CS 355 – Topics in Cryptography April 9, 2018

Lecture 3: Interactive Proofs and Zero-Knowledge

Instructors: Henry Corrigan-Gibbs, Sam Kim, David J. Wu

So far in the class, we have only covered basic cryptographic primitives like PRGs, PRFs, Digital Signatures, and so on. These primitives consist of non-interactive "one-shot" algorithms that satisfy some specific security properties. In the real world, these primitives are used as "tools" that are part of bigger and more complicated *interactive protocols*. In the next few lectures, we are going to discuss how to analyze these type of interactive protocols.

What does it mean for an interactive protocol to be secure? How do we prove security?

As a first step towards this goal, we are going to discuss *interactive proof systems* and *zero-knowledge* in this lecture. This will be our first introduction to a *simulation*-based notion of security, which is a very important concept in cryptography.

1 Interactive Proofs

Informally, the goal of a proof is to convince someone that a certain statement is true. A statement can consist of expressions such as "N is the product of two 1024-bit primes" or "(g, h) is such that $h = g^a$ and a is odd". It is not hard to see how the ability to prove such statements can be useful for cryptographic applications.

Languages. To treat proof systems in a rigorous manner, let's first formalize what a statement is. Following the complexity theoretic conventions, we will treat statements with respect to an instance of a language L. Recall that a *language* is simply a set of strings $L \subseteq \{0,1\}^*$. Then, a statement consists of the tuple (x, L) or more intuitively

$$\text{"}x \in L\text{"}.$$

Examples:

- "N is the product of two 1024-bit primes"

$$N \in \{pq \mid p, q \text{ are 1024-bit primes}\}.$$

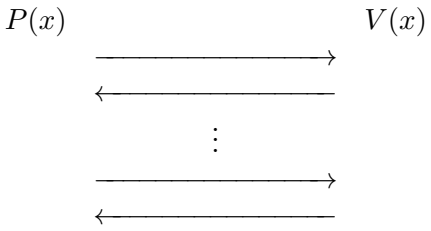
- "(g, h) is such that $h = g^a$ and a is odd"

$$(g, h) \in \{(g, g^a) \mid a \in \mathbb{Z}_q \text{ and } a \text{ is odd}\}.$$

- "Boolean formula φ does not have a satisfying assignment"

$$\varphi \in \mathsf{coSAT}.$$

Intuitive notion of proofs. An interactive proof system for a language L is a protocol between two algorithms: a (possibly unbounded) prover P and an efficient (probabilistic polynomial time) verifier V.



At the start of the protocol, both the prover P and the verify V are given some instance x. At the end of the protocol, the verifier V either accepts (it is convinced that $x \in L$) or rejects (it is not convinced that $x \in L$).

For a proof system to be useful, it must satisfy the following two properties:

1. **Completeness**: If $x \in L$, then an honest prover P that just follows protocol specification should be able to convince V.
2. **Soundness**: If $x \notin L$, then no prover P (that can possibly cheat by deviating from the protocol specification) should not be able to convince V.

NP. What types of languages have an interactive proof system? The simplest example is the class of languages in NP. Let $L \in \mathsf{NP}$. Then, by definition, there exists an efficient algorithm $M(\cdot, \cdot)$ such that

$$x \in L \iff \exists\ w \in \{0,1\}^{\mathsf{poly}(|x|)} \text{ s.t. } M(x, w) = 1.$$

Therefore, we can specify the protocol as follows:

- $P(x)$: Compute a witness $w \in \{0,1\}^{\mathsf{poly}(|x|)}$ and send it to V.
- $V(x)$: When P sends over w, check if $M(x, w) = 1$. Output "accept" if this is the case and "reject" otherwise.

Why interaction? So when the class of NP languages already have "one-shot" non-interactive proofs protocols, why do we even consider interactive proofs?

1. Every language $L \in \mathsf{NP}$ have non-interactive proofs, but sometimes, we might want to prove a statement that is not necessarily in NP. For example, consider the following statement in coNP: "Boolean formula φ does not have a satisfying assignment." It seems hard to prove such a statement without some sort of interaction.

   Fact: In computational complexity, the celebrated result of [3] shows that the class of languages that have interactive proofs (with polynomial number of rounds) is precisely the class of PSPACE.

   $$\mathsf{IP} = \mathsf{PSPACE}.$$

   We will not prove this fact in the course, but anyone curious can take a look at any textbook on complexity theory (e.g. [1]).

2. Even if the statement to be proved is in NP, for certain certain applications, requiring P to send over the whole witness w to V might be too costly. By allowing additional rounds of interaction between P and V, we can sometimes bring down the total communication between P and V smaller than $|w|$.

   Fact: This point is related to another celebrated result in complexity theory called the *PCP Theorem*. We will come back to this later in the course when we discuss SNARGs.

3. Interactive proofs give rise to proof systems that satisfy the additional property of *zero-knowledge*. This is what we are going to discuss next.

Defining interactive proofs (semi-)formally. Interactive proofs were defined in 1985 by Goldwasser, Micali, and Rackoff [2].

Definition 1.1 (Interactive Proofs)**.** Let L be any language. Let $\langle P, V\rangle$ be a protocol specification between a *prover* P and the verifier V. Then, we say that $\big(V, \langle P, V\rangle\big)$ is an *interactive proof system* for L if the following two properties are satisfied:

- **Completeness**: $\forall x \in L$,
$$\Pr[\langle P, V\rangle(x) = 1] \geq \frac{2}{3}.$$
- **Soundness**: $\forall x \notin L$, $\forall P^*$
$$\Pr[\langle P^*(x), V(x)\rangle = 1] \leq \frac{1}{3}.$$

We note that the constants $2/3$ and $1/3$ are arbitrarily chosen for simplicity. We can always amplify the completeness probability to $1 - \mathsf{negl}(\lambda)$ and the soundness probability to $\mathsf{negl}(\lambda)$ with repetition.

2 Zero-Knowledge

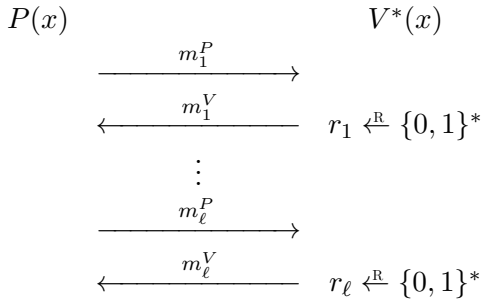
How can we prove a statement $x \in L$ without revealing anything else about x other than the fact that $x \in L$? For instance, how can we prove the statement "N is the product of two 1024-bit primes" without revealing the factorization p, q?

In this class, we will just restrict ourselves to the set of NP problems.[1] In this setting, for a language $L \in \mathsf{NP}$, we have two parties:

- An *honest* (unbounded) prover P with input (x, w) such that w is an NP witness of x. It follows the protocol specification exactly.
- A *dishonest* (PPT) verifier V^* with input x. It can deviate from the protocol specification.

The goal of the verifier V^* is to infer some information about x from its interaction with P.

[1] Zero-knowledge proofs are well-defined for any class of languages. In fact, any language in IP has a ZK proof system assuming OWFs.



In other words, the verifier is given (i) all the transcript $(m_1^P, \ldots, m_\ell^P)$, $(m_1^V, \ldots, m_\ell^V)$, and (ii) the internal coins (randomness) it used $r_1, \ldots, r_\ell$ throughout the protocol. Then, it tries to learn additional information about x. Generally, since the V^*'s messages $m_1^V, \ldots, m_\ell^V$ are completely determined by P's messages and V^*'s random coins, we generally omit it for redundancy when describing the transcript. Then, we define the *view* of V^* as the random variable

$$\mathsf{view}_{P,V^*(x)} = \{m_1^P, \ldots, m_\ell^P, r_1, \ldots, r_\ell\}.$$

Then, our goal is to require that no adversary can gain any additional "knowledge" about x from $\mathsf{view}_{P,V^*(x)}$. How do we formalize this?

What is knowledge? To formalize what it means for an adversary (verifier) to not learn any new knowledge about x, we must first reason about what "knowledge" actually is.

- Say you have $N = pq$, and also one of the factors p. Does this mean that you also have "knowledge" of q? Intuitively yes because even though you don't have the actual encoding of "p", you can always efficiently compute $q = N/p$.
- Say you have an encryption of a message $\mathsf{Enc}_{\mathsf{pk}}(x)$. Does this mean that you have "knowledge" x? Intuitively no because even though you have some encoding of "x" (that information theoretically determines x), you cannot efficiently recover x.

Therefore, in cryptography, "knowledge" is defined with respect to things that you can compute efficiently.

Defining Zero-Knowledge. Therefore, how do we define zero-knowledge? We say that a protocol is zero-knowledge if any information that an adversary could have derived from the transcript of the protocol, *could have* originally been computed efficiently just from x (without any transcript).

Definition 2.1 (Zero-Knowledge Proofs)**.** Let $L \in \mathsf{NP}$. Let $\langle P, V, \rangle$ be a protocol specification between a (possibly unbounded) prover P and a (PPT) verifier V. Then, we say that $\langle P, V \rangle$ is an *interactive proof system* for L if the following properties are satisfied:

- **Completeness**: $\forall\ x \in L$,
$$\Pr[\langle P(x,w), V(x)\rangle = 1] \geq \frac{2}{3}.$$
- **Soundness**: $\forall\ x \notin L$, $\forall\ P^*$
$$\Pr[\langle P^*(x,w), V(x)\rangle = 1] \leq \frac{1}{3}.$$

- **(computational) Zero-Knowledge**: $\forall\ V^*$, $\exists$ (PPT) Sim_{V^*} such that $\forall\ x \in L$,

$$\mathsf{View}[\langle P(x,w) \leftrightarrow V^*(x)\rangle] \approx_c \mathsf{Sim}_{V^*}(x).$$

  We call the algorithm Sim as the *simulator*.

At first, it might be useful to think about the simulator Sim_{V^*} as a *subroutine*. Then, we can parse the definition as follows. If there exists an adversary $\mathcal{A}$ that can compute any new information about x from the transcript $\mathsf{view}[\langle P(x,w) \leftrightarrow V^*(x)\rangle]$, then $\mathcal{A}$ could have just computed that information on its own (without interacting in the protocol) by calling the simulator $S(x)$ as a subroutine, get a "simulated" transcript, and then computing the same information about x.

3 ZK-Proofs for NP

Now that we have defined zero-knowledge proof systems, how do we construct them? In this lecture, we are going to construct a zero-knowledge proof system for the problem of 3-Coloring. Since 3-Coloring is NP-complete, this gives a zero-knowledge proof system for all of NP.

A graph $G = (V, E) \in$ 3-Coloring if there exists a way to assign each vertex $v \in V$ with a color $\{0, 1, 2\}$ such that no two vertices associated with an edge $(u, v) \in E$ are assigned the same color.

Formally, a graph $G = (V, E) \in$ 3-Coloring if

$$\exists\ \varphi : V \to \{0,1,2\} \text{ such that } \forall (u,v) \in E, \text{ we have } \varphi(u) \neq \varphi(v).$$

Here, the NP witness for G is the assignment φ.

3.1 Preliminaries: Commitments

For our ZK proof system for 3-Coloring, we rely on a primitive called *commitment* schemes. For simplicity, we will just use non-interactive commitments.

Definition 3.1 (Commitments)**.** An efficiently computable function $\mathsf{Comm} : \mathcal{M} \times \mathcal{R} \to \mathcal{C}$ is a (perfectly) binding commitment if it satisfies the following two properties:

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

$$\left\{\mathsf{Comm}(m_0, r) : r \xleftarrow{R} \mathcal{R}\right\} \approx_c \left\{\mathsf{Comm}(m_1, r) : r \xleftarrow{R} \mathcal{R}\right\}.$$

- **Binding**: For all $m_0, m_1 \in \mathcal{M}$, $r_0, r_1 \in \mathcal{R}$, if $m_0 \neq m_1$, then

$$\mathsf{Comm}(m_0, r_0) \neq \mathsf{Comm}(m_1, r_1).$$

We can think of a commitment as an envelope that you can't see inside, but binds you to a value. Commitment schemes can be built from a variety of assumptions OWFs, DLog, RSA, etc.

3.2 3-Coloring

Theorem 3.2. *If perfectly binding commitments exist, then there exists a zero-knowledge proof system for* 3-Coloring.

We specify our protocol as follows:

- **Start of the protocol**: At the start of the protocol, the prover P is provided G and an assignment φ and the verifier V is just provided G. Before the protocol, V initializes a counter rounds and set it $\mathsf{rounds} = 0$.
- **Step 1**: P samples a random permutation $\pi : \{0,1,2\} \to \{0,1,2\}$. For each $v \in V$, it computes a commitment
$$c_v \leftarrow \mathsf{Comm}(\pi(\varphi(v)), r_v).$$
It sends over the set of commitments $\{c_v\}_{v \in V}$ to the verifier V.
- **Step 2**: Upon receiving the commitments $\{c_v\}_{v \in V}$, V samples a random edge $(u,v) \xleftarrow{R} E$, and sends (u,v) to the prover P.
- **Step 3**: Upon receiving (u,v), P sends over the tuple $(\pi(\varphi(u)), r_u, \pi(\varphi(v)), r_v)$ to the verifier V.
- **End of Round**: Upon receiving $(\pi(\varphi(u)), r_u, \pi(\varphi(v)), r_v)$, V checks that
  - $\pi(\varphi(u)), \pi(\varphi(v)) \in \{0,1,2\}$
  - $\pi(\varphi(u)) \neq \pi(\varphi(v))$
  - $c_u = \mathsf{Comm}(\pi(\varphi(u)), r_u)$
  - $c_v = \mathsf{Comm}(\pi(\varphi(v)), r_v)$

  If any of these conditions are not satisfied, then V immediately outputs reject. Otherwise, it sets $\mathsf{rounds} = \mathsf{rounds} + 1$. If $\mathsf{round} = k$, then it outputs accept. Otherwise, it goes back to Step 1.

Completeness. Let $G \in$ 3-Coloring and let φ be a a valid witness for G. Then, for any $(u,v) \in E$ and any permutation π, we have $\pi(\varphi(u)) \neq \pi(\varphi(v))$ and $\pi(\varphi(u)), \pi(\varphi(v)) \in \{0,1,2\}$ by definition. Furthermore, since Comm is a deterministic function, the tuple $(\pi(\varphi(u)), r_u, \pi(\varphi(v)), r_v)$ that P sends to V at Step 3 must satisfy $c_u = \mathsf{Comm}(\pi(\varphi(u)), r_u)$ and $c_v = \mathsf{Comm}(\pi(\varphi(v)), r_v)$. Hence, the verifier V always accepts.

Soundness. If $G \notin$ 3-Coloring, then by definition, there does not exist a valid three coloring. Hence, for any assignment $\tilde{\varphi} = \pi \circ \varphi : V \to \{0,1,2\}$ that P^* uses in Step 1, there exists an edge $(u^*, v^*) \in E$ such that $\tilde{\varphi}(u^*) = \tilde{\varphi}(v^*)$. Assume that in Step 2, V chooses (u,v) such that $u = u^*$ and $v = v^*$. Let $(\mathsf{clr}_u, r_u, \mathsf{clr}_v, r_v)$ be the tuple that P^* sends to V in Step 3.

- If $\mathsf{clr}_u = \tilde{\varphi}(u^*)$ and $\mathsf{clr}_v = \tilde{\varphi}(v^*)$, or if $\mathsf{clr}_u, \mathsf{clr}_v \notin \{0,1,2\}$, then V immediately rejects.
- If $\mathsf{clr}_u \neq \tilde{\varphi}(u^*)$, then $c_u \neq \mathsf{Comm}(\mathsf{clr}_u, r_u)$ since Comm is perfectly binding.
- Likewise, if $\mathsf{clr}_v \neq \tilde{\varphi}(v^*)$, then $c_v \neq \mathsf{Comm}(\mathsf{clr}_v, r_v)$ since Comm is perfectly binding.

Hence, V always rejects in this case.

Now, what is the probability that V chooses (u, v) such that $u = u^*$ and $v = v^*$?

$$\Pr[u = u^* \wedge v = v^*] \leq \frac{1}{n^2}$$

where $n = |G|$.

Therefore, after t rounds, what is the probability that V still accepts?

$$\Pr[P^* \text{ gets lucky for } t \text{ rounds}] \leq \left(1 - \frac{1}{n^2}\right)^t \leq \left(e^{-\frac{1}{n^2}}\right)^t$$

where we used the identity $(1 + x) \leq e^x$.

Therefore, after repeating the protocol for $t \approx n^3$ times, this probability becomes negligible $\leq e^{-n}$.

Zero-Knowledge. To prove that the protocol satisfies zero-knowledge, we must do two things:

1. Specify an efficient simulator Sim that generates a transcript for the protocol.
2. Prove that $\forall\ V^*$, $\exists$ (PPT) Sim_{V^*} such that $\forall\ x \in L$,

$$\mathsf{View}[\langle P(x, w) \leftrightarrow V^*(x) \rangle] \approx_c \mathsf{Sim}_{V^*}(x).$$

Therefore, let us first specify a simulator as follows:

$\mathsf{Sim}_{V^*}(G)$:

1. Initialize $\mathsf{count} = 0$.
2. Commit to a random coloring φ of G.
3. Invoke $(u, v) \leftarrow V^*(G, c_1, \ldots, c_n)$.
4. If $\tilde{\varphi}(u) \neq \tilde{\varphi}(v)$, then ignore the transcript.
5. Otherwise, if $\tilde{\varphi}(u) = \tilde{\varphi}(v)$, then keep the transcript, and set $\mathsf{count} = \mathsf{count} + 1$.
6. If $\mathsf{count} = t$, then output all the transcript it kept. Otherwise, go to Step 2 and repeat.

Efficiency of the simulator. First, is Sim_{V^*} efficient? Let $(u, v) \in E$ be the edge that V^* submits in Step 2 of the protocol. Since Sim originally commits to a random coloring $\tilde{\varphi}$, we have that $\tilde{\varphi}(u) = \tilde{\varphi}(v)$ with probability $1/3$. Hence, Sim_{V^*} will have t set of transcript after approximately $3t$ repetitions.

Distribution of the simulator. We must now show that the set of transcripts output by $\mathsf{Sim}_{V^*}(G)$ is computationally indistinguishable from the set of transcripts from the real protocol. Let's list out all the elements that are contained in the transcript and show why each of these components are computationally indistinguishable.

- The commitments $\{c_v\}_{v \in V}$ that P sends in Step 1: In the real protocol, P commits to a valid coloring while $\mathsf{Sim}_{V^*}(G)$ commits to a random (possibly invalid) coloring. However, by the *Hiding* property of Comm, these two distribution of commitments are computationally indistinguishable.

- The coins or equivalently, the edge $(u, v) \in E$ that V^* chooses: Since Sim just runs V^*, the distribution of (u, v) in the real protocol and the distribution of (u, v) that is produced by Sim_{V^*} are identical.

- The tuple $(\mathsf{clr}_u, r_u, \mathsf{clr}_v, r_v)$ that P sends in Step 3: In the real protocol, P chooses a random permutation π. Hence, $\mathsf{clr}_u = \pi(\varphi(u))$ and $\mathsf{clr}_v = \pi(\varphi(v))$ are uniformly distributed in $\{0, 1, 2\}$ given that $\mathsf{clr}_u \neq \mathsf{clr}_v$. In the simulated distribution, Sim commits to a totally random coloring $\tilde{\varphi}$. Therefore, given that a transcript was not discarded, $\mathsf{clr}_u = \tilde{\varphi}(u)$ and $\mathsf{clr}_v = \varphi(v)$ are uniformly distributed given that $\varphi(u) \neq \varphi(v)$.

References

[1] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[2] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[3] Adi Shamir. Ip = pspace. *J. ACM*, 39(4):869–877, October 1992.