1 Lecture 16: MPC with Malicious Players

In general, when we do MPC with more than 2 players, we assume that the players can all communicate via pairwise channels. We still want all the players to learn the final output of the function without revealing anything about their own secret inputs (besides what is implied by the output) to any of the other players.

Honest-but-Curious is the most basic security, where we model wiretaps. A corrupt adversary will still follow the rules of the game, but may listen to any messages between any other pairs of players, and may see other players' secret inputs.

Malicious security is security that takes account for corrupt players behaving arbitrarily. In this setup, players will commit to their inputs so that computation can go on even if they go offline or deviate.

Finally,**t-resilience** is a concept where, even if an adversary corrupts up to t players, they still cannot learn the secret inputs of the other players.

As usual, we have **static corruptions**, which are players who are corrupt at the beginning of the protocol, and **adaptive corruptions**, who are players that become corrupt at some time during the protocol depending on what messages have been sent.

We have a couple important theorems regarding t-resilience.

The first, by BGW, states that there exists a protocol for $t < n/3$ which provides resilient MPC of a function f even for malicious corruption, as long as only pairwise communication is allowed.

Then, Rabin and Ben-Or showed that with broadcasts allowed, we may increase the bound to $t < n/2$.

We formalize these ideas with a simulator. For any t corrupted players, there exists a simulator S, which, given the output $f(x_1, \ldots, x_n)$, simulates the messages from honest players to corrupted players.

1.1 Shamir Secret Sharing

Our basic building block for the above simulator will be **Shamir Secret-Sharing**, also known as (t, n)**-secret-sharing**, where we have a dealer breaking up a secret among n players, such that any set of t players cannot learn anything about the secret, but any set of $t + 1$ players is able to completely reconstruct the secret.

We build this using polynomials. Given $t+1$ points of a degree-t polynomial, we can perfectly reconstruct the polynomial and completely determine all of its $t+1$ coefficients. However, with any fewer points, there are still infinitely many (or as many as the field permits) degree-t polynomials which satisfy the given set of points.

So, the dealer picks t coefficients $a_1, \ldots, a_t$ and sets the polynomial to be $p(x) = s + a_1 x + \ldots + a_t x^t$. Then, to each player $1 \leq i \leq n$, the dealer gives the value $p(i)$. Only when $t + 1$ or more players get together can they reconstruct the secret $s = p(0)$.

Returning to the theorem of BGW, we take the following steps:

0. Within a field F, we choose elements F_0, F_1 to represent $0, 1$, respectively.

1. We represent the function f as an arithmetic circuit via $\oplus$ and $\otimes$ in F.

2. Each player acts as a dealer and secret-shares all inputs via $(n/3, n)$-secret sharing, to all of the players, including the dealer.

3. Assuming that given input wires shared in this way, we can share the output of an $\oplus$ and an $\otimes$ gate in the same way, then we are done, and the protocol is complete.

For $\oplus$, an initial idea might be for each player to add up their secret shares on each of the wires, which would simulate adding the two polynomials together. The only problem is that the resulting polynomial's coefficients are not qutie random. So, we need the players to jointly create a random polynomial $q(x) = 0 + d_1x + \ldots, +d_tx^t$. Then, when we add the respective coefficients from q, we will have an essentially random polynomial, which is the same as a secret sharing of the result wire.

For $\otimes$, the idea is similar, but not exactly the same. If each player multiplies their own secrets together, we simulate multiplying the polynomials, which gives us a degree-$2t$, polynomial. But somehow we need to end up with a secret sharing that $t + 1$ people can figure out. So, at this moment, we have a degree-$2t$ polynomial $p'(x) = \alpha + \beta_1 x + \cdots + \beta_{2t}x^{2t}$. By Lagrange Interpolation, we can figure out public constants $L_1, \ldots, L_{2t+1}$ such that $\alpha = L_1y_1 + \cdots + L_{2t+1}y_{2t+1}$. Then, each player i computes their own y_i as the product of their secrets on the two wires, and then computes $L_i \cdot y_i$. Then, the player secret shares a new polynomial $L_iy_i + a_1x + \cdots + a_tx^t$. The players add all their secret shares together, and we end up with a secret sharing of a degree-t polynomial with constant term α.

1.2 Byzantine Agreement

A separate but related problem is the Byzantine Agreement problem.

The basic idea is that we have some number of parties who each have their own secret bit and can only communicate via pairwise channels. If all the parties agree on which bit to choose, then they will be successful. Otherwise, everyone fails.

So, all honest players must output the same value at the end, since this gives them the best chance of being successful.

As it turns out, this is impossible if we have n players, at least $n/3$ of which are corrupt. Suppose not. Consider the following (specific, but possible) case:

Player 1 is honest and has secret bit 1. Player 2 is honest and has secret bit 0. Player 3 is dishonest and has secret bit 1.

Since Players 1 and 2 are honest, Player 3 can learn each of their secret bits, and privately runs a simulation where each of the other players has the opposite secret bit, and he pretends to agree with both of them.

The communication scheme looks like a hexagon, where P2 talks to P1 and a P3a which seems to agree with P2. P1 talks to a P3b which seems to agree with P1. The

simulated version of P1 talks to P3a, who agrees with him, and to the simulated version of P2. The simulated version of P2 talks to P3b who seems to agree.

Then, to each of P1 and P2, it seems that P3 agrees, so they will each output their own bits, and ultimately disagree. But this breaks the byzantine agreement scheme, so we have a contradiction.