6.875J/18.425J Cryptography and Cryptanalysis	December 3, 2001

Lecture 23: Zero Knowledge Proofs

Instructor: Anna Lysyanskaya — *Scribed By: Adrian Birka*

In this lecture we will consider a new notion of proof, called interactive proof, and an important special case of these known as Zero Knowledge (ZK) proof. All the material from today's and next lectures is due to the seminal paper of Goldwasser, Micali and Rackoff, "The Knowledge Complexity of Interactive Proof Systems," which introduced the notion of interactive proof, as well as the notion of zero-knowledge. This paper led to much of modern cryptography, as well as tremendous advances in complexity theory, and won the Godel prize in theoretical computer science.

1 Classical vs. Interactive proofs

The best way to introduce the idea of an interactive proof is by an example, and only then to give a formal definition. That is what we do:

1.1 Example: Graph Non-Isomorphism problem

Given two non-isomorphic graphs, G and H, Non-Isomorphism problem is to prove that they are non-isomorphic. This problem will be hereinafter referred to as NISO problem.

Recall that two graphs are isomorphic if the vertices of one can be relabelled (permuted) to get the other graph. For example, the graphs $\{(1,2),(2,4),(4,3),(3,1)\}$ and the graph $\{(2,1),(1,3),(3,4),(4,2)\}$ are isomorphic, since relabelling vertices of the second produces the first. We just need to rename 3 to be 2, 4 to be 3 and 2 to be 4.

However, the graph $\{(1,2),(2,4),(3,4),(4,1)\}$ is not ismorphic to either one of those. In this case the proof is easy - in the last graph the vertix 4 has 3 incident edges on it, but in previous two graphs, every vertix has only two edges incident on it.

Recall the concept of a classical proof system. Given a language L, a proof system for L is a pair of algorithms P, V (the prover and the verifier), such that on input $x \in L$, P produces p, and on input (x,p), V accepts, and on input $x \notin L$, there is no p for which V accepts (x,p). We generally want p to be polynomial length in x and V to run in polynomial time. Of course, languages for which this is possible are exactly the collection NP.

Unfortunately, NP is not, as far as we know, big enough to contain some quite useful languages. For example, if one wants to prove that $G \not\cong H$, one could write down as the proof all permutations of G, and to verify the proof one would check that none of these permutations is H. However, this proof is exponentially long. It is not known how to provide a proof of length polynomial in $max(|G|,|H|)$, with the verifier also running in polynomial time. To help us with this problem, we turn to the concept of interactive proofs.

1.2 Interactive proof for NISO

In the classical concept of the proof system, the prover produces a proof of a fact, and the verifier than checks it and accepts or rejects the proof. However, in real life, a proof is anything that is convincing. So, why now allow interactive proofs, where the verifier becomes convinced by talking to the prover, not by just examining an already complete proof? That is what we do.

Consider the NISO problem. Suppose that the verifier and the prover run the following algorithm, hereinafter referred to as Dialogue A.

- Verifier picks G or H at random, and randomly permutes the vertices; sends the result R to the prover.
- Prover finds which of H, G is isomorphic to R and sends that to the verifier.
- If the prover sends the wrong graph (not the one picked in first step) to the verifier, verifier rejects; otherwise verifier repeats the previous steps until he is satisfied, and then accepts.

It should be clear intuitively that Dialogue A is convincing with small probability of error, since if G and H are isomorphic, the probability that any prover will be able to return always the same one as the verifier picked is very small. To make this more precise, we need to define formally the concept of an interactive proof system.

1.3 Interactive proof system

.

Defn: An interactive proof system for a language L is a pair of interactive algorithms, P, V such that:

- Completeness: All true statements are convincing.

  $$\exists \nu \in negl \; \forall x \in L \; Pr[(P(x) \leftrightarrow V(x)) \rightarrow d : d = Accept] \geqslant 1 - \nu(|x|)$$

  The probability $1 - \nu$ above is known as completeness probability. It is a theorem that any interactive proof system can be changed to have the completeness probability equal 1.

- Soundness: no false statement is convincing (no adversary can convince the verifier of any false statement).

  $$\forall x \notin L \; \forall A \in Algs \; \exists \nu \in negl \; Pr[(A(x) \leftrightarrow V(x)) \rightarrow d : d = Accept] \leqslant \nu(x)$$

  The probability ν above is known as soundness probability. Note that the adversary A is allowed to be computationally unbounded in the above definition.

In addition, we demand that V be a PPT, since we want proof verification to be efficient. We call an interactive proof system in which V is efficient an efficient interactive proof system.

Now that we have the definition, let's ask whether Dialogue A is an interactive proof system. The answer is yes.

Claim: Dialogue A is an efficient interactive proof system for NISO.

Proof: It is clear that V is efficient. We verify the other two conditions.

1. Completeness: If $G \not\cong H$, then R is isomorphic to at most one of them (and exactly one, if we assume, as we should, that V is playing by the rules). So the prover's response is always the same as what verifier expects, hence completeness is one.

2. Soundness: If $G \cong H$ then R is isomorphic to both, and which one the verifier expects is independent of R, hence independent of the adversary's choice. Thus, soundness is $1/2$ for a single round of the algorithm; to achieve negligible soundness, let k be $max(|G|, |H|)$ and do k rounds in Dialogue A.

□

It should be noted, that just as in the above proof, for a general proof system if the difference between completeness and soundenss can be made non-negligible, than by repeating the algorithm many times we can get negligible soundness and negligibly smaller than one completeness.

2 Zero Knowledge Proofs

It is often useful to be able to convince someone of something without revealing more information than you want. This is the concept that is formalized as Zero Knowledge proofs. As usual, we do an example first, to gain intuition, and proceed to formalize the intuition afterwards.

2.1 Example: Quadratic residuosity proof system

The goal is, given an RSA modulus n and a quadratic residue a mod n, to prove that $a \in QR(n)$. The proof system is as follows:

- Prover picks $r \leftarrow \mathbb{Z}_n^*$, then computes, $s = r^2$ and α such that $a = \alpha^2$.
- Prover sends s.
- Verifier picks $b \leftarrow \{0, 1\}$ and sends it.
- Prover computes $t = r\alpha^b$, sends t.
- Verifier checks that $t^2 = sa^b$. If not, rejects, if yes, repeat $|n|$ times, and if all pass, accept.

We first show that this is an interactive proof system for quadratic residuosity.

Proof:

1. Completeness: This is one for each round, since the prover always makes an answer that verifies.

2. Soundness: Suppose $a \notin QR(n)$, then for any s either $s \notin QR(n)$ or $sa \notin QR(n)$. b is chosen independently of s, so verifier rejects with probability $1/2$. For k rounds the acception probability is $(1/2)^k$, so soundness holds if there are $|n|$ rounds.

$\square$

The next question is, what do we mean that the verifier learns nothing from this proof? We recall the intuition with encryption: whatever an adversary can compute after having seen the cyphertext should have been computable without the cyphertext. In the case of zero knowledge proofs we want to make sure that whatever an adversary V^* learns interacting with P can be computed without talking to P.

However, this creates a paradox - if the verifier learns nothing, how can he be ever convinced of anything? The answer is that, since the only way P could have cheated is if he knew the verifier's random tape, verifier is convinced because he knows his random tape. So in our intuition we will allow the randomness that V^* had to be used in computing the same things that V^* is computing, but without talking to P. This brings us to formalize ZK proofs as follows:

2.2 Formalization of ZK proofs

In this section, we will write $V^*(x, R)$ to inicate that the algorithm V^* ran on input x and its random tape was R. R can then be used by other algorithms as a usual input.

Defn: A ZK proof system for a language L is an interactive proof system such that

$$\forall V^* \in IPPTF\ \exists M \in PPTF\ s.t.\ \forall D \in PPTF\ \exists \nu \in negl\ s.t. \forall x \in L$$

$$Pr[u_0 \leftarrow (V^*(x,R) \leftrightarrow P(x)), u_1 \leftarrow M(x,R),$$
$$b \leftarrow \{0,1\}, b' \leftarrow D(x,R,u_b) : b = b'] \leqslant 1/2 + \nu(|x|)$$

In words, no distinguisher D can distinguish output of V^* from that of M.

The question now is whether the example above satisfies this definition. The answer is yes.

Proof: The key idea to construct M is this: M has access to V^*'s randomness and to its code, so he can simulate running of V^*. By putting things on V^*'s receive tape and reading things off of its send tape, M can talk to V^*. If M can do it so that V^* notices no difference between this and talking to P, than M just outputs whatever V^* does, and the output will be indistinguishable from that of V^* talking to P.

So how do we arrange for M to simulate P? Well, the key is to go backwards. M does the following:

- Choose $t \leftarrow \mathbb{Z}_n^*$, $b' \leftarrow \{0,1\}$, set $s = t^2/a^{b'}$. Send s to V^*.
- Receive b from V^*; if $b \neq b'$, 'rewind' V^* to the state it was before it received s (this can be done, since M contains V^* inside of itself, and it knows the necessary randomness).
- If $b = b'$, send t.

Note that b' is distributed independently on s, and hence, since V^* only sees s before producing b, it is independent of b. Hence the probability of $b = b'$ is $1/2$ for each round, and is independent from round to round. Thus for each round of conversation between P and V^*, M will do on average 2 trials in the conversation with V^* it contains inside itself. Therefore, the expected running time is $2k$ trials, and is polynomial.

We must show that V^* sees the same conversation when it is inside M as when it talks to P.

Claim: For any sequence q of k rounds, $Pr[q$ is a sequence in a talk with $P] = Pr[q$ is generated by $m]$.

Proof: If q is not a sequence with ith element, for each i, being (s_i, b_i, t_i) with $t_i^2 = s^i a^{b_i}$, the probability is 0 in both cases. So suppose that the above relation holds for each element of the sequence q. Then $Pr[(s_1, b_1, t_1), \ldots, (s_k, b_k, t_k)$ comes from $P]$ is the probability that P picked $s_1 \ldots s_k$, i.e. $\left(\frac{1}{|\mathbb{Z}_n^*|}\right)^k$.

To compute $Pr[(s_1, b_1, t_1), \ldots, (s_k, b_k, t_k)$ comes from $M]$, we note that M would do an equivalent thing if it just prepicked many (say k^2) triples (s_i', b_i', t_i'), and then go through them and junk any where b_i' does not match the b sent by V^*. Since the choice of all these triples is independent from the choice of which triples to junk, the probability $Pr[(s_1, b_1, t_1), \ldots, (s_k, b_k, t_k)$ comes from $P]$ reduces just to $Pr[M$ picks $t_1 \ldots t_k$, which again is $\left(\frac{1}{|\mathbb{Z}_n^*|}\right)^k$.

□

The above claim finishes the proof that this is a ZK system, since the only part of the definition not yet proved, the efficiency of V, is clear.

□

In fact, we have shown more than what the definition of ZK system demands. We have shown that M and V^* talking to P produce equal probability distributions, so even if we allowed D to be computationally unbounded and to run either M or V^* talking to P as many times as it wants, it still would not be able to distinguish between the two.

But in general, ZK systems cannot necessarily be ran more than once without giving knowledge away. For example, suppose that P in the previous example were really stupid and published, for a prime p dividing n and a random string r it fixed in advance, either $p \oplus r$ or r each time it ran (choosing an option randomly). Then with one run, this is still a ZK system (since the additional published information looks random), but with multiple runs, after an average of two runs, the adversary would have seen both r and $r \oplus p$, and hence would learn n's factorization!