Zero-Knowledge Proofs (continued)

MIT - 6.5620
Lecture 18 (Fall 2026)

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

Recap

In the previous lecture we defined interactive proofs and zero-knowledge (ZK) proofs, and saw perfect zero-knowledge protocols for quadratic residuosity and graph isomorphism. Recall that a ZK proof is an interactive proof $(\mathcal{P}, \mathcal{V})$ for which the view of any polynomial-time verifier $\mathcal{V}^*$, on a true statement, can be simulated without the prover's witness. Depending on whether the simulated view is identical, statistically close, or computationally indistinguishable from the real view, we obtain perfect, statistical, or computational ZK. When discussing perfect simulation by rewinding, we allow expected-polynomial-time simulators; a strict polynomial-time cutoff instead gives statistically close simulation.

In the protocols from the previous lecture, the honest prover can run in polynomial time given an NP witness (e.g., a square root for quadratic residuosity or an isomorphism for graph isomorphism). Soundness, however, holds against even computationally unbounded cheating provers. What happens when the statement does not have a known short witness?

Graph Non-Isomorphism

Let

$$\mathsf{GNI} = \{(G, H) : G \ncong H\}$$

be the language of pairs of non-isomorphic graphs. Unlike graph isomorphism, it is not known whether GNI belongs to NP. One can check non-isomorphism by exhaustively testing all $n!$ permutations of the vertices, but this is an exponential-time procedure, not a known lower bound on the length of classical proofs.

There is nevertheless a particularly simple interactive proof for GNI. For simplicity, assume that G and H are graphs on the common vertex set $[n]$; graphs with different numbers of vertices are trivially non-isomorphic.

The protocol.

1. The verifier chooses a uniform bit $c \leftarrow \{0,1\}$ and a uniform permutation $\rho : [n] \to [n]$. It sets

$$K = \begin{cases} \rho(G) & c = 0, \\ \rho(H) & c = 1, \end{cases}$$

   and sends K to the prover.

2. The prover determines whether K is isomorphic to G or to H, and sends a bit b: $b = 0$ if $K \cong G$, and $b = 1$ if $K \cong H$.

3. The verifier accepts if and only if $b = c$.

Completeness and soundness. If $G \ncong H$, the graph K is isomorphic to exactly one of them. An all-powerful honest prover therefore always returns $b = c$, so completeness is perfect. If $G \cong H$, the distribution of K is identical for $c = 0$ and $c = 1$: in both cases, it is a uniformly random relabeling of the same graph. Thus K gives even an unbounded cheating prover no information about c. For every $\mathcal{P}^*$,

$$\Pr[(\mathcal{P}^*, \mathcal{V})(G, H) = 1] \leq \frac{1}{2} \qquad \text{if } G \cong H.$$

Repeating the protocol independently λ times, and accepting only if all repetitions accept, reduces the soundness error to $2^{-\lambda}$.

Honest-verifier zero knowledge. On a true statement $G \ncong H$, the honest verifier's view can be sampled without the prover: choose c and ρ as the verifier would, set $K = \rho(G)$ or $\rho(H)$ accordingly, and set the prover's reply to $b = c$. This is exactly the real view (including the verifier's random coins). Hence the protocol is perfect *honest-verifier* ZK.

What about a malicious verifier? Here we must be careful. A malicious $\mathcal{V}^*$ need not generate K by randomly permuting one of the two input graphs. It can send a graph K of its choice, thereby using the prover to learn which of G, H is isomorphic to K. The argument above therefore does *not* establish ZK against malicious verifiers.

One possible remedy is to have the verifier first give a zero-knowledge *proof of knowledge* of a permutation ρ and a bit c such that $K = \rho(G)$ if $c = 0$, or $K = \rho(H)$ if $c = 1$. This motivates a new notion: proving not merely that a statement is true, but that the prover knows a witness for it.

Zero-Knowledge Proofs of Knowledge

So far we have focused on decision problems, namely whether $y \in L$. Sometimes the statement itself is easy, but the prover wants to convince the verifier that it *knows a solution*. For example, let $\langle g \rangle$ be a group of prime order q and let $y = g^x$ be public. Every $y \in \langle g \rangle$ has a discrete logarithm, but it may be computationally difficult to find x. Alice wants to prove that she knows such an x, without revealing it.

A *proof of knowledge* is required to have a *knowledge extractor*: an efficient algorithm that, by interacting with a prover that convinces the verifier sufficiently often, can recover a corresponding witness. We will illustrate the idea without giving the full formal definition of an extractor, as we did in class.

A Proof of Knowledge of a Discrete Logarithm

For concreteness, one can take a prime $p = 2q + 1$ with q prime and choose $g \in \mathbb{Z}_p^*$ of order q; all exponent computations below are modulo q. The public input is $y = g^x \pmod{p}$, and the honest prover knows $x \in \mathbb{Z}_q$.

The protocol.

1. The prover chooses $r \xleftarrow{R} \mathbb{Z}_q$ and sends $z = g^r \pmod{p}$.
2. The verifier chooses a uniformly random challenge $c \xleftarrow{R} \{0, 1\}$.
3. The prover sends $s = r + cx \pmod{q}$.
4. The verifier accepts if and only if

$$g^s = zy^c \pmod{p}.$$

Completeness. For the honest prover, $g^s = g^{r+cx} = g^r(g^x)^c = zy^c$, so the verifier always accepts.

Extraction. Fix the first message z and the cheating prover's internal state immediately after sending it. Suppose an extractor can obtain accepting answers s_0 to challenge 0 and s_1 to challenge 1, by rewinding to this state. Then

$$g^{s_0} = z, \qquad g^{s_1} = zy,$$

and therefore $g^{s_1 - s_0} = y$. Thus $x = s_1 - s_0 \pmod{q}$ is a discrete logarithm of y. Two accepting transcripts with the same first message and different challenges determine a witness.

Why can an extractor obtain both transcripts? If a cheating prover is accepted with probability $1/2 + \varepsilon$, where ε is non-negligible, then

with probability at least 2ε (over the first message and the prover's state) it can answer *both* challenges. The extractor fixes this state and rewinds the prover, querying both challenges. Repeating this experiment extracts a witness in expected polynomial time when ε is inverse polynomial. This is stronger than ordinary soundness, since the statement $y \in \langle g \rangle$ is always true.

Zero knowledge. Against an honest verifier, a simulator chooses uniform $c \in \{0, 1\}$ and $s \in \mathbb{Z}_q$, and sets

$$z = g^s y^{-c} \pmod{p}.$$

It outputs (z, c, s). This has exactly the same distribution as the honest execution: in either case c and s are uniform, and the verification equation determines z. In fact, with a one-bit challenge this is also perfect ZK against a malicious verifier, using the same guess-and-rewind method as for quadratic residuosity. The simulator guesses the verifier's challenge, prepares (z, s) as above, and retries if the actual challenge differs. Since z is uniform and independent of the guessed challenge, each attempt succeeds with probability $1/2$, when assuming the prover aborts on invalid challenges and that the simulator handles such executions appropriately.

Larger Challenges

We can strengthen the protocol by replacing the one-bit challenge with a uniformly random element $c \xleftarrow{R} \mathbb{Z}_q$. The prover still sends $s = r + cx \pmod{q}$, and the verifier still checks $g^s = zy^c$.

Extraction works in the same way, except that we need two *distinct* accepting challenges $c_0 \neq c_1$, not necessarily 0 and 1. Given responses s_0, s_1 to the same first message z, we obtain

$$g^{s_1 - s_0} = y^{c_1 - c_0}, \qquad x = (s_1 - s_0)(c_1 - c_0)^{-1} \pmod{q}.$$

The inverse exists because q is prime. More generally, a prover that is accepted with probability noticeably larger than $1/q$ can be rewound to obtain two accepting challenges with non-negligible probability, and hence a witness.

Remark. Increasing the size of the challenge space improves the knowledge error (from $1/2$ to $1/q$), but the simple malicious-verifier ZK simulator that *guesses* the verifier's challenge would now succeed with probability only $1/q$. If q is exponential in the security parameter, this is not an efficient simulation. The larger-challenge protocol is still perfect *honest-verifier* ZK by sampling c, s first, but the argument above does not prove ZK against arbitrary malicious verifiers.

Zero knowledge versus proof of knowledge. These properties protect against different adversaries. ZK says that a malicious *verifier* learns nothing beyond what it could simulate. Knowledge soundness says that a successful malicious *prover* must in a meaningful sense possess a witness, because an extractor can recover it.

Public-Coin Interactive Proofs

Our original definition of interactive proofs allowed the verifier to keep its randomness secret. Is private randomness essential for the expressive power of interactive proofs? Remarkably, it is not: the Goldwasser–Sipser transformation shows that private-coin interactive proofs can be converted into public-coin ones, in which the verifier's messages consist of fresh random coins. We will see an important idea underlying this result: an interactive protocol for demonstrating that a set is large.

Proving That a Set Is Large

Let $S \subseteq U = \{0,1\}^n$ be a set for which membership can be efficiently verified, possibly with an NP witness. Simply sampling uniformly from U and counting hits in S is useful only when S has noticeable density in U. Instead, we will hash U into a much smaller set, $\{0,1\}^m$.

Suppose we have an efficiently samplable family $\mathcal{H}$ of hash functions $h : U \to \{0,1\}^m$ such that for any distinct $z, z' \in U$,

$$\Pr_{h \xleftarrow{R} \mathcal{H}}[h(z) = h(z')] \leq 2^{-m},$$

and each $h(z)$ is uniformly distributed in $\{0,1\}^m$. Such a family is called *universal*. We will construct one below.

The set-size protocol.

1. The verifier samples $h \xleftarrow{R} \mathcal{H}$ and $y \xleftarrow{R} \{0,1\}^m$, and sends (h, y) to the prover.
2. The prover tries to find $z \in S$ such that $h(z) = y$, and sends z together with a proof of membership in S (if needed).
3. The verifier accepts if and only if $z \in S$ is verified and $h(z) = y$.

Soundness. If $|S| \leq s2^m$, then for any (even unbounded) cheating prover,

$$\Pr[\mathcal{V} \text{ accepts}] \leq \Pr[y \in h(S)] \leq \frac{|S|}{2^m} \leq s.$$

The middle inequality is a union bound over the at most $|S|$ possible hash values of elements of S.

Completeness. If $|S| \geq c2^m$, then an all-powerful prover succeeds with probability at least $c/(1+c)$. To see this, let X be the number of $z \in S$ satisfying $h(z) = y$ and let $\mu = \mathbb{E}[X] = |S|/2^m$. Universality gives

$$\mathbb{E}[X^2] \leq \mu + \mu^2.$$

By the second-moment inequality,

$$\Pr[X > 0] \geq \frac{(\mathbb{E}X)^2}{\mathbb{E}[X^2]} \geq \frac{\mu}{1+\mu} \geq \frac{c}{1+c}.$$

Thus the probability that the random target has a preimage in S reflects the size of S. With a suitable gap between the sizes in the two cases, this gives an interactive proof.

Universal Hash Functions

There is a simple, explicit construction that requires no computational hardness assumption. For $A \in \mathbb{F}_2^{m \times n}$ and $b \in \mathbb{F}_2^m$, define

$$h_{A,b}(z) = Az + b \pmod 2.$$

Choose A and b uniformly at random. For every fixed z, the output is uniform. Moreover, for distinct z, z', the nonzero vector $z - z'$ satisfies

$$\Pr_{A,b}[h_{A,b}(z) = h_{A,b}(z')] = \Pr_{A}[A(z - z') = 0] = 2^{-m}.$$

In fact, the two hash values are pairwise independent (and you are familiar with this definition from problem set 4). Sampling and evaluating this hash family both take polynomial time. For graph nonisomorphism, an appropriately defined set (accounting for automorphisms) has size $n!$ if the graphs are isomorphic and $2n!$ otherwise. More generally, Goldwasser–Sipser use such set-size protocols to transform private-coin interactive proofs into public-coin ones.

Constructing Zero-Knowledge Proofs for All of NP

We saw examples of *perfect* zero-knowledge proofs. Can every language in NP have a perfect ZK proof? Not unless an unlikely complexity-theoretic collapse occurs: results of Aiello–Håstad and Fortnow imply that if every NP language has a statistical (and hence if every one has a perfect) ZK proof, then the polynomial hierarchy collapses.

Nevertheless, the following theorem shows that computational ZK is vastly more general.

Theorem (Goldreich–Micali–Wigderson, [1]). Assuming one-way functions exist, every language in NP has a computational zero-knowledge interactive proof.

In other words, every statement with a short, efficiently verifiable witness can be proved in zero knowledge! We will see the main construction for the NP-complete language 3Col; composing it with an NP reduction gives a ZK proof for any language in NP.

Zero-Knowledge Proof for 3-Coloring

The language 3Col consists of all graphs $G = (V, E)$ such that the vertices can be colored with three colors, $C : V \to \{1, 2, 3\}$, and adjacent vertices receive different colors: for every $(u, v) \in E$, $C(u) \neq C(v)$. Such a coloring is an NP witness for membership in 3Col.

Commitment Schemes

A commitment scheme, that we met in lecture 13, is the digital analogue of a sealed box. The sender first commits to a bit b, concealing it from the receiver, and can later open the commitment. The two main requirements are *hiding* (the receiver cannot learn b before the opening) and *binding* (the sender cannot successfully open the same commitment to two different bits).

Definition 1 (Commitment scheme)**.** In the simple noninteractive commit/open form, a commitment scheme consists of a setup algorithm Gen and a polynomial-time computable function Com:

- $\mathsf{Gen}(1^\lambda)$ outputs public parameters pp (possibly empty).
- To commit to $b \in \{0, 1\}$, the sender samples random coins r and sends $d = \mathsf{Com}(\mathsf{pp}, b; r)$. To open, the sender reveals (b, r); the receiver accepts if $d = \mathsf{Com}(\mathsf{pp}, b; r)$.

We require:

- **Computational hiding:** for $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ and independently uniform r_0, r_1, we have
$$(\mathsf{pp}, \mathsf{Com}(\mathsf{pp}, 0; r_0)) \approx_c (\mathsf{pp}, \mathsf{Com}(\mathsf{pp}, 1; r_1)).$$
- **Statistical binding:** except with negligible probability over the setup pp, there do not exist r_0, r_1 such that
$$\mathsf{Com}(\mathsf{pp}, 0; r_0) = \mathsf{Com}(\mathsf{pp}, 1; r_1).$$
If this is impossible for every pp, the scheme is *perfectly binding*.

The same idea extends to messages from a finite message space (in particular, the three possible colors), for example by committing independently to the bits of an encoding of the message. Recall that in Lecture 13 we constructed such commitments from one-way functions.

The ZK Protocol for 3-Coloring

In each repetition, the prover chooses a fresh random permutation π of a valid 3-coloring C, and for each vertex i sends an independently randomized commitment

$$d_i = \mathsf{Com}(\mathsf{pp}, \pi(C(i)); r_i).$$

(For the PRG-based construction from Lecture 13, the verifier first sends u and $\mathsf{pp} = u$.) The verifier chooses a uniformly random edge $(i, j) \in E$. The prover sends the openings $(\pi(C(i)), r_i)$ and $(\pi(C(j)), r_j)$. The verifier checks the openings and accepts if the two colors are legal and distinct.

Completeness follows immediately. Statistical binding ensures that, except with negligible probability over the honest setup, a cheating prover has effectively fixed the colors of all vertices before seeing the challenge. Hence on $G \notin \mathsf{3Col}$ the soundness error is at most $1 - 1/|E|$ plus a negligible term; sequential repetition reduces this to negligible.

The Simulator for a Malicious Verifier

Fix a polynomial-time malicious verifier $\mathcal{V}^*$, with any auxiliary input. The simulator $\mathcal{S}$ does not know a 3-coloring of G. It does the following:

1. Run the verifier's initial setup message, if the commitment scheme has one. Choose a uniformly random edge $(i^*, j^*) \in E$ as a *guess* for the verifier's challenge.
2. Give vertices i^* and j^* uniformly random, distinct colors from $\{1, 2, 3\}$, color all other vertices arbitrarily (say, red), and generate the corresponding commitments using fresh randomness.
3. Feed the commitments to $\mathcal{V}^*$. If it sends a valid edge $(i, j) \neq (i^*, j^*)$, restart with fresh verifier coins. If it aborts or sends an invalid challenge, output the corresponding view (with the honest prover aborting) with probability $1/|E|$, and otherwise restart.
4. If $(i, j) = (i^*, j^*)$, open the commitments of i and j and output the resulting view of $\mathcal{V}^*$ (including its coins).

We must show two things: that the simulator runs in expected polynomial time, and that its output is computationally indistinguishable from the real verifier's view. Unlike ideal boxes, the digital commitments reveal strings that depend on the coloring, so these claims require the hiding property.

Consider the following three experiments, including all of the verifier's random coins in its view.

- **Hybrid 0 (the simulator).** Run the simulator above, including its handling of aborts and invalid challenges.
- **Hybrid 1.** Run the same guess-and-restart experiment, but instead of the arbitrary coloring, use a *proper* 3-coloring with a uniformly random permutation of the colors. This hybrid is a conceptual experiment: it uses a witness and is not intended as an efficient simulator without a witness.
- **Hybrid 2 (the real view).** Run the honest prover with a randomly permuted proper coloring and output the malicious verifier's view, without guessing an edge or restarting.

Hybrid 0 versus Hybrid 1. The two endpoints of the guessed edge receive uniformly random distinct colors in *both* experiments, so their openings have the same distribution. The committed colors of the other vertices may differ, but they are never opened on a successful attempt. By the computational hiding of the commitment scheme, a polynomial-time malicious verifier cannot distinguish the corresponding vectors of commitments, up to a negligible advantage (using a hybrid over the polynomially many unopened commitments). This remains true when the verifier chooses the setup string u, because the PRG-based construction is hiding for every u. The same argument applies to the event that the verifier's chosen edge equals (i^*, j^*).

Expected running time. In Hybrid 1, each attempt is retained with probability exactly $1/|E|$: a valid challenge matches the independent uniform guess with this probability, while aborts and invalid challenges are retained with the same probability. By the indistinguishability argument above, the success probability in a single attempt of Hybrid 0 is $1/|E| \pm \mathrm{negl}(\lambda)$. In particular, it is $\Omega(1/|E|)$, and independent attempts give expected $O(|E|)$ trials. Alternatively, we can truncate after polynomially many trials, failing with only negligible probability.

Hybrid 1 versus Hybrid 2. These distributions are *identical*: Hybrid 1 retains every execution with probability exactly $1/|E|$, independently

of the verifier's view, including when the verifier aborts or sends an invalid challenge.

Combining the two steps shows that Hybrid 0 is computationally indistinguishable from the real view. Therefore the commitment-based 3-coloring protocol is computational ZK against *arbitrary polynomial-time malicious verifiers*. Sequential repetition preserves this property and amplifies soundness to negligible.

Conclusion. Together with the construction of commitments from one-way functions established in Lecture 13 and the NP-completeness of 3Col, this establishes the Goldreich–Micali–Wigderson theorem: assuming one-way functions, all of NP has computational zero-knowledge proofs.

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.