Lecture 39: GMW Protocol

Recall

- Last lecture we saw that we can securely compute any function using oblivious transfer (which can be constructed from the RSA assumption)
- However, the protocol is efficient only when the function has constant size

Today's Lecture: Summary

- Today we shall learn the Goldreich-Micali-Wigderson (GMW) Protocol to securely compute any function that can be efficiently computed

Recall: Additive Secret Sharing Scheme I

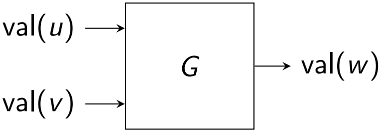
- Let $(G, \circ)$ be a group
- For any $s \in G$, we pick $s_A \xleftarrow{\$} G$ and define $s_B = \mathrm{inv}(s_A) \circ s$
- Note that just given s_A, the secret s is perfectly hidden
- Note that just given s_B, the secret s is perfectly hidden
- But, given both s_A and s_B we can reconstruct s
- This secret sharing scheme shall be referred to as the "additive secret sharing scheme" (you have already seen this scheme in the midterm)

An Example.

- Consider the group $(\{0,1\}, \oplus)$, where $\oplus$ is the bit-xor
- Then the additive secret shares of a secret bit s is $s_A \xleftarrow{\$} \{0,1\}$ and $s_B = s_A \oplus s$
- Note that the secret $s = s_A \oplus s_B$

Basic Step I

- Suppose we have two wires u and v
- The values of these two wires in a circuit be $\mathrm{val}(u)$ and $\mathrm{val}(v)$
- Suppose the secret shares of $\mathrm{val}(u)$ be $\mathrm{val}(u)_A$ and $\mathrm{val}(u)_B$
- Suppose the secret shares of $\mathrm{val}(v)$ be $\mathrm{val}(v)_A$ and $\mathrm{val}(v)_B$
- Let G be a gate where wire u and v are inputs and wire w is the output. For example, the gate G can be the AND-gate, NAND-gate, XOR-gate, etc.
- So, the value of the wire w is $\mathrm{val}(w) = G(\mathrm{val}(u), \mathrm{val}(v))$



Suppose

- Alice already has $\mathsf{val}(u)_A$ and $\mathsf{val}(v)_A$
- Bob already has $\mathsf{val}(u)_B$ and $\mathsf{val}(v)_B$
- Alice samples $\mathsf{val}(w)_A \xleftarrow{\$} \{0,1\}$

What is the share $\mathsf{val}(w)_B$?

$$\mathsf{val}(w)_B = \mathsf{val}(w)_A \oplus G\left(\overbrace{\mathsf{val}(u)_A \oplus \mathsf{val}(u)_B}^{\mathsf{val}(u)}, \overbrace{\mathsf{val}(v)_A \oplus \mathsf{val}(v)_B}^{\mathsf{val}(v)}\right)$$

So, the value $\mathsf{val}(w)_B$ is a function of 3-bit input from Alice and 2-bit input from Bob, i.e., it is a function of constant size. Now, we can efficiently and securely compute this function!

The GMW Protocol I

- Suppose Alice has private input $x = (x_1, x_2, \ldots, x_n)$
- Suppose Bob has private input $y = (y_1, y_2, \ldots, y_n)$
- Alice and Bob are interested in computing a function that is described by a circuit C. The output of the circuit is $z = C(x, y)$

Base Case. Additively secret sharing the input wires

- Suppose the wires $\{1, 2, \ldots, n\}$ correspond to Alice's input $(x_1, x_2, \ldots, x_n)$, respectively. Alice picks random $\mathsf{val}(i)_A \xleftarrow{\$} \{0, 1\}$, for $i \in \{1, 2, \ldots, n\}$. Alice sends $\mathsf{val}(i)_B = x_i \oplus \mathsf{val}(i)_A$ to Bob.
- Suppose the wires $\{n+1, n+2, \ldots, 2n\}$ correspond to Bob's input $(y_1, y_2, \ldots, y_n)$, respectively. Bob picks random $\mathsf{val}(n+i)_B \xleftarrow{\$} \{0, 1\}$, for $i \in \{1, 2, \ldots, n\}$. Bob sends $\mathsf{val}(n+i)_A = y_i \oplus \mathsf{val}(n+i)_B$ to Alice.

The GMW Protocol III

Inductively Computing Internal Wires. Suppose Alice and Bob want to securely compute the output of a gate G whose input wires are u and v, and the output wire is w. Assume, by induction hypothesis, that $\mathsf{val}(u)_A$ and $\mathsf{val}(v)_A$ are with Alice, and $\mathsf{val}(u)_B$ and $\mathsf{val}(v)_B$ are with Bob.

- First, Alice picks $\mathsf{val}(w)_A \xleftarrow{\$} \{0,1\}$
- Next, Alice and Bob securely compute the function that outputs the following value to Bob (we already know how to do this)

$$\mathsf{val}(w)_B = \mathsf{val}(w)_A \oplus G\left(\overbrace{\mathsf{val}(u)_A \oplus \mathsf{val}(u)_B}^{\mathsf{val}(u)}, \overbrace{\mathsf{val}(v)_A \oplus \mathsf{val}(v)_B}^{\mathsf{val}(v)}\right)$$

Repeat this for all gates.

The GMW Protocol IV

Finalizing the Output. Suppose the output wires are $\{s+1, s+2, \ldots, s+m\}$. Alice has the values $\mathrm{val}(s+i)_A$ and Bob has the values $\mathrm{val}(s+i)_B$, for $i \in \{1, 2, \ldots, m\}$.

- Alice and Bob exchange the values $\mathrm{val}(s+i)_A$ and $\mathrm{val}(s+i)_B$, for $i \in \{1, 2, \ldots, m\}$, to reconstruct $\mathrm{val}(s+i)$ This is the output $z = (\mathrm{val}(s+1), \mathrm{val}(s+2) \ldots, \mathrm{val}(s+m))$

So, we can securely evaluate any circuit in time proportional to its size!

An Example I

Consider the following example understand how the GMW-protocol can be helpful

- Consider the example of Dutch flower auction
- Suppose Alice has an n-bit bid that is even, and Bob has an n-bit bid that is odd
- So, each party has 2^{n-1} possible inputs (bids)
- If we securely evaluate this function using the approach introduced in the previous class, then we need 2^n rounds, which is inefficient

An Example II

How do we securely perform this task using the GMW-protocol?

- Write an efficient circuit that evaluates the maximum of the two inputs $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ (What is the smallest circuit that you can design?)
- Use RSA-based m-choose-1 OT protocol to securely compute this circuit using the GMW-protocol

An Example III

What are the tradeoffs between these two protocols?

- The first protocol is perfectly secure, while the second protocol is secure only against computationally bounded parties
- The first protocol is inefficient, while the second protocol is efficient