD}^{1,\alpha}}$ for $\alpha < 1$, we need to reduce the statistical difference for YES instances. The XOR Lemma (Lemma 3.1.16) accomplishes exactly this. If we augment Protocol 6.2.1 by having both parties (and the simulator) apply the XOR Lemma to the two distributions, then Lemmas 6.2.2 and 6.2.4 imply that the resulting proof system is statistical zero knowledge:

Proposition 6.2.8 *For every constant* $\alpha < 1$, $\overline{\mathrm{SD}^{1,\alpha}} \in \mathbf{SZK}$. *Moreover, it has a statistical zero-knowledge proof with the following properties:*

1. *The proof system is public coin.*

2. *Completeness error* 2^{-k} *and soundness error* $1/2$

3. *1 bit of verifier-to-prover communication.*

4. *Exchanges only three messages.*

5. *Black-box simulation zero knowledge against* all verifiers *with simulator deviation* 2^{-k}.

This suggests one way to prove that **HVSZK** = **SZK** — show that SD (Karp) reduces to $\mathrm{SD}^{1,1/2}$.[4] Even a *randomized* Karp reduction would suffice (as long as the error probability is

[4]Since **HVSZK** is closed under complement, it does not matter whether we complement these problems or not.

negligible, so it can get absorbed in the simulator deviation). Unfortunately, we do not know how to do that. In the next section, we will prove that **HVSZK** = **SZK** in a completely different way. Nonetheless, the question of whether SD reduces to $\mathrm{SD}^{1,1/2}$ remains an interesting one, as it would show that every problem in **HVSZK** has a constant-message public-coin **HVSZK** proof (resolving Open Problem 5.4.20). Obtaining a deterministic reduction seems more difficult, as it would have the dramatic consequences that **HVSZK** = **HVPZK** (by Proposition 3.1.11) and **HVSZK** $\subset$ **NP** $\cap$ **co-NP** (since $\mathrm{SD}^{1,1/2} \in$ **co-NP** and **HVSZK** is closed under complement).

6.3 Transforming honest-verifier proofs to cheating-verifier ones

We now state the main results of this chapter.

Theorem 6.3.1 HVSZK = SZK. *Moreover, every promise problem in* **HVSZK** *possesses a statistical zero-knowledge proof with the following properties:*

1. *Black-box simulation with simulator deviation* 2^{-k} *for* all verifiers.
2. *The proof system is public coin.*
3. *Perfect completeness.*

Theorem 6.3.2 *Every problem possessing a public-coin* **HVCZK** *proof system also has a public-coin* **CZK** *proof system. Moreover, the* **CZK** *proof system has a black-box simulator and perfect completeness.*

We stress that both of these theorems are *unconditional*. Similar results have been previously achieved under intractability assumptions; we discuss these in more detail below. Somewhat surprisingly, Theorem 6.3.2 indicates that the intractability assumptions used in constructing (public-coin) computational zero-knowledge proofs do not play an essential role in dealing with cheating verifiers, but rather their importance lies solely in the construction of the honest-verifier proofs.

We prove both Theorems 6.3.1 and 6.3.2 by exhibiting a transformation from public-coin honest-verifier zero-knowledge proofs to public-coin (cheating-verifier) zero-knowledge proofs; we then use Theorem 5.1.1 to obtain a result that applies to all of (private-coin) **HVSZK**. The transformation is very efficient, in that it preserves the complexity of original proof system in many respects; this is described in detail in Section 6.3.

6.3.1 Previous results

Conditional results. For computational zero knowledge, the question is completely resolved if one assumes that (nonuniformly) one-way functions exist. This is because, under that assumption, it is known that **HVCZK** = **CZK** = **IP** [GMW91, IY87, BGG$^+$88]. In addition, the computational zero-knowledge proofs produced by these constructions already have the extra properties given in Theorem 6.3.2 (public coins, perfect completeness, black-box simulation).

The problem of giving a general transformation from honest-verifier zero-knowledge proofs to cheating-verifier ones was first studied by Bellare, Micali, and Ostrovsky [BMO90b], who showed that **HVSZK** = **SZK** under the assumption that the Discrete Logarithm problem is hard. Already there, they observe the potential benefits of such a transformation that we discussed at the beginning of the chapter, and indeed, use theirs to deduce several new results about **SZK** under the same intractability assumption. At first, it seems puzzling that computational assumptions can be used in the supposedly "information-theoretic" setting of statistical zero knowledge. However, a careful examination of the definitions reveals that the standard class **SZK** doers refer to computational limitations: It requires a simulator only for all *polynomial-time* verifiers. The computational assumption is therefore used to limit the behavior of cheating verifiers. Later work gradually weakened the assumption used to prove **HVSZK** = **SZK**. Ostrovsky, Venkatesan, and Yung [OVY93] achieved it under the assumption that one-way permutations exist, and finally, Okamoto [Oka96] proved it using any bit-commitment scheme (and hence any one-way function [HILL99, Nao91]).

However, there is something dissatisfying about using intractability assumptions to prove that **HVSZK** = **SZK**. One of the appealing features of statistical zero knowledge is that it can often be exhibited unconditionally and maintains its zero-knowledge properties even against computationally unbounded verifiers (as formalized in Definition 6.1.3). Needless to say, the results proving **HVSZK** = **SZK** under intractability assumptions only yield **SZK** proofs that are zero-knowledge against polynomial-time verifiers.

Unconditional results. Previously, the only unconditional transformations of honest-verifier zero knowledge to cheating-verifier zero knowledge were restricted to *constant-message* public-coin proof systems. The first such transformation was due to Damgård [Dam93], and another (with improved message complexity) was given by Damgård, Goldreich, and Wigderson [DGW94]. Both of these results apply to all three forms of zero knowledge — perfect, statistical, and computational.

Di Crescenzo, Okamoto, and Yung [DOY97] also claim to prove that **HVSZK** $\subset$ **weak-SZK**, where **weak-SZK** is defined analogously to **weak-HVSZK** (Definition 2.4.2).

6.3.2 Overview

We prove Theorems 6.3.1 and 6.3.2 by transforming *public-coin* honest-verifier zero-knowledge proofs to cheating-verifier ones. This focus on public coins simplifies the task considerably, and once again illustrates the usefulness of private-to-public coin transformations as given by Theorem 5.1.1. In a public-coin proof system, the honest verifier's behavior is very structured; it simply sends random coins flips at each round of interaction. So "cheating" amounts to sending messages that are not selected uniformly at random. Thus, a natural approach to making such a proof system zero knowledge for cheating verifiers is to replace the verifier's messages with strings *jointly chosen* by the prover and verifier in a "random selection protocol." If the verifier's ability to bias the outcome of this protocol is sufficiently limited, then we have essentially forced its behavior to be "honest." However, we must also take care that we do not give the prover too much control over the outcome of the protocol, lest the resulting proof system will not be sound. Finally, it order to conclude that the final proof system is zero knowledge, it is not enough that a cheating verifier cannot bias the outcome too much; it is also important that the verifier does not learn anything from the random selection protocol itself. Thus, some sort of simulability property is also needed.

Various random selection protocols were constructed for this purpose in [Dam93, DGW94, Oka96], but all of these either rely on computational assumptions or are restricted to constant-round proof systems. We will construct a random selection protocol without either of these limitations. Our random selection protocol builds on the one of Damgård, Goldreich, Wigderson [DGW94], so we begin by describing the properties of their construction. For every positive polynomial p, they give a protocol (the "DGW random selection protocol") for two parties ("Arthur" and "Merlin") for selecting a string in $\{0,1\}^\ell$ with the following properties:

1. As long as Arthur plays 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 variation distance 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 a deterministic cheating strategy, the outcome of the protocol is uniformly distributed over some set of $\frac{2^\ell}{p(\ell)^4}$ strings.

When this protocol is used to transform honest-verifier proof systems into cheating-verifier ones, the verifier plays the role of Arthur and the prover that of Merlin. The resulting proof system is simulated in [DGW94] by running the honest-verifier simulator, and hoping that all verifier messages included in the transcript fall in the sets mentioned in Item 2 above. This strategy succeeds with probability $1/p(\ell)^{4i}$, where i is the number of verifier messages in the original proof system. If the original proof system exchanges only a constant number of messages, then the success probability is nonnegligible and the above suffices for producing a black-box simulation with respect to any cheating verifier strategy. But this approach fails when we have a nonconstant number of messages.

In this paper we modify the above transformation as follows. Rather than selecting a message, we use the DGW random selection protocol to specify (in a succinct manner) a set of 2^k messages (where k is the security parameter). Merlin is then supposed to select a message for Arthur, 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^k per each round. (We can first make the original proof system have an even smaller soundness error, so this should not scare us.) So the question is what we gained by doing so. Intuitively, we gained not having to simulate the DGW 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 $2^{-\Omega(k)}$ fraction of the α's. Proving the last statement is the major technical task needed to justify our construction.

A precise statement of the properties of our random selection protocol is given in the following lemma:

Proposition 6.3.3 *There is an interactive protocol* $\mathrm{RS} = (\mathrm{M}_{RS}, \mathrm{A}_{RS})$ *with the following properties on input* $(1^\ell, 1^q, 1^k)$.

1. *(Efficiency) The protocol is polynomially bounded, public coin for both* M_{RS} *and* A_{RS}*, and both parties can be implemented in polynomial time. In addition, the protocol exchanges only four messages (starting with* A_{RS}*).*

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

3. *(Strong Simulability) There exists a polynomial-time black-box simulator* S_{RS} *such that for all deterministic*[5] *Arthur strategies* A^*_{RS}*, the statistical difference between the following distributions is* $\mathrm{poly}(q,\ell) \cdot 2^{-\Omega(k)}$*:*

   (I) *Execute* $(A^*_{RS}, M_{RS})(1^\ell, 1^q, 1^k)$*, let* $\alpha \in \{0,1\}^\ell$ *be the output of the protocol, and let* v *be* A^*_{RS}*'s view of the interaction. Output* (v, α)*.*

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

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). Observe that the strong simulability condition also implies that for any Arthur strategy A^*_{RS}, the output of the random selection protocol will have statistical difference at most $2^{-\Omega(k)}$ from uniform.

Proposition 6.3.3 will be proven in Section 6.4, after we show how it can be used to transform honest-verifier zero-knowledge proof systems into cheating-verifier ones. It is reduced to proving the following generalization of Lemma 5.4.10 which may be of independent interest:

Lemma 6.3.4 *There exists a universal constant,* $c > 0$*, so that the following holds, for every* $\varepsilon, \delta > 0$*. Let* $\mathcal{D}$ *and* $\mathcal{R}$ *be finite sets,* $\mathcal{H}$ *be a 2-universal family of hash functions from* $\mathcal{D}$ *to* $\mathcal{R}$*, and let* 0 *be any fixed element of* $\mathcal{R}$*. Let* $S \subseteq \mathcal{H}$ *such that* $|S| \geq \delta|\mathcal{H}|$*, and* X *be a random variable ranging over a finite set* $\mathcal{D}$ *having collision probability at most* $\frac{\varepsilon}{|\mathcal{R}|}$ (i.e., $\sum_{x \in \mathcal{D}} \Pr[X = x]^2 \leq \frac{\varepsilon}{|\mathcal{R}|}$)*. Then the statistical difference between the following two distributions is at most* $c \cdot \varepsilon^{1/c}\delta^{-c}$*.*

(A) *Choose* $h \leftarrow S$*, and select* x *according to* X *conditioned on* $h(x) = 0$*. Output* (h, x)*.*

(B) *Choose* $x \leftarrow X$*, and select* $h \leftarrow \{h' \in S \,:\, h'(x) = 0\}$ *Output* (h, x)*.*

[5] 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 the subsequent section, the coins of Arthur will have already been fixed by the outer simulator.

Actually, a special case of this lemma, where X is uniform over $\mathcal{D}$ (and $|\mathcal{R}| = \varepsilon \cdot |\mathcal{D}|$) suffices for the current proof of Theorems 6.3.1 and 6.3.2. The stronger version was developed for an alternative proof, discovered first, which is totally superseded by the current proof.

6.3.3 The transformation

Now we present our transformation of proof systems. The properties of the transformation are given in the following theorem:

Theorem 6.3.5 *Any honest-verifier public-coin statistical (resp., computational) zero-knowledge proof system can be transformed into a (cheating-verifier) public-coin statistical (resp., computational) zero-knowledge proof system. Furthermore,*

1. *The resulting proof system exchanges twice as many messages as the original one.*
2. *The resulting prover strategy can be implemented in probabilistic polynomial time given oracle access to the original prover strategy.*[6]
3. *The resulting proof system has completeness error* $2^{-\Omega(k)}$ *and soundness error* $1/k$*, where* k *is the security parameter. In case the original proof system has perfect completeness, so does the resulting one.*
4. *The resulting proof system has a black-box simulator.*
5. *In case of statistical zero-knowledge, the black-box simulator works for all verifiers and has simulator deviation* $\mathrm{poly}(k) \cdot \mu(k) + 2^{-\Omega(k)}$*, where* $\mu(k)$ *is the original simulator deviation.*

Theorem 6.3.1 follows from combining Theorem 6.3.5 with Theorem 5.4.15 (and renaming k). Theorem 6.3.2 follows by combining Theorem 6.3.5 with the result of Fürer *et. al.* [FGM$^+$89] that transforms public-coin honest-verifier zero-knowledge proofs into ones with perfect completeness. One important feature of the transformation given in Theorem 6.3.5 is that it preserves the computational complexity of the prover strategy. Hence, for cryptographic applications, Theorem 6.3.5 is probably most useful on its own, without combining it with the other transformations of Theorem 5.4.15 or [FGM$^+$89], as those transformations do not have this feature. We also note that Theorem 6.3.5 yields a proof system with nonnegligible soundness error $1/k$, which can be reduced further by doing *sequential* repetitions. This cannot be improved (while preserving the message-complexity of the transformation) unless **NP** $\subset$ **BPP**. This is because only problems in **BPP** have constant-message public-coin **CZK** proof systems with negligible soundness error [GK96b], whereas all of **NP** has constant-message public-coin **HVCZK** proof systems (with negligible soundness error) [GMW91].

[6] Again, we use the conventions given after Definition 6.1.4 regarding how a polynomial-time algorithm can make use of a more powerful oracle which may use a superpolynomial number of random coins. In this case, we need not worry about the messages being too long since the specified prover strategy always sends polynomial-length messages.

We now proceed to give the transformation. Let (M_0, A_0) be any public-coin honest-verifier zero-knowledge proof system for a promise problem Π. In what follows, we always assume that the security parameter k is at least the input length $|x|$; this can be achieved by artificially increasing k if necessary. Let $m = m(k)$ denote the number of messages exchanged by (M_0, A_0) on security parameter k and, in the case of statistical zero knowledge, let $\mu = \mu(k)$ denote the simulator deviation. By taking $\mathrm{poly}(k)$ parallel repetitions of (M_0, A_0), we obtain a zero-knowledge proof system (M, A) with the following properties on security parameter k:

1. $m(k)$ messages are exchanged.

2. The soundness error is $2^{-k\cdot m}$ and the completeness error is 2^{-k}.

3. In the case of statistical zero knowledge, simulator deviation is $\mathrm{poly}(k) \cdot \mu(k)$.

4. M can be implemented in probabilistic polynomial time with oracle access to M_0.

We now describe how to obtain a cheating-verifier proof system $(\widetilde{M}, \widetilde{A})$ by replacing A's messages in (M, A) with our random selection protocol. For notational convenience, we assume that (M, A) exchanges $m = 2r$ messages, with A sending the first message.[7] We also assume (wlog) that all the A-messages are of the same length $\ell = \ell(k)$. We denote the i'th A-message by α_i and the i'th M-message by β_i. Throughout what follows, we will often drop the input x and security parameter k from the notation. Having fixed these conventions, we give the transformed proof system $(\widetilde{M}, \widetilde{A})$ in Protocol 6.3.6.

Protocol 6.3.6: Transformed proof system $(\widetilde{M}, \widetilde{A})$

Input: Instance x of Π and security parameter k

1. Repeat for $i = 1, \ldots, r$:
   (a) $\widetilde{M}, \widetilde{A}$: Execute the random selection protocol RS on input $(1^\ell, 1^{2kr}, 1^k)$ to obtain an output $\alpha_i \in \{0,1\}^\ell$.
   (b) $\widetilde{M}$: Select $\beta_i \leftarrow M(\alpha_1, \beta_1, \alpha_2, \beta_2, \ldots, \alpha_i)$ and send β_i to $\widetilde{A}$.

2. $\widetilde{A}$: Accept or reject as A would on transcript $(\alpha_1, \beta_1, \ldots, \alpha_r, \beta_r)$.

We now prove that Protocol 6.3.6 satisfies the requirements of Theorem 6.3.5.

[7] This assumption that the number of messages in (M, A) is even only affects the claim about message complexity in Theorem 6.3.5. The case when (M, A) exchanges an odd number m of messages is similar, and actually yields message complexity better than claimed ($2m - 1$ rather than $2m$).

Efficiency. Since the random selection protocol RS consists of 4 messages, with M sending the last message (which can be sent together with β_i), $(\widetilde{M}, \widetilde{A})$ exchanges $4r = 2m$ messages. This proves Property 1. Property 2, the prover's complexity, is clear, given that M's strategy in the Random Selection Protocol can be executed in probabilistic polynomial time.

Completeness and Soundness. The claim about the completeness error in Property 3 follows from the fact that (M, A) has completeness error 2^{-k} and the fact that, when M behaves honestly in the random selection protocol RS, the output has statistical difference at most $\mathrm{poly}(\ell, 2kr, r) \cdot 2^{-\Omega(k)} = 2^{-\Omega(k)}$ from uniform (by the strong simulability). It is also immediate that if (M, A) has perfect completeness, then so does $(\widetilde{M}, \widetilde{A})$.

For soundness, consider any cheating strategy $\widetilde{M}^*$ in Protocol 6.3.6 and fix a NO instance x of Π (which we hide from the notation). We write $\langle \widetilde{M}^*, \widetilde{A} \rangle_{2i}$ to denote the distribution of $(\alpha_1, \beta_1, \ldots, \alpha_i, \beta_i)$ in $\langle \widetilde{M}^*, \widetilde{A} \rangle$, and $\langle \widetilde{M}^*, \widetilde{A} \rangle_{2i-1}$ for the same distribution without β_i. From $\widetilde{M}^*$, we construct a cheating strategy M^* for the original protocol (M, A) as follows: On partial conversation transcript $\gamma = (\alpha_1, \beta_1, \ldots, \alpha_i)$, M^* gives response β_i with probability

$$\Pr\left[\langle \widetilde{M}^*, \widetilde{A} \rangle_{2i} = (\gamma, \beta_i) | \langle \widetilde{M}^*, \widetilde{A} \rangle_{2i-1} = \gamma\right].$$

We define random variables $\langle M^*, A \rangle_j$ analogously to $\langle \widetilde{M}^*, \widetilde{A} \rangle_j$.

The main claim needed to establish soundness states that, with each execution of the random selection protocol, the advantage $\widetilde{M}^*$ has over M^* increases by a multiplicative factor of at most 2^k (plus an additive term of $\frac{1}{2kr}$).

Claim 6.3.7 *Let S be any set of partial conversation transcripts consisting of j messages. Then,*

$$\Pr\left[\langle \widetilde{M}^*, \widetilde{A} \rangle_j \in S\right] \leq 2^{\lceil j/2 \rceil \cdot k} \cdot \Pr\left[\langle M^*, A \rangle_j \in S\right] + \frac{\lceil j/2 \rceil}{2kr}.$$

Now, setting $j = 2r$ and S to be the setting of accepting conversations and recalling that (M, A) has soundness error smaller than 2^{-2kr}, the claim says that $\widetilde{A}$ accepts in $(\widetilde{M}^*, \widetilde{A})$ with probability at most $2^{kr} \cdot 2^{-2kr} + r/(2kr) \leq 1/k$. We now give the somewhat tedious proof of the claim.

Proof: We prove the claim by induction on j. For $j = 0$, the statement is trivial. Assume it is true for j and we will prove it for $j + 1$. For any partial conversation γ consisting of j messages, let

$$\delta_\gamma \stackrel{\text{def}}{=} \max\left\{0, \Pr\left[\langle \widetilde{M}^*, \widetilde{A} \rangle_j = \gamma\right] - 2^{\lceil j/2 \rceil \cdot k} \cdot \Pr\left[\langle M^*, A \rangle_j = \gamma\right]\right\}.$$

Then, applying the inductive hypothesis to the set T of γ for which $\delta_\gamma > 0$, we see that

$$\sum_\gamma \delta_\gamma = \Pr\left[(\widetilde{M}^*, \widetilde{A}^*) \in T\right] - 2^{\lceil j/2 \rceil \cdot k} \cdot \Pr\left[\langle M^*, A \rangle_j \in T\right] \leq \frac{\lceil j/2 \rceil}{2kr}.$$

Now, let S denote any set of $j+1$-message conversation transcripts. Then,

$$\Pr\left[\langle\widetilde{M}^*,\widetilde{A}\rangle_{j+1}\in S\right]=\sum_{\gamma}\Pr\left[\langle\widetilde{M}^*,\widetilde{A}\rangle_{j}=\gamma\right]\cdot\Pr\left[\langle\widetilde{M}^*,\widetilde{A}\rangle_{j+1}\in S\mid\langle\widetilde{M}^*,\widetilde{A}\rangle_{j}=\gamma\right]. \quad (6.2)$$

Consider the case when j is even. Then, in $\langle M^*, A\rangle$, the $(j+1)$'st message is chosen uniformly in $\{0,1\}^\ell$ by A, and in $\langle\widetilde{M}^*,\widetilde{A}\rangle$, it is generated via the random selection protocol. Thus, by the soundness property of the random selection protocol, the following holds for any partial transcript γ:

$$\Pr\left[\langle\widetilde{M}^*,\widetilde{A}\rangle_{j+1}\in S\mid\langle\widetilde{M}^*,\widetilde{A}\rangle_{j}=\gamma\right]\le 2^k\cdot\Pr\left[\langle M^*,A\rangle_{j+1}\in S\mid\langle M^*,A\rangle_j=\gamma\right]+\frac{1}{2kr}.$$

Plugging this into Expression 6.2, we get:

$$\begin{aligned}
&\Pr\left[\langle\widetilde{M}^*,\widetilde{A}\rangle_{j+1}\in S\right]\\
&\le \sum_{\gamma}\left(\Pr\left[\langle\widetilde{M}^*,\widetilde{A}\rangle_{j}=\gamma\right]-\delta_\gamma\right)\cdot\left(\Pr\left[\langle\widetilde{M}^*,\widetilde{A}\rangle_{j+1}\in S\mid\langle\widetilde{M}^*,\widetilde{A}\rangle_{j}=\gamma\right]-\frac{1}{2kr}\right)\\
&\quad+\sum_{\gamma}\delta_\gamma\cdot\Pr\left[\langle\widetilde{M}^*,\widetilde{A}\rangle_{j+1}\in S\mid\langle\widetilde{M}^*,\widetilde{A}\rangle_{j}=\gamma\right]+\sum_{\gamma}\frac{1}{2kr}\cdot\Pr\left[\langle\widetilde{M}^*,\widetilde{A}\rangle_{j}=\gamma\right]\\
&\le \sum_{\gamma}\left(2^{\lceil j/2\rceil\cdot k}\cdot\Pr\left[\langle M^*,A\rangle_j=\gamma\right]\right)\cdot\left(2^k\cdot\Pr\left[\langle M^*,A\rangle_{j+1}\in S\mid\langle M^*,A\rangle_j=\gamma\right]\right)\\
&\quad+\frac{\lceil j/2\rceil}{2kr}+\frac{1}{2kr}\\
&= 2^{\lceil (j+1)/2\rceil k}\cdot\Pr\left[\langle M^*,A\rangle_{j+1}\in S\right]+\frac{\lceil (j+1)/2\rceil}{2kr}.
\end{aligned}$$

The case when j is odd is similar, but simpler. Instead of using the soundness of the random selection protocol, we use the fact that M^* generates message $j+1$ according to the same marginal distribution as $\widetilde{M}^*$. ■

Zero knowledge. Let S be the honest-verifier simulator for the original protocol (M, A). In Algorithm 6.3.8, we give a universal simulator $\widetilde{S}$ for $(\widetilde{M},\widetilde{A})$ which uses any verifier strategy $\widetilde{A}^*$ as a black-box.

To prove that the simulator has the desired properties, we first consider its output distribution in the case that the original honest-verifier simulator S is perfect: Let $\overline{S}^{\widetilde{A}^*}$ be the output distribution of $\widetilde{S}^{\widetilde{A}^*}$ if the output of S in Step 2 is replaced with a true sample $(\alpha_1,\beta_1,\ldots,\alpha_r,\beta_r)$ of $\langle M,\widetilde{A}^*\rangle$.

Claim 6.3.9 *$\overline{S}^{\widetilde{A}^*}(x)$ and $\langle\widetilde{M},\widetilde{A}^*\rangle(x)$ have statistical difference at most $2^{-\Omega(k)}$.*

Proof: Let us consider what happens in both the interaction between $\widetilde{M}$ and $\widetilde{A}^*$ and in the simulator $\overline{S}^{\widetilde{A}^*}$ conditioned on a partial transcript $\gamma_i=(t_1,\alpha_1,\beta_1,\ldots,t_i,\alpha_i,\beta_i)$. Let $A^{(i+1)}$ be $\widetilde{A}^*$ with history γ_i. The process by which t_{i+1} and α_{i+1} are obtained in the interaction

Algorithm 6.3.8: The simulator $\widetilde{S}^{\widetilde{A}^*}$ for Protocol 6.3.6

Input: An instance x of Π, a security parameter k, and oracle access to a cheating verifier strategy $\widetilde{A}^*$.

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

between $\widetilde{M}$ and $\widetilde{A}^*$ is exactly Distribution (I) in the strong simulability condition of the random selection protocol (Proposition 6.3.3), taking A_{RS}^* to be $A^{(i+1)}$. Now, in $\overline{S}^{\widetilde{A}^*}$, each α_{i+1} is uniform and independent of $(\alpha_1, \beta_1, \ldots, \alpha_i, \beta_i)$ (and thus also of γ_i). Therefore, the process by which t_{i+1} and α_{i+1} are obtained in $\overline{S}^{\widetilde{A}^*}$ 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 $\mathrm{poly}(\ell, 2kr)\cdot 2^{-\Omega(k)} = 2^{-\Omega(k)}$. Moreover, β_{i+1} is chosen according to the same distribution (conditioned on γ_i, t_{i+1} and α_{i+1}) in both $\langle \widetilde{M}, \widetilde{A}^* \rangle$ and $\overline{S}^{\widetilde{A}^*}$ — that is, according to the original M strategy. So β_{i+1} does not increase the statistical difference. Thus for every triple (t_i, α_i, β_i), the statistical difference accumulates by at most $2^{-\Omega(k)}$, for a total of $r \cdot 2^{-\Omega(k)} = 2^{-\Omega(k)}$. ■

Now we deduce Theorem 6.3.5, Items 4 and 5, from Claim 6.3.9.

Statistical zero knowledge. Using the output of S instead of a true sample from (M, A) can increase the simulator deviation by at most $\mathrm{StatDiff}(S, \langle M, A \rangle)$, which is exactly the simulator deviation for the protocol (M, A), which in turn is at most $\mathrm{poly}(k)$ times the simulator deviation for the original proof system (M_0, A_0).

Computational zero knowledge. We need to show that the probability ensembles $X_1 \stackrel{\text{def}}{=} \{\langle \widetilde{M}, \widetilde{A}^* \rangle(x, 1^k)\}_{x \in \Pi_Y, k \in \mathbb{N}}$ and $X_2 \stackrel{\text{def}}{=} \{\widetilde{S}^{\widetilde{A}^*}(x, 1^k)\}_{x \in \Pi_Y, k \in \mathbb{N}}$ are computationally indistinguishable for any probabilistic polynomial-time $\widetilde{A}^*$. Consider a third ensemble $X_3 \stackrel{\text{def}}{=}$

$\{\overline{S}^{\widetilde{A}^*}(x,1^k)\}_{x\in\Pi_Y,k\in\mathbb{N}}$. By Claim 6.3.9, 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 $\widetilde{A}^*$. This holds because X_2 and X_3 are obtained by performing the same probabilistic polynomial-time computation on the computationally indistinguishable ensembles $\{S(x,1^k)\}_{x\in\Pi_Y,k\in\mathbb{N}}$ and $\{(M,A)(x,1^k)\}_{x\in\Pi_Y,k\in\mathbb{N}}$, respectively.

6.4 Random selection

In this section, we describe our random selection protocol and prove Proposition 6.3.3. Our protocol builds on an earlier protocol of Damgård, Goldreich, and Wigderson [DGW94], which we describe now. Their protocol takes two parameters s and q and produces an element of $\{0,1\}^s$ as output. Informally, the protocol works as follows: First, A chooses a (succinctly described) partition of $\{0,1\}^s$ into cells of size $\mathrm{poly}(s,q)$. Then, M chooses a cell uniformly from the partition. Lastly, A uniformly selects an element of that cell, which is the output. These "partitions" are implemented using a family $\mathcal{F}_{s,q}$ of hash functions mapping $\{0,1\}^s$ to $\{0,1\}^t$, for $t = s - 4\log_2(3qs)$. The properties of this family of functions are given in the following lemma.

Lemma 6.4.1 *For every pair of integers $s,q\in\mathbb{N}$, there is a family of functions $\mathcal{F}_{s,q}$ mapping $\{0,1\}^s$ to $\{0,1\}^t$, for $t = s-4\log_2(3qs)$, with the following properties:*

1. *Each $f\in\mathcal{F}_{s,q}$ has a description of size* $\mathrm{poly}(s,q)$.
2. *There is a* $\mathrm{poly}(s,q)$*-time algorithm that, on input $f\in\mathcal{F}_{s,q}$ and $x\in\{0,1\}^s$, outputs $f(x)$.*
3. *There is a* $\mathrm{poly}(s,q)$*-time algorithm that, on input $f\in\mathcal{F}_{s,q}$, $y\in\{0,1\}^t$, lists all the elements of $f^{-1}(y)$. In particular, $|f^{-1}(y)|\le p(s,q)$ for some polynomial p.*
4. *For every $y\in\{0,1\}^t$ and $f\in\mathcal{F}_{s,q}$, $f^{-1}(y)$ is nonempty.*
5. *$\mathcal{F}_{s,q}$ is a family of almost s-wise independent hashing functions in the following sense: For every s distinct points $x_1,\ldots,x_s\in(\{0,1\}^s\setminus\{0,1\}^t0^{s-t})$, for a uniformly chosen $f\in\mathcal{F}_{s,q}$, the random variables $f(x_1),\ldots,f(x_s)$ are independently and uniformly distributed in $\{0,1\}^t$.*

Such a family can essentially be obtained by associating $\{0,1\}^s$ with $\mathrm{GF}(2^s)$ and taking all polynomials of degree $s-1$ over this field, with the output of the polynomials being truncated to t bits. The details of the construction can be found in [DGW94]. We can view each $f\in\mathcal{F}_{s,q}$ as defining a partition of $\{0,1\}^s$ into 2^t cells of the form $f^{-1}(y)$, each of size $\mathrm{poly}(s,q)$. For notational convenience, we will sometimes write *cell y* to refer to the cell $f^{-1}(y)$. A formal description of the DGW random selection protocol is given in Protocol 6.4.2.

In [DGW94], it was shown that Protocol 6.4.2 has the following properties (roughly speaking):

Protocol 6.4.2: DGW random selection protocol
DGW $= (M_{DGW}, A_{DGW})$ **[DGW94]**

Input: Parameters s and q (in unary)

1. A_{DGW}: Select $f \leftarrow \mathcal{F}_{s,q}$ and send it to M_{DGW} (*i.e.,* select a random partition).[a]
2. M_{DGW}: Select $y \leftarrow \{0,1\}^t$, and send it to A_{DGW} (*i.e.,* uniformly select a cell).
3. A_{DGW}: Select $x \leftarrow f^{-1}(y)$ (*i.e.,* uniformly select an element of the cell).

Output: x

[a] 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.

1. (Soundness) For any Merlin strategy M^*_{DGW}, the output distribution on $\{0,1\}^s$ 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 $1/\text{poly}(s,q)$ fraction of the elements $x \in \{0,1\}^s$ occur as possible outputs of the interaction (M_{DGW}, A^*_{DGW}) and given such an x, one can simulate in $\text{poly}(s,q)$-time A^*_{DGW}'s view of an interaction resulting in x.

The main hindrance in applying the protocol as used by [DGW94] is that the simulator is only guaranteed to work for a $1/\text{poly}(s,q)$ fraction of the x's. The new technique of this paper is to interpret the output x of the DGW protocol as a set of 2^k strings, from which a single string α is randomly selected by Merlin. It is this α, rather than x, that is the output of the random selection protocol. The family of sets of size 2^k that we use has the crucial property that every subset of them of density at least $1/\text{poly}(s,q)$ will still cover all but an exponentially vanishing fraction of α's. Because of this, we will be able to simulate the protocol for all but an exponentially vanishing fraction of the α's.

In order to define our sets of 2^k strings, we use 2-universal hash functions. Recall that for every pair of integers ℓ and m, we defined $\mathcal{H}_{\ell,m}$ to be the 2-universal family of all affine-linear (over GF(2)) functions from $\{0,1\}^\ell$ to $\{0,1\}^m$. Each such function is of the form $h(x) = Ax + b$, where A is an $m \times \ell$ matrix over GF(2) and b is an element of $\{0,1\}^m$, so elements of $\mathcal{H}_{\ell,m}$ can be uniquely represented by strings of length $m \cdot (\ell+1)$.

Our protocol takes three parameters ℓ, q, and k as input and produces an element of $\{0,1\}^\ell$ as output. The two parties use the DGW protocol to select an element h of $\mathcal{H} = \mathcal{H}_{\ell,\ell-k}$, and then Merlin selects the output uniformly from $h^{-1}(0)$. That is, the DGW random selection protocol is called with parameters s and q, where $s = (\ell-k)\cdot(\ell+1)$, so that its s-bit output can be interpreted as an element of $\mathcal{H}_{\ell,\ell-k}$. A full description of the protocol is given in Protocol 6.4.3.

Protocol 6.4.3: Our random selection protocol $\mathrm{RS} = (\mathrm{M}_{RS}, \mathrm{A}_{RS})$

Input: parameters ℓ, q, and k (in unary)

1. M_{RS}, A_{RS}: Set $s = (\ell - k) \cdot (\ell + 1)$, $t = s - 4\log_2(3qs)$, and $\mathcal{H} = \mathcal{H}_{\ell,\ell-k}$.
2. A_{RS}: Select $f \leftarrow \mathcal{F}_{s,q}$ and send it to M_{RS}.
3. M_{RS}: Select $y \leftarrow \{0,1\}^t$, and send it to A_{RS}.
4. A_{RS}: Select $h \leftarrow f^{-1}(y)$ and sent it to M_{RS}.
5. M_{RS}: Select $\alpha \leftarrow h^{-1}(0)$, viewing h as an element of $\mathcal{H}$. (If $h^{-1}(0) = \emptyset$ then α is defined to be 0^ℓ.)

Output: α

We now prove that Protocol 6.4.3 satisfies Proposition 6.3.3. Efficiency is immediate from the description of the protocol.

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 T is bounded above by the probability that $h^{-1}(0)$ contains an element of T. Now, for h chosen *uniformly* from $\mathcal{H}$ (instead of by the protocol), the probability that $h^{-1}(0)$ contains an element of T is at most

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

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. Recall that $p = p(s,q)$ is polynomial bound on the size of $f^{-1}(y)$ for any $f \in \mathcal{F}_{s,q}$, s is the description length for elements of $\mathcal{H} = \mathcal{H}_{\ell,\ell-k}$, and functions in $\mathcal{F}_{s,q}$ map $\{0,1\}^s$ to $\{0,1\}^t$, where $t = s - 4\log_2(3qs)$. For $\alpha \in \{0,1\}^\ell$, we write $\mathcal{H}_\alpha \stackrel{\text{def}}{=} \{h \in \mathcal{H} : h(\alpha) = 0\}$. With these notations, the simulator is given in Algorithm 6.4.4.

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

Let us now show that Distributions (I) and (II) in Proposition 6.3.3 have statistical

Algorithm 6.4.4: The random selection protocol simulator $S_{RS}^{A_{RS}^*}$

Input: Parameters ℓ, q, and k (in unary), $\alpha \in \{0,1\}^\ell$, and oracle access to A_{RS}^*

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

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

   (a) Choose h' uniformly from $\mathcal{H}_\alpha$.

   (b) Let $y = f(h')$ (*i.e.*, y is the cell containing h'). Compute $i \stackrel{\text{def}}{=} |f^{-1}(y) \cap \mathcal{H}_\alpha|$. With probability $1 - \frac{1}{i}$, 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 6.4.3 after being sent y in Step 6.4.3.

   (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`.

difference $\mathrm{poly}(s,q) \cdot 2^{-\Omega(k)}$. 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 Appendix B.

Lemma 6.4.5 *There exists a universal constant $c > 0$, so that the following holds: Let $\mathcal{H} = \mathcal{H}_{\ell,m}$ be the family of affine-linear maps from $\mathcal{D} = \{0,1\}^\ell$ to $\mathcal{R} = \{0,1\}^m$. Let $S \subset \mathcal{H}$ be such that $|S| \geq \delta|\mathcal{H}|$. Let $\varepsilon = \frac{|\mathcal{R}|}{|\mathcal{D}|}$. Then*

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

- **(A)** *Choose $h \leftarrow S$. Select $x \leftarrow h^{-1}(0)$. Output (h, x).*
- **(B)** *Choose $x \leftarrow \mathcal{D}$. Select $h \leftarrow S \cap \mathcal{H}_x$. Output (h, x).*

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

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

When we apply the lemma, we take $m = \ell - k$, $\varepsilon = 2^{-k}$, and $S = \{A^*_{RS}(y) \;:\; y \in \{0,1\}^t\}$. In other words, S is the set all possible *cell representatives* that A^*_{RS} can send in Step 6.4.3 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, $c \cdot \varepsilon^{1/c}\delta^{-c} = \mathrm{poly}(\ell, q, k) \cdot 2^{-\Omega(k)}$. 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 6.4.5. Now we will show that the Distribution (II′) is statistically close to Distribution (B).

Let us consider a single iteration of Step S2 in $S^{A^*_{RS}}_{RS}$. 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 nearly 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 $h' \leftarrow \mathcal{H}_\alpha$ is exactly

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

In Step 5b, any such choice is maintained with probability $1/|f^{-1}(y_0) \cap \mathcal{H}_\alpha|$. Thus the probability that $y = y_0$ after Steps 5a and 5b 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 5b.)

Now, since, in Step 5c, $h = A^*_{RS}(y)$ 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 5d, 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 6.4.5. 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 iterations is $2^{-\Omega(k)}$ for at least a $1 - 2^{-\Omega(k)}$ fraction of the α's in $\{0,1\}^\ell$. We do this by showing that each iteration produces output with probability at least k 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 5b and 5d. Observe that the simulator never exits in Step 5d if h' chosen in Step 5a lies in S, because then h will equal h'. This occurs with probability

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

By Lemma 6.4.5, for at least a $1 - 2^{-\Omega(k)}$ 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 5b is at least $1/|f^{-1}(y)|$, which is at least $1/p$ by the properties of the family $\mathcal{F}_{s,q}$. Thus, for a $1 - 2^{-\Omega(k)}$ fraction of the α's, a single iteration produces output with probability at least $1/(2(3sq)^4 \cdot p)$. Since there are $(2(3sq)^4 \cdot p) \cdot k$ iterations, output is produced with probability $1 - 2^{-\Omega(k)}$.

We have shown that Distribution (I$'$) is identical to Distribution (A) in Lemma 6.4.5 and Distribution (II$'$) has a statistical difference of $2^{-\Omega(k)}$ from Distribution (B). So, by Lemma 6.4.5, we conclude that Distributions (I) and (II) have statistical difference $2^{-\Omega(k)}$ and strong simulability is established.

6.5 Corollaries and open problems

We can use our transformation to translate many of the results about **HVSZK** to **SZK**. Some of the results about **HVSZK** were already implicitly translated when we used Theorem 5.4.15 as the starting point for the proof of Theorem 6.3.1. The nice properties of the proof system given by Theorem 5.4.15, such as public coins, perfect completeness, and exponentially small simulator deviation, are all preserved by our transformation and therefore appear in the statement of Theorem 6.3.1.

A number of additional results that can be immediately translated are those that just refer to properties of **HVSZK** *as a class of promise problems*; these now apply to **SZK** simply by the equality **HVSZK** = **SZK**:

Corollary 6.5.1 *Properties of* **SZK***:*

1. Entropy Difference *and* Statistical Difference *are complete for* **SZK**.
2. **SZK** *is closed under complement.*
3. *For every promise problem* Π, $\Phi(\Pi) \in \mathbf{SZK}$.[8]
4. **SZK** *is closed under* $\mathbf{NC}_1$ *truth-table reductions.*
5. **weak-SZK** = **SZK**.

For Item 5, we define **weak-SZK** via the obvious analogy to **weak-HVSZK**, and apply the chain of inclusions

$$\textbf{weak-SZK} \subset \textbf{weak-HVSZK} = \textbf{HVSZK} = \textbf{SZK} \subset \textbf{weak-SZK}.$$

We have omitted analogues of some of the results that only refer to the class **HVSZK** simply because the honest-verifier version of the result is the stronger one. This is the case with upper bounds on the complexity of **HVSZK**, such as Corollary 4.2.2 and Theorem 4.8.4, since the inclusion **SZK** $\subset$ **HVSZK** is obvious even without Theorem 6.3.1.

The equality **HVSZK** = **SZK** also has implications for knowledge complexity in the hint sense via Lemma 4.6.7. Specifically, if we define $\mathbf{SKC}^*_{\mathbf{hint}}(\kappa(n))$ to be the cheating-verifier version of the class $\mathbf{SKC}_{\mathbf{hint}}(\kappa(n))$, then Theorem 6.3.1 and Lemma 4.6.7 have the following consequence:

Corollary 6.5.2 *For every polynomially bounded function* $\kappa : \mathbb{N} \to \mathbb{N}$, $\mathbf{SKC}_{\mathbf{hint}}(\kappa(n)) = \mathbf{SKC}^*_{\mathbf{hint}}(\kappa(n))$*. Moreover, every problem in these classes has an interactive proof of statistical knowledge complexity* $\kappa(n)$ *in the hint sense* against cheating verifiers *with the following properties:*

1. *Black-box simulation with simulator deviation* 2^{-k} *for all verifiers.*
2. *The same hint function can be used for all verifiers.*
3. *Perfect completeness.*
4. *The proof system is public coin.*

By this equality of the $\mathbf{SKC}_{\mathbf{hint}}$ and $\mathbf{SKC}^*_{\mathbf{hint}}$ hierarchies, it follows that the $\mathbf{SKC}^*_{\mathbf{hint}}(\kappa(n))$ hierarchy must also collapse by logarithmic terms, as in Theorem 4.6.11.

For computational zero knowledge, we can combine Theorems 5.4.16 and 6.3.2 to obtain:

[8] For a definition of $\Phi(\cdot)$ and $\mathbf{NC}_1$ truth-table reductions, see Section 4.5

Corollary 6.5.3 *Every problem that has a 3-message honest-verifier computational zero-knowledge proof also has cheating-verifier computational zero-knowledge proof (which is public coin, has a black-box simulator, and has perfect completeness).*

Clearly, the main outstanding question about computational zero knowledge in this regard is whether a transformation can be given for all of **HVCZK** (unconditionally, of course).

Open Problem 6.5.4 *Does* **HVCZK** = **CZK***?*

A positive answer to Open Problem 5.4.19 would imply a positive answer to this problem.

It is important to note that several of our results about honest-verifier statistical zero-knowledge proofs do not translate to cheating-verifier proofs. For one, we do not obtain true cheating-verifier analogues of the results on the perfect knowledge complexity of **HVSZK** in Theorem 4.6.13, since we do not know how to relate the honest-verifier and cheating-verifier versions of the **PKC** classes.

A more significant result that does not translate is Corollary 4.1.1, which says that every problem in **HVSZK** has a *constant-message* **HVSZK** proof system (with additional nice properties). Even though the main transformation presented in this chapter preserves message complexity upto a constant factor (Theorem 6.3.5), to obtain a result for all of **HVSZK** we first had to apply the private-to-public coin transformation of Theorem 5.4.15, which does not preserve message complexity. The **HVSZK**-to-**SZK** transformation of Bellare, Micali, and Ostrovsky [BMO90b] *does* preserve message complexity (and applies directly to private-coin proofs), but it relies on an intractability assumption. Applying their transformation to the proof systems of Corollary 4.1.1, one obtains:

Proposition 6.5.5 *If the* Discrete Logarithm *problem is hard, then every problem in* **HVSZK** *has a (cheating-verifier) statistical zero-knowledge proof system with the following properties:*

1. *The proof system exchanges a constant number of messages.*
2. *Black-box simulation (for polynomial-time verifiers).*
3. *Completeness error and soundness error* 2^{-k}*.*

However, to obtain constant-message **SZK** proof systems *unconditionally* is still open.

Open Problem 6.5.6 *Does every problem in* **HVSZK** *have a constant-message* **SZK** *proof system?*

A positive answer to Open Problem 5.4.20 would also imply a positive answer to this problem, using even just the transformation of [DGW94].

Another property given by Corollary 4.1.1 that does not translate to the cheating-verifier proofs is the fact that the prover is deterministic. This is inevitable, as only **BPP** has cheating-verifier zero-knowledge proofs with a deterministic prover [GO94].

Chapter 7

Noninteractive SZK

Iteraction is at once a blessing and a curse for zero-knowledge proofs. On one hand, interaction is one of the ingredients that makes the seemingly paradoxical notion of zero knowledge feasible. On the other hand, in many cryptographic applications where one would like to use zero-knowledge proofs, interaction is either too expensive or completely unavailable. While considerable research has been devoted to reducing the amount of interaction in zero-knowledge proofs (*cf.,* Corollary 4.1.1, [FS89, BMO90a, GK96a, Oka96]), it cannot be completely removed in the GMR paradigm of a proof system. Indeed, Goldreich and Oren [GO94] have shown that GMR zero knowledge becomes trivial (*i.e.,* exists only for problems in **BPP**) if one requires that the proofs are noninteractive (*i.e.,* with only unidirectional communication).

Suprisingly, however, Blum, Feldman, and Micali [BFM88] showed that by augmenting the model slightly, it is possible to achieve zero knowledge in a noninteractive setting. Specifically, they assume that the 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 proven and the reference string) and sent from the prover to the verifier.

As in the interactive case, the zero-knowledge property is formalized by requiring that there is a probabilistic polynomial-time simulator whose output distribution is "close" to the verifier's view of the proof system (which now consists of the shared reference string and the proof sent by the prover). Various interpretations of "close" give rise to three variants of noninteractive zero knowledge proofs — perfect, statistical, and computational — defined analogously the interactive case. (Formal definitions will be given in Section 7.1.)

Noninteractive zero-knowledge proofs, on top of being more communication efficient by definition, 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], and public-key cryptosystems secure against chosen-ciphertext attack [BFM88, NY90, DDN91].

Until recently, most of the work on noninteractive zero knowledge has focused on the computational type (*cf.,* [BFM88, DMP87, DMP88, BDMP91, FLS99, KP98]). This is probably due to the early results which showed that all of **NP** has noninteractive computational zero knowledge proofs (under various assumptions [BFM88, DMP87, FLS99]), and the ensuing cryptographic applications [BFM88, NY90, BG89]. In contrast, for a long time the only (nontrivial) noninteractive *statistical* zero-knowledge proofs known were the one for QUADRATIC NONRESIDUOSITY [BDMP91] and variants of it [DDP94, DDP97], and hence the study of such proofs was rather limited.[1]

In this chapter, we shall see that noninteractive statistical zero knowledge is richer than might have been expected. Our first step towards demonstrating this is to exhibit two natural complete problems for **NISZK**, the class of problems possessing noninteractive statistical zero-knowledge proofs. This builds on earlier work of De Santis, Di Crescenzo, Persiano, and Yung [DDPY98], who exhibited the first complete problem for **NISZK**. The key feature of these complete problems is that they are natural restrictions of our complete problems for **SZK**, STATISTICAL DIFFERENCE and ENTROPY DIFFERENCE. Thus, we can use these problems to relate **SZK** and **NISZK**. Specifically, we show that if **SZK** is nontrivial, then so is **NISZK**, where by nontrivial we mean that the class contains problems outside of **BPP**. Recall that the hypothesis holds under various assumptions, such as the intractability of the DISCRETE LOGARITHM [GK93] problem or approximate versions of the SHORTEST VECTOR and CLOSEST VECTOR problems for lattices [GG98a]. By our result, under any of these assumptions, **NISZK** is also nontrivial, even though no versions of these problems were known to be in **NISZK**. Furthermore, we shed light on the question of whether **SZK** = **NISZK**, *i.e.,* whether all statistical zero-knowledge proofs can be made noninteractive. Namely, we show that **SZK** = **NISZK** if (and only if) **NISZK** is closed under complement. We note that [DDPY98] have claimed that **NISZK** is closed under complement, but this claim has been retracted [DDPY99].

Organization. In Section 7.1, we give the formal definitions of noninteractive zero-knowledge proofs and discuss some of the issues that arise in the definitions. In Section 7.2, we introduce the problems ENTROPY APPROXIMATION and STATISTICAL DIFFERENCE FROM UNIFORM, and state our Completeness Theorem for **NISZK**, which asserts that these two problems are complete for **NISZK**. The proof of the Completeness Theorem comes in Sections 7.3 and 7.4. In Section 7.5, we use the complete problems to study the relationship between **SZK** and **NISZK**. Section 7.6 contains some additional applications of the Completeness Theorem for **NISZK**.

7.1 The noninteractive model

We begin by defining noninteractive proof systems in the shared random string model.

Definition 7.1.1 (shared random string model) *A* noninteractive protocol in the shared random string model *os a pair of probabilistic algorithms (A, B) together with a polynomial-time computable function $\ell : \{0,1\}^* \to \mathbb{N}$. The communication from A to B on common input x, denoted $(A, B)(x)$,*[2] *is the following probabilistic experiment:*

[1] An exception is an unpublished manuscript of Bellare and Rogaway [BR90], which contains a noninteractive perfect zero-knowledge proof for the language of graphs with trivial automorphism group, along with some basic results about noninteractive perfect zero knowledge.

[2] We use the same notation as for interactive protocols, but it will always be clear from context which we are referring to. Strictly speaking, ℓ should also be included in the notation, but it too will always be clear from context.

1. *Select the* shared random string $\sigma \leftarrow \{0,1\}^{\ell(x)}$.
2. *Let* $m \leftarrow A(x, \sigma)$.
3. *Let answer* $\leftarrow B(x, \sigma, m)$.

If answer $=$ `accept` *(resp., answer* $=$ `reject`*), we say that* B accepts *(resp.,* rejects*). We say that* (A, B) *is* polynomially bounded *if* $\ell(x)$ *and* $|m|$ *are both bounded above by a polynomial in* $|x|$. B*'s* view *of* $(A, B)(x)$ *is the random variable* (σ, m).

The key features of the above definition are that both parties A and B have access to the random string σ, and B does not send any messages to A. Given this communication model, proofs and zero-knowledgeness are completely analogous to the interactive case.

Definition 7.1.2 (noninteractive proofs) *Let* P *and* V *be probabilistic algorithms and let* Π *be a promise problem.* (P, V) *is said to be an* noninteractive proof system (in the shared random string model) *for* Π *with* completeness error $c : \mathbb{N} \rightarrow [0,1]$*, and* soundness error $s : \mathbb{N} \rightarrow [0,1]$ *if the following conditions hold:*

1. *(Efficiency)* (P, V) *is polynomially bounded and* V *is polynomial-time computable.*
2. *(Completeness) If* $x \in \Pi_Y$*, then* V *accepts with probability at least* $1 - c(k)$ *in* $(P, V)(x, 1^k)$.
3. *(Soundness) If* $x \notin \Pi_Y$*, then* V *rejects with probability at least* $1 - s(k)$ *in* $(P, V)(x, 1^k)$.

We require that $c(k)$ *and* $s(k)$ *be computable in time* $\mathrm{poly}(k)$ *and that* $1 - c(k) > s(k) + 1/\mathrm{poly}(k)$*. If* $c \equiv 0$*, then we say that the proof system has* perfect completeness.

Definition 7.1.3 (noninteractive zero knowledge — NISZK, NIPZK) *A noninteractive proof system* (P, V) *for a promise problem* Π *is said to be* statistical zero knowledge *if there is a useful*[3] *probabilistic polynomial-time algorithm* S *and a negligible function* $\mu(\cdot)$ *such that for all* $x \in \Pi_Y$ *and all* $k > 0$, $\widetilde{S}(x, 1^k)$ *has statistical difference at most* $\mu(k)$ *from* V*'s view of* $(P, V)(x, 1^k)$*. The negligible function* μ *is called the* simulator deviation*. If* $\mu \equiv 0$*, then* (P, V) *is said to be* perfect zero knowledge. **NISZK** *(resp.,* **NIPZK***) denotes the class of promise problems possessing noninteractive statistical (resp., perfect) zero-knowledge proofs.*

Noninteractive computational zero knowledge (**NICZK**) is defined analogously, replacing statistical closeness with computational indistinguishability, as in Definition 2.3.7.

Note that noninteractive zero knowledge is closed under parallel repetition, so the completeness and soundness errors can always be made exponentially small. (The problems that arise with parallel repetition in interactive zero knowledge come from cheating verifiers, but there is no way for a verifier to cheat when there is no interaction.) In fact, it is shown in [BDMP91, BR90] that every noninteractive zero knowledge proof can be transformed into one with *perfect* completeness.

[3] Recall that a probabilistic algorithm A is called *useful* if $\Pr[A(x) = \mathtt{fail}] \leq 1/2$ for all x and $\widetilde{A}(x)$ denotes the output distribution of A on input x, conditioned on $A(x) \neq \mathtt{fail}$.

7.1.1 Relationship with the interactive proofs

It is easy to see that that noninteractive proofs are equivalent to 2-message public-coin interactive proofs, as the shared random string can play the role of the verifier's single random message (and conversely). Similarly, we see that each of the three types of noninteractive zero-knowledge proofs (perfect, statistical, and computational) are equivalent to the analogous types of 2-message public-coin *honest-verifier* zero-knowledge proofs. Hence, we have **NICZK** $\subset$ **HVCZK**, **NIPZK** $\subset$ **HVPZK**, and **NISZK** $\subset$ **HVSZK** $=$ **SZK**.

Without a zero knowledge constraint, the expressive power of noninteractive proof systems actually extends to all of **AM**; that is, the class of problems possessing constant-message private-coin interactive proofs (rather than just 2-message public-coin proofs). This follows from the transformation from private coins to public coins of Goldwasser and Sipser [GS89] (which preserves the number of messages exchanged up to an additive constant) and the Collapse Theorem of Babai and Moran [BM88] (which reduces the number of messages in any constant-message public-coin proof system to two).

Like its interactive counterpart, noninteractive computational zero knowledge "hits the roof" under an intractability assumption. Namely, it has been shown that **NICZK** $=$ **AM** under successively weaker intractability assumptions and ultimately one-way permutations [BFM88, BDMP91, FLS99].

7.1.2 Contrast with the original definitions

Our definitions of noninteractive zero knowledge are stricter than those of Blum *et. al.* [BFM88, BDMP91] in the same way that our definitions of interactive zero knowledge are stricter than the GMR definition. First, we require the simulators to run in strict (rather than expected) polynomial time, but allow a failure probability. Second, we use a separate security parameter, rather than the input length, to control the error parameters; this has also been done in a number of previous works on noninteractive zero knowledge [FLS99, Kil94, KP98]. As in the interactive case, the use of a security parameter has the nice consequence that noninteractive zero knowledge is closed under Karp reductions.

Proposition 7.1.4 *If Π has a noninteractive statistical zero-knowledge proof with simulator deviation $\mu(\cdot)$, and Γ (Karp-)reduces to Π, then Γ has a noninteractive statistical zero-knowledge proof with simulator deviation $\mu(\cdot)$. Thus,* **NISZK** *and* **NIPZK** *are closed under (Karp) reductions.*

Analogous to Definition 2.4.2, we define **weak-NISZK** to capture the ways in which the original definitions are weaker than ours.

Definition 7.1.5 (weak-NISZK)
A noninteractive proof system (P, V) for a promise problem Π is said to be weak statistical zero knowledge *if for every $c > 0$, there is probabilistic polynomial-time algorithm S_c such that for all but finitely many $x \in \Pi_Y$, $S_c(x)$ has statistical difference at most $1/|x|^c$ from V's view of $(P, V)(x, 1^{|x|})$.* **weak-NISZK** *denotes the class of promise problems possessing weak noninteractive statistical zero-knowledge proofs.*

Later in this chapter, we will prove that **weak-NISZK** = **NISZK**, so our results about **NISZK** (as we've defined it) also apply to **NISZK** as defined by [BFM88, BDMP91]. One other minor difference between our definition and that of Blum *et. al.* is that we allow the verifier to be probabilistic, whereas they require it to be deterministic. We feel that allowing a probabilistic verifier maintains the spirit of noninteractive zero knowledge. In any case, a probabilistic verifier can always be made deterministic by having the verifier use part of the shared random string in place of its random coin flips (in combination with standard error reduction via parallel repetition and majority/threshold rule).

7.1.3 Augmentations to the definitions

In applications, one often needs noninteractive zero-knowledge proofs that have additional properties beyond those guaranteed by Definition 7.1.3. For completeness, we briefly mention some of these properties below, though we will be working with Definition 7.1.3. Various formulations of these properties and methods for achieving them can be found in [BFM88, BDMP91, BG89, NY90, DY90, FLS99].

Proving many statements. In many applications of noninteractive zero knowledge, one needs to prove many statements noninteractively using the same shared random string, whereas our definition only refers to proving one statement. One way of proving t statements is to use t independent executions of the proof system, but this multiplies the length of the shared random string by a factor of t, and hence requires an *a priori* bound on the number of statements to be proven. Ideally, the shared random string would be a fixed length (polynomial in the input length and the security parameter) and can be used to prove an arbitrary (polynomial) number of statements. Definition 7.1.3 is sometimes referred to as *bounded* or *single-theorem* noninteractive zero knowledge in the literature.

Adaptive noninteractive zero knowledge. Another issue is whether the statements to be proven can be selected "adaptively" after the shared random string is published. Our definition only guarantees soundness and zero-knowledgeness if the statement to be proven is fixed before the shared random string is selected. Preserving soundness in the adaptive setting is not difficult — if one uses parallel repetitions to make the soundness error of a nonadaptive proof system sufficiently smaller than 2^{-n}, then with high probability the shared random string will be "good" (with respect to soundness) for all statements of length n, and thus it does not matter if the statement is selected after the proof. Preserving zero-knowledgeness in the adaptive setting, however, is much less straightforward.

Efficient provers. In order to actually implement a noninteractive zero-knowledge proof system, it is clearly necessary that the prover strategy can be implemented in polynomial time given, say, some auxiliary information. This only makes sense for problems in **NP**, as the auxiliary information can be viewed as an **NP**-proof.[4]

[4] Strictly speaking, it also makes sense for problems in **MA** [BM88], as the verification might be probabilistic.

Solutions for NICZK. Feige, Lapidot, and Shamir [FLS99] show how to achieve all of these properties for **NICZK** under intractability assumptions. Specifically, they show that every problem in **NP** has a many-theorem adaptive **NICZK** proof with an efficient prover, if trapdoor permutations exist. For **NISZK** and **NIPZK**, however, the relationship between Definition 7.1.3 and the many-theorem and adaptive variants is still open.

7.2 The Completeness Theorem

We consider the following restricted versions of Statistical Difference and Entropy Difference.

Definition 7.2.1 Statistical Difference from Uniform *is the promise problem* $\mathrm{SDU} = (\mathrm{SDU}_Y, \mathrm{SDU}_N)$, *where*

$$\begin{aligned} \mathrm{SDU}_Y &= \{X \, : \, \mathrm{StatDiff}\,(X, U_n) \leq 1/n\} \\ \mathrm{SD}_N &= \{X \, : \, \mathrm{StatDiff}\,(X, U_n) \geq 1 - 1/n\}\,. \end{aligned}$$

Above, X is a circuit encoding a probability distribution on $\{0,1\}^n$ (where n is the number of output gates of X), as in Definition 3.1.1, and U_n is the uniform distribution on $\{0,1\}^n$.

Definition 7.2.2 Entropy Approximation *is the promise problem* $\mathrm{EA} = (\mathrm{EA}_Y, \mathrm{EA}_N)$, *where*

$$\begin{aligned} \mathrm{EA}_Y &= \{(X,t) \, : \, \mathrm{H}(X) \geq t+1\} \\ \mathrm{EA}_N &= \{(X,t) \, : \, \mathrm{H}(X) \leq t-1\}\,. \end{aligned}$$

Above, X is circuit encoding a probability distribution, as in Definition 3.1.1, t is an integer, and $\mathrm{H}(\cdot)$ denotes the entropy function (Definition 3.3.1).

In Sections 7.3 and 7.4, we will prove the following completeness theorem for **NISZK**.

Theorem 7.2.3 (Completeness Theorem for NISZK) Entropy Approximation *and* Statistical Difference from Uniform *are complete for* **NISZK**.

It is interesting to informally compare this with the Completeness Theorem for **HVSZK** (= **SZK**) (Theorem 3.5.1):

> *Whereas (interactive) statistical zero knowledge captures those assertions that can be cast as comparing two efficiently samplable distributions to each other (either with respect to their statistical difference or their entropies), noninteractive statistical zero knowledge consists exactly of those assertions which can be cast as comparing a single distribution to the uniform distribution.*

As was the case with **HVSZK**, the complete problems for **NISZK** are useful tools for proving general theorems about the entire class. Our most dramatic application of these complete problems comes from the fact that they are natural restrictions of the complete problems for **SZK**. In Section 7.5, we exploit this relationship to get a better understanding of how **NISZK** compares to **SZK**. Other corollaries of the completeness theorem are given in Sections 7.4 and 7.6.

Prior to this work, De Santis, Di Crescenzo, Persiano, and Yung [DDPY98] showed that a different promise problem, called IMAGE DENSITY (ID) is complete for **NISZK**. Roughly speaking, the YES instances of ID are distributions on strings of some length n (encoded by circuits) which are statistically close to the uniform distribution on $\{0,1\}^n$, and the NO instances of ID are distributions whose support is a small fraction of $\{0,1\}^n$. Thus, for an appropriate quantification of "close" and "small fraction," ID is a restricted version of SDU. The main interesting feature of our complete problems (as compared to ID) is that they are more closely related to the complete problems for **SZK**. Specifically, we will exploit the connection between ENTROPY APPROXIMATION and ENTROPY DIFFERENCE in comparing **SZK** and **NISZK**.

We prove the Completeness Theorem via a "circle of reductions" analogous to (but simpler than) the one used to prove the Completeness Theorem for **HVSZK**. First, in Section 7.3, we prove that EA is in **NISZK**. Next, in Section 7.4, we show that SDU reduces to EA. Finally, also in Section 7.4, we complete the circle by showing that every problem in **NISZK** reduces to EA.

7.3 ENTROPY APPROXIMATION is in NISZK

7.3.1 The proof system

In this section, we exhibit a noninteractive statistical zero-knowledge proof system for ENTROPY APPROXIMATION. We begin by considering Protocol 7.3.1, which is a simple noninteractive protocol for proving that the support of a distribution X on $\{0,1\}^n$ is nearly all of $\{0,1\}^n$.

Protocol 7.3.1: Basic noninteractive proof system (P, V) for showing a distribution has large support

Input: Circuit X (with m input gates and n output gates), and shared random string $x \in \{0,1\}^n$

1. P: Select r uniformly from $\Omega_X(x) \stackrel{\text{def}}{=} \{r' \,:\, X(r') = x\}$ and send r to V. (If $\Omega_X(x) = \emptyset$, then send `fail` to V.)
2. V: Accept if $X(r) = x$, otherwise reject.

The prover's success probability in Protocol 7.3.1 is evident by inspection:

Claim 7.3.2 *The prover strategy given in Protocol 7.3.1 makes the verifier accept with probability exactly* $|\mathrm{Supp}(X)|/2^n$*, and no prover strategy can make the verifier accept with higher probability.*

Thus, the protocol is complete and sound: if the support of X is nearly all of $\{0,1\}^n$, the verifier will accept with high probability; and if the support is a small fraction of $\{0,1\}^n$, the verifier will reject with high probability no matter what strategy the prover uses. In fact, if X has not just large support, but is close to uniform, the protocol also can be simulated well, as done by Algorithm 7.3.3.

Algorithm 7.3.3: Simulator for Protocol 7.3.1

Input: Circuit X (with m input gates and n output gates)

1. Select $r \in \{0,1\}^m$. Let $x = X(r)$.
2. Output (x, r)

Claim 7.3.4 *The statistical difference between the output of Algorithm 7.3.3 and the verifier's view of Protocol 7.3.1 is exactly* $\mathrm{StatDiff}(X, U_n)$.

Proof: The statistical difference between the x-components of the two distributions is exactly $\mathrm{StatDiff}(X, U_n)$. Conditioned on x, r is selected uniformly from $\Omega_X(x)$ in both distributions, so it does not increase the statistical difference. ■

Thus, to give an **NISZK** proof system for EA, it suffices to give a transformation mapping YES instances to distributions that are close to uniform and NO instances to distributions with small support. This is given by the following lemma, which we prove in Section 7.3.2.

Lemma 7.3.5 *There is a polynomial-time computable function that takes an instance (X, t) of* EA *and a parameter k (in unary) and produces a distribution Z (encoded by a circuit which samples from it) such that, letting N be the number of output gates of Z, we have:*

1. *If $\mathrm{H}(X) \geq t+1$, then Z has statistical difference at most 2^{-k} from the uniform distribution on $\{0,1\}^N$, and*
2. *If $\mathrm{H}(X) \leq t-1$, then the support of Z is at most a 2^{-k} fraction of $\{0,1\}^N$.*

Lemma 7.3.5 essentially reduces to ENTROPY APPROXIMATION to IMAGE DENSITY, the complete problem of De Santis *et. al.* [DDPY98]. Combining Lemma 7.3.5 with Claims 7.3.2 and 7.3.4, we obtain:

Theorem 7.3.6 ENTROPY APPROXIMATION *is in* **NISZK**. *Moreover, it has a noninteractive statistical zero-knowledge proof system with simulator deviation 2^{-k} and a deterministic verifier.*

Protocol 7.3.1 (together with Lemma 7.3.5) gives a proof system with nonzero, though exponentially small, completeness error. However, this completeness error can be removed using a transformation given in [BDMP91, BR90], which converts noninteractive zero knowledge proofs into ones with perfect completeness. (That transformation preserves both statistical and computational zero knowledge, maintains an exponentially small simulator deviation in the case of statistical zero knowledge, and keeps the verifier deterministic.)

7.3.2 Proof of Lemma 7.3.5

We now prove Lemma 7.3.5. The transformation is based on techniques we have used many times — 2-universal hashing, the Leftover Hash Lemma, and flattening distributions (*cf.*, Sections 3.4.1 and 3.4.3 for the definitions). Let (X, t) be an instance of EA. Recall that the Leftover Hash Lemma converts nearly flat distributions with large entropy into nearly uniform ones. This suggests the following first attempt at constructing the distribution Z:

1. Let X' consists of many, say s, independent copies of X so that the entropy of X' is greater $s \cdot (t+1)$ for YES instances and less than $s \cdot (t-1)$ for for NO instances, while X' is Δ-flat, for $\Delta \ll s$.
2. Define Z to be the distribution $(h, h(x))$, where h is chosen uniformly from a 2-universal family of hash functions with range $\{0,1\}^{st}$ and x is sampled according to X'.

For a sufficiently large (but still polynomial) choice of the parameter s, this does indeed map YES instances (X, t) of EA to distributions Z that are close to uniform. Unfortunately, Z does not necessarily have small support when (X, t) is a NO instance. However, it almost works: The fact that the entropy of X' is much smaller than st implies that if we remove the very "light" strings from $\mathrm{Supp}(X')$ (*i.e.*, the strings assigned probability mass much smaller than $2^{-\mathrm{H}(X')}$), what remains is a set T of size much smaller than 2^{st}. The near-flatness of X' implies that $\Pr[X' \in T]$ is very close to 1. For any hash function h mapping to st bits, $h(T)$ will be a very small fraction of $\{0,1\}^{st}$. So, the reason that Z might still have large support is the rare event that we obtain a very light sample from X'. To deal with such light samples, note that a sample x being light means that $\{r : X'(r) = x\}$ is atypically small. So, we add to Z another two components $(h', h'(r))$, where h' is another hash function (mapping to a different number of bits) and r is is the input to X' used to produce the sample x used in the second component of Z. Thus, when x is one of these rare points outside T, $h'(r)$ will only hit a small fraction of its range, and Z will have small support.

To formalize this intuition, let (X, t) and k be given as in the lemma. Note that it suffices for the transformation to achieve error parameters $2^{-\Omega(k)}$ rather than 2^{-k}, as we can compensate for this by first increasing k by a constant factor. Let m be the number of input gates to X and n the number of output gates. Define $X' = \otimes^s X$, for $s = 4k \cdot m^2$. Thus, X' has sm input gates, sn output gates, and, by Lemma 3.4.6, is Δ-flat for $\Delta = \sqrt{(4km^2)} \cdot m = 2\sqrt{k} \cdot m^2$. Consider the following distribution Z:

Z**:** Choose $r \leftarrow \{0,1\}^{sm}$. Let $x = X'(r)$. Select $h_1 \leftarrow \mathcal{H}_{sn,st}$ and $h_2 \leftarrow \mathcal{H}_{sm,sm-st-k}$. Output $(h_1, h_1(x), h_2, h_2(r))$.

We denote the (jointly distributed) random variables corresponding to the components of Z by (H_1, Y_1, H_2, Y_2). We also use X' to denote the distribution of x and R to denote the distribution of r, so $Y_1 = H_1(X')$, $Y_2 = H_2(R)$ and $X' = X'(R)$.

Claim 7.3.7 *If* $\mathrm{H}(X) \geq t+1$, *then* Z *has statistical difference at most* $2^{-\Omega(k)}$ *from the uniform distribution on* $\mathcal{H}_{sn,st} \times \{0,1\}^{st} \times \mathcal{H}_{sm,sm-st-k} \times \{0,1\}^{sm-st-k}$.

Proof: First we analyze the distribution on the first two components $(H_1, H_1(X'))$. Note that X' has entropy at least $s\cdot(t+1) = st+2\sqrt{k}\Delta$. By the Δ-flatness of X', we can apply the Leftover Hash Lemma (Lemma 3.4.7) with parameters $\delta = 2^{-k+1}$ and $\varepsilon = 2^{-\sqrt{k}\Delta} < 2^{-k}$ to see that $(H_1, H_1(X')) = (H_1, Y_1)$ has statistical difference at most $2^{-\Omega(k)}$ from the uniform distribution on $\mathcal{H}_{sn,st} \times \{0,1\}^{st}$. It follows that with probability at least $1 - 2^{-\Omega(k)}$ over $(h_1, y_1) \leftarrow (H_1, Y_1)$,

$$\Pr\left[(H_1, Y_1) = (h_1, y_1)\right] \geq \frac{1}{2} \cdot \frac{1}{|\mathcal{H}_{sn,st} \times \{0,1\}^{st}|}.$$

Fix any pair (h_1, y_1) such that this holds. Then the conditional distribution $R|_{(H_1,Y_1)=(h_1,y_1)}$ is uniform over the set $\{r \,:\, h_1(X'(r)) = y_1\}$, which is of size

$$\begin{aligned} 2^{sm} \cdot \Pr\left[Y_1 = y_1 | H_1 = h_1\right] &= 2^{sm} \cdot \frac{\Pr\left[(H_1, Y_1) = (h_1, y_1)\right]}{\Pr\left[H_1 = h_1\right]} \\ &\geq 2^{sm} \cdot \frac{1/\left(2 \cdot |\mathcal{H}_{sn,st}| \cdot 2^{st}\right)}{1/|\mathcal{H}_{sn,st}|} \\ &= 2^{sm-st-1}. \end{aligned}$$

Thus, by the Leftover Hash Lemma, $(H_2, H_2(R))|_{(H_1,Y_1)=(h_1,y_1)}$ has statistical difference $2^{-\Omega(k)}$ from the uniform distribution on $\mathcal{H}_{sm,sm-st-k} \times \{0,1\}^{sm-st-k}$. Recalling that this holds with probability $1 - 2^{-\Omega(k)}$ over $(h_1, y_1) \leftarrow (H_1, Y_1)$ and that (H_1, Y_1) has statistical difference at most $2^{-\Omega(k)}$ from uniform, we conclude that (H_1, Y_1, H_2, Y_2) has statistical difference at most $2^{-\Omega(k)}$ from uniform. ■

Claim 7.3.8 *If* $\mathrm{H}(X) \leq t-1$, *then the support of* Z *is at most an* $O(2^{-k})$ *fraction of* $\mathcal{H}_{sn,st} \times \{0,1\}^{st} \times \mathcal{H}_{sm,sm-st-k} \times \{0,1\}^{sm-st-k}$.

Proof: Note that the entropy of X' is at most $s \cdot (t-1) \leq st - \sqrt{3k} \cdot \Delta - k$. We will show that, for every fixed $h_1 \in \mathcal{H}_{sm,st}$, the support $S = S_{h_1}$ of $(h_1(X'), H_2, H_2(R))$ is at most an $O(2^{-k})$ fraction of $D = \{0,1\}^{st} \times \mathcal{H}_{sm,sm-st-k} \times \{0,1\}^{sm-st-k}$. Clearly this suffices to prove the lemma.

Fix $h_1 \in \mathcal{H}_{sm,st}$. To bound the size of $S = S_{h_1}$, we divide it into three subsets, depending on the probability mass of the first component $h_1(X')$ (as compared to a "typical," unhashed sample from X'). Recall that a "typical" sample from X' has probability mass $\approx 2^{-\mathrm{H}(X')} \geq$

$2^{-st+\sqrt{3k}\cdot\Delta+k}$.

$$S_1 = \{(y_1, h_2, y_2) \in S \ : \ 2^{-st+k} < \Pr[h_1(X') = y_1]\} \qquad \text{(“not too light”)}$$

$$S_2 = \{(y_1, h_2, y_2) \in S \ : \ 2^{-st-2k} < \Pr[h_1(X') = y_1] \leq 2^{-st+k}\} \qquad \text{(“too light, but not much too light”)}$$

$$S_3 = \{(y_1, h_2, y_2) \in S \ : \ \Pr[h_1(X') = y_1] \leq 2^{-st-2k}\} \qquad \text{(“much too light”)}$$

Clearly, $S = S_1 \cup S_2 \cup S_3$. We will show that $|S_i|/|D| \leq O(2^{-k})$ for $i = 1, 2, 3$, and so $|S|/|D| \leq 3 \cdot O(2^{-k}) = O(2^{-k})$.

First, we bound $|S_1|$. Clearly, there can be at most 2^{st-k} values of y_1 such that $\Pr[h_1(X') = y_1] > 2^{-st+k}$, so the first components of elements of S_1 cover at most a 2^{-k} fraction of $\{0,1\}^{st}$. Hence S_1 is at most a 2^{-k} fraction of D.

Now we bound $|S_2|$. Consider the set

$$A = \{y_1 \ : \ 2^{-st-2k} < \Pr\left[h_1(X') = y_1\right] \leq 2^{-st+k}\}.$$

We will show that A is of size at most 2^{st-k+1}; like the previous case, it then follows that $|S_2|/|D| \leq 2^{-k+1}$. Note that if $h_1(x) \in A$, then $\Pr[X' = x] \leq \Pr[h_1(X') = h_1(x)] \leq 2^{-st+k}$. Thus, if $h_1(x) \in A$, then x is $\sqrt{3k}\cdot\Delta$-light (since X' has entropy at most $st-k-\sqrt{3k}\cdot\Delta$). By the Δ-flatness of X', $\Pr[h_1(X') \in A]$ is at most 2^{-3k+1}. Since every $y_1 \in A$ has probability mass at least 2^{-st-2k} under $h_1(X')$, it follows that $|A|$ is at most $2^{-3k+1}/2^{-st-2k} = 2^{st-k+1}$.

Finally, we bound $|S_3|$. Note that, for any y_1,

$$\Pr\left[h_1(X') = y_1\right] = 2^{-sm} \cdot \left|\{r \ : \ h_1(X'(r)) = y_1\}\right|.$$

Thus, for any y_1 such that $\Pr[h_1(X') = y_1] \leq 2^{-st-2k}$, there are at most $2^{-st-2k} \cdot 2^{sm}$ values of r consistent with $h_1(X'(r)) = y_1$. Hence, for any such y_1 and any h_2, the set of y_2 such that $(y_1, h_2, y_2) \in S$ is of size at most $2^{sm-st-2k}$ (because each such y_2 is of the form $h_2(r)$ for some r such that $h_1(X'(r)) = y_1$). This implies that S_3 is at most a $2^{sm-st-2k}/2^{sm-st-k} = 2^{-k}$ fraction of D. ■

We comment that the protocol obtained by combining the above transformation with Protocol 7.3.1 yields a protocol that is closely related to the standard lower bound protocol (Protocol 5.2.1). Indeed, proving an approximate lower bound on the entropy of a nearly flat distribution X is almost equivalent to proving an approximate lower bound on the size of $\mathrm{Supp}(X)$, except for difficulties caused by “light” samples. Our method for handling this difficulty can be viewed as using another lower bound protocol on the inputs to X.

7.4 Proof of the Completeness Theorem

In this section, we complete the proof of the completeness theorem for **NISZK**. First, we show that Statistical Difference from Uniform reduces to Entropy Approximation.

Lemma 7.4.1 SDU $\leq_{\texttt{Karp}}$ EA. *In particular,* SDU $\in$ **NISZK**.

Proof: Let X be an instance of SDU. First, we treat the case that $\log n > 4$, where n is the output length of the circuit X. In this case, we claim $X \mapsto (X, n-3)$ is a valid reduction to EA. The correctness of this reduction follows from the following claim relating the entropy of a distribution to its distance from the uniform distribution.

Claim 7.4.2 *Let X be any distribution on a universe $\mathcal{U}$ and let U denote the uniform distribution on $\mathcal{U}$. Then*

1. *If* $\mathrm{StatDiff}(X, U) \leq \alpha$, *then* $\mathrm{H}(X) \geq \log|\mathcal{U}| - \left(\alpha + \frac{1}{|\mathcal{U}|}\right) \cdot \log|\mathcal{U}|$.

2. *If* $\mathrm{StatDiff}(X, U) \geq \beta$, *then* $\mathrm{H}(X) \leq \log|\mathcal{U}| - \log\left(\frac{1}{1-\beta}\right)$.

Applying this claim with $\mathcal{U} = \{0,1\}^n$, $\alpha = 1/n$, and $\beta = 1 - 1/n$ shows that YES instances of SDU have entropy at least $n-2$ and NO instances have entropy at most $n - \log n \leq n-4$. This establishes the validity of the reduction.

Now we treat the case that $\log n < 4$. In this case, the statistical difference between X and U_n can be approximated in probabilistic polynomial time by sampling X sufficiently many times and counting the number of times each output occurs. So let $A(X)$ be the probabilistic algorithm which outputs 1 with probability at least $2/3$ when $X \in \mathrm{SDU}_Y$ and outputs 1 with probability at most $1/3$ when $X \in \mathrm{SDU}_N$. Now consider the circuit Y defined as follows:

Y**:** Run $A(X)$ to obtain output b. If $b = 1$ output 9 random bits, and if $b = 0$ output 0^9.

Now, if $X \in \mathrm{SDU}_Y$, then $\mathrm{H}(Y) \geq (2/3) \cdot 9 = 6$. If instead $X \in \mathrm{SDU}_N$, then $\mathrm{H}(Y) \leq \mathrm{H}_2(1/3) + (1/3) \cdot 9 \leq 4$. Thus $X \mapsto (Y, 5)$ is a valid reduction from SDU to EA in this case. ■

Now we complete the circle of reductions by showing that every problem in **weak-NISZK** reduces to SDU.

Lemma 7.4.3 *Every promise problem in* **weak-NISZK** *reduces to* SDU.

By the correspondence between noninteractive proofs and 2-message public-coin interactive proofs, we could apply the simulator analysis for public-coin statistical zero-knowledge proofs given in Section 3.2. However, since the case of noninteractive proof systems is much simpler, we give the reduction directly. Our reduction is essentially the same as the reduction of De Santis*et. al.* [DDPY98] to their complete problem IMAGE DENSITY, with a small complication caused by the fact that we allow the verifier to be probabilistic.

Proof: Let Π be any promise problem in **weak-NISZK**. Let (P, V) be a **weak-NISZK** proof system for Π and let $\ell = \ell(n)$ be a polynomial bound on the length of the shared random string on inputs of length n. We assume that (P, V) has completeness and soundness error at most $1/9\ell$ (actually these can be assumed to be exponentially small by repeating the proof system sufficiently many times in parallel). By the **weak-NISZK** property, there is a simulator S for (P, V) which achieves simulator deviation $1/3\ell$.

For an instance x of Π, consider the following distribution X_x:

X_x: Run $S(x)$ to obtain a simulated transcript (σ, proof). Run $V(x, \sigma, \mathit{proof})$ ℓ times. If V accepts in the majority of the executions, output σ. Otherwise, output 0^ℓ.

We claim that $x \mapsto X_x$ is the reduction we are seeking. Suppose $x \in \Pi_Y$. Consider the distribution $\overline{X}_x$ which is the same as X_x, except that (σ, proof) is taken from $(P, V)(x)$ instead of from $S(x)$. X_x and $\overline{X}_x$ have statistical difference at most the simulator deviation $(1/3\ell)$, so it suffices to show that $\overline{X}_x$ has statistical difference at most $2/3\ell$ from uniform. For (σ, proof) taken from $(P, V)(x)$, σ is distributed uniformly, so we need only analyze the probability that it is discarded and replaced with 0^ℓ in $\overline{X}_x$. Let B be the set of "bad" pairs (σ, proof)'s for which $\Pr[V(x, \sigma, \mathit{proof}) = \mathtt{accept}] \leq 2/3$. The probability that $(\sigma, \mathit{proof}) \in B$ is at most $1/3\ell$, for otherwise V would reject with probability greater than $(1/3\ell) \cdot (1/3) = 1/9\ell$, violating completeness. By the Chernoff Bound, for any pair $(\sigma, \mathit{proof}) \notin B$, the probability that $V(x, \sigma, \mathit{proof})$ accepts in the majority of ℓ independent executions is at least $1 - \exp(-\Omega(\ell))$. Thus, in $\overline{X}_x$, σ is replaced with 0^ℓ with probability at most $1/3\ell + \exp(-\Omega(\ell)) < 2/3\ell$, and hence $\overline{X}_x$ has statistical difference at most $2/3\ell$ from uniform.

Now suppose that $x \in \Pi_N$. Consider the set B of "bad" σ's for which there exists a *proof* such that $\Pr[V(x, \sigma, \mathit{proof}) = \mathtt{accept}] \geq 1/3$. The probability that a uniformly distributed σ is in B is at most $1/3\ell$, for otherwise there would be a prover strategy which makes V accept with probability greater than $(1/3\ell) \cdot (1/3) = 1/9\ell$, violating soundness. However, whenever $\sigma \notin B$, X_x outputs 0^ℓ with probability at least $1 - \exp(-\Omega(\ell))$ (by the Chernoff Bound). Hence, X_x is in $B \cup \{0^\ell\}$ with probability at least $1 - \exp(-\Omega(\ell)) \geq 1 - 1/3\ell$, whereas the uniform distribution is in $B \cup \{0^\ell\}$ with probability at most $1/3\ell + 2^{-\ell} \leq 2/3\ell$, for a statistical difference at least $[1 - 1/3\ell] - 2/3\ell = 1 - 1/\ell$. ■

The Completeness Theorem (Theorem 7.2.3) follows by combining Theorem 7.3.6 and Lemmas 7.4.1 and 7.4.3. We can draw a couple of immediate corollaries from our proof of the Completeness Theorem. By the fact that the reduction from **NISZK** to SDU actually works for all of **weak-NISZK**, we obtain:

Corollary 7.4.4 **weak-NISZK** = **NISZK**.

Since the complete problem EA possesses an **NISZK** proof system with exponentially vanishing simulator deviation (Theorem 7.3.6), so must all other problems in **NISZK**.

Corollary 7.4.5 *Every problem in* **NISZK** *possesses a noninteractive statistical zero knowledge proof system with simulator deviation* 2^{-k} *and a deterministic verifier.*

7.5 Comparing SZK and NISZK

7.5.1 Nontriviality of NISZK

In this section, we use the complete problems to relate **SZK** and **NISZK**. The first result is that if **NISZK** = **BPP** then **SZK** = **BPP**. This is done by giving a Cook reduction from Entropy Difference (ED) to Entropy Approximation (EA).

Lemma 7.5.1 *Suppose (X, Y) is an instance of* ED*. Let $X' = \otimes^3 X$ (resp., $Y' = \otimes^3 Y$) consist of 4 independent copies of X (resp., Y), and let n denote the output length of X'. Then,*

$$(X,Y) \in \mathrm{ED}_Y \implies \bigvee_{t=1}^{n} \left[\left((X',t) \in \mathrm{EA}_Y\right) \wedge \left((Y',t) \in \mathrm{EA}_N\right)\right]$$
$$(X,Y) \in \mathrm{ED}_N \implies \bigwedge_{t=1}^{n} \left[\left((X',t) \in \mathrm{EA}_N\right) \vee \left((Y',t) \in \mathrm{EA}_Y\right)\right]$$

Proof: Suppose $(X, Y) \in \mathrm{ED}_Y$, so that $\mathrm{H}(X') \geq \mathrm{H}(Y') + 3$. Since $(\mathrm{H}(X') - 1) - (\mathrm{H}(Y') + 1) \geq 1$, there must be some integer t in the interval $[\mathrm{H}(X') - 1, \mathrm{H}(Y') + 1]$, which implies that $(X', t) \in \mathrm{EA}_Y$ and $(Y', t) \in \mathrm{EA}_N$. Suppose instead $(X, Y) \in \mathrm{ED}_N$, so that $\mathrm{H}(Y') \geq \mathrm{H}(X') + 3$. Since $\mathrm{H}(X') + 1 < \mathrm{H}(Y') - 1$, every t is either greater than $\mathrm{H}(X') + 1$ or less than $\mathrm{H}(Y') - 1$. That is, for every t, $(X', t) \in \mathrm{EA}_N$ or $(Y', t) \in \mathrm{EA}_Y$. ■

Thus, we conclude:

Theorem 7.5.2 **NISZK** $\neq$ **BPP** *iff* **SZK** $\neq$ **BPP**.

Proof: By definition, **BPP** $\subset$ **NISZK** $\subset$ **HVSZK**, and **HVSZK** = **SZK** by Theorem 6.3.1. Thus, if **SZK** = **BPP**, then **NISZK** = **BPP**.

Now suppose that **NISZK** = **BPP**. In particular, there is a probabilistic polynomial time algorithm A which decides EA with exponentially small error probability. To prove that **SZK** = **BPP**, it suffices to exhibit a **BPP** algorithm for ED, since ED is **SZK**-complete. The algorithm is given as follows: Given an instance (X, Y) of ED, let X', Y', and n be as stated in Lemma 7.5.1. Run $A(X', t)$ and $A(Y', t)$ for $t = 1, \ldots, n$. If, for some t, $A(X', t) =$ YES and $A(Y', t) =$ NO, then output YES. Otherwise, output NO. By Lemma 7.5.1, this is a correct **BPP** algorithm for deciding ED. ■

7.5.2 Conditions under which NISZK = SZK

In this section, we use special properties of the reduction from ED to EA given in the previous section to shed additional light on the relationship between **NISZK** and **SZK**. Specifically, we will show that if **NISZK** is closed under complement, then in fact **NISZK** = **SZK**.

The key observation is that the reduction from ED to EA is nonadaptive (*i.e.,* all the queries to EA can be asked at once) and the final answer is computed by applying the simple Boolean formula of Lemma 7.5.1 to the responses. That is, it is an $\mathbf{NC}_1$ truth-table reduction, in the sense of Definition 4.5.10. In fact, the Boolean formula has constant depth; this property is captured by the following definition.

Definition 7.5.3 ($\mathbf{AC}_0$ truth-table reductions) *A truth-table reduction f between promise problems is an* $\mathbf{AC}_0$ truth-table reduction[5] *if the circuit C produced by the reduction on input*

[5]This terminology is inherited from the **AC** hierarchy of languages, where $\mathbf{AC}_i$ denotes the class of languages decided by (uniform) families of circuits of unbounded fan-in and depth $O(\log^i n)$. See, *e.g.,* [Pap94].

x has depth bounded by c_f, where c_f is a constant independent of x. (C may have unbounded fan-in.) If there is an $\mathbf{AC}_0$ truth-table reduction from Γ to Π, we write $\Gamma \leq_{\mathbf{AC}_0-\mathtt{tt}} \Pi$.

From Lemma 7.5.1, we have:

Proposition 7.5.4 ED $\leq_{\mathbf{AC}_0-\mathtt{tt}}$ EA.

By this proposition, if **NISZK** were closed under $\mathbf{AC}_0$ truth-table reductions, then ED would be in **NISZK** and hence **NISZK** = **SZK**. Thus, we would like to capture the minimal conditions for a complexity class to be closed under $\mathbf{AC}_0$ truth-table reductions. We define the following operator on promise problems to capture the notion of an unbounded fan-in AND gate.

Definition 7.5.5 (unbounded AND) *For any promise problem Π, we define* AND(Π) *to be the following promise problem:*

$$\begin{aligned} \mathrm{AND}(\Pi)_Y &= \{(x_1, x_2, \ldots, x_k) \,:\, k \geq 0,\ \forall i \in [1,k]\ x_i \in \Pi_Y\} \\ \mathrm{AND}(\Pi)_N &= \{(x_1, x_2, \ldots, x_k) \,:\, k \geq 0,\ \exists i \in [1,k]\ x_i \in \Pi_N\} \end{aligned}$$

We say a class **C** *of promise problems is* closed under unbounded AND *if $\Pi \in \mathbf{C}$ implies that* AND(Π) $\in$ **C**.

We have defined AND($\cdot$) so that it has the weakest promise condition possible to remain well-defined. In particular, AND$(\Pi)_N$ is defined to include x_i's that violate Π's promise, as long as just *one* of them is in Π_N.

We also need a way of combining two promise problems:

Definition 7.5.6 (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* DisjUn(Π, Γ) *defined as follows:*

$$\begin{aligned} \mathrm{DisjUn}(\Pi, \Gamma)_Y &= \{0\} \times \Pi_Y \cup \{1\} \times \Gamma_Y \\ \mathrm{DisjUn}(\Pi, \Gamma)_N &= \{0\} \times \Pi_N \cup \{1\} \times \Gamma_N \end{aligned}$$

We say a class **C** *of promise problems is* closed under disjoint union *if $\Pi, \Gamma \in \mathbf{C}$ implies that* DisjUn(Π, Γ) $\in$ **C**.

With these definitions, we can give some conditions that imply closure under $\mathbf{AC}_0$ truth-table reductions.

Lemma 7.5.7 *A promise class* **C** *is closed under $\mathbf{AC}_0$ truth-table reductions if the following conditions hold:*

1. **C** *is closed under Karp reductions.*
2. **C** *is closed under unbounded AND.*
3. **C** *is closed under disjoint union.*
4. **C** *is closed under complementation.*

Proof: As a first step, we observe that **C** is closed under unbounded OR (defined analogously to unbounded AND): DeMorgan's Laws say that $\mathrm{OR}(\Pi) = \overline{\mathrm{AND}(\overline{\Pi})}$, which is in **C**, by closure under unbounded AND and complementation. To generalize this to constant-depth circuits, we define for each $d \in \mathbb{N}$, a promise problem $\mathrm{Depth}^d(\Pi)$ which is defined exactly as $\Phi(\Pi)$ (Definition 4.5.3), except formulae ϕ are replaced with circuits C of depth at most d (using unbounded fan-in AND and OR gates).

By definition, $\Pi \leq_{\mathbf{AC}_0\mathtt{-tt}} \Gamma$ means that there exists some d such that $\Pi \leq_{\mathtt{Karp}} \mathrm{Depth}^d(\Gamma)$. Hence if we can show that for all $d \geq 0$ and promise problems $\Pi \in \mathbf{C}$, $\mathrm{Depth}^d(\Pi) \in \mathbf{C}$, the lemma will be established. We will prove this by induction on d.

First, observe that a depth 0 circuit is simply a variable or its negation. Hence, $\mathrm{Depth}^0(\Pi) \leq_{\mathtt{Karp}} \mathrm{DisjUn}(\Pi, \overline{\Pi}) \in \mathbf{C}$. (The reduction maps $(v_i, (x_1, \ldots, x_m)) \mapsto (0, x_i)$ and $(\neg v_i, (x_1, \ldots, x_m)) \mapsto (1, x_i)$). Now assume that $\mathrm{Depth}^d(\Pi) \in \mathbf{C}$. By definition, a depth $d+1$ circuit is an AND or an OR of some number of depth d circuits. (By applying DeMorgan's Laws, we may assume that all negations are applied directly to the variables.) Using this fact, we will argue that that

$$\mathrm{Depth}^{d+1}(\Pi) \leq_{\mathtt{Karp}} \mathrm{DisjUn}(\mathrm{AND}(\mathrm{Depth}^d(\Pi)), \mathrm{OR}(\mathrm{Depth}^d(\Pi))).$$

By the hypothesized closure properties of **C**, this implies that $\mathrm{Depth}^{d+1}(\Pi) \in \mathbf{C}$. The reduction works as follows. The input to the reduction is a pair $(C, \overline{x})$ where $\overline{x} = (x_1, x_2, \ldots x_m)$. First, the reduction extracts from C the circuits $C_1, C_2, \ldots, C_s$ that provide input to the topmost AND/OR gate, and sets $\sigma = 0$ (resp., $\sigma = 1$) if that gate is an AND (resp., OR) gate. Then the reduction outputs $(\sigma, ((C_1, \overline{x}), (C_2, \overline{x}), \ldots, (C_s, \overline{x})))$. It is clear that this map gives a Karp reduction from $\mathrm{Depth}^{d+1}(\Pi)$ to $\mathrm{DisjUn}(\mathrm{AND}(\mathrm{Depth}^d(\Pi)), \mathrm{OR}(\mathrm{Depth}^d(\Pi)))$, completing the induction step and the proof. ∎

Which of the conditions of Lemma 7.5.7 does **NISZK** satisfy? We have already shown that Condition 1 is satisfied by **NISZK** (Proposition 2.4.1). We now argue that Conditions 2 and 3 are also satisfied:

Lemma 7.5.8 **NISZK** *is closed under unbounded AND.*

Proof: Let Π be any problem in **NISZK**. Let (P, V) be a noninteractive statistical zero-knowledge proof system for Π with completeness and soundness errors at most $1/k$ and simulator devation $\mu(k)$ (where k is the security parameter). We now describe an **NISZK** proof system (P', V') for $\mathrm{AND}(\Pi)$: Given an instance $(x_1, \ldots, x_m)$ of $\mathrm{AND}(\Pi)$ and a security parameter k', P' and V' execute the (P, V) on each x_i, using security parameter $k = 2k'm$, and V' accepts if V accepts in each of these executions.

(P', V') is a noninteractive proof system for $\mathrm{AND}(\Pi)$ with completeness error at most $m \cdot 1/k \leq 1/k'$ and soundness error at most $1/k \leq 1/k'$. Moreover, it can be simulated by running the simulator for (P, V) on each of the inputs x_i. This gives simulator deviation $m \cdot \mu(k)$, which is (bounded by) a negligible function of k'. ∎

Lemma 7.5.9 **NISZK** *is closed under disjoint union.*

Proof: If $\Pi_0, \Pi_1 \in \mathbf{NISZK}$, an **NISZK** proof system for $\mathrm{DisjUn}(\Pi_0, \Pi_1)$ can be obtained as follows: On input (σ, x) and security parameter k, the prover and verifier execute the **NISZK** proof system for Π_σ on input x and security parameter k. ∎

Combining everything, we can give a condition under which **SZK** = **NISZK**.

Proposition 7.5.10 *If* **NISZK** *is closed under complementation, then* **SZK** = **NISZK**.

Proof: Suppose **NISZK** is closed under complementation. By Lemmas 7.5.7, 7.5.8, and 7.5.9 and Proposition 7.1.4, it follows that **NISZK** is closed under $\mathbf{AC}_0$ truth-table reductions. Combining Proposition 7.5.4 (ED $\leq_{\mathbf{AC}_0-\mathtt{tt}}$ EA) and Theorem 3.5.1 (ED is complete for **SZK**), we see that every problem in **SZK** $\mathbf{AC}_0$ truth-table reduces to EA. Thus, **SZK** $\subset$ **NISZK**. As **NISZK** $\subset$ **SZK** is true from the definition of **NISZK**, we conclude that **NISZK** = **SZK**. ■

Finally, we give a number of other conditions equivalent to **NISZK** = **SZK**.

Theorem 7.5.11 (conditions for SZK = NISZK) *The following are equivalent:*

1. **SZK** = **NISZK**.
2. **NISZK** *is closed under complement.*
3. **NISZK** *is closed under* $\mathbf{NC}_1$ *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")*

Proof: 1 ⇒ 3. This follows from Corollary 4.5.12, which states that **SZK** is closed under $\mathbf{NC}_1$ truth-table reductions.
3 ⇒ 2 ⇒ 1. The first implication is trivial and the second is Proposition 7.5.10.
1 ⇔ 4. This follows from the Completeness Theorems (Theorem 3.5.1 and 7.2.3), which assert that EA and SDU are complete for **NISZK**, and that ED and SD are complete for **SZK**, and Proposition 7.1.4 (that **NISZK** is closed under Karp reductions).
2 ⇔ 5. This follows from Theorem 7.2.3 (that EA and SDU are complete for **NISZK**) and Proposition 7.1.4 (that **NISZK** is closed under Karp reductions). ■

Theorem 7.5.11 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 $\mathbf{NC}^1$ 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] (*cf.*, Corollary 4.2.1). In light of this, the closure of **NISZK** under complement would not be quite as unexpected, and Theorem 7.5.11 illustrates that proving it would have wider consequences.

The last two conditions in Theorem 7.5.11 show that these questions about noninteractive 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** would have interesting consequences not just for **NISZK**, but also for **SZK**. Note that **NISZK** = **SZK** would imply that every problem in **SZK** = **HVSZK** has a 2-message public-coin **HVSZK** proof, giving a positive answer to Open Problem 5.4.20. By the transformation of Damgård, Goldreich and Wigderson [DGW94], this in turn would imply that every problem in **SZK** has a 4-message public-coin **SZK** proof system (against cheating verifiers, with inverse polynomial soundness error), giving a positive answer to Open Problem 6.5.6.

In summary, it would be very interesting to answer the following question.

Open Problem 7.5.12 *Does* **SZK** = **NISZK***?*

7.6 Other applications of the Completeness Theorem

7.6.1 Problems in NISZK

We can also use the complete problems to place other problems in **NISZK**. To do so, we need only exhibit a reduction from the given problem to one of the complete problems. The following observation will make exhibiting reductions to Entropy Approximation somewhat more convenient: While the definition of EA amounts to the problem approximating entropy up to ± 1, actually it is equivalent to approximating entropy up to any additive constant. More precisely, we have:

Proposition 7.6.1 *There is an efficient transformation that takes a triple (X, t_1, t_2), where X is a distribution encoded by a circuit and $t_1 > t_2$ are rational numbers, and produces a new distribution X' and an integer t such that*

$$\begin{aligned} \mathrm{H}(X) \geq t_1 \quad &\Rightarrow \quad (X', t) \in \mathrm{EA}_Y \\ \mathrm{H}(X) \leq t_2 \quad &\Rightarrow \quad (X', t) \in \mathrm{EA}_Y \end{aligned}$$

The transformation is computable in time polynomial in the input length and $1/(t_1 - t_2)$.

Proof: Let $m = \lceil \frac{3}{t_1 - t_2} \rceil$, $X' = \otimes^m X$, and $t = \lceil m t_2 \rceil + 1$. Then

$$\mathrm{H}(X) \geq t_1 \Rightarrow \mathrm{H}(X') \geq m t_1 \geq m t_2 + 3 \geq t + 1$$

and

$$\mathrm{H}(X) \leq t_2 \Rightarrow \mathrm{H}(X') \leq m t_2 \leq t - 1.$$

■

Explicit proof systems for the problems we consider below can be obtained by combining the reductions to EA given below with the protocol for Entropy Approximation given in Section 7.3. However, for these problems, the construction and analysis of the transformation given by Lemma 7.3.2 can be somewhat simplified, since the distributions are already flat. In particular, there are no "light" samples and hence the second hash function is unnecessary.

The first problem we show to be in **NISZK** is the following promise problem Number of Prime Factors (NPF):

$$\begin{aligned}\mathrm{NPF}_Y &= \{(n,k) \in \mathbb{N} \times \mathbb{N} \,:\, n \text{ has at most } k \text{ distinct prime factors}\}\\ \mathrm{NPF}_N &= \{(n,k) \in \mathbb{N} \times \mathbb{N} \,:\, n \text{ has more than } k \text{ distinct prime factors}\}\end{aligned}$$

Note that, since NPF_N is exactly the complement of NPF_Y, NPF is actually a language.

Proposition 7.6.2 Number of Prime Factors *is in* **NISZK**.

Proof: We reduce NPF to EA. The reduction is based on the following well-known fact.

Fact 7.6.3 *If an odd integer n has exactly k distinct prime factors, then the map from $\mathbb{Z}_n^*$ to $\mathbb{Z}_n^*$ given by $x \mapsto x^2 \bmod n$ is 2^k-to-1.*

Now, we reduce a pair (n,k) to EA as follows: By exhaustive search, find all the prime factors of n less than $4\log n$. Let t be the number of such prime factors, and let m be obtained by dividing all such prime factors out of n. Thus if n has at most (resp., more than) k prime factors, m has at most (resp., more than) $k-t$ prime factors. Now consider the following distribution:

$X_{n,k}$**:** Choose x uniformly in $\mathbb{Z}_m^*$ and output $x^2 \pmod m$.

By Fact 7.6.3, $X_{n,k}$ is uniform on a set of size $|\mathbb{Z}_m^*|/2^\ell = \phi(m)/2^\ell$, where ℓ is the number of prime factors of m, and hence $\mathrm{H}(X_{n,k}) = (\log \phi(m)) - \ell$. Now, since m has no prime factors smaller than $(\log n)/4$,

$$m \geq \phi(m) = m \prod_{p|m} \left(1 - \frac{1}{p}\right) \geq m \cdot \left(1 - \frac{1}{4\log n}\right)^{\log m} \geq m \cdot \left(1 - \frac{\log m}{4\log n}\right) \geq 3m/4.$$

Therefore,

$$\begin{aligned}(n,k) \in \mathrm{NPF}_Y \;&\Rightarrow\; \mathrm{H}(X_{n,k}) \geq \log(3m/4) - (k-t) > \log m - k + t - .5.\\ (n,k) \in \mathrm{NPF}_N \;&\Rightarrow\; \mathrm{H}(X_{n,k}) \leq \log m - (k-t+1) = \log m - k + t - 1\end{aligned}$$

Thus, taking $X = X_{n,k}$, $t_1 = \log m - k + t - .5$, and $t_2 = \log m - k + t - 1$ in Proposition 7.6.1, we see that NPF reduces to EA. ■

Now we consider a version Quadratic Nonresiduosity, which was the first problem shown to be in **NISZK**.

$$\begin{aligned}\mathrm{QNR}_Y &= \{(n,x) \in \mathbb{N} \times \mathbb{N} \,:\, x \text{ is a quadratic nonresidue modulo } n,\, n \text{ is odd},\\ &\qquad \text{and } n \text{ has exactly two distinct prime factors}\},\\ \mathrm{QNR}_N &= \mathbb{N} \times \mathbb{N} \setminus \mathrm{QNR}_Y\end{aligned}$$

Proposition 7.6.4 ([BDMP91]) Quadratic Nonresiduosity *is in* **NISZK**.

Proof: The fact that n is odd and has exactly two distinct prime factors can be proven in **NISZK** by Proposition 7.6.2. (Polynomial-time primality testing algorithms, as in [SS77, Mil76, Rab80] can be used to rule out n with exactly one distinct prime factor.) Thus, we may assume that n is of the correct form, and need only give a reduction to EA that works in this case. We also may assume that both of the prime divisors of n are larger than $(\log n)/4$, for otherwise one can factor n and decide if x is a quadratic residue in polynomial time. Finally, we may assume that $\gcd(x, n) = 1$, as $(n, x) \in \mathrm{QNR}_N$ if this is not the case, and this can be checked in polynomial time. Consider the following distribution:

$X_{n,x}$: Choose y uniformly in $\mathbb{Z}_n^*$. With probability $1/2$, output $y^2 \bmod n$, and with probability $1/2$, output $x \cdot y^2 \bmod n$.

First, suppose that x is a quadratic residue modulo n. Then $X_{n,x}$ is distributed uniformly on the quadratic residues modulo n, which, by Fact 7.6.3, is a set of size $\phi(n)/4 \leq n/4$. Thus, $\mathrm{H}(X_{n,x}) \leq \log n - 2$.

On the other hand, if x is a quadratic nonresidue modulo n, then the elements of $\mathbb{Z}_n^*$ of the form $x \cdot y^2$ are disjoint from those of the form y^2, so $X_{n,x}$ is uniformly distributed on a set of size $\phi(n)/2$. As in the proof of Proposition 7.6.2, the assumption that n has no small prime factors implies that $\phi(n) \geq 3n/4$, so $\mathrm{H}(X_{n,x}) \geq \log((3n/4)/2) \geq \log n - 1.5$.

Taking $X = X_{n,k}$, $t_1 = \log n - 1.5$, and $t_2 = \log n - 2$ in Proposition 7.6.1, we see that QNR reduces to EA. ∎

Blum *et. al.* [BDMP91] actually consider a slightly different version of QNR, in which the YES instances (n, x) also have the constraint that n is not a perfect square and that x has Jacobi symbol 1. They show that their version of QNR is actually in **NIPZK**.

The final example we consider is a variant of GRAPH ISOMORPHISM, observed to be in **NISZK** by Bellare and Rogaway [BR90]. If G is a graph, then $\mathrm{Aut}(G)$ denotes the group of isomorphisms from G to itself, also known as *automorphisms*. G is said to be *rigid* if $\mathrm{Aut}(G)$ consists of only the identity map. The problem we consider is RIGID GRAPHS (RG), given by:

$$\begin{aligned} \mathrm{RG}_Y &= \{G : G \text{ is rigid}\} \\ \mathrm{RG}_N &= \{G : G \text{ is not rigid}\} \end{aligned}$$

Proposition 7.6.5 ([BR90]) RIGID GRAPHS *is in* **NISZK**.

Proof: Consider the following distribution for any graph G:

X_G: Uniformly select a permutation π on the vertices of G and output $\pi(G)$.

Standard group theory implies that X_G is distributed uniformly on a set of size $n!/|\mathrm{Aut}(G)|$. So if G is rigid, $\mathrm{H}(X_G) = \log n!$, whereas if G is not rigid, $\mathrm{H}(X_G) \leq \log(n!/2) = \log n! - 1$. Thus, taking $X = X_G$, $t_1 = \log n!$, and $t_2 = \log n! - 1$ in Proposition 7.6.1, we see that RG reduces to EA. ∎

7.6.2 A Polarization Lemma for SDU

Combining Lemmas 7.3.5 and 7.4.1, we obtain an SDU-analogue of the Polarization Lemma (Lemma 3.1.12).

Lemma 7.6.6 *There is a polynomial-time computable function that takes a distribution X on $\{0,1\}^n$ (encoded by a circuit) and a parameter k (in unary) and outputs a new distribution X' on $\{0,1\}^{n'}$ such that*

$$\begin{aligned} \mathrm{StatDiff}\left(X, U_n\right) \leq \frac{1}{n} \quad &\Rightarrow \quad \mathrm{StatDiff}\left(X', U_{n'}\right) \leq 2^{-k} \\ \mathrm{StatDiff}\left(X, U_n\right) \geq 1-1/n \quad &\Rightarrow \quad \mathrm{StatDiff}\left(X', U_{n'}\right) \geq 1-2^{-k} \end{aligned}$$

Moreover, in the latter case, the support of X' is at most a 2^{-k} fraction of $\{0,1\}^{n'}$.

This can be generalized somewhat, observing that the reduction from SDU to EA given in Lemma 7.4.1 actually works for more general thresholds:

Lemma 7.6.7 (Polarization Lemma for SDU) *Let $\alpha, \beta : \mathbb{N} \to \mathbb{N}$ be any two functions such that $\alpha(n)$ and $\beta(n)$ are computable in time $\mathrm{poly}(n)$ and, for some constant c,*

$$\log\left(\frac{1}{1-\beta(n)}\right) \geq \alpha(n) \cdot n + \frac{1}{n^c}.$$

Then, there is a polynomial-time computable function that takes a distribution X on $\{0,1\}^n$ (encoded by a circuit) and a parameter k (in unary) and outputs a new distribution X' on $\{0,1\}^{n'}$ such that

$$\begin{aligned} \mathrm{StatDiff}\left(X, U_n\right) \leq \alpha(n) \quad &\Rightarrow \quad \mathrm{StatDiff}\left(X', U_{n'}\right) \leq 2^{-k} \\ \mathrm{StatDiff}\left(X, U_n\right) \geq \beta(n) \quad &\Rightarrow \quad \mathrm{StatDiff}\left(X', U_{n'}\right) \geq 1-2^{-k} \end{aligned}$$

Moreover, in the latter case, the support of X' is at most a 2^{-k} fraction of $\{0,1\}^{n'}$.

Lemma 7.6.7 is proven using Claim 7.4.2 together with Proposition 7.6.1. We do not know whether an analogous lemma can be proven for any pair of constant thresholds $0 < \alpha < \beta < 1$. One might hope to obtain such a result using the approach used in the Polarization Lemma for statistical difference — alternating procedures which increase and decrease statistical difference. However, while the Direct Product construction for increasing statistical difference also applies to SDU, the XOR construction does not, as neither distribution it produces is uniform even if one of the original distributions is uniform. Thus, the following problem remains open:

Open Problem 7.6.8 *Can the Polarization Lemma for SDU be extended to any pair of constant thresholds $0 < \alpha < \beta < 1$?*

Chapter 8

Conclusion

In this thesis, we have addressed a number of fundamental questions about statistical zero-knowledge proofs. Our main tools in this investigation were the two complete problems Entropy Difference and Statistical Difference. The central role played by these problems in our study is a dramatic illustration of the power of completeness as a *positive* tool.

First, these complete problems gave us a tight characterization of the problems that possess statistical zero-knowledge proofs. Namely, we saw that the class **SZK** can be identified with "approximate statistical properties of samplable distributions." Then these complete problems provided a starting point for understanding a number of important aspects of statistical zero-knowledge proofs. Among the issues we addressed were efficiency, closure properties, private coins vs. public coins, honest verifiers vs. cheating verifiers, and interactive vs. noninteractive proofs. Although we managed to answer some of the basic questions in these areas, a number of intriguing problems remain. We have described many of these open problems in the relevant chapters, but there are a few worth highlighting here.

Efficient SZK proof systems. In the course of this thesis, we have shown how to transform an arbitrary **HVSZK** proof system into one with various desirable additional properties, such as being zero knowledge versus cheating verifiers (Theorem 6.3.1), exchanging a constant number of messages (Corollary 4.1.1), and using public coins (Theorem 5.1.1). However, we do not know how to achieve the constant-message property together with either of the other two properties. In particular, the following questions remain open (Open Problems 6.5.6, 5.4.20, and 7.5.12):

- Does every problem in **HVSZK** possess a constant-message **SZK** proof system?
- Does every problem in **HVSZK** possess a constant-message public-coin **HVSZK** proof system?
- Does **HVSZK** = **NISZK**?

Recall that a positive answer to the second or third questions implies a positive answer to the previous ones.

Extending more techniques to CZK. One broad research project is to extend more of the techniques developed here to other forms of zero-knowledge proofs, such as computational zero-knowledge proofs and zero-knowledge "arguments" [BCC88] (which we did not discuss). In particular, three questions about computational zero knowledge stand out (Open Problems 4.7.5, 5.4.19, 6.5.4).

- Can one exhibit a natural complete problem for (honest-verifier) computational zero knowledge? or at least give a nontrivial result such as Proposition 4.7.3 without restricting to public coins?
- Can private-coin (honest-verifier) computational zero-knowledge proofs be transformed into public-coin ones? (We showed how to do this for 3-message private-coin proofs.)
- Does honest-verifier computational zero knowledge equal cheating-verifier computational zero knowledge? (We answered this question in the positive for the case of public-coin proofs.)

Recall that it is only of interest to answer these questions unconditionally, as essentially everything about computational zero knowledge has been resolved under the assumption that one-way functions exist.

More complete problems. Another general research avenue is to exhibit additional natural complete problems for **SZK**. In particular, it would be very interesting to exhibit a combinatorial or number-theoretic complete problem, such as one of the problems of cryptographic interest known to be in **SZK**. While we have primarily used **SZK**-completeness as a positive tool, it also could provide strong evidence of intractability, as **SZK** contains many problems believed to be hard. Indeed, we are in need of alternatives to **NP**-completeness for hardness results, as there are important cases in which it seems out of reach. For example, for most of the problems on which modern cryptography is based, we would like to prove hardness results, but **NP**-hardness is unlikely due to these problems lying in $\mathbf{AM} \cap \mathbf{co\text{-}AM}$ (*cf.,* [Bra79, BHZ87, GG98a, GG98b]). In contrast, this does not rule out the possibility of **SZK**-hardness, as $\mathbf{SZK} \subset \mathbf{AM} \cap \mathbf{co\text{-}AM}$.

SZK vs. PZK. A final open problem is the relationship between statistical zero knowledge and perfect zero knowledge (Open Problem 4.6.14). In fact, it was this question, asked to us by Shafi Goldwasser, that started the research in this thesis, and unfortunately the answer remains a mystery. For a number of years after zero-knowledge proofs were introduced, there were no natural examples of problems known to be in **SZK** but not known to be in **PZK**; now the complete problems STATISTICAL DIFFERENCE and ENTROPY DIFFERENCE are examples of such problems. On one hand, this may be regarded as evidence that the classes are different. On the other hand, the problem of proving that $\mathbf{SZK} = \mathbf{PZK}$ is now reduced to giving a perfect zero-knowledge proof for either of the complete problems.

Appendix A

Chernoff Bounds

The following useful bound shows that if one has n independent events, each which occur with probability p, then roughly np of the events occur with high probability.

Theorem A.1 (Chernoff Bound [Che52]) *Suppose $X_1, \ldots, X_n$ are independent random variables such that for all i, $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p$. Let $X = \frac{1}{n}\sum_{i=1}^{n} X_i$. Then for any $\delta > 0$,*

$$\begin{aligned} \Pr[X \geq p + \delta] &\leq \exp\left(-2n\delta^2\right), \text{ and} \\ \Pr[X \leq p - \delta] &\leq \exp\left(-2n\delta^2\right). \end{aligned}$$

The following generalization of the Chernoff Bound to non-Boolean random variables will also be useful to us.

Theorem A.2 (Hoeffding Inequality [Hoe63]) *Suppose $X_1, \ldots, X_n$ are independent random variables with mean μ, taking values in the real interval $[a, b]$, and $X = \frac{1}{n}\sum_{i=1}^{n} X_i$. Then for any $\Delta > 0$,*

$$\begin{aligned} \Pr[X \geq \mu + \Delta] &\leq \exp\left(\frac{-2n\Delta^2}{(b-a)^2}\right), \text{ and} \\ \Pr[X \leq \mu - \Delta] &\leq \exp\left(\frac{-2n\Delta^2}{(b-a)^2}\right). \end{aligned}$$

A proof of this version of the Hoeffding Inequality can be found in [Hof95, Sec. 7.2], and the Chernoff Bound above can be obtained by setting $a = 0$, $b = 1$, $\Delta = \delta$, and $\mu = p$.

Appendix B

Hashing Lemmas

In this appendix, we prove the two lemmas about 2-universal hash functions that we used — Lemmas 5.4.10 and 6.4.5.

B.1 Proof of Lemma 5.4.10

In this section, we prove the hashing lemma used to analyze the transformation from private-coin zero-knowledge proofs to public-coin ones in Chapter 5. We restate the lemma here:

Lemma B.1.1 (implicit in [Oka96]) *Let $\mathcal{H}$ be a 2-universal family of hash functions mapping a domain $\mathcal{D}$ to a range $\mathcal{R}$ and let 0 be any fixed element of $\mathcal{R}$. Let Z be a distribution on $\mathcal{D}$ such that with probability $1-\delta$ over z selected according to Z, $\Pr[Z=z] \leq \varepsilon/|\mathcal{R}|$. Then the following two distributions have statistical difference at most $3(\delta+\varepsilon^{1/3})$:*

(A) *Choose h uniformly in $\mathcal{H}$. Select z according to Z conditioned on $h(z)=0$. Output (h,z).*

(B) *Choose z according to Z. Select h uniformly in $\{h' \in \mathcal{H} \,:\, h(z)=0\}$. Output (h,z).*

We denote the two distributions on pairs (h,z) in Lemma B.1.1 by $A=(A_{\mathcal{H}},A_Z)$ and $B=(B_{\mathcal{H}},B_Z)$. By the definition of statistical difference, it suffices to show that for every set $S \subset \mathcal{H}\times\mathcal{D}$, $\Pr[A\in S]-\Pr[B\in S] \leq 3(\delta+\varepsilon^{1/3})$. In order to do this, we first will argue that for "most" pairs (h,z), $\Pr[A=(h,z)]$ is not too much greater than $\Pr[B=(h,z)]$. Observe that both distributions A and B only output pairs (h,z) such that $h(z)=0$. Now, for any $(h,z)\in\mathcal{H}\times\mathcal{D}$ such that $h(z)=0$, we have

$$\begin{aligned}\Pr[A=(h,z)] &= \Pr[A_{\mathcal{H}}=h]\cdot\Pr[A_Z=z|A_{\mathcal{H}}=h]\\ &= \frac{1}{|\mathcal{H}|}\cdot\frac{\Pr[Z=z]}{\sum_{w\in h^{-1}(0)}\Pr[Z=w]},\end{aligned}$$

and

$$\Pr[B=(h,z)] = \Pr[B_Z=z]\cdot\Pr[B_H=h|B_Z=z]$$

$$\begin{aligned} &= \Pr[Z = z] \cdot \frac{1}{|\{h' : h'(z) = 0\}|} \\ &= \Pr[Z = z] \cdot \frac{|\mathcal{R}|}{|\mathcal{H}|}, \end{aligned}$$

where the last equality follows from 2-universality.

Thus, showing that $\Pr[A = (h, z)]$ is not too much greater than $\Pr[B = (h, z)]$ for most pairs (h, z) amounts to showing that for most h, $\sum_{w \in h^{-1}(0)} \Pr[Z = w]$ is not too much smaller than $1/|\mathcal{R}|$. In order to prove a lower bound on this sum (for most h), we restrict the sum to a slightly smaller set of w's. Let $L = \{w \in \mathcal{D} : \Pr[Z = w] \leq \varepsilon/|\mathcal{R}|\}$, so by hypothesis, $\Pr[Z \in L] = 1 - \delta$. For $w \in \mathcal{D}$ and $h \in \mathcal{H}$, define indicator functions

$$\chi_w(h) = \begin{cases} 1 & \text{if } h(w) = 0 \\ 0 & \text{otherwise} \end{cases}$$

Define $f(h) = \sum_{w \in L} \Pr[Z = w] \cdot \chi_w(h)$. Thus,

$$\sum_{w \in h^{-1}(0)} \Pr[Z = w] = \sum_{w \in \mathcal{D}} \Pr[Z = w] \cdot \chi_w(h) \geq f(h)$$

By 2-universality, for h selected uniformly in $\mathcal{H}$, the random variables $\{\chi_w(h)\}_{w \in \mathcal{D}}$ each have mean $1/|\mathcal{R}|$ and are pairwise independent. Thus,

$$\operatorname*{E}_h[f(h)] = \sum_{w \in L} \frac{\Pr[Z = w]}{|\mathcal{R}|} = \frac{1 - \delta}{|\mathcal{R}|}$$

and

$$\begin{aligned} \operatorname*{Var}_h[f(h)] &\leq \sum_{w \in L} \frac{\Pr[Z = w]^2}{|\mathcal{R}|} \\ &\leq \sum_{w \in L} \frac{\Pr[Z = w] \cdot \varepsilon}{|\mathcal{R}|^2} \\ &\leq \frac{\varepsilon}{|\mathcal{R}|^2} \end{aligned}$$

By Chebyshev's Inequality,

$$\Pr_h\left[f(h) - \frac{1 - \delta}{|\mathcal{R}|} < \frac{-\varepsilon^{1/3}}{|\mathcal{R}|}\right] \leq \frac{\operatorname{Var}_h(f(h))}{(\varepsilon^{1/3}/|\mathcal{R}|)^2} \leq \varepsilon^{1/3}.$$

Let $G = \{h \in \mathcal{H} : f(h) \geq (1 - \delta - \varepsilon^{1/3})/|\mathcal{R}|\}$ be the set "good" h's for which $f(h)$ is not too much smaller than $1/|\mathcal{R}|$. Then for every $z \in \mathcal{D}$ and $h \in G$,

$$\Pr[A = (h, z)] \leq \frac{\Pr[Z = z]}{|\mathcal{H}|} \cdot \frac{|\mathcal{R}|}{1 - \delta - \varepsilon^{1/3}} = \frac{\Pr[B = (h, z)]}{1 - \delta - \varepsilon^{1/3}}.$$

Thus, for any $S \subset \mathcal{H} \times \mathcal{D}$,

$$\begin{aligned}
\Pr[A \in S] &\leq \Pr[A \in S \text{ and } A_{\mathcal{H}} \in G] + \Pr[A_{\mathcal{H}} \notin G] \\
&\leq \frac{\Pr[B \in S \text{ and } B_{\mathcal{H}} \in G]}{1 - \delta - \varepsilon^{1/3}} + \varepsilon^{1/3} \\
&\leq \Pr[B \in S] + \left(\frac{\delta + \varepsilon^{1/3}}{1 - \delta - \varepsilon^{1/3}}\right) \cdot \Pr[B \in S] + \varepsilon^{1/3} \\
&\leq \Pr[B \in S] + 3(\delta + \varepsilon^{1/3}),
\end{aligned}$$

(as long as $\delta + \varepsilon^{1/3} \leq 1/2$, which we may assume as otherwise the lemma is trivially satisfied). This completes the proof.

B.2 Proof of Lemma 6.4.5

Here we provide a proof of the hashing lemma used to analyze the transformation from honest-verifier zero-knowledge proofs to cheating-verifier zero-knowledge proofs. We restate the lemma here:

Lemma B.2.1 *There exists a universal constant $c > 0$, so that the following holds: Let $\mathcal{H} = \mathcal{H}_{\ell,m}$ be the family of affine-linear maps from $\mathcal{D} = \{0,1\}^\ell$ to $\mathcal{R} = \{0,1\}^m$. Let $S \subset \mathcal{H}$ be such that $|S| \geq \delta|\mathcal{H}|$. Let $\varepsilon = |\mathcal{R}|/|\mathcal{D}|$. Then*

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

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

$B = (B_{\mathcal{H}}, B_X)$: *Choose $x \leftarrow \mathcal{D}$. Select $h \leftarrow S \cap \mathcal{H}_x$.*[1] *Output (h, x).*

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

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

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 Lemma B.2.1 for the special case of perfect hash functions.

Sublemma B.2.2 *Part 1 of Lemma B.2.1 holds when S contains only perfect hash functions.*

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

Claim B.2.3 $\mathrm{StatDiff}(A_X, B_X) \leq 3\varepsilon^{1/3}/\delta$.

[1] Recall that $\mathcal{H}_x$ denotes $\{h \in \mathcal{H} : h(x) = 0\}$.

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

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

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

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

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 $h \in \mathcal{H}$. This will enable us to use the 2-universality of $\mathcal{H}$ and 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 $\Pr_{h\in\mathcal{H}}[\chi_x(h) = 1] = 1/|\mathcal{R}| = 1/(\varepsilon \cdot |\mathcal{D}|)$. Thus,

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

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

By Chebyshev's Inequality,

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

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

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

Recall that

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

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

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

And the claim is proved. □

We are now ready to complete the proof of the sublemma. For all $x \in \mathcal{D}$ 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 $\mathrm{StatDiff}(A, B) = \mathrm{StatDiff}(A_X, B_X) \le 3\varepsilon^{1/3}/\delta$, and the sublemma is established. ■

Before we argue Part 1 of Lemma B.2.1 in general, we will show how Part 2 follows from Sublemma B.2.2. In the sequel, it will be convenient to introduce the following notation: For any subset $I \subseteq \mathcal{H}$ and $x \in \mathcal{D}$, we will write I_x to denote the set $\{h \in I : h(x) = 0\}$.

In order to apply Sublemma B.2.2, 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,

$$|S'| \ge |S| - \varepsilon\cdot|\mathcal{H}| \ge (1 - \frac{\varepsilon}{\delta})|S| \ge (\delta - \varepsilon)\cdot|\mathcal{H}|.$$

We define the following two modifications of the distributions A and B, using S' instead of S:

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

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

The following claim establishes Part 2 of the Hashing Lemma:

Claim B.2.4 *Let $\epsilon_1 \stackrel{\text{def}}{=} \frac{3\varepsilon^{1/3}}{\delta-\varepsilon}$. For at least a $(1 - \sqrt{\epsilon_1})$ fraction of $x \in \mathcal{D}$, $\frac{|S_x|}{|\mathcal{H}_x|} \ge \delta/2$.*

Proof of claim: By the definition of A'_X,

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

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

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

Thus,

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

where the last equality follows from $\varepsilon\cdot|\mathcal{D}| = |\mathcal{R}|$ and $|\mathcal{R}|\cdot|\mathcal{H}_x| = |\mathcal{H}|$. 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 $x \in \mathcal{D}$,

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

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, since $|S'| \geq (1-\frac{\varepsilon}{\delta})\cdot|S|$, the statistical difference between A and A' can be at most ε/δ. 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 B.2.5 *For every $x \in \mathcal{D}$, $\Pr_{h\in\mathcal{H}_x}[h \text{ is imperfect}] \leq \varepsilon$.*

Proof of claim: Observe that for any $x \in \mathcal{D}$, $\mathcal{H}_x$ 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 $x \in \mathcal{D}$, 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 B.2.4 and Claim B.2.5 above, we have that for at least a $(1-\sqrt{\epsilon_1})$ fraction of $x \in \mathcal{D}$, this probability is at most $\epsilon_2 \stackrel{\text{def}}{=} \varepsilon\cdot(2/\delta)$. Thus, $\mathrm{StatDiff}\,(B, B') \leq (1-\sqrt{\epsilon_1})\cdot\epsilon_2+\sqrt{\epsilon_1} \leq$

$\epsilon_2 + \sqrt{\epsilon_1}$. We have already observed that $\mathrm{StatDiff}\,(A', A) \leq \frac{\varepsilon}{\delta}$, and Sublemma B.2.2 showed that $\mathrm{StatDiff}\,(B', A') \leq \epsilon_1$. Hence $\mathrm{StatDiff}\,(A, B) \leq \epsilon_1 + \frac{\varepsilon}{\delta} + \epsilon_2 + \sqrt{\epsilon_1}$, and the Hashing Lemma is established. ∎

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*, pages 469–478, Las Vegas, Nevada, 29 May–1 June 1995.

[AH91] William Aiello and Johan Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *Journal of Computer and System Sciences*, 42(3):327–345, June 1991.

[AB84] Miklós 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.

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 284–293, El Paso, Texas, 4–6 May 1997. See also ECCC TR96-065.

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, January 1998.

[Bab85] László Babai. Trading group theory for randomness. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 421–429, Providence, Rhode Island, 6–8 May 1985.

[BM88] László 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.

[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.

[BMO90a] 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, Baltimore, Maryland, 14–16 May 1990.

[BMO90b] 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, Baltimore, Maryland, 14–16 May 1990.

[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.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive 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 Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25:127–132, 1987.

[BP89] Joan Boyar and René Peralta. On the concrete complexity of zero-knowledge proofs. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 507–525. Springer-Verlag, 1990, 20–24 August 1989.

[Bra79] Gilles Brassard. Relativized cryptography. In *20th Annual Symposium on Foundations of Computer Science*, pages 383–391, San Juan, Puerto Rico, 29–31 October 1979. IEEE.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, October 1988.

[CW79] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, April 1979.

[Che52] Hermann Chernoff. A measure of the asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[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, Shaker Heights, Ohio, 3–5 1971 1971.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, Inc., 2nd edition, 1991.

[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].

[Dam93] Ivan B. Damgård. Interactive hashing can simplify zero-knowledge protocol design without computational assumptions (extended abstract). In Douglas R. Stinson, editor, *Advances in Cryptology—CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 100–109. Springer-Verlag, 22–26 August 1993.

[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 *35th Annual Symposium on Foundations of Computer Science*, pages 454–465, Santa Fe, New Mexico, 20–22 November 1994. IEEE.

[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.

[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.

[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.

[DY90] Alfredo De Santis and Moti Yung. Cryptographic applications of the non-interactive metaproof and many-prover systems. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology—CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 366–377. Springer-Verlag, 1991, 11–15 August 1990.

[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.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions in Information Theory*, IT-22(6):644–654, 1976.

[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. Full version to appear in *SIAM J. on Computing.*

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer-Verlag, 1985, 19–22 August 1984.

[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.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 1999. To appear. Preliminary version in *FOCS '90.*

[FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 526–545. Springer-Verlag, 1990, 20–24 August 1989.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 416–426, Baltimore, Maryland, 14–16 May 1990.

[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.

[FSS84] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.

[Gol90] Oded Goldreich. A note on computational indistinguishability. *Information Processing Letters*, 34(6):277–281, 28 May 1990.

[Gol93] Oded Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, 6(1):21–53, 1993.

[Gol95] Oded Goldreich. *Foundations of Cryptography (Fragments of a Book)*. Weizmann Institute of Science, February 1995. Available from `http://www.eccc.uni-trier.de/eccc/`.

[Gol99] Oded Goldreich. Some useful trivial tricks. `http://www.wisdom.weizmann.ac.il/~oded/tricks.html`, September 1999.

[GG98a] Oded Goldreich and Shafi Goldwasser. On the limits of non-approximability of lattice problems. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 1–9, Dallas, 23–26 May 1998.

[GG98b] Oded Goldreich and Shafi Goldwasser. On the possibility of basing cryptography on the assumption that P $\neq$ NP. Theory of Cryptography Library: Record 98-05, February 1998. `http://theory.lcs.mit.edu/~tcryptol`.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer-Verlag, 17–21 August 1997. See also ECCC TR96-056.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[GK96a] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for np. *Journal of Cryptology*, 9(3):167–190, 1996.

[GK96b] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, February 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 Symposium on Theory 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 ACM*, 38(1):691–729, 1991.

[GNW95] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR lemma. Technical Report TR95–050, Electronic Colloquium on Computational Complexity, March 1995. `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 *32nd Annual Symposium on Foundations of Computer Science*, pages 59–68, San Juan, Puerto Rico, 1–4 October 1991. IEEE. See also full version, July 1996.

[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, 23–26 May 1998.

[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. Springer-Verlag, 1999, 15–19 August 1999. To appear.

[GS98] Oded Goldreich and Madhu Sudan. Computational indistinguishability: A sample hierarchy. In *Proceedings of the Thirteenth Annual IEEE Conference on Computational Complexity*, Buffalo, NY, June 1998. IEEE Computer Society Press.

[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.

[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.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, August 1999.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[Hof95] Micha Hofri. *Analysis of Algorithms: Computational Methods & Mathematical Tools*. Oxford University Press, 1995.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, Washington, 15–17 May 1989.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235, Research Triangle Park, North Carolina, 30 October–1 November 1989. IEEE.

[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] R. 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.

[Kil94] Joe Kilian. On the complexity of bounded-interaction and noninteractive zero-knowledge proofs. In *35th Annual Symposium on Foundations of Computer Science*, pages 466–477, Santa Fe, New Mexico, 20–22 November 1994. IEEE.

[KMO89] Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 474–479, Research Triangle Park, North Carolina, 30 October–1 November 1989. IEEE.

[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.

[KvM99] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, pages 659–667, Atlanta, 1–4 May 1999.

[LLS75] Richard E. Ladner, Nancy A. Lynch, and Alan L. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–123, December 1975.

[Lev73] Leonid A. Levin. Universal'nyĭe perebornyĭe zadachi (Universal search problems : in Russian). *Problemy Peredachi Informatsii*, 9(3):265–266, 1973.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, October 1992.

[Mil76] Gary L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13(3):300–317, 1976. Working papers presented at the ACM-SIGACT Symposium on the Theory of Computing (Albuquerque, N.M., 1975).

[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*, pages 649–658, Philadelphia, Pennsylvania, 22–24 May 1996. See also preprint of full version, Oct. 1997.

[Ore87] Yair Oren. On the cunning power of cheating verifiers: Some observations about zero knowledge proofs (extended abstract). In *28th Annual Symposium on Foundations of Computer Science*, pages 462–471, Los Angeles, California, 12–14 October 1987. IEEE.

[Ost91] Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference*, pages 133–138, Chicago, Illinois, 30 June–3 July 1991. IEEE Computer Society Press,.

[OVY93] Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Interactive hashing simplifies zero-knowledge protocol design. In *Proceedings of Eurocrypt '93, Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[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 $\mathcal{NP}$. In *37th Annual Symposium on Foundations of Computer Science*, pages 494–503, Burlington, Vermont, 14–16 October 1996. IEEE.

[Rab80] Michael O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 387–394, Baltimore, Maryland, 14–16 May 1990.

[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.

[SV97] Amit Sahai and Salil P. Vadhan. A complete promise problem for statistical zero-knowledge. In *38th Annual Symposium on Foundations of Computer Science*, pages 448–457, Miami Beach, Florida, 20–22 October 1997. IEEE.

[Sch88] Uwe Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37(3):312–323, December 1988.

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, October 1992.

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 330–335, Boston, Massachusetts, 25–27 April 1983.

[Sip97] Michael Sipser. *Introduction to the Theory of Computation.* PWS Publishing, 1997.

[SS77] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM Journal on Computing*, 6(1):84–85, 1977.

[Spi71] Philip M. Spira. On time-hardware complexity tradeoffs for Boolean functions. In *Proceedings of the 4th Hawaii Conference on Systems Sciences*, pages 525–527, 1971.

[TW87] Martin Tompa and Heather Woll. Random self-reducibility and zero knowledge interactive proofs of possession of information. In *28th Annual Symposium on Foundations of Computer Science*, pages 472–482, Los Angeles, California, 12–14 October 1987. IEEE.

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 November 1982. IEEE.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, 27–29 October 1986. IEEE.