Theoretical Cryptography, Lectures 23-24

Instructor: Manuel Blum
Scribe: Ryan Williams

April 17, 2006

1 Certified Mail

Last time, we introduced the idea of all-or-nothing sequential certified mail, and claimed that oblivious transfer can be used to implement it. We defined an oblivious transfer of $N_A = P_A \cdot Q_A$ from Alice to Bob as follows:

Bob:	Choose x at random from $\mathbb{Z}^*_{N_A}$.
Bob $\rightarrow$ Alice:	$x^2 \bmod N_A$.
Alice $\rightarrow$ Bob:	$\pm x$ or $\pm y$, via P_A and Q_A

After Alice sends Bob a root, then *alea iacta est*: the die is cast. Bob knows if he can factor or not. Alice does not know if Bob can factor or not. Formally speaking, what do we mean by "does not know"? We mean that *any* randomized algorithm who sees only what Alice sees (N_A, $x^2 \bmod N$, and a root) cannot determine if Bob has distinct roots of $x^2 \bmod N$ with probability greater than $1/2$. Without knowing the root that Bob chose, the best that any algorithm can do is just randomly guess.

The above protocol is called a $1/2$-$O.T.$, meaning that the chance of Bob getting the factors of N_A is $1/2$.

Here are two possible final exam questions based on the above:

1. Bob can "cheat" in that, if he actually *does* get the factors of N_A (by knowing both x and y), he can still just say back to Alice "nope, didn't get it". Turn the above 1/2-O.T. into a modified O.T. where Bob cannot cheat. That is, if he does get the factors of N_A from Alice, then Alice can catch him if he says he did not.

2. Create a 1/8-O.T., where Bob has exactly 1/8 chance of getting information from Alice. Here, "information" is in general an all-or-nothing concept: *e.g.* he either obtains an entire factorization or nothing, he either gets a bit from Alice or nothing from Alice, *etc.*. (By "nothing", we of course mean that the information Alice sends can be easily simulated by Bob, without any knowledge of N_A or Alice: in this sense, getting $-x$ when Bob knows x is "nothing.")

2 Certified Mail: Mail in Exchange for a Receipt

We discussed the idea of certified mail in the previous lecture. To recall, Alice wants to send mail to Bob, getting back a receipt. We want the property that Bob gets the mail *if and only if* Alice gets a valid receipt for it. The properties our mail protocol will have are:

- If Alice follows protocol, then she knows (with 1/8 probability of being fooled) whether or not Bob got the mail and whether or not she has a receipt.
- If Bob follows protocol, then he knows for certain whether or not he got mail. The chance that he is cheated by Alice (she gets a valid receipt but doesn't get mail is) at most 1/8.

We'll give the ideas for two protocols. The first protocol has the main gist of what will work, but it doesn't quite work. The second protocol will be a working one that is a slight augmentation of the first.

3 Protocol 1

First, we assume upfront that all messages from Alice to Bob are digitally signed with Alice's signature, and all messages from Bob to Alice are signed with Bob's signature.

The protocol is that Alice encodes her message m using an encryption that relies on the factorization of a number N_A, call this encryption $E_{N_A}(m)$. She first sends $E_{N_A}(m)$ to Bob. The following is then repeated, until Bob stops the protocol: Alice sends the factors of N_A to Bob with a 1/8-O.T. scheme.

Each round has 1/8 chance of successfully sending Bob the factors. The sequence of messages that Alice sends and gets from Bob are her receipt: in court, she can claim that she tried to send the factors enough times for Bob to have factored with reasonable probability. She has no way to tell when she has sent the factors, but she can say *what* she sent. Now if Alice stops the protocol, her chance of guessing correctly what she should be sending at the current iteration is just 1/8. Hence, if she stops the protocol, the chance that a judge (who gets to see everything) catches her (stopping before Bob got the factors) is 7/8.

However, to quote a favorite movie starring Bill Murray, *What About Bob?* Suppose Bob stops the protocol before receiving the factors of N_A. Of course, he doesn't know how to decrypt and get m, but what keeps Alice from thinking otherwise? Bob can act as if he got the message, but yet in court a judge could see that Alice never sent the correct roots to Bob. Therefore, in this case, while Alice's receipt "looks valid" to her (since Bob stopped), Bob doesn't get the factors. This asymmetry is patched up in the second protocol.

4 Protocol 2

The second protocol introduces a bit more symmetry into Alice and Bob's interactions. To do it, we need to define a notion of *junk mail*. A junk mail is a string j, that Bob can efficiently detect

to be *not* the mail he wants. (He does not know m, but he can tell that j cannot be m.) However, each j has the case that, under an encryption scheme E_N, Bob cannot tell which of $E_N(m)$ and $E_N(j)$ corresponds to j. So junk has the property that it's easily to distinguish it from legit mail when decrypted, but it cannot be distinguished from legit mail when it is encrypted.

The concept of junk mail can be implemented in many ways. One possible way is for Alice and Bob to agree that the "middle bit" of the binary mail string must be 0, otherwise it is junk. Presumably, a good encryption of a string will make it difficult to tell if the decryption has a middle bit of 0 or not.

Let m be the message to be sent, j be a junk message, p be a large prime, and g be a generator of $\mathbb{Z}_p^*$. Below is a high-level overview of the new protocol.

Repeat:	
Bob → Alice:	1/8-O.T. of the factors of a large number N_B that Bob knows.
Alice:	If Alice knows the factors of N_B, she sets $m^* = m$, else $m^* = j$. *(thus with probability* 7/8*, she sends junk)*
Repeat:	
Alice :	Pick a large random N_A.
Alice → Bob:	N_A and $E_{N_A}(m^*)$, along with a 1/8-O.T. of the factors of N_A.
Bob → Alice:	"read m^*" or "can't read m^*", along with a ZK-proof of this
Until Bob sent "read m^*"	
Alice → Bob:	If $m^* = j$, then Alice proves that Bob's 1/8-O.T. asked her to send junk. If $m^* \neq j$ then **stop**.
End repeat.	

We'll briefly sketch how the judge comes in here. The judge only comes up if Alice claims she sent the mail and Bob denies this. The result of the "judge protocol" will be that either

- Judge substantiates the receipt, and declares it valid, or
- Judge denies the receipt, ruling that m was never sent.

The judge subpoenas all communications. The judge will be aware that he/she got all communications except for possibly the last one. Alice claims she sent a certain last communication, and Bob claims he didn't get it.

The critical last communication to worry about here is when the last communication is: "Alice sends Bob a root r of $x^2 \bmod N$," and $m^* = m$. That is, Alice is trying to send the mail, and she is currently sending the crucial root that will either make or break Bob's reception. Now, Bob can prove to the judge that he knows x, but he cannot prove that he knows the other root. Thus, after learning which root that Bob knows, the judge can easily determine whether or not Bob would get the mail from r. Depending on which, the judge rules in either Alice's or Bob's favor.