CS 600.442 – Modern Cryptography 10/10/2016

Lecture 9: Zero Knowledge Proofs (I)

Instructor: Abhishek Jain *Scribe: Alishah Chator*

1 What is a Proof?

Generally speaking, a proof is a demonstration of the veracity of statement through a line of deductive reasoning. In the realm of mathematics, this involves reducing the statement into a series of axioms and assumptions that are known to be valid.

Properties of Proofs. Here are two natural properties that a proof must satisfy: first, a verifier should accept the proof if the statement is true. This is known as Correctness. Second, if the statement is false, then any proof should be rejected by the verifier. This is known as Soundness.

It is also important that a proof can be efficiently verified. In particular, a proof that would clearly show if a statement is true, but cannot be verified efficiently is not acceptable. For example, consider the following proof for the existence of infinite primes: a list containing all primes. Clearly, both the generation of the proof and the verification would take an undefined amount of time, so this proof is not useful.

To ensure that proofs can be efficiently verified, we require that the verifier must be polynomial time in the length of the statement.

Must a proof be non-interactive? In exploring the structure of a proof, an important question that arises is whether a proof must be non-interactive? Or, can can a proof be in the form of a conversation between a prover and verifier, where at the end of the conversation the verifier is convinced.

Indeed, we will now formalize the notion of interactive proofs and show that they are extremely powerful!

2 Interactive Protocols

We start by establishing some notation and definitions related to interactive protocols.

Definition 1 (Interactive Turing Machine) *An interactive Turing machine (ITM) is a Turing machine with two additional tapes: a read-only tape to receives messages and a write-only one to send messages.*

Definition 2 (Interactive Protocol) *An interactive protocol is a pair of ITMs such that the read tape of the first ITM is the send tape of the second and vice-versa. An interactive protocol proceeds in rounds, where in each round only one ITM is active, while the other is idle. The protocol is finished when both machines halt.*

Definition 3 (Protocol execution) *A (randomized) execution of an interactive protocol* (M_1, M_2) *refers to the ITMs executing all the rounds of the protocol until they halt. An execution of* (M_1, M_2) *on inputs* (x_1, x_2) *and auxiliary inputs* (z_1, z_2) *is denoted as* $M_1(x_1, z_1) \leftrightarrow M_2(x_2, z_2)$.

Definition 4 (Protocol Output) *The output of M_i in an execution e of (M_1, M_2) is denoted as $Out_{M_i}(e)$*

Definition 5 (View of ITM) *The view of M_i in an execution e of (M_1, M_2) consists of its input, random tape, auxiliary tape and all the protocol messages it sees. It is denoted as $View_{M_i}(e)$.*

3 Interactive Proofs

Interactive proofs involve a pair of ITMs P and V, where P denotes the prover and V denotes the verifier.

Definition 6 (Interactive Proofs) *A pair of ITMs (P,V) is an interactive proof system for a language L if V is a PPT machine and the following properties hold:*

- *Completeness: For every $x \in L$,*

$$Pr[Out_V[P(x) \leftrightarrow V(x)] = 1] = 1$$

- *Soundness: There exists a negligible function $\nu(\cdot)$ s.t. $\forall x \notin L$ and for all adversarial provers P^*,*

$$Pr[Out_V[P^*(x) \leftrightarrow V(x)] = 1] \leq \nu(|x|)$$

Remark 1 *In this definition, the prover does not have to be efficient. The restriction of efficient provers will be visited later.*

Remark 2 *Note, however, that an* adversarial *prover can be unbounded*

What this definition is saying is that to satisfy the Completeness property, the output of V in the execution of the interactive protocol between P and V should always be accept as long as the statement x is a valid member of the language L of statements. In order to met the Soundness property, regardless of the adversarial prover P^*'s strategy, if x is not a valid statement then, the probability that the output of V in the execution of the interactive protocol between P^* and V is accept must be negligible.

3.1 Why Interactive Proofs?

A natural question that we should ask at this point is, why should we consider *interactive* proofs?

Indeed, for languages that are in NP, for each statement in the language there exists a polynomial sized witness. Specifically, for any NP language L with associated relation R and any statement $x \in L$, there exists a witness w s.t. checking $R(x, w) = 1$ confirms that $x \in L$. This means that w is a non-interactive proof for x.

Example. A simple example of non-interactive proofs using witnesses is Graph Isomorphism. Two Graphs are isomorphic if there is a permutation that maps one graph to the other. In this situation the permutation is the witness, as if a permutation can be shown then clearly a mapping must exist between the two graphs.

In light of the above, the question is why even bother with interactive proofs? Why not always use non-interactive proofs?
There are two main reasons for using interactive proofs:

- Proving statements for languages not known to be in NP (i.e., when a "short" witness is not available).
- Achieving a privacy guarantee for the prover

In particular, here are some known results that establish the power of interaction:

- Shamir proved that IP = PSPACE. That is the space of languages with interactive proof systems (with a single prover) is equivalent to the space of languages decidable in polynomial space.
- Babai-Fortnow-Lund established that MIP = NEXP. That is the space of languages with Multi-prover interactive proof systems is equivalent to the space of languages decidable in non-deterministic exponential time.
- Goldwasser-Micali-Rackoff presented the notion of Zero Knowledge, where verifier learns nothing from the proof beyond the validity of the statement.

In what follows, we will demonstrate the power of interaction by constructing interactive proofs for a language in co-NP, and then later, we will formalize the notion of zero knowledge.

Below, we first establish some general notation for graphs that we will later use.

4 Notation for Graphs

Definition 7 (Graph) *A Graph $G = (V, E)$ where V is a set of vertices and E is a set of edges s.t. $|V| = n$, $|E| = m$*

Definition 8 *Π_n is the set of all permutations π over n vertices.*

Definition 9 (Graph Isomorphism) *$G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$ are isomorphic if there exists a permutation π s.t:*

- $V_1 = \{\pi(v) | v \in V_0\}$
- $E_1 = \{(\pi(v_1), \pi(v_2)) | (v_1, v_2) \in E_0\}$

Remark 3 *We will also use the notation $G_1 = \pi(G_0)$*

Graph Isomorphism is an NP problem, so even if we did not explain what the witness for this problem is, we know it must have one and thus can be proved non-interactively. However, there is a related problem that is not known to be in NP and thus cannot be efficiently proved using a witness.

Definition 10 (Graph Non-Isomorphism) *G_0 and G_1 are non-isomorphic if there exists no permutation $\pi \in \Pi_n$ s.t. $G_1 = \pi(G_0)$*

5 Interactive Proof for Graph Non-Isomorphism

Suppose we want to prove that two graphs G_0 and G_1 are not isomorphic. Note that graph non-isomorphism is in co-NP, and not known to be in NP.

A naive way to prove this is by enumerating all possible permutations over n vertices and showing that there is no permutation $\pi, G_1 \neq \pi(G_0)$. Note, however, that this cannot be verified efficiently.

Fortunately this is where the power of interaction comes in. We now demonstrate an interactive proof system for graph non-isomorphism.

Common Input: $x = (G_0, G_1)$

Protocol (P,V): Repeat the following procedure n times using fresh randomness

Verifier		Prover
Chooses a random $b \in \{0, 1\}$, $\pi \in \Pi_n$. Compute $H = \pi(G_b)$		
	$\xrightarrow{\text{send H}}$	
		Compute b' s.t. H and $G_{b'}$ are isomorphic
	$\xleftarrow{\text{Send } b'}$	

V(x,b,b'): V outputs 1 if b'=b and 0 otherwise.

We now argue that protocol (P, V) is an interactive proof. As per the definition, we have to establish that it satisfies the properties of Completeness and Soundness:

- Completeness: If G_0 and G_1 are not isomorphic, then an unbounded prover can always find b' s.t. b'=b. This is because H would only be isomorphic to one of the two graphs.
- Soundness: If G_0 and G_1 are isomorphic, then H is isomorphic to both G_0 and G_1. Thus in a single iteration, an unbounded prover can guess b with probability at most 1/2. Since each iteration is independent, over n iterations, the probability of prover success is at most 2^{-n}, which is negligible.
- Additionally, the verifier is clearly efficient.

6 Interactive Proofs with Efficient Provers

Up until this point, the provers we were dealing with were inefficient. If there were not, then the previous protocol would have established that graph non-isomorphism is in NP.

However, what if we want Interactive proofs with efficient provers? One reason for this is because now, we can hope to implement prover strategies using standard computers or human beings (who are PPT machines). Further, we can hope to construct interactive proofs that are also zero knowledge.

In order to construct interactive proofs with efficient provers, we can only deal with languages in NP. In particular, for any statement x, we will provide a witness w for x as a private input to the prover. Then, we require that the prover strategy is be efficient when it is given a witness w for the statement x that it attempts to prove.

Definition 11 *An interactive proof system (P, V) for a language L witness relation R is said to have an efficient prover if P is a PPT and the completeness condition holds for every $w \in R(X)$*

Remark 4 *Even though honest P is efficient, we still require soundness guarantee against all possible adversarial provers.*

6.1 Interactive proof for Graph Isomorphism

We now construct an interactive proof for proving that two graphs G_0 and G_1 are isomorphic. At first, this may seem a little strange since as discussed earlier, there exists a simple non-interactive proof for the same: the prover simply sends the permutation that maps G_0 to G_1 to the verifier. Indeed, if the prover is provided this permutation as input, then it is already efficient.

The problem, however, with this protocol is that V learns the permutation π. Now using this permutation, it is able to repeat the proof to someone else.

This raises the natural question whether there is a way to interactively prove isomorphism without revealing the witness. Even better yet, can we construct construct a proof that reveals nothing to V beyond the validity of the statement?

Below, we construct such an interactive proof system for graph isomorphism.

Common Input: $x = (G_0, G_1)$

P's witness: $G_1 = \pi(G_0)$

Protocol (P,V): Repeat the following procedure n times using fresh randomness.

Prover		Verifier
Chooses a random $\sigma \in \Pi_n$, Compute $H = \pi(G_0)$		
	$\xrightarrow{\text{send H}}$	
		Choose random bit b $\in \{0, 1\}$
	$\xleftarrow{\text{send b}}$	
	$\xrightarrow{\text{if b=0, sends } \sigma}$	
	$\xrightarrow{\text{else send } \phi = \sigma \cdot \pi^{-1}}$	

V(x,b,b'): V outputs 1 iff $H = \phi(G_b)$

Proof of Completeness: If G_0 and G_1 are isomorphic, then V always accepts since $\sigma(G_0) = H$ and $\sigma(\pi^{-1}(G_1)) = \sigma(G_0) = H$.

Proof of Soundness: If G_0 and G_1 are not isomorphic, then H is isomorphic to one of the graphs but not both. Since b is chosen randomly after fixing H, H is not isomorphic to G_b with probability 1/2. Thus an unbounded adversarial prover can succeed with probability at most 1/2. Over n independent iterations, the prover can succeed with probability at most 2^{-1}.

In this protocol, one can see that intuitively, V obtained no information other than a random permutation of G_b. This is something he could have generated on its own, so intuitively, the protocol does not reveal any information.

Below, we formalize the idea of zero knowledge and then later, we will prove that the graph isomorphism protocol constructed above is in fact zero knowledge.

7 Zero Knowledge

Intuitively, a protocol is zero knowledge if the verifier does not "gain any knowledge" from interacting with the prover besides the validity of the statement. Towards formalizing this idea, the first natural question is how to formalize "does not gain any knowledge?"

Here are some rules to help in this direction:

- Rule 1: Randomness is for free
- Rule 2: Polynomial-time computation is for free

In other words, learning the result of a random process or a polynomial time computation gives us no knowledge.

The next question, however, is what is knowledge? To answer this question, let us understand when knowledge is conveyed.

- Scenario 1: Someone tells you he will sell you a 100-bit random string for \$1000.
- Scenario 2: Someone tells you he will sell you the product of two prime numbers of your choice for \$1000.
- Scenario 3: Someone tells you he will sell you the output of an exponential time computation (e.g., isomorphism between two graphs) for \$1000.

Which of these offers should you accept?

Note that in the first scenario, we can generate a random string for free by flipping a coin. The second scenario can also be obtained for free since multiplying is a polynomial-time operation. However, since an exponential-time operation is hard to compute for a PPT machine, scenario 3 is the best one to consider.

The moral of the story is that we do not gain any information from an interaction if we could have performed it on our own. This leads us to the correct intuition for zero knowledge:

Intuition for Zero Knowledge: A protocol (P, V) is zero knowledge if V can generate a protocol transcript on its own, without talking to P. If this transcript is indistinguishable from a real execution, then clearly V does not learn anything by talking to P.

To formalize this intuition, we will use the idea of a Simulator as we did when defining semantic security for encryption.

Definition 12 (Honest Verifier Zero Knowledge) *An interactive proof (P,) for a language L with witness relation R is said to be honest verifier zero knowledge if there exists a PPT simulator S s.t. for ever n. u. PPT distinguisher D, there exists a negligible function $\nu(\cdot)$ s.t. for every $x \in L, w \in R(x), z \in 0, 1^*$, D distinguishes between the following distributions with probability at most $\nu(n)$:*

- $\{View_V[P(x, w) \leftrightarrow V(x, z)]\}$
- $\{S(1^n, x, z)\}$

In other words what V sees throughout the protocol is something that could have come up with on its own (by simply running the simulator with input x and z).

Remark 5 *The auxiliary input z to V captures any a priori information V may have about x. Definition promises that V does not gain any other knowledge.*

Issue. A problem with the above definition is that it promises security only of the verifier V follows the protocol. What if V is malicious and deviates from the honest strategy? In this case, we need a simulator S for every, possibly malicious (efficient) verifier strategy V^*.

We now present a definition of zero-knowledge for this case. For technical reasons, we allow the simulator to run in expected polynomial time.

Definition 13 (Zero Knowledge) *An interactive proof (P, V) for a language L with witness relation R is said to be zero knowledge if for every n.u. PPT adversary V^*, there exists an expected PPT simulator S s.t. for every n.u. PPT exists an expected PPT a negligible function $\nu(\cdot)$ s.t. for every $x \in L, w \in R(x), z \in 0, 1^*$, D distinguishes between the following distributions with probability at most $\nu(n)$:*

- $\{View_{V^*}[P(x,w) \leftrightarrow V^*(x,z)]\}$
- $\{S(1^n, x, z)\}$

Remark 6 *If the distributions are statistically close, then we call it statistical zero knowledge. If they are identical then it is know as perfect zero knowledge.*

We see that in this revised definition, no matter the verifier's strategy, the view of V^* is indistinguishable from the output of the simulator.