CS 6501 Week 7: Zero-Knowledge Proof Systems

Focus thus far in the course: protecting communication (e.g., message confidentiality and message integrity)

Next few weeks: protecting computations

Zero-knowledge: a defining idea at the heart of theoretical cryptography
- ↳ Idea will seem very counter-intuitive, but surprisingly powerful
- ↳ Showcases the importance and power of definitions (e.g., "What does it mean to know something?")

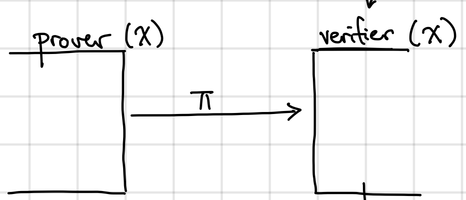
We begin by introducing the notion of a "proof system"
- Goal: A prover wants to convince a verifier that some statement is true

e.g., "This Sudoku puzzle has a unique solution"
"The number N is a product of two prime numbers p and q"
"I know the discrete log of h base g"

these are all examples of statements

We model this as follows:

the verifier is assumed to be an efficient algorithm



x: statement that the prover is trying to prove (known to both prover and verifier)
↳ We will write $\mathcal{L}$ to denote the set of true statements (called a "language")

π: the proof of x

↳ $b \in \{0,1\}$ – given statement x and proof π, verifier decides whether to accept or reject

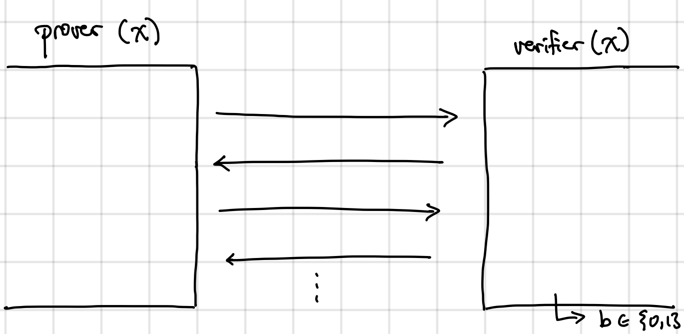
Properties we care about:
- Completeness: Honest prover should be able to convince honest verifier of true statements
$$\forall x \in \mathcal{L}: \Pr[\pi \leftarrow P(x) : V(x,\pi) = 1] = 1$$
- Soundness: Dishonest prover cannot convince honest verifier of false statement
$$\forall x \notin \mathcal{L}: \Pr[\pi \leftarrow P(x) : V(x,\pi) = 1] < \frac{2}{3}$$

Important: We are not restricting to efficient provers

Typically, proofs are "one-shot" (i.e., single message from prover to verifier) and the verifier's decision algorithm is deterministic
↳ Languages with these types of proof systems precisely coincide with NP (proof of statement x is to send NP witness w)

Going beyond NP: we augment the model as follows
- Add randomness: the verifier can be a randomized algorithm
- Add interaction: verifier can ask "questions" to the prover

Interactive proof systems [Goldwasser-Micali-Rackoff]:



Set of languages that have an interactive proof system is denoted IP.

Theorem (Shamir): IP = PSPACE

languages that can be decided in polynomial space [very large class of languages!]

Takeaway: interaction and randomness is very useful
↳ In fact, enables a new property called zero-knowledge

Consider following example: Suppose prover wants to convince verifier that $N = pq$ where p, q are prime (and secret).

prover (N, p, q) verifier (N)

$$\xrightarrow{\pi = (p, q)}$$

↓

accept if $N = pq$ and reject otherwise

Proof is certainly complete and sound, but now verifier also learned the factorization of N... (may not be desirable if prover was trying to convince verifier that N is a proper RSA modulus (for a cryptographic scheme) without revealing factorization in the process

↳ In some sense, this proof conveys information to the verifier [i.e., verifier learns something it did not know before seeing the proof]

Zero-knowledge: ensure that verifier does not learn anything (other than the fact that the statement is true)

How do we define "zero-knowledge"? We will introduce a notion of a "simulator."

Definition. An interactive proof system $\langle P, V \rangle$ for a language $\mathcal{L}$ is zero-knowledge if for all efficient (and possibly malicious) verifiers V^*, there exists an efficient simulator S such that for all $x \in \mathcal{L}$:

$$\underbrace{\text{View}_{V^*}(\langle P, V \rangle(x))}_{\substack{\text{random variable denoting the set of messages} \\ \text{sent and received by } V^* \text{ when interacting with the prover } P \text{ on input } x}} \overset{c}{\approx} S(x)$$

What does this definition mean?

$\text{View}_{V^*}(P \leftrightarrow V^*(x))$: this is what V^* sees in the interactive proof protocol with P

$S(x)$: this is a function that only depends on the statement x, which V^* already has

If these two distributions are indistinguishable, then anything that V^* could have learned by talking to P, it could have learned just by invoking the simulator itself, and the simulator output only depends on x, which V^* already knows

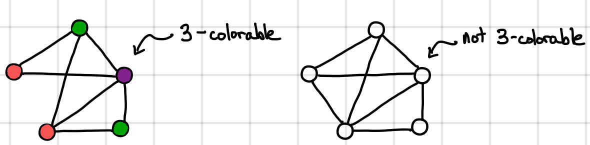
↳ In other words, anything V^* could have learned (i.e., computed) after interacting with P, it could have learned without ever talking to P!

Very remarkable definition!

More remarkable: If one-way functions exist, then every language $\mathcal{L} \in IP$ has a zero-knowledge proof system.

↳ Namely, anything that can be proved can be proved in zero-knowledge!

We will show this theorem for NP languages. Here it suffices to construct a single zero-knowledge proof system for an NP-complete language. We will consider the language of graph 3-colorability.



3-coloring: given a graph G, can you color the vertices so that no adjacent nodes have the same color?

cryptographic analog of a sealed "envelope"

We will need a commitment scheme (see HW2). A (non-interactive) commitment scheme consists of two main algorithms (Commit, Verify)

- $\text{Commit}(1^\lambda, m) \to (c, r)$: Takes a message m and outputs the commitment c and an opening r
- $\text{Verify}(m, c, r) \to b$: Checks if c is a valid opening to m (with respect to opening r)

[The commitment scheme might also take public parameters (see HW2), but for simplicity, we omit them / leave them implicit]

Requirements:

- Correctness: for all messages m:

$$\Pr[(c,r) \leftarrow \text{Commit}(1^\lambda, m) : \text{Verify}(m, c, r) = 1] = 1$$

- Hiding: for all efficient adversaries A, if $(m_0, m_1) \leftarrow A(1^\lambda)$

$$\{(c,r) \leftarrow \text{Commit}(1^\lambda, m_0) : c\} \overset{c}{\approx} \{(c,r) \leftarrow \text{Commit}(1^\lambda, m_1) : c\}$$

- Binding: for all efficient adversaries A, if

$$\Pr[(m_0, m_1, c, r_0, r_1) \leftarrow A(1^\lambda) : m_0 \neq m_1 \text{ and } \text{Verify}(m_0, c, r_0) = 1 = \text{Verify}(m_1, c, r_1)] = \text{negl}(\lambda)$$

↳ We will require perfect binding [for every commitment c, there is only 1 possible m to which the prover can open c]

A ZK protocol for graph 3-coloring:

prover (G) ← contains n nodes, m edges

verifier (G)

Prover:
- let $K_i \in \{0,1,2\}$ be a 3-coloring of G
- choose random permutation $\pi \xleftarrow{R} \text{Perm}[\{0,1,2\}]$
- for $i \in [n]$: $(c_i, r_i) \leftarrow \text{Commit}(1^\lambda, K_i)$

Prover → Verifier: $c_1, \dots, c_n$

Verifier: $(i, j) \xleftarrow{R} E$

Verifier → Prover: i, j

Prover → Verifier: $(K_i, r_i), (K_j, r_j)$

↳ accept if $K_i \neq K_j$ and $K_i, K_j \in \{0,1,2\}$

$\text{Verify}(K_i, c_i, r_i) = 1 = \text{Verify}(K_j, c_j, r_j)$

reject otherwise

Intuitively: Prover commits to a coloring of the graph

Verifier challenges prover to reveal coloring of a single edge

Prover reveals the coloring on the chosen edge and opens the entries in the commitment

Completeness: By inspection [if coloring is valid, prover can always answer the challenge correctly]

Soundness: Suppose G is not 3-colorable. Let $K_1, \dots, K_n$ be the coloring the prover committed to. If the commitment scheme is perfectly binding, $c_1, \dots, c_n$ uniquely determine $K_1, \dots, K_n$. Since G is not 3-colorable, there is an edge $(i,j) \in E$ where $K_i = K_j$ or $i \notin \{0,1,2\}$ or $j \notin \{0,1,2\}$. [Otherwise, G is 3-colorable with coloring $K_1, \dots, K_n$.] Since the verifier chooses an edge to check at random, the verifier will choose (i,j) with probability $1/|E|$. Thus, if G is not 3-colorable,

$$\Pr[\text{verifier rejects}] \geq \frac{1}{|E|}$$

Thus, this protocol provides soundness $1 - \frac{1}{|E|}$. We can repeat this protocol $O(|E|^2)$ times sequentially to reduce soundness error to

$$\Pr[\text{verifier accepts proof of false statement}] \leq \left(1 - \frac{1}{|E|}\right)^{|E|^2} \leq e^{-|E|} = e^{-m} \quad \left[\text{since } \left(1 - \frac{1}{x}\right)^x \leq \frac{1}{e}\right]$$

<u>Zero Knowledge</u>: We need to construct a simulator that outputs a valid transcript given only the graph G as input.
Let V^* be a (possibly malicious) verifier. Construct simulator S as follows:

1. Choose $k_i \leftarrow \{0,1,2\}$ for all $i \in [n]$.
   Let $(c_i, r_i) \leftarrow \mathrm{Commit}(1^\lambda, k_i)$.
   Give $(c_1, \ldots, c_n)$ to V^*.

   Simulator does <u>not</u> know coloring so it commits to a random one

2. V^* outputs an edge $(i,j) \in E$
3. If $k_i \neq k_j$, then S outputs (k_i, k_j, r_i, r_j).
   Otherwise, restart and try again (if fails λ times, then abort)

Simulator succeeds with probability $2/3$ (over choice of $k_1, \ldots, k_n$). Thus, simulator produces a valid transcript with prob. $1 - \frac{1}{3^\lambda} = 1 - \mathrm{negl}(\lambda)$ after λ attempts. It suffices to show that simulated transcript is indistinguishable from a real transcript.

- Real scheme: prover opens k_i, k_j where $k_i, k_j \xleftarrow{R} \{0,1,2\}$ [since prover randomly permutes the colors]
- Simulation: k_i and k_j sampled uniformly from $\{0,1,2\}$ and conditioned on $k_i \neq k_j$, distributions are identical

In addition, (i,j) output by V^* in the simulation is distributed correctly since commitment scheme is computationally-hiding (e.g. V^* behaves essentially the same given commitments to a random coloring as it does given commitment to a valid coloring

If we repeat this protocol (for soundness amplification), simulator simulate one transcript at a time