1 Lecture 11

1.1 Oblivious transfer

Problem:

- Alice sends $b$ to Bob, but only with $Pr = \frac{1}{2}$.
- Bob receives $b$ with $Pr = \frac{1}{2}$ with $Pr = \frac{1}{2}$ he receives $\sharp$ junk.
- Alice does not learn what Bob received.

Rabins Obvious Transfer, 1/2 - OT [Rabin 1981]

1. A picks large $p, q$ (exp.: $p == q == 3(mod4)$).
2. A sends $N$ to B
3. B picks $x \in R Z_N$ computes $t = x^2 \mod N$ and sends $\bar{s} = \sqrt{t}$ to A
4. B calculates $N = p \cdot q = (x + y) \cdot (x - y) = x^2 - y^2$
5. A chooses $s \in R S$ where $S = [x, -x, y, -y]$ and sends $s$ to B
6. If $s = \pm x$ then B learns nothing. If $s = \pm y$ then B can compute $p, q$.

One-out-of-two oblivious transfer (1-2-OT)

Goal:
Alice sends 2 bits to OT-box.
Bob picks $i$ which bit he wants to receive.
OT outputs $b_i$ to Bob and discards $b_{1-i}$.

Application:
Private Information retrieval.
Claim: 1-2-Ot and 1/2-OT can be transform in each other.
1-2-Ot $\Rightarrow$ 1/2-OT
1. A sends $b$ to B with $Pr = \frac{1}{2}$.
2. A chooses $r, l \in R 0, 1$
3. A inputs to 1-2-OT : If $l = 0 : (b_0, b_1) = (b, r)$. If $l = 1 : (b_0, b_1) = (r, b)$.
4. B selects \( i \in \{0, 1\} \) and sends \( i \) to OT. Note OT output \( b \) iff \( i = l \).

5. A sends \( l \) to B over standard channel.

6. B compares \( l = i \). B knows he learned \( b \), else B knows he learned \( \sharp \).

\[ \frac{1}{2}-\text{OT} \Rightarrow 1-2-\text{OT} \] [Crepeau?]

1. A and B agree on security parameters \( n, m \) where \( n \approx 3m \).

2. A chooses \( n \) random bits \( r_1 \ldots r_n \).

3. A and B run \( \frac{1}{2} \)-OT for each \( r_i \). Result: B knows \( \approx \frac{1}{2} \) of \( r_i \), but A does not know which ones.

4. B picks \( U = (i_1, \ldots, i_m), V = (i_{m+1}, \ldots, i_{2m}) \) with \( U \cap V = \emptyset \). B knows \( r_i \forall i \in U \).

5. Bob sends : \((x, y) = (U, V)\) if he wants to learn \( b_0 \) or \((x, y) = (V, U)\) if he wants to learn \( b_1 \).

6. A computes : 
\[
z_0 = \bigoplus x \in X r_x \quad \text{and} \quad z_1 = \bigoplus y \in Y r_y
\]
and sends \((w_1, w_2) = (b_0 \oplus z_0, b_1 \oplus z_1)\) to B.

7. B can use the bits from \( U \) to compute \( z_k = \bigoplus i \in U r_i \) and finds \( b_k = z_k \oplus w_k \)

q.e.d.

- There are protocols for \( k \) out of \( n \) OT.
- With OT we can construct secure MPC protocols that can realize (almost) any function.
- OT + digital cash can be used for completely anonymous e-payment systems.
  Digital cash: protects the identity of the buyer.
  OT: prevent the seller from learning what was purchased.

Millionaires Problem MPC-protocol using OT.

\( f(a, b) \) is a poly size boolean circuit, consisting of AND and XOR gates.

Construct a protocol, so that at any gate A (holding \( x \)'s) and B (holding \( y \)'s) will have a share of the output.

Each wire in a boolean circuit is represented by one bit \( b_i = x_i \oplus y_i \).

**Input:**

\( T^A \) (bits of Alice)
\( T^B \) (bits of Bob)

**represent \( f(a, b) : x^A \oplus x^B \) : \( x^A \in T^A, x^B \in T^B \)**

**Sharing phase:**

1. A generates a random string \( a^B \) and computes \( a^A = a \oplus a^B \).
2. A sends \( a^B \) to B.
3. B generates a random string \( b^A \) computes \( b^B = b \oplus b^A \).
4. B sends \( b^A \) to A.
Computation phase:

XOR-Gate
\[ x \oplus y = (x^A \oplus x^B) \oplus (y^A \oplus y^B) = (x^A \oplus y^A) \oplus (x^B \oplus y^B) \]

1. A computes \( x^A \oplus y^A \).
2. B computes \( x^B \oplus y^B \).

And-Gate
\[ x \cdot y = (x^A \oplus x^B) \cdot (y^A \oplus y^B) = (x^A \cdot y^A) \oplus (x^A \cdot y^B) \oplus (x^B \cdot y^A) \oplus (x^B \cdot y^B) = A \oplus ? \oplus ? \oplus B \]

Let M be a 1-2-OT box.

Case: \((x^A \cdot y^B)\)

1. A generates \( r^A \in_R 0, 1 \)
2. A’s input to M: \((b_0, b_1) = ((x^A \cdot 0) \oplus r^A, (x^A \cdot 1) \oplus r^A)\)
3. B inputs \( y^B \) to M.
4. M outputs \((x^A \cdot y^B) \oplus r^A \) to B. B stores this as \( w^B \).

Note that \( x^A = r^A \oplus w^B \). But B does not learn anything about \( x^A \). A does not learn \( y^B \).

The case \( x^B \cdot y^A \) is similar. Bob provides inputs to OT.

Finally A and B assemble shares:

1. A computes \((x \cdot y)^A = (x^A \cdot y^A) \oplus r^A \oplus w^A\)
2. B computes \((x \cdot y)^B = (x^B \cdot y^B) \oplus r^B \oplus w^B\)

Reconstruction phase:
A und B compine ther shares and learn the output of \( f(a, b) \).