CS 282A/MATH 209A: Foundations of Cryptography	© Prof. Rafail Ostrovsky

Lecture 10

Lecture date: Feburary 12, 2024 *Scribes: Yihan Lin*

1 Oblivious Transfer

1.1 Rabin Oblivious Transfer

In an Oblivious Transfer (OT) protocol, a sender transmits one of several pieces of information to a receiver. The receiver obtains only the specific piece it need, while the sender can't learn which piece was transferred.

Rabin oblivious transfer is a kind of formalization of "noisy wire" communication. A Rabin OT machine models the following behavior. The sender(S) sends a bit b into the OT machine. The machine then flips a coin, and receiver(R) has a probability of $\frac{1}{2}$ getting b, $\frac{1}{2}$ getting nothing (notated as # in Fig. 1). S does not know which output R received.

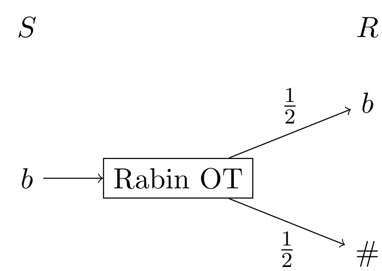

Figure 1: Rabin oblivious transfer

1.2 1-2-Oblivious Transfer (1-2-OT)

In 1-2-OT, sender S sends two bits (b_0, b_1) to the OT machine. Receiver R sends a selected bit s to the OT machine indicating which bit from S it want to get. R will only get the specified bit b_s but not b_{1-s} from the machine, while S knows both bits but has no idea which one R received.

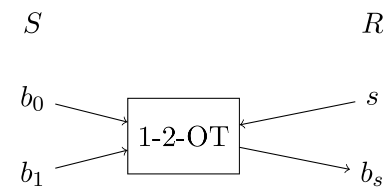

Figure 2: 1-2-oblivious transfer

One Example of 1-2-OT

S has a bit x, R has a bit y, our goal is to calculate $x\, op\, y$ without leaking x and y to each other, where op is a bit operation. We construct a 1-2-OT as below:

1. S and R generate secret bits x and y respectively,
2. Since S doesn't know the value of y, it sends both $x\, op\, 0$ and $x\, op\, 1$ to the OT machine,
3. R sends y to the machine, and receives $x\, op\, y$ according to y.

Here, R only knows the outcome of $x\, op\, y$ without knowing x, S knows all possible outcomes without learning y.

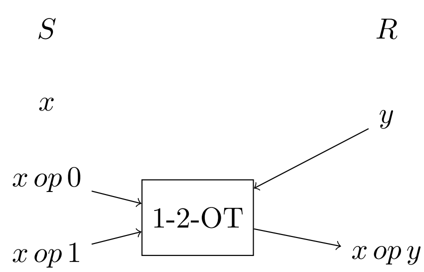

Figure 3: Example of 1-2-oblivious transfer

2 Secret Sharing

Secret Sharing (SS) refers to methods for distributing a secret among a group, in such a way that no individual holds intelligible information about the secret bits, but when a sufficient number of individuals combine their 'shares', the secret can be reconstructed.

Suppose we want to secretly share a bit b with A and B. We can coin-flip a random bit r, and give $\alpha = r$ to A, give $\beta = b \oplus r$ to B. In this case, we can reconstruct b by XOR α and

β. For A and B, the bit they get looks totally random, which means both of them can't figure out b only with their piece of share.

2.1 A Solution for Secret Sharing Boolean Circuit Computation

Boolean circuit is a circuit which turns inputs into boolean bit. It's structure is shown as Fig. 4.

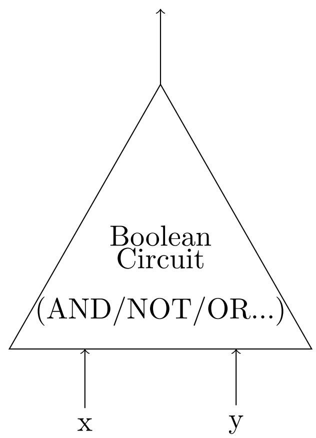

Figure 4: Boolean circuit

Suppose there are two honest-but-curious players, A and B, each has a portion of the inputs to a boolean circuit and wish to determine the output without revealing their inputs. They can do this using secret sharing.

For **XOR** circuit, let A has a_1, a_2, B has b_1, b_2, they want to compute $F = (a_1 \oplus b_1) \oplus (a_2 \oplus b_2)$. Because of the commutative and associative property of XOR, we can safely conclude that $(a_1 \oplus b_1) \oplus (a_2 \oplus b_2) = (a_1 \oplus a_2) \oplus (b_1 \oplus b_2)$. Therefore, A and B can xor their pieces of bits first, and xor the result of A and B to generate the final output. Since xor of two bits can be seen as a coin-flip, and one player doesn't know the composition of the two bits of the other, therefore the output of $a_1 \oplus a_2$ $(b_1 \oplus b_2)$ is totally random to B (A).Thus, they can get the final output of XOR without leaking information to the other player.

For **AND** circuit, things are a little bit more complex. Let A has a_1, a_2, B has b_1, b_2, and they want to compute $F = (a_1 \oplus b_1) \wedge (a_2 \oplus b_2)$. First we unfold this formula:

$$(a_1 \oplus b_1) \wedge (a_2 \oplus b_2) = (a_1 \wedge a_2) \oplus (a_1 \wedge b_2) \oplus (a_2 \wedge b_1) \oplus (b_1 \wedge b_2)$$

where $(a_1 \wedge a_2)$ can be directly calculated by A and $(b_1 \wedge b_2)$ can be calculated by B. Next we compute $(a_1 \wedge b_2)$ and $(a_2 \wedge b_1)$ with 1-2-oblivious transfer.

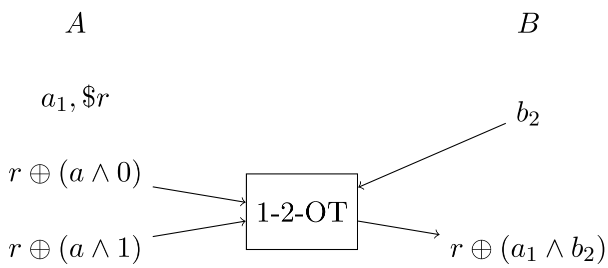

Figure 5: Computation of $a_1 \wedge b_2$ for AND circuit

An intuition solution to compute $a_1 \wedge b_2$ is, as what we did in 1-2-OT part, A sends $(a_1 \wedge 0)$ and $(a_1 \wedge 1)$ to the OT machine, B sends b_2 to the machine, and B receives $(a_1 \wedge b_2)$. However, there is a potential risk of leaking a_1 to B. If $a_1 \wedge b_2 = 1$, then there is no doubt that $a_1 = 1$; or if $a_1 \wedge b_2 = 0$ and $b_2 = 1$, then B will know $a_1 = 0$.

Therefore, to ensure the secret sharing, we add a random bit r to hide a_1. Specifically, A chooses a random bit r, and sends $r \oplus (a_1 \wedge 0)$ and $r \oplus (a_1 \wedge 1)$ to the OT machine, and B receives $r \oplus (a_1 \wedge b_2)$. Since r is totally unknown to B, for any outcome it receives, the probability of $a_1 = 1$ and $a_1 = 0$ is the same for B, and thus we secure the sharing process. To eliminate the influence of r in the final F, A will do xor for $a_1 \wedge a_2$. Since for any x, $x \oplus x = 0$, therefore $(a_1 \wedge a_2) \oplus (a_1 \wedge b_2) = (r \oplus (a_1 \wedge a_2)) \oplus (r \oplus (a_1 \wedge b_2))$. The process of calculating $a_2 \wedge b_1$ is the same. Thus, the final formula will be like this:

$$F = (r_1 \oplus r_2 \oplus (a_1 \wedge a_2)) \oplus (r_1 \oplus (a_1 \wedge b_2)) \oplus (r_2 \oplus (a_2 \wedge b_1)) \oplus (b_1 \wedge b_2)$$

, where r_1 and r_2 are random bits picked for calculating $a_1 \wedge b_2$ and $a_2 \wedge b_1$ respectively.

Some problems for thought:

1. For n $(n > 1)$ non-collusion players, at least how many random bits are needed to compute the AND circuit of all players without leaking any information, i.e., $x_1 \wedge x_2 \wedge ... \wedge x_n$, where x_i is the secret bit of Player i? Currently, researchers already proved that 2 random bits are necessary, and 8 bits are sufficient.

2. If there are more than two players, what will happen if players collude?

3 Construct 1-2-OT with Trapdoor One-Way Permutation Family

Suppose S has two message b_0, b_1 to be transfer, we can construct a 1-2-OT with a trapdoor one-way permutation family through the following process:

1. S picks a trapdoor one-way permutation $f : \{0,1\}^n \rightarrow \{0,1\}^n$, p is it's hard core bit, and S knows its trapdoor while R doesn't

2. S sends f and p to R

3. R randomly picks an x_s with selected bit s, and compute $y_s = f(x_s)$. Then R randomly picks a y_{1-s} and sends y_0, y_1 (i.e. y_s, y_{1-s}) to S

4. S compute $x_0 = f^{-1}(y_0)$, $x_1 = f^{-1}(y_1)$ using the trapdoor, and sends $b_0 \oplus < x_0, p >$ and $b_1 \oplus < x_1, p >$ to the OT machine

5. R sends x_s to the OT machine and receives $b_s \oplus < x_s, p >$, and then computes b_s using x_s and p

Since R knows x_s and p, it can compute b_s in polynimial time. However, for b_{1-s}, R doesn't know x_{1-s} because f is a one-way permutation and R doesn't have the trapdoor. As a result, R cannot open b_{1-s}. In this way, we construct a 1-2-oblivious transfer with a trapdoor one-way permutation.

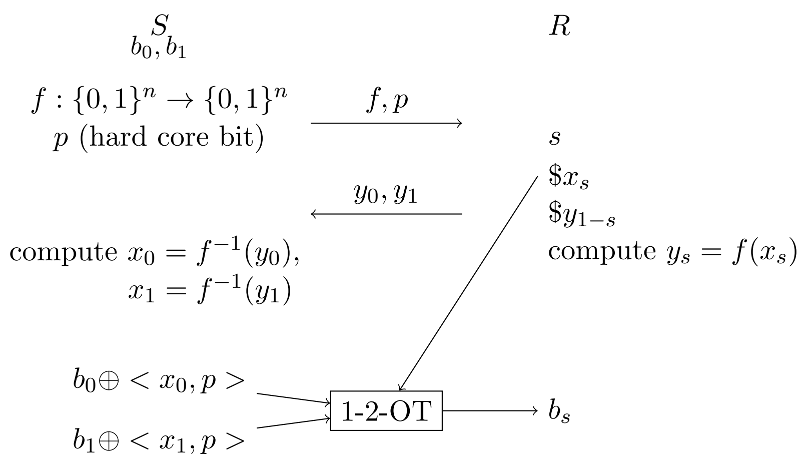

Figure 6: 1-2-OT with Trapdoor One-Way Permutation