Theoretical Cryptography, Lectures 18-20

Instructor: Manuel Blum
Scribes: Ryan Williams and Yinmeng Zhang

March 29, 2006

1 Content of the Lectures

These lectures will cover how someone can prove in zero-knowledge that they know the factorization of a number N.

2 Simulation - A Definition

A protocol is ZK if the Verifier gains no advantage from talking to the Prover.

We can describe any Verifier who talks to the Prover as $\mathcal{A}^{\mathcal{P}}$, a Turing Machine A with access to the oracle P. We want to say that the output of $\mathcal{A}^{\mathcal{P}}$ is something the Verifier could have come up with for herself.

That is, for any efficient $\mathcal{A}^{\mathcal{P}}$ there is a corresponding efficient $\mathcal{B}$ such that the output of $\mathcal{B}$ has the same distribution as the output of $\mathcal{A}^{\mathcal{P}}$

Previously, we had defined simulation to be the ability to construct an efficient subroutine which could be used in place of the Prover Oracle. That is, at every point $\mathcal{A}$ queried the oracle, our subroutine could be used instead. This is a *blackbox simulation*, since we treat $\mathcal{A}$ as a black box with sockets which can take either a Prover Oracle or the subroutine we constructed.

This definition is more general, in that it could be that $\mathcal{B}$ somehow uses the structure of $\mathcal{A}$.

3 The number of roots of a quadratic

Let $N = p_1^{e_1} \cdots p_k^{e_k}$ be the prime factorization of N, where each p_i is a distinct prime.

Consider the polynomial $p(x) = (x^2 - a) \bmod N$, for $a \in Z_N^*$.

Question: How many roots does $p(x)$ have, if N is prime?

- *Answer: Either* 0 *or* 2. *$GF(N)$ is a field, hence the degree-two polynomial $x^2 - a$ has at most two roots, and if r is a root then so is $-r$.*

If N is a prime power, the answer is still true. We won't cover why this is the case, but we will note that the argument is **not** the same. That is, $\mathbb{Z}_{p^e}^*$ is not a field for any prime p and integer $e > 1$. There *is* a field of p^e elements for any $e > 1$, but it is *not* $\mathbb{Z}_{p^e}^*$. One quick way to see this is that the number of elements in $\mathbb{Z}_{p^e}^*$ is *not* p^e, but rather $(p-1) \cdot p^{e-1}$.

What happens for more general N? Let $N = p_1 \cdot p_2$ where $p_1 \neq p_2$ are both odd.

Claim 3.1 *The polynomial $p(x)$ has either 0 or 4 roots.*

Proof. By the Chinese Remainder Theorem,

$$a \equiv x^2 \bmod N \iff a \equiv x^2 \bmod p_1 \text{ and } a \equiv x^2 \bmod p_2.$$

From the question and answer above, we know that the two equations on the right-hand side each have either 0 or 2 solutions. Let $x_1, -x_1$ be roots of $(x^2 - a) \bmod p_1$, if they exist, and let $x_2, -x_2$ be roots of $(x^2 - a) \bmod p_2$, if they exist. The LHS equation has a solution iff both equations on the RHS have solutions, in which case $a = \langle a_1 \bmod p_1, a_2 \bmod p_2 \rangle$ (in Chinese Remainder representation) has square roots $\langle x_1, x_2 \rangle$, $\langle -x_1, x_2 \rangle$, $\langle x_1, -x_2 \rangle$, $\langle -x_1, -x_2 \rangle$ when written in Chinese remainder representation. □

The claim also holds for $N = p_1^{e_1} p_2^{e_2}$.

In general, we can say the following.

Claim 3.2 *For $N = p_1^{e_1} \cdots p_k^{e_k}$, the polynomial $p(x)$ has either 0 roots or 2^k roots.*

The proof of this for $N = p_1 \cdots p_k$, where the p_i are distinct, is extremely similar to the proof of the claim above, just generalized to the case for arbitrary k.

4 Knowing Roots = Knowing Factors

Let $N = p_1 \cdot p_2$, and let $x = \langle x_1, x_2 \rangle$, $y = \langle x_1, -x_2 \rangle$ be distinct roots of the polynomial $x^2 - a \bmod N$, as in the proof of Claim 3.1. (By "distinct", we mean it is *not* the case that $x = -y$.)

It turns out that knowledge of the roots x and y is *equivalent* to knowledge of p_1 and p_2!

4.1 Knowing Factors ⇒ Knowing Roots

First, let's consider the direction where p_1 and p_2 are known, and we have to compute x and y as roots of $x^2 - a$ in polytime. Knowing p_1 and p_2 means that one can compute the two roots of $(x^2 - a) \bmod p_1$, and the two roots of $(x^2 - a) \bmod p_2$, by standard field arithmetic. Let the first pair of roots be $r_1, -r_1$ and the second pair be $r_2, -r_2$. Then $x, -x$ and $y, -y$ are equivalent to:

$$\langle r_1, r_2 \rangle, \langle -r_1, r_2 \rangle, \langle r_1, -r_2 \rangle, \langle -r_1, -r_2 \rangle.$$

That is, the Chinese remainder representations of the four roots are given by r_1 and r_2. But converting to and from Chinese remainder representation can be done in polytime, so we're done with this direction.

4.2 Knowing Roots ⇒ Knowing Factors

Perhaps the more interesting direction is that knowing the distinct roots of any polynomial of the form $(x^2 - a) \bmod N$ leads to knowledge of N's factors. This direction is established by the following theorem.

Theorem 4.1 *Let $N = p_1 p_2$, and let x and y be distinct roots of the polynomial $p(x) = (x^2 - a) \bmod N$. Let $A = \{gcd(x + y, N), gcd(x - y, N)\}$ and $B = \{p_1, p_2\}$. Then $A = B$.*

Proof. By definition, $a \equiv x^2 \equiv y^2 \bmod N$, where $x \neq y$, $x \neq -y$. Therefore $(x^2 - y^2) \equiv 0 \bmod N$, so N divides $(x^2 - y^2) = (x + y)(x - y)$.

Suppose N divided $x + y$. Then $x + y \equiv 0 \bmod N$, so $x \equiv -y \bmod N$. But since $x, y < N$, this implies $x = -y$, contradicting distinctness. Therefore, N does not divide $x + y$. Similar argument shows that N does not divide $x - y$.

Thus, N divides $x^2 - y^2$, but does *not* divide the factors $x - y$ and $x + y$ of $x^2 - y^2$. It follows that both $gcd(N, x - y)$ and $gcd(x + y)$ are *non-trivial*: one of them is p_1, and the other is p_2. □

To obtain a more "set-theoretic" intuition for what is going on in the above proof, suppose we represented integers as multisets of their prime factors, *e.g.* $N = \{p_1, p_2\}$. Then, $gcd(A, B)$ is taking the *intersection* of sets A and B, and divisibility is subsethood: "A divides B" is equivalent to "the set for A is a subset of the set for B". The number $x^2 - y^2$ is the union of the sets for $x + y$ and $x - y$. Now, N is a subset of $x^2 - y^2$, but is *not* a subset of $x + y$, nor is it a subset of $x - y$. Therefore, exactly one of p_1, p_2 is in the set for $x + y$, and the other one is in the set for $x - y$. Hence the gcd of N and these numbers yields p_1 and p_2.

4.3 Aside: An application

Here's a short, neat application of the above equivalence. Again, let $N = p_1 p_2$. Suppose there is a black box B such that, for any $a \in \mathbb{Z}_N^*$ that is a square, $B(a)$ is some x such that $x^2 \equiv a \bmod N$. That is, B is a **black box for sqrt mod** N.

Theorem 4.2 *Given a black box B for sqrt mod N, one can factor N with high probability in polynomial time, with only a constant number of queries to B.*

Proof. The key idea is that $B(a)$ outputs exactly one square root of a. If we somehow knew the *other* square root as well, then we could factor. Here's how we do that. Pick $r \in \mathbb{Z}_N^*$ uniformly at random. Compute $a = r^2 \bmod N$. Now, look at $s = B(a)$. With probability $1/2$, $s \neq -r$ and $s \neq r$! That is, there are four roots of a, and with probability $1/2$, we chose a root that is distinct from the one that B outputs. Therefore, if we pick $r_1, \ldots, r_k \in \mathbb{Z}_N^*$ at random, and look at $B(r_1), \ldots, B(r_k)$, with probability at least $1 - 1/2^k$, we have two distinct square roots of some a, and can therefore factor N. □

5 Zero-Knowledge Proof of Factorization Knowledge

We finally come to our original task, which is to find a zero-knowledge protocol for proving that one knows the factorization of N. For simplicity, we'll assume that $N = pq$, where $p \neq q$ are primes.

At the top level, we shift the problem to getting the Prover to show he can compute square roots mod N. Here's a first stab at a zero-knowledge protocol, suggested by Barry Hon:

Verifier:	Select $x \in \mathbb{Z}_N^*$ uniformly at random. Compute $a = x^2 \bmod N$.
V $\rightarrow$ P:	a
Prover:	Compute roots of $(x^2 - a) \bmod N$, call them $\pm x, \pm y$.
	If there are no roots, then randomly make up two numbers x and y, and negate them.
P $\rightarrow$ V:	$(f(x), f(y))$ where f is a one-way hash. (f is meant to look random)
Verifier:	Compute $z = f(x)$, accept iff z is found among the pair sent.

The main problem with this approach is that it requires a very random-looking f in order to be zero-knowledge. (If f is truly "random", then the Verifier can generate what the Prover sends by picking a random number for $f(y)$, and checking $f(x)$.) Ideally, we'd like to catch the Verifier, if she doesn't actually know a square root of a. (The step where we randomly make up x and y is suspect.)

5.1 A Formalization of 'random-looking"

A *random oracle* is a function $f : \{0,1\}^n \rightarrow \{0,1\}^n$ chosen uniformly at random from all functions with that domain and range. One way to picture it is a $2^n \times n$ array, where every entry is 0 or 1 based on a random coin flip. $f(x)$, then, is just the xth row of the array read as an n-bit binary number. Unfortunately, it takes exponential time to fill up such an array, and exponential space to maintain it.

However, random oracles are really great for proofs. We can access $f(x)$ quickly, while inverting $f(x)$ requires searching through the whole array. The random oracle is an "optimal 1-way function", and we can now polish up our previous proofs by substituting "random oracle" for "gray paint". Just as before, begin able to show security with access to a random oracle is interesting, even though ideally we would like a protocol which does not need it.

5.2 The Better Solution

Here, we want the Verifier to really know the square root of the number she sends the Prover – reaching into our Bag o' Tricks, we require the Verifier to send a ZKP of that fact. The new protocol is

Verifier:	Select $x \in \mathbb{Z}_N^*$ uniformly at random. Compute $a = x^2 \bmod N$.
V $\rightarrow$ P:	a

Verifier:	Select $y \in \mathbb{Z}_N^*$ uniformly at random. Compute $b = y^2$ and $c = a/b$.
V$\rightarrow$ P:	b, c
P$\rightarrow V$:	0 or 1 uniformly at random
V$\rightarrow$ P:	If 0 $\sqrt{b}$, else $\sqrt{c}$
Prover:	If the square root is wrong, ABORT.

Prover	Select $y \in \mathbb{Z}_N^*$ uniformly at random. Compute $b = y^2$ and $c = a/b$.
P$\rightarrow$ V:	b, c
V$\rightarrow P$:	0 or 1 uniformly at random
P$\rightarrow$ V:	If 0 $\sqrt{b}$, else $\sqrt{c}$
Verifier:	If the square root is wrong, REJECT.

Of course, we repeat the above for k times, as usual. If all iterations do not reject, then ACCEPT.

In order to fully show that the above works, recall that we would need to prove soundness, completeness, and zero-knowledge.