CS 355 – Topics in Cryptography April 8, 2019

Lecture 3: Interactive Proofs

Florian Tramèr

Most of these notes are from last year's lecture by Sam Kim. Thanks Sam!

So far in the class, we have mainly covered basic cryptographic primitives like OWFs, PRGs, PRFs, 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*. These were introduced in a series of papers in the 1980s, culminating in the seminal work of Goldwasser, Micali and Rackoff in 1985 [1] that also defined the notion of Zero-Knowledge Proofs. We'll cover these in the next lecture.

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". The ability to prove such statements is very useful in many 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"

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

One-shot proofs. Let's first look at "standard" proofs without any interaction. Think, for example, of a mathematical theorem written down in some lecture notes that aims to convince you—the reader—that the proof is correct.

We can view such proofs for a language L as a (single-round) protocol between two algorithms: a (possibly unbounded) prover P and an efficient verifier V.

$$P(x) \quad \xrightarrow{\qquad\pi\qquad} \quad V(x)$$

At the start of the protocol, both the prover P and the verifier V are given some statement x. The prover then sends the verifier a proof π, and 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 the protocol specification should be able to convince V.

2. **Soundness**: If $x \notin L$, then no prover algorithm P^* (that can arbitrarily cheat by deviating from the protocol specification) should be able to convince V.

NP. What types of languages have such a (non-interactive) proof system? It turns out that when the prover is deterministic, this is exactly 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.

Conversely, if a complete and sound proof system for L exists, wherein the prover sends a single message $\pi \in \{0,1\}^{\mathsf{poly}(|x|)}$, then for any $x \in L$ there must exist some π such that $V(x, \pi)$ accepts, so $x \in \mathsf{NP}$.

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

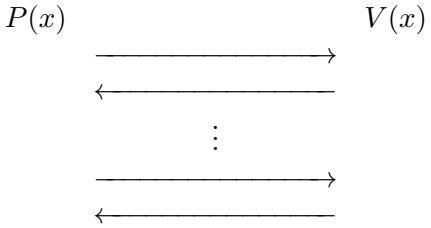
1. Sometimes, we might want to prove a statement that is not necessarily in NP. For example, consider the following statement in coNP: "Boolean formula φ does not have a satisfying assignment." It seems hard to prove such a statement without some sort of interaction.

2. Even if the statement to be proved is in NP, for 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 a celebrated result in complexity theory called the *PCP Theorem* and to a class of cryptographic proofs called 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 in the next lecture.

Defining interactive proofs (semi-)formally. Extending our previous definition, an interactive proof system for a language L is a (possibly multi-round) protocol between two algorithms: an unbounded prover P and an efficient (possibly randomized) verifier V.



At the end of the protocol with input x, the verifier V either accepts (i.e., it outputs 1 if it is convinced that $x \in L$) or rejects (i.e., it outputs 0 if it is not convinced that $x \in L$).

More formally, for a prover P, verifier V, and input x, let $\langle P, V \rangle(x)$ denote the output of a protocol run on input x (this is V's output, which is a single bit).

Definition 1.1 (Interactive Proofs)**.** Let L be any language. We say that $\big(P, V\big)$ 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**: $\forall x \notin L$, $\forall$algorithmsP^*
$$\Pr[\langle P^*(x), V(x) \rangle = 1] \leq \frac{1}{3}.$$

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.

A note on verifier randomness. You may have noticed that compared to the initial proof system which we showed to be equivalent to NP, the definition of interactive proofs adds not only interaction between P and V, but it also allowed the verifier V to be non-deterministic (and thus err with some probability).

Both of these generalizations are actually necessary to get the full power of interactive proof systems. Indeed:

- We showed that *non-interactive* proof systems with *deterministic* V exist exactly for those languages in NP.
- If we allow for *interaction* but V remains *deterministic*, we still only get proofs for NP (you will show this in your homework).
- If we let V be *randomized*, but the protocol remains *non-interactive*, we get proofs for languages in the so-called *Merlin-Arthur* (MA) class. Under plausible complexity assumptions, $\mathsf{MA} = \mathsf{NP}$ (although we do not know how to prove this), so we still do not seem to have gained much.

- When we allow for *interaction* and *non-deterministic* V, we get the full strength of interactive proofs. The languages which have such proofs are defined as the class IP. Shamir showed that IP = PSPACE, the class of languages that can be recognized with polynomial space and exponential time [2]! PSPACE contains many languages that we believe to be outside of NP (e.g., problems in coNP, #SAT, etc.)

References

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

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