SZK-Completeness, Probabilistically Checkable Proofs

This lecture, we will complete the proof of SZK-completeness of the Statistical Closeness problem, and start on the topic of Probabilistically Checkable Proofs. We recall the following facts about the statistical distance between joint distributions.

Fact 0.1. *For any distributions (X_0, Y_0) and (X_1, Y_1),*

$$\Delta((X_0, Y_0), (X_1, Y_1)) \geq \max\left[\Delta(X_0, X_1), \Delta(Y_0, Y_1)\right]$$

Fact 0.2. *For any distributions (X_0, Y_0) and (X_1, Y_1),*

$$\Delta((X_0, Y_0), (X_1, Y_1)) \leq \Delta(X_0, X_1) + \mathrm{E}_{x \leftarrow X_0}\left[\Delta(Y_0|_{X_0=x}, Y_1|_{X_1=x})\right]$$

In particular, if X_0 is independent of Y_0 and X_1 is independent of Y_1,

$$\Delta((X_0, Y_0), (X_1, Y_1)) \leq \Delta(X_0, X_1) + \Delta(Y_0, Y_1)$$

1 Completeness of $\mathrm{SC}^{\beta,\alpha}$

It turns out that the problem of determining the statistical distance between distributions really captures the essence of statistical zero-knowledge proofs. $\mathrm{SC}^{\beta,\alpha}$ is actually complete for the class SZK for any constants (β, α) such that $\alpha^2 > \beta$. That is, it is contained in SZK and any problem in SZK can be reduced to it. That it is contained in SZK follows from an interesting reduction from $\mathrm{SC}^{\beta,\alpha}$ to $\mathrm{SD}^{\alpha',\beta'}$ that we will not cover (see Vadhan's thesis [Vad99] for the reduction).

We will see now how any problem that has an HVSZK proof can be reduced to $\mathrm{SC}^{\beta,\alpha}$ (this is stronger than having SZK in this statement). We will see only a very special case of the reduction that still has the main ideas. Suppose a language L has an SZK proof (P, V) with the following properties:

- It has two messages and is public-coin. That is, V first sends a uniformly random $\alpha \leftarrow \{0,1\}^a$, P responds with a $\beta \in \{0,1\}^b$, and V then outputs $out = accept/reject$.
- The completeness, soundness, and zero-knowledge errors are all negligible.

I am not aware of any non-trivial SZK protocol that has these properties, but this will do for the sake of illustration. The actual reduction works by first transforming any SZK protocol to a public-coin one using a zero-knowledge version of the set lower-bound protocol we saw in an earlier lecture, and then applying an extension of the reduction we will see now.

Our objective is to map any instance x to two distributions (D_0, D_1) (the circuits sampling them will be obvious from our descriptions) such that if $x \in L$, $\Delta(D_0, D_1) \leq \beta$, and if $x \notin L$, $\Delta(D_0, D_1) \geq \alpha$. The honest verifier's view consists of (α, β, out) – denote the random variables corresponding to this by (A, B, O). We will make use of the simulator S that takes input x and security parameter 1^λ, and outputs $(\alpha_S, \beta_S, out_S)$, whose distance from the actual view is $\mathrm{negl}(\lambda)$; define (A_S, B_S, O_S) as random variables corresponding to these.

Looking at the simulator's guarantee, we see that we already have some of what we are asking for – when $x \in L$, the distribution of the S's output and V's view are very close. In particular, A_S is close to the uniform distribution over $\{0,1\}^a$ (which we will denote by U_a). Further, due to the negligible completeness and zero-knowledge errors, the output of the verifier on the simulated transcript $V(x, A_S, B_S)$ is *accept* with all but negligible probability.

We would be done if, whenever $x \notin L$, A_S is far from U_a. This need not be the case, however, as no explicit restrictions are placed on the simulator's output when $x \notin L$. Soundness, though, implies that in this case, for most $\alpha \in \{0,1\}^a$, there does not exist a β such that $V(x, \alpha, \beta) = accept$. So, if A_S is very close to U_a, then except with negligible probability, the β_S will be such that $V(x, \alpha_S, \beta_S) = reject$.

To reiterate, when $x \in L$, A_S is negligibly close to U_a and $V(x, A_S, B_S) = accept$ with high probability. Whereas, when $x \notin L$, either A_S is far from U_a, or $V(x, A_S, B_S) = reject$ with high probability. These

two properties together – whether A_S is close to U_a and whether $V(x, A_S, B_S)$ accepts most of the time – distinguish the behaviour of the simulator in the cases $x \in L$ an $x \notin L$. We will use this to construct the two distributions that the reduction will map x to as follows:

- D_0:
  1. Sample $\alpha \leftarrow \{0,1\}^a$
  2. Output $(1, \alpha)$
- D_1:
  1. For some large λ, compute $(\alpha_S, \beta_S) \leftarrow S(x, 1^\lambda)$
  2. Run $S(x, 1^\lambda)$ a large $\mathsf{poly}(\lambda)$ times. Set $b = 1$ if all of these (α'_S, β'_S) are such that $V(x, \alpha'_S, \beta'_S) = \textit{accept}$. Set $b = 0$ otherwise.
  3. Output (b, α_S)

When $x \in L$, the distance between D_0 and D_1 is bounded as:

$$\Delta((1, A), (b, A_S)) \leq \Delta(1, b) + \Delta(U_a, A_S) \leq \mathrm{negl}(\lambda)$$

where the first inequality uses Fact 0.2. The second follows because when $x \in L$, all the $V(x, \alpha'_S, \beta'_S)$ accept except with negligible probability, and A_S is negligibly close to U_a. When $x \notin L$, the distance is bounded as:

$$\Delta((1, A), (b, A_S)) \geq \max\left[\Delta(1, b), \Delta(U_a, A_S)\right] \geq 0.1$$

where the first inequality uses Fact 0.1. The second follows because, as argued earlier, either A_S is far from U_a, or at least one of the $V(x, \alpha'_S, \beta'_S)$ rejects except with negligibly probability. From here, the Polarisation Lemma can be used to reduce to $\mathrm{SC}^{\beta,\alpha}$ for any constants $\alpha > \beta$ as desired.

2 Properties of SZK

The class SZK of problems that have statistical zero-knowledge proofs has a number of interesting properties that were discovered throughout the 90's. The discovery of complete problems for SZK (in particular $\mathrm{SC}^{\beta,\alpha}$ and the Entropy Difference problem, which you might encounter in the next problem set) greatly simplified the proofs of many of these properties. We will state a few of them below without proof. Refer to Vadhan's thesis [Vad99] for proofs of all of these statements.

1. Any language that has an HVSZK proof also has a public-coin HVSZK proof. This makes use of a zero-knowledge version of the set lower-bound protocol we saw a few lectures ago.
2. HVSZK = SZK. That is, any language that has a honest-verifier SZK proof also has an actual SZK proof. This follows from replacing the public coins in a public-coin HVSZK protocol with the outcome of a coin-tossing protocol. We will cover coin-tossing protocols towards the end of the course if we have time.
3. SZK is closed under complement. This follows from a statistical distance reversal procedure that makes clever use of hashing. This implies that $\mathrm{SD}^{\alpha,\beta}$ is also complete for SZK when $\alpha^2 > \beta$.
4. SZK is closed under composition with OR. That is, if $L_1, L_2 \in \mathsf{SZK}$, then the language L of tuples (x_1, x_2) such that either $x_1 \in L$ or $x_2 \in L$ is also contained in SZK. This follows from the completeness of SD and applications of Facts 0.1 and 0.2.
5. The above two statements imply that SZK is closed under composition with polynomial-sized Boolean formulas.
6. The existence of constant-round interactive proofs for the complete problem $\mathrm{SD}^{\alpha,\beta}$ implies that $\mathsf{SZK} \subseteq \mathsf{AM}$. The closure under complement further implies that $\mathsf{SZK} \subseteq \mathsf{AM} \cap \mathsf{coAM}$.

The class also has some interesting connections to cryptography. For instance, it is known that the existence of an average-case hard language in SZK implies the existence of one-way functions (and thus pseudorandom generators, commitment schemes, etc.) [Ost91]. The proof of this statement also becomes easier given the completeness of $\mathrm{SD}^{\alpha,\beta}$. At a very high level, average-case hardness of $\mathrm{SD}^{\alpha,\beta}$ implies the existence of two distributions that are statistically far but computationally indistinguishable. This is equivalent to the existence of one-way functions. Details are beyond the scope of this course, but refer to the note by Goldreich [Gol90].

Further, several problems that are central to cryptography such as quadratic residuosity, and several lattice problems [GG00], are contained in this class. The existence of a hard problem in this class is implied by some advanced cryptographic primitives like rerandomisable encryption.

3 Probabilistically Checkable Proofs

In a traditional proof system, the proof of a theorem consists of a sequence of assertions that are checked by the verifier in turn, with the proof being accepted if all of them check out. In the other kinds of proof systems we have seen so far, while there have been other resources available to the verifier and prover (interaction and randomness, in particular), the final decision to accept by the verifier is, in general, very similar – it looks at the entire transcript of its interaction with the prover, and decides whether to accept. In contrast, a *Probabilistically Checkable Proof (PCP)* is a proof that the verifier can verify by looking at just a small part of it.

Slightly more formally, a PCP for a theorem is a string π, potentially very long. A verifier of this proof is a randomised algorithm that picks a small set of coordinates of this string to look at (or "query"), and decides to accept or reject based on what the bits in those coordinates of π are. As always, the verification procedure should be such that a proof that makes the verifier accept should exist if the theorem is true, and if it is false, the verifier should reject any candidate proof with high probability.

The fact that such a proof is possible at all for non-trivial theorems is remarkable, and is yet another wonderful product of incorporating randomness into proof systems. Such proof systems were first studied in the early 90's, when the techniques developed in the study of interactive proofs made such proofs possible for non-trivial languages [BFLS91]. Interest in these proofs surged after it was discovered that the existence of PCPs for hard languages could be used to prove that certain optimisation problems were hard to solve even approximately [FGL+91]. Subsequent work culminated in the proof of the celebrated PCP theorem [AS92, ALM+92], one of the gems of theoretical computer science.

PCPs have since been used to prove the hardness of approximation of various problems of interest, but for decades they remained solely a theoretical tool. In the past few years, however, PCPs and related proof systems have found very interesting practical applications in constructions and implementations of succinct non-interactive arguments, which have found use in many cryptocurrencies. We will briefly look at both of these applications in coming lectures. An interesting account of the early development of research in interactive proofs and PCPs may be found in an article by O'Donnell [O'D05].

3.1 Defining PCPs

As usual, we will look at PCPs that prove membership in some language L. The verifier in this proof system will be a polynomial-time randomised algorithm V that gets an instance $x \in \{0,1\}^n$ as input, and outputs accept or reject. We will think of V as having oracle access to a proof string $\pi \in \{0,1\}^m$ that purports to be a PCP for the statement $x \in L$. V make queries of the form $i \in [m]$ to the oracle, in response to which it receives the bit $\pi[i]$, which is i bit of the proof string.

Definition 3.1. A PPT verification algorithm V is a *Probabilistically Checkable Proof system (PCP)* for a language L if the following are satisfied.

- **Completeness:** For any $x \in L$ there exists a proof π such that $V^{\pi}(x)$ always accepts.
- **Soundness:** For any $x \notin L$, and for all strings π^*, $V^{\pi^*}(x)$ accepts with probability at most $1/2$.

Note that, while we require the verifier to run in polynomial time, we do not even explicitly consider the prover that potentially came up with the proof π. The complexity of such a prover will become relevant in later applications to cryptographic protocols, but for the most part we will only concern

ourselves with the existence of proofs π, and not how they are computed. Instead, the parameters of a PCP that we will care about are the randomness complexity of the verifier (the number of random bits it uses), and the number of queries it makes (the number of bits of π it reads). Keeping track of the query complexity is natural since we care about proofs that need few queries to verify, and the reason we care about the verifier's randomness will become clearer when we get to the applications of PCPs. For now, note that these two quantities provide an upper bound on how long π needs to be (which has a bearing on the prover's complexity).

Definition 3.2. For any functions $q, \rho : \mathbb{N} \to \mathbb{N}$, the class $\mathsf{PCP}[q, \rho]$ consists of all languages that have a PCP V such that, when given input x, V makes at most $O(q(|x|))$ queries, and uses at most $O(\rho(|x|))$ bits of randomness. Further, V is said to be a PCP$[q, \rho]$ system.

For ease of notation, we will often fix the length of inputs x in our discussion, and use q and ρ as numbers rather than functions.

Claim 3.1. *In any* PCP$[q, \rho]$ *system V for a language, the valid proofs can be made to have length at most $q \cdot 2^{\rho}$.*

The constant $1/2$ in Definition 3.1 is again not important, as the soundness error can be decreased by repetition. We will be concerned only with non-adaptive verifiers – ones that make all their queries to the proof oracle at the same time, rather than making queries that depend on answers to previous queries (that is, behaving adaptively). In the regime of parameters we will mostly be interested in, this will not make a difference.

Claim 3.2. *If a language L has a* PCP$[q, \rho]$ *with an adaptive verifier, then it has a* PCP$[2^q, \rho]$ *with a non-adaptive verifier.*

Exercise 1. *Prove Claims 3.1 and 3.2.*

3.2 PCPs and IPs

The formulation of PCPs originally followed from that of multi-prover interactive proofs, which we will see later in the course if there is time. There are also simple transformations from any interactive proof to a PCP. This is illustrated by the following PCP for the GNI problem. Note that to describe a PCP, we need to describe what the proof string looks like for instances in the language, and what the verification algorithm is.

For GNI for any instance size n, given the proof π is indexed by graphs H over n vertices. That is, the length of the proof is equal to the number of different graphs over $[n]$, and each index $i \in [n]$ is mapped to a different graph H in some canonical manner. For an input (G_0, G_1), and any H over n vertices,

- $\pi[H] = 0$ if H is isomorphic to G_0,
- $\pi[H] = 1$ if H is isomorphic to G_1, and,
- $\pi[H] = 0$ or 1 arbitrarily if H is not isomorphic to either

Notice that this emulates the honest prover's algorithm in the IP for GNI. The verifier's algorithm similarly follows the verifier's algorithm in the IP given input (G_0, G_1):

1. Sample a random bit b and a random relabelling R
2. Compute $H \leftarrow R(G_b)$
3. Query $b' \leftarrow \pi[H]$, and accept iff $b = b'$

The proof of completeness and soundness of this PCP follows directly from those of the IP for GNI. The verifier here makes only one query, and the number of random bits it uses is 1 plus log of the number of possible relabellings of graphs on n vertices. This implies the following.

Claim 3.3. $GNI \in \mathsf{PCP}[1, n \log n]$

It also similarly turns out that this PCP is zero-knowledge for honest verifiers, but we will not be discussing this much. A similar approach to how we obtained this PCP may be used to obtain a PCP from any interactive proof, though with substantial loss in parameters.

Claim 3.4. *Suppose a language L has an interactive proof with $2k$ messages where the verifier has randomness complexity r, its messages are each of length at most a, and the prover's messages of length at most b. Then, L has a* $\mathsf{PCP}[bk, r]$ *proof of length at most $O(bk \cdot 2^{ak})$.*

Corollary 3.1. $\mathsf{IP} \subseteq \mathsf{PCP}[\mathsf{poly}, \mathsf{poly}]$

Exercise 2. *Prove Claim 3.4.*

References

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992* [DBL92], pages 14–23.

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992* [DBL92], pages 2–13.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31. ACM, 1991.

[DBL92] *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*. IEEE Computer Society, 1992.

[Din06] Irit Dinur. The PCP theorem by gap amplification. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 241–250. ACM, 2006.

[FGL+91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost np-complete (preliminary version). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 2–12. IEEE Computer Society, 1991.

[GG00] Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. *J. Comput. Syst. Sci.*, 60(3):540–563, 2000.

[Gol90] Oded Goldreich. A note on computational indistinguishability. *Inf. Process. Lett.*, 34(6):277–281, 1990.

[O'D05] Ryan O'Donnell. A history of the PCP Theorem. https://courses.cs.washington.edu/courses/cse533/05au/pcp-history.pdf, 2005. [Online; accessed 02-October-2021].

[Ost91] Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 - July 3, 1991*, pages 133–138. IEEE Computer Society, 1991.

[Vad99] Salil Pravin Vadhan. *A study of statistical zero-knowledge proofs*. PhD thesis, Massachusetts Institute of Technology, 1999.