U.C. Berkeley — CS276: Cryptography — Lecture 29
Professor David Wagner — 2. May 2006, Scribe: Milos Drezgic

Lecture 29

1 One out of Two Oblivious Transfer

In 1-out-of-2 oblivious transfer, Alice and Bob execute a protocol where, upon completion, Bob learns one of two Alice's secret messages (but not both). Moreover we would like Alice not to know which value Bob has learned. The scheme is based on the RSA and blind signatures.

Alice sends a public key (n, e) to Bob together with encrypted values of her two secret values. Bob chooses a random $r \leftarrow \mathbb{Z}/n\mathbb{Z}$ and selects one of two ciphertexts by blinding it with the random value r. Bob then sends this ciphertext to Alice. Alice decrypts it and returns it to Bob so that he can learn x_b. Alice will perform this protocol only once.

In these scheme, Alice learns nothing; the privacy of Bob's bit b is information theoretically secure. On the other side Bob can not learn anything about x_{1-b}, the message that he did not select, assuming that RSA is secure.

This protocol can also be easily generalized to k out of n oblivious transfer.

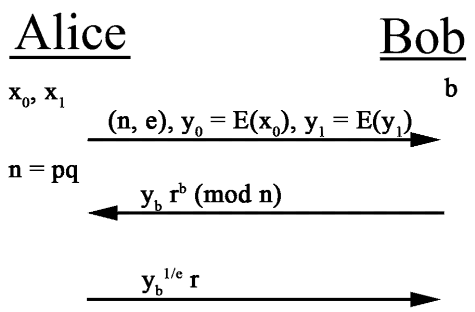

Figure 1: One out of Two Oblivious Transfer

2 Dating protocol

Alice and Bob are very shy but would like nevertheless to find out whether they are interested in each other. By the following protocol they will compute the AND function of their interest without directly revealing their interest. Both Alice and Bob have their secret bits x and y that encodes whether they are interested in the other person; $x = 1$ means that Alice is interested in Bob, $x = 0$ means that Alice is not interested, and so on for y. Upon completion of this protocol each party will be able to learn no more than what is implied by the value of their own input and value of the output of the AND function.

In the honest-but-curious setting, the protocol is as follows. Suppose that Alice's secret input is $x = 0$. Then she sets $(z_0 = 0, z_1 = 0)$ and Bob will select z_0 or z_1 according to the value of his secret input, using one out of two oblivious transfer. At the end Bob sends to Alice the result z_b that he have learned. In the case when $x = 1$ Alice sets $(z_0 = 0, z_1 = 1)$.

This protocol works only if both parties follow the protocol. Yet any protocol secure for honest-but-curious parties can be modified to a protocol that is secure against malicious parties (who might deviate from the protocol) in the standard way described in previous lectures. At each step of the protocol, Alice uses her own secret input and some public input to compute some function of these two values that she will hand to Bob. Bob will do the same thing and than hand control to the Alice. At each step, Alice or Bob can use their transcripts and zero knowledge proofs to convince the other party that they have followed the protocol without cheating.

In the next section, we discuss a generalization of this problem where the function that Alice and Bob want to compute is not the AND function but rather some general circuit.

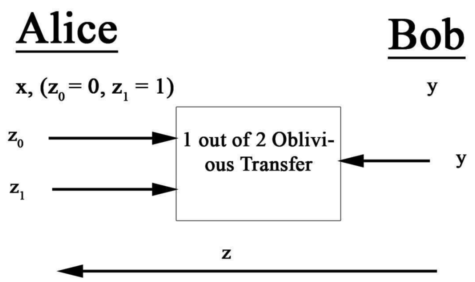

Figure 2: Dating Protocol

3 Secure Circuit Evaluation

Yao's secure circuit evaluation protocol allows Alice and Bob, who have secret inputs x and y respectively, to evaluate an arbitrary polynomial-size Boolean circuit $C(x, y)$ that is known to everyone. The complexity of Yao's scheme is polynomial in the length of x and y and the size of the circuit C.

The construction is as follows. We replace every gate in the circuit with encrypted version of gates. At every step of the protocol Alice or Bob compute some function of their own secret input and the encrypted value received from the other party. If necessary each party will be able to prove to the other party that they performed this step correctly by using zero knowledge proofs. Encrypted gates are formed as follows. For example, the truth table for an encrypted AND gate is as follows:

A	B	Out
x_0	y_0	z_0
x_0	y_1	z_0
x_1	y_0	z_0
x_1	y_1	z_1

Here $x_0, x_1, y_0, y_1, z_0, z_1$ are six random AES keys chosen by Alice. The value x_0 is used to represent that the A-input to the AND gate is fed the bit 0; the value x_1 represents that this input receives the bit 1; and so on. Now Alice computes the following:

$$E_{x_0}(E_{y_0}(z_0 00..00)), \quad E_{x_0}(E_{y_1}(z_0 00..00)), \quad E_{x_1}(E_{y_0}(z_0 00..00)), \quad E_{x_1}(E_{y_1}(z_1 00..00)) \ .$$

Then she permutes these four values and sends them to Bob. If Bob has x_b and $y_{b'}$, Bob can decrypt each of these four values in turn until he finds one that leads to a successful decryption, and in this way Bob will be able to learn $z_{b \ AND \ b'}$ from x_b and $y_{b'}$. However, Bob will not be able to decrypt any of the other three values. Redundancy is added to the

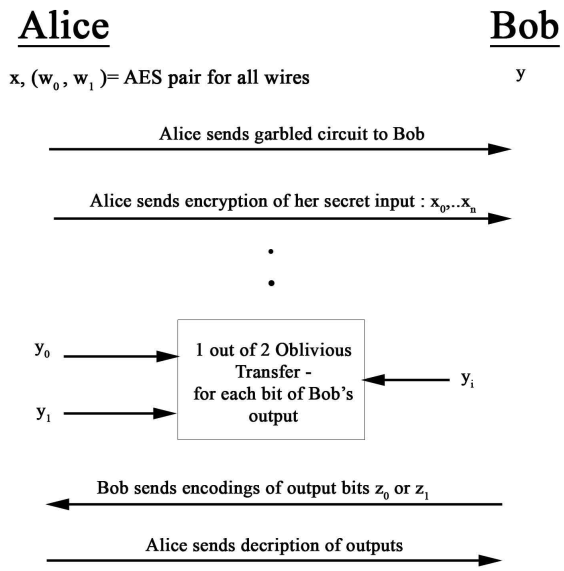

Figure 3: Secure Circuit Evaluation

plaintext before encryption so that Bob will be able to recognize when has succeeded in decrypting these ciphertexts with the proper keys. This is one encrypted AND gate. Of course, we can do something very similar for any other two-input gate. The protocol for the whole circuit is built by encrypting each gate of the circuit C in this way.

Alice has her secret input x and Bob has his secret input y. Alice chooses a pair of keys w_0, w_1 for every wire w in the circuit. Then she constructs a garbled circuit that she sends to Bob. After that Alice sends n keys to Bob, one for each bit of her input x. Now we use oblivious transfer to let Bob learn the appropriate value y_0 or y_1 for each bit of his secret input y. Bob then uses his encrypted gates to compute the Boolean circuit (albeit in encoded form). Bob learns a value z_0 or z_1 for each output wire from the circuit C. He then sends that encoding of z to Alice who outputs the value of z. This protocol is secure in the honest-but-curious model; it only works if all the parties follow the protocol correctly.

This problem is a simplified version of a more general one where Alice has her secret value x and Bob his secret value y, and we would like to ensure that Alice learns $F_1(x_1, x_2)$ and Bob to learns $F_2(x_1, x_2)$. If we had a trusted third party (TTP), then achieving such a goal would be easy. Alice and Bob would send their secret to the TTP; the TTP would compute the function

$$F(x, y) = (F_1(x_1, x_2), F_2(x_1, x_2))$$

and then the TTP would provide $F_1(x_1, x_2)$ to Alice and $F_2(x_1, x_2)$ to Bob. The following theorem on that note is of great interest as it presents a feasibility result showing that this general problem can also be solved without a trusted third party.

Theorem 1 (Secure Multi-party Computation)
If there are n parties, anything we can compute with third party we can do without it as well.

There are some technical caveats. Parties can abort the computation early and decline to participate any further, after the protocol has begun but before it has completed. In that case security of the protocol is not guaranteed, and one of the parties might end up with some advantage. No one can learn anything extra that they aren't supposed to learn by aborting. However, if Alice aborts early, she might be able to learn her own output $F_1(x_1, x_2)$ while preventing Bob from learning his output $F_2(x_1, x_2)$. Thus Alice might be able to get an advantage over Bob by aborting early.

Note that secure circuit evaluation is a special case of the secure multi-party computation problem, where $n = 2$ and the functionality being computed has the form $F(x_1, x_2) = (C(x_1, x_2), C(x_1, x_2))$.

Here are some basis applications for secure multi-party computation:

- Key exchange – Alice and Bob want securely to agree on session key r chosen uniformly at random. Our function F is in this case:

$$F(-, -) = (r, r) \ .$$

  where r is chosen randomly each time F is invoked (thus, F is a randomized function).

- Secure communication – Alice inputs the message x she wants to send, and Bob receives it:

$$F(x, -) = (-, x) \ .$$

- Voting :

$$F(x_1, \ldots, x_n) = (w, \ldots, w)$$

  where w is majority of $(x_1, \ldots, x_n)$.

- Auctions : $F(x_1, \ldots, x_n, -) = (j, \ldots, j, (j, x_k))$, where $j = \text{argmax}_i \ x_i$ and k is the index of second highest bidder.

By the Secure Multi-party Computation Theorem all of these problems have polynomial time protocols. Unfortunately, the protocols so obtained are impractical as for each gate in the circuit there is at least one public key operation. Thus even for modest size circuits cost of this scheme is high, and the resulting protocols are slow. Nonetheless, this theorem is a useful "possibility result" which helps us understand which distributed computation tasks are possible, at least in principle.