CS276: Cryptography October 27, 2015

Proof Systems and Zero-Knowledge Proofs

Instructor: Alessandro Chiesa *Scribe: Pasin Manurangsi*

This lecture was given by Manuel Sabin.

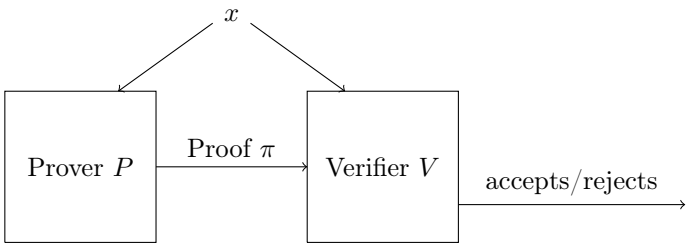
1 Introduction

Let us think about the following fundamental question: what is a proof? Or more specifically, what are the properties of a proof? Mathematicians and computer scientists have different answers to this question. From the latter perspective, below are a couple properties of a good proof.

- The proof should be efficiently verifiable, i.e., there is an efficient algorithm (verifier) that can check whether the proof is correct in polynomial time.
- The verifying process should be sound, i.e., the mentioned verifier should accept if and only if the proof is correct.

With the properties above in mind, we introduce the concept of *proof systems*. A proof system consists of a *prover* and a *verifier*. At the beginning, both of them are given an input x. Then, the prover needs to send a proof π to convince the verifier that x has some property. Informally, there are two properties required for proof systems in general:

- (Completeness) If x indeed has the desired property and an honest prover sends a correct proof, the verifier should accept the proof.
- (Soundness) If x does not have the desired property but a dishonest prover sends a fake proof, the verifier should reject the proof.



Formal definitions for proof systems will be given later. As we will see below, soundness is sometimes allowed to be imperfect, i.e., it is fine if a fake proof is rejected with high probability. Furthermore, the computational powers for the prover and the verifier differ for each class of proof systems.

Today we will study different proof systems with emphasis on the so-called *Zero-Knowledge Proof System*, a proof system in which, despite being able to verify that the proof is correct, the verifier cannot learn anything from the proof.

2 Proof Systems for NP Problems

The above definition of proof systems evokes the definition of NP. Recall that a language L is in NP if and only if, there exists a polynomial-time algorithm V such that

$$x \in L \Leftrightarrow \exists w, V(x, w) = 1.$$

We can simply view w as a proof that $x \in L$ and V as the verifying algorithm. We can then define the proof system as follows:

- The computationally unbounded prover P, when received x, searches for a witness w such that $V(x, w) = 1$. P then sends x as a proof to the verifier.
- The verifier V just runs $V(x, w)$ where w is the proof received from the prover. Note that the verifier runs in polynomial time.

It is clear that the completeness and (perfect) soundness hold here.

For concreteness, let us consider the following example of NP language.

Definition 1 (Graph Isomorphism) *The Graph Isomorphism language GI consists of all tuples of graphs (G_0, G_1) where $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$ are isomorphic undirected graphs, i.e., there exists a permutation $\sigma : V_0 \to V_1$ such that $(u, v) \in E_0$ if and only if $(\sigma(u), \sigma(v)) \in E_1$.*

For Graph Isomorphism, a witness is just the permutation σ and a proof system for it can be defined as follows:

- The prover enumerates to find the permutation σ and sends it to the verifier.
- The verifier checks whether the input σ satisfies $(u, v) \in E_0$ if and only if $(\sigma(u), \sigma(v)) \in E_1$.

3 Interactive Proof Systems

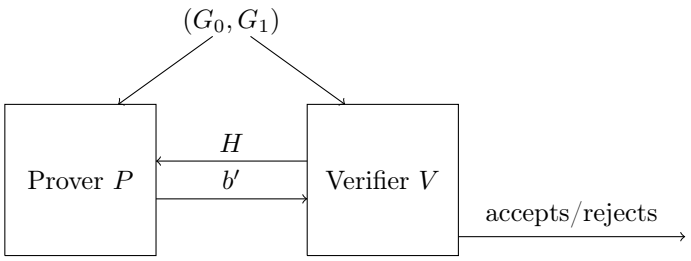
Unfortunately, many problems outside of NP are not known to have proof systems as simple as those in NP. We will use Graph Non-Isomorphism, the complement of GI, as our example here.

Definition 2 (Graph Non-Isomorphism) *The Graph Non-Isomorphism language GNI consists of all tuples of graphs (G, G^0) such that $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$ are non-isomorphic undirected graphs, i.e., there exists no permutation $\sigma : V_0 \to V_1$ such that $(u, v) \in E_0$ if and only if $(\sigma(u), \sigma(v)) \in E_1$.*

In this case, unlike Graph Isomorphism, it is unclear how a prover can convince the verifier that there is no such permutation. For instance, due to limited computational power of the verifier, the prover cannot send all the permutations and the permuted graphs according to each permutation to the verifier.

To tackle this, we introduce *interactive proof systems*. In an interactive proof system, the verifier and the prover can send multiple messages to each other and they can use previous messages to help determine their actions. It turns out that, in contrast to non-interactive proof systems, we can create an interactive proof system for GNI as follows.

- The verifier works as follows:
  - Randomly select a bit $b \leftarrow \{0,1\}$.
  - Randomly select a permutation $\sigma : V_b \to V_b$.
  - Permute G_b according to σ. Let us call the resulting graph H.
  - Query the prover with H to ask for b' such that H is isomorphic to $G_{b'}$.
  - Accepts if and only if $b' = b$.
- The prover works as follows:
  - When received a query H, checks (by enumerating all permutations) whether H is isomorphic to G_0. If so, answer 0. Otherwise, answer 1.



To see that this is a valid proof system, we need to check its soundness and completeness.

- (Completeness) Suppose $(G_0, G_1) \in GNI$. Since the two graphs are not isomorphic, H is only isomorphic to one of them. As a result, the honest prover can tell which graph H is isomorphic to and recover $b' = b$, which means that the verifier accepts.
- (Soundness) Suppose $(G_0, G_1) \notin GNI$. In this case, H is isomorphic to both graphs. Even if a dishonest prover wants to fake the proof, he has no way of knowing what b is. As a result, he can only guess b right half the time.

We note that, while the soundness is only $1/2$, the verifier can make multiple queries and reduce the soundness to be negligible in the size of the input.

We conclude this section by giving a formal definition for interactive proof systems.

Definition 3 (Interactive Proof System) *A language L has an interactive proof system if and only if there exists (P, V) where V is polynomial bounded such that*

- *(Completeness) $\forall x \in L, \Pr[output_V(P(x) \leftrightarrow V(x)) = 1] = 1$, and,*
- *(Soundness) $\forall x \notin L, \forall P^*, \Pr[output_V(P^*(x) \leftrightarrow V(x)) = 1]$ is negligible in $|x|$,*

where $output_V(P(x) \leftrightarrow V(x))$ denote the output of V when V interacts with P on input x.

Remark 4 *What happens if the verifier is deterministic? If the verifier is deterministic, then the prover can simulate the verifier. Hence, in this case, interactive proof systems have only as much power as non-interactive proof systems.*

4 Zero-Knowledge Proof Systems

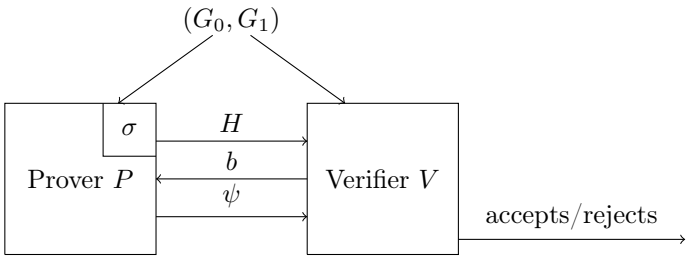
Interaction not only allows proof systems to prove more languages, it also allows us to design proof systems that have interesting properties, one such property is *zero-knowledge*. Informally, a proof system satisfies this property if the verifier "does not learn anything" from the proof besides the fact that the input x is indeed in the language.

For simplicity, we restrict ourselves to languages in NP. In this particular scenario, the prover does not want the verifier to learn the witness w. Moreover, we will assume that the prover has polynomially bounded computational power but the prover knows w (as a part of the prover's input).

Recall the proof system for Graph Isomorphism from Section 2. In the proof system, the prover sends the permutation σ, which is the witness, to the verifier directly. Thus, the verifier learns σ. Here we will design a proof system such that the verifier does not learn anything about σ.

The prove system works as follows:

- First, the prover random a permutation $\varphi : V_0 \to V_0$. Let H be the result of permuting G_0 by φ. The prover sends H to the verifier.
- The verifier selects a random bit $b \leftarrow \{0, 1\}$. The verifier then sends b to the prover to ask the prover to show that H is isomorphic to G_b.
- If $b = 0$, the prover sends $\psi = \phi$ back to the verifier. Otherwise, the prover sends $\psi = \phi \circ \sigma^{-1}$ where σ is the permutation that maps G_0 to G_1.
- The verifier accepts if and only if H is equal to G_b permuted by ψ.



To check that this is a valid proof system, we show its completeness and soundness here.

- (Completeness) Suppose $(G_0, G_1) \in GI$. In this case, it is obvious that the verifier accepts if the verifier is honest.
- (Soundness) Suppose $(G_0, G_1) \notin GI$. No matter what H is, H can only be isomorphic to at most one of (G_0, G_1). As a result, since b is selected at random, the probability that the prover can finds ψ is at most $1/2$ (even if the prover is dishonest).

Again, we can repeat the process multiple times to reduce the soundness to be negligible in $|(G_0, G_1)|$.

Furthermore, the proof system above seems like a zero-knowledge proof system since, even if the verifier receives ψ, it should not help him find σ. More formally, we say that a proof system is zero-knowledge if the distribution of messages in the interaction can be simulated by the verifier himself. This means that he does not learn anything at all from the interaction. The formal definition of a zero-knowledge proof system is shown below.

Definition 5 (Zero-Knowledge Proof System) *A language $L \in NP$ has a zero-knowledge proof system if there exists a pair (P, V) such that both P, V are polynomially-bounded and*

- *(Completeness)* $\forall x \in L, \forall w \in R(x), \Pr[output_V(P(x,w) \leftrightarrow V(x)) = 1] = 1$.
- *(Soundness)* $\forall x \notin L, \forall P^*, \Pr[output_V(P^*(x) \leftrightarrow V(x)) = 1]$ *is negligible in* $|x|$.
- *(Zero-Knowledge) There exists a PPT S (simulator) such that*

$$\forall x \in L, \forall w \in R(x), view_V(P(x,w) \leftrightarrow V(x)) = S(x).$$

where $R(x)$ denote the set of all witnesses of x and $view_V(P(x,w) \leftrightarrow V(x))$ denote the distribution of the messages in the interactions.

Remark 6 *Zero-knowledge definition we used above is called perfect honest-verifier zero-knowledge. There is also a notation for dishonest verifier where the simulator must be able to simulate $view_{V^*}(P(x,w) \leftrightarrow V^*(x))$ even for dishonest V^*. Moreover, when $view_V(P(x,w) \leftrightarrow V(x)) = S(x)$ is changed to $view_V(P(x,w) \leftrightarrow V(x)) \stackrel{c}{=} S(x)$ or $view_V(P(x,w) \leftrightarrow V(x)) \stackrel{s}{=} S(x)$, then the notation becomes computational zero-knowledge and statistical zero-knowledge respectively.*

Finally, note that, for our interactive proof system for GI, a simulator $S(G_0, G_1)$ can be defined as follows:

- Randomly select a bit $b \in \{0, 1\}$.
- Randomly select a permutation ψ.
- Let H be the result of permuting G_b by ψ.
- Output (H, ψ, b).

It is easy to check that the distribution of (H, ψ, b) output from $S(G_0, G_1)$ is the same as that in the interactions. Hence, the defined proof system for GI is indeed a zero-knowledge proof system.