T-79.511 Special Course on Cryptology / Zero Knowledge: Statistical Zero Knowledge

Petteri Kaski

November 10, 2001

1 Introduction

The concept of an interactive proof protocol for deciding language membership providing zero knowledge to the proof verifier was introduced by Goldwasser, Micali and Rackoff [9]. Informally, an interactive proof protocol is *zero-knowledge* if interacting with the prover does not provide any additional knowledge to the verifier (apart possibly from the validity of the language membership assertion being proven), *regardless* of the strategy used by the verifier in interacting with the prover. In other words, the zero-knowledge property of an interactive proof system is a property of the prover[1] that is measured against all "malicious" verifier strategies aimed at extracting additional knowledge from the interaction.

The hard task in presenting a formal definition for zero-knowledge is quantifying what constitutes an "increase in knowledge" resulting from an interaction. In the model of Goldwasser, Micali and Rackoff, "knowledge" is informally defined as everything computable from the available input within a fixed (randomized) model of computation (e.g. a probabilistic polynomial-time Turing machine). Thus, interaction with the prover results in zero knowledge gain if the result of the interaction was known, that is, computable before the interaction. Computability has to be given a probabilistic interpretation in this context because the prover is randomized. Namely, a verifier strategy interacting with the prover only determines a probability distribution on the set of possible interactions—the random coin tosses made by the prover (on the outcomes of which the verifier is assumed to have no control on) then determine which of the possible interactions actually takes place. In other words, after a verifier strategy has been fixed, the best the verifier can achieve is to randomly sample the induced distribution. Now, if this distribution was available for sampling beforehand, then intuitively no knowledge gain results from interacting with the prover.

Goldwasser, Micali and Rackoff considered three variations of zero knowledge gain of the above kind in their paper:

- ▷ *Perfect* zero knowledge is the strongest of these notions. It quantifies the property that, no matter what computational resources[2] the verifier has and what strategy the verifier uses, an interaction with the prover on a valid assertion results in no additional knowledge to the verifier. More formally, any probability distribution that is induced on the set of possible prover-verifier interactions by a verifier strategy must be perfectly reproducible (s.o. samplable with equal probability) by the verifier alone.
- ▷ *Statistical* zero knowledge slightly relaxes the perfect reproducibility constraint by allowing negligible statistical difference between the distribution induced by a verifier interaction strategy and an approximate distribution that the verifier can sample on its own. Statistical zero knowledge is still a very strong requirement in the sense that even an unbounded computational adversary cannot distinguish

[1] This should be contrasted with the fact that the soundness of an interactive proof system is a property of the verifier that is measured against all "malicious" prover strategies aimed at convincing the verifier of an invalid assertion.

[2] Strictly speaking, the verifier is always assumed to have certain *minimum* computational resources (e.g. probabilistic polynomial time) to guarantee the ability to generate simulated interactions with the prover.

between the two distributions with nonnegligible probability if only a polynomial number of interactions are allowed with the prover.

- ▷ *Computational* zero knowledge is the weakest of these notions. It requires that the approximate distribution is only computationally indistinguishable (by probabilistic polynomial time algorithms, say) from the distribution induced by interaction with the prover.

In this manuscript we focus on statistical indistinguishability and statistical zero-knowledge. The aim is to formally define statistical zero-knowledge proof systems and to illustrate the differences in definition between statistical and other forms of zero-knowledge. The main tool in this discussion will be probability ensembles and notions of indistinguishability (i.e., perfect, statistical, and computational indistinguishability) between ensembles. Material for this part of the survey was mostly obtained from the recent book of Goldreich [8]. Our further discussion is mainly based on Vadhan's excellent PhD Thesis [14] on statistical zero-knowledge proofs. As it would be impossible to cover all of the material in Vadhan's thesis in a short survey such as this, we shall focus on two subtopics. The first is the relationship of problems that admit statistical zero-knowledge proofs to the main complexity classes such as **NP**. The second topic provides a concrete example of a problem that admits a statistical zero-knowledge proof system. Namely, we discuss STATISTICAL DIFFERENCE, a problem that is complete for **HVSZK**, the class of (promise) problems that admit honest-verifier statistical zero-knowledge proofs.

The following sketch indicates the relevance of a complete problem for **HVSZK** to the study of general (cheating-verifier) statistical zero-knowledge proofs. (For further motivation and for more detailed formulation and proofs of the results informally stated below, the reader is encouraged to consult Vadhan's thesis.)

- ▷ Arbitrary private-coin **HVSZK** proof systems can be transformed to public-coin **HVSZK** proof systems while preserving communication-efficiency.

- ▷ Arbitrary public-coin **HVSZK** proof systems can be transformed to cheating-verifier statistical zero-knowledge proof systems while preserving communication-efficiency. In other words, **HVSZK** $\subseteq$ **SZK**, where **SZK** is the class of promise problems that admit cheating-verifier statistical zero-knowledge proofs.

- ▷ So, (since evidently **SZK** $\subseteq$ **HVSZK**) any problem complete for **HVSZK** is also complete for **SZK**.

2 Statistical difference

In order to talk about statistical indistinguishability and, subsequently, statistical zero knowledge, we must first make precise what is meant by statistical difference.

Definition 2.1 Let X and Y be random variables taking values in a discrete universe $\mathcal{U}$. The *statistical difference* of X and Y is defined as

$$\Delta(X,Y) \stackrel{\text{def}}{=} \max_{S\subseteq\mathcal{U}} |\Pr(X\in S) - \Pr(Y\in S)|.$$

We include the following discussion to provide the reader with some intuition over the statistical difference metric. In particular, we give two equivalent ways to define Δ. For $S\subseteq\mathcal{U}$, let us define

$$\begin{aligned}\delta(S) &\stackrel{\text{def}}{=} \Pr(X\in S) - \Pr(Y\in S)\\ &= \sum_{x\in S} \Pr(X=x) - \Pr(Y=x).\end{aligned}$$

Observe that $\delta(S)$ is increased (respectively, decreased) by adding elements $x \in U$ for which $\Pr(X = x) > \Pr(Y = x)$ (respectively, $\Pr(X = x) < \Pr(Y = x)$) to S. Moreover, $\delta(S)$ remains unchanged if elements x with $\Pr(X = x) = \Pr(Y = x)$ are either added or removed. So, it must be that the set

$$S_X \stackrel{\text{def}}{=} \{x\in\mathcal{U} \;:\; \Pr(X=x) > \Pr(Y=x)\}$$

maximizes $\delta(S)$. Conversely, the set

$$S_Y \stackrel{\text{def}}{=} \{x\in\mathcal{U} \;:\; \Pr(X=x) < \Pr(Y=x)\}$$

minimizes $\delta(S)$. Let us denote by $\bar{S}$ the complement of S with respect to $\mathcal{U}$. By the axioms of probability we have $\delta(\bar{S}) = -\delta(S)$, so $\delta(S_X) = -\delta(S_Y)$. This results in the following alternative definition for the statistical difference measure:

Proposition 2.2 *Let X and Y be random variables taking values in a discrete universe $\mathcal{U}$. Then,*

$$\begin{aligned}\Delta(X,Y) &= \Pr(X \in S_X) - \Pr(Y \in S_X) \\ &= \Pr(Y \in S_Y) - \Pr(X \in S_Y).\end{aligned}$$

The third alternative definition for Δ results from identifying random variables over $\mathcal{U}$ with weight-one sequences. Since we are working over a discrete universe, it is possible to enumerate the elements of $\mathcal{U}$. Fix an enumeration $u_1, u_2, \ldots \in \mathcal{U}$. A random variable X taking values in $\mathcal{U}$ is now conveniently described by a vector $\vec{X} \stackrel{\text{def}}{=} (x_1, x_2, \ldots) \in [0,1]^{\mathbb{N}}$, where $x_i \stackrel{\text{def}}{=} \Pr(X = u_i)$ for all $i \in \mathbb{N}$.

Proposition 2.3 *Let X and Y be random variables taking values in a discrete universe $\mathcal{U}$. Then,*

$$\Delta(X,Y) = \frac{1}{2}|\vec{X} - \vec{Y}|_1 \stackrel{\text{def}}{=} \frac{1}{2}\sum_{i=1}^{\infty} |x_i - y_i|$$

Proof. Fix S_X and S_Y as in prior discussion. Then, by partitioning the sum

$$|\vec{X} - \vec{Y}|_1 = \sum_{u \in \mathcal{U}} |\Pr(X = u) - \Pr(Y = u)|$$

into the three disjoint sets S_X, S_Y, $\overline{S_X \cup S_Y}$, and applying Proposition 2.2, we obtain $|\vec{X} - \vec{Y}|_1 = 2\Delta(X,Y)$ since $\Pr(X = u) = \Pr(Y = u)$ for all $u \in \overline{S_X \cup S_Y}$. ∎

We conclude this section by establishing several properties of Δ that justify the term "distinguishability metric" used earlier. Let us recall that the *support* of a random variable X is the set

$$\mathrm{Supp}(X) \stackrel{\text{def}}{=} \{u \in \mathcal{U} \ : \ \Pr(X = u) > 0\}.$$

Proposition 2.4 *Let X, Y, Z be three random variables taking values in a discrete universe $\mathcal{U}$. Then,*

1. *$\Delta(X,Y) \geq 0$ with equality if and only if X and Y are identical.*
2. *$\Delta(X,Y) \leq 1$ with equality if and only if X and Y have disjoint supports.*
3. *$\Delta(X,Y) = \Delta(Y,X)$.*
4. *$\Delta(X,Z) \leq \Delta(X,Y) + \Delta(Y,Z)$.*
5. *For any function $f : \mathcal{U} \to \mathcal{U}$, we have $\Delta(f(X), f(Y)) \leq \Delta(X,Y)$.*

Proof. Item 1 is an immediate consequence of Definition 2.1. Item 2 follows from the triangle inequality and Proposition 2.3. Item 3 is an immediate corollary of Proposition 2.3. Item 4 follows from Definition 2.1 combined with the triangle inequality. Item 5 follows from the fact that preimages of f that intersect the support of X, Y have cardinality at least one. ∎

3 Probability ensembles and indistinguishability

Recall the introduction where the zero-knowledge property of the prover of an interactive proof system was informally defined using various degrees of indistinguishability between the probability distribution induced by a verifier strategy interacting with the prover and an approximate distribution producible by the verifier on its own. In this section we shall make these indistinguishability notions precise.

We start by defining *probability ensembles*, which will be the main tool in discussing indistinguishability.

Definition 3.1 A *probability ensemble* over an index set $\mathcal{I}$ is a collection of $\{0,1\}^*$-valued random variables $\{X(i)\}_{i \in \mathcal{I}}$.

Most of the probability ensembles in this manuscript will be indexed by a combination of a string $w \in L \subseteq \{0,1\}^*$ and a *security parameter* $k \in \mathbb{N}$. The intuition is that k is a parameter that indicates the level of similarity (with respect to the distinguishability measures we shall define shortly) between the same-index random variables from two ensembles. The larger the parameter k, the harder it is to distinguish the two random variables by sampling.

Remark 1. The reason why it is useful to introduce a security parameter to the definitions is that uncoupling the security of a scheme from the instance length results in problem classes that are closed under reductions. (For example, the problem class **HVSZK**, discussed shortly, is closed under reductions if the security is uncoupled from the instance length; see [14, p. 29].) Another useful consequence is that low error-probability and low distinguishing-probability proofs can be given even for instances of short length.

Remark 2 To simplify the presentation we require that all probability ensembles in this manuscript have *polynomially bounded support*, that is,

$$\mathrm{Supp}(X(w,k)) \subseteq \bigcup_{j=0}^{\mathrm{poly}(|w|+k)} \{0,1\}^j$$

for all $w \in L$, $k \in \mathbb{N}$. (This assumption is evidently justified since any efficient interactive proof protocol has a transcript length bounded by a polynomial of the instace length $|w|$ and the security parameter k.)

Before proceeding to discuss the indistinguishability measures, we need to define what constitutes in an asymptotic sense an "acceptable risk" with respect to the security parameter k. This concept is formalized by the following definition of a negligible function.

Definition 3.2 A function $\mu : \mathbb{N} \to [0,1]$ is *negligible* if for every polynomial $q(k)$ there exists a $k_0 \in \mathbb{N}$ such that $\mu(k) < 1/q(k)$ for all $k \geq k_0$.

Note that a negligible function remains negligible even if multiplied by any polynomial.

3.1 Statistical indistinguishability

Definition 3.3 Let $L \subseteq \{0,1\}^*$. Two probability ensembles, $\{X(w,k)\}_{w\in L,k\in\mathbb{N}}$ and $\{Y(w,k)\}_{w\in L,k\in\mathbb{N}}$, are called *statistically indistinguishable* if there exists a negligible function μ such that

$$\Delta(X(w,k), Y(w,k)) \leq \mu(k)$$

holds for all $w \in L$, $k \in \mathbb{N}$. As a special case, we say that the two ensembles are *perfectly indistinguishable* if we can take $\mu(k) = 0$ for all $k \in \mathbb{N}$.

3.2 Computational indistinguishability

Let X be a $\{0,1\}^*$-valued random variable and suppose D is a (probabilistic) algorithm[3] with input in $\{0,1\}^*$. We write $\Pr(D(a,X) = 1)$ for the probability that D outputs 1 given $a \in \{0,1\}^*$ and $x \leftarrow X$ as input, where the probability is calculated over the outcomes of X and the random choices made by the algorithm at each step of a computation. The selection of x and the random choices made by the algorithm are assumed to be independent.

Definition 3.4 Let $L \subseteq \{0,1\}^*$. Two probability ensembles $\{X(w,k)\}_{w\in L,k\in\mathbb{N}}$ and $\{Y(w,k)\}_{w\in L,k\in\mathbb{N}}$ are *indistinguishable by probabilistic polynomial-time (PPT) algorithms* if for every probabilistic polynomial-time algorithm D there exists a negligible function μ such that

$$\big|\Pr\big(D(w,1^k,X(w,k)) = 1\big) - \Pr\big(D(w,1^k,Y(w,k)) = 1\big)\big| \leq \mu(k)$$

for all $w \in L$, $k \in \mathbb{N}$.

Remark 1. The intuition in the preceding definition is that D outputs 1 when it "thinks" that the sample given as input originated from $X(w,k)$. The definition states that deciding correctly the origin of a sample becomes increasingly hard as k is increased, regardless of the probabilistic polynomial-time algorithm D used.

Remark 2. Observe that the running time of D is bounded by $\mathrm{poly}(|w|+k)$ because of our requirement that all ensembles have polynomially bounded support.

3.3 Nonuniform computational indistinguishability

A stronger notion of computational indistinguishability is obtained by allowing the distinguishing algorithms to be *nonuniform.* Recall that an algorithm is nonuniform if it is supplied an "advice" string as auxiliary input based on the length of the original input. For the purposes of the following definition we assume (to simplify the presentation) that the advice string supplied to a computation is

[3] For the purposes of this manuscript we adopt a suitably standardized precise nondeterministic Turing machine as our model of randomized computation (cf. [12, Ch. 11]). More specifically, we assume that from each state and from each read symbol configuration there are exactly two alternative transitions to choose from. So, a bit string of appropriate length describes the choices made by the algorithm at each step of a computation. The output probabilities of the algorithm are then calculated as the fraction of bit strings that lead to a particular output.

a function of $|w| + k$ and has length bounded by $\mathrm{poly}(|w| + k)$.

Definition 3.5 Let $L \subseteq \{0,1\}^*$. Two probability ensembles $\{X(w,k)\}_{w\in L, k\in\mathbb{N}}$ and $\{Y(w,k)\}_{w\in L,k\in\mathbb{N}}$ are *indistinguishable by nonuniform polynomial-time (NUPT) algorithms* if for every nonuniform polynomial-time algorithm D there exists a negligible function μ such that

$$\big|\Pr\big(D(w,1^k,X(w,k))=1\big) - \Pr\big(D(w,1^k,Y(w,k))=1\big)\big| \le \mu(k)$$

for all $w \in L$, $k \in \mathbb{N}$.

Remark. Observe that we do not allow randomization in a nonuniform algorithm. This does not restrict the distinguishable probability ensembles, as we shall see later.

3.4 Relationships between notions of indistinguishability

This section is devoted to establishing relationships between the indistinguishability notions just defined. We shall proceed from the strongest to the weakest, that is, from statistical indistinguishability to indistinguishability by PPT algorithms.

Proposition 3.6 *If $X = \{X(w,k)\}_{w\in L,k\in\mathbb{N}}$ and $Y = \{Y(w,k)\}_{w\in L,k\in\mathbb{N}}$ are statistically indistinguishable, then they are indistinguishable by NUPT algorithms.*

Proof. We proceed in the contrapositive direction. Suppose that X and Y are distinguishable by a NUPT algorithm D. Then there exists a $w \in L$, a polynomial $q(k)$, and an infinite set $K \subseteq \mathbb{N}$ such that

$$\big|\Pr\big(D(w,1^k,X(w,k))=1\big) - \Pr\big(D(w,1^k,Y(w,k))=1\big)\big| > 1/q(k)$$

for all $k \in K$. Let

$$S_k \stackrel{\text{def}}{=} \{u \in \{0,1\}^* \;:\; D(w,1^k,u)=1\}$$

for all $k \in \mathbb{N}$ to obtain

$$\begin{aligned}
&\Delta(X(w,k),Y(w,k)) \ge \\
&\quad \ge |\Pr(X(w,k)\in S_k) - \Pr(Y(w,k)\in S_k)| \\
&\quad = |\Pr(D(w,1^k,X(w,k))=1) - \\
&\qquad \Pr(D(w,1^k,Y(w,k))=1)| \\
&\quad > 1/q(k)
\end{aligned}$$

for every $k \in K$. Thus, X and Y are not statistically indistinguishable. ■

Note that nowhere in the above proof we utilized the facts that the distinguisher algorithm was (a) nonuniform or (b) polynomial-time. So, the above proof goes through even for distinguishers which are arbitrary recursive functions. This demonstrates that statistical indistinguishability is a very strong requirement. In this sense the following proposition should not come as a surprise.

Proposition 3.7 *There exist ensembles that are indistinguishable by NUPT algorithms, but which have statistical difference close to one.*

Proof. The construction presented here is essentially the one in [8, Proposition 3.2.3]. We shall consider ensembles indexed only by the security parameter. Let $U(k)$ be the uniform distribution on $\{0,1\}^k$. Let $R(k)$ be a distribution on $\{0,1\}^k$ induced by selecting (possibly with repetition) $2^{k/2}$ strings $r_1,\dots,r_{2^{k/2}} \in \{0,1\}^k$, where each string assigns a probability mass of $2^{-k/2}$ to $R(k)$. By Proposition 2.3, we have

$$\Delta(R(k),U(k)) \ge 2^{k/2}(2^{-k/2} - 2^{-k}) = 1 - 2^{-k/2}.$$

So, the ensembles $U = \{U(k)\}_{k\in\mathbb{N}}$ and $R = \{R(k)\}_{k\in\mathbb{N}}$ are statistically very different regardless of how the strings inducing R are chosen.

The harder part of the proof is naturally establishing the existence of an R that is indistinguishable from U by NUPT algorithms. The existence proof is nonconstructive and relies on the probabilistic method. Let $\xi_i \stackrel{\text{def}}{=} U(k)$ for all $i = 1,\dots,2^{k/2}$. These independent random variables represent the random choices made in constructing $R(k)$. Our aim is to show that, for every k, there exists an outcome $\xi_1 = r_1,\dots,\xi_{2^{k/2}} = r_{2^{k/2}}$ such that the induced ensemble R is indistinguishable by NUPT algorithms from U.

Recall that the computation of every $\mathrm{poly}(k)$ time-bounded algorithm (with a $\mathrm{poly}(k)$-length advice string) on an input of size k can be encoded using a Boolean circuit of size $\mathrm{poly}(k)$ with k inputs (cf. [12, Theorem 8.2]). Fix a sufficiently compact encoding scheme for Boolean circuits (e.g. a size m circuit is encoded by a bit string of length at most $\mathrm{poly}(m)$). Then, for every fixed NUPT algorithm (and the associated advice string collection) there exists a $k_0 \in \mathbb{N}$ such that for every $k \geq k_0$ the Boolean circuits describing the operation of the algorithm on inputs of length k can be encoded using $2^{k/4}$ bits.

Fix any encoding $z \in \{0,1\}^{2^{k/4}}$ of a circuit C_z and let $p_z \stackrel{\text{def}}{=} \Pr(C_z(U(k)) = 1)$. Without loss of generality[4] we may assume that C_z has only one output bit. So, using the Chernoff bound (Theorem A.1), we obtain

$$\Pr\left(\left|p_z - \sum_{i=1}^{2^{k/2}} \frac{C_z(\xi_i)}{2^{k/2}}\right| \leq 2^{-k/8}\right) < 2^{-2^{k/4}}$$

for all $z \in \{0,1\}^{2^{k/4}}$ that encode a Boolean circuit. But then, since there are at most $2^{2^{k/4}}$ distinct circuits, we must have

$$\Pr\left(\exists z : \left|p_z - \sum_{i=1}^{2^{k/2}} \frac{C_z(\xi_i)}{2^{k/2}}\right| \leq 2^{-k/8}\right) < 1.$$

So, there exists at least one outcome

$$\xi_1 = r_1, \quad \xi_2 = r_2, \quad \cdots \quad \xi_{2^{k/2}} = r_{2^{k/2}}$$

such that

$$|\Pr(C_z(U(k)) = 1) - \Pr(C_z(R(k)))| < 2^{-k/8} \quad (1)$$

for all $z \in \{0,1\}^{2^{k/4}}$. Fix such an outcome for every k to obtain $R = \{R(k)\}_{k\in\mathbb{N}}$. Now, since $2^{-k/8}$ is negligible and the distinguishing behaviour of every NUPT algorithm D is incorporated in (1) from some value $k_0 \in \mathbb{N}$ onwards, we must have that U and R are indistinguishable by NUPT algorithms. ■

We now proceed to consider computational indistinguishability and the uniform and nonuniform distinguisher models. The following proposition demonstrates that (deterministic) nonuniform polynomial-time distinguishers are at least as effective as probabilistic (uniform) polynomial-time distinguishers.

Proposition 3.8 *If $X = \{X(w,k)\}_{w\in L,k\in\mathbb{N}}$ and $Y = \{Y(w,k)\}_{w\in L,k\in\mathbb{N}}$ are indistinguishable by NUPT algorithms, then they are indistinguishable by PPT algorithms.*

[4]Since the task is to distinguish between two distributions, it clearly suffices to have two output values to indicate the origin of a sample.

Proof. We proceed in the contrapositive direction. Suppose that X and Y are distinguishable by a PPT algorithm D. (Recall from Section 3.2 that we regard PPT algorithms as suitably standardized nondeterministic Turing machines.) Then there exists a $w \in L$, a polynomial $q(k)$, and an infinite set $K \subseteq \mathbb{N}$ such that

$$\begin{aligned}&|\Pr\big(D(w,1^k,X(w,k)) = 1\big) - \\ &\quad\Pr\big(D(w,1^k,Y(w,k)) = 1\big)| > 1/q(k)\end{aligned} \quad (2)$$

for all $k \in K$. Fix a $k \in K$. (In what follows we omit the fixed w, k to simplify the notation.) Let U denote the random variable that takes values uniformly at random in $\{0,1\}^{\text{poly}(|w|+k)}$, and denote by D_r the Turing machine obtained from D by fixing its nondeterministic random choices from r. Now, the left hand side of (2) can be written as

$$\begin{aligned}&\Big|\sum_r \big(\Pr(D_r(X) = 1,\ U = r) - \\ &\qquad \Pr(D_r(Y) = 1,\ U = r)\big)\Big| \leq \\ &\sum_r \Pr(U = r)\big|\Pr(D_r(X) = 1\big) - \Pr(D_r(Y) = 1)\big|,\end{aligned}$$

where the last inequality assumes that X, U and Y, U are independent. Consequently, there must exist an $r \in \{0,1\}^{\text{poly}(|w|+k)}$ with

$$|\Pr(D_r(X) = 1\big) - \Pr(D_r(Y) = 1)| > 1/q$$

So, for every $k \in K$ such an r can be supplied as the advice string to a NUPT algorithm that behaves exactly as D would, except that the random choices are always fixed by r. It is straightforward to verify that such an algorithm can distinguish between the ensembles X and Y with nonneglibigle probability. ■

Remark 1. Note that the independence assumption made in Section 3.2 (and used in the previous proof) is certainly justified since it would be unreasonable to assume that the random choices between the two alternatives made by the distinguisher algorithm would somehow be correlated with the sample to be distinguished.

Remark 2. The above proof can be modified in a straightforward fashion to show that two ensembles are indistinguishable by deterministic NUPT algorithms if and only if they are indistinguishable by probabilistic NUPT algorithms. So, adding a probabilistic element to a NUPT algorithm does not increase its distinguishing capability.

The following proposition does not lie entirely within the scope of the present manuscript (that is, statistical zero knowledge), but we choose to present it nevertheless because it illustrates an interesting difference between uniform and nonuniform models of computation.

Theorem 3.9 *There exist ensembles that are indistinguishable by PPT algorithms, but which can be distinguished using NUPT algorithms with probability close to one.*

Proof. This construction is from [6]. We again consider ensembles indexed only by the security parameter. Let $U(k)$ denote the uniform distribution on $\{0,1\}^k$. Fix any value $k \in \mathbb{N}$, and suppose $M_1, M_2, \ldots$ is an enumeration of probabilistic algorithms (s.o. nondeterministic Turing machines) that halt for all inputs. Denote by $p_j(x)$ the probability that M_j outputs 1 on $x \in \{0,1\}^k$, i.e., $p_j \stackrel{\text{def}}{=} \Pr(M_j(x) = 1)$. Fix an integer $t < 2^k$ and define $\vec{p}(x) \stackrel{\text{def}}{=} (p_1(x), \ldots, p_t(x)) \in \mathbb{R}^t$. Then $P \stackrel{\text{def}}{=} \{\vec{p}(x) \ : \ x \in \{0,1\}^k\}$ is a collection of 2^k vectors in $\mathbb{R}^t$. Now, for each M_j we have

$$\Pr(M_j(U(k)) = 1) = \sum_{x \in \{0,1\}^k} 2^{-k} \Pr(M_j(x) = 1)$$

assuming that the coin tosses of M_j and the selection of x are independent events. But then the t-dimensional vector

$$\vec{p} = \sum_{x \in \{0,1\}^k} 2^{-k} \vec{p}(x)$$

encodes in its jth component the probability $\Pr(M_j(U(k)) = 1)$. Observe that $\vec{p}$ belongs to the convex hull spanned by the points in P. Since the hull is at most t-dimensional, there exist $t+1$ points $\vec{p}(x_1), \ldots, \vec{p}(x_{t+1}) \in P$ such that $\vec{p}$ is contained in their convex hull. In other words, there exist $\lambda_1, \ldots, \lambda_{t+1} \in [0,1]$ with $\sum_{i=1}^{t+1} \lambda_i = 1$ such that $\vec{p} = \sum_{i=1}^{t+1} \lambda_i \vec{p}(x_i)$.

Define a random variable $T(k)$ over $\{0,1\}^*$ by setting $\Pr(T(k) = x_i) \stackrel{\text{def}}{=} \lambda_i$ for all $i = 1, \ldots, t+1$. It is now straightforward to verify that $U(k)$ and $T(k)$ are indistinguishable by $M_1, \ldots, M_t$ because

$$\Pr(M_j(T(k)) = 1) = \Pr(M_j(U(k)) = 1).$$

So, by selecting e.g. $t = k$ we obtain two probability ensembles $T = \{T(k)\}_{k \in \mathbb{N}}$ and $U = \{U(k)\}_{k \in \mathbb{N}}$ indistinguishable by PPT algorithms (in fact, by *any* uniform algorithm). Now since $t = k$, the support of $T(k)$ has size $t + 1$. Thus, the support can be described by a bit string of length $k(t+1) \leq \text{poly}(k)$. This description can be used as the advice string to a nonuniform algorithm D that outputs 1 if and only if its input $x \in \text{Supp}(T(k))$. (Such an algorithm is obviously implementable to run in time $\text{poly}(k)$.) The distinguishing gap for D is therefore

$$\begin{aligned} \left|\Pr(D(T(k)) = 1) - \Pr(D(U(k)) = 1)\right| = \\ = 1 - (k+1)2^{-k}, \end{aligned}$$

so D can be used to distinguish the ensembles T, U with overwhelming probability. ∎

4 Zero-knowledge proof systems

We now turn our attention to zero-knowledge interactive proof systems for deciding promise problems.

4.1 Basic definitions

Definition 4.1 A *promise problem* is a pair $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$, where $\Pi_{\text{yes}} \subseteq \{0,1\}^*$ and $\Pi_{\text{no}} \subseteq \{0,1\}^*$ are disjoint sets.

A promise problem differs from standard language membership recognition by the fact that there are instances (namely, the instances in $\{0,1\}^* \setminus (\Pi_{\text{yes}} \cup \Pi_{\text{no}}))$ for which the decision made by the algorithm (or protocol) is allowed to be arbitrary. Standard language membership problems are special cases of promise problems with $\Pi_{\text{yes}} \cup \Pi_{\text{no}} = \{0,1\}^*$.

We assume a degree of familiarity with interactive and zero-knowledge proof systems, so we shall not present all the definitions in the main text and defer them to the appendix.

Briefly, the soundness and completeness errors in an interactive protocol are tied to the security parameter k, as opposed to the usual input instance length. The verifier's view of an interaction on common input w is represented by the random variable $\text{view}_V(w)$, where the individual outcome probabilities are determined by the random strings supplied to the prover (and to the verifier). Our simulators are strictly polynomial-time and are allowed to fail with probability $\leq 1/2$. The output distibution $\tilde{S}$ of a simulator S is conditioned on nonfailure, as usual.

We shall consider two notions of zero knowledge for interactive proof systems. In *honest-verifier* zero knowledge the verifier plays by the book (as specified in the underlying interactive protocol) and gains no knowledge from interacting with the prover. In *cheating-verifier* zero knowledge the verifier is allowed to deviate arbitrarily from the original interactive protocol.

Definition 4.2 An interactive proof system (P, V) for a promise problem Π is said to be *honest-verifier perfect/statistical/computational zero knowledge* if there exists a polynomial-time simulator S such that the probability ensembles

$$\{\mathrm{view}_V(x, 1^k)\}_{x\in\Pi_{\mathrm{yes}},k\in\mathbb{N}}, \quad \{\tilde{S}(x, 1^k)\}_{x\in\Pi_{\mathrm{yes}},k\in\mathbb{N}}$$

are perfectly/statistically/computationally indistinguishable.

Remark. We identify "computational indistinguishability" with indistinguishability by NUPT algorithms.

Definition 4.3 We denote the classes of promise problems that admit honest-verifier perfect/statistical/computational zero knowledge proof systems by **HVPZK/HVSZK/HVCZK**, respectively.

In the following definition we denote by V'_z and S_z the algorithms V' and S supplied with an additional auxiliary input string $z \in \{0,1\}^*$. The running time bound on both V' and S is assumed to remain a function of the nonauxiliary input.

Definition 4.4 An interactive proof system (P, V) for a promise problem Π is said to be *(cheating-verifier) perfect/statistical/computational zero knowledge* if for every probabilistic polynomial time V' there exists a polynomial-time simulator S such that the probability ensembles

$$\{\mathrm{view}_{V'_z}(x, 1^k)\}_{x\in\Pi_{\mathrm{yes}},k\in\mathbb{N},z\in\{0,1\}^*},$$
$$\{\tilde{S}_z(x, 1^k)\}_{x\in\Pi_{\mathrm{yes}},k\in\mathbb{N},z\in\{0,1\}^*}$$

are perfectly/statistically/computationally indistinguishable.

Remark 1. Observe that the definitions for computational indistinguishability provide the index string $w \in L$ as input to a distinguisher algorithm. For the purposes of the preceding definition we assume that both strings x, z are provided to a distinguisher as input.

Remark 2. Cheating-verifier zero-knowledge as just defined is often called *auxiliary-input* zero-knowledge. For motivation why the additional security provided by the auxiliary input is desirable, consult e.g. [5, 8].

Definition 4.5 We denote the classes of promise problems that admit cheating-verifier perfect/statistical/computational zero knowledge proof systems by **PZK/SZK/CZK**, respectively.

4.2 Relationships between SZK and other complexity classes

In this section we review (some of the many) known results relating **SZK** to other complexity classes. Our aim is to simply relate **SZK** to **CZK** and **NP**. (Vadhan's thesis [14] is an excellent reference for results beyond those mentioned here.)

From the analysis conducted in Section 3.4 we obtain immediately the inclusions

$$\mathbf{BPP} \subseteq \mathbf{HVPZK} \subseteq \mathbf{HVSZK} \subseteq \mathbf{HVCZK} \subseteq \mathbf{IP}$$

and

$$\mathbf{BPP} \subseteq \mathbf{PZK} \subseteq \mathbf{SZK} \subseteq \mathbf{CZK} \subseteq \mathbf{IP},$$

where the containment between **BPP** and **IP** is obvious by the definitions of these classes. Also, it is clear by definition that **SZK** $\subseteq$ **HVSZK**. Perhaps somewhat surprisingly, the converse inclusion also holds. That is, honest-verifier statistical zero-knowledge proof systems can always be transformed to ones secure against cheating verifiers.

Theorem 4.6 **HVSZK** = **SZK** *[7].*

(A self-contained proof of this result appears in [14].)

The strong requirement of statistical indistinguishability comes with a complexity-theoretic cost. Although many problems believed to be hard (such as QUADRATIC RESIDOSITY [9] and

Graph Isomorphism [10]) admit statistical zero-knowledge proofs, it is generally believed that **NP** is not contained in **SZK**. This is because $\mathbf{NP} \subseteq \mathbf{SZK}$ implies that the polynomial hierarchy collapses. This can be seen from $\mathbf{HVSZK} \subseteq \mathbf{AM} \cap \mathbf{coAM}$ [1, 4] and the fact that if $\mathbf{NP} \subseteq \mathbf{coAM}$, then the polynomial hierarchy collapses [3].

For **CZK** it is known that the existence of nonuniformly one-way functions implies $\mathbf{CZK} = \mathbf{IP}$ [2, 8, 11]. Since $\mathbf{IP} = \mathbf{PSPACE}$ [13], it is thus very likely that the inclusion $\mathbf{SZK} \subseteq \mathbf{CZK}$ is proper.

It is an open problem (with no significant evidence pointing either way) whether strict inclusion holds between **PZK** and **SZK** [14].

5 A problem complete for HVSZK.

To provide the reader a concrete example of a problem that admits a statistical zero-knowledge proof system, we consider the promise problem Statistical Difference that involves deciding the magnitude of the statistical difference between two distributions. This section is based on [14, Ch. 3].

Definition 5.1 Let X be a Boolean circuit with n inputs and m outputs. The *random variable induced by X on $\{0,1\}^m$* is defined by

$$\Pr(X = a) \stackrel{\text{def}}{=} 2^{-n}|\{y \in \{0,1\}^n \ : \ a = X(y)\}|$$

for all $a \in \{0,1\}^m$, where $X(y)$ denotes the output of X produced by input y.

(Note that we abuse notation and denote by X both the circuit and the random variable it defines. The actual interpretation will be clear from the context.)

Definition 5.2 For $\alpha, \beta \in [0,1]$, define the promise problem $\mathrm{SD}^{\alpha,\beta} = (\mathrm{SD}^{\alpha,\beta}_{\text{yes}}, \mathrm{SD}^{\alpha,\beta}_{\text{no}})$ by

$$\mathrm{SD}^{\alpha,\beta}_{\text{yes}} \stackrel{\text{def}}{=} \{(X,Y) \in \{0,1\}^* \ : \ \Delta(X,Y) \geq \alpha\},$$
$$\mathrm{SD}^{\alpha,\beta}_{\text{no}} \stackrel{\text{def}}{=} \{(X,Y) \in \{0,1\}^* \ : \ \Delta(X,Y) \leq \beta\},$$

where X and Y encode Boolean circuits and the statistical difference is calculated for the induced random variables X, Y.

Definition 5.3 Statistical Difference is the promise problem $\mathrm{SD}^{2/3,1/3}$.

Statistical Difference is a central problem for statistical zero-knowledge since it is *complete* for the class **HVSZK** (and hence for **SZK** since $\mathbf{HVSZK} = \mathbf{SZK}$). Recall that completeness for a class of problems means that any problem in the class is reducible to the complete problem, which also belongs to the class. This combined with the fact that **HVSZK** is closed under reductions[5] allows us obtain a convenient characterization of the class of problems that admit statistical zero-knowledge proofs: A problem has a (HV)SZK proof system if and only if it is reducible to Statistical Difference.

In what follows we will sketch a HVSZK proof system for Statistical Difference. The completeness proof for **HVSZK** is unfortunately too long to be covered in this survey. The reader is referred to [14] for details of the completeness proof.

5.1 A HVPZK proof system for $\mathrm{SD}^{1,0}$

Before turning to Statistical Difference, let us first consider the easier problem $\mathrm{SD}^{1,0}$. By Proposition 2.4 the task is then to decide whether two random variables encoded by circuits X_0, X_1 are identical (a "no"-instance) or have disjoint support (a "yes"-instance).

A simple random sampling strategy yields a HVPZK proof system for this problem. Suppose the circuits X_0, X_1 in the problem instance have n inputs and m outputs. The protocol is as follows:

V: The verifier selects randomly a $b \in \{0,1\}$ and a sample $x \leftarrow X_b$. (In other words, the verifier selects randomly a $r \in \{0,1\}^n$ and computes $x = X_b(r)$.) The verifier sends x to the prover.

P: The prover computes whether $\Pr(X_0 = x) > \Pr(X_1 = x)$. If so, then $c = 0$; otherwise $c = 1$. The prover sends c to the verifier.

V: If $b = c$, the verifier accepts. Otherwise the verifier rejects.

It is easy to see that the protocol has perfect completeness. Moreover, the soundness error on a "no"-instance is exactly $1/2$.

[5] That is, any problem reducible to a problem in **HVSZK** is itself also in **HVSZK**

The simulator for the honest verifier is straightforward. Given a "yes"-instance X_0, X_1 as input, the simulator first selects randomly a $b \in \{0,1\}$ and samples $x \leftarrow X_b$ using a random string $r \in \{0,1\}^n$. Then it outputs (x, b, yes) as the messages exchanged in the simulated protocol run and the concatenation of b and r as the verifier's random string. Since a "yes"-instance is considered, the prover reply c in the actual protocol is always equal to b. So, the simulator's output distribution is equal to the distribution induced by the honest verifier interacting with the prover.

5.2 Extension to $\mathrm{SD}^{2/3,1/3}$

Without attaching any particular promise problem yet to the context, let us consider what happens in the protocol when $0 < \Delta(X_0, X_1) = \delta < 1$. Namely, what is the verifier's acceptance probability against an arbitrary prover strategy?

We shall demonstrate that the prover strategy is the best possible by deriving an alternative way to view the verifier's random selection process for x, b. For convenience, we identify the strings in $\{0,1\}^m$ with $0, 1, \ldots, 2^m - 1$. Also, put $p_i \stackrel{\text{def}}{=} \Pr(X_0 = i)$ and $q_i \stackrel{\text{def}}{=} \Pr(X_1 = i)$. Now define

$$\begin{aligned} y_i &\stackrel{\text{def}}{=} \max(p_i - q_i, 0), \\ z_i &\stackrel{\text{def}}{=} \max(q_i - p_i, 0), \\ w_i &\stackrel{\text{def}}{=} \min(p_i, q_i). \end{aligned}$$

Then,

$$p_i = y_i + w_i, \qquad q_i = z_i + w_i.$$

The intuition is that if we draw a graph of the probability masses of X_0 and X_1 with respect to the outcomes in $\{0,1\}^m$, then w_i measures the "common region," where probability masses associated to i overlap, and y_i and z_i measure the "excess" probability mass associated to i by X_0 and X_1, respectively. (Naturally, at most one of y_i, z_i is nonzero for any given i.) So, by Proposition 2.2 we must have

$$\sum_i y_i = \sum_i z_i = \delta,$$

which leaves $\sum_i w_i = 1 - \delta$, since $\sum_i p_i = \sum_i q_i = 1$.

Let us introduce random variables D, D_0^+, and D_1^+ on $\{0,1\}^m$ by setting

$$\begin{aligned} \Pr(D = i) &= w_i/(1-\delta), \\ \Pr(D_0^+ = i) &= y_i/\delta, \\ \Pr(D_1^+ = i) &= z_i/\delta \end{aligned}$$

for all $i = 0, 1, \ldots, 2^m - 1$. Note that this simply means normalizing w_i, y_i, z_i. Now the verifier's first step in which b, x are selected can be (perfectly) simulated as follows:

1. Flip a biased coin d with $\Pr(d = 0) = 1 - \delta$ and $\Pr(d = 1) = \delta$.
2. If $d = 0$, then let $x \leftarrow D$ and select $b \in \{0,1\}$ uniformly at random.
3. If $d = 1$, then select $b \in \{0,1\}$ uniformly at random.
   (a) If $b = 0$, then let $x \leftarrow D_0^+$.
   (b) If $b = 1$, then let $x \leftarrow D_1^+$.
4. Proceed with b, x as in the original protocol.

Observe that this is simply an alternative way to view the original protocol, it is *not* required that the verifier should be able to behave as above.

From the alternative view of the verifier's strategy it can be seen that the prover strategy always succeeds guessing correctly if $d = 1$. On the other hand, if $d = 0$, then no prover can do better than guess b with probability $1/2$ since b, x are independent. So, the probability that *any* prover strategy causes the verifier to accept on an instance with $\Delta(X_0, X_1) = \delta$ is at most

$$1/2 \cdot \Pr(d = 0) + 1 \cdot \Pr(d = 1) = \frac{1+\delta}{2},$$

with equality if the prover strategy in the protocol is applied. We have thus settled the acceptance probability of the protocol for arbitrary distributions.

What about the statistical difference between the simulator's output distribution and the distribution induced by the verifier interacting with the prover?

Clearly, if the prover succeeds, then the transcript of the protocol is always equal to one produced by the simulator. On the other hand, if the prover fails (which occurs with probability

$1-(1+\delta)/2 = (1-\delta)/2)$, then the transcript is always distinct from the ones produced by the simulator. So, we have

$$\Delta(\text{view}_V(X_0, X_1, 1^k), \tilde{S}(X_0, X_1, 1^k)) = \frac{1-\delta}{2}$$

for all $k \in \mathbb{N}$.

Since the statistical difference is constant with respect to k instead of a negligible function, the protocol is not statistical zero-knowledge. However, this is no major obstacle, since there exist tools for *polarizing* the statistical difference of two distributions.

Lemma 5.4 (Polarization Lemma) *Let $\alpha, \beta \in [0,1]$ be any two constants such that $\alpha^2 > \beta$. There is a polynomial-time computable function* Polarize$_{\alpha,\beta}$ *that takes a triple $(X_0, X_1, 1^k)$, where X_0 and X_1 are distributions encoded by circuits, and outputs a pair of circuits (Y_0, Y_1) such that*

$$\Delta(X_0, X_1) \geq \alpha \Rightarrow \Delta(Y_0, Y_1) \geq 1 - 2^{-k},$$
$$\Delta(X_0, X_1) \leq \beta \Rightarrow \Delta(Y_0, Y_1) \leq 2^{-k}.$$

Observe that this lemma is precisely the tool we need to transform the HVPZK proof system for $\text{SD}^{1,0}$ into a HVSZK proof system for Statistical Difference. Simply apply Polarize$_{2/3,1/3}$ to the input distributions (X_0, X_1) and run the original protocol on the distributions (Y_0, Y_1) output by the polarization procedure.

Unfortunately we do not have the space (or more accurately, the time) to prove the Polarization Lemma in detail. However, we indicate the two constructions that are alternated to produce the desired distributions from the input distributions. We denote by $Z \otimes W$ the distribution (circuit) that is obtained by independently sampling the distributions Z and W. (In other words, $Z \otimes W$ denotes the circuit where Z and W are glued side by side.) The following lemma shows that this gluing construction increases the statistical difference between distributions, but only by an amount bounded from above as well.

Lemma 5.5 (Direct Product Lemma) *Let X, Y be distributions such that $\Delta(X,Y) = \delta$. Then, for all $k \in \mathbb{N}$,*

$$1 - 2e^{-k\delta^2/2} \leq \Delta(\otimes^k X, \otimes^k Y) \leq k\delta.$$

($\otimes^k X$ denotes the circuit obtained by gluing k copies of X side by side.) The other construction presents a circuit that decreases statistical difference between distributions.

Lemma 5.6 (XOR Lemma) *There is a polynomial-time computable function that maps a triple $(X_0, X_1, 1^k)$ to a pair of circuits (Y_0, Y_1) such that*

$$\Delta(Y_0, Y_1) = \Delta(X_0, X_1)^k.$$

Specifically, Y_0 and Y_1 are defined as follows.

Y_0: *Uniformly select $(b_1, \ldots, b_k) \in \{0,1\}^k$ such that $b_1 \oplus \cdots \oplus b_k = 0$ and output a sample of $X_{b_1} \otimes \cdots \otimes X_{b_k}$.*

Y_1: *Uniformly select $(b_1, \ldots, b_k) \in \{0,1\}^k$ such that $b_1 \oplus \cdots \oplus b_k = 1$ and output a sample of $X_{b_1} \otimes \cdots \otimes X_{b_k}$.*

The circuits Y_0 and Y_1 are clearly implementable using a suitable multiplexer/demultiplexer construction in combination with a "parity-enforcer" that takes as input $(b_1, \ldots, b_k)$, flips b_k if the input has incorrect parity, and outputs the resulting string $(b_1, \ldots, b_k)$.

The Polarization Lemma is now proved by applying the XOR Lemma from both sides (that is, before and after the Direct Product Lemma is applied) to control the construction in the Direct Product Lemma so that a statistical difference of $\leq \beta$ is polarized to $\leq 2^{-k}$ and a statistical difference of $\geq \alpha$ is polarized to $\geq 1 - 2^{-k}$.

References

[1] W. Aiello and J. Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *J. Comput. System. Sci.* **42** (1991) no. 3, 327–345.

[2] M. Ben-Or, O. Goldreich, S. Goldwasser, *et al.* Everything provable is provable in zero-knowledge. Proc. CRYPTO '88, LNCS vol. 403, p. 40–51, Springer-Verlag, 1988.

[3] R. B. Boppana, J. Håstad and S. Zachos. Does coNP have short interactive proofs? *Inform. Process. Lett.* **25** (1987), 127–132.

[4] L. Fortnow. The complexity of perfect zero-knowledge. In *Advances in Computing Research* (S. Micali, ed.) volume 5, p. 327–343. JAC Press, 1989.

[5] O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology* **7** (1994) no. 1, 1–32.

[6] O. Goldreich and B. Meyer. Computational indistinguishability: algorithms vs. circuits. *Theoret. Comput. Sci.* **191** (1998) no. 1–2, 215–218.

[7] O. Goldreich, A. Sahai and S. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. Proc. 30th ACM STOC, p. 399-408, ACM, New York, 1998.

[8] O. Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, Cambridge, England, 2001.

[9] S. Goldwasser, S. Micali and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.* **18** (1989) no. 1, 186–208.

[10] O. Goldreich, S. Micali and A. Widgerson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM* **38** (1991) no. 1, 691–729.

[11] R. Impagliazzo and M. Yung. Direct minimum knowledge computations. Proc. CRYPTO '87, LNCS vol. 293, p. 40–51, Springer-Verlag, 1987.

[12] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, Reading, Massachusetts, 1994.

[13] A. Shamir. IP = PSPACE. *J. ACM* **39** (1992) no. 4, 869–877.

[14] S. P. Vadhan. A Study of Statistical Zero-Knowledge Proofs. PhD Thesis, Department of Mathematics, Massachusetts Institute of Technology, 1999.

A Appendix

Theorem A.1 (Chernoff Bound) *Let $p \leq 1/2$ and suppose $X_1, X_2, \ldots, X_n$ are independent random variables taking values in $\{0,1\}$ with $\Pr(X_i = 1) = p$ for every $i = 1, \ldots, n$. Then,*

$$\Pr\left(\left|\frac{\sum_{i=1}^n X_i}{n} - p\right| > \epsilon\right) < 2e^{-\frac{\epsilon^2 n}{2p(1-p)}}$$

for all $0 < \epsilon \leq p(1-p)$.

A.1 Interactive proof systems

Definition A.2 An *interactive protocol* is a pair of functions (P, V) that for every $x \in \{0,1\}^*$ and $r_P, r_V \in \{0,1\}^{\mathbb{N}}$ produce an *interaction*

$$\langle P, V\rangle(x; r_P, r_V) \stackrel{\text{def}}{=} (m_1, \ldots, m_{2l}) \in \{0,1\}^*,$$

where

1. either $m_{2l} = \text{yes}$ (V *accepts* x) or $m_{2l} = \text{no}$ (V *rejects* x); and
2. for all $i = 1, \ldots, l$,
   (a) $m_{2i-1} = P(x, m_1, \ldots, m_{2i-2}; r_P)$; and
   (b) $m_{2i} = V(x, m_1, \ldots, m_{2i-1}; r_V)$.

An interactive protocol *polynomially bounded* if there exists a polynomial p such that, for all $x \in \{0,1\}^*$, both $l \leq p(|x|)$ and $|m_i| \leq p(|x|)$ for all $i = 1, \ldots, l$.

Definition A.3 Let Π be a promise problem. An interactive protocol (P, V) is said to be an *interactive proof system* for Π with *completeness error* $c : \mathbb{N} \to [0,1]$ and *soundness error* $s : \mathbb{N} \to [0,1]$ if the following conditions hold:

1. (P, V) is polynomially bounded and V is polynomial-time computable.[6]
2. If $x \in \Pi_{\text{yes}}$, then

   $$\Pr(\text{V accepts } (x, 1^k)) \geq 1 - c(k),$$

   where r_P, r_V are selected uniformly at random.

[6] The polynomial time bound of V is assumed to hold w.r.t. input length excluding the string $r_V \in \{0,1\}^{\mathbb{N}}$, which is assumed to be provided separately (e.g. as the content of a dedicated Turing machine tape) from the other input.

3. If $x \in \Pi_{\text{no}}$, then, for all functions P',

$$\Pr(\text{V rejects } (x, 1^k)) \geq 1 - s(k),$$

where r_P, r_V are selected uniformly at random and V is assumed to interact with P'.

Remark. Note that no particular behaviour is required of the interactive proof system if $x \notin \Pi_{\text{yes}} \cup \Pi_{\text{no}}$. In other words, if the "promise" $x \in \Pi_{\text{yes}} \cup \Pi_{\text{no}}$ is violated, then any behaviour is acceptable. Furthermore, note that the completeness and soundness errors are specified w.r.t. a separate error parameter $k \in \mathbb{N}$.

Definition A.4 We denote by **IP** the class of promise problems that admit an interactive proof system with completeness error $c(k)$ and soundness error $s(k)$, where both $c(k)$ and $s(k)$ are assumed to be computable in time $\text{poly}(k)$ and satisfy $c(k) + s(k) < 1 - 1/\text{poly}(k)$ for all $k \in \mathbb{N}$.

If we omit the completeness and soundness errors then they are assumed to satisfy the requirements presented in the previous definition.

A.2 Zero knowledge

Definition A.5 Let (P, V) be an interactive proof system for a promise problem Π and suppose $x \in \{0,1\}^*$. A string $(m, r'_V) \in \{0,1\}^*$ is V's *view* of an interaction between P and V on common input x if there exist $r_P, r_V \in \{0,1\}^{\mathbb{N}}$ such that $m = \langle P, V\rangle(x; r_P, r_V)$ and r'_V is the $\text{poly}(|x|)$-length prefix of r_V accessible to V.

The $\{0,1\}^*$-valued random variable $\text{view}_V(x)$ associates to each view of V on input x its occurence probability when $r_P, r_V \in \{0,1\}^{\mathbb{N}}$ are selected uniformly at random.

Definition A.6 A probabilistic polynomial-time algorithm S with output in $\{0,1\}^* \cup \{\perp\}$ is a *polynomial-time simulator* if $\Pr(S(x) = \perp) \leq 1/2$ for all $x \in \{0,1\}^*$. The *output distribution* of S on input x is the random variable $\tilde{S}(x)$ defined by

$$\Pr\big(\tilde{S}(x) = y\big) \stackrel{\text{def}}{=} \Pr\big(S(x) = y \mid S(x) \neq \perp\big)$$

for all $y \in \{0,1\}^*$.

Remark. Strictly speaking, a simulator is only *expected* polynomial-time.

A.3 Reducibility and completeness

The following definitions are standard, but we present them nevertheless for the reader's convenience.

Definition A.7 A promise problem Π is *(Karp) reducible* to a promise problem Π' if there exists a polynomial-time deterministic algorithm R that satisfies

(i) $R(x) \in \Pi'_{\text{yes}}$ for all $x \in \Pi_{\text{yes}}$; and

(ii) $R(x) \in \Pi'_{\text{no}}$ for all $x \in \Pi_{\text{no}}$.

Such an algorithm R is called a *(Karp) reduction* from Π to Π'.

Definition A.8 Let $\mathcal{C}$ be a class of promise problems. A promise problem Π is *complete* for the class $\mathcal{C}$ (briefly, Π is *$\mathcal{C}$-complete*) if

(i) $\Pi \in \mathcal{C}$; and

(ii) All promise problems in $\mathcal{C}$ are reducible to Π.