15-503/859P: Cryptography **Lecturer:** Manuel Blum
Topic: Another version of Lecture 22 **Date:** 13 April 2006
Scribe: Yinmeng Zhang

22.1 Overview

Last time we did PKE with

$$f(x) = x^2 \pmod{N}$$

where

$$\underbrace{N}_{\text{Public Key}} = \underbrace{P * Q}_{\text{Private Keys}} \quad P,Q \text{ odd primes}$$

Such that anyone knowing N could encrypt, but being able to decrypt was provably equivalent to knowing P (or Q).

Today,

1. Signatures with $f(x) = x^2 \pmod{N}$
2. Oblivious Transfer with $f(x) = x^2 \pmod{N}$
3. Introducing All-or-Nothing Certified Mail

22.2 Signatures

Alice's *signature* for a message m is something which she alone can easily compute, but everyone else can verify. Thus, anyone reading a signed message from Alice can be sure that she was the one who wrote it.

As usual, Alice's public key is N_A, and her private key are its two odd prime factors P_A and Q_A. Our high-level idea is that everyone can compute $f(x) = x^2 \mod N_A$, but only Alice can compute the inverse $f^{-1}(x) = \sqrt{x} \mod N_A$.

Our first stab at an encryption scheme:

$$Signature_A(m) = \langle m, \sqrt{m} \mod N_A \rangle$$

To verify the signature, the reader checks $m \equiv (\sqrt{m})^2 \mod N_A$.

22.2.1 Note on Encryption

If Alice wants to send a signed, encrypted message to Bob, she sends $S_A(E_B(m))$ - encrypted with *Bob*'s public key, and signed under *Alice*'s public key. $S_A(E_A(m))$ would be rather disastrous, as the signature scheme inverts the encryption.

22.2.2 Pluses and Minuses

Being able to take square roots is equivalent to being able to factor N_A, which is a plus. Unfortunately, there are a number of minuses.

1. Once you have signed a message, such as "Withdraw one billionty dollars from Alice's bank account", an Eavesdropper could take the signed message and use it multiple times. The standard solution to this is to time stamp all messages before signing them, so that they can only be used once.
2. Multiple messages could have the same signature.
3. Suppose your Evil Secretary hands you a message to sign – which she constructed by squaring a random number a! Once you sign it, there is a 50/50 chance you have revealed the other square root, and your Evil Secretary will know your secret.
4. From signatures on messages m_1, m_2, anyone can compute the signature on their product – $S[m_1 m_2] = S[m_1]S[m_2]$.
5. If you send the same message multiple times, you had better send the same square root each time, or you will be revealing your secret. An easy fix to this is to use a deterministic algorithm to decide which root to send.
6. The scheme fails if m isn't a quadratic residue.

22.2.3 Improved Scheme

Fix and publish a collision-free hash function h which maps to $\mathbb{Z}_N^*$. Given message m, we choose random r and send

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

The quality of the signature now hinges on the quality of the hash function.

By definition of collision-free, it is hard to find pairs x, y such that $h(x) = h(y)$. This implies the new scheme is resistant to Evil Secretaries, since she can no longer predict what you will take the square root of.

The scheme no longer commutes with multiplication, and the randomness makes it safe to send m multiple times.

But can we can force $h(m \circ r)$ to have a square root? If h maps uniformly onto $\mathbb{Z}_N^*$, 1/4 of the inputs will hash to something with a square root. Somehow, we would like to say that by choosing random r we will quickly hit upon one such that $h(m \circ r)$ is a quadratic residue.

22.3 Oblivious Transfer using $f(x) = x^2 \pmod N$

An *oblivious transfer* or OT protocol allows the sender to transfer information to the receiver while remaining oblivious to what information was sent. Here, we describe a protocol where Bob receives information half the time, and Alice cannot distinguish when this occurs.

Suppose Alice has $N_A = P_A * Q_A$ as usual. Alice OTs the factors of N_A to B as follows:

1. $B \to A$ $x^2 \mod N_A$
2. $A \to B$ a root of x^2

At this point B has a 50/50 shot of being able to factor N, and Alice is oblivious.

The "50/50 shot" guarantee leads to some nice applications. For example, by iterating the protocol, Alice can share a secret key with Bob in expected polynomial time. Alternatively, this can be used to implement a coin flip over the telephone, where Bob wins iff he can factor N_A. But how can we use the fact Alice doesn't know what was sent over?

22.4 Certified Mail

All-or-nothing sequential certified mail (which we will abbreviate AoNSCM) is a protocol which allows Alice and Bob to exchange a piece of mail and a receipt containing a description of the mail sent. *All-or-Nothing* means Bob receives any information about the mail iff Bob receives the whole mail iff Alice receives the whole receipt. *Sequential* means we will not use any nasty tricks which require Alice and Bob to send mail simultaneously – Alice and Bob can talk one after the other, as in all our previous protocols. *Certified* refers to the receipt.

The post office actually has something called "certified mail", but it is vastly inferior to the protocol described here. You can get proof that something was sent through the mail, but the receipt will not contain information on what was sent, so your receiver cannot get verification you sent the correct thing.

22.4.1 Impossible!

The interesting thing about AoNSCM is that it seems impossible. Since we require the protocol to be sequential, consider the very last message sent, wlog let the message be sent by Bob. Just before he sends the message, Bob knows he will receive no more information from Alice, so he must already have the message. Alice is still waiting for a message. It seems that Bob could just abort the protocol and leave Alice hanging. In fact, using this kind of argument we can show it is impossible for Alice and Bob to exchange mail both ways (simultaneous key exchange). Somehow, a receipt is still possible.

22.4.2 Yet Useful!

Suppose A is Alice and B is her Broker, and consider the following conversation:

1. $A \to B$: Please buy stock for \$1! signed, Alice

2. If stock goes to \$.5, $B \to A$ "Sorry, you lost a lot of money", and B can prove A wanted to buy the stock because the message was signed.

   If stock goes to \$2, $B \to A$ "Sorry, I was on vacation.", and keeps the money to himself. Alice *cannot* prove she wanted to buy the stock... unless we can get her a receipt.

We will develop a protocol for AoNSCM next time.