A Study of Perfect Zero-Knowledge Proofs

by

Lior Malka

B.Sc., Ben-Gurion University, Israel, 2001
M.Sc., Ben-Gurion University, Israel, 2004

A Dissertation Submitted in Partial Fulfillment
of the Requirements for the Degree of

Doctor of Philosophy

in the Computer Science Department

A Study of Perfect Zero-Knowledge Proofs

by

Lior Malka

B.Sc., Ben-Gurion University, Israel, 2001
M.Sc., Ben-Gurion University, Israel, 2004

Supervisory Committee

Abstract

Perfect zero-knowledge proofs enable one party (the prover) to prove an assertion to another party (the verifier) but without revealing anything but the truth of the assertion. The class of problems admitting such proofs is rich, including GRAPH-ISOMORPHISM, QUADRATIC-RESIDUOUSITY, and other problems that play a key role in cryptography and complexity theory. Due to their strong privacy guarantee, perfect zero-knowledge proofs are very difficult to study. Despite extensive research since the 1980s, especially in the area of statistical zero-knowledge proofs, many fundamental questions about them remain open, and it is not even clear how to address these questions. This thesis initiates a general investigation of perfect zero-knowledge proofs. Our main results are as follows.

- We prove that all the known problems admitting perfect zero-knowledge (**PZK**) proofs can be characterized as non-interactive instance-dependent commitment schemes, and use this result to generalize and strengthen previous results, as well as to prove new results about **PZK** problems.

- We give a new *error shifting technique* that allows us to overcome barriers in the study of **PZK**. Using this technique we present the first complete problem for the class of problems admitting non-interactive perfect zero-knowledge proofs (**NIPZK**), and the first hard problem for the class of problems admitting public-coin **PZK** proofs.

- We make the first investigation into one of the most important questions in the field, namely, whether the number of rounds in **PZK** proofs can be collapsed to a constant. We give the first perfectly hiding commitment scheme, and prove that obtaining such scheme that is also constant round is *equivalent* to collapsing the rounds in **PZK** proofs to a constant.

Contents

List of Figures

Acknowledgements

When I started my PhD in computer science, back in January 2004, I had no idea what difficulties I would meet down the road. Surprisingly, it was not the Canadian winter. My supervisors at the time, Bruce Kapron and Valerie King, hosted me at their house, took excellent care of me, and introduced me to the city of Victoria. And although we worked together, my relationship with Bruce and Valerie is more like a family. They helped and supported me in everything that I did.

Venkatesh Srinivasan became my supervisor early in my first year. We were both interested in complexity theory, so it was natural for us to study zero-knowledge protocols. We spent a lot of time together, and I learned a lot from him, not just about theoretical computer science, but also about the academic world.

There is a very big group of friends who instilled me with confidence, encouraged me when times were hard, and more importantly- trusted me. Each of these people did or said something that had a profound impact on my life. A partial list of these people include: Eti Vainer, Katarina Sebestova, Christiaan Piller, Ollie Ayling, Elad Schiller, Cassandra Morton, Tricia Best, Joe Parsons, Anissa Agah, Jennifer Murdoch, Warren Shenkenfelder, Allan Scott, Darcy Lindberg, Lindsey McDowell, and John Orser. On the academic side: Ivan Visconti, Giuseppe Persiano, Omer Reingold, Oded Goldreich, Salil Vadhan, Cynthia Dwork, Wendy Myrvold, Amos Beimel, Ilia Goldstein, Charlie Rackoff, Amit Sahai, Rafail Ostrovsky, Omkant Pandey, and Vipul Goyal. Sorry if I have forgotten a few.

My parents, siblings, cousins, uncles, and aunts were always there for me, and without their help I would not be able to get this far. Above all I want to thank my grandmother Aliza. When I was 22, infected with the travel bug and complaining about life as a computer science bachelor student, she said that one day I will be a doctor. I really thought that she was joking.

Chapter 1

Introduction

Perfect zero-knowledge protocols enable one party (the prover) to prove an assertion to another party (the verifier) but without revealing anything but the truth of the assertion [46]. Other variants of these protocols (*statistical* and *computational* zero-knowledge protocols) have been studied extensively. In these variants the prover is allowed to leak a small amount of information to the verifier. In contrast, perfect zero-knowledge protocols require that the prover leak absolutely no information to the verifier. This rigid definition provides the highest level of privacy to the prover, but it also makes perfect zero-knowledge protocols hard to study. This is so because there are many useful tools for zero-knowledge protocols, but these tools have a side effect that they cause the prover to leak a small amount of information to the verifier. Such tools greatly facilitated the study of statistical and computational zero-knowledge protocols, but they cannot be used to study perfect zero-knowledge protocols. Consequently, many fundamental questions that have been answered in the statistical and the computational settings remain open in the perfect setting.

The goal of this thesis is to initiate a study of perfect zero-knowledge proofs. In this chapter we give background and context to our research. We begin with an informal discussion that motivates the notion of zero-knowledge. Then we informally describe the notions of perfect, statistical and computational zero-knowledge protocols, and explain how the study of statistical zero-knowledge proofs inspired this thesis. Our results appear at the end of the section.

1.1 Zero-Knowledge Protocols

Zero-knowledge protocol enable one party (*the prover*) to prove an assertion to another party (*the verifier*) but without revealing any information other than the validity of the assertion [46]. To demonstrate this concept, we use the zero-knowledge protocol of [41] for GRAPH-ISOMORPHISM. In this protocol the input to the prover and the verifier is a pair of graphs $\langle G_0, G_1\rangle$, and the goal of the prover is to convince the verifier that the graphs are isomorphic, but without revealing any information. If G_0 and G_1 are isomorphic, then the prover also has a permutation π such that $\pi(G_0) = G_1$ (i.e., π is an isomorphism).

Before we describe the protocol of [41], notice that a simple idea is to have the prover send π to the verifier, and let the verifier use π to check whether the graphs are isomorphic or not. This guarantees that the verifier accepts only if the graphs are isomorphic. However, from a cryptographic perspective this idea is undesirable because the prover also reveals π to the verifier, and this is information that the verifier may not have been able to compute on its own (because the verifier runs in polynomial time, and computing an isomorphism may require more time than that).

Amazingly, the protocol of [41] allows the prover to convince the verifier, but without revealing anything, not even π. Informally, the idea is to have the prover send a random copy G of G_0 to the verifier, and then let the verifier reply with a random bit b. The prover now replies with a permutation π' between G and G_b. Thus, when the graphs are isomorphic the verifier accepts, and it learns nothing other than this fact. If the graphs are not isomorphic, then the verifier rejects with probability $1/2$, but this can be reduced using repetition (assuming that the verifier follows the protocol).

A common application for zero-knowledge protocols is in *identification schemes*, due to Feige, Fiat, and Shamir [35]. For example, a user can choose isomorphic graphs $\langle G_0, G_1 \rangle$ together with an isomorphism π between them, and then use these graphs to register in various online services, such as online-banking, e-mail accounts, and so on. Now the user can log into these services by proving that the graphs are isomorphic. The advantages of this mechanism is that one identity can be used for various accounts, and only the user knows the password (we think of the isomorphism π as the password). Of course, there are many technical issues that need to be dealt with. For example, in addition to choosing $\langle G_0, G_1 \rangle$ and an isomorphism, the user must make sure that no efficient adversary who sees G_0 and G_1 can compute an isomorphism between them. Also, compared to standard passwords, π may be more difficult to remember. Yet, this application demonstrates the potential of zero-knowledge protocols.

1.2 Background

Zero-knowledge protocols were introduced in the 1980s by Goldwasser, Micali, and Rackoff [46], who also gave the first zero-knowledge proof, namely, the zero-knowledge proof for QUADRATIC-RESIDUOUSITY. Following this, Goldreich, Micali and Wigderson [41] showed that GRAPH-ISOMORPHISM has a zero-knowledge proof, which we described in the previous section. In both proofs, the messages exchanged between the prover and the verifier leak absolutely no information to the verifier but the truth of the assertion being proved. Such protocols are called *perfect* zero-knowledge.

Another line of research that started in the 1980s is zero-knowledge protocols for any **NP** language (as opposed to a particular language). In these protocols the prover can prove any **NP** statement by using a cryptographic primitive called *a bit commitment scheme*. The first zero-knowledge proof for **NP** is due to [41]. It uses the computationally hiding commitment of [67], and thus it is *computational* (as opposed to perfect) zero-knowledge. Informally, in a computational zero-knowledge protocol the amount of information leaked by the prover is negligible from the perspective of an efficient algorithm.[1]

We remark that, under various number theoretic assumptions, perfect zero-knowledge *arguments* for **NP** have been constructed in both the interactive [20, 21] and the non-interactive models [47] (informally, *arguments* [19] require that no *efficient prover* can make the verifier accept false statements, whereas the stronger notion of proofs [46] requires that no prover can make the verifier accept false statements). However, we are concerned with the *unconditional* study of perfect zero-knowledge *proofs*. Also, notice that perfect (and even statistical) zero-knowledge proofs for **NP** are unlikely to exist, as this would imply the collapse of the polynomial-time hierarchy [37, 3, 18].

Towards the end of 2000 Sahai and Vadhan [77] discovered that there are natural problems admitting *statistical* zero-knowledge proofs. Informally, in a statistical zero-knowledge protocol the amount of information leaked by the prover is negligible. We can now list the three notions of zero-knowledge:

- **Perfect Zero-Knowledge** - the amount of information that the prover leaks to the verifier is 0.
- **Statistical Zero-Knowledge** - the amount of information leaked to the verifier is negligible.
- **Computational Zero-Knowledge** - from the perspective of any probabilistic polynomial-time Turing machine, the amount of information leaked to the verifier is negligible.

This shows that the notions of statistical and perfect zero-knowledge are very close. Specifically, both require that the amount of information leaked be either 0 or small, regardless of the computational power of the observer. In contrast, a computational zero-knowledge protocol may leak a lot of information, but if a Turing machine only has polynomial time to inspect the messages exchanged between the prover and the verifier, then the amount of information that it can gain from them is negligible.

1.3 Motivation

The research in this thesis started with the observation that all the results from the statistical setting do not apply to the perfect setting. For example, Ong and Vadhan [73] recently showed a transformation that takes any statistical zero-knowledge proof, and turns it into a *constant-round* statistical zero-knowledge proof. That is, in any statistical zero-knowledge proof the number of messages exchanged between the prover and the verifier can be reduced to a constant. Unfortunately, when we apply this transformation to perfect zero-knowledge proofs, we do not get constant-round perfect zero-knowledge proofs. Rather, we get constant-round *statistical* zero-knowledge proofs. Again, this phenomenon occurs with all the general results from the statistical setting, including transformations from private-coins to public-coins [71, 42], from honest to malicious verifier [42], and from inefficient to efficient provers [72]. These transformations

[1] *Negligible* means that, for any polynomial p, the amount is less than $1/p(|x|)$ for all sufficiently large inputs x.

apply to statistical zero-knowledge proofs, and they even extend to the computational setting [87], but they do not apply to perfect zero-knowledge proofs.

Intuitively, the results from the statistical setting do not apply to the perfect setting because they use tools whose side effect is that they cause the prover to leak a small amount of information to the verifier. Such tools, like lower-bound sub-protocols [6, 45, 83] or circuits manipulation [77], enrich the study of statistical zero-knowledge and make it more flexible. Unfortunately, since perfect zero-knowledge proofs require that the prover leak absolutely no information, we cannot use these tools in the perfect setting. Consequently, many fundamental questions that have been solved in the statistical setting remain open in the perfect setting. We believe that addressing these questions has good motivation from the perspective of both complexity theory and cryptography:

Complexity Theory. One of the most important questions in computer science is whether the complexity classes **P** and **NP** coincide. This makes perfect zero-knowledge proofs interesting because all the known problems admitting non-trivial perfect zero-knowledge proofs, like GRAPH-ISOMORPHISM, are in **NP**, but not known to be **NP**-complete or in **P**. We remark that many of these problems are also *random-self reducible* [4], and such problems have been studied extensively in the context of both zero-knowledge [4, 84, 12, 79] and complexity theory (c.f., [1, 36]). This applies to GRAPH-ISOMORPHISM in particular (c.f., [41, 18, 85, 55]).

Cryptography. Perhaps the most attractive feature of perfect zero-knowledge protocols is that they provide perfect privacy to the prover. That is, unlike statistical zero-knowledge protocols, where the prover leaks a small amount of information to the verifier, in perfect zero knowledge proofs the prover leaks absolutely no information to the verifier. This makes them valuable to cryptography.

In addition, many problems that admit perfect zero-knowledge proofs, like QUADRATIC-RESIDUOUSITY and DISCRETE-LOGARITHM, play a central role in cryptography, and they are used in key agreement, encryption schemes, and digital signatures (c.f., [31, 33]). Finally, as we demonstrated earlier with GRAPH-ISOMORPHISM, there are perfect zero-knowledge protocols that yield identification schemes, such as the protocol of [35] for QUADRATIC-RESIDUOUSITY.

1.4 Our Results

We informally describe our main results.

1.4.1 Characterizing Non-interactive Instance-Dependent Commitment Schemes (NIC)

We started our research by considering all the known problems admitting perfect zero-knowledge proofs. We observed that these problems admit 3-round perfect zero-knowledge proofs, and then we proved that a problem admits such a proof if and only if it admits a simple combinatorial object, which we called *a* *non-interactive instance-dependent commitment-scheme (*NIC*).* The advantage of NIC is that they allow us to study all the known problems admitting perfect zero-knowledge proofs from a new direction. Indeed, we used NIC to strengthen and generalize previous results, as well as to prove new results about problems admitting perfect zero-knowledge proofs. These results, described in Chapter 3, are joint work with Bruce Kapron and Venkatesh Srinivasan [51].

1.4.2 Perfect Simulation and a Complete Problem for NIPZK

Following our characterization of the *known* problems admitting perfect zero-knowledge proofs, we sought to provide a general framework (through complete problems) that would capture *all the problems* admitting perfect zero-knowledge proofs. We present the first complete problem for the class of problems possessing *non-interactive* perfect zero-knowledge proofs (**NIPZK**), and the first hard problem for the class of problems possessing public-coin perfect-zero-knowledge proofs. To obtain these problems we use a new *error shifting technique*, which has other useful applications. These results, published in [59], are described in Chapter 4.

1.4.3 The Round Complexity of Perfect Zero-Knowledge Proofs

Using the tools we developed, we can now address the question whether perfect zero-knowledge proofs have a constant number of rounds. This question is of both theoretical and practical importance. We give the first evidence that *perfectly* (as opposed to statistically) hiding instance-dependent commitment schemes can be constructed from any problem that has a perfect zero-knowledge proof, and show that obtaining such a scheme that is also *constant-round* is not only sufficient, but also necessary to collapse the number of rounds in perfect zero-knowledge proofs. We construct a non-interactive, perfectly hiding scheme whose binding property holds on all but an exponentially small fraction of the inputs, and define a preamble to address the binding property. An interesting consequence is the use of the circuits from our **NIPZK**-complete problem in the commitment scheme of Naor [67], which leads to a new instant-dependent commitment scheme for **NIPZK** problems admitting a small soundness error. These results, first published in [60], are described in Chapter 5.

Chapter 2

Definitions

In this section we define protocols, proofs, indistinguishability, and zero-knowledge. To make these definitions more intuitive, we start with a simple example of a zero-knowledge proof for the problem $\mathrm{SD}^{0,1}$, due to Sahai and Vadhan [77]. The formal definitions are given in Sections 2.2- 2.5.

2.1 A Simple Zero-Knowledge Protocol

In this section we describe a zero-knowledge proof that uses exactly the same idea as the proof of [41] for GRAPH-ISOMORPHISM. Informally, the prover and the verifier are given a pair of circuits $\langle X_0, X_1 \rangle$, and the goal of the prover is to convince the verifier that the circuits represent the same distribution, but without revealing anything to the verifier except for the truth of this assertion. That is, the proof will be perfect zero-knowledge.

2.1.1 Preliminaries

We start with notation. Let $X : \{0,1\}^m \to \{0,1\}^n$ be a function mapping binary strings of length m to binary strings of length n. In addition to being a function, we can think of X as a distribution. For example, if there are k inputs to X that make it output the string y, then the probability that X outputs y, denoted $\Pr[X = y]$, is $k/2^m$. That is, we make the convention that the input to X is uniformly distributed.

The common input to the protocol is a pair of functions $\langle X_0, X_1 \rangle$ represented as circuits. When we say that X_0 and X_1 are identically distributed, we mean that the distributions represented by the circuits are identically distributed. That is, $\Pr[X_0 = y] = \Pr[X_1 = y]$ for any y. When we say that X_0 and X_1 are disjoint, we mean that the ranges of X_0 and X_1 are disjoint. That is, $X_0(r_0) \neq X_1(r_1)$ for any r_0, r_1.

2.1.2 Motivating the Protocol

Recall that in our example the prover and the verifier are given two circuits X_0 and X_1, and the prover wants to prove to the verifier that X_0 and X_1 are identically distributed. To simplify the presentation, we only consider two cases: either X_0 and X_1 are identically distributed, or they are disjoint (it is not known how to entirely remove this restriction, but it can be relaxed [77]). If the assertion that X_0 and X_1 are identically distributed is true, then the verifier should accept, but without learning anything but the truth of this assertion. If X_0 and X_1 are disjoint, then the verifier should reject, regardless of how a malicious prover may behave, and we do not care about what the verifier learns from the malicious prover.

The protocol is as follows. If X_0 and X_1 represent the same distribution, there is at least one pair $\langle r_0, r_1\rangle$ such that $X_0(r_0) = X_1(r_1)$, and the prover uniformly chooses r_0, computes $m = X_0(r_0)$, and then uniformly chooses $r_1 \in X_1^{-1}(m)$. The prover sends m to the verifier. The verifier replies with a random bit b, and the prover sets $r = r_b$, and sends r to the verifier. The verifier accepts if $X_b(r) = m$, and rejects otherwise. This protocol if formally described in Figure 2.1.

A Zero-Knowledge Protocol for $\mathrm{SD}^{0,1}$

Common input: a pair of circuits $\langle X_0, X_1\rangle$. Let m be the number of input bits to X_0.

1. The prover uniformly chooses $r \in \{0,1\}^m$, computes $m = X_0(r)$, and sends m to the verifier.
2. The verifier uniformly chooses $b \in \{0,1\}$, and sends b to the prover.
3. The prover uniformly chooses an element r from the set $X_b^{-1}(m) \overset{\text{def}}{=} \{r | X_b(r) = m\}$, and sends r to the verifier.
4. The verifier accepts if $X_b(r) = m$, and rejects otherwise.

Figure 2.1: A simple zero-knowledge proof

2.1.3 Analysis of the Protocol

We analyze Protocol 2.1, assuming the verifier is honest. If X_0 and X_1 are identically distributed, then for any string m it holds that $|X_0^{-1}(m)| = |X_1^{-1}(m)|$, which implies that the verifier always accepts. In the case that X_0 and X_1 are disjoint, given m there is $i \in \{0,1\}$ such that $X_i(r) \neq m$ for any r. Since b is chosen uniformly, if X_0 and X_1 are disjoint, then the verifier rejects with probability $1/2$. We will later see that these properties are called *completeness* and *soundness*, respectively, and a protocol admitting them is called a *proof*.

To show that this proof is perfect zero-knowledge, we need to show that the verifier learns absolutely nothing from the prover. We start with the observation that the verifier has three sources of information: its randomness b, the common input $\langle X_0, X_1\rangle$, and the messages exchanged $\langle m, b, r\rangle$. Since the verifier always knows the common input and its own random input, the zero-knowledge property is established by showing that the verifier can compute the messages $\langle m, b, r\rangle$ on its own, without interacting with the prover. This procedure is called *the simulator*. Thus, we want to show a simulator that computes the *transcript* $\langle m, b, r\rangle$ given b and X_0, X_1.

Notice that there could be different transcripts $\langle m, b, r\rangle$ in the interaction, each appearing with a certain probability. We want the simulator to output these transcripts with the same probability. This can be done as follows: the simulator uniformly chooses r', computes $m' = X_{b'}(r')$ using the bit b' of the verifier, and outputs $\langle m', b', r'\rangle$. We chose different names for these messages because we want to compare two probability spaces: the one containing transcripts $\langle m, b, r\rangle$ from the interaction, and the one containing outputs $\langle m', b', r'\rangle$ of the simulator.

It remains to show that the output $\langle m', b', r'\rangle$ of the simulator is identically distributed to the transcripts $\langle m, b, r\rangle$ exchanged between the prover and the verifier. This is rather straight forward, but we provide the analysis for completeness. Our first observation is that, since X_0 and X_1 are identically distributed, the distribution on the messages $\langle m, b, r\rangle$ remains the same if instead of computing $m = X_0(r)$, the prover uniformly chooses $c \in \{0, 1\}$ and computes $m = X_c(r)$. Now, since the modified prover and the simulator compute the first message in the same way, m and m' are identically distributed. We continue to the message b'. Since b is uniformly distributed and independent of m, we need to show that b' is also uniformly distributed and independent of m'. This follows from the fact that X_0 and X_1 represent the same distribution (given m', the value of b' can be 0 with probability $1/2$, and 1 with probability $1/2$). Hence, conditioned on $m = m'$, the bits b and b' are identically distributed. We continue to the message r'. Since r' is uniformly chosen, given m' and b' the message r' is uniformly distributed in $X_{b'}^{-1}(m')$. Since the prover also chooses r uniformly from $X_b^{-1}(m)$, the transcripts $\langle m, b, r\rangle$ are identically distributed to the output $\langle m', b', r'\rangle$ of the simulator. Thus, Protocol 2.1 is a perfect zero-knowledge proof for $\mathrm{SD}^{0,1}$.

2.2 Conventions

In this section we give common conventions to be used throughout this thesis.

Problems. A *string* x is a finite sequence of symbols from the alphabet $\{0, 1\}$, and $|x|$ denotes the length of x. As usual, ϵ denotes the empty string, and x^k denotes k concatenations of x. The set $\{0, 1\}^n$ contains all the strings of length n. When we refer to sets we mean countable sets of strings. Some of our results refer to *languages*, which are sets, and some refer to *promise-problems* (or *problems* for short) [34]. A *problem* Π is a pair $\langle \Pi_Y, \Pi_N\rangle$ of disjoint sets, and the complement of Π is defined as $\overline{\Pi} \overset{\text{def}}{=} \langle \Pi_N, \Pi_Y\rangle$. The set Π_Y contains *YES instances*, and the set Π_N contains *NO instances*. A language L can be defined as $\langle L, \overline{L}\rangle$.

Probability. For background on probability theory we refer the reader to [66, 25]. We only consider discrete probability spaces. As usual, the *uniform distribution* on the set $\{0, 1\}^n$ is the probability space that assigns the probability $1/2^n$ to each string in this set. We already mentioned that circuits will be treated as distributions, assuming the uniform distribution on their input. For example, if X is a circuit that takes inputs of length n, and there are k inputs that make X output the string y, then $\Pr[X = y] = k/2^n$. The distribution represented by X is also denoted X.

Turing machines and circuits. A Turing machine M *runs in polynomial-time* if there is a polynomial p such that for any input x the computation of M on x, denoted $M(x)$, takes at most $p(|x|)$ steps. When we write $M(x) = 1$ we mean that M *accepts* x, and when we write $M(x) = 0$ we mean that M *rejects* x. A Turing machine *decides* a problem if $M(x) = 1$ when x is a YES instance of the problem, and $M(x) = 0$ when x is a NO instance.

A Turing machine M is *probabilistic* if it has a special *random tape* in which each bit is uniformly chosen, and this tape is refreshed for each execution. We denote by $\Pr[M(x) = 1]$ the probability that M outputs 1 given input x, where the probability is over the choices of the random tape. The class **BPP** contains all the languages L that can be decided by a a probabilistic, polynomial-time Turing machine M. That is, $\Pr[M(x) = 1] \geq 2/3$ when $x \in L$, and $\Pr[M(x) = 1] \leq 1/3$ when $x \notin L$.

A sequence of circuits $\{C_n\}_{n \in \mathbb{N}}$ is a *non-uniform family of polynomial-size circuits* if there is a polynomial p such that $|C_n| \leq p(n)$ for all n, where $|C_n|$ is some binary encoding of circuits. It is well known that for any sequence $\{M_n\}_{n \in \mathbb{N}}$ of Turing machines, if there is a polynomial p such that for all n it holds that M_n runs in time at most $p(n)$ on inputs of length n, then the sequence can be encoded by a family of polynomial-size circuits (c.f., [75, 39]).

Complexity. The class of languages decided by polynomial-time, deterministic Turing machines is denoted **P**. The famous class **NP** contains all languages decided by polynomial-time, non-deterministic Turing machines. Alternatively, any **NP** language L can be associated with a relation R, a polynomial p, and a deterministic, polynomial-time Turing machine M. The relation R contains pairs $\langle x, w \rangle$ satisfying $|w| \leq p(|x|)$, and w is called *a witness* for x. The machine M takes both x and w as input, and it accepts if and only if $x \in L$.

Let $\mathcal{C}$ be a class of problems. A problem $\langle \Pi_Y, \Pi_N \rangle$ is *hard* for $\mathcal{C}$ (or simply $\mathcal{C}$-hard) if for any problem $\langle \Pi'_Y, \Pi'_N \rangle \in \mathcal{C}$ there is a deterministic, polynomial-time Turing machine f such that if $x \in \Pi_Y$, then $f(x) \in \Pi'_Y$, and if $x \in \Pi_N$, then $f(x) \in \Pi'_N$. Such f is called a *Karp reduction*. A problem is *complete* for $\mathcal{C}$ (or simply $\mathcal{C}$-complete) if it is contained in $\mathcal{C}$ and hard for $\mathcal{C}$. For example, HAMILTONIAN-CIRUIT is **NP**-complete because it is in **NP**, and any language in **NP** Karp reduces it [38].

The definition of classes in terms of languages naturally extends to problems, except that when we talk about problems we only consider YES and NO instances, whereas in languages we consider all the strings (that is, L and $\overline{L}$). For example, $\langle \Pi_Y, \Pi_N \rangle$ is an **NP**-problem if there is a non-deterministic, polynomial-time Turing machine that accepts $x \in \Pi_Y$ and always rejects $x \in \Pi_N$. That is, we do not care about instances not in $\Pi_Y \cup \Pi_N$.

2.3 Interactive Protocols

We define the notion of an interactive protocol, originally due to Goldwasser, Micali, and Rackoff [46]. Instead of formulating interaction in terms of interactive Turing machines, we adopt the more general formulation using functions, noted by Goldwasser and Sipser [45]. That is, an interactive protocol is simply a pair of functions sending messages to each other until one of the functions terminate. Formally,

Definition 2.3.1 (Interactive Protocols) *An interactive protocol is a pair $\langle P, V\rangle$ of functions. The* interaction *between P and V on common input x is the following random process.*

1. *Let r_P and r_V be random inputs to P and V, respectively.*
2. *repeat the following for $i = 1, 2, \ldots$*
   - *(a) If i is odd, let $m_i = P(x, m_1, \ldots, m_{i-1}; r_P)$.*
   - *(b) If i is even, let $m_i = V(x, m_1, \ldots, m_{i-1}; r_V)$.*
   - *(c) If $m_i \in \{\texttt{accept}, \texttt{reject}, \texttt{fail}\}$, then exit loop.*

We say that V accepts *x if $m_i = \texttt{accept}$ for an even i. Interactions yield* transcripts *$\langle x, m_1, \ldots, m_p; r_V\rangle$, and we call the strings m_i* messages*. The probability space containing all the transcripts is called* the view of *V* on *x, and is denoted $\langle P, V\rangle(x)$. The* round complexity *of $\langle P, V\rangle$ is a function p such that for any x, and any interaction on input x, the number of messages exchanged is at most $p(|x|)$. We say that $\langle P, V\rangle$ is* constant round *if p is a constant.*

We say that $\langle P, V\rangle$ is public coin *if V always sends independent portions of r_V, and its last message is a deterministic function of the messages exchanged.*

Now we can define interactive proofs [46]. Informally, a problem has an interactive proof if it has an interactive protocol in which a *common input* x is given to the prover and the verifier, the verifier runs in time polynomial in $|x|$, and it accepts if x is a `YES` instance, and rejects if x is a `NO` instance (the probabilities to accept and reject are far by at least the reciprocal of a polynomial). Formally,

Definition 2.3.2 (Interactive proofs and arguments) *Let $\Pi = \langle \Pi_Y, \Pi_N\rangle$ be a problem, and let $\langle P, V\rangle$ be an interactive protocol. We say that $\langle P, V\rangle$ is an* interactive proof *for Π if there is a, and $c(n), s(n) : \mathbb{N} \to [0, 1]$ such that $1 - c(n) > s(n) + 1/n^a$ for any n, and the following conditions hold.*

- *Efficiency: V is a probabilistic Turing machine whose running time over the entire interaction is polynomial in $|x|$ (this implies that the number of messages exchanged is polynomial in $|x|$).*
- *Completeness: if $x \in \Pi_Y$, then V accepts in $\langle P, V\rangle(x)$ with probability at least $1 - c(|x|)$. The probability is over r_P and r_V (the randomness for P and V, respectively).*

- *Soundness: if $x \in \Pi_N$, then for any function P^* it holds that V accepts in $\langle P^*, V\rangle(x)$ with probability at most $s(|x|)$. The probability is over the randomness r_V for V.*

If the soundness condition holds with respect to non-uniform polynomial-size circuits P^*, then we say that $\langle P, V\rangle$ is an interactive argument *for* Π.

The function c is the completeness error*, and the function s is the* soundness error. *We say that $\langle P, V\rangle$ has* perfect completeness *(respectively,* perfect soundness*) if $c \equiv 0$ (respectively, $s \equiv 0$).*

We denote by **IP** the class of problems admitting interactive-proofs [46], and by **AM** the class of problems admitting *public-coin, constant-round* interactive-proofs [6, 56].

Definition 2.3.3 (Efficient prover) *Let $\langle P, V\rangle$ be an interactive proof or argument for an* ***NP*** *problem $\Pi = \langle \Pi_Y, \Pi_N\rangle$. We say that P is an* efficient prover *if given an arbitrary* ***NP*** *witness w for $x \in \Pi_Y$ the prover runs in time polynomial in $|x|$.*

2.4 Indistinguishability

The notion of zero-knowledge is based on indistinguishability between two ensembles: the output of the simulator, and interactions between the prover and the verifier.

A *probability ensemble* is a sequence $\{Y_x\}_{x\in I}$ of random variables, where I is countable set of strings. Indistinguishability is defined in terms of distance between ensembles. A function $f(n)$ is *negligible* if all of its outputs are small when the inputs are large enough. Formally, f is negligible on I if for any polynomial p there is N such that for all $x \in I$ of length at least N it holds that $f(|x|) < 1/p(|x|)$. When I is clear from the context we simply say that $f(n)$ is *negligible*.

We will consider three notions of indistinguishability: computational, statistical, and perfect. Computational indistinguishability is defined in terms of advantage of a *distinguisher* D. Given two distributions Y_x and Z_x, and a circuit D whose output is 0 or 1, the *advantage* of D to distinguish Y_x from Z_x is defined as

$$\mathtt{adv}(D, Y_x, Z_x) \stackrel{\text{def}}{=} |\Pr[D(Y_x) = 1] - \Pr[D(Z_x) = 1]|,$$

where $\Pr[D(X) = 1]$ is the probability that D outputs 1 given an element chosen according to the distribution X. Notice that if D is probabilistic, then according to our convention this probability is also over the uniform distribution on the randomness of D.

Statistical indistinguishability makes no reference to circuits. Given two discrete distributions X and Y, the *statistical distance* between them is

$$\Delta(X, Y) \stackrel{\text{def}}{=} 1/2 \cdot \sum_{\alpha} |\Pr[X = \alpha] - \Pr[Y = \alpha]| = \max_{S}(|\Pr[X \in S] - \Pr[Y \in S]|).$$

The formal definition of the three notions of indistinguishability follows.

Definition 2.4.1 (Indistinguishability) *Two probability ensembles $\{Y_x\}_{x\in I}$ and $\{Z_x\}_{x\in I}$ are* computationally indistinguishable *if $\mathtt{adv}(D, Y_x, Z_x)$ is negligible on I for all non-uniform polynomial-size circuits D. They are* statistically identical *(respectively,* statistically indistinguishable*) if $\Delta(Y_x, Z_x)$ is identically 0 (respectively, negligible) on I.*

Variants of the problem STATISTICAL-DISTANCE (SD) will play a central role in this thesis. This problem originated from the study of **SZK** due to [77]. Its instances are pairs of circuits. As we remarked in Section 2.1.1, we can treat circuits as distributions (using the convention that the inputs are uniformly chosen) or as boolean functions. Instances of SD are statistically close as `YES` instances, and statistically far as `NO` instances. Formally,

Definition 2.4.2 *The problem* $\mathrm{SD}^{\alpha,\beta}$ [77] is the pair $\langle \mathrm{SD}^{\alpha}_{\mathrm{Y}}, \mathrm{SD}^{\beta}_{\mathrm{N}}\rangle$, where

$$\mathrm{SD}^{\alpha}_{\mathrm{Y}} = \{\langle X_0, X_1\rangle \mid \Delta(X_0, X_1) \leq \alpha\}, \text{ and}$$
$$\mathrm{SD}^{\beta}_{\mathrm{N}} = \{\langle X_0, X_1\rangle \mid \Delta(X_0, X_1) \geq \beta\}, \text{ and}$$

X_0 and X_1 are circuits (treated as distributions).

We remark that $\mathrm{SD} \overset{\mathrm{def}}{=} \mathrm{SD}^{1/3,2/3}$ is **SZK**-complete, and since **SZK** is closed under complement [71, 77], $\overline{\mathrm{SD}}$ is also **SZK**-complete. In this thesis we are only interested in the **NP** problem $\mathrm{SD}^{0,\beta}$ where $\beta = 1$ or $\beta \geq 1/2$ (or some other non-negligible constant).

2.5 Zero-Knowledge

Informally, an interactive proof (or an interactive argument) is *zero-knowledge* if there is a simulator such that the view of the verifier and the output of the simulator are indistinguishable. To simplify the presentation we chose a definition where the simulator is not allowed to fail. The relaxed definition (where the simulator is allowed to fail with probability at most $1/2$) requires that conditioned on not failing, the output of the simulator be indistinguishable from the view of the verifier. Most of our results hold with respect to this relaxed definition, and in fact our result from Section 4.4 shows that for the case of honest verifiers the two notions are equivalent. We use S^V to denote a Turing machine S with oracle access to Turing machine V.

Definition 2.5.1 (Zero-knowledge protocols) *A protocol $\langle P, V\rangle$ for a problem $\Pi = \langle \Pi_Y, \Pi_N\rangle$ is* perfect *(respectively,* statistical, computational*)* zero-knowledge *if there is a probabilistic, polynomial-time Turing machine S, called* the simulator*, such that for any probabilistic, polynomial-time Turing machine V^*,*

$$\{\langle P, V^*\rangle(x)\}_{x\in\Pi_Y} \quad \text{and} \quad \{S^{V^*}(x)\}_{x\in\Pi_Y}$$

are statistically-identical (respectively, statistically indistinguishable, computationally indistinguishable.) The class of problems admitting perfect (respectively, statistical, computational) zero-knowledge protocols *is denoted* ***PZK*** *(respectively,* ***SZK***, ***CZK****.) When the above ensembles are indistinguishable for $V^* = V$ we say that $\langle P, V\rangle$ is* honest-verifier, perfect *(respectively,* statistical, computational*)* zero-knowledge*, and we denote the respective classes by* ***HVPZK***,***HVSZK****, and* ***HVCZK****.*

We remark that the above definition allows S only oracle access to the V^*. That is, for any input q the simulator can evaluate $V^*(q)$ in one step, and it does not have access to the Turing machine describing V^*. This notion is known as *black-box* simulation (as opposed to *non-black-box* simulation, where S can also read the Turing machine describing V^*) [39, 7]. Another notion of zero-knowledge considers V^* with an *auxiliary input*, which is useful in cases where the zero-knowledge protocol is executed within another protocol, and the verifier have more initial information than only the common input. Clearly, in such case the simulator is also allowed to use the auxiliary input.

We also remark that the literature has observed a technical issue with Definition 2.5.1. Specifically, the simulator runs in polynomial time (for a *fixed* polynomial), but it needs to choose random tapes for verifiers V^*, each of whom runs in time described by an arbitrary polynomial. In other words, the simulator may not have enough time to write down the random string. Although this would not make a difference in this thesis, for the sake of formality we adopt the approach that swaps the quantifiers in Definition 2.5.1. That is, we require that for any verifier V^* there is a simulator S that simulates the interaction between P and V^*.

Chapter 3

Non-interactive Instance-Dependent Commitment Schemes

When Goldwasser, Micali, and Rackoff [46] introduced the concept of zero-knowledge, they also gave the first example of a language that unconditionally admits a zero-knowledge proof. Namely, the perfect zero-knowledge (**PZK**) proof for QUADRATIC-RESIDUOUSITY. Subsequently, GRAPH-ISOMORPHISM was shown to have a **PZK** proof [41], and this was later generalized to all random self-reducible (RSR) languages [4], and monotone boolean formulae over RSR languages [79].

Although each of these problems has its own **PZK** proof, the proofs themselves have the same structure. That is, three messages are exchanged, and the message of the verifier (i.e., the second message) is a randomly chosen bit. We call such protocols *V-bit protocols*.[1] What is interesting about these protocols is that all the known problems admitting **PZK** proofs have the structure of a V-bit protocol. For example, the problem $\mathrm{SD}^{0,1}$ mentioned in Chapter 2.1, and the language DISCRETE-LOGARITHM [4]. Thus, a natural question that follows is whether this fact can be useful for studying the entire class of known problems admitting **PZK** proofs (instead of studying each problem individually).

In this chapter we show that indeed, all the known problems admitting **PZK** proofs can be studied through *non-interactive, instance-dependent commitment schemes* (NIC). We achieve this result by using the technique of Damgård [29] to construct NIC from V-bit zero-knowledge protocols, and by using the idea of Itoh, Ohta and Shizuya [50] to obtain V-bit zero-knowledge protocols from NIC. This characterization of V-bit zero-knowledge protocols as NIC applies also to the statistical and the computational settings. Thus, although we are interested in **PZK**, our discussion will be general, and will include statistical and computational zero-knowledge (**SZK** and **CZK**, respectively) proofs.

Next, we use the technique of De Santis, Di Crescenzo, Persiano, and Yung [79], to show that NIC can be combined in a monotone boolean formula fashion (i.e., with AND and OR connectives). Combining

[1]The notions of V-bit protocols and Σ-protocols [26] are similar in that both refer to 3-round protocols, but different in that V-bit protocols make no reference to zero-knowledge or special soundness. However, it will later follow from our results that a problem admits a V-bit *zero-knowledge* proof if and only if it admits a Σ-protocol.

this with our characterization result, we obtain what we call *the* NIC *framework*. This framework allows us to study all the known languages admitting **PZK** proofs. Since it also applies to the statistical and the computational settings, we strengthen and unify many previous results.

3.0.1 Motivation

Much of the study of zero-knowledge protocols relies on the existence of bit commitment schemes (equivalently, one-way functions [49, 67]). Intuitively, commitment schemes allow a *sender* to commit to a bit b such that the *receiver* cannot learn b from the commitment (this property is called *hiding*), and at the same time the sender cannot change the commitment to another value (this property is called *binding*).

Itoh, Ohta and Shizuya [50] suggested an alternative approach to commitment schemes. They observed that in the protocol of [41] for **NP** the scheme should be hiding when the input is a `YES` instance and binding when it is a `NO` instance, but the hiding and the binding properties do not need to hold simultaneously. Using this observation, they constructed such a scheme for *specific* languages such as GRAPH-ISOMORPHISM and QUADRATIC-RESIDUOUSITY. By using the scheme (instead of a bit commitment scheme) in the protocol of Blum [15] for **NP**, they obtained perfect zero-knowledge (**PZK**) proofs with efficient provers for these languages (different proofs for these languages were known before [46, 41, 84]).

The schemes of [50] are different from commitment schemes because they also take an instance x of a problem as an input, and the hiding and the binding properties depend on whether x is a `YES` or a `NO` instance. For example, the problem $\mathrm{SD}^{0,1}$, discussed in Section 2.1 and defined in Section 2.4, has such a scheme [63]. Namely, given a pair of circuits $\langle X_0, X_1\rangle$ as an instance, a commitment to a bit b can be computed by choosing a random string r and outputting $y = X_b(r)$. Thus, if X_0 and X_1 represent the same distribution, then y perfectly hides b, and if they are disjoint, then y cannot be a commitment to both 0 and 1, and hence y binds to b. We call such a scheme *a non-interactive instance-dependent commitment scheme* (NIC). The term *non-interactive* means that only one message is sent by the sender (i.e., the receiver does not send anything), and the term *instance-dependent* means that the hiding and the binding properties depend on the instance x. The approach of instance-dependent commitment schemes turned out to be very successful in the study of zero-knowledge protocols ([50, 63, 62, 70, 69, 51, 72, 73, 23]).

3.0.2 Main Results

Using the technique of [29] we show that if a problem has a V-bit zero-knowledge protocol, then it has a non-interactive instance-dependent commitment scheme (NIC). Using the technique of [50] we then prove that the opposite is also true. Our result applies not only to the perfect setting, but also to the statistical and the computational settings. This shows a tight relation between two natural but restrictive types of commitment schemes and zero-knowledge protocols.

Theorem 3.0.2 *A promise-problem* Π *has a* V*-bit* **HVPZK** *(respectively,* **SZK***) proof if and only if* Π *has a perfectly (respectively, statistically) hiding* NIC*. Similarly,* Π *has a* V*-bit* **CZK** *proof if and only if* $\Pi \in$ ***NP***

and Π *has a computationally hiding* NIC.

In addition to our theorem, we prove two lemmas. The first lemma proves that any random self-reducible [4] (RSR) problem has a perfectly hiding NIC. This folklore lemma follows from [84, 79], but here we provide the proof for completeness. The second lemma uses the technique of [79] to show that NIC can be combined in a monotone boolean formula fashion (i.e., with `AND` and `OR` connectives). Together with our theorem, these lemmas yields a useful framework that enables us to achieve unconditional results about various zero-knowledge protocols.

3.0.3 Organization

Our main theorem is proved in Section 3.2, and our proof for random-self reducible languages is given in Section 3.3. The closure result is in Section 3.4, and the consequences of our framework are summarized in Section 3.5. We start with the definition of NIC.

3.1 Non-interactive, Instance-Dependent Commitment-Schemes (NIC)

In this section we define the notion of a *non-interactive instance-dependent commitment schemes* (NIC).

To motivate the notion of a NIC we start with the familiar notion of a *non-interactive bit commitment scheme*. Intuitively, such a scheme allows a sender to commit to a bit b such that the receiver cannot learn the value of b, yet the sender cannot change b. More precisely, the scheme is an efficient function $f(b;r)$, and to commit to b the sender chooses randomness r, computes $y = f(b;r)$, and sends y to the receiver. This is the *commit phase*. In the *reveal phase* the sender sends b and r to the receiver, who computes $f(b;r)$, thus confirming that y is indeed a commitment to b. The receiver does not send anything (hence the term *non-interactive*). The scheme is *hiding* if b cannot be determined from y, and *binding* if y binds the sender to b (that is, $f(0;r) \neq f(1;r')$ for any $r \neq r'$).[2]

Intuitively, a NIC for a problem Π is a non-interactive commitment scheme where the hiding and the binding properties depend on instances of Π, and may not hold simultaneously. That is, instead of $f(b;r)$ we consider $f(x,b;r)$, and the hiding and binding properties depend on whether x is a `YES` or a `NO` instance of Π. The following definition is identical to the *positively opaque and negatively transparent* scheme of [50], and as was observed in [62], we can generalize it to the statistical and the computational settings.

Definition 3.1.1 (NIC) *Let* $\Pi = \langle \Pi_{\mathrm{Y}}, \Pi_{\mathrm{N}} \rangle$ *be a promise-problem, and let* $f(x,b;r)$ *be a probabilistic Turing machine running in time polynomial in* $|x|$*. The inputs to* f *are a string* x *(denoting an instance of* Π*), a bit b, and a string r (denoting the randomness of* f*).*

We say that f *is* **binding** *on* Π_{N} *if for any* $x \in \Pi_{\mathrm{N}}$*, and for any* r *and* r' *it holds that* $f(x,0;r) \neq f(x,1;r')$*. We say that* f *is* perfectly *(respectively,* statistically, computationally*)* **hiding** *on* Π_{Y} *if the*

[2]The notion of *interactive commitment schemes* is similar, except that the sender and the receiver can interact. Both notions are useful in the study of zero-knowledge protocols, but non-interactive schemes are more convenient to work with.

ensembles $\{f(x,0)\}_{x\in\Pi_{\mathrm{Y}}}$ and $\{f(x,1)\}_{x\in\Pi_{\mathrm{Y}}}$ are statistically identical (respectively, statistically indistinguishable, computationally indistinguishable), where $f(x,b)$ is a random variable obtained by uniformly choosing r, and outputting $f(x,b;r)$.

We say that f is a perfectly *(respectively,* statistically, computationally*)* hiding NIC *for* Π *if f is binding on Π_{N}, and perfectly (respectively, statistically, computationally) hiding on Π_{Y}.*

Perfectly and statistically hiding NIC are different from computationally hiding NIC. Firstly, in a perfectly or a statistically hiding NIC the hiding and the binding properties cannot hold at the same time, whereas in a computationally hiding NIC they may [50, 39]. Secondly, if Π has a perfectly or a statistically hiding NIC f, then as a class of problems **NP** contains Π. This is so because if $x \in \Pi_{\mathrm{Y}}$, then there is a pair $\langle r, r'\rangle$ such that $f(x,0;r) = f(x,1;r)$, and if $x \in \Pi_{\mathrm{N}}$, then no such pair exists. However, Π may not be in **NP** if f is computationally hiding. Finally, as was observed by [50], if a problem has a statistically hiding NIC, then it cannot be **NP**-complete, unless the polynomial hierarchy collapses [37, 3, 18]. We give an example of a NIC.

Example 3.1.2 *A* NIC *for the language* GRAPH-ISOMORPHISM *[12, 50]. Let $f(x,b;r)$ be a function that given a pair of graphs $x = \langle G_0, G_1\rangle$ on n vertices uses r to define a random permutation π over $\{1,\ldots,n\}$, and outputs $y = \pi(G_b)$. If the graphs are isomorphic, then y is isomorphic to both G_0 and G_1, and b cannot be determined from y. Conversely, if the graphs are not isomorphic, then y cannot be isomorphic to both G_0 and G_1. Thus, f is a perfectly hiding* NIC *for* GRAPH-ISOMORPHISM.

Another example is the statistically hiding NIC of [63] for $\mathrm{SD}^{1/2,1}$. Recall that by Definition 2.4.2, instances of $\mathrm{SD}^{1/2,1}$ are pairs of circuits $\langle X_0, X_1\rangle$ treated as distributions (under the convention that the input to the circuit is uniformly distributed). The statistical distance between X_0 and X_1 is $1/2$ for YES instances, and 1 for NO instances. Notice that statistical distance of 1 means that $X_0(r) \neq X_1(r')$ for any r and r'. Also, by taking many samples from each circuit, we obtain a new pair of circuits such that if X_0 and X_1 are disjoint, then so is the new pair, and if the the statistical distance between X_0 and X_1 is $1/2$, then the statistical distance between the circuits in the new pair is $1/2^n$, where $n = |X_0|$ (this, and another polarization technique can be found in [77]). Hence, $\mathrm{SD}^{1/2,1}$ defines a statistically hiding NIC: to commit to b we uniformly choose r and output $X_b(r)$.

In fact, the notion of a NIC is very close to the problem SD. For example, if f is a perfectly hiding NIC, then $\langle f(x,0), f(x,1)\rangle$ is a pair of circuits with statistical distance 0 when x is a YES instance, and statistical distance 1 when x is a NO instance. Thus, another way to look at our main result is that $\mathrm{SD}^{0,1}$ is complete for the class of problems admitting perfectly hiding NIC (equivalently, the class of problems admitting V-bit perfect zero-knowledge proofs). However, notice that the random input for the NIC for GRAPH-ISOMORPHISM is a permutation, and there $n!$ such inputs. Thus, unless $n!$ is a power of 2, this randomness cannot be represented by a bit string, which means that GRAPH-ISOMORPHISM is not known to be reducible to $\mathrm{SD}^{0,1}$.

3.2 Characterizing V-bit Zero-Knowledge Protocols

In this section we introduce the notion of V-bit protocols and prove Theorem 3.0.2. We only consider proofs, but our result also applies to arguments, in which case it yields NIC where the binding property holds with respect to computationally bounded senders.

Examples of V-bit protocols include the protocol of [77] for $\mathrm{SD}^{0,1}$, discussed in Section 2.1, the protocol of [41] for GRAPH-ISOMORPHISM, and the protocols of [15, 41] for **NP**. These protocols are public-coin, they have perfect completeness, and they admit the following structure: the prover sends the first message m_1, the verifier sends back a random bit b, the prover replies with a message m_2, and the verifier accepts or rejects. Since V sends only one bit, we call these protocols V-*bit protocols*. Formally,

Definition 3.2.1 (V-bit protocol) *Let $\langle P, V\rangle$ be a proof or an argument for a problem $\Pi = \langle \Pi_{\mathrm{Y}}, \Pi_{\mathrm{N}}\rangle$. We say that $\langle P, V\rangle$ is* V-bit *if for any $x \in \Pi_Y$ the interaction between P and V is as follows: P sends m_1 to V, and V replies with a uniformly chosen bit b. P replies by sending m_2 to V, and V accepts or rejects x based on $\langle x, m_1, b, m_2\rangle$. If $x \in \Pi_Y$, then V always accepts.*

We do not know if any V-bit zero-knowledge protocol is also a Σ-protocol. However, from our characterization result it will follow that a problem admits a V-bit zero-knowledge proof if and only if it admits a Σ-protocol. This will be discussed in more detail in Section 3.5.1.

3.2.1 From NIC to V-bit Zero-Knowledge Protocols

We show that if a problem has a NIC, then it has a V-bit zero-knowledge protocol. The proof is standard, and follows easily by plugging the NIC into the zero-knowledge protocols for **NP** [15, 41], as in [50].

Lemma 3.2.2 *If a problem Π has a perfectly (respectively, statistically) hiding* NIC*, then Π has a public-coin* **HVPZK** *(respectively,* **SZK***) V-bit proof with an efficient prover. If $\Pi \in$* **NP***, and Π has a computationally hiding* NIC*, then Π has a public-coin* **CZK** *V-bit proof with an efficient prover.*

Proof:(sketch) Recall that if a problem has a perfectly or a statistically hiding NIC, then it is contained in **NP**. Thus, we can use the zero-knowledge protocol of [15] for the **NP**-complete problem HAMILTONIAN-CIRUIT (HC). Specifically, given input $x \in \Pi_Y \cup \Pi_N$, the prover and the verifier initially reduce x to an instance G of HC, and then execute the protocol of [15] using the NIC f for Π as a bit commitment scheme. This protocol can be informally described as follows:

- The prover picks a random permutation π. Let A be the matrix representing the graph $\pi(G)$. The prover sends commitments to all the entries of A.
- The verifier replies with a random bit b.
- If $b = 0$, then the prover opens all the commitments. It also sends π. If $b = 1$, then the prover only opens the entries of A representing a Hamiltonian circuit in $\pi(G)$.

- The verifier accepts only if the reply of the prover is correct.

Perfect completeness follows from the fact that if $x \in \Pi_{\mathrm{Y}}$, then G has a Hamiltonian circuit, and hence the verifier always accepts. Thus, the protocol is V-bit. The prover is efficient because the NIC is efficient, and the witness for x can be efficiently transformed into a witness for G or $\pi(G)$. The protocol is sound because when $x \in \Pi_{\mathrm{N}}$ the scheme is binding and G does not have a Hamiltonian circuit. This implies that the verifier rejects with probability $1/2$.

The zero-knowledge property follows from the hiding property of the NIC. Specifically, in the perfect setting the verifier is honest, and if $b = 0$, then the simulator commits to $\pi(G)$, where π is a random permutation. If $b = 1$, then it commits to the matrix whose entries are all 1. This guarantees perfect simulation. Notice that if we allow the simulator to fail, then it can choose either one of these options with probability $1/2$, and achieve perfect simulation even for malicious verifiers. The same simulator applies also to the statistical and the computational settings, even if we do not allow the simulator to fail (specifically, we execute the simulator $|x|$ times, and output the first transcript, or `fail` if all executions failed, which happens with probability at most $1/2^{|x|}$). □

In the next section we will show that if a problem has a V-bit zero-knowledge proof, then it has a NIC. Hence, the above lemma yields a compiler that transforms any V-bit, zero-knowledge proof (i.e., honest-verifier, inefficient prover) into a *malicious verifier* V-bit zero-knowledge *proof of knowledge* with *an efficient prover*. To achieve this, the compiler constructs the NIC for the V-bit zero-knowledge protocol, and then uses it in the V-bit protocol of Blum [15], which has an efficient prover, and is zero-knowledge against malicious verifiers.

3.2.2 From V-bit Zero-Knowledge Protocols to NIC

Using the idea of [29] we now show how to construct a NIC from a simulator S for any V-bit zero-knowledge protocol $\langle P, V\rangle$. We start with the following idea: to commit to a bit b, execute $S(x)$ using randomness r, obtain a transcript $\langle m_1, b', m_2\rangle$ such that $b = b'$ and V accepts, and output m_1 as a commitment. Let us verify that this NIC is hiding on `YES` instances and binding on `NO` instances. If x is a `YES` instance, then the perfect completeness property guarantees that we always obtain transcripts where V accepts, and since b cannot be determined from such m_1, the commitment is hiding. Conversely, by the soundness property, if x is a `NO` instance, then there are no transcripts $\langle m_1, 0, m_2\rangle$ and $\langle m_1, 1, m_2'\rangle$ such that V accepts in both. However, the issue with this idea is that b' may not be equal to b. To overcome this issue we redefine the commitment to be $\langle m_1, b' \oplus b\rangle$. That is, we execute $S(x)$, obtain $\langle m_1, b', m_2\rangle$, and output $\langle m_1, b' \oplus b\rangle$. Intuitively, since b' is hidden, the bit $b' \oplus b$ is also hidden. Our lemma follows.

Lemma 3.2.3 *Let $\Pi = \langle \Pi_{\mathrm{Y}}, \Pi_{\mathrm{N}}\rangle$ be a promise-problem. If Π has a* V*-bit, public-coin* **HVPZK** *(respectively,* **HVSZK**, **HVCZK***) proof, then Π has a* NIC *that is perfectly (respectively, statistically, computationally) hiding on Π_{Y} and perfectly binding on Π_{N}.*

Proof: Fix a public-coin V-bit **HVPZK** (respectively, **HVSZK**, **HVCZK**) proof $\langle P, V\rangle$ for Π. We assume that $\langle P, V\rangle$ has a simulator S that outputs either $\texttt{fail}$, or transcripts in which V accepts. Using S we define a NIC f for Π as follows. Let $f(x, b; r)$ be the function that executes $S(x)$ with randomness r. If f obtains a transcript $\langle x, m_1', b', m_2'\rangle$ such that $V(x, m_1', b', m_2') = \texttt{accept}$, then f outputs $\langle m_1', b' \oplus b\rangle$. Otherwise, f outputs b.

We show that f is binding on Π_{N}. Let $x \in \Pi_{\mathrm{N}}$. Notice that for any r and b it holds that $f(x, b; r)$ outputs one bit if and only if $f(x, b; r) = b$. Thus, if f outputs one bit, then there are no r and r' such that $f(x, 0; r) = f(x, 1; r')$. For the case where $f(x, b; r)$ outputs a pair $\langle \tilde{m}_1, \tilde{b}\rangle$, recall that $\tilde{b} = b' \oplus b$, where b' is taken from some transcript $\langle x, m_1', b', m_2'\rangle$. Thus, by the definition of f, for any $\tilde{m}_1, \tilde{b}, r$ and r' it holds that $f(x, 0; r) = f(x, 1; r') = \langle \tilde{m}_1, \tilde{b}\rangle$ if and only if there are m_2 and m_2' and such that $V(x, \tilde{m}_1, 0, m_2) = V(x, \tilde{m}_1, 1, m_2') = \texttt{accept}$. However, $\langle P, V\rangle$ is public coin, and by the soundness property of $\langle P, V\rangle$ there are no m_1, m_2 and m_2' such that $V(x, m_1, 0, m_2) = V(x, m_1, 1, m_2') = \texttt{accept}$. Hence, if f does not output one bit, then there are no r and r' such that $f(x, 0; r) = f(x, 1; r')$. We conclude that f is perfectly binding on Π_{N}.

The rest of the proof shows that f is hiding on Π_{Y}. We start with the statistical setting. To show that f is statistically hiding we need to calculate the statistical distance between commitments to 0 and commitments to 1 over $x \in \Pi_{\mathrm{Y}}$. The following probabilities are over the randomness r for f.

$$\begin{aligned}
\Delta(f(x,0), f(x,1)) &= \frac{1}{2}\sum_{\alpha} \left|\Pr[f(x,0) = \alpha] - \Pr[f(x,1) = \alpha]\right| \\
&= \frac{1}{2}\sum_{m_1} \left|\Pr[f(x,0) = \langle m_1, 0\rangle] - \Pr[f(x,1) = \langle m_1, 0\rangle]\right| + \\
&\quad \frac{1}{2}\sum_{m_1} \left|\Pr[f(x,0) = \langle m_1, 1\rangle] - \Pr[f(x,1) = \langle m_1, 1\rangle]\right| + \\
&\quad \frac{1}{2}\sum_{b\in\{0,1\}} \left|\Pr[f(x,0) = b] - \Pr[f(x,1) = b]\right| .
\end{aligned}$$

Notice that the third sum (i.e., the sum over b) equals $\Pr[S(x) = \texttt{fail}]$, the probability that S fails. Now, by Definition 2.5.1 of zero-knowledge, when S is a **HVPZK** simulator it never fails. Thus, $\Pr[S(x) = \texttt{fail}] = 0$. It remains to deal with the sums over m_1. We show that the first sum is upper bounded by $\Delta(\langle P, V\rangle(x), S(x)) - \Pr[S(x) = \texttt{fail}]/2$, and since a symmetric argument applies to the second sum, the total will be upper bounded by $2 \cdot \Delta(\langle P, V\rangle(x), S(x))$. The following probabilities for $\langle P, V\rangle(x)$ and

$S(x)$ are over the randomness to P, V and S, respectively.

$$\begin{aligned}
\tfrac{1}{2}\textstyle\sum_{m_1} \quad & |\Pr[f(x,0)=\langle m_1,0\rangle] - \Pr[f(x,1)=\langle m_1,0\rangle]| = \\
\tfrac{1}{2}\textstyle\sum_{m_1} \quad & |\sum_{m_2}\Pr[S(x)=\langle m_1,0,m_2\rangle] - \sum_{m_2}\Pr[S(x)=\langle m_1,1,m_2\rangle]| = \\
\tfrac{1}{2}\textstyle\sum_{m_1} \quad & |\sum_{m_2}\Pr[S(x)=\langle m_1,0,m_2\rangle] - \sum_{m_2}\Pr[\langle P,V\rangle(x)=\langle m_1,0,m_2\rangle] \\
& -(\sum_{m_2}\Pr[S(x)=\langle m_1,1,m_2\rangle] - \sum_{m_2}\Pr[\langle P,V\rangle(x)=\langle m_1,1,m_2\rangle])| \le \\
\tfrac{1}{2}\textstyle\sum_{m_1,m_2} \quad & (|\Pr[S(x)=\langle m_1,0,m_2\rangle] - \Pr[\langle P,V\rangle(x)=\langle m_1,0,m_2\rangle]| + \\
& |\Pr[S(x)=\langle m_1,1,m_2\rangle] - \Pr[\langle P,V\rangle(x)=\langle m_1,1,m_2\rangle]|) = \\
& \Delta(\langle P,V\rangle(x),S(x)) - \Pr[S(x)=\mathtt{fail}]/2\,.
\end{aligned}$$

In the first equality above we used the fact that S outputs transcripts in which V accepts. In the second equality we used the fact that $\langle P,V\rangle$ is public-coin, which implies that for any m_1 the probability of choosing an element of $\langle P,V\rangle(x)$ whose prefix is $\langle m_1,0\rangle$ equals the probability of choosing an element of $\langle P,V\rangle(x)$ whose prefix is $\langle m_1,1\rangle$. In the last equality we used the fact that $\langle P,V\rangle(x)$ never outputs `fail`, whereas $S(x)$ outputs `fail` with probability $\Pr[S(x)=\mathtt{fail}]$. We conclude that $\Delta(f(x,0),f(x,1)) \le 2\cdot\Delta(S(x),\langle P,V\rangle(x))$. Hence, if S is a **HVPZK** (respectively, **HVSZK**) simulator, then $\Delta(S(x),\langle P,V\rangle(x))$ is 0 for any $x\in\Pi_{\mathrm{Y}}$ (respectively, negligible on Π_{Y}), which implies that f is perfectly (respectively, statistically) hiding on Π_{Y}.

It remains to deal with the case that S is a **HVCZK** simulator. The analysis is analogues to the statistical setting, but in reverse. We define the function $f'(\cdot,b)$ just like f, except that instead of executing the simulator, f' receives a transcript $\langle m_1,b',m_2\rangle$ and outputs $\langle m_1,b'\oplus b\rangle$. Thus, $f'(S(x),b)$ and $f(x,b)$ are identically distributed for any $b\in\{0,1\}$. Assume towards a contradiction that there is a non-uniform family D of polynomial-size circuits that distinguishes $\{f(x,0)\}_{x\in\Pi_{\mathrm{Y}}}$ and $\{f(x,1)\}_{x\in\Pi_{\mathrm{Y}}}$. Thus, D distinguishes $\{f'(S(x),0)\}_{x\in\Pi_{\mathrm{Y}}}$ and $\{f'(S(x),1)\}_{x\in\Pi_{\mathrm{Y}}}$, and the following expression is non-negligible:

$$\begin{aligned}
& |\Pr[D(f'(S(x),0))=1] - \Pr[D(f'(S(x),1))=1]| \le \\
& |\Pr[D(f'(S(x),0))=1] - \Pr[D(f'(\langle P,V\rangle(x),0))=1]| + \\
& |\Pr[D(f'(S(x),1))=1] - \Pr[D(f'(\langle P,V\rangle(x),1))=1]|\,.
\end{aligned}$$

Above we used the fact that $\langle P,V\rangle$ is V-bit, which implies that $f'(\langle P,V\rangle(x),0)$ and $f'(\langle P,V\rangle(x),1)$ are identically distributed for any $x\in\Pi_{\mathrm{Y}}$. It follows that there is $b\in\{0,1\}$ such that D distinguishes $\{f'(\langle P,V\rangle,b)\}_{x\in\Pi_{\mathrm{Y}}}$ and $\{f'(S(x),b)\}_{x\in\Pi_{\mathrm{Y}}}$. This contradicts the fact that S is a **HVCZK** simulator. We conclude that f is computationally hiding on Π_{Y}. The lemma follows. □

Theorem 3.0.2 immediately follows from Lemmas 3.2.2 and 3.2.3. Thus, we get a characterization of V-bit zero-knowledge protocols as NIC. We remark that Theorem 3.0.2 can be extended to arguments, in which case it yields NIC where the binding property holds with respect to computationally bounded senders. Also, it can be extended to relaxed notions of V-bit protocols (e.g., where perfect completeness or public-coins are not required), but we avoid these extensions because they require changing the definition of a NIC.

3.3 Random Self-Reducibility Implies NIC

We prove the folklore theorem that if a problem is random self-reducible (RSR), then it has a perfectly hiding NIC. The advantage of this result is that it enables us to replace the notion of random-self reducibility with the simpler notion of a NIC. For example, in our closure result we can combine RSR problems with problems that are not known to be RSR (such as versions of SD, and the lattice problems of [63]). Hence, we are able to strengthen and unify the results of [84, 79, 50], and achieve all the improvements claimed in the introduction.

Informally, the notion of random self-reducibility [4] considers a set of strings x, each associated with a polynomial-time relation R_x on pairs $\langle z, w\rangle$. Given x and z, there is an algorithm $\mathtt{S}$ uses randomness r to sample the domain of R_x. Specifically, if $z \in R_x$, then $\mathtt{S}$ outputs y such that $\langle y, w'\rangle \in R_x$ for some w'. For example, in the case of GRAPH-ISOMORPHISM the string x is a graph G, and $d(R_G)$ contains all the graphs G' isomorphic to G. Given G and G', the algorithm $\mathtt{S}$ picks a random permutation π and outputs $G'' = \pi(G')$, which is uniformly distributed in R_G.

Two algorithms are at the heart of the notion of random self-reducibility: A_1, which converts a witness for y into a witness for z, and A_2, which converts witness for z into a witness for y. Both A_1 and A_2 use r. Also, there is an algorithm $\mathtt{G}$ that generates random pairs $\langle z', w'\rangle$ from R_x. All of the algorithms are efficient. Again, using GRAPH-ISOMORPHISM as an example, if π' is a witness for G'' (that is, $\pi'(G'') = G$), then $A_1(G, G'', \pi, \pi'') = \pi' \circ \pi$ (because $\pi'(\pi(G')) = G$). Similarly, if π'' is a witness for G' (that is, $\pi''(G') = G$), then $A_2(G, G', \pi, \pi'') = \pi'' \circ \pi^{-1}$ (because $\pi''(\pi^{-1}(G'')) = G$). The algorithm $\mathtt{G}$ simply outputs a random permutation of G. The following definition is similar to that of [79].

Definition 3.3.1 (A random self-reducible problem) *Let p be a polynomial, and let $\mathcal{N} \subset \{0,1\}^*$ be a countable set such that for each $x \in \mathcal{N}$ it holds that R_x is an* **NP***-relation, and the* domain *of R_x, which is denoted $d(R_x) \overset{def}{=} \{z | \exists w\ \langle z, w\rangle \in R_x\}$, satisfies $|d(R_x)| \leq p(|x|)$. The language* $\mathrm{L} \overset{def}{=} \{\langle x, z\rangle | x \in \mathcal{N}, \exists w\ \langle z, w\rangle \in R_x\}$ *is* random self-reducible (RSR) *if there are polynomial time algorithms G, A_1, A_2, and S such that $S(x, z; r) = y \in d(R_x)$ for any $x \in \mathcal{N}$, z, and r, and the following conditions hold.*

1. *If $z \in d(R_x)$, and r is uniformly distributed, then y is uniformly distributed in $d(R_x)$.*

2. *A witness for y yields a witness for z, and vice versa. That is, $\langle z, A_1(x, y, r, w')\rangle \in R_x$ for any $\langle y, w'\rangle \in R_x$, and $\langle y, A_2(x, z, r, w'')\rangle \in R_x$ for any $\langle z, w''\rangle \in R_x$.*

3. *$G(x;r) = \langle z', w'\rangle \in R_x$, and if r is uniformly distributed, then z' is uniformly distributed in $d(R_x)$, and w' is uniformly distributed in $\{w | \langle z, w\rangle \in R_x\}$.*

We prove that random self-reducible problems have a perfectly hiding NIC. Our proof uses the idea behind the construction of the subroutine in the protocol of [79] (see Section 3.3 in [79]). Given $\mathcal{N}$ and R_x as in Definition 3.3.1, we define the problem $\Pi^{\mathrm{L}} \overset{\text{def}}{=} \langle \Pi^{\mathrm{L}}_{\mathrm{Y}}, \Pi^{\mathrm{L}}_{\mathrm{N}}\rangle$, where $\Pi^{\mathrm{L}}_{\mathrm{Y}} \overset{\text{def}}{=} \{\langle x, z\rangle | x \in \mathcal{N}, \exists w \; \langle z, w\rangle \in R_x\}$, and $\Pi^{\mathrm{L}}_{\mathrm{N}} \overset{\text{def}}{=} \{\langle x, z\rangle | x \in \mathcal{N}, \forall w \; \langle z, w\rangle \notin R_x\}$.

Lemma 3.3.2 *If* L *is a random self-reducible language, then* Π^{L} *has a perfectly hiding* NIC*.*

Proof: Let $\mathrm{L} \overset{\text{def}}{=} \{\langle x, z\rangle | x \in \mathcal{N}, \exists w \; \langle z, w\rangle \in R_x\}$ be a random self-reducible language. Consider the algorithms S and G from Definition 3.3.1. Let $G'(x;r)$ be the algorithm that executes $G(x;r)$, obtains $\langle z', w'\rangle$, and outputs z'. We use S and G' to commit to 0 and 1, respectively. Formally, we define our NIC to be a probabilistic polynomial-time Turing machine $f(x, z, b; r)$ that on input $\langle x, z\rangle \in \Pi^{\mathrm{L}}_{\mathrm{Y}} \cup \Pi^{\mathrm{L}}_{\mathrm{N}}$, bit b, and randomness r outputs $S(x, z; r)$ if $b = 0$, and $G'(x; r)$ if $b = 1$.

The efficiency of f follows from the efficiency of S and G. We show that f is perfectly hiding. By Definition 3.3.1, $S(x, z; r) = y$ is uniformly distributed over $d(R_x)$ when r is uniformly distributed and $\langle x, z\rangle \in \Pi^{\mathrm{L}}_{\mathrm{Y}}$. Similarly, $G(x;r) = \langle z', w'\rangle$ and z' is uniformly distributed over $d(R_x)$ when r is uniformly distributed and $x \in \mathcal{N}$. Since the output of f is uniformly distributed over $d(R_x)$ for any b and $\langle x, z\rangle \in \Pi^{\mathrm{L}}_{\mathrm{Y}}$, the ensembles $\{f(x, z, 0; r)\}_{\langle x,z\rangle \in \Pi^{\mathrm{L}}_{\mathrm{Y}}}$ and $\{f(x, z, 1; r)\}_{\langle x,z\rangle \in \Pi^{\mathrm{L}}_{\mathrm{Y}}}$ are statistically identical, and therefore f is perfectly hiding on $\Pi^{\mathrm{L}}_{\mathrm{Y}}$.

We show that f is binding on $\Pi^{\mathrm{L}}_{\mathrm{N}}$. Let $\langle x, z\rangle \in \Pi^{\mathrm{L}}_{\mathrm{N}}$. Assume towards a contradiction that there are r and r' such that $S(x, z; r) = f(x, z, 0; r) = f(x, z, 1; r') = G'(x; r')$, and denote $y = S(x, z; r)$. By the definition of G', there is w' such that $G(x; r') = \langle G'(x; r'), w'\rangle = \langle y, w'\rangle \in R_x$. By the property of A_1 from Definition 3.3.1, it follows that $\langle z, A_1(x, y, r, w')\rangle \in R_x$. Hence, $\langle x, z\rangle \in \Pi^{\mathrm{L}}_{\mathrm{Y}}$, in contradiction to the choice of $\langle x, z\rangle \in \Pi^{\mathrm{L}}_{\mathrm{N}}$. Thus, f is binding on $\Pi^{\mathrm{L}}_{\mathrm{N}}$. $\square$

Notice that in the above proof we did not use Algorithm A_2 from Definition 3.3.1. Neither did we use the fact that A_1 runs in polynomial time, nor did we use the witness that G outputs.

3.4 Closure of Problems Possessing NIC under Monotone Boolean Formulae

In this section we show that the class of problems possessing NIC is closed under *arbitrary* (as opposed to fixed) monotone boolean formulae. Combining this lemma with our previous results yields the NIC framework and its consequences. We start with notation, and formalize our theorem in Section 3.4.1

Intuitively, our goal is to obtain a V-bit zero-knowledge protocol where statements about instances from several problems are being proved. For example, the input may be an instance of GRAPH-ISOMORPHISM and an instance of the lattice problems of [63], and the prover will prove that at least one of these instances is a YES instance. More generally, the input is a vector $\langle \varphi, \langle x_1, \ldots, x_n \rangle \rangle$, where φ is a monotone boolean formula and $x_1, \ldots, x_n$ are instances of a problem Π, and the statement being proved is that $x_1, \ldots, x_n$ satisfy φ. We remark that the x_i can be from different problems, but to simplify the presentation we use only one problem.

To formalize the above intuition we need the following definitions. A *boolean variable* is a variable that can only take the values 0 or 1. We say that ϕ is a monotone boolean formula if ϕ is a boolean variable, or ϕ is of the form $\phi_0 \wedge \phi_1$ or $\phi_0 \vee \phi_1$, where both ϕ_0 and ϕ_1 are monotone boolean formulae. Given a problem $\Pi = \langle \Pi_{\mathrm{Y}}, \Pi_{\mathrm{N}} \rangle$ and $x \in \Pi_{\mathrm{Y}} \cup \Pi_{\mathrm{N}}$, we define the *characteristic function* χ_Π of Π as follows: if $x \in \Pi_{\mathrm{Y}}$, then $\chi_\Pi(x) = 1$, and if $x \in \Pi_{\mathrm{N}}$, then $\chi_\Pi(x) = 0$. Let ϕ be a boolean formula over $a_1, \ldots, a_m$, and let $x_1, \ldots, x_n \in \Pi_{\mathrm{Y}} \cup \Pi_{\mathrm{N}}$ for some $n \geq m$. The *evaluation* of ϕ in $\vec{x} = \langle x_1, \ldots, x_n \rangle$ is denoted $\phi(\vec{x})$, and equals 1 if and only if ϕ is satisfied when a_i is assigned $\chi_\Pi(x_i)$ for each $1 \leq i \leq m$.

We say that a class C of problems is closed under *arbitrary*, monotone boolean formulae if $\Pi \in C$ implies that $\Phi(\Pi) \in C$, where $\Phi(\Pi)$ is defined as follows.

Definition 3.4.1 *Let* $\Pi = \langle \Pi_{\mathrm{Y}}, \Pi_{\mathrm{N}} \rangle$ *be a problem. The problem* $\Phi(\Pi) \overset{def}{=} \langle \Phi(\Pi)_{\mathrm{Y}}, \Phi(\Pi)_{\mathrm{N}} \rangle$ *is defined as*

$$\Phi(\Pi)_{\mathrm{Y}} \overset{def}{=} \{\langle \phi, x_1, \ldots, x_n \rangle | \phi(\chi_\Pi(x_1), \ldots, \chi_\Pi(x_n)) = 1\}$$
$$\Phi(\Pi)_{\mathrm{N}} \overset{def}{=} \{\langle \phi, x_1, \ldots, x_n \rangle | \phi(\chi_\Pi(x_1), \ldots, \chi_\Pi(x_n)) = 0\},$$

where ϕ *is a monotone boolean formula over* $a_1, \ldots, a_m$ *such that* $m \leq n$*, and* $x_i \in \Pi_{\mathrm{Y}} \cup \Pi_{\mathrm{N}}$ *for all* $1 \leq i \leq n$*. We define* $\Phi(\Pi)^k \overset{def}{=} \langle \Phi(\Pi)^{\mathrm{k}}_{\mathrm{Y}}, \Phi(\Pi)_{\mathrm{N}} \rangle$*, where* $\Phi(\Pi)^{\mathrm{k}}_{\mathrm{Y}}$ *is defined as*

$$\Phi(\Pi)^{\mathrm{k}}_{\mathrm{Y}} \overset{def}{=} \{\langle \phi, x_1, \ldots, x_n \rangle | \phi(\chi_\Pi(x_1), \ldots, \chi_\Pi(x_n)) = 1 \wedge \forall i\ |x_i|^k \geq |\phi, x_1, \ldots, x_n|\}.$$

3.4.1 The Closure Result - How to Combine NIC in a Monotone Boolean Formula Fashion

Our closure result states that if a problem Π has a NIC, then the problem $\Phi(\Pi)$ also has a NIC. Consequently, we get that the class of problems possessing NIC (equivalently, V-bit zero-knowledge proofs) is closed under arbitrary, monotone boolean formulae. Notice that this s stronger than saying that the class of problems admitting NIC is closed under the AND and the OR operators.

Theorem 3.4.2 *For any problem* Π *that has a* NIC f*, and for any* $k \in \mathbb{N}$*, there is a* NIC f' *such that*

1. *if* f *is a perfectly hiding* NIC *for* Π*, then* f' *is a perfectly hiding* NIC *for* $\Phi(\Pi)$.
2. *if* f *is a statistically (respectively, computationally) hiding* NIC *for* Π*, then* f' *is a statistically (respectively, computationally) hiding* NIC *for* $\Phi(\Pi)^{\mathrm{k}}$.

We will prove the above theorem by constructing a new NIC for $\Phi(\Pi)$ from the NIC for Π. The advantage of this approach is that we do not need to work with involved notions such as interaction or zero-knowledge. Also, by using the formulation of $\Phi(\Pi)$, we obtain a protocol that works for any formula.

Using the technique of [79] we construct NIC as follows. If f is a NIC for Π, then a NIC for instances of the form $z = \langle a \wedge b, x_1, x_2 \rangle$ can be defined by $f'(z, b; r) = \langle f(x_1, b), f(x_2, b) \rangle$. Thus, if both x_1 and x_2 are `YES` instances of Π, then f' is hiding (because both $f(x_1, b)$ and $f(x_2, b)$ are hiding), and if x_1 or x_2 is a `NO` instance, then f' is binding (because at least one of $f(x_1, b)$ and $f(x_2, b)$ is binding). Notice that we omitted the randomness for f, but the intention is that f' uses independent randomness in each execution. A similar idea applies to the `OR` connector. That is, a NIC for instances of the form $z = \langle a \vee b, x_1, x_2 \rangle$ can be defined by $f'(z, b; r) = \langle f(x_1, b_1), f(x_2, b_2) \rangle$, where b_1 is uniformly chosen, and b_2 is chosen such that $b_1 \oplus b_2 = b$. Thus, if at least one of x_1, x_2 is a `YES` instance of Π, then f' is hiding (because either $f(x_1, b_1)$ or $f(x_2, b_2)$ is hiding), and if both x_1 and x_2 are `NO` instances, then f' is binding (because both $f(x_1, b_1)$ and $f(x_2, b_2)$ are binding). The following construction generalizes these ideas to any monotone boolean formula.

Construction 3.4.3 *We define a recursive function $f'(\phi, \vec{x}, b; r)$. Let f be a* NIC, *and let $b \in \{0, 1\}$. Let ϕ be a monotone boolean formula over the variables $a_1, \ldots, a_m$, and let $\vec{x} = \langle x_1, \ldots, x_n \rangle$ be a vector of n strings, where $n \geq m$. The randomness r for f' is of length polynomial in $|\langle \phi, \vec{x} \rangle|$, and the polynomial is determined from the construction of f', described below.*

1. *If $\phi = a_i$ for some $1 \leq i \leq m$, then return $f(x_i, b, r)$.*
2. *Partition r into r_0 and r_1 (that is, the concatenation $r_0 r_1$ equals r).*
3. *If $\phi = \phi_0 \wedge \phi_1$, then return $\langle f'(\phi_0, \vec{x}, b, r_0), f'(\phi_1, \vec{x}, b, r_1) \rangle$.*
4. *If $\phi = \phi_0 \vee \phi_1$, then return $\langle f'(\phi_0, \vec{x}, b_0, r_0), f'(\phi_1, \vec{x}, b_1, r_1) \rangle$, where $b_0 \in \{0, 1\}$ is uniformly distributed, and b_1 is chosen such that $b_0 \oplus b_1 = b$.*

3.4.2 Proof of the Closure Result

In this section we prove Theorem 3.4.2 from the previous section. This theorem states that if f is a NIC for a problem Π, then f' defined from f as in Construction 3.4.3 is a NIC for $\Phi(\Pi)$. We use the technique of [79]. Intuitively, this technique admits a simple analysis in the perfect setting because the advantage of the adversary remains zero at every stage of construction 3.4.3, and therefore it sums up to zero. However, in the statistical and the computational settings the advantage is non-negligible, and the total may not be negligible. This is why we introduced the constant k in Definition 3.4.1, and this is why we need to provide a more involved analysis. We start with the binding property.

Lemma 3.4.4 *If a function f is binding on a set Π_N, then f' from Construction 3.4.3 is binding on $\Phi(\Pi)_N$.*

Proof: We prove the lemma by induction on the number ℓ of connectives in ϕ. For the base case, $\ell = 0$ and therefore f and f' are identical. Since f is binding on Π_N, we get that f' is binding on $\Phi(\Pi)_N$. Assume the induction hypothesis for all $\ell \geq 1$. Let ϕ be a monotone boolean formula with $\ell + 1$ connectives, and let $\langle \phi, \vec{x} \rangle \in \Phi(\Pi)_N$. Consider the case where $\phi = \phi_0 \wedge \phi_1$, and assume towards contradiction that there are r_0, r_0' and r_1, r_1' such that

$$f'(\phi, 0; r_0 r_0') = \langle f'(\phi_0, 0; r_0), f'(\phi_1, 0; r_0') \rangle = \langle f'(\phi_0, 1; r_1), f'(\phi_1, 1; r_1') \rangle = f'(\phi, 1; r_1 r_1').$$

Since $\langle \phi, \vec{x} \rangle \in \Phi(\Pi)_N$, we can fix $b \in \{0,1\}$ for which $\phi_b(\vec{x}) = 0$. Hence, $f'(\phi_0, 0; r_b) = f'(\phi_0, 1; r_b')$, and since ϕ_b has at most ℓ connectives, we get a contradiction to the induction hypothesis. The case where $\phi = \phi_0 \vee \phi_1$ is similar. Specifically, assume towards contradiction that there are r_0, r_0, r_1, r_1' and $b_0, b_0', b_1, b_1' \in \{0,1\}$ such that $b_0 \oplus b_0' \neq b_1 \oplus b_1'$, and

$$f'(\phi, b_0 \oplus b_0'; r_0 r_0') = \langle f'(\phi_0, b_0; r_0), f'(\phi_1, b_0'; r_0') \rangle = \langle f'(\phi_0, b_1; r_1), f'(\phi_1, b_1'; r_1') \rangle = f'(\phi, b_1 \oplus b_1'; r_1 r_1').$$

Thus, there is $d \in \{0,1\}$ such that $b_d \neq b_d'$ and $f'(\phi_0, 0; r_d) = f'(\phi_0, 1; r_d')$. Since ϕ_d has at most ℓ connectives, we get a contradiction to the induction hypothesis. □

In the following section we prove the hiding property in the statistical setting, hence obtaining Theorem 3.4.2 for perfectly and statistically hiding NIC.

The Hiding Property in the Statistical Setting

Recall that f' outputs $\langle f(x_1, b), f(x_2, b) \rangle$ on input $z = \langle a \wedge b, x_1, x_2 \rangle \in \Phi(\Pi)$ and bit b, and that if both x_1 and x_2 are `YES` instances, then f' is hiding (because both $f(x_1, b)$ and $f(x_2, b)$ are hiding). This motivation works in the perfect setting, but in the statistical setting the output of $f(x_1, b)$ and $f(x_2, b)$ may not perfectly hide the bit b. Intuitively, both $f(x_1, b)$ and $f(x_2, b)$ may leak a small amount of information about b. Thus, we need to quantify this amount. We use the following lemma, which is similar to the *Direct Product Lemma* and the *XOR Lemma* from Vadhan's thesis [86]. To simplify the presentation we omit $\vec{x}$ and r from the parameters to f'. The proof is technical, and appears in Section 3.6.1.

Lemma 3.4.5 *Let f' be a function, let $\vec{x}$ be a vector of strings, and let ϕ_0 and ϕ_1 be monotone boolean formula. Then,*

$$\begin{aligned} \Delta(f'(\phi_0 \wedge \phi_1, 0), f'(\phi_0 \wedge \phi_1, 1)) &\leq \Delta(f'(\phi_0, 0), f'(\phi_0, 1)) + \Delta(f'(\phi_1, 0), f'(\phi_1, 1)), \text{and} \\ \Delta(f'(\phi_0 \vee \phi_1, 0), f'(\phi_0 \vee \phi_1, 1)) &\leq \Delta(f'(\phi_0, 0), f'(\phi_0, 1)) \cdot \Delta(f'(\phi_1, 0), f'(\phi_1, 1)). \end{aligned}$$

Now we prove Theorem 3.4.2 in the statistical setting. The idea is to recursively apply the above lemma, and to carefully add the the amount of information leaked at each stage of Construction 3.4.3. For this purpose we introduce the notation of $P(\phi)$, which denotes the multiset containing all the indices of boolean variables in a formula ϕ (e.g., if $\phi = (\alpha_1 \vee \alpha_2) \wedge \alpha_1$, then $P(\phi) = \{1, 1, 2\}$).

Lemma 3.4.6 *If a function f is perfectly (respectively, statistically) hiding on a set Π_Y, then for any $k \in \mathbb{N}$ Construction 3.4.3 of f' is perfectly (respectively, statistically) hiding on $\Phi(\Pi)_Y$ (respectively, $\Phi(\Pi)_Y^k$).*

Proof: Let $k \in \mathbb{N}$. We start with the statistical setting, and the perfect setting will follow. Our goal is to show that the statistical distance between commitments to 0 and commitments to 1 is negligible. Thus, as a first step we prove that for any vector $\langle \phi, \vec{x} \rangle = \langle \phi, \langle x_1, \ldots, x_n \rangle \rangle$ it holds that

$$\Delta\big(f'(\phi, \vec{x}, 0), f'(\phi, \vec{x}, 1)\big) \leq \sum_{i \in P(\phi)} \Delta\big(f(x_i, 0), f(x_i, 1)\big).$$

We prove the above hypothesis by induction on the number ℓ of connectives in ϕ. The base case is trivial because $\ell = 0$, and therefore f and f' are identical. Assume the induction hypothesis for all $\ell \geq 1$, and let $\langle \phi, \vec{x} \rangle = \langle \phi, x_1, \ldots, x_n \rangle \in \Phi(\Pi)^k_Y$. Notice that regardless of whether ϕ equals $\phi_0 \wedge \phi_1$ or $\phi_0 \vee \phi_1$, by Lemma 3.4.5 it holds that

$$\Delta(f'(\phi, \vec{x}, 0), f'(\phi, \vec{x}, 1)) \leq \Delta(f'(\phi_0, \vec{x}, 0), f'(\phi_0, \vec{x}, 1)) + \Delta(f'(\phi_1, \vec{x}, 0), f'(\phi_1, \vec{x}, 1)).$$

Now we apply the induction hypothesis to both ϕ_0 and ϕ_1. Hence, we get that

$$\Delta(f'(\phi, \vec{x}, 0), f'(\phi, \vec{x}, 1)) \leq \sum_{i \in P(\phi_0)} \Delta(f(x_i, 0), f(x_i, 1)) + \sum_{i \in P(\phi_1)} \Delta(f(x_i, 0), f(x_i, 1)).$$

Since $P(\phi) = P(\phi_0) \cup P(\phi_1)$, the induction follows. Notice that if f is perfectly hiding, then the above sum equals 0, and thus f' is perfectly hiding on $\Phi(\Pi)_Y$. In the statistical setting we are not done because we need to show that this sum is negligible in the length of $\langle \phi, \vec{x} \rangle$. Thus, we proceed to the next step.

Let $a \in \mathbb{N}$. Since f is statistically hiding on Π_Y, there is N such that $\Delta(f(x, 0), f(x, 1)) \leq 1/|x|^{ak+k}$ for any $x \in \Pi_\mathrm{Y}$ of length at least N. Notice that by Definition 3.4.1 of $\Phi(\Pi)^k$, there is N' such that for any $\langle \phi, x_1, \ldots, x_n \rangle \in \Phi(\Pi)^k_Y$ of length at least N' it holds that $|x_i| \geq N$ for each $1 \leq i \leq n$. Hence, fixing N' we are guaranteed that for any $\langle \phi, \vec{x} \rangle = \langle \phi, x_1, \ldots, x_n \rangle \in \Phi(\Pi)^k_Y$ of length of at least N' it holds that $\Delta(f(x_i, 0), f(x_i, 1)) \leq 1/|x_i|^{ak+k}$ for each $1 \leq i \leq n$, and by the fact that we proved using induction,

$$\Delta(f'(\phi, \vec{x}, 0), f'(\phi, \vec{x}, 1)) \leq \sum_{i \in P(\phi)} \Delta(f(x_i, 0), f(x_i, 1)) \leq \sum_{i \in P(\phi)} 1/|x_i|^{ak+k}.$$

It remains to show that $\sum_{i \in P(\phi)} 1/|x_i|^{ak+k} \leq 1/|\langle \phi, \vec{x} \rangle|^a$ for any $\langle \phi, \vec{x} \rangle \in \Phi(\Pi)^k$ of length at least N'. This follows from Definition 3.4.1 of $\Phi(\Pi)^k$ because for any $\langle \phi, x_1, \ldots, x_n \rangle \in \Phi(\Pi)^k_Y$ and $1 \leq i \leq n$ it holds that $|\langle \phi, x_1, \ldots, x_n \rangle| \leq |x_i|^k$, which implies that for any $1 \leq i \leq n$ the total number of variables in ϕ is at most $|x_i|^k$. Hence, for any $\langle \phi, x_1, \ldots, x_n \rangle \in \Phi(\Pi)^k_Y$ there is $1 \leq j \leq n$ such that

$$\sum_{i \in P(\phi)} 1/|x_i|^{ak+k} \leq |P(\phi)| \cdot 1/|x_j|^{ak+k} \leq |x_j|^k \cdot 1/|x_j|^{ak+k} \leq 1/|\langle \phi, x_1, \ldots, x_n \rangle|^a.$$

We conclude that $\Delta(f'(\phi, \vec{x}, 0), f'(\phi, \vec{x}, 1)) \leq 1/|\langle \phi, \vec{x} \rangle|^a$ for any $a \in N$ and sufficiently long $\langle \phi, \vec{x} \rangle \in \Phi(\Pi)^k_Y$. Thus, f' is statistically hiding on $\Phi(\Pi)^k_Y$. $\square$

The Hiding Property in the Computational Setting

In the computational setting we need a lemma analogous to Lemma 3.4.5. Roughly speaking, we use a distinguisher D on $\phi = \phi_0 \wedge \phi_1$ to construct circuits C_0 and C_1 such that either C_0 is a distinguisher on ϕ_0 or C_1 is a distinguisher on ϕ_1. Notice that we also need to make sure that the size of C_0 and C_1 is related to the size of D, so that later, when we apply this lemma inductively, the size of the resulting distinguisher will still be polynomial. The proof is technical, and appears in Section 3.6.2.

Lemma 3.4.7 *Let f' be the function from construction 3.4.3, let ϕ_0 and ϕ_1 be monotone boolean formulae, and let $\vec{x}$ be a vector of strings. Given a circuit D, for each $i \in \{0,1\}$ there are circuits C_i and E_i of size at most $|D| + |f'| + |\phi_{\bar{i}}| + \sum_{j \in P(\phi_{\bar{i}})} |x_j|$ each such that*

$$\mathit{adv}(D, f'(\phi_0 \wedge \phi_1, 0), f'(\phi_0 \wedge \phi_1, 1)) \leq \mathit{adv}(C_0, f'(\phi_0, 0), f'(\phi_0, 1)) + \mathit{adv}(C_1, f'(\phi_1, 0), f'(\phi_1, 1)),$$
$$\textit{and } \mathit{adv}(D, f'(\phi_0 \vee \phi_1, 0), f'(\phi_0 \vee \phi_1, 1)) \leq \mathit{adv}(E_i, f'(\phi_i, 0), f'(\phi_i, 1)).$$

Finally, we prove Theorem 3.4.2 in the computational setting. The proof is complicated because we start with a distinguisher D whose input contains an instance of $\Phi(\Pi)^k$, and from this distinguisher we need to construct a distinguisher C whose input is an instance of Π. To do this, we will define an infinite sequence of $x \in \Pi_Y$ from the sequence $\Phi(\Pi)^k_Y$, making sure that the size of C is polynomial in the size of D.

Lemma 3.4.8 *If f is a computationally hiding* NIC *on a set Π_Y, then for any $k \in \mathbb{N}$ Construction 3.4.3 of f' is computationally hiding on $\Phi(\Pi)^k_Y$.*

Proof: Let $k \in \mathbb{N}$. Our goal is to show that if a circuit D distinguishes commitments of formula ϕ (i.e., commitments to 0 and commitments to 1), then there is a circuit C_i that distinguishes commitments on one of the x_i, and the size of C_i is polynomial in the size of D. First, we prove that for any circuit D and any vector $\langle \phi, \vec{x} \rangle$ there are circuits C_i, each of size at most $|D| + |P(\phi)| \cdot |f'| + |\phi| + \sum_{j \in P(\phi)} |x_j|$, such that

$$\mathtt{adv}\big(D, f'(\phi, \vec{x}, 0), f'(\phi, \vec{x}, 1)\big) \leq \sum_{i \in P(\phi)} \mathtt{adv}\big(C_i, f(x_i, 0), f(x_i, 1)\big).$$

We prove the above hypothesis by induction on the number ℓ of connectives in ϕ. The base case is trivial because $\ell = 0$, and ϕ is a boolean variable (i.e., $\phi = a_i$). Thus, f and f' are identical, and we can take $C_i = D$. Assume the induction hypothesis for all $\ell \geq 1$, let D be a circuit, and let $\langle \phi, \vec{x} \rangle = \langle \phi, x_1, \ldots, x_n \rangle$ be a vector. We only treat the case $\phi = \phi_0 \wedge \phi_1$ because the case $\phi_0 \vee \phi_1$ is similar. Omitting $\vec{x}$, by Lemma 3.4.7, there are circuits C_0 and C_1 such that

$$\mathtt{adv}(D, f'(\phi_0 \wedge \phi_1, 0), f'(\phi_0 \wedge \phi_1, 1)) \leq \mathtt{adv}(C_0, f'(\phi_0, 0), f'(\phi_0, 1)) + \mathtt{adv}(C_1, f'(\phi_1, 0), f'(\phi_1, 1)),$$

and the size of C_0 is at most $|D| + |f'| + |\phi_1| + \sum_{j\in P(\phi_1)} |x_j|$. Thus, by the induction hypothesis for ϕ_0, there are circuits C_i^0 such that

$$\mathtt{adv}\big(C_0, f'(\phi_0, \vec{x}, 0), f'(\phi_0, \vec{x}, 1)\big) \leq \sum_{i\in P(\phi_0)} \mathtt{adv}\big(C_i^0, f(x_i, 0), f(x_i, 1)\big),$$

and the size of each of the circuits C_i^0 is at most $\big(|D| + |P(\phi_1)| \cdot |f'| + |\phi_1| + \sum_{j\in P(\phi_1)} |x_j|\big) + |P(\phi_0)| \cdot |f'| + |\phi_0| + \sum_{j\in P(\phi_0)} |x_j|$, which equals $|D| + |P(\phi)| \cdot |f'| + |\phi| + \sum_{j\in P(\phi)} |x_j|$. A similar argument applies to C_1. Thus, denoting the circuits corresponding to C_1 by C_i^1 we get that

$$\begin{aligned}\mathtt{adv}(D, f'(\phi_0 \wedge \phi_1, 0), f'(\phi_0 \wedge \phi_1, 1)) \quad &\leq \quad \sum_{i\in P(\phi_0)} \mathtt{adv}\big(C_i^0, f(x_i, 0), f(x_i, 1)\big) + \\ &\qquad \sum_{i\in P(\phi_1)} \mathtt{adv}\big(C_i^1, f(x_i, 0), f(x_i, 1)\big).\end{aligned}$$

Since the size of the circuits C_i^0 and C_i^1 is as stated in hypothesis, the induction follows.

In the rest of the proof we show that the advantage is negligible in the length of $\langle\phi, \vec{x}\rangle$. Formally, we assume towards a contradiction that there is $a \in \mathbb{N}$, a polynomial-size circuit D, and an infinite sequence I of vectors $\langle\phi, \vec{x}\rangle \in \Phi(\Pi)^k_Y$ such that $\mathtt{adv}(D, f'(\phi, \vec{x}, 0), f'(\phi, \vec{x}, 1)) \geq 1/|\langle\phi, \vec{x}\rangle|^a$ for all $\langle\phi, \vec{x}\rangle \in I$, and then we show that this contradicts the fact that f is a computationally hiding NIC on Π_Y.

Fix D, I and a. Recall that for any vector $\langle\phi, \vec{x}\rangle$ it holds that $|P(\phi)| \leq |\langle\phi, \vec{x}\rangle|$, and that the size of D and f' is polynomial in the size of $|\langle\phi, \vec{x}\rangle|$. Thus, by the fact that we proved using induction there is a polynomial p such that for each $\langle\phi, \vec{x}\rangle \in I$ there are circuits C_i of size at most $p(|\langle\phi, \vec{x}\rangle|)$ and $\sum_{i\in P(\phi)} \mathtt{adv}\big(C_i, f(x_i, 0), f(x_i, 1)\big) \geq \mathtt{adv}\big(D, f'(\phi, \vec{x}, 0), f'(\phi, \vec{x}, 1)\big) \geq 1/|\langle\phi, \vec{x}\rangle|^a$. Now, by Definition 3.4.1 of $\Phi(\Pi)^k$, for any $\langle\phi, \vec{x}\rangle = \langle\phi, x_1, \ldots, x_n\rangle \in \Phi(\Pi)^k_Y$ and $1 \leq j \leq n$ it holds that $|x_j|^k \geq |\langle\phi, \vec{x}\rangle|$. Thus, for each each $\langle\phi, \vec{x}\rangle \in I$ there is $1 \leq j \leq n$ and a circuit C_{x_j} of size at most $p(|x_j|^k)$ such that $\mathtt{adv}\big(C_{x_j}, f(x_j, 0), f(x_j, 1)\big) \geq |x_j|^{-k} \cdot 1/|\langle\phi, \vec{x}\rangle|^a \geq 1/|x_j|^{ak-k}$. Since there are infinitely many such x_j, we get a contradiction to the premise that f is a computationally hiding NIC on Π_Y. □

3.5 Consequences - the NIC Framework

We showed that the class of problems possessing V-bit zero-knowledge proofs can be represented by NIC, it is closed under arbitrary monotone boolean formula, and it contains random-self reducible problems. Now we show that this yields a framework that connects the notion of NIC to many zero-knowledge protocols in various settings, and allows us to study all the known **PZK** problems through NIC.

Studying Zero-Knowledge Protocols Through NIC

Consider the *non-black-box* protocol of Barak [7]. This protocol is important because it has desirable properties that are impossible to achieve with black-box simulation [40, 22, 8]. Specifically, it is a public-coin, computational zero-knowledge (**CZK**) argument with a constant number of rounds, perfect completeness, and a negligible soundness error. This protocol applies to any **NP** problem, and it assumes the existence of commitment schemes, and collision-resistent hash functions.

Our framework enables us to replace the commitment scheme in the protocol of Barak [7] with a NIC. For example, we get that if a problem has a perfectly hiding NIC (e.g., GRAPH-ISOMORPHISM, DISCRETE-LOGARITHM, etc.), then the protocol of [7] is a **PZK** argument for this problem. This is a significant improvement of the zero-knowledge property from computational to perfect, but it applies to problems admitting NIC, whereas the result of Barak [7] applies to **NP**. Furthermore, this is the first evidence that *perfect* (as opposed to *computational*) zero-knowledge arguments with *non-black-box* simulators can be achieved using only collision resistant hash-functions.

Corollary 3.5.1 (using [7]) *Assuming the existence of collision-resistant hash functions, if a problem has a perfectly (respectively, statistically) hiding* NIC*, then it has a constant-round, public-coin* ***PZK*** *(respectively,* ***SZK****) argument with a negligible soundness error. The protocol has a non-black-box simulator that runs in strict polynomial time, and it remains zero-knowledge even when composed concurrently polynomially many times, for any fixed polynomial. The same applies to computationally hiding* NIC *if, in addition, the underlying problem is in* ***NP***.

We remark that the existence of collision resistant hash-functions implies that of commitment schemes, but the hiding property of the schemes is computational, and thus they only yield computational (as opposed to perfect) zero-knowledge arguments.

Abstraction and Closure

Itoh, Ohta, and Shizuya [50] also observed that in the protocol of [41] for GRAPH-NONISOMORPHISM a NIC with reversed properties can replace the commitment scheme. By *reversed* we mean that the hiding property holds on NO instances of the problem (instead of YES instances), and the binding property holds on YES instances (instead of NO instances). Micciancio, Ong, Sahai, and Vadhan [62] showed that this also applies to the protocol of Prabhakaran, Rosen and Sahai [76], and then they constructed such NIC for specific problems (e.g., GRAPH-NONISOMORPHISM, and variants of STATISTICAL-DISTANCE).

Our framework strengthens and simplifies the results of [50, 62]. For example, since we already have a characterization result, unlike in [62], we do not need to construct NIC with reversed properties for specific problems (e.g., GRAPH-NONISOMORPHISM) or to be familiar with the definition of these problems (e.g., the lattice problems of [63]). Also, our framework shows that such NIC are closed under monotone boolean formulae. Thus, when we plug our framework to the theorem of [62] we get that arbitrary, monotone boolean formulae over a large class of problems (which contains, e.g., the complement of any random self-reducible problem) *unconditionally* have a concurrent zero-knowledge proof.

Corollary 3.5.2 (using [76, 62]) *If a problem* Π *has a statistically (respectively, computationally) hiding* NIC*, then* $\overline{\Pi}$ *has a public-coin, concurrent* ***SZK*** *proof (respectively, argument). The simulator is black-box, and the zero-knowledge property holds even with respect to computationally unbounded verifiers.*

The above improvements to the work of [62] demonstrate that in some cases we can take protocols that deal with a specific problem that has a NIC (or whose complement has a NIC), and then generalize these protocols to any problem that has a NIC, and monotone boolean formulae over such problems. This allows us to obtain more general results, without the need to refer to specific problems. For example, such improvements also apply to the local zero-knowledge protocols of [61], and to the quantum zero-knowledge protocols of [88].

Unifying Previous Works, Random Self-Reducibility, and Σ-protocols

Our framework replaces the notions of random self-reducibility [4] and Σ-protocols [26] with the simpler notion of a NIC, and then ties them with all the improvements mentioned above. For example, our framework unifies under the theme of NIC the results of Tompa and Woll [84], De Santis, Di Crescenzo, Persiano, and Yung [79], and Itoh, Ohta, and Shizuya [50]. To further clarify this contribution, we explicitly list the relevant main results of these papers.

- Tompa and Woll [84] showed that any self-reducible problem has a perfect zero-knowledge (**PZK**) proof with an efficient prover.
- De Santis, Di Crescenzo, Persiano, and Yung [79] generalized the result of [84] to monotone boolean formulae over random self-reducible (RSR) problems.
- Itoh, Ohta, and Shizuya [50] defined what we call a perfectly hiding NIC, and by showing that specific languages (e.g., variants of GRAPH-ISOMORPHISM) admit such a NIC, they obtained **PZK** proofs with efficient provers for these problems.

The above works focus on RSR problems and deal only with the perfect setting. In contrast, our framework includes RSR problems, as well as problems that are not known to be RSR (such as variants of SD [77]). Our framework also considers the statistical and the computational settings, where closure under monotone boolean formulae is technically more involved. Hence, our framework yields stronger results under one simple theme, and these results apply also to statistical and computational zero-knowledge. Specifically, we get the following.

Corollary 3.5.3 (using [79, 50, 29]) *Consider the class of problems admitting perfectly hiding* NIC*, statistically hiding* NIC*, and* ***NP*** *problems admitting computationally hiding* NIC*. This class contains* RSR *problems, and it is closed under* arbitrary *(as opposed to fixed) monotone boolean formulae.*

Any problem in this class has a zero-knowledge proof with an efficient prover, and the zero-knowledge property is inherited from the hiding property of the scheme. Furthermore, the zero-knowledge proof is also a proof of knowledge *[39].*

3.5.1 Sigma protocols

Σ-protocols [26] are related to V-bit protocols in that they are also 3-round and public-coin, but instead of sending a bit, the verifier sends a string. We do not know if any V-bit zero-knowledge protocol is also a Σ-protocol. However, using our characterization result we prove that these notions are in fact equivalent. That is, a language has a V-bit zero-knowledge protocol if and only if it has a Σ-protocol. Thus, all of our improvements immediately extend to problems admitting Σ-protocols.

Lemma 3.5.4 *A problem Π has a V-bit **HVPZK** (respectively, **HVSZK**, **HVCZK**) proof if and only if Π has a perfect (respectively, statistical, computational) Σ-protocol.*

Sigma protocols play an important role in cryptography. Such protocols were given by Schnorr [81] and Guillou and Quisquater [48], and the notion of Σ-protocols was later formalized in the thesis of Cramer [26]. We start with the definition of Σ-protocols.

Definition 3.5.5 (Σ-protocol [30]) *Let p be a polynomial, and let R be a relation such that $|w| \leq p(|x|)$ for any $\langle x, w\rangle \in R$. An interactive protocol $\langle P, V\rangle$ is a* Σ-protocol *for R if V runs in polynomial time, and the following properties hold.*

- **Public-coin, 3-round:** *on common input x, the prover P sends a, the verifier V replies with a uniformly chosen string e, the prover sends back z, and V accepts or rejects based on $\langle x, a, e, z\rangle$.*
- **Perfect completeness:** *if there is w such that $\langle x, w\rangle \in R$, then V accepts x with probability 1 over the randomness for P and V.*
- **Special soundness:** *there is a polynomial-time Turing machine M such that for any x, if $\langle a, e, z, \texttt{accept}\rangle$ and $\langle a, e', z', \texttt{accept}\rangle$ are in $\langle P, V\rangle(x)$ and $e \neq e'$, then $M(a, e, e', z, z') = w$ and $\langle x, w\rangle \in R$.*
- **Special honest-verifier zero-knowledge:** *there is a probabilistic, polynomial-time Turing machine S, called* the simulator, *such that for any $\langle x, w\rangle \in R$ and e, the output of $S(x, e)$ is identically distributed to $\langle P, V_e\rangle(x)$, where V_e is the verifier that sends e as its random string.*

Definition 3.2.1 of a V-bit protocol is similar to that of a Σ-protocol in that both of them consider 3-round, public-coin protocols with perfect completeness.

The difference between the notions is that V-bit protocols make no reference to relations, zero-knowledge, or special soundness. Also, in V-bit protocols the verifier sends only one bit, whereas in Σ-protocols the verifier sends a string e. However, as was observed by Damgård, a protocol remains Σ-protocol even if instead of sending e the verifier sends one bit b, and e is defined as b followed by zeroes [30]. Thus, if a relation R has a Σ-protocol, then R has a V-bit zero-knowledge protocol. Now we show that the opposite is also true, thus proving Lemma 3.5.4.

Proof of Lemma 3.5.4: (sketch) Let $\Pi = \langle \Pi_Y, \Pi_N \rangle$ be a problem, and let $\langle P, V \rangle$ be a V-bit zero-knowledge proof for Π. We show that Π has Σ-protocol with the same zero-knowledge property. We start with the observation that as a class of promise problems, Π is in **NP**. This is so because if $x \in \Pi_Y$, then there are prover messages m_1, m_2, m_2' such that V accepts on both transcripts $\langle x, m_1, 0, m_2 \rangle$ and $\langle x, m_1, 1, m_2' \rangle$, and if $x \in \Pi_N$, then no such transcripts exist. Thus, $\langle m_1, m_2, m_2' \rangle$ is a witness for x. By our characterization result, Π has a NIC f. Thus, the protocol $\langle P', V' \rangle$ of Blum [15], where P' proves to V' that $x \in \Pi_Y$ using the witness $\langle m_1, m_2, m_2' \rangle$, is a zero-knowledge proof for Π. Notice that the resulting proof inherits its zero-knowledge property from the hiding property of the NIC, and it has perfect completeness. Since the proof is also V-bit, it satisfies the special soundness and the special honest-verifier zero-knowledge conditions. $\square$

3.6 Proofs of Lemma 3.4.5 and Lemma 3.4.7

In this section we provide the proofs for the technical lemmas from Section 3.4.

3.6.1 Proof of Lemma 3.4.5

Intuitively, this lemma sums the statistical distance for the `AND` and the `OR` operators. We start with the case where $\phi = \phi_0 \wedge \phi_1$. Recall that for any bit b it holds that $\Pr[f'(\phi, b) = \langle \alpha, \beta \rangle] = \Pr[f'(\phi_0, b) = \alpha] \cdot \Pr[f'(\phi_1, b) = \beta]$. Hence,

$$\begin{aligned}
&\Delta(f'(\phi,0), f'(\phi,1)) = \\
&\sum_{\alpha,\beta} |\Pr[f'(\phi_0,0)=\alpha]\cdot\Pr[f'(\phi_1,0)=\beta] - \Pr[f'(\phi_0,1)=\alpha]\cdot\Pr[f'(\phi_1,1)=\beta]| = \\
&\sum_{\alpha,\beta} |\Pr[f'(\phi_0,0)=\alpha]\cdot\Pr[f'(\phi_1,0)=\beta] - \Pr[f'(\phi_0,1)=\alpha]\cdot\Pr[f'(\phi_1,0)=\beta] + \\
&\qquad \Pr[f'(\phi_0,1)=\alpha]\cdot\Pr[f'(\phi_1,0)=\beta] - \Pr[f'(\phi_0,1)=\alpha]\cdot\Pr[f'(\phi_1,1)=\beta]| \le \\
&\sum_{\alpha,\beta} |\Pr[f'(\phi_0,0)=\alpha]\cdot\Pr[f'(\phi_1,0)=\beta] - \Pr[f'(\phi_0,1)=\alpha]\cdot\Pr[f'(\phi_1,0)=\beta]| + \\
&\sum_{\alpha,\beta} |\Pr[f'(\phi_0,1)=\alpha]\cdot\Pr[f'(\phi_1,0)=\beta] - \Pr[f'(\phi_0,1)=\alpha]\cdot\Pr[f'(\phi_1,1)=\beta]| = \\
&\sum_{\beta} \Pr[f'(\phi_1,0)=\beta]\cdot\Big(\sum_{\alpha} |\Pr[f'(\phi_0,0)=\alpha] - \Pr[f'(\phi_0,1)=\alpha]|\Big) + \\
&\sum_{\alpha} \Pr[f'(\phi_0,1)=\alpha]\cdot\Big(\sum_{\beta} |\Pr[f'(\phi_1,0)=\beta] - \Pr[f'(\phi_1,1)=\beta]|\Big) = \\
&\Delta(f'(\phi_0,0), f'(\phi_0,1)) + \Delta(f'(\phi_1,0), f'(\phi_1,1)).
\end{aligned}$$

The case where $\phi = \phi_0 \vee \phi_1$ is different because the bit b is shared between two bits b_0 and b_1. Specifically,

$$\Pr[f'(\phi,0) = \langle\alpha,\beta\rangle] = \frac{1}{2}\cdot\Pr[f'(\phi_0,0)=\alpha]\cdot\Pr[f'(\phi_1,0)=\beta] + \frac{1}{2}\cdot\Pr[f'(\phi_0,1)=\alpha]\cdot\Pr[f'(\phi_1,1)=\beta]$$

and

$$\Pr[f'(\phi,1) = \langle\alpha,\beta\rangle] = \frac{1}{2}\cdot\Pr[f'(\phi_0,0)=\alpha]\cdot\Pr[f'(\phi_1,1)=\beta] + \frac{1}{2}\cdot\Pr[f'(\phi_0,1)=\alpha]\cdot\Pr[f'(\phi_1,0)=\beta].$$

Hence,

$$\begin{aligned}
&\Pr[f'(\phi,1) = \langle\alpha,\beta\rangle] - \Pr[f'(\phi,0) = \langle\alpha,\beta\rangle] = \\
&1/2(\Pr[f'(\phi_0,0)=\alpha]\cdot\Pr[f'(\phi_1,0)=\beta] + \Pr[f'(\phi_0,1)=\alpha]\cdot\Pr[f'(\phi_1,1)=\beta]) - \\
&1/2(\Pr[f'(\phi_0,0)=\alpha]\cdot\Pr[f'(\phi_1,1)=\beta] + \Pr[f'(\phi_0,1)=\alpha]\cdot\Pr[f'(\phi_1,0)=\beta]) = \\
&1/2\cdot\Pr[f'(\phi_1,0)=\beta](\Pr[f'(\phi_0,0)=\alpha] - \Pr[f'(\phi_0,1)=\alpha]) - \\
&1/2\cdot\Pr[f'(\phi_1,1)=\beta](\Pr[f'(\phi_0,0)=\alpha] - \Pr[f'(\phi_0,1)=\alpha]) = \\
&1/2\cdot(\Pr[f'(\phi_1,0)=\beta] - \Pr[f'(\phi_1,1)=\beta])(\Pr[f'(\phi_0,0)=\alpha] - \Pr[f'(\phi_0,1)=\alpha]).
\end{aligned}$$

Using the above equality we conclude that

$$\begin{aligned}
&\Delta(f'(\phi,0), f'(\phi,1)) = \\
&\frac{1}{2}\sum_{\langle\alpha,\beta\rangle} |\Pr[f'(\phi,1) = \langle\alpha,\beta\rangle] - \Pr[f'(\phi,0) = \langle\alpha,\beta\rangle]| = \\
&\frac{1}{2}\cdot\frac{1}{2}\sum_{\langle\alpha,\beta\rangle} |(\Pr[f'(\phi_1,0)=\beta] - \Pr[f'(\phi_1,1)=\beta])(\Pr[f'(\phi_0,0)=\alpha] - \Pr[f'(\phi_0,1)=\alpha])| = \\
&\Big(\frac{1}{2}\sum_{\beta}|\Pr[f'(\phi_1,0)=\beta] - \Pr[f'(\phi_1,1)=\beta]|\Big)\cdot\Big(\frac{1}{2}\sum_{\alpha}|\Pr[f'(\phi_0,0)=\alpha] - \Pr[f'(\phi_0,1)=\alpha]|\Big) \\
&= \Delta(f'(\phi_1,0), f'(\phi_1,1))\cdot\Delta(f'(\phi_0,0), f'(\phi_0,1)).
\end{aligned}$$

3.6.2 Proof of Lemma 3.4.7

Intuitively, this lemma sums the computational distance for a circuit D under both the `AND` and the `OR` operators. Fix $\phi_0, \phi_1, \vec{x}$ and D. To simplify the presentation we omit $\vec{x}$. We start with the $\wedge$ operator. Let C_0 (respectively, C_1) be the circuit that on input y obtains a random sample y' of $f'(\phi_0, 0)$ (respectively,

$f'(\phi_1, 1)$), and outputs $D(y', y)$ (respectively, $D(y, y')$). Thus, by Construction 3.4.3 of f',

$$\begin{aligned}
&\texttt{adv}(D, f'(\phi_0 \wedge \phi_1, 0), f'(\phi_0 \wedge \phi_1, 1)) = \\
&|\Pr[D(\langle f'(\phi_0, 0), f'(\phi_1, 0)\rangle) = 1] - \Pr[D(\langle f'(\phi_0, 1), f'(\phi_1, 1)\rangle) = 1]| = \\
&|\Pr[D(\langle f'(\phi_0, 0), f'(\phi_1, 0)\rangle) = 1] - \Pr[D(\langle f'(\phi_0, 0), f'(\phi_1, 1)\rangle) = 1] + \\
&\Pr[D(\langle f'(\phi_0, 0), f'(\phi_1, 1)\rangle) = 1] - \Pr[D(\langle f'(\phi_0, 1), f'(\phi_1, 1)\rangle) = 1]| \leq \\
&\texttt{adv}(C_0, f'(\phi_1, 0), f'(\phi_1, 1)) + \texttt{adv}(C_1, f'(\phi_0, 0), f'(\phi_0, 1)).
\end{aligned}$$

We turn our attention to the $\vee$ operator. Let E_0 (respectively, E_1) be the circuit that on input y uniformly picks $b' \in \{0, 1\}$, obtains a random sample y' of $f'(\phi_1, b')$ (respectively, $f'(\phi_0, b')$), and outputs $b' \oplus D(y, y')$ (respectively, $b' \oplus D(y', y)$). We only consider the case of E_0 because the case of E_1 is symmetric. Thus, by Construction 3.4.3 of f',

$$\begin{aligned}
&\Pr[D(f'(\phi_0 \vee \phi_1, 0)) = 1] = \\
&1/2 \cdot \Pr[D(\langle f'(\phi_0, 0), f'(\phi_1, 0)\rangle) = 1] + 1/2 \cdot \Pr[D(\langle f'(\phi_0, 1), f'(\phi_1, 1)\rangle) = 1], \text{ and} \\
&\Pr[D(f'(\phi_0 \vee \phi_1, 1)) = 1] = \\
&1/2 \cdot \Pr[D(\langle f'(\phi_0, 1), f'(\phi_1, 0)\rangle) = 1] + 1/2 \cdot \Pr[D(\langle f'(\phi_0, 0), f'(\phi_1, 1)\rangle) = 1],
\end{aligned}$$

and therefore

$$\begin{aligned}
&\texttt{adv}(E_0, f'(\phi_0, 0), f'(\phi_0, 1)) = \\
&|\Pr[E_0(f'(\phi_0, 0)) = 1] - \Pr[E_0(f'(\phi_0, 1)) = 1]| = \\
&|1/2 \cdot \Pr[D(\langle f'(\phi_0, 0), f'(\phi_1, 0)\rangle) = 1] - 1/2 \cdot \Pr[D(\langle f'(\phi_0, 0), f'(\phi_1, 1)\rangle) = 1] - \\
&1/2 \cdot \Pr[D(\langle f'(\phi_0, 1), f'(\phi_1, 0)\rangle) = 1] + 1/2 \cdot \Pr[D(\langle f'(\phi_0, 1), f'(\phi_1, 1)\rangle) = 1]| = \\
&|\Pr[D(f'(\phi_0 \vee \phi_1, 0)) = 1] - \Pr[D(f'(\phi_0 \vee \phi_1, 1)) = 1]| = \\
&\texttt{adv}(D, f'(\phi_0 \vee \phi_1, 0), f'(\phi_0 \vee \phi_1, 1)).
\end{aligned}$$

It remains to show that the size of C_i and E_i is as stated. Notice that each circuit takes a string y as input, and invokes D on y and y', where y' is obtained by using Construction 3.4.3 of f', with $\phi_{\bar{i}}$ and $\{x_j | j \in P(\phi_{\bar{i}})\}$ hardwired into it. Thus, the size of each of C_i and E_i is at most $|D| + |f| + |\phi_{\bar{i}}| + \sum_{j \in P(\phi_{\bar{i}})} |x_j|$.

3.7 Open Questions

We showed a V-bit protocol for any problem that has a NIC, and this protocol has soundness error $1/2$, which is inherent to public-coin black-box zero-knowledge protocols [40]. Our open question is whether the soundness error can be reduced to $1/2^n$. Indeed, the protocol of [10] achieves this for random-self reducible problems, but it does not seem to apply to problems admitting NIC.

Chapter 4

Perfect Simulation and A Complete Problem for NIPZK

The NIC framework presented in the previous chapter allowed us to study the *known problems* admitting **PZK** proofs. In this chapter we are interested in a framework that would provide means to study *all the problems* admitting **PZK** proofs. Such framework can be provided by *hard* or *complete problems*. For example, in the case of **NP**, the study of the entire class is facilitated through **NP**-complete problems [24, 58, 57]. Similarly, the **SZK**-complete problems of [77, 44] and similar characterizations for **CZK** [87] were used to prove many results about statistical and computational zero-knowledge protocols, such as equivalence between private-coin and public-coin, equivalence between honest and malicious verifier, and much more [71, 42, 86, 70, 69, 72, 73, 23]. Thus, our goal is to find complete problems for the case of perfect zero-knowledge proofs.

The issue is that, like other techniques used in the study of **SZK**, the reductions from the study of statistical zero-knowledge proofs [77, 80, 43] introduce a small error into the simulation. In the statistical setting this is not a problem because a small error is allowed. In contrast, perfect zero-knowledge protocols do not allow any error in the simulation, and therefore these reductions do not apply to the perfect setting.

We remark that the issue with the error incurred by the reductions from the statistical setting was addressed by Sahai and Vadhan [77]. However, they were only able to overcome it in simple cases (e.g., when the underlying problems have perfect completeness), or to provide *unnatural* complete problems that are defined in terms of the class itself. Their approach was to overcome difficulties in the reduction itself, whereas our approach is to consider both the reduction and the protocol, and fix them together.

4.0.1 Main results

We apply a new *error shifting technique* to the reductions and the zero-knowledge proofs from the statistical setting. Intuitively, our technique shifts to the protocol errors that would otherwise become simulation errors. This description is very loose, but we chose it because our technique can be applied in various different contexts, and in each of these contexts it takes a different form. In the case of reductions, instead of dealing with the error in the reduction itself [77], our technique shifts the error forward to the protocol, where it is no longer a simulation error. Consequently, we obtain complete and hard problems for the perfect setting.

Our results apply to both the interactive and the non-interactive model. In the *non-interactive* model, due to Blum, Feldman, and Micali [16], a common random string is available to both parties, and only one message is sent from the prover to the verifier.

We start by applying the error shifting technique to the reduction of Goldreich, Sahai, and Vadhan [43] and its non-interactive statistical zero-knowledge (**NISZK**) proof. We note that the reduction of [43] originated from the reduction of De Santis, Di Crescenzo, Persiano, and Yung [80] for **NISZK**. Consequently, we obtain the first complete problem for the class of problems possessing non-interactive perfect zero-knowledge proofs (**NIPZK**).

Theorem 4.0.1 UNIFORM *is* **NIPZK**-complete.

Informally, instances of UNIFORM are circuits similar to the **NISZK**-complete problem STATISTICAL DISTANCE FROM UNIFORM (SDU) [43], but they have an additional output bit. Ignoring this bit, we can think of YES instances of UN as circuits that represent the uniform distribution, whereas NO instance are circuits that hit only a small fraction of their range. The difference between SDU to our problem is that YES instances of UNIFORM represent the uniform distribution, whereas YES instances of SDU represent a distribution that is only statistically close to uniform. This difference is natural because it reflects the difference between perfect and statistical simulation.

In the interactive model we obtain a similar result. That is, we apply the error shifting technique to the reduction of [77], thus obtaining a hard problem for the class of problems possessing public-coin **HVPZK** proofs. Instances of our hard problem are triplets of circuits. Again, ignoring one of these circuits, our problem is a variant of STATISTICAL-DISTANCE (SD) [77]. That is, we can think of YES instances of our problem as pairs of circuits representing the same distribution, whereas instances of the reduction of [77] are circuits representing statistically close distributions. Essentially, this means that $\mathrm{SD}^{0,1/2}$ is hard for the class of problems possessing public-coin **HVPZK** proofs.

Theorem 4.0.2 IDENTICAL DISTRIBUTIONS *is hard for the class of problems possessing public-coin **HVPZK** proofs.*

We remark that a complete problem for **PZK** was given in [77], but unlike our problems, it is unnatural, and is defined in terms of the class itself.

4.0.2 Organization

Our completeness result is given in Section 4.2, and our hard problem is described in Section 4.3. In Section 4.4 we show applications of the error shifting technique, such as closure of **NIPZK** under the OR operator, and the equivalence between notions of zero-knowledge with respect to simulators that fail. We start with definitions.

4.1 Definitions

In this section we define protocols, proofs, and zero-knowledge in the non-interactive model. These definitions are analogous to those given in Chapter 2. We start with non-interactive protocols, where a common random string is available to both parties, and the prover sends only one message. Formally,

Definition 4.1.1 (Non-interactive protocols) *A* non-interactive protocol $\langle b, P, V\rangle$ *is a triplet (or simply a pair* $\langle P, V\rangle$*, making* b *implicit), where* P *and* V *are functions, and* $b \in \mathbb{N}$*. We denote by* r_P *the random inputs to* P*. The* interaction *between* P *and* V *on* common input x *is the following random process.*

1. *Uniformly choose* r_P*, and choose a* common random string $r_I \in \{0,1\}^{|x|^b}$.
2. *Let* $\pi = P(x, r_I; r_P)$*, and let* $m = V(x, r_I, \pi)$*.*
3. *Output* $\langle x, r_I, \pi, m\rangle$*.*

We call $\langle P, V\rangle(x) \overset{def}{=} \langle x, r_I, \pi\rangle$ *the* view *of* V *on* x*. We say that* V accepts x *(respectively,* rejects x*) if* $m = \mathtt{accept}$ *(respectively,* $m = \mathtt{reject}$*).*

Definition 4.1.1 considers a deterministic verifier V, but it is equivalent to a the definition that considers a probabilistic V (the idea is to let V use a portion of r_I as its randomness, and since this may allow the prover to cheat, we require P to send one proof that makes V accept on many portions of r_I [56]).

We continue to the definition of non-interactive proofs [16, 17]. Like in interactive proofs, we require that the verifier accept `YES` instances and reject `NO` instances, but now the probabilities are also over the choice of the common random string. Formally,

Definition 4.1.2 (Non-interactive proofs) *A non-interactive protocol* $\langle b, P, V\rangle$ *is a* non-interactive proof *for a problem* Π *if there is* $a \in \mathbb{N}$ *and* $c(n), s(n) : \mathbb{N} \to [0,1]$ *such that* $1 - c(n) \geq s(n) + 1/n^a$ *for any* n*, and the following conditions hold.*

- *Efficiency:* V *runs in time polynomial in* $|x|$*.*
- *Completeness:* V *accepts all* $x \in \Pi_{\mathrm{Y}}$ *with probability at least* $1 - c(|x|)$ *over* r_I *and* r_P*.*
- *Soundness:* $\Pr_{r_I}[V(x, r_I, P^*(x, r_I)) = \mathtt{accept}] \leq s(|x|)$ *for any function* P^* *and any* $x \in \Pi_{\mathrm{N}}$*.*

The function c *is called the* completeness error*, and the function* s *is called the* soundness error*. We say that* $\langle P, V\rangle$ *has* perfect completeness *if* $c \equiv 0$*.*

Alternatively, by letting V choose the common random string, we can view non-interactive proofs as 2-round public-coin proofs. Finally, we define zero-knowledge proofs in the non-interactive model.

Definition 4.1.3 (Non-interactive, zero-knowledge protocols) *A non-interactive protocol $\langle P, V\rangle$ is* perfect zero-knowledge (**NIPZK**) *for a problem $\Pi = \langle \Pi_{\mathrm{Y}}, \Pi_{\mathrm{N}}\rangle$ if there is a probabilistic, polynomial-time Turing machine S, called the* simulator*, such that the ensembles*

$$\{\langle P, V\rangle(x)\}_{x\in\Pi_{\mathrm{Y}}} \quad \text{and} \quad \{S(x)\}_{x\in\Pi_{\mathrm{Y}}}$$

are statistically identical.

If these ensembles are statistically indistinguishable, then $\langle P, V\rangle$ is a non-interactive statistical zero-knowledge (**NISZK**) *protocol for* Π. *Similarly, if the ensembles are computationally indistinguishable, then $\langle P, V\rangle$ is non-interactive* computational zero-knowledge (**NICZK**) *protocol for* Π.

The class of problems possessing **NIPZK** *(respectively,* **NISZK**, **NICZK***) protocols is also denoted* **NIPZK** *(respectively,* **NISZK**, **NICZK***).*

Following the above remark, non-interactive zero-knowledge proofs are public-coin, 2-round honest-verifier zero-knowledge proofs, and therefore they can be transformed into constant-round, public-coin zero-knowledge proofs using the transformation of Damgård, Goldreich, and Micali [28].

4.2 Perfect Simulation and A Complete Problem for NIPZK

In this section we introduce the *error shifting technique*. Using this technique we modify the reduction of [43], hence obtaining a **NIPZK**-complete problem.

Starting with some background, we give the definition of STATISTICAL DISTANCE FROM UNIFORM (SDU), the **NISZK**-complete problem of [43]. Instances of this problem are circuits. These circuits are treated as distributions, under the convention that the input to the circuit is uniformly distributed. Specifically, YES instances are circuits representing a distribution that is close to uniform, and NO instances are circuits representing a distribution that is far from uniform.

Definition 4.2.1 *Define* $\mathrm{SDU} \overset{def}{=} \langle \mathrm{SDU}_Y, \mathrm{SDU}_N\rangle$ *as*

$$\mathrm{SDU}_Y = \{X \mid \Delta(X, U_n) < 1/n\}, \text{ and}$$
$$\mathrm{SDU}_N = \{X \mid \Delta(X, U_n) > 1 - 1/n\},$$

where X is a circuit with n output bits, and U_n is the uniform distribution on $\{0,1\}^n$.

We informally describe the reduction of [43] to SDU. This reduction originated from the work of [80]. Given a **NISZK** problem Π, this reduction maps instances x of Π to circuits X of SDU. The circuit uses the simulator S from the proof of Π. Specifically, X executes $S(x)$, and obtains a transcript. This transcript contains a simulated message of the prover, and a simulated reference string. If the verifier accepts in this transcript, then X outputs the simulated reference string. Otherwise, X outputs the all-zero string. Intuitively, this reduction works because if x is a $\texttt{YES}$ instance, then the simulated reference string is almost uniformly distributed, and thus X is a $\texttt{YES}$ instance of SDU. Conversely, if x is a $\texttt{NO}$ instance, then the verifier rejects on most reference strings, and thus X is a $\texttt{NO}$ instance of SDU.

The issue with the reduction of [43]. When we apply the above reduction to **NIPZK** problems, it is natural that we should get a **NIPZK**-complete problem whose instances are circuits that represents the uniform distribution. This is so because the circuit X outputs the simulated reference string, and when the simulation is perfect, this string is uniformly distributed. Indeed, if we apply the above reduction to **NIPZK** problems that have perfect completeness, then the verifier will accept, and thus we will get a circuit X that represents the uniform distribution. However, if the underlying problem does not have perfect completeness, then the distribution represented by X will be skewed. This will cause problems later, when we try to construct a proof system and a simulator for our complete problem. Hence, this reduction does not apply to **NIPZK**.

To overcome the above issue, instead of working only with the reduction to SDU, our idea is to modify both the reduction and the proof system for SDU at the same time.

The Error Shifting Technique. In its most general form, *the* error shifting technique *shifts into the protocol errors that would otherwise become simulation errors.* This description is a very loose, but we chose it because our technique can be applied in various different contexts, and in each of these contexts it takes a different form. However, the following application will clarify our technique.

► **The first step of the error shifting technique** is to identify where the simulation error comes from, and then isolate it. In our case, the error comes from the reduction: if the verifier rejects, then the circuit X does not represent the uniform distribution. Thus, the error comes from the completeness error of the underlying problem. To separate this error, we add an extra output bit to the circuit X. That is, X executes the simulator, and it outputs the simulated reference string followed by an extra bit. This bit takes the value 1 if the verifier accepts, and 0 if the verifier rejects.

► **The second step of the error shifting technique** is to shift the error forward, to the completeness or the soundness error of the protocol. In our case, from the circuit X to the protocol of our complete problem. This step is not trivial because we cannot just use the protocol of [43] for SDU. Specifically, in this protocol the prover sends a string r, and the verifier accepts if $X(r)$ equals the reference string. If we use this idea in our case, then we will get a simulation error. Thus, we modify this protocol by starting with the simulator, and constructing the prover based on the simulator. Informally, the simulator samples the circuit X, and the verifier accepts if the extra bit in this sample is 1. The prover simply mimics the simulator. This shows that the error was shifted from X to the completeness error (of a new protocol).

The above reduction yields our **NIPZK**-complete problem UNIFORM. A formal description of the above reduction and our proof system is given in the next section.

4.2.1 A Complete Problem for NIPZK

In this section we formalize the intuition given in the previous section, thus proving our first result.

We start with the definition of UNIFORM (UN). Recall that when we applied the error shifting technique we got circuits X with an extra output bit. We use the convention that $n+1$ denotes the number of output bits of X. We need the following notation.

- T_X is the set of outputs of X that end with a 1. Formally, $T_X \overset{\text{def}}{=} \{x | \exists r\ X(r) = x$, and the suffix of x is $1\}$. As we shall see, the soundness and completeness properties will imply that the size of T_X is large for `YES` instances of UN, and small for `NO` instances of UN.

- X' is the distribution on the first n bits that X outputs. That is, X' is obtained from X by taking a random sample of X, and then outputting the first n bits. As we shall see, the zero-knowledge property will imply that if X is a `YES` instance of UN, then X' is the uniform distribution on $\{0,1\}^n$.

Now, letting X be a circuit with $n+1$ output bits, we say that X is *β-negative* if $|T_X| \le \beta \cdot 2^n$. That is, T_X is small, and contains at most $\beta \cdot 2^n$ strings. We say that X is *α-positive* if X' is the uniform distribution on $\{0,1\}^n$ and $\Pr_{x \leftarrow X}[x \in T_X] \ge \alpha$. This implies that T_X is large, and contains at least $\alpha \cdot 2^n$ strings.

Definition 4.2.2 *The problem* UNIFORM *is defined as* $\mathrm{UN} \overset{\text{def}}{=} \langle \mathrm{UN_Y}, \mathrm{UN_N} \rangle$, *where*

$$\begin{aligned} \mathrm{UN_Y} &= \{X | X \textit{ is } 2/3-\textit{positive}\}, \textit{ and} \\ \mathrm{UN_N} &= \{X | X \textit{ is } 1/3-\textit{negative}\}. \end{aligned}$$

To prove that UN is **NIPZK**-complete we first show that the reduction from the previous section reduces every **NIPZK** problem to UN.

Lemma 4.2.3 UN *is* **NIPZK**-hard.

Proof: Let $\Pi = \langle \Pi_\mathrm{Y}, \Pi_\mathrm{N} \rangle$ be a **NIPZK** problem. Fix a non-interactive protocol $\langle P, V \rangle$ for Π with completeness and soundness errors $1/3$. Let r_I denote the common reference string in $\langle P, V \rangle$, and fix i such that $|r_I| = |x|^i$ for any $x \in \Pi_\mathrm{Y} \cup \Pi_\mathrm{N}$. Fix a simulator S for $\langle P, V \rangle$. Since S is efficient, we can fix an efficient transformation t and an integer ℓ such that on input $x \in \Pi_\mathrm{Y} \cup \Pi_\mathrm{N}$ the output of $t(x)$ is a circuit S' that executes S on inputs x and randomness r_S of length $|x|^\ell$. That is, $t(x) = S'$, and on input a string r_S of length $|x|^\ell$ the output of $S'(r_S)$ is the output of $S(x; r_S)$.

We show that Π Karp reduces to UN. That is, we define a polynomial-time Turing machine that on input $x \in \Pi_\mathrm{Y} \cup \Pi_\mathrm{N}$ outputs a circuit X such that if $x \in \Pi_\mathrm{Y}$, then $X \in \mathrm{UN_Y}$, and if $x \in \Pi_\mathrm{N}$, then $X \in \mathrm{UN_N}$. The circuit $X : \{0,1\}^{|x|^\ell} \to \{0,1\}^{|x|^i+1}$ carries out the following computation.

- Let r_S be the $|x|^\ell$-bit input to X, and let $S' = t(x)$. Execute $S'(r_S)$, and obtain $S(x; r_S) = \langle x, r'_I, m' \rangle$.

- If $V(x, r'_I, m') = \texttt{accept}$, then output the string r'_I1 (i.e., the concatenation of r'_I and 1). Otherwise, output r'_I0.

Now we analyze our reduction. Let $x \in \Pi_\mathrm{Y}$, and let X be the output of the above reduction on x. We show that X is $2/3$-positive. Consider the distribution on the output $\langle x, r'_I, m'\rangle$ of $S(x)$. Since $S(x)$ and $\langle P, V\rangle(x)$ are identically distributed, r'_I is uniformly distributed. Thus, X' (i.e., the distribution on the first $|x|^i$ output bits of X) is uniformly distributed. It remains to show that $\Pr[X \in T_X] \geq 2/3$. This immediately follows from the perfect zero-knowledge and completeness properties of $\langle P, V\rangle$. That is, the output of S is identically distributed to $\langle P, V\rangle(x)$, and V accepts in $\langle P, V\rangle$ with probability at least $2/3$.

Let $x \in \Pi_\mathrm{N}$, and let X be the output of the above reduction on x. We show that X is $1/3$-negative. Assume towards contradiction that X is β-negative for some $\beta < 1/3$. We define a prover P^* that behaves as follows on CRS r_I. If $r_I1 \in T_X$, then there is an input r_S to X such that $X(r_S) = r_I1$. By the construction of X, there is randomness r_S for the simulator such that $S(x; r_S) = \langle x, r_I, m'\rangle$, and $V(x, r_I, m') = 1$. In this case P^* sends r_S to V. If $r_I1 \notin T_X$, then P^* fails. Notice that P^* makes V accept on any r_I such that $r_I1 \in T_X$. Since $|T_X| > 2^{|x|^i}/3$, and since r_I is uniformly chosen in $\langle P^*, V\rangle$, the probability that $r_I1 \in T_X$ is strictly greater than $1/3$. Thus, V accepts in $\langle P^*, V\rangle(x)$ with probability strictly greater than $1/3$, and contradiction to the soundness error of $\langle P, V\rangle$. Hence, X is $1/3$-negative. $\square$

To prove Theorem 4.0.1 it remains to give a **NIPZK** proof for UN.

Lemma 4.2.4 UN *has a* **NIPZK** *proof with a deterministic verifier.*

Proof: We start with our non-interactive proof for UN. This proof is based on our simulator, which we describe later. On input $X : \{0,1\}^\ell \to \{0,1\}^{n+1}$ and common reference string $r_I \in \{0,1\}^n$ the prover P picks z according to the distribution X such that the n-bit prefix of z equals r_I. Such a z exists because X' (i.e., the distribution on the first n bits of X) is the uniform distribution when $X \in \mathrm{UN}_\mathrm{Y}$. The prover uniformly picks $r \in X^{-1}(z)$, and sends r to the verifier V. The deterministic verifier accepts if $X(r) = r_I1$, and rejects otherwise. Our prover is based on the following simulator. Let S be a probabilistic, polynomial-time Turing machine that on input X uniformly picks $r' \in \{0,1\}^\ell$, and computes $z' = X(r')$. The simulator assigns the n bit prefix of z' to r'_I (i.e., the simulated reference string), and outputs $\langle X, r'_I, r'\rangle$.

Let $X \in \Pi_\mathrm{Y}$. We show that S perfectly simulates $\langle P, V\rangle$. Consider the distribution $S(X)$ on simulated transcripts $\langle X, r'_I, r'\rangle$, and the distribution $\langle P, V\rangle(X)$ on the view $\langle X, r_I, r\rangle$ of V. Since X' is uniformly distributed over $\{0,1\}^n$, the string r'_I obtained by the simulator is uniformly distributed over $\{0,1\}^n$. Since r_I is uniformly distributed, r'_I and r_I are identically distributed. It remains to show that r and r' are identically distributed conditioned on $r_I = r'_I$. For each $y \in \{0,1\}^n$, we define B_y to be the set of all strings $\hat{r}$ for which the prefix of $X(\hat{r})$ is y. Now, for any simulated reference string r'_I, the randomness r' chosen by the simulator is uniformly distributed in $B_{r'_I}$. Similarly, for any reference string r_I the message of the prover is a string r chosen uniformly from B_{r_I}. Hence, conditioned on $r_I = r'_I$, the strings r and r' are identically distributed. We conclude that $S(X)$ and $\langle P, V\rangle(X)$ are identically distributed for any $X \in \Pi_\mathrm{Y}$.

Turning our attention to the completeness property, we show that V accepts X with probability at least $2/3$. By the zero-knowledge property, the output $\langle X, r'_I, r'\rangle$ of $S(X)$ is identically distributed to the view $\langle X, r_I, r\rangle$ of V on X. Thus, it is enough to show that when choosing a transcript $\langle X, r'_I, r'\rangle$ according to $S(x)$ the probability that $V(X, r'_I, r') = 1$ is at least $2/3$. Since S uniformly chooses r', and since X is $2/3$-positive, the probability that $X(r) \in T_X$ is at least $2/3$. Thus, the probability that the suffix of $X(r)$ is 1 is at least $2/3$. Hence, V accepts X with probability at least $2/3$.

The soundness property follows easily. Let $X \in \mathrm{UN_N}$. Since X is $1/3$-negative, $|T_X| \leq 1/3 \cdot 2^n$. Since r_I is uniformly distributed, the probability that $r_I 1 \in T_X$ is at most $1/3$. Hence, if $X \in \mathrm{UN_N}$, then V accepts X with probability at most $1/3$. □

4.3 A Hard Problem for Public-Coin PZK Proofs

In this section we use the error shifting technique to modify the reduction of [77] for public-coin **HVSZK** proofs. Hence, we obtain a hard problem for the class of problems possessing public-coin **HVPZK** proofs. We do not know if this problem is also complete, but we remark that even the similar problem $\mathrm{SD}^{0,1/2}$ is not known to admit a **HVPZK** proof [77]. We start with motivation.

The reduction of [77] originated from the works of Fortnow and Aiello and Håstad [37, 3]. Informally, given a problem Π that has a public-coin **HVSZK** proof, the reduction maps instances x of Π to pairs of circuits $\langle X_0, X_1\rangle$. The circuits X_0 and X_1 are statistically close when x is a YES instance of Π, and statistically far when x is a NO instance of Π.

The issue with this reduction is that it does not apply to the perfect setting. Specifically, when we apply it to YES instances of a problem that has a public-coin **HVPZK** proof, we get a pair of circuits $\langle X_0, X_1\rangle$ that are only statistically close, but not identically distributed. This is unnatural because the closeness between X_0 and X_1 reflects the closeness of the simulation. Thus, in the perfect setting we expect X_0 and X_1 to be *identically distributed*, as in $\mathrm{SD}^{0,1/2}$ (this problem is a variant of STATISTICAL-DISTANCE [77], which we defined in Section 2.4.)

Definition 4.3.1 [77] The problem $\mathrm{SD}^{0,1/2}$ is the pair $\langle \mathrm{SD}^{0,1/2}_{\mathrm{Y}}, \mathrm{SD}^{0,1/2}_{\mathrm{N}}\rangle$, where

$$\mathrm{SD}^{0,1/2}_{\mathrm{Y}} = \{\langle X_0, X_1\rangle |\ \Delta(X_0, X_1) = 0\}, \text{ and}$$
$$\mathrm{SD}^{0,1/2}_{\mathrm{N}} = \{\langle X_0, X_1\rangle |\ \Delta(X_0, X_1) \geq 1/2\}.$$

Sahai and Vadhan [77] were aware of this issue, and they addressed it by directly calculating the errors of the underlying problem. However, their technique applies only in certain cases (for example, when the underlying problem has a proof with perfect completeness). In the next section we will show how to overcome this issue by using the error shifting technique. Essentially, we obtain a hard problem where YES instances are pairs of circuits representing identical distributions, and NO instances are circuits representing statistically far distributions. Formally, our hard problem is as follows.

Definition 4.3.2 *The problem* IDENTICAL DISTRIBUTIONS *is* ID $\overset{def}{=} \langle \mathrm{ID_Y}, \mathrm{ID_N} \rangle$*, where*

$$\mathrm{ID_Y} = \{\langle X_0, X_1, Z\rangle \mid \Delta(X_0, X_1) = 0 \ \textit{ and } \ \Pr[Z = 1] \geq 2/3\}, \textit{ and}$$
$$\mathrm{ID_N} = \{\langle X_0, X_1, Z\rangle \mid \Delta(X_0, X_1) \geq 1/2 \ \textit{ or } \ \Pr[Z = 1] \leq 1/3\}.$$

4.3.1 Modifying the Reductions for Public-Coin HVSZK Proofs

In this section we show that ID is hard for the class of problems admitting public-coin **HVPZK** proofs. Using the error shifting technique we show that, essentially, we can treat $\mathrm{SD}^{0,1/2}$ as a hard problem for this class. Starting with some background, we describe the reduction of [77].

Notation. Let $\langle P, V\rangle$ be a public-coin **HVPZK** proof for a problem Π with a simulator S. Given a string x we use $v \overset{\text{def}}{=} v(|x|)$ to denote the number of rounds in the interaction between P and V on input x. That is, in round i the prover P sends m_i and V replies with a random string r_i, until P sends its last message m_v, and V accepts or rejects. We denote the output of $S(x)$ by $\langle x, m_1, r_1, \ldots, m_v\rangle$.

The reduction of [77] maps instances x of Π to pairs of circuits $\langle X_0', X_1'\rangle$. These circuits are constructed from the circuits X_i and Y_i, defined as follows. The circuit X_i chooses randomness, executes $S(x)$ using this randomness, and outputs the simulated transcript, truncated at the i-th round. That is, X_i obtains $\langle x, m_1, r_1, \ldots, m_v\rangle$, and outputs $\langle m_1, r_1, \ldots, m_i, r_i\rangle$. The circuit Y_i is defined exactly the same, except that it replaces r_i with a truly random string r_i'.

- $X_i(r)$: execute $S(x; r)$ to obtain $\langle x, m_1, r_1, \ldots, m_v\rangle$. Output $\langle m_1, r_1, \ldots, m_i, r_i\rangle$.
- $Y_i(r, r_i')$: execute $S(x; r)$ to obtain $\langle x, m_1, r_1, \ldots, m_v\rangle$. Output $\langle m_1, r_1, \ldots, m_i, r_i'\rangle$.

Notice that X_i and Y_i represent the same distribution when x is a `YES` instance. This is so because $S(x)$ perfectly simulates the view of the verifier, and therefore r_i is uniformly distributed, just like r_i'. We define $X = X_1 \otimes \cdots \otimes X_v$. That is, X executes all the circuits X_i and outputs the concatenation of their outputs. Similarly, we define $Y = Y_1 \otimes \cdots \otimes Y_v$. Again, X and Y are identically distributed when x is a `YES` instance. Now, the pair $\langle X_0', X_1'\rangle$ is defined from $\langle X, Y\rangle$ as follows. The circuit X_1' outputs 1 followed by the output of Y. The circuit X_0' outputs the output of Z followed by the output of X, where Z is the circuit that obtains polynomially many transcripts (by executing $S(x)$, each time with a fresh random string), and outputs 1 if the verifier accepts in the majority of these transcripts, and 0 otherwise.

The issue with the reduction of [77]. The above reduction does not apply to the perfect setting (except for the case where $\langle P, V\rangle$ have perfect completeness). This is so because there is a non-zero probability that Z will output 0, in which case X_0' and X_1' will not represent the same distribution. To overcome this issue we use the error shifting technique in two steps, just like we did in the previous section. Our goal is to show that, essentially, $\mathrm{SD}^{0,1/2}$ is hard for the class of problems admitting public-coin **HVPZK** proofs.

As a first step, we separate the error that the circuit Z incurs. Thus, instead of including Z in the circuits X_0' and X_1', our reduction simply maps an instance x of Π to the triplet $\langle X, Y, Z\rangle$. By the analysis from [77], if x is a YES instance, then X and Y are identically distributed, and Z outputs 1 with high probability. Such a triplet is a YES instance of our hard problem. Similarly, if x is a NO instance, then either X and Y are statistically far, or Z outputs 0 with a high probability. Such a triplet is a NO instance of our hard problem. The following lemma shows that IDENTICAL DISTRIBUTIONS (ID) is hard for the class of problems admitting public-coin **HVPZK** proofs.

Lemma 4.3.3 (Implicit in [77]) *For any problem $\Pi = \langle \Pi_{\mathrm{Y}}, \Pi_{\mathrm{N}}\rangle$ possessing a public-coin **HVPZK** proof there is a Karp reduction mapping strings x to circuits $\langle X, Y, Z\rangle$ with the following properties.*

- *If $x \in \Pi_Y$, then $\Delta(X,Y) = 0$ and $\Pr[Z = 1] \geq 2/3$.*
- *If $x \in \Pi_N$, then $\Delta(X,Y) \geq 1/2$ or $\Pr[Z = 1] \leq 1/3$.*

Indeed, instances of ID are triplets of circuits $\langle X, Y, Z\rangle$, as opposed to pairs $\langle X, Y\rangle$. Our goal is to show that ID and $\mathrm{SD}^{0,1/2}$ are essentially the same. Hence, we continue to our second step.

Recall that the second step of the error shifting technique is to shift the error forward to the protocol. However, $\mathrm{SD}^{0,1/2}$ is not known to have a **PZK** proof. Thus, our second step is to modify *any* **PZK** protocol $\langle P, V\rangle$ for $\mathrm{SD}^{0,1/2}$ into a **PZK** protocol $\langle P, V'\rangle$ for ID. That is, we take an arbitrary protocol $\langle P, V\rangle$ for $\mathrm{SD}^{0,1/2}$, and then we show that even if the input to this protocol is an instance $\langle X, Y, Z\rangle$ of ID (instead of a pair $\langle X, Y\rangle$), then the behavior of P and the modified verifier V' on input $\langle X, Y, Z\rangle$ is identical to the behavior of $\langle P, V\rangle$ on input $\langle X, Y\rangle$. This will show that the two problems are essentially the same, and therefore we will be done.

Our modification is as follows. On input $\langle X, Y, Z\rangle$ the first step of the modified verifier V' is to estimate the value of $\Pr[Z = 1]$, and reject if this value is at most $1/3$. If V' did not reject, then P and V' execute $\langle P, V\rangle$ on input $\langle X, Y\rangle$. This modification is a part of the error shifting technique because we shift the error from the circuit Z into an arbitrary protocol $\langle P, V\rangle$ for $\mathrm{SD}^{0,1/2}$.

We analyze the modified protocol $\langle P, V'\rangle$ for our hard problem. We observe that V' is very unlikely to reject if $\Pr[Z = 1] \geq 2/3$. We also observe that if the protocol continues, then either $\langle X, Y, Z\rangle$ is a YES instance of our hard problem and $\Delta(X,Y) = 0$, or $\langle X, Y, Z\rangle$ is a NO instance of our hard problem and $\Delta(X,Y) \geq 1/2$. Thus, in this case the behavior of P and V' on instances of our hard problem is identical to the behavior of P and V on instances of $\mathrm{SD}^{0,1/2}$. This shows that although we did not prove that $\mathrm{SD}^{0,1/2}$ is hard for the class of problems admitting public-coin **HVPZK** proofs, it can be treated as such (because any protocol for this problem can be immediately modified to a protocol with the same properties for ID).

4.4 Applications

In this section we show two applications of our results. The first one uses the error shifting technique to show an equivalence between notions of zero-knowledge with respect to simulators that fail. The second shows that under certain restrictions **NIPZK** is closed under the `OR` operator. Notice that even in statistical setting, where we have more techniques to work with, it is not clear how to prove (or disprove) this result.[1]

4.4.1 Obtaining Simulators That Do Not Fail

We use the error shifting technique to show that the notion of honest-verifier zero-knowledge where the simulator is allowed to fail is equivalent to the one where it is not allowed to fail. This holds in both the interactive and the non-interactive models, and regardless of whether the simulator runs in strict or expected polynomial-time.

Starting with background, we recall that the notion of perfect zero-knowledge requires that the view of the verifier be identically distributed to the output of the simulator [46]. Later, this notion was relaxed by allowing the simulator to output `fail` with probability at most $1/2$, and requiring that, conditioned on the output of the simulator not being `fail`, it be identically distributed to the view of the verifier [28].

A known trick to remove the `fail` output is to execute the simulator for $|x|$ times (where x is the input to the simulator), and output the first transcript, or `fail` if the simulator failed in all $|x|$ executions [39]. This works for statistical and computational zero-knowledge, but not for perfect zero-knowledge. Notice that in all of these cases we actually introduce an extra error into the simulation, and we do not understand why. Furthermore, despite the fact that important problems have **PZK** proofs (e.g., GRAPH-ISOMORPHISM, QUADRATIC-RESIDUOUSITY [46, 41, 84]), all of these proofs have a simulator that outputs `fail` with probability $1/2$. Now we fix this issue.

The transformation. Let $\langle P, V\rangle$ be a **HVPZK** proof for a problem Π, and let S be a simulator for $\langle P, V\rangle$. Notice that S may fail with some probability. We use the error shifting technique to obtain a simulator S' that does not fail.

Recall that the error shifting technique is applied in two steps: we need to find where the error is coming from, and then we shift it forward. For the first step, we observe that when S outputs `fail`, the verifier V actually learns that S failed. This is something that V does not learn from the prover P (because transcripts between P and V are never of the form `fail`). Intuitively, the error comes from the fact that P is not showing V that $S(x)$ may output fail with some probability. We are done with the first step. In the second step we shift this error forward by letting P teach V that $S(x)$ may output `fail`. That is, on input x, the new prover P' executes $S(x)$ for $|x|$ times, and if $S(x) = \texttt{fail}$ in all of these executions, then P' outputs `fail`. Otherwise, P' behaves like P. In other words, we shifted the error from the simulation to the protocol. Notice that the execution of $S(x)$ by P' may involve invoking V, but since the verifier is honest, its code can be incorporated into P'.

The new simulator S' simply executes S, and if all executions failed, then it behaves just like P'. Namely, it outputs the transcript $\langle x, \texttt{fail}; r_V\rangle$, where r_V is the randomness of V. Otherwise, S' outputs a simulated transcript of S. Notice that we increased the completeness error by $1/2^n$, but by executing

[1] [80] claimed that **NISZK** is closed under the `OR` operator, but this claim has been retracted.

$S(x)$ polynomially many times, the probability that P' will fail can be made extremely small. We conclude that $\langle P', V\rangle$ is a **PZK** proof for Π with a simulator S' that never fails.

4.4.2 Under Certain Restrictions NIPZK is Closed Under the OR Operator

We use our **NIPZK**-complete problem to show that under certain restrictions **NIPZK** is closed under the OR operator. We remark that these restrictions are severe, but our goal is to show the usefulness of our complete problem, rather than proving a closure result (in fact, even with these restrictions it is hard to see how to prove this result).

Motivation. We want to construct a **NIPZK** proof where the prover and the verifier are given two instances x and y of some problem $\Pi \in$ **NIPZK**, and the verifier accepts only if either x or y are YES instances of Π. Since we now have a **NIPZK**-complete problem, we can construct a protocol where the prover and the verifier reduce x and y to circuits X and Y, respectively, and then work with these circuits.

A natural approach to design our protocol is to ask what is the difference between YES and NO instances of UN, and then, based on this difference, to design a protocol and a simulator. As we saw, instances of UN differ in their number of output strings that end with a 1. That is, $|T_X| + |T_Y|$ is large if either X or Y is a YES instance, and small if both X and Y are NO instances. Thus, it seems that we should use lower bound protocols [45]. However, we avoid using these protocols because they incur error into the simulation, and we do not know how to remedy this problem.

Thus, we take a different approach. Instead of focusing on the difference between YES and NO instances, we focus on the simulation. That is, instead of starting with the protocol, taking care of completeness and soundness, we start with the simulator, taking care of perfect zero-knowledge. Indeed, this approach is implicit in Section 4.2, where we first modified the simulator, and then modified the prover to mimic the simulator. This approach has the advantage that we retain perfect simulation, but on the other hand we are forced to make restrictions in order to guarantee completeness and soundness.

The protocol. Recall that the prover and the verifier are given instances X and Y of UN, and the verifier should accept if $X \in \mathrm{UN_Y}$ or $Y \in \mathrm{UN_Y}$. As usual, we use $n+1$ to denote the number of output bits of X and Y. Since the main obstacle is how to achieve perfect simulation, we start with the zero-knowledge property. That is, we start with the simulator, and then we design the protocol based on the simulator.

Consider a simulator that uniformly picks r_X and r_Y, and computes $z = X(r_X) \oplus Y(r_Y)$. The simulator may not know which of X or Y is a YES instance of UN. However, the n-bit prefix of z is uniformly distributed because either X' or Y' represent the uniform distribution. This observation allows us to use the n-bit prefix of z as the simulated reference string.

Our simulator informs the following protocol: on reference string r_I the prover sends r_X and r_Y to the verifier such that the n-bit prefix $X(r_X) \oplus Y(r_Y)$ equals r_I. The issue with this protocol is that we need to make two restrictions in order to prove completeness and soundness.

Achieving completeness. Suppose that the verifier accepts only if the last bit of both $X(r_X)$ and $Y(r_Y)$ is 1. This works when both circuits X and Y are `YES` instances of UN. However, if one of the circuits is a `NO` instance of UN, then it is possible that all the strings outputted by this circuit end with a 0 (e.g, for any r_X the suffix of $X(r_X)$ is 0), and this will make V reject.

Since we do not know how to overcome this issue without introducing error into the simulation, we add the restriction that instances of $\mathrm{PU_Y}$ be 1-positive. That is, for any circuit $Z \in \mathrm{PU_Y}$, all the strings that Z outputs have 1 as the rightmost bit. Intuitively, this restriction helps the simulator in identifying `NO` instances. For example, if a sample of X ends with a 0, then X must be a `NO` instance. However, notice that X could be a `NO` instance and still have outputs that end with a 1. Thus, this help is limited.

We redefine the simulator based on the above restriction. As before, the simulator uniformly picks r_X and r_Y, computes $z = X(r_X) \oplus Y(r_Y)$, and if both $X(r_X)$ and $Y(r_Y)$ end with a 1, then the simulator uses the n-bit prefix of z to simulate the reference string. Otherwise, one of the samples ends with a 0. For example, suppose that $X(r_X)$ ends with a 0. This implies that Y is a `YES` instance. Hence, the simulator uses the n bit prefix of $Y(r_x)$ to simulate the reference string. Similarly, we redefine the verifier. That is, when the verifier receives $\langle r_X, r_Y\rangle$ from the prover, it only checks that the n-bit prefix of $Y(r_Y)$ equals to the reference string, and that $Y(r_Y)$ ends with a 1.

Achieving soundness. Notice that even when both X and Y are `NO` instances, there could be many combinations for $X(r_X) \oplus X(r_Y)$. That is, for most reference strings r_I a cheating prover may find r_X and r_Y such that the n bit prefix of $X(r_X) \oplus Y(r_Y)$ equals r_I, and both $X(r_X)$ and $Y(r_Y)$ end with a 1. This compromises the soundness property. Since we do not know how to overcome this issue without introducing error into the simulation, we restrict the number of such pairs.

Discussion. We used our **NIPZK**-complete problem to show that under certain restrictions **NIPZK** is closed under the `OR` operator. Indeed, we added severe restrictions to retain perfect simulation, but without our complete problem it is not clear how to prove this result (even with these restrictions). Thus, we interpret these restrictions as evidence that in the perfect setting there are few techniques to work with. Recall that even in the statistical setting, where we have more techniques to work with, such closure result is not known.

Lemma 4.4.1 *Let Π be a* **NIPZK** *problem with a proof $\langle P', V'\rangle$, and let $c \in \mathbb{N}$ such that on input of length n the reference string is of length n^c. If $\langle c, P', V'\rangle$ has perfect completeness and soundness error $2^{1-n^c/2}$, then $\Pi \vee \Pi$ has a* **NIPZK** *proof with perfect completeness, and soundness error $1/3$.*

Proof: Let $\langle x_0, x_1\rangle$ such that $x_i \in \Pi_\mathrm{Y} \cup \Pi_\mathrm{N}$ for each $i \in \{0, 1\}$, and let $n = |x_0|$. We start with the case where $|x_0| = |x_1|$ because when we reduce x_0 and x_1 to UN we get circuits whose output length is equal. As we will see, the general case follows easily using the same proof.

We construct a **NIPZK** protocol $\langle P, V\rangle$ for $\Pi \vee \Pi$. Initially, P sets $i = 0$ if both x_0 and x_1 are in Π_Y. Otherwise, there is a unique i such that $x_i \in \Pi_\mathrm{N}$, and P fixes this i. In addition, for each $i \in \{0, 1\}$ both P and V reduce x_i to an instance X_i of UN.

Recall that $\langle c, P', V' \rangle$ is a **NIPZK** proof for Π such that on input of length n the reference string is of length n^c. By the properties of the reduction to UN, for each $i \in \{0,1\}$ the circuit X_i has $n^c + 1$ output gates and the following properties hold. If $x_i \in \Pi_{\mathrm{Y}}$, then X'_i is the uniform distribution on $\{0,1\}^{n^c}$, and samples of X_i end with a 1. If $x_i \in \Pi_{\mathrm{N}}$, then $|T_{X_i}| \leq 2^{-(n^c/2+1)} \cdot 2^{n^c} = 2^{n^c/2-1}$.

The protocol proceeds as follows. Recall that P initially computes i. Thus, the first step of P is to uniformly choose a string r_i, and assign y the output of $X_i(r_i)$, excluding the rightmost bit. On reference string r_I, if $X_i(r_i) = y0$, then P uniformly chooses $r_{\bar{i}} \in X_{\bar{i}}^{-1}(r_I 1)$, and sends $\langle r_0, r_1 \rangle$ to V. Otherwise, $X_i(r_i) = y1$, in which case P uniformly chooses $r_{\bar{i}} \in X_{\bar{i}}^{-1}(y1 \oplus r_I 0)$, and sends $\langle r_0, r_1 \rangle$ to V. The verifier accepts if $\langle r_0, r_1 \rangle$ are correctly computed. Namely, V computes $X_0(r_0)$ and $X_1(r_1)$, and if there is $i \in \{0,1\}$ such that $X_i(r_i)$ ends with a 0 and $X_{\bar{i}}(r_{\bar{i}}) = r_I 1$, then V accepts. Otherwise, if $X_0(r_0) \oplus X_1(r_1) = r_I 0$ (that is, both $X_0(r_0)$ and $X_1(r_1)$ end with a 1), then V accepts. Otherwise, V rejects.

The completeness property of $\langle P, V \rangle$ follows from its zero-knowledge property. Thus, we start the simulator S for $\langle P, V \rangle$. As in $\langle P, V \rangle$, the simulator reduces $\langle x_0, x_1 \rangle$ to $\langle X_0, X_1 \rangle$. The simulator uniformly chooses r_0 and r_1, and computes $X_0(r_0)$ and $X_1(r_1)$. If there is $i \in \{0,1\}$ such that $X_i(r_i)$ ends with a 0 (i.e., $X_i \in \mathrm{PU_N}$), then S outputs $\langle \langle x_0, x_1 \rangle, r'_I, \langle r_0, r_1 \rangle \rangle$, where r'_I equals the n^c-bit prefix of $X_{\bar{i}}(r_{\bar{i}})$. Otherwise, S outputs $\langle \langle x_0, x_1 \rangle, r'_I, \langle r_0, r_1 \rangle \rangle$, where r'_I equals the n^c-bit prefix of $X_0(r_0) \oplus X_1(r_1)$. In both cases r'_I is uniformly distributed, and $\langle r_0, r_1 \rangle$ are distributed as in $\langle P, V \rangle$. Thus, S perfectly simulates $\langle P, V \rangle$. Since S always outputs accepting transcripts, $\langle P, V \rangle$ has perfect completeness.

We turn our attention to the soundness property. Let $x_0, x_1 \in \Pi_{\mathrm{N}}$, and let $\langle r_0, r_1 \rangle$ be the message received by V. We consider two cases in which V accepts. In the first case there is $i \in \{0,1\}$ such that $X_i(r_i)$ ends with a 0, and $X_{\bar{i}}(r_{\bar{i}}) = r_I 1$. Since $|T_{X_{\bar{i}}}| \leq 2^{n^c/2-1}$, and r_I is uniformly distributed, it follows that in the first case V accepts with probability at most $2 \cdot \Pr_{r_I}[X_{\bar{i}}(r_{\bar{i}}) = r_I 1] \leq 2 \cdot 2^{-(n^c/2+1)}$. The reason we multiplied the probability by 2 is because a cheating P^* may use either X_0 or X_1. In the second case the suffix of both $X_0(r_0)$ and $X_1(r_1)$ is 1, and $X_0(r_0) \oplus X_1(r_1) = r_I 0$. In this case the probability over r_I that $X_0(r_0) \oplus X_1(r_1) = r_I 0$ is at most $1/4$ because $|T_{X_0}| \cdot |T_{X_1}| \leq 2^{n^c/2-1} \cdot 2^{n^c/2-1} = 2^{n^c}/4$, and r_I is uniformly distributed. We conclude that in total V accepts with probability at most $1/4 + 2 \cdot 2^{-(n^c/2+1)}$, which is $1/3$ for sufficiently large inputs.

Recall that in the beginning of this proof we considered the case where $|x_0| = |x_1|$. In this case the length of the output of X_0 equals that of X_1. The general case can be treated exactly the same, except that X_0 and X_1 are modified before the protocol begins. For example, if $|x_0| = n$ and $|x_1| = n + a$ (for some $a \in \mathbb{N}$), then we simply add $(n+a)^c - n^c$ input gates to X_0. These gates are outputted as the prefix of X_0. Call this new circuit X'_0. Now both X'_0 and X_1 have $(n+a)^c + 1$ output bits, and X'_0 inherits the properties of X_0 (that is, for any α and β, if X_0 is α-positive, then X'_0 is α-positive, and if X_0 is β-negative, then X'_0 is β-negative). Thus, we can apply the proof as above. The lemma follows. □

4.5 Conclusion and Open Questions

We explained why reductions that apply to the statistical setting do not apply to the perfect setting. Using the error shifting technique we modified these reductions, thus obtaining complete and hard problems, and interesting applications. A natural open question that follows is whether IDENTICAL DISTRIBUTIONS, our hard problem for the class of problems admitting public-coin **HVPZK** proofs, is also complete for this class. An easier task would be to provide a **HVPZK** proof for ID that is unnecessarily public-coin (this question was mentioned in [77]).

Chapter 5

The Round Complexity of Perfect Zero-Knowledge Proofs

In Chapter 3 we characterized all the *known* problems admitting perfect zero-knowledge (**PZK**) proofs. The fact that all of these problems admit 3-round public-coin protocols (which we called V-bit protocols) is an intriguing phenomenon. Thus, a natural question that follows is whether *all the problems* admitting public-coin **PZK** proofs have a constant number of rounds. Addressing this question is important because it may explain this phenomenon, or alternatively, we may discover a new natural problem that has a **PZK** proof with more than three rounds. Either of these results would be a major progress.

Another motivation to this question is the relationship between statistical and perfect zero-knowledge proofs. Specifically, Ong and Vadhan [73] showed that statistical zero-knowledge proofs can be transformed into ones that have a constant number of rounds. Hence, if the round complexity of **PZK** proofs cannot be collapsed to a constant, then **SZK** $\neq$ **PZK**, thus solving a long standing open question. Notice that when we apply the transformation of [72] to **PZK** proofs, we get constant-round *statistical* zero-knowledge proofs. That is, the zero-knowledge property is degraded. In the case that **PZK** proofs do not have a constant number of rounds, we would get a tradeoff between privacy and the ability to prove assertions using only a few messages. That is, either the prover uses polynomially many messages and it leaks no information, or the prover uses a constant number of messages, but then it must leak some information. Goldreich and Krawczyk [40] showed that a similar tradeoff occurs with public-coin zero-knowledge proofs: either the verifier uses private coins, which makes it difficult for a cheating prover to prove false statements, or the verifier uses public-coins, but then the prover is more likely to cheat the verifier.

5.0.1 Approach

Our goal is to show that if a problem has a public-coin **PZK** proof, then it has a **PZK** proof with a constant number of rounds. Despite the importance of this problem and the interest in the round complexity of interactive protocols in general (see Section 5.0.3), it has not been addressed before.

As a warm up, recall that Itoh, Ohta, and Shizuya [50] observed that in the protocols for **NP**, instance-dependent commitment-schemes can replace bit commitment-schemes. Vadhan [87] observed that the same applies to the protocol of [13] for **AM**. Now, assume that public-coin **PZK** problems admit instance-dependent commitment-schemes, and that the schemes are perfectly hiding and constant-round. Since any **PZK** problem has an **AM** proof [37, 3, 77], we could plug the scheme into the protocol of [13] for **AM** and get a constant-round **PZK** proof. That is, we would collapse the round complexity of public-coin **PZK** proofs to a constant.

Before we proceed, let us examine the difficulties lying ahead. Firstly, current constructions of instance-dependent commitment schemes are only statistically or computationally hiding [63, 87, 70, 73, 23] (our scheme from Chapter 3 is perfectly hiding, but it only applies to V-bit **PZK** protocols). Thus, it is not clear at all whether perfectly-hiding schemes can be constructed from public-coin **PZK** proofs. Secondly, even if such schemes exist, we need to construct ones that are also constant-round. Finally, obtaining constant-round perfectly hiding instant-dependent schemes from public-coin **PZK** problems might be too strong of a requirement. That is, we do not know if a weaker condition may suffice to collapse the round complexity of public-coin **PZK** proofs to a constant.

5.0.2 Main results

Our first question was whether it is at all possible to obtain a *perfectly hiding* scheme. We answered this question on the affirmative.

Theorem 5.0.1 *If a problem admits an honest-verifier perfect zero-knowledge (**HVPZK**) proof, then it has perfectly hiding instance-dependent commitment scheme. If the proof is constant-round (or public-coin), then so is the scheme. The scheme is secure against honest receivers, and the sender is inefficient. The same applies to **HVSZK** and **HVCZK**, in which case the scheme is statistically (respectively, computationally) hiding.*

Our scheme has the same requirements as the scheme of Vadhan [87]: we require hiding on `YES` instances and binding on `NO` instances, we consider the honest-verifier case (as was done also in [70, 73]), and since the sender is inefficient, we require that the scheme be simulatable. Notice that there are various definitions for commitments and instance-dependent commitments in the literature, each tailored to a specific application. In our case the difficulty is in achieving a perfectly hiding scheme. Thus, in our definition we observe that unlike in commitments schemes, where the sender always succeeds in producing a commitment, in *instance-dependent* commitment-schemes this is only necessary on `YES` instances, but not on `NO` instances. This observation allows us to achieve perfect hiding.

Since our scheme inherits its round complexity from the protocol, it does not collapse the round complexity of public-coin **PZK** proofs to a constant. We conclude that the difficulty in collapsing the rounds of public-coin **PZK** proofs is *not* in obtaining a perfectly hiding instance-dependent commitment scheme, but rather in obtaining such a scheme that is also constant-round. But do we have to obtain such a scheme in order to achieve this goal? Using the idea behind the zero-knowledge proof for GRAPH-ISOMORPHISM [41], we show that indeed, the two questions are equivalent. That is, *constant-round*, perfectly hiding instance-dependent commitment-schemes are not only sufficient, but also necessary to collapse the round complexity of public-coin **PZK** proofs to a constant.

Corollary 5.0.2 *A problem admits a* constant-round*, perfectly (respectively, statistically) hiding instance-dependent commitment scheme if and only if it admits a* constant-round ***HVPZK** (respectively, **HVSZK**) proof. The same applies to **HVCZK** if the problem admits a constant-round interactive proof.*

This corollary yields a simple and elegant equivalence between zero-knowledge and instance-dependent commitments. Recall that Vadhan [87] was the first to show an equivalence between zero-knowledge and commitments, and this was recently improved by Ong and Vadhan [73] to schemes that have an efficient sender and a standard binding property (as opposed to $\binom{2}{1}$-binding). The difference between our equivalence to that of [73] is that the equivalence of [73] only applies to the statistical and the computational setting, but it yields a scheme with additional properties, such as being constant-round and having an efficient sender. In contrast, our equivalence applies in all settings (including the perfect), but since our scheme inherits its properties from the protocol, it does not have any additional properties.

In our second attempt to construct a constant-round, perfectly hiding instance-dependent commitment scheme, we combined the circuits of IDENTICAL DISTRIBUTIONS (our hard problem for the class of problems admitting public-coin **PZK** proofs) with the error shifting technique from Chapter 4. We were able to construct a non-interactive, perfectly hiding scheme with a polynomial-time sender. Although this scheme is not binding, for any polynomial p, the fraction of random inputs to the scheme that violates the binding property can be made as small as $1/2^{p(n)}$, where n is the input length.

Lemma 5.0.3 *If a problem admits a public-coin **HVPZK** proof, then it has a* non-interactive*, perfectly hiding instance-dependent commitment scheme with an efficient sender. Given common input x of length n, the scheme is binding on all but $1/2^n$ fraction of its random inputs.*

Informally, this lemma shows that we can collapse the rounds of public-coin **PZK** proofs if we can make sure that the prover does not choose its randomness from a small set. This relationship might be implicit in other works [56, 73], but here we show that it holds for the case of public-coin **PZK** proofs. Also, the literature offers a variety of techniques allowing two parties to jointly choose a random string (e.g., hashing [45, 28], interactive hashing [74, 27, 68], and random selection [14, 78]), and our lemma provides an avenue where such techniques can be used.

As a first step towards making sure that the prover does not choose its randomness from a small set, we defined a *preamble*. The first part of the preamble defines a set A which is big on `YES` instances and small on `NO` instances. Intuitively, A represents all the choices of randomness for the sender, and it contains a small subset B of strings that violate the binding property. The second part uses the set A to define a string r such that on YES instances r can be any string in A, and on NO instances r is unlikely to be a string in B. The preamble provides a framework for choosing randomness for the sender, while at the same time making sure that perfect hiding is maintained.

To put the preamble concept to the test, we applied it to the simple cases of 3-round public-coin **PZK** proofs and non-interactive perfect zero-knowledge (**NIPZK**) proofs. In both cases we constructed the first part of the preamble, but we were only able to construct the second part under assumptions on the soundness of the underlying problem. An interesting consequence of independent interest is that we use the circuits of UNIFORM (our **NIPZK**-complete problem) in the commitment scheme of Naor [67]. This allows us to obtain a new (essentially, non-interactive) instant-dependent commitment scheme with an efficient sender for **NIPZK** problems admitting a small soundness error.

5.0.3 Related Work

There round complexity of interactive protocols has been extensively studied.

In the context of **IP**, Goldwasser and Sipser [45] showed that any interactive proof can be transformed into a *public-coin* proof with essentially the same number of rounds. The famous collapse theorem of Babai and Moran [6, 56] showed that for any **AM** problem (i.e., any problem with a *constant-round* public-coin proof) the number of rounds can be collapsed to two. Informally, the idea is to let the verifier send its randomness in advance, but at the same time play many copies of the protocol with the prover (in parallel). To prevent an exponential growth in the size of the game tree, after each four rounds the verifier chooses one branch on which the game will continue. We commented in Section 4.1 that this idea can be used to obtain deterministic verifiers in non-interactive zero-knowledge proofs. However, we cannot use this idea to collapse the rounds in interactive zero-knowledge proofs because it is not known how to simulate different branches of the interaction.

For the relationship between **AM** and **NP** we refer the reader to [18, 5, 54, 65]. We mention that the *unbounded* levels of the public-coin proof hierarchy are not believed to be contained in the *bounded* levels of the polynomial-time hierarchy [2].

The round complexity of zero-knowledge protocols has always been of great interest (c.f., [53, 9], and the recent works of [52, 64]). Fortnow [37], and Aiello and Håstad [3] showed that **SZK** $\subseteq$ **AM**, and an alternative proof was given in [77]. Since **PZK** $\subseteq$ **SZK**, this means that **PZK** $\subseteq$ **AM**. As for **CZK**, if one way functions exist, then any **IP** proof can be turned into a **CZK** proof with essentially the same number of rounds [13]. Thus, collapsing the rounds of **CZK** proofs essentially boils down to collapsing the rounds of **IP** (equivalently, **PSPACE** [82]). We note that Goldreich and Krawczyk [40] showed that public-coin **CZK** proofs with black-box simulators exist only for trivial problems (i.e., problems in **BPP**). This result clearly extends to **PZK** proofs in particular, but since it assumes proofs *with a negligible soundness error*, it does not apply here. A review on the round complexity of zero-knowledge *arguments* for **NP** can be found in [11].

5.0.4 Organization

In Section 5.1 we show how to construct instance-dependent commitment schemes from zero-knowledge protocols. In Section 5.2 we construct a scheme from the hard problem of [59], and in Section 5.3 we try to fix the binding property of this scheme using our preamble.

5.1 Trivial Instance-Dependent Commitment Schemes

In this section we prove Theorem 5.0.1 by showing how to obtain a *perfectly hiding* instance-dependent commitment scheme from any **HVPZK** proof. Our idea also applies to **HVSZK** and **HVCZK** proofs. As a consequence, we get an equivalence between zero-knowledge and instance dependent commitments, thus proving Corollary 5.0.2.

We start with our definition of instance-dependent commitment schemes. This definition has the same requirements as the scheme of Vadhan [87]: we require hiding on `YES` instances and binding on `NO` instances, we consider the honest-verifier case (as was done also in [70, 73]), and since the sender is inefficient, we require that the scheme be simulatable. Notice that we do not care if the prover fails to produce commitments on `NO` instances (because when x is a `NO` instance we only care about soundness). Thus, we allow the sender to fail in the commit phase, and require that the failure probability be negligible on `YES` instances.

Definition 5.1.1 *An* instance-dependent commitment scheme *for a problem* $\Pi = \langle \Pi_Y, \Pi_N \rangle$ *is a protocol* $\langle S, R \rangle$ *between a* sender S *(with input a bit* b*) and a* receiver R*. The randomness of* R *and* S *is denoted* r_S *and* r_R*, respectively. The running time of* R *is polynomial in* $|x|$*, where* x *is the* common input*. The protocol has two parts:*

- **The commit phase.** *This is the first part of the protocol. If both* S *and* R *follow their instructions, then with probability at least* $1-2^{-|x|}$ *over their randomness this stage ends successfully. The* commitment *of* S *to* b *is denoted by* $\langle S_b, R\rangle(x)$*. It contains* x*, the messages exchanged in this phase, and* r_R*.*

- **The reveal phase.** *This is the second part of the protocol. In this part* S opens the commitment to b *by sending* b *and* r_S *to* R*. The receiver either accepts or rejects* b*. In this stage the view of the receiver is simply denoted* $\langle r_S, b\rangle$*.*

The protocol satisfies three properties:

Hiding. $\langle S, R\rangle$ *is* perfectly (respectively, statistically, computationally) hiding *on* Π_Y *if* $\{\langle S_0, R\rangle(x)\}_{x\in\Pi_Y}$ *and* $\{\langle S_1, R\rangle(x)\}_{x\in\Pi_Y}$ *are identical (respectively, statistically indistinguishable, computationally indistinguishable).*

Binding. $\langle S, R\rangle$ *is* statistically binding *on* Π_N *if for any function* S^* *and common input* $x \in \Pi_N$*, the probability over* r_R *that* R *accepts both* 0 *and* 1 *in the reveal phase is at most* $1/2^{|x|}$.

Simulation *(against an honest receiver). $\langle S, R\rangle$ is* perfectly (respectively, statistically, computationally) simulatable against the honest receiver *if there is a probabilistic Turing machine M that runs in time polynomial in x such that for any b it holds that $\{M(x, b)\}_{x\in\Pi_Y}$ and $\{\langle\langle S_b, R\rangle(x), \langle r_S, b\rangle\rangle\}_{x\in\Pi_Y}$ are identical (respectively, statistically indistinguishable, computationally indistinguishable).*

We start with the forward direction of Corollary 5.0.2.

Lemma 5.1.2 *If a problem admits a perfectly hiding instance-dependent commitment scheme, then it has a **HVPZK** proof. If the scheme is constant-round (or public-coin), then so is the **HVPZK** proof.*

Proof: We use the idea behind the proof systems for GRAPH-ISOMORPHISM [41]. That is, the prover commits to the bit 0, and the verifier replies with a random bit b. The verifier accepts only if the prover opens the commitment to the bit b. Soundness follows from the fact that the commitment is binding on NO instances. The hiding property of the scheme guarantees that the same commitment can be opened to both 0 and 1, and thus the protocol is complete. The protocol is **HVPZK** because the simulator can guess b, and then simulate a commitment to b with the honest receiver by executing $M(x, b)$, where M is guaranteed by the simulation requirement from Definition 5.1.1. Notice that the fact that the scheme may fail does not affect the perfect simulation because, just like the prover, the simulator will fail in the commit phase. □

The above proof also applies to **HVSZK** problems, but it may not apply **HVCZK** because the prover may not be able to open commitments to both 0 and 1. Instead, we can plug the instance-dependent commitment scheme in the protocol of [13] for **AM**, and if the underlying problem has a constant round interactive proof, then we get a constant-round public-coin **HVCZK** proof. Notice that in all cases we can apply the transformation of [28] to the constant-round honest-verifier zero-knowledge proof, and obtain a constant-round zero-knowledge proof.

We proceed to prove Theorem 5.0.1 by showing how to construct instance-dependent commitment schemes from zero-knowledge protocols. Combining this with the above lemma, we obtain Corollary 5.0.2. Again, we deal with **HVPZK**, but the proof easily extends to **HVSZK** and **HVCZK**.

Proof of Theorem 5.0.1: Let Π be a problem admitting a constant-round **HVPZK** proof $\langle P, V\rangle$. Since we deal with the honest verifier, the completeness and soundness error can be reduced to $1/2^n$. We use $\langle P, V\rangle$ to construct an instance-dependent commitment scheme for Π. The idea is to use the soundness property of $\langle P, V\rangle$ to obtain binding, the completeness and zero-knowledge properties to obtain hiding, and the zero-knowledge property to obtain simulation.

Formally, let S_b denote the sender with a bit b, let R to denote the receiver, and let x denote the input. In the commit phase S and R execute P and V on input x, respectively. There are two cases.

- If V accepts, then the sender does not send b, and the commit phase terminates successfully. Notice that the bit b takes no part in the execution of the commit phase. In the reveal phase the sender simply reveals b (without sending its randomness), and the receiver accepts.

- If V rejects, then both the commit and the reveal phases terminate. That is, in the commit phase the sender sends `fail`, and in the reveal phase the sender does not send anything and the receiver rejects.

We verify the properties of the scheme. Let $n \stackrel{\text{def}}{=} |x|$. If x is a NO instance, then the scheme is binding because R rejects with probability at least $1 - 2^{-n}$ over r_R. If x is a YES instance and both S and R follow their instructions, then the commit phase terminates successfully because V accepts x with probability at least $1 - 2^n$ over the randomness of S and R. Since S does not send b in the commit phase, the scheme is perfectly hiding. Notice that with probability at most $1/2^n$ the sender fails in the commit phase, but the bit b is still hidden. The simulator M for $\langle S, R \rangle$ simply mimics the sender, and it can be easily constructed from the **HVPZK** simulator S of $\langle P, V \rangle$. Formally, $M(x, b)$ obtains a transcript $\langle x, m_1, m_2, \ldots, m_v; r_V \rangle$ of $S(x)$, and if V accepts in this transcript, then M outputs $\langle \langle x, m_1, m_2, \ldots, m_v; r_V \rangle, \langle \epsilon, b \rangle \rangle$, where ϵ is the empty string, and b is the bit of the sender. Otherwise, just like the prover, it adds the `fail` message to the transcript, and outputs $\langle \langle x, m_1, m_2, \ldots, m_v, \texttt{fail}; r_V \rangle, \epsilon \rangle$. □

Notice that in the above proof S is a **HVPZK** simulator, and thus M perfectly simulates the commitment. However, although b is not involved in the commit phase, if S is a **HVSZK** or a **HVCZK** simulator, them M will only statistically or computationally simulate the commit phase, and thus the hiding property will be statistical or computational, respectively.

5.2 Instance-Dependent Commitments from Hard Problems

In this section we prove Lemma 5.0.3 by constructing a perfectly hiding instance-dependent commitment scheme. Although our scheme is not binding, the binding property holds on almost all the inputs, and this shows that we can collapse the rounds of public-coin **PZK** proofs if we can make sure that the prover does not choose its randomness from a small set.

Since we have a hard problem for the class of problems admitting public-coin **HVPZK** proofs, we can use the approach of Vadhan [87], which utilizes complete problems (or similar characterizations) to construct the scheme. That is, we use the circuits of our problem IDENTICAL DISTRIBUTIONS (ID) from Section 4.3. Recall that instances of ID are triplets $\langle X_0, X_1, Z \rangle$ of circuits, and as we explained in Section 4.3 the circuit Z can be ignored because in any zero-knowledge proof (or an instance-dependent commitment scheme) for ID the verifier can sample Z and reject immediately if $\Pr[Z = 1] \leq 1/3$. Thus, throughout this chapter, when we refer to ID, we actually refer to instances $\langle X_0, X_1 \rangle$ of $\mathrm{SD}^{0,1/2}$. That is, as YES instances X_0 and X_1 represent the same distribution, and as NO instances they represent statistically far distributions.

5.2.1 A Perfectly Hiding Scheme That is Almost Binding

Our goal is to construct a constant-round, perfectly hiding instance-dependent commitment scheme for ID. Micciancio and Vadhan [63] showed that $\mathrm{SD}^{0,1}$ has such a scheme: a commitment to the bit b is a random sample of X_b. With respect to ID, this idea guarantees perfect hiding on YES instances because X_0 and X_1 represent the same distribution, and thus it is impossible to determine b from y. However, this idea does not guarantee binding on NO instances of ID because there could be r and r' such that $X_0(r) = X_1(r')$, which may allow the sender to open y as a commitment to both 0 or 1 .

Our idea is to use *multiple intertwined samples*. That is, we use $n = |\langle X_0, X_1\rangle|$ additional samples, and the string r appears in all of them. Formally, to commit to a bit b the prover chooses $n+1$ random strings $r, r_1, \ldots, r_n$, and it sends to the verifier the commitment $\vec{y} = \langle X_b(r), X_b(r \oplus r_1), \ldots, X_b(r \oplus r_n)\rangle$. As before, in the reveal phase the prover sends b and $r, r_1, \ldots, r_n$, and the verifier checks that $\vec{y}$ was computed correctly. This scheme is described in Figure 5.1.

An instance-dependent scheme $\langle S, R\rangle$

Common input: a pair of circuits$\langle X_0, X_1\rangle$. Let $n = |\langle X_0, X_1\rangle|$.
Private input for S: a bit b.

The *sender S* commits to a bit b as follows:

1. S uniformly chooses a string r, and computes $y \overset{\text{def}}{=} X_b(r)$.
2. S uniformly chooses strings $r_1, \ldots, r_n$, and computes $y_i \overset{\text{def}}{=} X_b(r \oplus r_i)$.
3. S sends $\vec{y} \overset{\text{def}}{=} \langle y, y_1, \ldots, y_n\rangle$ to the *receiver R*.

In the reveal phase S sends $r, r_1, \ldots, r_n$ to R.

Figure 5.1: A perfectly hiding scheme whose binding property holds on almost all the random inputs.

The first observation about the modified scheme is that if $r, r_1, \ldots, r_n$ are uniformly chosen, then the strings $r, r \oplus r_1, \ldots, r \oplus r_n$ are also uniformly chosen and *independent*. Thus, the modified scheme retains the perfect hiding property. The second observation is that the modified scheme is not binding. However, notice that in the previous scheme the sender could cheat using any pair $\langle r, r'\rangle$ for which $X_0(r) = X_1(r')$, and many such pairs may exist. In contrast, in the modified scheme the sender can cheat using only a small fraction of the strings $r_1, \ldots, r_n$, regardless of the number of pairs $\langle r, r'\rangle$ for which $X_0(r) = X_1(r')$ (intuitively, replacing $X_0(r)$ with $X_1(r')$ affects the rest of the samples, which requires a cheating sender to adjust the strings $r_1, \ldots, r_n$). Hence, $\vec{y}$ cannot be opened as a commitment to both 0 and 1, except for a small fraction of the strings $r_1, \ldots, r_n$. To formalize this, we start with one sample.

Lemma 5.2.1 *Let X_0 and X_1 be circuits. Let r and r' be strings such that $X_0(r) = X_0(r')$, and let $\alpha \overset{\text{def}}{=} \Delta(X_0, X_1)$. If r_1 is uniformly chosen, then the probability that $X_0(r \oplus r_1) = X_1(r' \oplus r_1)$ is at most $1 - \alpha$.*

Proof: We use two sets in our analysis. The first set contains strings y that are more likely to be outputted by X_0 than by X_1, and the second set is defined analogously. Formally,

$$X_0^+ \overset{\text{def}}{=} \{y|\Pr[X_0 = y] \geq \Pr[X_1 = y]\}, \text{ and } X_1^+ \overset{\text{def}}{=} \{y|\Pr[X_1 = y] > \Pr[X_0 = y]\}.$$

Using these sets we upper bound the probability that $X_0(r \oplus r_1) = X_1(r' \oplus r_1)$.

$$\begin{aligned} & \Pr_{r_1}[X_0(r \oplus r_1) = X_1(r' \oplus r_1)] \\ = \; & \Pr_{r_1}[(X_0(r \oplus r_1) = X_1(r' \oplus r_1)) \wedge X_1(r' \oplus r_1) \notin X_0^+] + \\ & \Pr_{r_1}[(X_0(r \oplus r_1) = X_1(r' \oplus r_1)) \wedge X_1(r' \oplus r_1) \in X_0^+]. \end{aligned}$$

Clearly, the first expression in the above sum is upper bounded by $\Pr_{r_1}[X_1(r' \oplus r_1) \notin X_0^+]$. The same applies to the second expression, but we use the equality in this expression to replace $X_1(r' \oplus r_1) \in X_0^+$ with $X_0(r \oplus r_1) \in X_0^+$. Hence, we get that

$$\begin{aligned} & \Pr_{r_1}[(X_0(r \oplus r_1) = X_1(r' \oplus r_1)) \wedge X_1(r' \oplus r_1) \notin X_0^+] + \\ & \Pr_{r_1}[(X_0(r \oplus r_1) = X_1(r' \oplus r_1)) \wedge X_1(r' \oplus r_1) \in X_0^+] \\ \leq \; & \Pr_{r_1}[X_1(r' \oplus r_1) \notin X_0^+] + \Pr_{r_1}[X_0(r \oplus r_1) \in X_0^+] \\ = \; & 1 - \Pr_{r_1}[X_1(r' \oplus r_1) \in X_0^+] + \Pr_{r_1}[X_0(r \oplus r_1) \in X_0^+]. \end{aligned}$$

Now we use a fact that follows from the definition of statistical distance (see Fact 3.1.9 in [86]). According to this fact, $\Delta(X_0, X_1) = \Pr[X_0 \in X_0^+] - \Pr[X_1 \in X_0^+]$. Thus, since r and r' are fixed, we get that

$$\Delta(X_0, X_1) = \Pr_{r_1}[X_0(r \oplus r_1) \in X_0^+] - \Pr_{r_1}[X_1(r' \oplus r_1) \in X_0^+].$$

Since $\Delta(X_0, X_1) = \alpha$, we get that $\Pr_{r_1}[X_0(r \oplus r_1) = X_1(r' \oplus r_1)] \leq 1 - \alpha$. $\square$

It follows that by taking more samples, we can reduce the number of strings that allow a cheating sender to open commitments to both 0 and 1. Formally, let X_0 and X_1 be circuits on inputs of length m, and let $n = |\langle X_0, X_1 \rangle|$. We claim that for any r and r', if $r_1, \ldots, r_{2n}$ are uniformly chosen, then with probability at most 2^{-n} it holds that

$$\langle X_0(r), X_0(r \oplus r_1), \ldots, X_0(r \oplus r_{2n}) \rangle = \langle X_1(r), X_1(r \oplus r_1), \ldots, X_1(r \oplus r_{2n}) \rangle.$$

This is so because by Lemma 5.2.1, the probability over $r_1, \ldots, r_{2n}$ that the above equality holds is at most $(1 - \alpha)^{2n} \leq 2^{-2n}$ (recall that $\alpha = \Delta(X_0, X_1) \geq 1/2$ when $\langle X_0, X_1 \rangle$ is a `NO` instance of ID). Since there are at most $2^{2m} \leq 2^n$ pairs $\langle r, r' \rangle$ for which $X_0(r) = X_1(r')$, our scheme is not binding with probability at most $2^n \cdot 2^{-2n} \leq 2^{-n}$. Lemma 5.0.3 follows.

5.3 A Preamble for Jointly Choosing Randomness

In the previous section we constructed a scheme that is not binding if the sender chooses its randomness from a small set. In this section we define a *preamble* that provides a framework for choosing randomness for the sender, while at the same time making sure that perfect hiding is maintained. Such a preamble would fix the binding property of our scheme, thus collapsing the round complexity of public-coin **PZK** proofs to a constant. We then test the preamble on the simple cases of 3-round public-coin **PZK** proofs (Section 5.3.1) and **NIPZK** proofs (Section 5.3.2), and obtain interesting consequences.

Motivation. Since the randomness of our scheme is chosen by the sender, a cheating sender may be able to open the commitment to both 0 and 1. Hence, it makes sense to restrict the randomness used by the sender. In the statistical setting Goldreich and Vadhan [44] used the hashing technique of Goldwasser and Sipser [45], whereby one party chooses a hash function h, and the other party is restricted to strings r such that $h(r) = 0$. Indeed, forcing the sender to use randomness from the small set $h^{-1}(0)$ will make our scheme binding, but in the perfect setting it compromises the hiding property (a similar issue occurs in [70], where the interactive hashing technique due to Ding, Harnik, Rosen, and Shaltiel [32] is used; interactive hashing was introduced by Naor, Ostrovsky, Venkatesan, and Yung [68]).

Thus, we need an *instance-dependent random-selection* protocol. That is, a protocol that would restrict the randomness of the sender in a way that depends on the common input. We formalize this using a *preamble*. The first part of the preamble defines a set A which is big on YES instances and small on NO instances. Intuitively, A represents all the choices of randomness for the sender, and it contains a small subset B of strings that violate the binding property. The second part uses the set A to define a string r such that on YES instances r can be any string in A, and on NO instances r is unlikely to be a string in B. More formally,

1. **Defining a set.** Let x be an instance of ID, and let $p(n)$ denote the length of the random input to our scheme. The sender and the receiver execute a protocol that defines a set $A \subseteq \{0,1\}^{p(n)}$, where $n \stackrel{\text{def}}{=} |x|$. If x is a YES instance, then $|A| = 2^{p(n)}$, and if x is a NO instance, then $A \ll 2^{p(n)}$.

2. **Randomizing the set.** Let $B \subsetneq \{0,1\}^{p(n)}$ be the set of "bad" strings (those that violate the binding property of our scheme). Using A the parties define a string r. If x is a YES instance, then r can equally be any string in $A = \{0,1\}^{p(n)}$, and if x is a NO instance, then r is unlikely to be in B.

Suppose that we could construct such a preamble for ID. We could then execute S and R from our scheme in Figure 5.1, and have S commit to its bit b using r as randomness. If x is a YES instance, then r can be any string in A, and thus $\vec{y}$ perfectly hides b. If x is a NO instance, then $r \notin B$ with high probability over the randomness of the receiver, and thus $\vec{y}$ binds the sender to b.

5.3.1 The Case of 3-round Public-Coin PZK Proofs

Our goal is to construct a preamble for any problem that admits a public-coin **PZK** proof. Since we do not know how to do it, we deal with the simple case of 3-round public-coin **PZK** proofs. Notice that the preamble must have an *efficient sender*, or else we could directly apply Theorem 5.0.1 to the 3-round public-coin **PZK** proof, and obtain a constant-round, perfectly hiding instance-dependent commitment-scheme.

Consider a 3-round, public-coin **PZK** proof $\langle P, V\rangle$ with a simulator M, and let $\langle x, m_1, r_1, m_2\rangle$ denote the output of $M(x)$. That is, on input x the prover sends m_1, the verifier sends r_1, the prover replies with m_2, and based on these messages the verifier accepts of rejects. To simplify the presentation, we let $|r_1| = n \stackrel{\text{def}}{=} |x|$, and denote by n^c the length of the random input to our commitment scheme $\langle S, R\rangle(x)$.

Preamble - Step 1. The first step of our preamble is to define a set A. This can be done by having the sender execute $M(x)$, obtain a transcript $\langle x, m_1, r_1, m_2\rangle$, and send m_1 to the receiver. We define A to be the set of all r_1 such that $M(x) = \langle x, m_1, r_1, m_2\rangle$ and $V(x, m_1, r_1, m_2) = \texttt{accept}$. Actually, we want A to contain strings of length n^c. Thus, we let the sender sample M for n^{c-1} times, obtain a vector $\vec{r}$ of n^{c-1} messages r_1, and send a vector of n^{c-1} messages m_2 to the receiver. Suppose that $\langle P, V\rangle$ has soundness error $1/2$ and perfect completeness. Thus, if x is a YES instance, then $A = \{0,1\}^{n^c}$, and if x is a NO instance, then A contains at most a $1/2^{n^c-1}$ fraction of the strings in $\{0,1\}^{n^c}$.

Preamble - Step 2. The second step of our preamble is to define a string r that would later be used by the sender in our commitment scheme. Let B be the set of all strings that violate the binding property of our scheme. By Lemma 5.0.3, B contains at most a $1/2^n$ fraction of the strings in $\{0,1\}^{n^c}$. We remark that if $A \cap B = \emptyset$, then we could simply define $r = \vec{r}$ (i.e., the randomness for S is the concatenation of the n^c messages r_1), but of course, this may not be the case. Suppose that $\langle P, V\rangle$ has a very small soundness error of $1/2^{n-(n^{-c+2})/2}$. In such a case we can let the receiver send a random string r' to the sender, and define $r \stackrel{\text{def}}{=} \vec{r} \oplus r'$. When x is a NO instance the probability that $r \in B$ is at most $|A| \cdot 1/2^n = (2^n/2^{n-(n^{-c+2})/2})^{n^{c-1}}/2^n = 2^{-n/2}$.

Thus, if the sender in our scheme uses r as its randomness, then the scheme is binding on NO instances. If x is a YES instance, then r is hidden from the receiver, and thus our scheme is perfectly hiding. We can remove the assumption on perfect completeness by allowing the sender to fail (this happens with small probability because, after executing M many times, the sender is likely to obtain n^c accepting transcripts). Unfortunately, we do not know how to remove the restriction on the soundness.

5.3.2 The Case of Non-Interactive PZK Proofs

Our goal was to collapse the number of rounds in public-coin **PZK** proofs to a constant. We could achieve this goal if our scheme was binding. We tried to construct a preamble that would fix the binding property, but we were unsuccessful even for the simple case of 3-round public-coin **PZK** proofs.

In this section we want to provide a better understanding into the difficulties involved. Thus, we try to construct the preamble for the other simple case of **NIPZK** proofs (which can be viewed as a 2-round, public-coin **HVPZK** proofs). Although we could not construct the preamble, our investigation yields two interesting consequences. Firstly, we show how to use the circuits from the study of **NIPZK** in the commitment scheme of Naor [67]. This leads to a new perfectly-hiding instance-dependent commitment for **NIPZK** problems with a small soundness error. Secondly, we show how to use hash functions without damaging the hiding property. This is useful because, as we mentioned earlier, most hashing techniques (e.g., [45, 32]) do not apply in the perfect setting.

Since we are dealing with the non-interactive setting, our underlying problem will be UNIFORM (UN), our **NIPZK**-complete problem from Section 4.2. Recall that `YES` instances of UN are circuits that represent the uniform distribution, and `NO` instances are circuits that have a small range. Actually, the circuits have an additional output bit, but it can be ignored (in the same way that we ignored the circuit Z of the problem IDENTICAL DISTRIBUTIONS). Thus, throughout this section we will be working with a variant of STATISTICAL DISTANCE FROM UNIFORM (SDU), the **NISZK**-complete problem of [43].

Definition 5.3.1 *Define* $\mathrm{SDU}' \overset{def}{=} \langle \mathrm{SDU}'_Y, \mathrm{SDU}'_N \rangle$ *as*

$$\begin{aligned} \mathrm{SDU}'_Y &= \{X \mid \Delta(X, U_n) = 0\}, \text{ and} \\ \mathrm{SDU}'_N &= \{X \mid \mathrm{Rng(X)} < 2^{\mathrm{n}}/3\}, \end{aligned}$$

where X is a circuit with n output bits, and U_n is the uniform distribution on $\{0,1\}^n$.

Motivation. Our goal is to construct a constant-round, perfectly hiding instance-dependent commitment scheme, this time for SDU'. Again, the scheme must have an *efficient sender*, or else it trivially exists by Theorem 5.0.1 (because **NIPZK** proofs are constant-round **HVPZK** proofs in particular).

As a warm up, consider the commitment scheme of Naor [67], which uses a pseudo-random generator $G : \{0,1\}^n \to \{0,1\}^{3n}$. In this scheme the receiver sends a random string $r \in \{0,1\}^{3n}$ to the sender. The sender chooses a random string $r' \in \{0,1\}^n$, and commits to 0 by sending $G(r') \oplus r$, and to 1 by sending $G(r')$. To see why this scheme is binding against computationally unbounded senders, consider a commitment $G(r')$. Since the range of G contains at most a $1/2^{2n}$ fraction of the strings $\{0,1\}^{3n}$, the probability that $G(r') \oplus r$ falls back into $\mathrm{Rng(G)}$ (that is, $G(r') \oplus r = G(r'')$ for some r'') is at most 2^{-2n}. Thus, the scheme is binding with probability at least $|\mathrm{Rng(G)}| \cdot 2^{-2\mathrm{n}} = 2^{-\mathrm{n}}$.

We apply this idea to instances of SDU'. That is, on circuit X with n output bits the receiver sends a uniformly chosen $r \in \{0,1\}^n$, and the sender commits to 0 by sending $X(r') \oplus r$, and to 1 by sending $X(r')$. The resulting instance-dependent scheme is perfectly hiding on `YES` instances. If SDU' has a very small soundness error of $2^{-n/4}$, then its `NO` instances satisfy $|\mathrm{Rng(X)}| \leq 2^{\mathrm{n}/4}$, and by the same argument as above, the probability over r that there are r' and r'' such that $X(r') \oplus r = X(r'')$ is at most $|\mathrm{Rng(X)}| \cdot 2^{-3\mathrm{n}/4} \leq 2^{-\mathrm{n}/2}$. Of course, the range of X may be bigger, and thus we cannot use this idea.

Constructing a Preamble. We modify the scheme of Naor [67] using hash functions. Intuitively, the sender will commit to 0 by sending $h'(r)$, and to 1 by sending $X(r_0)$. The string r_0 is chosen by the sender, the string r is chosen by the receiver, and h' is a hash function chosen jointly. The idea is that, if X is a NO instance, then it has a small range, and thus it is unlikely that $h'(r) \in \mathrm{Rng(X)}$. The scheme guarantees perfect hiding on YES instances, but as we shall see it does not guarantee binding. We formally describe our scheme using the preamble idea.

Let X be a circuit with n output bits, let c be some constant, and define $X' \overset{\text{def}}{=} \oplus^{n^c} X$ as the circuit that takes n^{c-1} strings r_i, and outputs $X(r_1), \ldots, X(r_{n^{c-1}})$. The circuit X' has n^c output gates. If X is a YES instance, then X' represents the uniform distribution, and if X is a NO instance, then X' has a small range.

Preamble - Step 1. In the first step the sender picks two samples of X', and sends the XOR to the receiver. That is, the sender picks r_0, r_1, computes $h_0 = X'(r_0), h_1 = X'(r_1)$, and sends $y = h_0 \oplus h_1$ to the receiver. Using the notation of our preamble, A is a set of hash functions. If X is a YES instance, then $A = \{0,1\}^{n^c}$, and if X is a NO instance, then A contains a small fraction of $\{0,1\}^{n^c}$.
Preamble - Step 2. In the second step the receiver replies with a uniformly chosen hash function h and an input r for h. The sender uses h to define $h' = h_0 \oplus h$. Informally, this ensures that h' does not belong to the set B of hash functions that do not evenly spread their domain over their range.

Now we execute the scheme. The sender commits to $b = 0$ by sending $h'(r)$, and to $b = 1$ by sending $X(r_0)$. If X is a YES instance, then both h' and r_0 are hidden from the receiver, and the scheme is hiding. If X is a NO instance, then both X and X' have small ranges. Since h' is a good hash function, it is unlikely to map r to $\mathrm{Rng(X)}$, and thus the scheme should be binding. Unfortunately, this is not the case because, although h' is likely to be a good hash function, there are $|\mathrm{Rng(X')}|$ possibilities for h_0. In other words, although h' is good, $h'(r)$ may fall into $\mathrm{Rng(X)}$.

5.4 Conclusion

We initiated a preliminary investigation into the question whether the round complexity of public-coin **PZK** proofs can be collapsed to a constant. We gave the first perfectly hiding instance-dependent commitment scheme, and showed that obtaining such a scheme that is also constant round is equivalent to achieving this collapse. We then tried to construct a constant-round, perfectly hiding scheme using the circuits from the hard problem for public-coin **PZK** proofs [59]. Although we could not fix the binding property of the scheme, our attempts had some interesting consequences, including a connection between choosing the randomness of the sender and collapsing the rounds, the definition of the preamble, the difficulty in constructing the preamble, and the use of the circuits of the **NIPZK**-complete problem in the scheme of Naor [67].

Bibliography

[1] Martín Abadi, Joan Feigenbaum, and Joe Kilian. On hiding information from an oracle. *J. Comput. Syst. Sci.*, 39(1):21–50, 1989.

[2] William Aiello, Shafi Goldwasser, and Johan Håstad. On the power of interaction. *Combinatorica*, 10(1):3–25, 1990.

[3] William Aiello and Johan Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *J. of Computer and System Sciences*, 42(3):327–345, June 1991.

[4] Dana Angluin and David Lichtenstein. Provable security in cryptosystems: a survey. Technical Report 288, Department of Computer Science, Yale University, 1983.

[5] Vikraman Arvind and Johannes Köbler. On pseudorandomness and resource-bounded measure. *Theor. Comput. Sci.*, 255(1-2):205–221, 2001.

[6] László Babai. Trading group theory for randomness. In *STOC*, pages 421–429, 1985.

[7] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001.

[8] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 484–493, New York, NY, USA, 2002. ACM Press.

[9] Boaz Barak, Yehuda Lindell, and Salil Vadhan. Lower bounds for non-black-box zero knowledge. *J. Comput. Syst. Sci.*, 72(2):321–391, 2006.

[10] M. Bellare, S. Micali, and R. Ostrovsky. The (true) complexity of statistical zero knowledge. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 494–502, New York, NY, USA, 1990. ACM.

[11] Mihir Bellare, Markus Jakobsson, and Moti Yung. Round-optimal zero-knowledge arguments based on any one-way function. In *EUROCRYPT*, pages 280–305, 1997.

[12] Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. Perfect zero-knowledge in constant rounds. In *22nd STOC*, pages 482–493, 1990.

[13] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In *CRYPTO*, pages 37–56, 1988.

[14] Manuel Blum. Coin flipping by telephone - a protocol for solving impossible problems. In *COMPCON*, pages 133–137, 1982.

[15] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the ICM,pp*, pages 1444–1451, 1986.

[16] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge proofs and their applications. In *Proceedings of the 20th STOC, ACM*, 1988.

[17] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giussepe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.

[18] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Inf. Process. Lett.*, 25(2):127–132, 1987.

[19] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[20] Gilles Brassard and Claude Crépeau. Zero-knowledge simulation of boolean circuits. In *CRYPTO*, pages 223–233, 1986.

[21] Gilles Brassard, Claude Crépeau, and Moti Yung. Everything in NP can be argued in perfect zero-knowledge in a bounded number of rounds (extended abstract). In *EUROCRYPT '89: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 192–195, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[22] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires omega$\sim$(log n) rounds. In *STOC*, pages 570–579, 2001.

[23] André Chailloux, Dragos Florin Ciocan, Iordanis Kerenidis, and Salil P. Vadhan. Interactive and noninteractive zero knowledge are equivalent in the help model. In *TCC*, pages 501–534, 2008.

[24] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158, 1971.

[25] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2 edition, July 18, 2006.

[26] Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, CWI and Uni.of Amsterdam, 1996.

[27] Ivan Damgård. Interactive hashing can simplify zero-knowledge protocol design without computational assumptions (extended abstract). In *CRYPTO*, pages 100–109, 1993.

[28] Ivan Damgård and Oded Goldreich Avi Wigderson. Hashing functions can simplify zero-knowledge protocol design (too). Technical Report RS-94-39, BRICS, November 1994.

[29] Ivan B. Damgård. On the existence of bit commitment schemes and zero-knowledge proofs. In *CRYPTO '89: Proceedings on Advances in cryptology*, pages 17–27, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[30] Ivan B. Damgård. On σ-protocols. Available online at www.daimi.au.dk/ ivan/Sigma.pdf, 2005.

[31] Whitfield Diffie and Martin E. Hellman. Multiuser cryptographic techniques. In *AFIPS National Computer Conference*, pages 109–112, 1976.

[32] Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. *J. Cryptology*, 20(2):165–202, 2007.

[33] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithm. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1984.

[34] 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.

[35] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *J. Cryptology*, 1(2):77–94, 1988.

[36] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

[37] 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.

[38] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, January 15, 1979.

[39] Oded Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.

[40] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

[41] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[42] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *STOC*, pages 399–408, 1998.

[43] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Can statistical zero knowledge be made non-interactive? or on the relationship of SZK and NISZK. In *CRYPTO*, pages 467–484, 1999.

[44] Oded Goldreich and Salil P. Vadhan. Comparing entropies in statistical zero-knowledge with applications to the structure of SZK. In *IEEE Conference on Computational Complexity*, pages 54–73, 1999.

[45] S. Goldwasser and M. 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.

[46] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[47] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT*, pages 339–358, 2006.

[48] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In *EUROCRYPT*, pages 123–128, 1988.

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

[50] Toshiya Itoh, Yuji Ohta, and Hiroki Shizuya. A language-dependent cryptographic primitive. *J. Cryptology*, 10(1):37–50, 1997.

[51] Bruce Kapron, Lior Malka, and Venkatesh Srinivasan. Characterizing non-interactive instance-dependent commitment-schemes (NIC). In *34th International Colloquium on Automata, Languages and Programming (ICALP 2007)*, volume 4596 of *LNCS*, pages 328–339, 2007.

[52] Jonathan Katz. Which languages have 4-round zero-knowledge proofs? In *TCC*, pages 73–88, 2008.

[53] Joe Kilian, Erez Petrank, and Charles Rackoff. Lower bounds for zero knowledge on the internet. In *FOCS*, pages 484–492, 1998.

[54] Adam Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002.

[55] Johannes Köbler. On graph isomorphism for restricted graph classes. In *CiE*, pages 241–256, 2006.

[56] Babai László and Shlomo Moran. Arthur-merlin games: A randomized proof system and a hierarchy of complexity classes. *J. of Computer and System Sciences*, 36:254–276, 1988.

[57] Leonid A. Levin. Universal'nyĭe perebornyĭe zadachi (universal search problems). *Problemy Peredachi Informatsii*, 9(3):265–266, 1973.

[58] Karp R. M. Reducibility among combinatorial problems. In J. W. Thatcher and R. E. Miller, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, Inc., 1972.

[59] Lior Malka. How to achieve perfect simulation and a complete problem for non-interactive perfect zero-knowledge. In *TCC*, pages 89–106, 2008.

[60] Lior Malka. Instance-dependent commitment schemes and the round complexity of perfect zero-knowledge proofs. Technical Report TR08-068, Electronic Colloquium on Computational Complexity (ECCC), July 3, 2008.

[61] Silvio Micali and Rafael Pass. Local zero knowledge. In *STOC*, pages 306–315, 2006.

[62] Daniele Micciancio, Shien Jin Ong, Amit Sahai, and Salil P. Vadhan. Concurrent zero knowledge without complexity assumptions. In *TCC*, pages 1–20, 2006.

[63] Daniele Micciancio and Salil P. Vadhan. Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In *CRYPTO*, pages 282–298, 2003.

[64] Daniele Micciancio and Scott Yilek. The round-complexity of black-box zero-knowledge: A combinatorial characterization. In *TCC*, pages 535–552, 2008.

[65] Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing arthur-merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[66] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, April 2005.

[67] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[68] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for p using any one-way permutation. *J. Cryptology*, 11(2):87–108, 1998.

[69] Minh-Huyen Nguyen, Shien Jin Ong, and Salil Vadhan. Statistical zero-knowledge arguments for NP from any one-way function. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pages 3–14, October 2006. Berkeley, CA.

[70] Minh-Huyen Nguyen and Salil Vadhan. Zero knowledge with efficient provers. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 287–295, New York, NY, USA, 2006. ACM Press.

[71] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. *J. Comput. Syst. Sci.*, 60(1):47–108, 2000.

[72] Shien Jin Ong and Salil P. Vadhan. Zero knowledge and soundness are symmetric. In *EUROCRYPT*, pages 187–209, 2007.

[73] Shien Jin Ong and Salil P. Vadhan. An equivalence between zero knowledge and commitments. In *TCC*, pages 482–500, 2008.

[74] Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Interactive hashing simplifies zero-knowledge protocol design. In *EUROCRYPT*, pages 267–273, 1993.

[75] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, December 10, 1993.

[76] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002.

[77] Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero-knowledge. *J. ACM*, 50(2):196–249, 2003.

[78] Saurabh Sanghvi and Salil P. Vadhan. The round complexity of two-party random selection. In *STOC*, pages 338–347, 2005.

[79] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. On monotone formula closure of SZK. In *IEEE Symposium on Foundations of Computer Science*, pages 454–465, 1994.

[80] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. Image density is complete for non-interactive-SZK (extended abstract). In *ICALP*, pages 784–795, 1998.

[81] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[82] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[83] Michael Sipser. A complexity theoretic approach to randomness. In *STOC*, pages 330–335, 1983.

[84] Martin Tompa and Heather Woll. Random self-reducibility and zero-knowledge interactive proofs of possession of information. In *28th FOCS*, pages 472–482, 1987.

[85] Jacobo Torán. On the hardness of graph isomorphism. *SIAM J. Comput.*, 33(5):1093–1108, 2004.

[86] Salil P. Vadhan. *A study of statistical zero-knowledge proofs*. PhD thesis, MIT, 1999.

[87] Salil P. Vadhan. An unconditional study of computational zero knowledge. *SIAM J. Comput.*, 36(4):1160–1214, 2006.

[88] John Watrous. Zero-knowledge against quantum attacks. In *STOC*, pages 296–305, 2006.