Theoretical Cryptography, Lecture 22

Instructor: Manuel Blum
Scribes: Ryan Williams

April 12, 2006

1 Content of the next few lectures

- The function $f(x) = x^2 \bmod N$ and its applications to:
- Signatures
- Oblivious Transfer
- Certified Mail

2 Introduction

One of the goals of this lecture is to promote a certain method for doing interesting research in cryptography. One way is to take a simple number-theoretic function, and just analyze the hell out of it. Take something that behaves in a peculiar way, and ask "What can we do with this? How can we make use of it?" Motivated by the previous lectures, the function we' look at here is $f(x) = x^2 \bmod N$, where N is a composite made up of two large primes.

Let N be an integer corresponding to a public key of Alice, where $N = P_A \cdot Q_A$, two distinct primes, which Alice knows privately.

Consider the function $f(x) = x^2 \bmod N$.

Anyone who knows N can compute $f(x)$ for any x, readily. But in order to invert f in general, one needs to know the factors P_A and Q_A. That is, if you're given y and can find x such that $f(x) = y$, then you can factor N. We saw this earlier: knowing the possible square roots $x, -x, y, -y$ of $x^2 \bmod N$ is equivalent to knowing the factorization of N. In particular, $gcd(x - y, N)$ and $gcd(x + y, N)$ are non-trivial (*i.e.* they are not equal to 1), and hence they produce a factor of N.

Assuming we can't factor numbers like N in general, what can be done with f? Well, by assuming factorization is hard, we're assuming multiplication is *one-way*: easy to compute, hard to invert. Similarly, the function f is *one-way*.

3 Digital Signatures

What might a one-way function like f be good for? Recall that we are also assuming that Alice lets people know N, but does not let them know N's factors. Therefore, Alice has a capability that others do not have. This is the kind of property we would like for *signatures* to have. In an ideal world, Alice's signature would be a piece of data that only Alice can produce, and no one else can–forgery would be impossible. Just to get the ball rolling, let's propose that Alice's signature function for a message m is the following:

$$Sig_A(m) = \langle m, \sqrt{m} \bmod N \rangle.$$

Before we do anything else, let's consider the minuses of using the above function. Here are some we found in class:

- The message m may not have a square root mod N! Hmm, this means that some messages can't be signed using this approach.
- Suppose Alice signs message m_1 and m_2, and sends the signatures s_1, s_2 of these to people. Then the signature on $m_1 \cdot m_2$ (where $\cdot$ is multiplication) can be forged! It is just $s_1 \cdot s_2$! This is a very undesirable property. Of course, one should always beware of people asking you to sign strange documents...
- Eric Li observed that people asking Alice to sign arbitrary documents can factor N, and therefore impersonate Alice from thereon. In particular, a malicious party could choose a random number r, compute $r^2 \bmod N$, and ask Alice to sign it. Alice gives back some square root of r^2, and it is a distinct root $s \neq r$ with $1/2$ probability. But armed with $r, -r, s, -s$, the malicious party can then factor N, and sign messages just like Alice can.

So there are many problems with using a square root as a signature.

4 Fixing The Signature

To prevent outside parties from factoring so easily, and to try to make it so that more messages can actually be signed, we'll throw randomness into the signature function, and a hash function for good measure.

$$Sig_A(m) = \langle m, r, \sqrt{h(m \circ r)} \bmod N \rangle,$$

where r is an integer chosen uniformly at random from $[1, N]$, the operation $\circ$ is string concatenation, and h is a collision-free hash function, determined by a different integer N'. (Recall that a collision-free hash function is such that if one can find x and y such that $h(x) = h(y)$, then one can factor a large integer. So finding a collision in the hash function is considered difficult.)

What are some pluses of using the above function? For one, if m is n-bits long, then the hash $h(m \circ r)$ will only be of length $O(\log n)$. So, other than the random bit string r, the signature will be somewhat short, compared to $\sqrt{m}$ (which would be roughly $n/2$-bits long).

Does the new signature scheme address the problems we found earlier?

- First, $h(m \circ r)$ *still* may not be a square, so there may not be a square root. However, Manuel assures us that the quadratic reciprocity assumption indicates that for randomly chosen r, $h(m \circ r)$ is a square with probability roughly $1/4$.

- Can we forge a signature of $m_1 \cdot m_2$ using signatures for m_1 and m_2? By getting signatures for m_1 and m_2, we have $\sqrt{h(m_1 \circ r_1)}$ and $\sqrt{h(m_2 \circ r_2)}$ for random r_1, r_2. Multiplying them would give $\sqrt{h(m_1 \circ r_1)h(m_2 \circ r_2)}$, which is extremely unlikely to be the same as $\sqrt{h((m_1 \cdot m_2) \circ r_3)}$, for some random r_3.

- Can a malicious party use Alice's signature (of their carefully chosen document) to factor N? The signature scheme now *introduces* randomness into the message to be signed, and only gives the square root of $h(m \circ r)$. The malicious party can certainly compute $y = h(m \circ r)$, but it is highly unlikely that the party knows another square root of y.

5 Oblivious Transfer

Let's return back to our function $f(x) = x^2 \bmod N$. It turned out to be all right for signature schemes, after some tweaking. Why not use it for something else? Perhaps we can use f to communicate information in an interesting way.

Suppose we have two communicating parties, one of which can compute f^{-1} and one of which cannot. Bob (who does not know the factors of N) sends an x, and Alice (who knows the factors) sends back $y \in f^{-1}(x)$. That is, she sends back exactly one square root of x.

We ask: How might this simple exchange be useful? What can we claim is happening when this exchange occurs? Suppose further that Bob actually knew a square root of x beforehand: he took some uniform random $r \in \mathbb{Z}_N^*$ and set $x = r^2 \bmod N$. Then with probability $1/2$, Alice sends him $y \neq \pm r$, but $y^2 = x^2 \bmod N$. That is, with probability exactly $1/2$, Alice essentially sent the factors of N to Bob, by sending him the other square root of x. *At this point, Alice does not know if Bob knows the factors of N, or not.* If Alice also sent along a message m that is encrypted using the factors of N, then (assuming factorization is hard) there's a $1/2$ chance that Bob can decrypt the sent message, and Alice doesn't know which outcome occurred. This kind of protocol is called an *oblivious transfer.*

Initially, it seems odd, that one might desire a protocol where the sender doesn't even know if the message went through to the receiver. It would appear to defeat the purpose of good communication. But it turns out that a good deal of cryptographic apparatus is built on oblivious transfer. For example, oblivious transfer can be used to do secure computation, where k parties each hold inputs $x_1, \ldots, x_k$, and they all wish to compute a function $f(x_1, \ldots, x_k)$ without giving away information about their inputs. (We won't go into this here. For more, check out Rafail Ostrovsky's lecture notes at `http://www.cs.ucla.edu/~rafail/TEACHING/WINTER-2005/L10/L10.pdf`.)

A simple application of the above oblivious transfer is "coin-tossing across the room". Alice and Bob would like to perform a coin toss over a wire. In particular, we'd like to simulate the situation where Alice tosses the coin across the room to Bob, so that Bob can see the result of the toss but

Alice cannot, yet Bob can prove to Alice what the outcome is. Instead of thinking of the toss as heads/tails, let's think of it in terms of who wins and who loses the toss. We say that Bob wins iff he can transmit the factors of N to Alice at the end of the protocol. Then, Bob wins the toss with $1/2$ probability, and can prove to Alice that he won.

We'll show how oblivious transfer can be used to accomplish a far more applicable task: truly certified mail. The concept will be introduced in the following, and expounded upon in further lectures.

6 All-or-Nothing Sequential Certified Mail

Certified mail is mail in exchange for a receipt. But how good is a receipt? When you send a certified letter, all it proves is that you sent something– but you could have sent an empty envelope. One desirable property of certified mail is for the sender to obtain *true* receipts for mail, that really proves that a particular message was sent. In order to have true receipts, we need for the mail to have an "all-or-nothing" property. That is, if Alice gets a valid receipt, then Bob gets the mail. If Alice does not get a valid receipt, then Bob gets *nothing.* For a motivating example, suppose Alice wants to buy stock, and Bob is her broker. On Friday, Alice sends to Bob the request: "Please buy 100,000 shares of stock with ticker symbol `BLUM` for \$1"[1]. We want Alice to have legal proof that Bob got her message, regardless of the stock's outcome. For if Bob buys the stock and it goes to \$2 over the weekend, then Bob can say "Gee, I was on vacation, didn't get your message." If the stock goes to \$0.02, then Bob can prove that Alice asked him to buy the stock. We want for Alice to be able to prove that Bob got the message in the \$2 case.

In the following lectures, we will describe how all-or nothing, sequential, certified mail can be done. "Sequential" refers to the somewhat annoying but seemingly unavoidable property that Alice and Bob must interact in several rounds in order for the mail to be delivered. At the end of this interaction, Bob gets either the entire message, or nothing. Alice gets a complete receipt (a proof of that the mail was sent), or a bogus receipt. Moreover, the two parties know what the outcome is, at the end of the interaction.

Note that this task of certified mail is *almost* impossible. One can imagine Alice and Bob trying to exchange secrets in an all-or-nothing sequential fashion. The problem is that, in a sequential protocol, there is some point at which one of them knows both of the secrets, but the other does not. At that point, the informed party can simply walk away from the protocol. The difference here is that there is an asymmetry of desired information: Alice wants a receipt and knows the mail, and Bob just wants the mail. This asymmetry allows for certified mail to be possible.

[1] Turing Machine Repair: A Family Business. Inquire Within. Expected IPO in 3rd Quarter 2006