Zero-Knowledge Proofs (continued)

MIT - 6.5620
Lecture 19 (Fall 2026)

> ***Warning:*** This document is a rough draft and may contain bugs. Please feel free to email us with corrections.

Recap

In the last class, we constructed the zero-knowledge proof for the language 3COL, and hence for all of NP, assuming the existence of commitment schemes.

We also saw zero-knowledge proofs of knowledge, where an extractor can obtain a witness from a successful prover by *rewinding* it and sending different challenges. As another example of zero knowledge for NP, the prover for graph Hamiltonicity commits to a random relabeling of the graph: the verifier asks either to see the full relabeled graph or to see only a Hamiltonian cycle. The simulator guesses which challenge will be asked and rewinds if necessary.

Outline

- Explain why interaction is necessary for zero knowledge without setup.
- Define non-interactive zero-knowledge.
- Construct NIZK for NP in the common reference string model, based on quadratic residuosity.

Is Interaction Necessary for Zero Knowledge?

Suppose that a language $\mathcal{L}$ has a one-message zero-knowledge proof system in the plain model, without a common reference string or other setup. Then $\mathcal{L} \in \mathsf{BPP}$. Indeed, on input x, run the zero-knowledge simulator to produce a proof π and check whether $V(x, \pi) = 1$. If $x \in \mathcal{L}$, completeness and zero knowledge imply that V accepts with high probability. If $x \notin \mathcal{L}$, soundness guarantees that V rejects even a proof generated by the simulator. Thus, non-interactive zero knowledge for hard languages requires some additional setup.

There are two main approaches to non-interactive zero knowledge: the Random Oracle Model (using the Fiat–Shamir transformation [3]) and the common reference string (CRS) model. We will focus on the latter, but first let us define non-interactive zero knowledge.

Non-Interactive ZK *(*NIZK*)*

Definition 1. A NIZK for a language $\mathcal{L} \in \mathsf{NP}$ consists of three (non-interactive) PPT algorithms (Gen, P, V):

1. Gen**:** Takes as input the security parameter 1^λ and outputs crs.
2. **Prover** P**:** Takes as input crs and an instance x and a corresponding witness w, and outputs a proof π.
3. **Verifier** V**:** Takes as input crs an instance x and a proof π and outputs a bit 0 or 1, indicating whether it rejects or accepts the proof.

The following three properties are required to hold:

- **Completeness:** There exists a negligible function μ such that for every $\lambda \in \mathbb{N}$ and every $(x, w) \in \mathcal{R}_\mathcal{L}$,

  $$\Pr[V(\mathsf{crs}, x, P(\mathsf{crs}, x, w)) = 1] \geq 1 - \mu(\lambda)$$

  where the probability is over $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$ and over the randomness of the prover P.
- **(Adaptive) Soundness:** For every poly-size (cheating) prover P^* there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$,

  $$\Pr[P^*(\mathsf{crs}) = (x, \pi) : \ x \notin \mathcal{L} \ \wedge \ V(\mathsf{crs}, x, \pi) = 1] \leq \mu(\lambda)$$

  where the probability is over $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$.
- **(Non-Adaptive) Zero-knowledge:** There exists a PPT simulator S, such that for every polynomial time instance generator algorithm $\mathcal{I}$, that on input 1^λ outputs $(x, w) \in \mathcal{R}_\mathcal{L}$,

  $$(\mathsf{crs}, x, P(\mathsf{crs}, x, w)) \approx S(1^\lambda, x)$$

  where $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$ and $(x, w) = \mathcal{I}(1^\lambda)$.

Remark. One can strengthen the above zero-knowledge condition to be multi-theorem ZK which asserts that there exists a simulator that simulates multiple proofs (as opposed to only one). Indeed, this would be a more meaningful definition. One can also strengthen the definition by considering an adaptive version, where the instance generator algorithm $\mathcal{I}$ takes as input crs and may generate the instances (x, w) as a function of crs. Indeed, this adaptive multi-instance ZK version is the desired one.

The Common Reference String Model

In the common random string model, a trusted party generates a uniformly random string crs and gives it to both the prover and the verifier before the proof. More generally, in the common *reference* string model the string is sampled according to a specified distribution $\mathsf{Gen}(1^\lambda)$, for example a distribution of products of two primes. In either case, the CRS is sampled independently of the instance to be proved. The prover then sends a single message π to the verifier.

The first construction of NIZK for NP was given by Blum, Feldman, and Micali [1], based on quadratic residuosity. Subsequent constructions were given by Feige, Lapidot, and Shamir [2], based on factoring, but focus on the first construction. It has two steps: first construct NIZK for a special language concerning quadratic non-residuosity, and then construct NIZK for 3SAT, obtaining NIZK for all of NP.

Quadratic Residuosity: A Review

Let N be an odd integer, and let $\mathbb{Z}_N^*$ be its multiplicative group. An element $a \in \mathbb{Z}_N^*$ is a *quadratic residue* if there exists $r \in \mathbb{Z}_N^*$ such that $a = r^2 \bmod N$. Denote the set of quadratic residues by QR_N.

For an odd prime p, the Legendre symbol $(\frac{a}{p})$ is 1 if a is a quadratic residue modulo p, and -1 otherwise. For $N = \prod_{j=1}^{k} p_j^{e_j}$, the *Jacobi symbol* is

$$\left(\frac{a}{N}\right) = \prod_{j=1}^{k} \left(\frac{a}{p_j}\right)^{e_j}.$$

A useful fact is that the Jacobi symbol can be computed in polynomial time without factoring N. Let

$$J_N^+ = \left\{a \in \mathbb{Z}_N^* : \left(\frac{a}{N}\right) = 1\right\}, \qquad \mathsf{QNR}_N = J_N^+ \setminus \mathsf{QR}_N.$$

Thus every square has Jacobi symbol 1, but the converse is not necessarily true.

Good integers. We say that an odd integer N is *good* if it is neither a prime power nor a perfect square, and exactly half the elements of J_N^+ are quadratic residues. Equivalently, $N = p^i q^j$, for distinct odd primes p, q and $i, j \geq 1$ not both even. In particular, if N is good and $y \in \mathsf{QNR}_N$, then for every $r \in J_N^+$, exactly one of r and yr is a quadratic residue. The product of two elements of QNR_N is a quadratic residue.

For a general odd N with k distinct prime factors, precisely a 2^{-k} fraction of $\mathbb{Z}_N^*$ are quadratic residues. If N is not a perfect square,

then half the elements of $\mathbb{Z}_N^*$ lie in J_N^+; for $k \geq 3$, at most a quarter of these are quadratic residues. These facts are useful for soundness.

Quadratic Residuosity Assumption (QRA). Let $N = pq$ for suitable random large primes p, q. The quadratic residuosity assumption asserts that, given only N, a random element of QR_N is computationally indistinguishable from a random element of QNR_N.

NIZK *for Quadratic Non-Residuosity*

Consider the language GOOD of pairs (N, y) such that N is good and $y \in \mathsf{QNR}_N$. The prover can use the factorization of N as a witness. We first describe a protocol with an idealized CRS that depends on N; we will then explain the problem with this assumption.

Protocol.

1. The common string consists of $m = \text{poly}(\lambda)$ uniformly random, independent elements $r_1, \ldots, r_m \in J_N^+$.
2. For every $i \in [m]$, the prover sends a bit $b_i \in \{0, 1\}$ and an element $s_i \in \mathbb{Z}_N^*$ satisfying
$$s_i^2 = y^{b_i} r_i \pmod{N}.$$
In other words, the prover sends either a square root of r_i or a square root of yr_i.
3. The verifier checks that N is odd, is neither a prime power nor a perfect square, that $y \in J_N^+$, and that all the above equations hold.

Completeness. If $(N, y) \in \mathsf{GOOD}$, then for each $r_i \in J_N^+$ exactly one of r_i and yr_i is a quadratic residue. Thus, a prover that knows the factorization of N can always compute the required square roots (chosen uniformly among the possible roots).

Soundness. Suppose $(N, y) \notin \mathsf{GOOD}$ and the verifier's checks on N and the Jacobi symbol pass. If N has exactly two distinct prime factors, then N is good and necessarily $y \in \mathsf{QR}_N$, so both r_i and yr_i have square roots for only half of the choices of r_i. If N has at least three distinct prime factors, then at most a quarter of J_N^+ consists of quadratic residues; hence, for at least half the choices of r_i, neither r_i nor yr_i is a quadratic residue. In either case, the probability that all m tests pass is at most 2^{-m}.

Perfect zero knowledge. The simulator chooses independently $b_i \xleftarrow{R} \{0,1\}$ and $s_i \xleftarrow{R} \mathbb{Z}_N^*$ and sets

$$r_i = y^{-b_i} s_i^2 \pmod{N}.$$

When $(N, y) \in \mathsf{GOOD}$, this generates precisely the distribution of the real CRS together with the proof; in particular, the simulator does not need the factorization of N.

Remark. The preceding protocol is not yet a genuine CRS-model NIZK, because the distribution of $r_i \in J_N^+$ depends on the input N. Instead we use a sufficiently long uniformly random CRS, independent of the statement, and interpret and filter its bits to obtain elements of J_N^+ (up to negligible sampling error). The same protocol can then be used, with suitable parameters to ensure soundness even when the statement is chosen after seeing the CRS.

NIZK *for* 3SAT

Let φ be a 3-CNF formula with n variables and m clauses. The prover knows a satisfying assignment $w = (w_1, \ldots, w_n)$ and wants to prove that φ is satisfiable without revealing w. Recall that 3SAT is NP-complete.

1. **Choose (N, y).** The prover chooses a good integer $N = pq$, together with $y \in \mathsf{QNR}_N$, and sends a NIZK proof that $(N, y) \in \mathsf{GOOD}$, using the preceding protocol.

2. **Encode the satisfying assignment.** For each variable x_i, choose a random $r_i \in \mathbb{Z}_N^*$ and let

$$z_i = \begin{cases} r_i^2 \bmod N & \text{if } w_i = 0, \\ y r_i^2 \bmod N & \text{if } w_i = 1. \end{cases}$$

   The encoding of the literal x_i is z_i, and that of $\neg x_i$ is $yz_i \bmod N$. Since $y \in \mathsf{QNR}_N$, exactly one of these two encodings is a quadratic non-residue. Thus a literal is true if and only if its encoding is in QNR_N.

3. **Prove that each clause is satisfied.** Consider a clause with encoded literals (a_1, b_1, c_1). To show that the clause is true, it suffices to show that its quadratic-residuosity *signature* is not $(\mathsf{QR}, \mathsf{QR}, \mathsf{QR})$. The prover generates seven additional triples (a_j, b_j, c_j), for $j = 2, \ldots, 8$, so that the eight triples represent all eight possible signatures. It also reveals the square roots of the three elements of one designated auxiliary triple, proving that its signature is $(\mathsf{QR}, \mathsf{QR}, \mathsf{QR})$.

The remaining task is a *proof of coverage*. For each of polynomially many triples (r, s, t) obtained from the CRS, the prover identifies some $j \in [8]$ and reveals square roots of

$$ra_j, \quad sb_j, \quad tc_j \pmod{N}.$$

This proves that (r, s, t) and (a_j, b_j, c_j) have the same signature. With overwhelming probability, the random triples include all eight signatures. Consequently, all eight auxiliary and original triples must have different signatures, and the original triple cannot have the all-residue signature already assigned to an auxiliary triple.

Completeness and soundness. The honest prover can construct the required eight triples and provide all the square roots. Soundness follows from the soundness of the proof for (N, y) and of the proof of coverage. The full analysis, including the CRS parameters, is left as an exercise (as in the lecture).

Zero knowledge. The simulator instead chooses $y \in \mathsf{QR}_N$ and simulates the proof for (N, y). It can then choose encodings so that *both* a variable and its negation are encoded as quadratic non-residues. Hence all literals can appear true, and the simulator can construct all clause proofs without knowing a satisfying assignment. Under the QRA, the simulated view is computationally indistinguishable from the real one. This gives computationally zero-knowledge, statistically sound NIZK for all of NP in the CRS model.

Applications and Open Questions

Zero-knowledge proofs can prevent identity theft, prove properties of secrets, and help transform secure protocols against curious adversaries into protocols secure against malicious adversaries. Extensions include non-malleable, concurrent, resettable, and quantum zero knowledge. Open questions include finding natural complete problems for perfect and computational zero knowledge and understanding whether statistical zero knowledge coincides with perfect zero knowledge.

References

[1] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88,

page 103–112, New York, NY, USA, 1988. Association for Computing Machinery.

[2] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *31st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 308–317. IEEE, 1990.

[3] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. 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 186–194. Springer, 1986.