Electronic Colloquium on Computational Complexity, Report No. 132 (2009)

How to Achieve Perfect Simulation and a Complete Problem for Non-interactive Perfect Zero-Knowledge

Lior Malka
Department of Computer Science
University of Victoria, BC, Canada
liorma@cs.uvic.ca

October 10, 2009

Abstract. Perfect zero-knowledge (**PZK**) proofs have been constructed in various settings and they are also interesting from a complexity theoretic perspective. Yet, virtually nothing is known about them. This is in contrast to statistical zero-knowledge proofs, where many general results have been proved using an array of tools, none of which apply to **PZK**. To overcome this barrier we introduce a new *error shifting technique*. This technique helps to achieve perfect simulation and may be useful also in contexts outside of zero-knowledge. Using this technique, we give the first complete problem for the class of problems admitting non-interactive perfect zero-knowledge (**NIPZK**) proofs, and the first hard problem for the class of problems admitting public-coin **PZK** proofs. We hope that our technique and complete problems will facilitate the study of perfect zero-knowledge proofs.

Key words: cryptography, non-interactive, perfect zero-knowledge, perfect simulation, error shifting, complete problems.

1 Introduction

Perfect zero-knowledge protocols allow one party (the *prover*) to prove the validity of an assertion to another party (the *verifier*), but without leaking any information [15]. This is formalized using the notion of a simulator, and requiring that the simulation error be zero. The notion of perfect zero-knowledge can be relaxed to *statistical* zero-knowledge, where the prover leaks a negligible amount of information, and *computational* zero-knowledge, where this leakage is not noticeable by computationally bounded verifiers.

In the past few years there has been great progress in proving general results about the class of problems admitting statistical zero-knowledge (**SZK**) proofs. These results include complete problems, equivalence between private and public-coin, honest and malicious verifier, efficient provers, and more ([22, 26, 11, 13, 33, 21]). Various techniques, such as lower-bound protocols [14] and transformations that polarize and reverse the statistical distance represented by circuits [26], were used in proving these results. Unfortunately, these and other techniques used in the study of statistical zero-knowledge proofs do not apply to the class of problems admitting perfect zero-knowledge (**PZK**) proofs. Intuitively, these techniques manipulate the protocol in a way that introduces a small error into the simulation. This is not an issue in the case of statistical zero-knowledge, where a small simulation error is allowed, but it is an issue in the case of perfect zero-knowledge, where no simulation error is allowed. Consequently, many fundamental questions about **PZK** remain open.

ISSN 1433-8092

Perfect zero-knowledge protocols have been constructed in various settings and there are many reasons to study them. From a cryptographic perspective, they provide the maximum level of privacy for the prover. Under number theoretic assumptions, any language in **NP** has a perfect zero-knowledge argument [5], and recently a non-interactive argument was discovered [16]. Their simple definition also makes them ideal to use as a test bed for studying zero-knowledge in new settings. Recent examples include the local zero-knowledge protocol of [20] and the quantum zero-knowledge protocol of [35]. Perfect zero-knowledge proofs are also intriguing from a complexity theoretic perspective. There are well known problems that unconditionally admit **PZK** proofs, such as QUADRATIC-RESIDUOUSITY, DISCRETE-LOG, and GRAPH-ISOMORPHISM [15, 32, 10]. These problems are in **NP**, but not known to be in **P** or **NP**-complete. Moreover, they all admit 3-round proofs, yet we do not know how to prove that all **PZK** proofs have a constant number of rounds (this was recently proven for **SZK** [23], but again, the techniques do not extend to **PZK**). Our goal is to develop new tools that will facilitate the study of perfect zero-knowledge proofs, and apply these tools to build a complexity theoretic understanding of these proofs.

1.1 Our results

As we mentioned earlier, techniques used in the study of statistical zero-knowledge proofs introduce error into the simulation, and therefore cannot be applied to perfect zero-knowledge proofs. To overcome this difficulty we introduce what we call *an error shifting technique*. Roughly speaking, the idea is to first identify where the error is coming from, and then shift it forward to the protocol where it does not affect the simulation. This is in contrast to techniques from the statistical setting, where the error is incorporated into the constructions, thus leading to simulation errors later on. Notice that the notion of simulation is central to cryptography. Thus, our general technique can help achieve perfect simulation in contexts outside of zero-knowledge.

The first domain to which we apply the error shifting technique is complete problems. Recall that a problem Π is said to be *hard* for some complexity class $\mathcal{C}$ if every problem in the class $\mathcal{C}$ efficiently reduces to it. The problem Π is said to be *complete* for $\mathcal{C}$ if Π is hard for $\mathcal{C}$ and Π is in $\mathcal{C}$. Complete problems are a powerful tool because they represent an entire class. Thus, by proving a result with respect to a complete problem we get a general result about the entire class. Indeed, most of the study of statistical zero-knowledge proofs was made possible by first finding complete problems and then using them to prove more advanced results. This also means that providing complete problems for the perfect setting is an important step towards translating the results from the statistical setting to the perfect setting.

We obtain complete and hard problems in both the interactive and the non-interactive setting. In the non-interactive setting we consider STATISTICAL DISTANCE FROM UNIFORM, the complete problem of Goldreich Sahai and Vadhan [12] (based on [30]) for the class of problems admitting non-interactive statistical zero-knowledge (**NISZK**) proofs. Instances of this problem are circuits that represent distributions, under the convention that the input to the circuit is uniformly distributed. More formally, YES instances represent distributions that are close to uniform, and NO instances have a small range. Intuitively, we shift the error from the reduction, through the circuits, and into the protocol. Hence, we obtain the first complete problem for the class of problems admitting non-interactive perfect zero-knowledge (**NIPZK**) proofs.

Theorem 1**.** The problem UNIFORM (UN) is **NIPZK**-complete.

Our problem UNIFORM is similar to STATISTICAL DISTANCE FROM UNIFORM, except that YES instances of UNIFORM are circuits representing the uniform distribution (the circuits also have an additional output bit for shifting the error forward). The difference between the problems is natural as it reflects the difference between perfect and statistical simulation.

Turning our attention to the interactive model, we consider STATISTICAL-DISTANCE (SD), the complete problem of Sahai and Vadhan [26] for the class of problems admitting statistical zero-knowledge (**SZK**) proofs. Instances of this problem are pairs $\langle X, Y \rangle$ of circuits. As YES instances, X and Y represent statistically close distributions, and as NO instances, X and Y are represent statistically far distributions. Again, by applying the error shifting technique to the reduction of [26], we obtain circuits X and Y that are identically distributed as YES instances, and statistically far as NO instances (our reduction also produces a third circuit for shifting the error forward). We show that this problem is hard for the class of problem admitting perfect zero-knowledge (**PZK**) proofs of the public-coin type.

Theorem 2 **(informal).** The perfect variant of STATISTICAL-DISTANCE is hard for the class of problems admitting **public-coin-PZK** (and **public-coin-HVPZK**) proofs.

Our theorems and the error shifting technique can facilitate the study of perfect zero-knowledge proofs in both the interactive and the non-interactive setting. For example, our hard problem was used in [19] to study the round complexity of perfect zero-knowledge proofs and to prove an equivalence between zero-knowledge and instance-dependent commitment schemes in the perfect setting (a more meaningful equivalence was recently given [23], but it only applies to the statistical and the computational settings). We give two additional applications.

The first application shows equivalence between two notions of simulation. Specifically, we show that the notion of zero-knowledge where the simulator is allowed to fail (also known as *abort*) is equivalent to the notion of zero-knowledge where the simulator is not allowed to fail. This result is with respect to any fixed verifier and strict (as opposed to expected) polynomial-time simulators. Such equivalence was not known previously, and various works were using different notions of simulation. By providing an equivalence between these notions of simulation, we make these works comparable. Our idea uses the error shifting technique and can be applied also to simulators outside of zero-knowledge.

The second application considers closure properties of **NIPZK**. That is, using UNIFORM, we prove that **NIPZK** is closed under the OR operator. The downside of our lemma is that it holds under a strong condition on the soundness and completeness error of the underlying problem. However, the issues that we encounter illuminate the difficulties of working with perfect zero-knowledge proofs, and we believe that exploring new ideas, even if we have to make strong assumptions as a starting point, is useful. We mention that no such closure result is known in the case of non-interactive statistical zero-knowledge (**NISZK**) proofs, where small simulation errors are allowed and more tools and techniques are available (c.f., [30, 12]).

1.2 Related Work

As we mentioned earlier, virtually nothing is known about perfect zero-knowledge proofs. The only exception is the work of [6], which shows a transformation from honest-verifier **PZK** proofs to malicious-verifier **PZK** proofs. This transformation applies only to constant-round, public-coin proofs.

Our work is inspired by the study of statistical zero-knowledge proofs, and we build on the results of [26, 12] (based on [8, 1, 30]). Sahai and Vadhan [26] showed a **HVPZK**-complete problem, but their problem is unnatural, and defined in terms of the class itself. They also tried to modify the reductions from the statistical setting so that they apply to the perfect setting, but their idea works only in certain cases (e.g., when the underlying problem has perfect completeness). Bellare and Rogaway [3] showed other basic results about **NIPZK**, but their notion of zero-knowledge allows simulation in expected (as opposed to strict) polynomial-time. This notion is disadvantageous, especially when non-interactive protocols are executed as sub-protocols. The literature offers a variety of **NIPZK** proofs for specific problems (c.f., [3, 4, 27]) and other results about **NIPZK** proofs that apply to problems with special properties (c.f., [27, 28, 29]).

1.3 Organization

We use standard definitions, to be found in Section 2. In Section 3 we present the error shifting technique and use it to obtain a **NIPZK**-complete problem. In Section 4 we apply this technique to the interactive setting, where we obtain a hard problem. In Section 5 we show some applications of these results.

2 Preliminaries

We study complexity classes of *promise problems* [7], which are a generalization of languages. Formally, $\Pi \overset{\text{def}}{=} \langle \Pi_{\text{Y}}, \Pi_{\text{N}} \rangle$ is a problem if $\Pi_{\text{Y}} \cap \Pi_{\text{N}} = \emptyset$. The set Π_{Y} contains the `YES` instances of Π, and the set Π_{N} contains the `NO` instances of Π. We define $\overline{\Pi} \overset{\text{def}}{=} \langle \Pi_{\text{N}}, \Pi_{\text{Y}} \rangle$. Any language L can be defined as a promise problem $\langle L, \overline{L} \rangle$.

Just like the study of statistical zero-knowledge proofs, our promise problems will be defined in terms of circuits. A circuit $X : \{0,1\}^m \rightarrow \{0,1\}^n$ is a boolean function, encoded in some way (e.g. [24]), but we mainly treat X as a distribution, under the convention that the input to the circuit is uniformly distributed. For example, given a set T, the probability $\Pr[X \in T]$ equals $\Pr_r[X(r) \in T]$, where r is uniformly chosen from $\{0,1\}^m$. The statistical distance between circuits, or more generally, the *statistical distance* between two discrete distributions X and Y, is defined as $\Delta(\text{X}, \text{Y}) \overset{\text{def}}{=} \sum_\alpha |\Pr[X = \alpha] - \Pr[Y = \alpha]|$.

2.1 Protocols and proofs

We study both interactive and non-interactive perfect zero-knowledge proofs, using standard definitions [9]. We start with the definition of a non-interactive protocol, which we customize for the context of zero-knowledge proofs.

Definition 2.1 (Non-interactive protocols) *A* non-interactive protocol $\langle c, P, V \rangle$ *is a triplet (or simply a pair* $\langle P, V \rangle$*, making* c *implicit), where* P *and* V *are functions, and* $c \in \mathbb{N}$. *We use* r_P *to denote the random input to* P. *The* interaction *between* P *and* V *on* common input x *is the following random process.*

1. *Uniformly choose* r_P *and a* common random string $r_I \in \{0,1\}^{|x|^c}$.
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 $\langle P, V \rangle(x) \overset{\text{def}}{=} \langle x, r_I, \pi \rangle$ *the* view *of* V *on* x. *We say that* V accepts x *(respectively,* rejects x*) if* $m = \texttt{accept}$ *(respectively,* $m = \texttt{reject}$*).*

Definition 2.1 considers a deterministic verifier V, and is equivalent to the definition that considers a probabilistic verifier [18]. The definition of interactive protocols is a simple extension of the above, except that there is no common random string, V has random input r_V, and P and V exchange messages until one of them accepts, rejects, or fails. Formally,

Definition 2.2 (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 $m_i = P(x, z, m_1, \ldots, m_{i-1}; r_P)$.*
   (b) *If i is even, let $m_i = V(x, z, m_1, \ldots, m_{i-1}; r_V)$.*
   (c) *If $m_i \in \{\texttt{accept}, \texttt{reject}, \texttt{fail}\}$, then exit loop.*

Each interaction yields a transcript *$\langle x, z, m_1, \ldots, m_p; r_V\rangle$, and the strings m_i are called* messages. *The probability space containing all the transcripts is called* the view of V on x, *and is denoted $\langle P, V\rangle(x)$. We say that V* accepts x *if $m_i = \texttt{accept}$ for an even i.*

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. We say that $\langle P, V\rangle$ is* constant round *if there is a constant c such that in any interaction the number of messages exchanged in at most c.*

A *proof* for a problem is a protocol that admits certain properties with respect to the problem. Informally, the verifier is efficient, with high probability it accepts `YES` instances of the problem , and with low probability it accepts `NO` instances (even if a computationally unbounded prover is cheating). In the following definition the difference between these probabilities is expressed via a non-negligible function c.

Definition 2.3 (Non-interactive proofs) *A non-interactive protocol $\langle c, P, V\rangle$ 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: $\Pr_{r_I}[V(x, r_I, P^*(x, r_I)) = \texttt{accept}] \leq s(|x|)$ 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 $c \equiv 0$.*

Although the completeness and soundness errors are defined using functions, in both the interactive and the non-interactive model our reductions will actually use $c \equiv s \equiv \frac{1}{3}$. This is without loss of generality because the reductions consider honest verifiers and therefore the errors can be reduced via parallel repetition.

Interactive proofs are defined from interactive protocols in exactly the same way, except that there is no reference string. Formally,

Definition 2.4 (Interactive proofs) *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.*

2.2 Zero-knowledge

We proceed to the definition of zero-knowledge. Intuitively, a protocol is zero-knowledge if the view of the verifier can be produced by the verifier itself, without help from the prover. This is formalized using the notion of a polynomial-time simulator that creates this view. The following definition considers a simulator that does not fail, but in Section 5 we give an alternative definition where the simulator is allowed to fail, and show that the two are equivalent.

Definition 2.5 (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 strict, 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***) proofs is also denoted* **NIPZK** *(respectively,* **NISZK**, **NICZK***).*

This definition can be extended to the interactive setting in the natural way. In the following, S^{V^*} denotes oracle access of S to the Turing machine V^*.

Definition 2.6 (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 strict, probabilistic, polynomial-time Turing machine S, called* the simulator*, such that for any strict, probabilistic, polynomial-time Turing machine V^* it holds that*

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

*are statistically-identical (respectively, statistically indistinguishable, computationally indistinguishable.) The class of problems having 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**.*

3 A complete problem for NIPZK

In this section we introduce the *error shifting technique* and use it to obtain the first complete problem for the class of problems admitting non-interactive perfect zero-knowledge (**NIPZK**) proofs. The proof system that we obtain has interesting characteristics, which we discuss later. We start with motivation, and give formal definitions and proofs in Section 3.1.

We describe STATISTICAL DISTANCE FROM UNIFORM (SDU), the **NISZK**-complete problem of [12], and explain why the reduction and the protocol for this problem cannot be applied to **NIPZK**. Instances of SDU are circuits that represent 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 3.1 $\mathrm{SDU} \overset{def}{=} \langle \mathrm{SDU}_Y, \mathrm{SDU}_N \rangle$, *where*

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

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

The reduction of [12] (based on [30]) reduces any **NISZK** problem Π to SDU through a sequence of reductions. The part of this reduction that we modify is as follows. Let x be an instance of Π and let $\langle P, V \rangle$ be a **NISZK** proof for Π with a simulator S. The instance x is reduced to a circuit X which executes $S(x)$ and obtains a transcript. The 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 `YES` instance, then the simulated reference string is almost uniformly distributed, and thus X is a `YES` instance of SDU. Conversely, if x is a `NO` instance, then the verifier rejects on most reference strings, and thus X is a `NO` instance of SDU.

When we apply the reduction of [12] to **NIPZK** problems Π, and x is a `YES` instance, the output of S perfectly simulates the reference string. Thus, we expect to obtain a circuit X that represents the uniform distribution. However, if Π does not have perfect completeness, then the verifier may reject x, which skews the distribution represented by X. This will cause problems later, when we try to construct a proof system and a simulator for the complete problem. We overcome this issue using the *error shifting technique*.

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 very loose, but we chose it because our technique can be applied in different contexts, and in each of these contexts it takes a different form. The following application will clarify our technique.

► **The first step of the error shifting technique** is to identify where the simulation error comes from. In our case, 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. Since we need to shift this error forward, we first separate it by adding an extra output bit to the circuit X. That is, X executes the simulator and 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 for our complete problem. This step is not trivial because we cannot just use the protocol of [12] for SDU. Specifically, in this protocol the prover sends a string r, and the verifier accepts if $X(r)$ equals the reference string. A simple analysis can show that even if we adapt this idea to our modified circuit, 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 shifts the error from X to the completeness error of the new protocol. We make this intuition formal in the next section.

3.1 A complete problem for NIPZK

In this section we formalize the intuition given in the previous section, thus proving that UNIFORM is **NIPZK**-complete. Our proof system has interesting characteristics, which we discuss after proving that UNIFORM is hard for **NIPZK**.

Theorem 3.2 UNIFORM (UN) *is* **NIPZK**-complete.

Recall that instances of UNIFORM are circuits X. Essentially, as a YES instance X represents the uniform distribution, and as a NO instance X has a small range. However, recall that X also has an extra rightmost output bit. To formally describe these properties, we use the convention that $n+1$ denotes the number of output bits of X. We use T_X to denote the outputs of X that end with the bit 1. Formally, $T_X \stackrel{\text{def}}{=} \{x1 | \exists r \text{ s.t. } X(r) = x1\}$, where $x1$ denotes the concatenation of the string x with the bit 1. Also, we use X' to denote the distribution on the n bit prefix of the output of X. That is, X' is obtained by picking a random input r, computing $X(r)$, and taking the n bit prefix of $X(r)$. As we shall see, when X is a YES instance of UNIFORM, the zero-knowledge and completeness properties would imply that T_X is large and X' is the uniform distribution. Conversely, when X is a NO instances of UNIFORM, the soundness property would imply that $|T_X|$ is small.

The problem UNIFORM is defined in terms of T_X and X'. Formally, given a circuit X with $n+1$ output bits, we say that X is *β-negative* if $|T_X| \leq \beta \cdot 2^n$. That is, T_X has at most $\beta \cdot 2^n$ elements. We say that X is *α-positive* if X' is the uniform distribution on n bits and $\Pr_r[X(r) \in T_X] \geq \alpha$. This notion is not symmetric to that of β-negative, but it does imply that T_X has at least $\alpha \cdot 2^n$ elements.

Definition 3.3 *The problem* UNIFORM *is defined as* $\text{UN} \stackrel{\text{def}}{=} \langle \text{UN}_\text{Y}, \text{UN}_\text{N} \rangle$*, where*

$$\begin{aligned} \text{UN}_\text{Y} &= \{X | X \textit{ is } 2/3-\textit{positive}\}, \textit{ and} \\ \text{UN}_\text{N} &= \{X | X \textit{ is } 1/3-\textit{negative}\}. \end{aligned}$$

The constants $2/3$ and $1/3$ come from the underlying completeness and soundness errors, and as we mentioned in Section 2, these can be obtained from the definitions using repetition.

Proceeding to the completeness result, we recall that proving that a problem is complete for a given class requires proving that the problem is hard for the class (that is, any problem in the class reduces to this problem) and that it is in the class. Thus, we first show that the reduction from the previous section reduces every **NIPZK** problem to UNIFORM.

Lemma 3.4 UNIFORM *is* **NIPZK**-hard.

Proof: Let $\Pi = \langle \Pi_\text{Y}, \Pi_\text{N} \rangle$ 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 $x \in \Pi_\text{Y} \cup \Pi_\text{N}$. Fix a simulator S for $\langle P, V \rangle$. Let $\ell \in \mathbb{N}$, and let S' denote a circuit that on input $x \in \Pi_\text{Y} \cup \Pi_\text{N}$ and randomness r_S of length $|x|^\ell$ outputs $S'(r_S) \stackrel{\text{def}}{=} S(x; r_S)$.

We show that Π Karp reduces to UNIFORM. That is, we define a polynomial-time Turing machine that on input $x \in \Pi_\text{Y} \cup \Pi_\text{N}$ outputs a circuit $X : \{0,1\}^{|x|^\ell} \to \{0,1\}^{|x|^i+1}$ such that if $x \in \Pi_\text{Y}$, then $X \in \text{UN}_\text{Y}$, and if $x \in \Pi_\text{N}$, then $X \in \text{UN}_\text{N}$. On input r_S of length $|x|^\ell$ the circuit X executes $S'(r_S)$ and obtains $S(x; r_S) = \langle x, r'_I, m' \rangle$. If $V(x, r'_I, m') = \texttt{accept}$, then X outputs the string $r'_I 1$ (i.e., the concatenation of r'_I and 1), and otherwise it outputs $r'_I 0$.

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 $r_I 1 \in T_X$, then there is an input r_S to X such that $X(r_S) = r_I 1$. 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 $r_I 1 \notin T_X$, then P^* fails. Notice that P^* makes V accept on any r_I such that $r_I 1 \in T_X$. Since $|T_X| > 2^{|x|^i}/3$, and since r_I is uniformly chosen in $\langle P^*, V \rangle$, the probability that $r_I 1 \in T_X$ 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. □

It remains to prove that UNIFORM is in **NIPZK**, but before describing our proof system we remark that it exhibits some unusual characteristics. The first characteristic is that the prover and the verifier are actually constructed based on the simulator. The second characteristic follows from the first, namely, it is possible that on `YES` instances there are messages that the prover can send, which will make the verifier accept, but instead the prover is sending to the verifier a message on which the verifier rejects. To the best of our knowledge, such characteristics have never been explicitly observed before in the literature.

Lemma 3.5 UNIFORM *has a* **NIPZK** *proof with a deterministic verifier.*

Proof: Our prover and verifier for UNIFORM are based on the simulator, but describing the simulator before the proof is somewhat counter intuitive. Thus, we start with the proof. 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 $X(r) = r_I 1$, 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, $|T_X| \le 1/3 \cdot 2^n$. 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 $1/3$. $\square$

Theorem 3.2 follows from Lemmas 3.4 and 3.5.

4 A hard problem for public-coin PZK proofs

In this section we give the first hard problem for the class of problems admitting public-coin, honest-verifier perfect zero-knowledge (**HVPZK**) proofs. This problem was used in [19] to study the round complexity of perfect zero-knowledge proofs and to prove an equivalence between zero-knowledge and instance-dependent commitment schemes. Notice that since **PZK** $\subseteq$ **HVPZK**, our problem is also hard for public-coin **PZK** proofs.

Starting with motivation, we describe STATISTICAL-DISTANCE (SD), the complete problem of [26] for **SZK**. Instances of this problem are pairs $\langle X, Y\rangle$ of circuits. As YES instances, X and Y represent statistically close distributions, and as NO instances, X and Y are represent statistically far distributions. More generally, $\mathrm{SD} \overset{\text{def}}{=} \mathrm{SD}^{\frac{1}{3},\frac{2}{3}}$, where $\mathrm{SD}^{\alpha,\beta}$ is defined as follows

Definition 4.1 $\mathrm{SD}^{\alpha,\beta} \overset{\text{def}}{=} \langle \mathrm{SD}^{\alpha,\beta}_{\mathrm{Y}}, \mathrm{SD}^{\alpha,\beta}_{\mathrm{N}}\rangle$*, where*

$$\begin{aligned} \mathrm{SD}^{\alpha,\beta}_{\mathrm{Y}} &= \{\langle X_0, X_1\rangle \mid \Delta(\mathrm{X_0},\mathrm{X_1}) \le \alpha\}\,, \textit{ and} \\ \mathrm{SD}^{\alpha,\beta}_{\mathrm{N}} &= \{\langle X_0, X_1\rangle \mid \Delta(\mathrm{X_0},\mathrm{X_1}) \ge \beta\}\,. \end{aligned}$$

We remark that SD and $\overline{\mathrm{SD}}$ are referred to in the literature as the same problem because both of them are complete for **SZK** and reduce to each other. The reduction of [26] (based on [8, 1]) takes any problem that admits a public-coin, honest-verifier statistical zero-knowledge (**HVSZK**) proof and reduces it to SD. The issue with this reduction is that, when we apply it to the class of problems admitting public-coin, honest-verifier perfect zero-knowledge (**HVPZK**) proofs, we get a pair of circuits $\langle X_0, X_1\rangle$ that, as YES instances, 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 $\mathrm{SD}^{0,\frac{1}{2}}$.

Definition 4.2 $\mathrm{SD}^{0,\frac{1}{2}} \overset{\text{def}}{=} \langle \mathrm{SD}^{0,\frac{1}{2}}_{\mathrm{Y}}, \mathrm{SD}^{0,\frac{1}{2}}_{\mathrm{N}}\rangle$*, where*

$$\begin{aligned} \mathrm{SD}^{0,\frac{1}{2}}_{\mathrm{Y}} &= \{\langle X_0, X_1\rangle \mid \Delta(\mathrm{X_0},\mathrm{X_1}) = 0\}\,, \textit{ and} \\ \mathrm{SD}^{0,\frac{1}{2}}_{\mathrm{N}} &= \{\langle X_0, X_1\rangle \mid \Delta(\mathrm{X_0},\mathrm{X_1}) \ge \frac{1}{2}\}. \end{aligned}$$

In the next section we describe the reduction to SD in more detail and show that, essentially, $\mathrm{SD}^{0,\frac{1}{2}}$ is hard for the class of problems admitting public-coin **HVPZK** proofs.

4.1 A hard problem for public-coin HVPZK proofs

We show that, essentially, $\mathrm{SD}^{0,\frac{1}{2}}$ is hard for the class of problems admitting public-coin **HVPZK** proofs. This is done by applying the error shifting technique to the reduction of [26], which we now describe.

Let Π be a problem with a public-coin **HVPZK** proof $\langle P, V\rangle$ and 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 $\langle x, m_1, r_1, \dots, m_v\rangle$. The reduction of [26] maps instances x of Π to pairs of circuits $\langle X', Y'\rangle$. 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 $\langle x, m_1, r_1, \dots, m_v\rangle$, and outputs $\langle m_1, r_1, \dots, m_i, r_i\rangle$. 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 $\langle x, m_1, r_1, \dots, m_v\rangle$. Output $\langle m_1, r_1, \dots, m_i, r_i\rangle$.
- $Y_i(r, r'_i)$: execute $S(x; r)$ to obtain $\langle x, m_1, r_1, \dots, m_v\rangle$. Output $\langle m_1, r_1, \dots, m_i, r'_i\rangle$.

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. Using $\otimes$ to denote the concatenation of circuits, let $X = X_1 \otimes \cdots \otimes X_v$. That is, X executes all the circuits X_i and outputs the concatenation of their outputs. Similarly, let $Y = Y_1 \otimes \cdots \otimes Y_v$. Again, X and Y are identically distributed when x is a YES instance. Now, the pair $\langle X', Y'\rangle$ 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 outputs 1 if with high probability $S(x)$ outputs accepting transcripts, and 0 otherwise. Notice that Z can achieve this by running independent executions of $S(x)$ and estimating the probability that $S(x)$ output an accepting transcript.

The reduction of [26] does not apply to public-coin **HVPZK** proofs because on YES instances x it is possible that V rejects x, which would make the circuit Z output 0 with non-zero probability, and this leads to a non-zero statistical distance between X' and Y'. We overcome this issue using the error shifting technique. Recall that the first step of the error shifting technique is to identify where the simulation error comes from. In this case, the error comes from the circuit Z. Since we need to shift this error forward, instead of including Z in the circuits X' and Y', we separate the error and map instances x of Π to triplets $\langle X, Y, Z\rangle$. This leads to the following definition.

Definition 4.3 *The problem* IDENTICAL DISTRIBUTIONS *is defined as* $\mathrm{ID} \overset{\textit{def}}{=} \langle \mathrm{ID_Y}, \mathrm{ID_N}\rangle$, *where*

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

By the above discussion, 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, by the simulator analysis from [26] (c.f. [8, 1, 25, 17]), 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. Thus, IDENTICAL DISTRIBUTIONS is hard for the class of problems admitting public-coin **HVPZK** proofs.

Lemma 4.4 *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* $\Delta(\mathrm{X}, \mathrm{Y}) = 0$ *and* $\Pr[Z = 1] \geq 2/3$.

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

The second step of the error shifting technique is to shift the error forward, to the completeness or soundness error of the protocol. However, we do not have a **HVPZK** proof for IDENTICAL DISTRIBUTIONS, and even $\mathrm{SD}^{0,\frac{1}{2}}$ is not known to have one (this was an open question in [26]). Thus, we show that given an arbitrary zero-knowledge protocol for $\mathrm{SD}^{0,\frac{1}{2}}$, the error can be shifted from the circuit Z to this protocol. In particular, this shows that any perfect zero-knowledge (**PZK**) proof for $\mathrm{SD}^{0,\frac{1}{2}}$ is a **PZK** proof for IDENTICAL DISTRIBUTIONS. Furthermore, we will preserve all the properties of the original protocol.

The error is shifted as follows. Let $\langle P, V\rangle$ be an arbitrary zero-knowledge protocol for $\mathrm{SD}^{0,\frac{1}{2}}$. We construct a new protocol $\langle P', V'\rangle$ on instances $\langle X, Y, Z\rangle$ of ID (instead of a pair $\langle X, Y\rangle$ of $\mathrm{SD}^{0,\frac{1}{2}}$). We let $P' = P$ and define V' just like V, except that before the protocol begins, V' estimates the value of $\Pr[Z=1]$ and rejects if this value is at most $1/3$. If V' did not reject, then P' and V' execute $\langle P, V\rangle$ on input $\langle X, Y\rangle$. Analyzing this protocol is straightforward. Notice that V' is very unlikely to reject if $\Pr[Z=1] \geq 2/3$, and that if the protocol continues, then either $\langle X, Y, Z\rangle$ is a `YES` instance of our hard problem and $\Delta(\mathrm{X},\mathrm{Y}) = 0$, or $\langle X, Y, Z\rangle$ is a `NO` instance of our hard problem and $\Delta(\mathrm{X},\mathrm{Y}) \geq 1/2$. Hence, 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 $\mathrm{SD}^{0,\frac{1}{2}}$. This shows that, essentially, $\mathrm{SD}^{0,\frac{1}{2}}$ is hard for **public-coin-HVPZK**. In particular, if $\langle P, V\rangle$ is a public-coin **HVPZK** proof for $\mathrm{SD}^{0,\frac{1}{2}}$, then $\langle P', V'\rangle$ is a public-coin **HVPZK** proof for IDENTICAL DISTRIBUTIONS. This would imply that our hard problem is actually complete for this class.

5 Applications

Our error shifting technique and hard problem were used in [19] to study perfect zero-knowledge proofs. In this section we show two additional applications of our results. The first one shows an equivalence between two notions of simulation. The second shows that, under certain conditions, non-interactive perfect zero-knowledge (**NIPZK**) proofs are closed under the `OR` operator.

5.1 Obtaining simulators that do not fail

Zero-knowledge protocols have been defined in the literature with respect to simulators that are either allowed or not allowed to fail (also known as *abort*). We show that these notions are equivalent with respect to any fixed verifier.

We first recall that the definitions of zero-knowledge used in this paper (Definitions 2.5 and 2.6) require that the output of the simulator be "close" to the view of the verifier. A relaxation of this notion due to [6] allows the simulator to fail with probability at most $\frac{1}{2}$, and requires that, conditioned on non-failure, the output of the simulator be "close" to the view of the verifier. Notice that the constant $\frac{1}{2}$ is arbitrary as any non-negligible error probability can be reduced via repetition. The formal definition follows.

Definition 5.1 (Zero-knowledge protocols with simulators that can fail) *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^* *the following holds:*

1. *There is* $N \in \mathbb{N}$ *such that for all* $x \in \Pi_Y$ *with* $|x| \geq N$ *it holds that* $\Pr[S^{V^*}(x) = \mathtt{fail}] \leq \frac{1}{2}$*, where the probability is over the randomness of* S *and* V^*.

2. *Letting $\hat{S}^{V^*}(x)$ denote the distribution on the output of $S^{V^*}(x)$ conditioned on $S^{V^*}(x) \neq \texttt{fail}$, the following distributions are statistically-identical (respectively, statistically indistinguishable, computationally indistinguishable)*

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

It is well known that in the statistical and the computational settings, a simulator S that is allowed to fail can be converted to a simulator S' that is not allowed to fail. On common input x this can be done simply by running $|x|$ executions of $S(x)$, each with a fresh random input, and outputting the first non-fail output. If all executions fail, then $S(x)$ simply outputs `null`, but since this happens with probability at most $1/2^n$, the error that the `null` message introduces into the simulation is negligible. Thus, $S(x)$ is indistinguishable from the view of the verifier. Clearly, this simple idea does not apply to the perfect setting. In fact, since the simulation error is increased, this idea suggests that perhaps by allowing the simulator to fail, the prover may leak some knowledge to the verifier. By using the error shifting technique, we overcome this issue and show that the two notions of simulation are equivalent. In the following lemma we only consider the interactive setting because the same idea applies to the non-interactive setting.

Lemma 5.2 *For any* fixed *malicious verifier V^*, a problem Π has a perfect zero-knowledge proof according to Definition 5.1 if and only if it has a perfect zero-knowledge proof according to Definition 2.6.*

Proof: Trivially, Definition 2.6 implies Definition 5.1. In the forward direction, let Π be a problem with a perfect zero-knowledge protocol $\langle P, V\rangle$ and a simulator S that fails with probability at most $\frac{1}{2}$. The first step of the error shifting technique is to identify where the error is coming from and isolate it. In this case, the error comes from the failure probability of the simulator, and it is already separated from the output of the simulator. Hence, we proceed to the next step of the error shifting technique. That is, we shift the error into the protocol.

On input x we define a new prover P' whose first step is to run $|x|$ executions of $S(x)$. If $S(x)$ fails in all $|x|$ executions, then $P'(x)$ sends `null` to the verifier V and the protocol terminates. Otherwise, it behaves just like $P(x)$. The new simulator S' for $\langle P', V\rangle$ is modified to run $|x|$ executions of the original simulator $S(x)$. If all executions fail, then just like P', it sends `null` to the verifier V and the protocol terminates. Otherwise, one of the outputs of $S(x)$ is not `fail`, and S' outputs the first such non-fail output.

We analyze the new simulator S'. Fix a malicious verifier V^* and consider all sufficiently long $x \in \Pi_Y$. The first observation is that $S'^{V^*}(x)$ never fails. The second observation is that both P' and S'^{V^*} send to V^* the message `null` with the same probability. Conditioned on S' not sending this message, the output of $S'^{V^*}(x)$ is identically distributed to the output of $S^{V^*}(x)$, which, by Definition 5.1, is identically distributed to the view $\langle P, V^*\rangle(x)$ of the malicious verifier. Conditioned on P' not sending the `null` message, $\langle P', V^*\rangle(x)$ and $\langle P, V^*\rangle(x)$ are identically distributed because P' behaves just like P. Thus, $S^{V^*}(x)$ and $\langle P', V^*\rangle(x)$ are identically distributed. We conclude that the two notions are equivalent. $\square$

The above proof explains why we need to fix V^* in advance. That is, V^* is part of S', which in our proof is part of P, and since the prover has to be fixed, so does V^*. Notice that from a practical perspective, the privacy of the prover can be guaranteed only if the simulator S' can simulate any malicious verifier V^*, not just a fixed V^*. Our lemma does not provide this guarantee, but it does assure that once the malicious verifier V^* is fixed, the privacy of the prover is guaranteed even if the simulator is allowed to fail. In particular, Definitions 2.6 and 5.1 are equivalent with respect to the honest verifier V.

5.2 Under certain restrictions NIPZK is closed under the `OR` operator

In this section we show that **NIPZK** is closed under the OR operator. Unfortunately, we could only prove this lemma by making a strong condition on the soundness and completeness error of the underlying problem. However, the issues that we encounter illuminate the difficulties of working with perfect zero-knowledge proofs, and we believe that exploring new ideas, even if we have to make strong assumptions as a first step, is useful. We mention that no such closure result is known in the case of non-interactive statistical zero-knowledge (**NISZK**) proofs (c.f., [30, 12]).

5.2.1 Background

We first recall that a complexity class $\mathcal{C}$ is closed under the OR operator (denoted $\vee$) if for any two problems $\Pi, \Gamma \in \mathcal{C}$ it holds that $\Pi \vee \Gamma \in \mathcal{C}$, where $\Pi \vee \Gamma \overset{\text{def}}{=} \langle (\Pi \vee \Gamma)_Y, (\Pi \vee \Gamma)_N \rangle$,

$$\begin{aligned} (\Pi \vee \Gamma)_Y &= \{\langle x, y\rangle | x \in \Pi_Y \vee y \in \Gamma_Y\}\ , \text{ and} \\ (\Pi \vee \Gamma)_N &= \{\langle x, y\rangle | x \in \Pi_N \wedge y \in \Gamma_N\}\ . \end{aligned}$$

Since we are working with promise-problems, this definition only considers elements that are YES or NO instances of Π and Γ. Thus, in the definition of $(\Pi \vee \Gamma)_Y$, when one of x, y is not a YES instance, the intention is that this element is a NO instance of either Π or Γ.

To show that **NIPZK** is closed under the OR operator, we need to show a non-interactive protocol where the prover and the verifier are given two instances x of Π and y of Γ, and the prover gives a perfect zero-knowledge proof that convinces the verifier that at least one of x, y is a YES instance. Since we have a **NIPZK**-complete problem, both Π and Γ reduce to it. This simplifies our task because we only need to deal with circuits X of UNIFORM (UN). That is, the prover and the verifier are given $X, Y \in \text{UN}_Y \cup \text{UN}_N$, and the verifier should accept only if X or Y is a YES instance. Otherwise, the verifier should reject.

The first challenge is that no protocol for OR closure is known for the class of problems admitting non-interactive statistical zero-knowledge (**NISZK**) proofs. In contrast to our previous results, where we could build on ideas from the statistical setting, here we have no such advantage. Thus, we try the natural approach of building a proof based on the difference between YES and NO instances.

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. This suggests that we should construct a **NIPZK** proof where the verifier accepts or rejects based on sizes of sets. Indeed, this idea proved useful in the study of statistical zero-knowledge proofs [22, 13]. Specifically, the lower-bound protocol of Goldwasser and Sipser [14] (c.f. [31, 2, 1]) was applied by Goldreich and Vadhan [13] to bound sets defined from circuits (c.f. [22]). Roughly speaking, given a circuit T, the idea is to use a random string (such as the reference string) to define a hash function h, and let the prover find y such that both $h(y) = 0$ and $y \in T$. The verifier accepts only if such y exists. By the properties of the hash function, for most random strings h it holds that $h^{-1}(0)$ is a random set of more or less the same size. Thus, when T is large, $h^{-1}(0) \cap T$ is non-empty, and the verifier accepts. Conversely, when T is small, $h^{-1}(0) \cap T$ is empty, and the verifier rejects. This protocol can also be simulated. Specifically, the simulator first chooses a random sample $y \in T$, and then it chooses h uniformly, conditioned on $h(y) = 0$.

When we apply the lower-bound protocol in our case, the common reference string would define h, and the prover would choose r_X and r_Y such that $h(X(r_X), Y(r_y)) = 0$. The simulator would do the same, but in reverse order. That is, the simulator would first choose r_X and r_Y, compute $X(r_X)$ and (r_X), and finally choose h such that $h(X(r_X), Y(r_y)) = 0$. The issue with this simulator is that it defines the common reference string from h, and h may be only statistically close to uniform. This would only add a small error to the simulation, which is why hashing techniques are useful in the study of statistical zero-knowledge. However, since perfect zero-knowledge protocols allow no error in the simulation, we cannot use this technique for **NIPZK**.

5.2.2 The protocol.

We describe our 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 $Y \in \mathrm{UN_Y}$. We can assume that both circuits have the same number of output bits because we can always add input gates that are mapped directly to output gates. This retains the `YES` or `NO` properties of the circuit. We denote the number of output bits in X and Y by $n + 1$.

Since our main obstacle is achieving perfect zero-knowledge, we take a rather counter intuitive approach, starting with the simulator. Consider a simulator that uniformly picks r_X and r_Y, and computes $z = X(r_X) \oplus Y(r_Y)$. Since at least one of the circuits is a `YES` instance, the output of this circuit is uniformly distributed. Hence, the simulator uses the n-bit prefix of z to define the reference string. This simulator informs a prover who, on reference string r_I, sends r_X and r_Y to the verifier such that the n-bit prefix of $X(r_X) \oplus Y(r_Y)$ equals r_I. Intuitively, the simulator demonstrates that by XOR-ing the outputs of the circuits, the prover does not leak to the verifier which of the circuits is a `YES` instance.

The challenge now is how to define the verifier so that completeness and soundness are satisfied. We start by defining a verifier that accepts only if the rightmost 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, but even when both X and Y are `NO` instances, there could be many combinations for $X(r_X) \oplus X(r_Y)$. That is, given reference string 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 the rightmost bit of both $X(r_X)$ and $Y(r_Y)$ is 1. This will violate the soundness property. Since we do not know how to overcome this issue, we restrict the number of such combinations. Turning our attention to the completeness property we observe that if one of the circuits is a `NO` instance of UN, then it is possible that all of the strings outputted by this circuit have 0 as their rightmost bit (e.g, for any r_X the rightmost bit 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 $\mathrm{UN_Y}$ be 1-positive. That is, `YES` instances always have 1 as the rightmost bit of their outputs. Notice that the output of `NO` instances can also have 1 as the rightmost bit. However, by requiring that `YES` instances be 1 positive, we help the simulator identify `NO` instances (because only `NO` instances can output a string whose rightmost bit is 0).

We redefine the simulator and the proof based on the above restriction. That is, 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, the rightmost bit of one of the samples is 0. For example, suppose this sample is $X(r_X)$. This implies that Y is a `YES` instance. In this case the simulator defines the reference string based on the n bit prefix of $Y(r_x)$. To reflect this modification in the proof, upon receiving $\langle r_X, r_Y\rangle$ from the prover, the verifier accepts if the n-bit prefix of $Y(r_Y)$ equals the reference string and $Y(r_Y)$ ends with a 1. The above intuition is formalized in the following lemma.

Lemma 5.3 *Let Π and Γ be* **NIPZK** *problems. Consider the reduction from these problems to instances of* UNIFORM*, and denote by X circuits with $n+1$ output bits obtained by the reduction. If as `YES` instances the circuits X are 1-positive, and as `NO` instances the circuits X are $2^{-(1+n/2)}$ negative, then $\Pi \vee \Gamma \in$***NIPZK**.

Proof: Let $\langle x_0, x_1\rangle$ such that $x_i \in \Pi_\mathrm{Y} \cup \Pi_\mathrm{N}$ for each $i \in \{0, 1\}$. We construct a **NIPZK** protocol $\langle P, V\rangle$ for $\Pi \vee \Gamma$. Initially, P sets $i = 0$ if both x_0 and x_1 are `YES` instances. Otherwise, there is a unique i such that $x_{\bar{i}}$ is a `NO` instance, 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. Notice that X_0 and X_1 may have a different number of output gates. This number depends on several parameters, such as the length of x_0 and x_1. However, we can easily make this number equal. Specifically, if X_0 has n output gates and X_1 has $n - a < n$ output gates, then we define a new circuit X_1' by adding a input gates to X_1. The output of X_1' is the values on these a input gates, followed by the output of X_1. This modification guarantees that both circuits have $n + 1$ output gates and that X_1' inherits the properties of X_1. That is, for any α and β, if X_1 is α-positive, then X_1' is α-positive, and if X_1 is β-negative, then X_1' is β-negative.

Thus, for each $i \in \{0, 1\}$ the circuit X_i has $n + 1$ output gates and, by the premise, the following properties hold. If x_i is a `YES` instance of either Π or Γ, then the rightmost bit in the outputs of X_i is 1, and X_i' (the distribution on the n bit prefix of X_i) is uniform on $\{0, 1\}^n$. Conversely, if x_i is a `NO` instance of either Π or Γ, then $|T_{X_i}| \leq 2^{-(1+n/2)} \cdot 2^n = 2^{-1+n/2}$ (recall that T_X is the set of outputs of X whose rightmost bit is 1).

After the prover sets i, the protocol proceeds as follows. The first step of P is to uniformly choose a string $r_{\bar{i}}$, and assign y the output of $X_{\bar{i}}(r_{\bar{i}})$, excluding the rightmost bit. Let r_I denote the reference string. If $X_{\bar{i}}(r_{\bar{i}}) = y0$, then P uniformly chooses $r_i \in X_i^{-1}(r_I 1)$, and sends $\langle r_0, r_1 \rangle$ to V. Otherwise, $X_{\bar{i}}(r_{\bar{i}}) = y1$, in which case P uniformly chooses $r_i \in X_i^{-1}(y1 \oplus r_I 1)$, 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 the rightmost bit of $X_{\bar{i}}(r_{\bar{i}})$ is 0 and $X_i(r_i) = r_I 1$, 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 $X_1(r_1)$ have 1 as the rightmost bit), then V accepts. Otherwise, V rejects.

The completeness property of $\langle P, V \rangle$ follows from its zero-knowledge property. Thus, we start with 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., $X_i \in \mathrm{UN_N}$), then S outputs $\langle \langle x_0, x_1 \rangle, r_I', \langle r_0, r_1 \rangle \rangle$, where r_I' is the n-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' is the n-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 the rightmost bit of $X_i(r_i)$ is 0, and $X_{\bar{i}}(r_{\bar{i}}) = r_I 1$. Since $|T_{X_{\bar{i}}}| \leq 2^{n/2-1}$, and r_I is uniformly distributed, it follows that in the first case V accepts with probability at most $2 \cdot \Pr_{r_I}[X_{\bar{i}}(r_{\bar{i}}) = r_I 1] \leq 2 \cdot 2^{-(n/2+1)}$. Notice that the probability is multiplied by 2 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 $|T_{X_0}| \cdot |T_{X_1}| \leq 2^{n/2-1} \cdot 2^{n/2-1} = 2^n/4$ and r_I is uniformly distributed. We conclude that in total V accepts with probability at most $1/4 + 2 \cdot 2^{-(n/2+1)}$, which is $1/3$ for sufficiently large inputs. The lemma follows. $\square$

6 Conclusion

Our research was motivated by the fact that there are many general results about statistical zero-knowledge proofs, but none of them applies to perfect zero-knowledge proofs. Consequently, many fascinating fundamental questions about perfect zero-knowledge proofs remain open. For example, can we collapse the number of rounds in perfect zero-knowledge proofs to a constant? Do perfect and statistical zero-knowledge coincide (this question motivated the thesis of [34]), and under what implications to complexity theory?

Perhaps we are still far from answering fundamental questions, but prior to this paper, virtually nothing was known about perfect zero-knowledge proofs. In this paper we provided complete and hard problems. We introduced the error shifting technique, applied it in several places, and showed how it can be used to achieve perfect simulation. We took one step further and showed two applications, one to the notion of simulation and the other to the closure of **NIPZK**. In doing so, we improved the current understanding of perfect zero-knowledge proofs, provided tools for future work in this area, and illuminated the difficulties lying ahead. We hope that this insight will facilitate the study of perfect zero-knowledge proofs.

References

[1] 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.

[2] László Babai. Trading group theory for randomness. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 421–429, New York, NY, USA, 1985. ACM.

[3] Mihir Bellare and Phillip Rogaway. Noninteractive perfect zero-knowledge. Unpublished manuscript, June 1990.

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

[5] 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.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 399–408, New York, NY, USA, 1998. ACM.

[12] 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 '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 467–484, London, UK, 1999. Springer-Verlag.

[13] 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.

[14] 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.

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

[16] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *Proceedings of Eurocrypt 2006, volume 4004 of LNCS*, pages 339–358. Springer, 2006.

[17] Iftach Haitner, Omer Reingold, Salil P. Vadhan, and Hoeteck Wee. Inaccessible entropy. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 611–620. ACM, 2009.

[18] 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.

[19] Lior Malka. Instance-dependent commitment schemes and the round complexity of perfect zero-knowledge proofs. *Electronic Colloquium on Computational Complexity (ECCC)*, 15(068), 2008.

[20] Silvio Micali and Rafael Pass. Local zero knowledge. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 306–315, New York, NY, USA, 2006. ACM.

[21] 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.

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

[23] Shien Jin Ong and Salil P. Vadhan. An equivalence between zero knowledge and commitments. In Ran Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*. Springer, 2008.

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

[25] Erez Petrank and Gábor Tardos. On the knowledge complexity of NP. In *FOCS*, pages 494–503, 1996.

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

[27] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. The knowledge complexity of quadratic residuosity languages. *Theor. Comput. Sci.*, 132(1-2):291–317, 1994.

[28] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Randomness-efficient non-interactive zero-knowledge (extended abstract). In *Automata, Languages and Programming*, pages 716–726, 1997.

[29] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. On NC^1 boolean circuit composition of non-interactive perfect zero-knowledge. In *MFCS*, pages 356–367, 2004.

[30] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. Image density is complete for non-interactive-SZK (extended abstract). In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *Automata, Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark, July 13-17, 1998, Proceedings*.

[31] Michael Sipser. A complexity theoretic approach to randomness. In *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 330–335, New York, NY, USA, 1983. ACM.

[32] Martin Tompa and Heather Woll. Random self-reducibility and zero-knowledge interactive proofs of possession of information. In *FOCS '87: 28th Annual Symposium on Foundations of Computer Science, 12-14 October 1987, Los Angeles, California, USA*, pages 472–482. IEEE, 1987.

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

[34] Salil P. Vadhan. On transformation of interactive proofs that preserve the prover's complexity. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 200–207, New York, NY, USA, 2000. ACM.

[35] John Watrous. Zero-knowledge against quantum attacks. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 296–305, New York, NY, USA, 2006. ACM.