Notes for Lecture 26

Scribed by Anindya De, posted May 4, 2009

Summary

In this lecture, we show that the protocol for quadratic residuosity discussed last week is indeed zero-knowledge. Next we move on to the formal definition of *proof of knowledge*, and we show that the quadratic residuosity protocol is also a proof of knowledge. We also start discussing the primitives required to prove that any language in NP admits a zero-knowledge proof.

1 The Quadratic Residuosity Protocol

Last time we considered the following protocol for quadratic residuosity:

- Verifier's input: an integer N (product of two unknown odd primes) and a integer $r \in \mathbb{Z}_N^*$;
- Prover's input: N, r and a square root $x \in Z_N^*$ such that $x^2 \bmod N = r$.
- The prover picks a random $y \in Z_N^*$ and sends $a := y^2 \bmod N$ to the verifier
- The verifier picks at random $b \in \{0, 1\}$ and sends b to the prover
- The prover sends back $c := y$ if $b = 0$ or $c := y \cdot x \bmod N$ if $b = 1$
- The verifier checks that $c^2 \bmod N = a$ if $b = 0$ or that $c^2 \equiv a \cdot r \pmod N$ if $b = 1$, and accepts if it is so.

Clearly, the protocol is complete i.e. if $x^2 \bmod N = r$, then the verifier accepts with probability 1. To show soundness of the protocol, note that if r is not a quadratic residue mod$N$, then for any $a \in Z_N^*$, both a and ar cannot be quadratic residues in mod N. Hence, in case r is not a quadratic residue, the verifier rejects with probability at least $\frac{1}{2}$.

We now show that the above protocol is also zero knowledge. More precisely, we show the following.

Theorem 1 *For every verifier algorithm V^* of complexity $\leq t$ there is a simulator algorithm of average complexity $\leq 2t + (\log N)^{O(1)}$ such that for every odd composite N, every r which is a quadratic residue* (mod N) *and every square root of x of r, the distributions*

$$S^*(N, r) \tag{1}$$

and

$$P(N, r, x) \leftrightarrow V^*(N, r) \tag{2}$$

are identical.

PROOF: The simulator S^* is defined as follows. It first picks $b_1 \in \{0, 1\}$ uniformly at random. It also picks $y \in Z_n^*$ uniformly at random. If $b_1 = 0$, set $a = y^2$ and if $b_1 = 1$, set $a = y^2 r^{-1}$. Note that irrespective of the value of b_1, a is a uniformly random element of Z_n^*. With this S^* simulates the interaction as follows. First, it simulates the prover by sending a. If the second round reply from V^* (call it b) is not the same as b_1, then it aborts the simulation and starts again. If not, then clearly $c = y$ is the reply the prover will send for both $b = 0$ and $b = 1$. Hence whenever the simulation is completed, the distribution of the simulated interaction is same as the actual interaction. Also observe that b_1 is a random bit statistically independent of a while b is totally dependent on a (and probably some other random coin tosses). Hence in expectation, in two trials of the simulation, one will be able to simulate one round of the actual interaction.Hence the expected time required for simulation is the time to simulate V^* twice and the time to do couple of multiplications in Z_n^*. So, in total it is at most $2t + (\log N)^{O(1)}$. □

2 Proofs of Knowledge

Suppose that L is a language in NP; then there is an NP relation $R_L(\cdot, \cdot)$ computable in polynomial time and polynomial $p(\cdot)$ such that $x \in L$ if and only if there exists a witness w such that $|w| \leq p(|x|)$ (where we use $|z|$ to denote the length of a bit-string z) and $R(x, w) = 1$.

Recall the definition of *soundness* of a proof system (P, V) for L: we say that the proof system has soundness error at most ϵ if for every $x \notin L$ and for every cheating prover strategy P^* the probability that $P^*(x) \leftrightarrow V(x)$ accepts is at most ϵ. Equivalently, if there is a prover strategy P^* such that the probability that $P^*(x) \leftrightarrow V(x)$ accepts is bigger than ϵ, then it must be the case that $x \in L$. This captures the fact that if the verifier accepts then it has high confidence that indeed $x \in L$.

In a proof-of-knowledge, the prover is trying to do more than convince the verifier that a witness exists proving $x \in L$; he wants to convince the verifier that he (the prover) *knows* a witness w such that $R(x, w) = 1$. How can we capture the notion that an algorithm "knows" something?

Definition 2 (Proof of Knowledge) *A proof system (P, V) for an NP relation R_L is a proof of knowledge with knowledge error at most ϵ and extractor slowdown es if there is an algorithm K (called a knowledge extractor) such that, for every prover strategy P^* of complexity $\leq t$ and every input x, if*

$$\mathbb{P}[P^*(x) \leftrightarrow V(x) \text{ accepts}] \geq \epsilon + \delta$$

then $K(P^*, x)$ outputs a w such that $R(x, w) = 1$ in average time at most

$$es \cdot (n^{O(1)} + t) \cdot \delta^{-1}$$

In the definition, giving P^* as an input to K means to give the code of P^* to K. A stronger definition, which is satisfied by all the proof systems we shall see, is to let K be an oracle algorithm of complexity $\delta^{-1} \cdot es \cdot poly(n)$, and allow K to have oracle access to P^*. In such a case, "oracle access to a prover strategy" means that K is allowed to select the randomness used by P^*, to fix an initial part of the interaction, and then obtain as an answer what the next response from P^* would be given the randomness and the initial interaction.

Theorem 3 *The protocol for quadratic residuosity of the previous section is a proof of knowledge with knowledge error $1/2$ and extractor slowdown 2.*

Proof: Consider an a such that the prover returns the correct answer both when $b = 0$ and $b = 1$. More precisely, when $b = 0$, prover returns a in the third round of the interaction and if $b = 1$, prover returns $a.r$ in the third round of interaction. If we can find such an a, then upon dividing the answers (for the cases when $b = 1$ and $b = 0$) returned by the prover strategy in third round, we can get the value of r. Note that if verifier V accepts with probability $\frac{1}{2} + \delta$, then by a Markov argument, we get that with probability δ, a randomly chosen $a \in Z_n^*$ is such that for both $b = 0$ and $b = 1$, the prover returns the correct answer. Clearly, the knowledge error of the protocol is $\frac{1}{2}$ and for one particular a, the prover strategy is executed twice. So, the extractor slowdown is 2. Note that in expectation, we will be sampling about $\frac{1}{\delta}$ times before we get an a with the aforementioned property. Hence, the total expected time for running K is $2 \cdot ((\log N)^{O(1)} + t) \cdot \delta^{-1}$ $\square$

3 Uses of Zero Knowledge proofs

In the coming lectures, we shall consider general multi-party protocols, an example of which might be playing "poker over the phone/internet". In this, one needs to devise a protocol such that n mutually distrusting players can play poker or any other game over the internet.

Our approach will be to first devise a protocol for the "honest but curious" case, in which all the players follow the protocol but they do try to gain information about others by simply tracking all the moves in the game. To go to the general case, we will require that every player gives a zero knowledge proof that it played honestly in every round. As one can see, this statement is much more general than say that two graphs are isomorphic. However, it is a statement which has a short certificate and hence is in NP. This gives motivation for our next topic which is developing zero knowledge protocols for every language in NP. We shall describe an important primitive for this purpose called *commitment schemes*.

4 Introduction to Commitment scheme

Consider the situation when Alice and Bob are interacting using a protocol. The protocol may want that at some stage Alice commits to a value so that she cannot go back on it later. Simultaneously, it may also require that it is infeasible for Bob to know what the value is to which Alice has committed unless much later in the interaction (may be when Alice wants to reveal it). This kind of a scheme is called a commitment scheme. A real world situation is the following. Alice writes the value she wants to commit to, on a piece of paper and puts it inside a locked box. Subsequently, she sends the locked box to Bob without its key. In this situation, it is not possible for Alice to repudiate what she had committed to but at the same time unless Bob has the key, he also cannot the value Alice has committed to.

There are two parts to such a protocol. One is that Alice cannot deny her commitment and another is that Bob cannot know the contents without help from Alice. It should be clear that exactly one of these things can be information theoretically hard. That is, we may have exactly one of following two situations: It is information theoretically impossible for Alice to go back on her commitment but only computationally infeasible for Bob to know the commitment without Alice's consent and vice versa.