CS 65500
Advanced Cryptography

Lecture 12: Semi-Honest BGW - II
& Multiparty GMW

Instructor: Aarushi Goel

Spring 2025

Agenda

→ Security Proof for the semi-honest BGW Protocol $(t < n/2)$

→ Semi-honest Multiparty GMW Protocol $(t < n)$

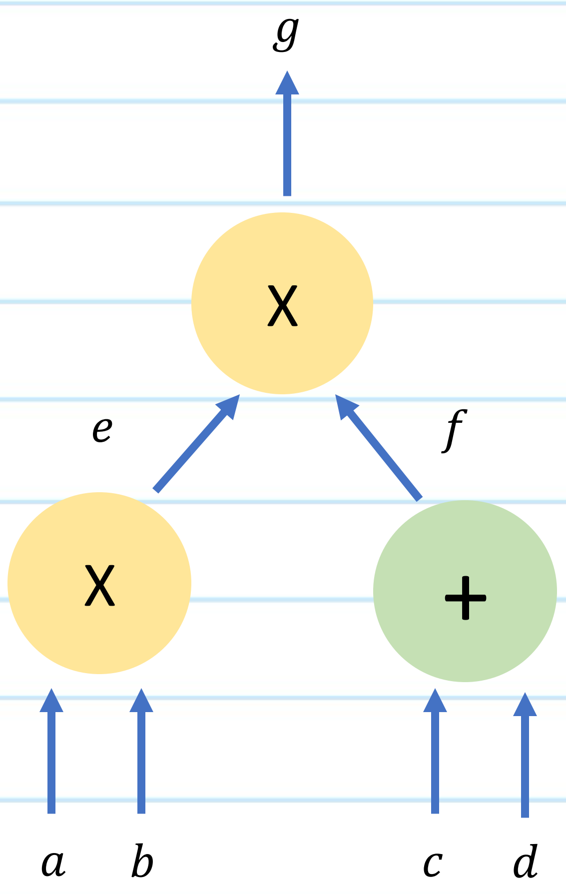
Midterm is on March 6

(Syllabus: everything covered until Feb 27)

Review class on March 4

(Please bring any questions you may have to class)

BGW Protocol : Overview



* Input Sharing: Parties start by computing and sending (t,n) threshold shares of their respective inputs.

* Circuit Evaluation: gate-by-gate evaluation over secret-shared values. In other words, compute secret-share of all intermediate wire values one by one

* Output Reconstruction: All parties reveal their shares of the output wire values to each other & then reconstruct.

BGW Protocol

Multiplication

given shares of a, b, the parties need to compute shares of $e = a \times b$

$$[e]_{2t} = [a]_t \times [b]_t$$

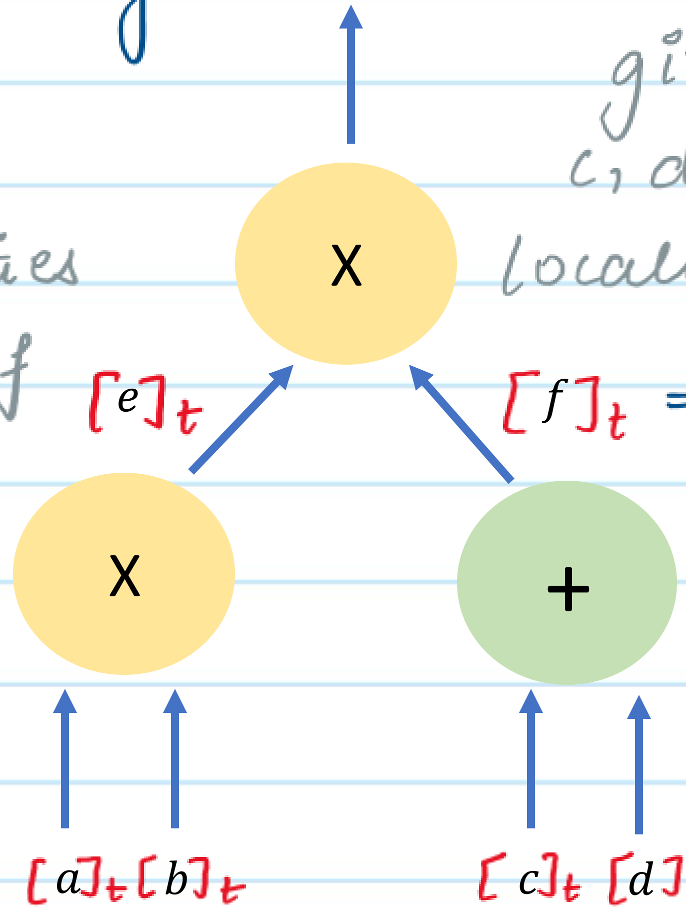
$$[[e]_{2t}]_t \xleftarrow{\text{share}} [e]_{2t}$$

exchange $[[e]_{2t}]_t$

$$[e]_t \xleftarrow{\text{reconstruct}} [[e]_{2t}]_t$$

Reconstruct Output.

$g \leftarrow [g]_t$



Input sharing using a (t, n) Shamir-sharing

Addition

given shares of c, d, parties can locally compute:

$$[f]_t = [c]_t + [d]_t$$

BGW Protocol: Multiplication Gates.

Given: $[a]_t$, $[b]_t$ **To compute:** $[e = a \cdot b]_t$

$\forall i \in [n]$, Each party P_i does the following:

1. locally computes $\bar{e}_i = a_i \times b_i$
2. Computes (t, n) Shamir Sharing of $\bar{e}_i$
   $(\bar{e}_{i1}, \ldots, \bar{e}_{in}) \leftarrow \text{Share}(\bar{e}_i)$
3. $\forall j \in [n]$, Send $\bar{e}_{ij}$ to Party P_j
4. Let $\lambda_1, \ldots \lambda_n$ be Lagrange coefficients such that $ab = \lambda_1 \bar{e}_1 + \lambda_2 \bar{e}_2 + \cdots \lambda_n \bar{e}_n$
   Party P_i computes $ab_i = \lambda_1 \bar{e}_{1i} + \lambda_2 \bar{e}_{2i} + \cdots + \lambda_n \bar{e}_{ni}$

1. $[e]_{2t} = [a]_t \times [b]_t$
2. $[e]_{2t} \rightarrow [[e]_{2t}]_t$
3. exchange shares of shares
4. $[[e]_{2t}]_t \rightarrow [e]_t$

BGW Protocol : Security

What do we want to Prove?

→ BGW is an n-party protocol for securely computing f in the presence of a semi-honest adversary who corrupts at most $t < n/2$ parties.

→ $\exists$ a simulator, s.t. for any t-sized subset $C \subset [n]$ of corrupt parties and $\forall$ $x_1, \ldots, x_n$, it can simulate a view using inputs of the corrupt parties & output of f that is indistinguishable from the adversary's view in the real protocol.

→ for simplicity, let's consider an adv who corrupts exactly $n/2 - 1$ parties.

Simulator: $S_c(\{x_i\}_{i\in C}, f(x_1,\ldots,x_n))$

1 Input Sharing: $\forall i \in C$, compute $\text{Share}(x_i) \to x_{i1},\ldots, x_{in}$
$\forall i \notin C$, compute $\text{Share}(0) \to x_{i1},\ldots, x_{in}$

Output $\{x_{ij}\}_{i\in C, j\in[n]}$, $\{x_{ij}\}_{i\notin C, j\in C}$

2. Circuit Evaluation: $\forall i \in C$:

→ Addition ($f = c+d$): compute $f_i = c_i + d_i$

→ Multiplication ($e = a\times b$): compute $\bar{e}_i = a_i \times b_i$

compute $\text{Share}(\bar{e}_i)$

$\forall j \notin C$, compute $\bar{e}_{j,1},\ldots, \bar{e}_{j,n} \leftarrow \text{Share}(0)$

compute

$e_i = \lambda_1 \bar{e}_{1i} + \cdots + \lambda_n \bar{e}_{ni}$

Output $\{\bar{e}_{ij}\}_{i\in C, j\in[n]}$, $\{\bar{e}_{ij}\}_{i\notin C, j\in C}$, $\{e_i\}_{i\in C}$

3. Output Reconstruction: For each output wire y, let $\{y_i\}_{i\in C}$ be the shares that the simulator computed during circuit eval.

Interpolate $(y, \{y_i\}_C)$ to reconstruct a polynomial $p(x)$ such that $p(0) = y$.

$\forall j \in [n] \setminus C$: $\quad y_j = y(\alpha_j)$.

output $\{y_i\}_{i\in[n]}$

Indistinguishability Argument

We now need to show that the output of this simulator is indistinguishable from the adversary's view in the real protocol.

→ Input Sharing: The only difference between the real protocol and what the simulator does is how the shares of honest parties' inputs are computed. In the real protocol these shares are computed using honest parties' real inputs, while the simulator simply computes shares of 0. But since the adversary only sees t-shares, indistinguishability follows privacy of (t, n) Shamir secret sharing scheme — any subset of t-shares corresponding to two different secrets are perfectly indistinguishable

→ Circuit Evaluation:

* Addition: There is no difference between what the simulator does and what happens in the real protocol

* Multiplication: The only difference between what the simulator does and what happens in the real protocol is how the shares of $\bar{c}_j$ are computed for each $j \notin C$. Indistinguishability between the view of the adversary in the real protocol & what the simulator computes follows from privacy of the secret sharing scheme - similar to that in the input sharing phase.

→ Output Reconstruction: In the real protocol, the adversary gets all honest parties' shares for the output wires. These shares are such that together with corrupt parties' shares, they form a valid (t,n) Shamir secret sharing of the output.

The simulator instead uses the output and shares of corrupt parties to interpolate a unique degree-t polynomial. It then evaluates this polynomial on appropriate points to derive shares of the honest parties. Together all these shares constitute a valid (t,n) Shamir secret sharing of the output.

Multiparty Semi-Honest GMW Protocol

→ n-parties want to jointly compute a binary function $f:\{0,1\}^n \to \{0,1\}^k$. – represented as a binary circuit with NOT and AND gates.

→ We will assume for simplicity that $\forall i \in [n]$, Party P_i has input bit $x_i \in \{0,1\}$. and all parties learn the output $y_1 \ldots y_k \leftarrow f(x_1, \ldots, x_n)$

→ This protocol is secure against a semi-honest adversary corrupting $t < n$ parties

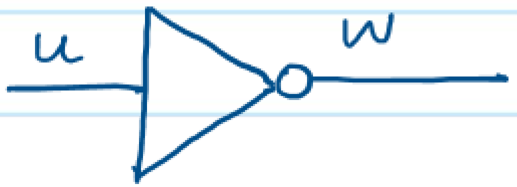
→ **Building Blocks:** Additive secret sharing & 1-out-of-2 OT.

Multiparty Semi-Honest GMW Protocol

* Input-Sharing: $\forall i \in [n]$, party P_i computes additive shares $x_{i1}, \ldots x_{in}$ of x_i. $\forall j \in [n]$, Party P_i sends x_{ij} to party P_j.

* Circuit-Evaluation:

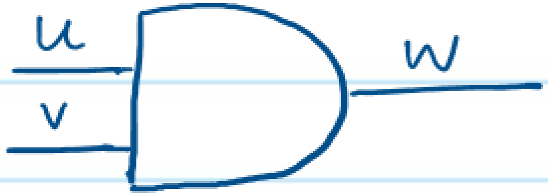
NOT Gate



Gives shares of u, parties compute their shares of w as follows:

- Party P_1, computes $w_1 = u_1 \oplus 1$
- $\forall i \in [n] \setminus \{1\}$, Party P_i computes $w_i = u_i$

AND gate



given shares of u and v, parties want to compute shares $w_1, \ldots, w_n$ of w, such that

$$w_1 \oplus \cdots \oplus w_n = u \cdot v$$
$$= (u_1 \oplus \cdots \oplus u_n) \cdot (v_1 \oplus \cdots \oplus v_n)$$
$$= \bigoplus_{i \in [n]} u_i v_i \oplus \bigoplus_{\substack{i,j \in [n] \\ i \neq j}} u_i v_j$$

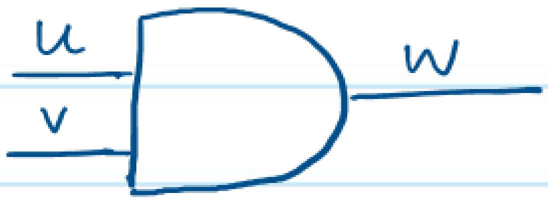
Party P_i can compute itself

Parties P_i & P_j must collaborate to compute this.

How can P_i & P_j collaborate to compute additive shares of $u_i v_j$?

→ use 1 out of 2 Oblivious transfer!

AND gate



$\forall i \in [n]$ party P_i does the following:

- It holds u_i, v_j.
- $\forall j \in [n] \setminus \{i\}$: 1. Sample $r_{ij} \xleftarrow{\$} \{0,1\}$
  2. Participate in a 1-out-of-2 OT protocol with P_j, where P_i acts as sender & P_j is the receiver.
  P_i has inputs $a_{ij0} = r_{ij} \oplus (u_i \cdot 0)$ & $a_{ij1} = r_{ij} \oplus (u_i \cdot 1)$
  P_j has input $b = v_j$.
  At the end, P_j gets a_{ijb}
- $w_i = u_i \cdot v_i \oplus \bigoplus_{j \in [n] \setminus \{i\}} (r_{ij} \oplus a_{jiv_i})$

* Output Reconstruction:

For all output wires $y_1, \ldots, y_k$, $\forall i \in [n]$, party P_i:

- sends its shares $y_{1i}, \ldots, y_{ki}$ to all other parties
- reconstructs: $y_1 = y_{11} \oplus \cdots \oplus y_{1n}$

$$y_2 = y_{21} \oplus \cdots \oplus y_{2n}$$

$$\vdots$$

$$y_k = y_{k1} \oplus \cdots \oplus y_{kn}$$

* Exercise: Think about why this protocol is secure.