Lecture 18: Zero-Knowledge Interactive Proofs (Cont.)

Notes by Yael Kalai

MIT - 6.5620

Lecture 18 (November 12, 2025)

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Outline

- Construct a ZK proof for the language of Quadratic Residues.
- Construct ZK proof for all of NP.
- Construct a commitment scheme from one-way functions.

Recap

Definition 1 (ZK)**.** Fix a language $L \in \mathsf{NP}$ with a corresponding NP relation R. We say that an interactive proof $(\mathcal{P}, \mathcal{V})$ for L is *(computational) zero-knowledge* if for every PPT verifier $\mathcal{V}^*$ there exists a PPT simulator $\mathcal{S}$ such that for every polynomials $\ell_1 = \ell_1(\lambda)$ and $\ell_2 = \ell_2(\lambda)$, the following holds:

For every $\lambda \in \mathbb{N}$ every $x \in L$ of size $|x| = \ell_1(\lambda)$, every witness w with $(x, w) \in R$, and every auxiliary input $z \in \{0,1\}^{\ell_2(\lambda)}$, the following ensembles (over the security parameter) are computationally indistinguishable:

$$\left\{ \mathsf{View}\Big((\mathcal{P}(w), \mathcal{V}^*)(1^\lambda, x, z) \Big) \right\}_{\lambda \in \mathbb{N}} \approx \left\{ S(1^\lambda, x, z) \right\}_{\lambda \in \mathbb{N}}.$$

In addition, it is required that $\mathcal{P}$ runs in polynomial time (given a witness w).

Remark. We say that an interactive proof $(\mathcal{P}, \mathcal{V})$ for L is (computational) honest-verifier ZK if the above holds only for the honest verifier $\mathcal{V}$.

Construction of ZK Protocol for the NP *Language* $\mathbb{QR}$

Goldwasser Micali and Rackoff [2] showed how to construct ZK proof for some NP languages. For example, they constructed a ZK proof for the language

$$\mathbb{QR} = \{(N, y) : \exists x \in \mathbb{Z}_N^* \text{ s.t. } y = x^2 \bmod N\}$$

The proof is very simple! Given an instance (N, y) it proceeds as follows:

1. The prover chooses a random $r \leftarrow \mathbb{Z}_N^*$ and sends to the verifier the element $s = r^2 \bmod N$.
2. The verifier chooses a random bit $b \leftarrow \{0,1\}$.
3. If $b = 0$ the prover sends $z = r$. If $b = 1$ the prover sends $z = r \cdot x$, where x is the witness (i.e., $y = x^2 \bmod N$).
4. If $b = 0$ the verifier accepts iff $z^2 = s \bmod N$. If $b = 0$ the verifier accepts iff $z^2 = s \cdot y \bmod N$.

This protocol is ZK. Let us first prove that it is honest-verifier ZK. The simulator will choose a random bit $b \leftarrow \{0,1\}$ on behalf of the verifier. If $b = 0$ it will first choose the prover's first message, by sampling a random $r \leftarrow \mathbb{Z}_N^*$ and setting $s = r^2 \bmod N$, and then set the prover's second message to be $z = r$. If $b = 1$ then it will first choose the prover's second message, by sampling a random element $z \leftarrow \mathbb{Z}_N^*$, and then set the prover's first message to be $s = z^2 \cdot y^{-1} \bmod N$. Note that this protocol is *perfect* honest-verifier zero-knowledge. Namely, the simulated view is identical to the real view.

This protocol is also (malicious verifier) ZK. The simulator is very similar to the one above. It will guess the (malicious) verifier's bit $b \leftarrow \{0,1\}$, and produce (s, z) as above. If $\mathcal{V}(x, s) = b$ then output (s, b, z) as the simulated transcript. Else, try again. Repeat for at most $k = \text{poly}(\lambda)$ times (to ensure the simulator runs in polynomial time). Note that this protocol is *statistical zero-knowledge*. Namely, the simulated view is statistically close to the real view. The reason the simulation is not perfect is that the simulator fails with probability 2^{-k}.

This protocol has perfect completeness and soundness $1/2$. To amplify the soundness we simply repeat the entire protocol.

Remark. To maintain the zero-knowledge property, we need to repeat the protocol *sequentially*. Sequential repetition preserves zero-knowledge. This is where the auxiliary information z is useful. We think of prior executions as auxiliary input.

It is tempting to repeat the protocol in parallel, to avoid blowing up the round complexity. Unfortunately, in general, parallel repetition does not preserve ZK (see HW 4). If all we want is honest-verifier ZK then we can repeat the protocol in parallel.

Constructing Zero-Knowledge Proofs for all of NP

Goldreich Micali and Wigderson proved that every language in NP has a zero-knowledge proof, assuming one-way functions exist [1]. In other words, every proof can be made zero-knowledge!

We will see a zero-knowledge proof for a specific NP-complete language called 3Col which contains the set of all graphs $G = (V, E)$ such that the set of vertices V can be colored by three colors: $C : V \to \{1, 2, 3\}$ such that no two adjacent vertices have the same color. Namely, for every $(u, v) \in E$, $C(u) \neq C(v)$.

We first show how to convert a coloring $C : V \to \{1, 2, 3\}$, which is a proof that the graph G is 3-colorable, into a "physical" zero-knowledge proof.

1. The prover does the following:
   (a) Choose a random permutation $\pi : \{1, 2, 3\} \to \{1, 2, 3\}$. Denote by $V = \{1, 2, \ldots, n\}$.
   (b) For every $i \in [n]$ place the color $\pi(C(i))$ in an opaque locked box and send the n locked boxes to the verifier.
2. The verifier chooses a random edge $(i, j) \in E$ and sends (i, j) to the prover.
3. The prover sends the keys that open only box i and box j.
4. The verifier accepts if and only if the colors in these boxes are distinct and are legal (i.e., belong to the set $\{1, 2, 3\}$).

Is this protocol zero-knowledge? It is definitely honest-verifier zero-knowledge since the the only thing the verifier learns is two distinct random colors. We can simulate this by choosing a random edge (u, v) (on behalf of the verifier), sampling two distinct random colors $c_u, c_v \in \{1, 2, 3\}$, storing these colors in the boxes corresponding to the vertices u and v, and then opening these boxes. This view is identical to the real view.

It is also (malicious verifier) zero-knowledge. The simulator can guess the verifier's edge (u, v) randomly, store the random distinct colors $c_u, c_v \in \{1, 2, 3\}$ in their boxes, give the boxes to the malicious verifier, and if the verifier sends back the edge (u, v) the continue the simulation. If not, try again!

It has completeness 1. The soundness is only $1 - \frac{1}{|E|}$ but can be amplified via repetitions. By repeating $k = |E| \cdot \lambda$, with probability $(1 - 1/|E|)^{|E| \cdot \lambda}$ the simulator will fail with probability $2^{-\Omega(\lambda)}$. Each time we repeat we need to choose a fresh permutation.

Currently, we think of the boxes as ideal opaque physical boxes. But soon we will replace them with a digital commitment to each color, and a malicious verifier can choose its edge as a function of this commitment string.

Remark. The above protocol is a physical protocol, where the prover sends opaque locked boxes. Such boxes have a digital analogue. This is called a commitment scheme.

Commitment Schemes

Definition 2. A commitment scheme corresponding to a message space $\mathcal{M}$ consists of a pair of algorithms $(\mathsf{Gen}, \mathsf{Com})$:

- Gen is a PPT algorithm that takes as input the security parameter 1^λ and outputs public parameters, denoted by $\mathsf{pp} \in \{0,1\}^{\mathrm{poly}(\lambda)}$.
- Com is a polynomial-time computable function that takes an input public parameters pp, a message $m \in \mathcal{M}$ and randomness $r \xleftarrow{R} \{0,1\}^\lambda$ and outputs a commitment $\mathsf{Com}(\mathsf{pp}, m, r)$.

One can think of both Gen and Com as randomized algorithms. We chose to explicitly include the randomness of Com, and hence think of it as being deterministic since when the commitment is opened the randomness is revealed

The following two properties are required to hold:

- **Hiding:** For every $m_0, m_1 \in \mathcal{M}$,

$$(\mathsf{pp}, \mathsf{Com}(\mathsf{pp}, m_0, r_0)) \approx (\mathsf{pp}, \mathsf{Com}(\mathsf{pp}, m_1, r_1))$$

  for $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ and $r_0, r_1 \xleftarrow{R} \{0,1\}^\lambda$.

- **Binding:** For every (even all-powerful) adversary $\mathcal{A}$:

$$\Pr_{\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)} [A(\mathsf{pp}) = (m_0, r_0, m_1, r_1) \text{ s.t. } m_0 \neq m_1 \ \wedge \ \mathsf{Com}(\mathsf{pp}, m_0, r_0) = \mathsf{Com}(\mathsf{pp}, m_0, r_0)] = \mathrm{negl}(\lambda).$$

Remark. Note that if we replace each physical opaque locked boxes with a commitment to to the relevant color, then the ZK protocol above becomes a computational ZK protocol, due to the computational hiding guarantee. It still has same completeness and soundness guarantees (up to a negligible loss in soundness due to the fact that there is a negligible probability that pp is sampled in a way that does not bind the adversary).

Remark. In the above definition, the hiding requirement is computational and the binding is statistical. One can define a commitment scheme to be statistically hiding and only computationally binding (as you will see in the HW). If we replace our opaque boxes with such a commitment then we obtain statistical ZK (due to the statistical hiding property of the commitment scheme), but get soundness only against computationally bounded cheating provers since an all-powerful cheating prover can break the binding of the commitment scheme. We will talk more about such computational soundness guarantee in the upcoming classes.

Constructing a Commitment Scheme

It is known how to construct a commitment scheme from the minimal assumption that one-way functions exist. We will see two constructions: The first assumes the existence of one-way functions. The second assumes the existence of injective one-way functions, but does not need any public parameters!

Construction from OWFs: Let $G : \{0,1\}^* \to \{0,1\}^*$ be a PRG that takes inputs of length λ to outputs of length 3λ length doubling pseudorandom generator (which we know how to construct from any OWF).

- $\mathsf{Gen}(1^\lambda)$ outputs a uniformly random string $u \leftarrow \{0,1\}^{3\lambda}$.
- $\mathsf{Com}(b, r) = G(r) \oplus b \cdot u$.

Hiding follows from the assumption that $G(r)$ is indistinguishable from uniform in $\{0,1\}^{3\lambda}$, which in turn implies that

$$G(r) \oplus u \approx G(r).$$

Binding follows from the fact that

$$\Pr_{u \leftarrow \{0,1\}^{3\lambda}}[\exists r, r' \in \{0,1\}^\lambda : G(r) \oplus u = G(r')] =$$
$$\Pr_{u \leftarrow \{0,1\}^{3\lambda}}[\exists r, r' \in \{0,1\}^\lambda : G(r) \oplus G(r') = u] \leq$$
$$\frac{2^{2\lambda}}{2^{3\lambda}} = 2^{-\lambda}.$$

Construction from injective OWF without pp*:* We will see a simple construction from injective one-way functions. This construction does not need pp.

Fix an injective one-way function $f : \{0,1\}^* \to \{0,1\}^*$. Let P be a (randomized) hardcore predicate corresponding to f. Define

$$\mathsf{Com}(b, (r, s)) = (f(r), s, P(r, s) \oplus b)$$

Note that this scheme has no public parameters. Computational hiding follows from the fact that P is a hardcore predicate of f, which by definition implies that

$$(f(r), s, P(r, s)) \approx (f(r), s, U)$$

where $U \leftarrow \{0,1\}$ is a uniformly random bit.

Binding follows from the fact that f is injective.

References

[1] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 171–185. Springer, 1986.

[2] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304. ACM, 1985.