Secure Multi-Party Computation

Suppose a set of n parties denoted by $P_1, \ldots, P_n$ with private inputs $x_1, \ldots, x_n$ want to compute $f(x_1, \ldots, x_n)$ without rebealing anything about each of their inputs

- This is the goal of MPC

Examples:

- Collaborating on financial / healthcare / market research without rebealing each of their inputs
- Machine learning without sharing raw data
- Secure auctions

We assume that every pair of parties has a private and authenticated channel

- We assume the network is synchronous
  - Protocols proceeds in rounds, and at the end of each round, all messages sent were received

Defining **security**:

- Take 1:
  - By the end of the protocol, no party learns any information about the others' inputs
  - This is impossible since everyone learns $f(x_1, \ldots, x_n)$
- Take 2:
  - By the end of the protocol, no party learns any information about the others' inputs besides the output f
  - This doesn't provide enough security because we want even more
    - Let's say we were implementing an auction
    - We don't want it to be somehow possible for someone else to be able to guarantee outbid someone by 1 even if they don't know that person's bid
- Take 3:
  - The MPC protocol should be as secure as an idealized world where everyone sends their result to a trusted third party and they compute the output
  - We formalize this below

Security

An n party protocol securely computes a function in the presence of at most t corruptions if for every PPT adversary $\mathcal{A}$ controlling at most t parties, there exists a PPT simulator $\mathcal{S}$ controlling the same parties in the ideal world such that for any inputs, $\mathrm{Real}_{\mathcal{A}}$ and $\mathrm{Ideal}_{\mathcal{S}}$ are indistinguishable

- $\mathrm{Real}_{\mathcal{A}}$ refers to the output of all parties after running the protocol where the adversarial parties output whatever they want
- $\mathrm{Ideal}_{\mathcal{S}}$ is the same story but they all hand their inputs to a secure third-party

There are two main types of adversaries:

- Malicious: the adversary controlling the parties completely controls them and sends whatever the adversary wants
- Honest-but-curious: an adversary controlling the parties learns their inputs and all messages sent / received, but cannot modify the messages

BGW Protocol

The BGW protocol consists of three phases:

1. Secret Sharing of Inputs
2. Computation on Shares
3. Reconstruction

Secret Sharing of Inputs

Fix a prime $p > n$. Each party then uses Shamir's secret sharing scheme to share their input across $t+1$ parties:

- Select uniformly at random t coefficients $a_1, \ldots, a_t$ from $\mathrm{GF}[p]$ and let:

$$g_b(x) = b + a_1 x + \cdots + a_t x^t$$

- Send $g_b(j)$ to party j

So now, any subset of $t+1$ parties can reconstruct any other parties' input by working together

Computation on Shares

We think of the function f as some combination of addition and multiplication gates

For addition gates:

- Suppose the inputs to a gate are a and b
- These are shared via polynomials g_a and g_b
- We need to compute the i th share of g_{a+b}
- We note that the definition of g_{a+b} is a random polynomial such that $g_{a+b}(0) = a + b$
  - We note that $g_{a+b} = g_a + g_b$ works
  - Therefore we can compute the i th share of g_{a+b} by $g_{a+b}(i) = g_a(i) + g_b(i)$

For multiplication by a constant:

- We can also just use $g_{ca} = c * g_a$

For multiplication gates, it is more complicated:

- We cannot just use $g_{ab} = g_a g_b$
  - This is degree $2t$!
  - We would need $2t+1$ parties to reconstruct, and every element is not random because it is the product of two polynomials
- We do two steps to fix this:
  - Degree reduction
  - Rerandomization

Degree Reduction: Each party reduces it share of a degree $2t$ polynomial to a share of some degree t polynomial

> **Theorem**
>
> There exists a constant matrix $A \in \mathrm{GF}[p]^{n \times n}$ such that for every degree $2t$ polynomial $g(x) = h_0 + h_1 x + \cdots + h_{2t} x^{2t}$, there is a degree t truncated polynomial $\hat{g}(t) = h_0 + h_1 x + \cdots + h_t x^t$ such that:
>
> $$(\hat{g}(1), \ldots, \hat{g}(n))^T = A \cdot (g(1), \ldots, g(n))$$

With this A, we can represent the multiplication instead as:

- Compute $A g_{ab}(i) = A(g_a(i) g_b(i))$
- Share this value around

Rerandomization: The problem is this polynomial might not be very random anymore! So we rerandomize the degree $2t$ polynomial before truncating it

- Each party P_i secret shares the value 0 using a $2t+1$ secret sharing scheme
  - Namely, each party P_i finds a random $2t$ polynomial g_i such that $g_i(0) = 0$
  - It then sends the values of $g_i(j)$ to party P_j
- Each party P_i then sums up all of the values it receives and adds it to its current share of g_{ab}
  - Note that this in the end will still evaluate to ab on 0 since this is the i th share of the polynomial $g_a b + g_1 + g_2 + \cdots + g_n$
- This is the polynomial that is then passed into degree reduction

Reconstruction

The parties then each send their share of the output to whomever should receive said output

- These parties then reconstruct

Security

Intuitively, each party holds random shares and an adversary cannot reconstruct any input with only t controlled parties

- Note that this doesn't rely on cryptography!
- It is information theoreticlly secure

What happens if we have malicious parties who send random junk?

- Recall from Shamir's secret sharing scheme that we can reconstruct even if $\frac{n-t-1}{2}$ parties are corrupted
- This means that as long as $t < n/3$, then we can reconstruct!
- If the adversary only sends junk on the reconstruction phases, then we are good
- If the adversary sends junk on the secret sharing phase, then it is no longer secure
    - I.e. if it sends things that do not correspond to a degree t function
    - This is fixable if we use a verified secret sharing scheme
        - A dealer "proves" that it send shares corresponding to a degree t polynomial (or degree $2t$ in rerandomization)
        - If we use one of these, then we are fully secure against $t < n/3$ adversaries