1 Recap: Zero-Knowledge Proofs

Definition 1.1. A zero-knowledge interactive proof system with soundness error $s \in [0,1]$ for a language $L \subseteq \{0,1\}^*$ is a pair of algorithms: a (possibly computationally unbounded) prover P, and a ppt verifier V, having the following properties:

1. Completeness: for all $x \in L$, $\text{out}_V[P(x) \leftrightarrow V(x)] = 1$, with probability 1.
2. Soundness: for any $x \notin L$ and any (possibly unbounded) P^*,
$$\Pr\left[\text{out}_V[P^*(x) \leftrightarrow V(x)] = 1\right] \leq s.$$
3. Zero-knowledgeness: for every nuppt (possibly cheating) verifier V^*, there exists an nuppt simulator $\mathcal{S}$ such that for all $x \in L$,
$$\text{view}_{V^*}[P(x) \leftrightarrow V^*(x)] \approx \mathcal{S}(x).$$

The type of indistinguishability in the zero-knowledge condition can either be statistical (i.e., $\overset{s}{\approx}$) or computational (i.e., $\overset{c}{\approx}$). We also say that the proof system is efficient if the prover's strategy can be implemented by a randomized algorithm $P(x,w)$ that runs in time polynomial in the length of its first argument, and any $x \in L$ has a "hint" (or witness) w for which P functions as it should.

2 Which Languages Have ZKPs?

We've seen zero-knowledge (sometime just honest-verifier ZK) proofs for graph non-isomorphism and graph isomorphism. There are similar proofs for quadratic residuosity and non-residuosity modulo $N = pq$. (As an exercise, try to design them.) What other languages have zero-knowledge proofs? If we had to start from scratch every time we wanted a ZKP for a new language, it would be very cumbersome and time-consuming. Here, we will see a very powerful and general result of Goldreich, Micali and Wigderson: informally, it says that "any language for which membership can be efficiently verified can be proved in zero-knowledge."

Theorem 2.1. Assume that there exists a statistically binding / computationally hiding commitment scheme, defined below. In particular, such a commitment scheme can be constructed from any one-way permutation, or even any one-way function. Then for any NP-language L (also defined below), there is a zero-knowledge proof system (with efficient prover) for L.

2.1 Preliminaries

2.1.1 NP and NP-hardness

For completeness, we recall the definition of the class NP of languages that are decidable in nondeterministic polynomial time.

Definition 2.2. A language $L \subseteq \{0,1\}^*$ is said to be in NP if there exists a deterministic algorithm $W(x,w)$, running in time polynomial in the length of its first argument x, such that $x \in L$ if and only if there exists some $w \in \{0,1\}^*$ for which $W(x,w)$ accepts.

(Note that even though W is a deterministic machine, its second argument w captures the nondeterminism in the definition.) If $W(x, w)$ accepts, we say that w is a "witness" (or a "certificate," or a "proof") that $x \in L$. Recall that any language $L \in \mathsf{P}$ (i.e., one for which membership can be decided in deterministic polynomial time) is also in NP: the machine $W(x, w)$ can just ignore its second argument and decide whether $x \in L$ on its own. But NP also contains many languages that are not believed to be in P, such as $\mathsf{3SAT}$, the language of all satisfiable 3CNF formulas; a witness w is a satisfying assignment for the given formula.

Also recall that L' is said to be an NP-hard language if for every NP-language L, there is a deterministic polynomial-time algorithm (a reduction) R_L such that $x \in L$ if and only if $R_L(x) \in L'$. A language L' is said to be NP-complete if it is NP-hard, and is itself in NP. Many important problems are NP-complete, including $\mathsf{3SAT}$, graph 3-colorability (defined below), graph Hamiltonicity, and so on.

2.1.2 Commitment Scheme

Consider the following scenario. Alice and Bob are betting on the result of a coin toss: Bob tosses the coin and Alice calls it while it's still in the air. This game is trivial to analyze. But what if Alice and Bob are not at the same location — what if they are playing this game over the phone? If Alice announces her call first, then Bob can lie about the outcome of his coin toss and always win. If Bob announces the outcome first, then Alice can just call the same thing and always win. In order to ensure fair play, we want Alice to commit to her call before the coin flip, in a way that conceals her choice from Bob but does not allow her to change her call after Bob announces the outcome of the coin flip. The idea is that Alice can run a randomized algorithm Com on her guess, then later reveal it and the random coins she used when running Com. This should allow Bob to check that she revealed her commitment correctly, while hiding the value of her guess until it is revealed. A formal definition follows.

Definition 2.3. A statistically binding, computationally hiding commitment scheme for message space $\{0,1\}^\ell$ is a ppt algorithm Com having the following properties:

- Statistical (perfect) binding: for all distinct $m_0, m_1 \in \{0,1\}^\ell$ and all random coins $r_0, r_1 \in \{0,1\}^*$,
$$\mathsf{Com}(m_0; r_0) \neq \mathsf{Com}(m_1; r_1).$$
- Computational hiding: For all messages $m_0, m_1 \in \{0,1\}^\ell$, over the random coins of Com,
$$\mathsf{Com}(m_0) \stackrel{c}{\approx} \mathsf{Com}(m_1).$$

Remark 2.4. A commitment scheme is much like a public-key encryption scheme, in that it computationally hides the committed message, but commitment is potentially weaker. The reason is that there need not be any decryption algorithm; instead, the message and randomness are revealed by Alice at a later point.

It is simple to construct a commitment scheme for a single bit using a one-way permutation f with a hard-core predicate h. Define $\mathsf{Com}(b \in \{0,1\}, r \in \{0,1\}^n) = (f(r), b \oplus h(r))$. The scheme is clearly binding: given any commitment $(y \in \{0,1\}^n, c \in \{0,1\})$, there is a unique $r = f^{-1}(y)$ and hence the committed bit is $b = c \oplus h(r)$. That the above scheme is also hiding follows by noticing

that $(f(r), h(r)) \stackrel{c}{\approx} (f(r), U_1) \stackrel{c}{\approx} (f(r), 1 \oplus h(r))$. To extend the scheme to multiple-bit messages, just execute Com in parallel with independent randomness on each bit; hiding follows by a simple hybrid argument.

In fact, as we have seen in the homework, it is possible to construct an interactive (two-message) commitment scheme using a pseudorandom generator, which can be constructed (with a lot of work) from any one-way function.

3 Zero-Knowledge for NP

We can now prove Theorem 2.1. For the reasons given below, it will suffice to give a zero-knowledge proof (with efficient prover) for the graph 3-colorability problem $\mathsf{3COL}$, defined as follows.

Definition 3.1. We say that a graph $G = (V, E)$ is 3-colorable if there exists a function $\pi \colon V \to \{1, 2, 3\}$ (a "coloring" of the vertices) such that $\pi(i) \neq \pi(j)$ for every edge $(i, j) \in E$; that is, the endpoints of each edge have distinct colors. The language $\mathsf{3COL}$ is the set of graphs G that are 3-colorable.

It suffices to give a zero-knowledge proof for $\mathsf{3COL}$ because $\mathsf{3COL}$ is NP-complete. In more detail, to get a ZKP for any language $L \in \mathsf{NP}$, the prover and verifier (and also the simulator) on input x first compute $x' = R_L(x)$, where R_L is a deterministic poly-time reduction from L to $\mathsf{3COL}$; such a reduction is guaranteed to exist because $\mathsf{3COL}$ is NP-complete. (In fact, the Cook-Levin reduction, which works for any $L \in \mathsf{NP}$, is a "canonical" reduction that can be used.) Then the prover and verifier run the ZKP for $\mathsf{3COL}$ on $x' = R_L(x)$; because $R_L(x) \in \mathsf{3COL}$ if and only if $x \in L$, this yields a ZKP for L. Note that to get an efficient prover for L, we also need an efficient reduction that maps any witness for $x \in L$ to a witness for $R_L(x) \in \mathsf{3COL}$; fortunately, the Cook-Levin reduction provides this as well.

3.1 Interactive Proof for 3COL

Let $G = ([n], E)$ be a graph on n vertices, which are numbered $1, \ldots, n$ without loss of generality.

Consider the setup again. P wants to convince V that there is a valid 3-coloring $\pi \colon [n] \to \{1, 2, 3\}$ of the graph G. The prover and verifier are both given the graph G, and the prover is additionally given π. The basic idea is that the prover commits to a random permutation of the coloring π, then the verifier challenges the prover to reveal the colors of the two vertices on a single edge. If the prover correctly reveals two different colors, then the verifier accepts. Intuitively, the protocol is sound because if the graph is not 3-colorable, the verifier has at least a $1/|E|$ chance of choosing an improperly colored edge. Again intuitively, the protocol is zero-knowledge because the verifier only sees the commitments to each vertex (which hide the colors), plus openings to two distinct random colors. The formal protocol follows.

$P(G, \pi)$		$V(G)$
Choose random perm ρ of $\{1,2,3\}$. For all $i \in [n]$, let $k_i = \rho(\pi(i))$ and let $c_i = \mathsf{Com}(k_i; r_i)$.		
	$\xrightarrow{c_1, \ldots, c_n}$	
		Choose $(u, v) \leftarrow E$
	$\xleftarrow{(u,v)}$	
If $(u, v) \in E$,		
	$\xrightarrow{(k_u, r_u), (k_v, r_v)}$	
		Accept iff $k_u, k_v \in \{1,2,3\}$ are distinct and $c_i = \mathsf{Com}(k_i, r_i)$ for $i = u, v$.

The following claim is immediate by inspection:

Claim 3.2. The interactive proof system (P, V) described above is complete.

Claim 3.3. The proof system (P, V) described above has soundness error at most $1 - 1/|E| \leq 1 - 1/n^2$.

Proof. Suppose that G is not 3-colorable, and let P^* be any possibly unbounded prover. By statistical binding of Com, for any value of a commitment c_i sent by P^*, there is at most one color $k_i \in \{1,2,3\}$ for which P^* could acceptably open the commitment c_i. Because G is not 3-colorable, for any values of the colors $k_i \in \{1,2,3\}$, there must be at least one edge $(u, v) \in E$ for which $k_u = k_v$. The verifier chooses this edge with probability $1/|E|$, and no message from P^* could cause it to accept in this case. □

Claim 3.4. The proof system (P, V) described above is computational zero-knowledge.

Proof. Let $G \in \mathsf{3COL}$, and let π be an arbitrary valid 3-coloring of G. Let V^* be an arbitrary nuppt cheating verifier. We need to show that there exists an nuppt simulator $\mathcal{S}$ such that

$$\text{view}_{V^*}[P(G, \pi) \leftrightarrow V^*(G)] \overset{c}{\approx} \mathcal{S}(G).$$

As usual, we will give a "black-box" simulator $\mathcal{S}$ that uses V^* only as an oracle. Informally, the simulator prepares an initial message to V^* (i.e., the commitments c_i) so that it knows how to correctly answer one of the possible challenge edges. If V^* happens to ask for that edge, the simulator completes the transcript with the correct reply, otherwise it rewinds and tries again until it succeeds (up to a certain number of attempts). More formally, the simulator $\mathcal{S}^{V^*}(G)$ is defined as follows:

1: Choose $(u, v) \leftarrow E$ uniformly at random.
2: Choose distinct uniformly random $k_u, k_v \in \{1,2,3\}$. For all other $i \in [n]$, set $k_i = 1$.
3: For each $i \in [n]$, compute commitments $c_i = \mathsf{Com}(k_i; r_i)$.
4: Run V^* with fresh random coins r_{V^*}, send the c_is to V^*, and receive a challenge edge $(u^*, v^*) \in E$.
   (If V^* does not reply with a valid edge, then just output the view so far.)
5: if $(u^*, v^*) = (u, v)$ then

6: Output the view $(G, r_{V^*}, \{c_i\}, (k_u, r_u), (k_v, r_v))$.
7: else
8: Restart from the beginning (up to n^3 total iterations).

To prove that $\mathcal{S}$ is a valid simulator, i.e., that $\text{view}_{V^*}[P(G,\pi) \leftrightarrow V^*(G)] \overset{c}{\approx} \mathcal{S}(G)$, we will consider a sequence of hybrid experiments that successively change from the view in a true interaction with P to the view output by $\mathcal{S}$. The hybrid experiments are defined as follows, along with the reasons why successive hybrids are indistinguishable, which will complete the proof.

- H_0 is the view of V^* in an interaction with the real prover $P(G, \pi)$.
- H_1 is the view of V^* in an interaction with an algorithm $P'(G, \pi)$. The algorithm P' works almost identically to P, but it first chooses an edge $(u, v) \leftarrow E$ before preparing the commitments c_i as usual. It then replies to V^*'s challenge (u^*, v^*) only if $(u^*, v^*) = (u, v)$; otherwise, the experiment is re-run from the beginning (with fresh random coins), up to n^3 iterations. In other words, $P'(G, \pi)$ generates messages exactly as $P(G, \pi)$ does, but has the rewinding strategy of $\mathcal{S}$.

  Observe that conditioned on the event $(u^*, v^*) = (u, v)$, the view of V^* when interacting with P' is identical to its view when interacting with P. And because the choice of (u, v) by P' is statistically independent of its initial message to V^*, the probability that $(u^*, v^*) = (u, v)$ is exactly $1/|E| \geq 1/n^2$, and some iteration succeeds, except with negligible probability. We conclude that $H_0 \overset{s}{\approx} H_1$.
- H_2 is the view of V^* in an interaction with an algorithm $P''(G, \pi)$. The algorithm P'' works almost identically to P': it chooses $(u, v) \leftarrow E$ and sets the colors $k_u = \rho(\pi(u))$, $k_v = \rho(\pi(v))$ as before, but sets all other colors $k_i = 1$ (ignoring the valid coloring π). It then constructs the commitments c_i using the colors k_i, and either replies to V^* or rewinds in exactly the same way as before. In other words, $P''(G, \pi)$ still uses the valid coloring π for vertices u and v, but uses $\mathcal{S}$'s coloring strategy for the others.

  Using the computational hiding property of the commitment scheme, one can show that $H_1 \overset{c}{\approx} H_2$. To do this formally, we would actually consider a sequence of "sub-hybrids" in which one color k_i at a time is changed from its "true" value $\rho(\pi(i))$ (as in H_1) to its "simulated" value 1 (as in H_2). We remark that the reduction from breaking the commitment scheme to distinguishing adjacent sub-hybrids uses non-uniformity in an essential way, because to simulate the entire view of V^* (given a commitment to either $\rho(\pi(i))$ or 1), the simulator needs to have G and π "hard-coded" into it. (This is a rather technical point, but is an important one for a rigorous proof.)
- H_3 is the view of V^* in an interaction with $\mathcal{S}(G)$. It is easy to see that H_2 and H_3 are identical: for any valid 3-coloring π, over the random choice of ρ by P'', the values $k_u = \rho(\pi(u))$ and $k_v = \rho(\pi(v))$ are distinct random colors, independent of everything else (just as in the definition of $\mathcal{S}$). □

4 Applications of ZK

We conclude by mentioning a few fascinating applications of zero knowledge proofs for NP.

Deniability / Non-transferability. Suppose that a CIA agent Alice wants to prove her credentials to an MI5 agent Bob. One way to do this is for the CIA to give her a signature, signed under the CIA's public key, for the message "Alice is a CIA agent." Alice can then give this signature to Bob. However, what if Bob goes rogue and wants to sell his list of known CIA agents (with proof, in the form of valid signatures!) to the highest bidder? The problem is that signature identifying Alice as a CIA agent is transferable. Instead, Alice should prove to Bob using zero-knowledge that she has a signature for the message "Alice is a CIA agent." This is an NP-statement, since the signature is a valid witness (which Alice has in her possession). Bob will believe the proof, but he will not be able to convincingly transfer the transcript of that proof to anybody else, because for all they know, Bob could have created the transcript on his own by running the simulator! In other words, Alice's proof is deniable, in that she can plausibly claim that she was not responsible for producing it.

Enforcing honest behavior. Here the idea is that players in an interactive protocol can prove that they are not "cheating," i.e., that they are not deviating from prescribed (and known) programs that they are supposed to be running (on their own secret inputs).

Roughly, the idea is that at the outset of the protocol, the players are required to commit to the secret inputs and random coins of the "honest" (prescribed) algorithms that they are supposed to run. They then carry out their computations, and with each output message they prove to each other in zero-knowledge that the message was honestly computed under the committed inputs and coins. By soundness, we know that the players must really act honestly in order to be able to provide a valid proof. By zero-knowledge, the proofs do not compromise the privacy of their secret inputs.