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

Abstract. We study perfect zero-knowledge proofs (**PZK**). Unlike statistical zero-knowledge, where many fundamental questions have been answered, virtually nothing is known about these proofs.

We consider reductions that yield hard and complete problems in the statistical setting. The issue with these reductions is that they introduce errors into the simulation, and therefore they do not yield analogous problems in the perfect setting. We overcome this issue using an *error shifting technique*. This technique allows us to remove the error from the simulation. Consequently, we obtain 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 **PZK** proofs.

We get the following applications. Using the error shifting technique, we show that the notion of zero-knowledge where the simulator is allowed to fail is equivalent to the one where it is not allowed to fail. Using our complete problem, we show that under certain restrictions **NIPZK** is closed under the `OR` operator. Using our hard problem, we show how a *constant-round, perfectly hiding* instance-dependent commitment may be obtained (this would collapse the round complexity of public-coin **PZK** proofs to a constant).

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

1 Introduction

Zero-knowledge protocols allow one party (the prover) to prove an assertion to another party (the verifier), yet without revealing anything beyond the validity of the assertion [18, 5]. These protocols protect the privacy of the prover, which makes them very useful to cryptography. Zero-knowledge protocols can guarantee three levels of privacy: perfect, statistical, and computational. This is formulated using the notion of simulation. When the simulation error is zero, the protocol is *perfect zero-knowledge*. This means that the verifier learns absolutely nothing from the prover. When the simulation error is negligible, the protocol is either *statistical zero-knowledge* or *computational zero-knowledge*. This means that the prover leaks a little amount of information to the verifier.

In this paper we focus on perfect zero-knowledge protocols. These protocols are interesting from a cryptographic perspective because, unlike statistical or computational zero-knowledge protocols, they provide the highest level of privacy to the prover. Such protocols exist for a variety of well known languages, such as GRAPH-ISOMORPHISM, DISCRETE-LOG, variants of QUADRATIC-RESIDUOUSITY, and more ([31, 13, 6, 26, 21]). The fact that the complexity of these languages is an open question also makes perfect zero-knowledge protocols interesting from a complexity theoretic perspective.

Unfortunately, working with perfect zero-knowledge protocols is difficult. This is so because they do not allow any error in the simulation. In contrast, statistical zero-knowledge protocols allow a small error in the simulation. This means that in the statistical setting we can use a variety of techniques, even if they introduce a small error into the simulation. Indeed, such techniques were used to prove many fundamental results about statistical zero-knowledge proofs (**SZK**). These results include complete problems, equivalence between private-coin and public-coin, equivalence between honest and malicious verifier, and much more ([24, 26, 14, 16, 32, 23]). These results do not apply to the perfect setting because they use techniques that introduce error into the simulation, and such techniques cannot be used in the perfect setting. Consequently, virtually nothing is known about perfect zero-knowledge proofs (**PZK**).

1.1 Our results

In this paper we consider reductions that yield hard and complete problems in the statistical setting. The issue with these reductions is that they introduce errors into the simulation, and therefore they do not yield analogous problems in the perfect setting.

Our goal is to overcome this issue. This is important because if we understand why techniques from the statistical setting introduce error into the simulation, then we might be able to fix these techniques, and then apply them to the perfect setting. This will enable us to translate the results from statistical zero-knowledge to perfect zero-knowledge. In addition, results from the statistical setting are proved using the tool of complete and hard problems. Thus, to be able to prove these results in the perfect setting it is important that we obtain complete and hard problems for the perfect setting.

We remark that we are not the first to observe the fact that reductions from the statistical setting do not apply to the perfect setting. Specifically, in the case of hard problems, [26] showed that the reduction could eliminate the error using approximation techniques. However, this solution does not yield a hard problem in the perfect setting because it only applies in certain cases (for example, when the underlying problem has perfect completeness).

In this paper we modify the reductions from the statistical setting so that they yield hard and complete problems in the perfect setting. To do this we use what we call *the error shifting technique*. What is new about this technique is that instead of dealing with the error in the reduction itself, we shift it forward to the protocol. Intuitively, our reduction isolates the error that the underlying problem incurs, and shifts it forward to the protocol, where it is no longer a simulation error. Consequently, we obtain complete and hard problems for the perfect setting. We remark that the error shifting technique is also useful in other contexts (e.g., in Section 4 we use it to show that the notion of zero-knowledge where the simulator is allowed to fail is equivalent to the one where it is not allowed to fail).

The error shifting technique applies to reductions in both the interactive and the non-interactive models. In the non-interactive model we apply it to the reduction of [30, 15], thus obtaining the first complete problem for the class of problems possessing non-interactive perfect zero-knowledge proofs (**NIPZK**).

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

Informally, instances of UNIFORM are circuits that 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. This problem is identical to the **NISZK**-complete problem STATISTICAL DISTANCE FROM UNIFORM (SDU) [15], except that YES instances of UNIFORM represent the uniform distribution, whereas YES instances of SDU represent a distribution that is only "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 [26], 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) [26]. 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 [26] are circuits representing "close" distributions.

Theorem 2 **(informal).** Essentially, $\overline{\mathrm{SD}^{1/2,0}}$ is hard for **public-coin-HVPZK**.

To demonstrate the usefulness of our **NIPZK**-complete problem we prove that under certain restrictions **NIPZK** is closed under the OR operator. What is special about this result is that even in statistical setting, where we have more techniques to work with, it is not clear how to prove (or disprove) it.[1] Also, we show how our hard problem may lead to a *constant-round, perfectly hiding* instance-dependent commitment-scheme. Notice that except for [21], who used the techniques of [8] to construct such a scheme for V-bit protocols, all the known instance-dependent commitment-schemes are only statistically hiding [33, 23, 22]. Thus, using our hard problem it might be possible to collapse the round complexity of public-coin **PZK** proofs to a constant. These applications can be found in Section 4.

[1] [30] claimed that **NISZK** is closed under the OR operator, but this claim has been retracted.

1.2 Related Work

As we mentioned, virtually nothing is known about perfect zero-knowledge proofs. The only exception is the result of [9], who showed a transformation from *constant-round, public-coin* **HVPZK** proofs to ones that are **PZK**. Also, a **HVPZK**-complete problem was given by [26], but it is unnatural, and defined in terms of the class itself. We remark that perfect zero-knowledge *arguments* for **NP** languages have been constructed under various unproven assumptions (e.g., [19, 7]), but we are interested in the unconditional study of perfect zero-knowledge proofs.

Variants of QUADRATIC-RESIDUOUSITY and QUADRATIC-NONRESIDUOUSITY were shown to to be in **NIPZK** by [6, 27]. Bellare and Rogaway [3] showed that a variant of GRAPH-ISOMORPHISM is in **NIPZK**. They also showed 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. Other aspects of **NIPZK** were studied in [27–29], but they apply to problems with special properties.

1.3 Organization

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

2 A complete problem for NIPZK

In this section we introduce the *error shifting technique*. Using this technique we modify the reduction of [15], 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 [15]. 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 2.1. *Define* $\mathrm{SDU} \stackrel{def}{=} \langle \mathrm{SDU}_Y, \mathrm{SDU}_N \rangle$ *as*

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

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 [15] to SDU. This reduction originated from the work of [30]. 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 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.

The issue with the reduction of [15]. 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 [15] 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.

2.1 A complete problem for NIPZK

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

Theorem 2.1. UNIFORM *is* **NIPZK**-complete.

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, $T_X \stackrel{\text{def}}{=} \{x | \exists r \; X(r) = x$, 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 bit, we say that X is β-*negative* if $|T_X| \leq \beta \cdot 2^n$. 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 $\Pr_{x \leftarrow X}[x \in T_X] \geq \alpha$. This implies that T_X is large, and contains at least $\alpha \cdot 2^n$ strings.

Definition 2.2. *The problem* UNIFORM *is defined as* $\mathrm{UN} \stackrel{\text{def}}{=} \langle \mathrm{UN_Y}, \mathrm{UN_N} \rangle$, *where*

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

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

Lemma 2.1. 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 $r'_I 1$ (i.e., the concatenation of r'_I and 1). Otherwise, output $r'_I 0$.

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.

To prove Theorem 2.1 it remains to give a **NIPZK** proof for UN.

Lemma 2.2. 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 $X(r) = r_I 1$, 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 $X \in \mathrm{UN}_{\mathrm{N}}$. 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 $X \in \mathrm{UN}_{\mathrm{N}}$, then V accepts X with probability at most $1/3$.

3 A hard problem for public-coin PZK proofs

In this section we use the error shifting technique to modify the reduction of [26] for public-coin **HVSZK** proofs. Hence, we obtain a hard problem for the class of problems possessing public-coin **HVPZK** proofs (**AM** $\cap$ **HVPZK**). We start with motivation.

The reduction of [26] originated from the works of [11, 1]. Informally, given a problem Π that has a public-coin **HVSZK** proof, this 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 the complement of $\mathrm{SD}^{1/2,0}$.

Definition 3.1. *The problem* $\overline{\mathrm{SD}^{1/2,0}}$ [26] is the pair $\langle \overline{\mathrm{SD}^{1/2,0}}_{\mathrm{Y}}, \overline{\mathrm{SD}^{1/2,0}}_{\mathrm{N}}\rangle$, where

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

Sahai and Vadhan [26] 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 3.2. *The problem* IDENTICAL DISTRIBUTIONS *is* $\mathrm{ID} \stackrel{def}{=} \langle \mathrm{ID_Y}, \mathrm{ID_N}\rangle$*, where*

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

3.1 Modifying the reductions for public-coin HVSZK proofs

In this section we show that ID is hard for **AM** $\cap$ **HVPZK**, and then we conclude that, essentially, $\overline{\mathrm{SD}^{1/2,0}}$ is also hard for **AM** $\cap$ **HVPZK**. Starting with some background, we describe the reduction of [26].

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 $v \stackrel{\text{def}}{=} v(|x|)$ 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 [26] 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 outputs 1 if with high probability $S(x)$ outputs accepting transcripts, and 0 otherwise.

The issue with the reduction of [26]. 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, $\overline{\mathrm{SD}^{1/2,0}}$ is hard for **AM** $\cap$ **HVPZK**

Our first step is to 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 [26], 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 or 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 **AM** $\cap$ **HVPZK**.

Lemma 3.1 (Implicit in [26]). *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* $\Pr[Z = 1] \leq 1/3$*.*

Indeed, instances of ID are triplets of circuits $\langle X, Y, Z \rangle$, as opposed to pairs $\langle X, Y \rangle$. Thus, we are not done yet. We need to show that ID and $\overline{\mathrm{SD}^{1/2,0}}$ 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, $\overline{\mathrm{SD}^{1/2,0}}$ is not known to have a **PZK** proof. Thus, our second step is to modify any **PZK** protocol $\langle P, V \rangle$ for this problem into a **PZK** protocol $\langle P, V' \rangle$ for ID. That is, we take an arbitrary protocol $\langle P, V \rangle$ for $\overline{\mathrm{SD}^{1/2,0}}$, 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 $\overline{\mathrm{SD}^{1/2,0}}$.

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 $\Delta(\mathrm{X}, \mathrm{Y}) = 0$, or $\langle X, Y, Z \rangle$ is a `NO` instance of our hard problem and $\Delta(\mathrm{X}, \mathrm{Y}) \geq 1/2$. 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 $\overline{\mathrm{SD}^{1/2,0}}$.

Our modification shows that although we did not prove that $\overline{\mathrm{SD}^{1/2,0}}$ is hard for **AM** $\cap$ **HVPZK**, it can be treated as such (because any protocol that we design for this problem can be immediately modified to a protocol with the same properties for ID).

4 Applications

We show an application of the error shifting technique. We also show how our complete and hard problems can facilitate the study of zero-knowledge in the perfect setting.

4.1 Obtaining simulators that do not fail

We use the error shifting technique to show that the notion of 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 [18]. 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 [9].

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 [12]. 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 [18, 13, 31]), 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 **PZK** 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`). Thus, the error comes from the fact that P is not teaching 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.

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 $S(x)$ 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.

4.2 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 [17]. 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 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 (See Appendix B for the proof). 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.

4.3 Applications of the AM ∩ HVPZK-hard problem

In this paper we showed that IDENTICAL DISTRIBUTIONS (ID) is hard for the class of problems admitting public-coin **HVPZK** proofs, and that we can treat it as $\overline{\mathrm{SD}^{1/2,0}}$. Unfortunately, our result is restricted to public-coin. In contrast, the reduction of [26] for **HVSZK** (which follows from the works of [11, 1, 25]) is not restricted to public-coin, but it manipulates distributions in a way that skews the distributions, and we do not know how to apply it to **HVPZK**.

However, what is special about ID, and what makes it different from SD, is that its YES instances are pairs of circuits $\langle X_0, X_1 \rangle$ representing identical (as opposed to statistically close) distributions. Thus, they can be used to obtain perfectly (as opposed to statistically) hiding instant-dependent commitment-schemes. Using the observation of [20], such schemes could then be plugged into the protocols for **NP** [4, 13], thus yielding a **HVPZK** proof for ID. We mention that, except for [21], who used the techniques of [8] to construct a perfectly hiding scheme for V-bit protocols, all the known instance-dependent commitment-schemes are only statistically hiding [33, 23, 22].

Notice that if we can use instances $\langle X_0, X_1 \rangle$ of ID to construct a *constant-round*, perfectly hiding, instance-dependent commitment-scheme, then we would collapse the round complexity of public-coin **HVPZK** proofs. One idea for a commitment is to take a sample of X_b. That is, given common input $\langle X_0, X_1 \rangle$, a commitment to a bit b is computed by uniformly choosing r and outputting $X_b(r)$. Thus, on YES instances the scheme is perfectly hiding. However, the scheme may not be binding on NO instances because there could be r and r' for which $X_0(r) = X_1(r')$. Thus, other techniques are needed to make sure that the binding property is achieved.

5 Conclusion

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, we obtained complete and hard problems, and interesting applications. We believe that insight provided here will be useful in the study of perfect zero-knowledge proofs.

References

1. W. Aiello and J. Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *J. of Computer and System Sciences*, 42(3):327–345, June 1991.
2. L. Babai and S. Moran. Arthur-merlin games: A randomized proof system and a hierarchy of complexity classes. *J. of Computer and System Sciences*, 36:254–276, 1988.
3. Mihir Bellare and Phillip Rogaway. Noninteractive perfect zero-knowledge. Unpublished manuscript, June 1990.
4. Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the ICM,pp*, pages 1444–1451, 1986.
5. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge proofs and their applications. In *Proceedings of the 20th STOC, ACM*, 1988.
6. Manuel Blum, Alfredo De Santis, Silvio Micali, and Giussepe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.
7. 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.
8. Ivan Damgård. Interactive hashing can simplify zero-knowledge protocol design without computational assumptions (extended abstract). In *CRYPTO*, pages 100–109, 1993.
9. Ivan Damgård and Oded Goldreich Avi Wigderson. Hashing functions can simplify zero-knowledge protocol design (too). Technical Report RS-94-39, BRICS, November 1994.
10. 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.
11. L. Fortnow. The complexity of perfect zero-knowledge. In Silvio Micali, editor, *Advances in Computing Research*, volume 5, pages 327–343. JAC Press, Inc., 1989.
12. Oded Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.
13. 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.
14. Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *STOC*, pages 399–408, 1998.
15. 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.
16. 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.

17. 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.
18. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
19. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT*, pages 339–358, 2006.
20. Toshiya Itoh, Yuji Ohta, and Hiroki Shizuya. A language-dependent cryptographic primitive. *J. Cryptology*, 10(1):37–50, 1997.
21. 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.
22. 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.
23. 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.
24. Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. *J. Comput. Syst. Sci.*, 60(1):47–108, 2000.
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 *ICALP*, pages 784–795, 1998.
31. M. Tompa and H. Woll. Random self-reducibility and zero-knowledge interactive proofs of possession of information. In *28th FOCS*, pages 472–482, 1987.
32. Salil P. Vadhan. *A study of statistical zero-knowledge proofs*. PhD thesis, MIT, 1999.
33. Salil P. Vadhan. An unconditional study of computational zero knowledge. In *FOCS*, pages 176–185, 2004.

A Preliminaries

We use standard definitions [12]. We study *promise-problems* [10], which are a generalization of languages. Formally, $\Pi \stackrel{\text{def}}{=} \langle \Pi_{\mathrm{Y}}, \Pi_{\mathrm{N}} \rangle$ is a problem if $\Pi_{\mathrm{Y}} \cap \Pi_{\mathrm{N}} = \emptyset$. The set Π_{Y} contains the `YES` instances of Π, and the set Π_{N} contains the `NO` instances of Π. We define $\overline{\Pi} \stackrel{\text{def}}{=} \langle \Pi_{\mathrm{N}}, \Pi_{\mathrm{Y}} \rangle$.

Let $X : \{0,1\}^m \rightarrow \{0,1\}^n$ be a circuit. We treat X both as a circuit and as a distribution (under the convention that the input to the circuit is uniformly distributed). For example, given a set T, the probability $\Pr_{x \leftarrow X}[x \in T]$ equals $\Pr_{r \leftarrow U_m}[X(r) \in T]$, where U_m is the uniform distribution on $\{0,1\}^m$, and $d \leftarrow D$ denotes choosing an element d according to the distribution D. The *statistical distance* between two discrete distributions X and Y is $\Delta(\mathrm{X},\mathrm{Y}) \stackrel{\text{def}}{=} \sum_\alpha |\Pr[X = \alpha] - \Pr[Y = \alpha]|$. We define non-interactive protocols.

Definition A.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 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 $r_I \in \{0,1\}^{|x|^c}$.
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 $\langle P, V\rangle(x) \stackrel{\text{def}}{=} \langle x, r_I, \pi\rangle$ the view *of V on x. We say that V* accepts *x (respectively,* rejects *x) if $m = \texttt{accept}$ (respectively, $m = \texttt{reject}$).*

Definition A.1 considers a deterministic V, and is equivalent to a the definition that considers a probabilistic V [2]. We define non-interactive proofs.

Definition A.2 (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 $c(n), s(n) : \mathbb{N} \rightarrow [0,1]$ 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 $c \equiv 0$.*

We proceed to zero-knowledge. Our definition considers simulators that do not fail, which is justified by our result from Section 4.

Definition A.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***).*

B Under certain restrictions NIPZK is closed under OR

Lemma B.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., $X_i \in \mathrm{PU}_{\mathrm{N}}$), 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.