CSC 310: Information Theory **Winter 2026**

Tutorial 12: Shamir Secret Sharing and Secure Computation

Instructor: Swastik Kopparty *TA: Ziyang Jin*

Date: 1 April 2026

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

12.1 Review of Shamir Secret Sharing

Let $\mathbb{F}$ be a finite field. Let $s \in \mathbb{F}$ be the secret to be shared. Suppose we want to share the secret s to n parties such that any subset of at most t parties cannot deduce any information about the secret, and any $t+1$ parties can reconstruct the secret. This is called a t-out-of-n secret sharing.

Essentially, the intuition behind Shamir secret sharing is that any $t+1$ points uniquely determine a degree-t polynomial. On the other hand, if you are just given t points, then there are $|\mathbb{F}|$ possibilities how the degree-t polynomial will look like, and thus there is $1/|\mathbb{F}|$ probability of guessing the polynomial correctly.

To set up the secret sharing, we fix a set of evaluation points $x_1, \dots, x_n \in \mathbb{F}$, which will be publicly known. In practice, people may use $x_1 = 1, \dots, x_n = n$. A secret sharing scheme consists of a pair of algorithms (Share, Recon). A t-out-of-n Shamir secret sharing is defined as follows:

- $\underline{\mathsf{Share}(s)}$: Sample a degree-$t$ polynomial over $\mathbb{F}$ by sampling the coefficients $a_1, \dots, a_t \in \mathbb{F}$ uniformly at random. The polynomial is then $p(x) = s + a_1 x + a_2 x^2 + \dots a_t x^t$. For each $i \in [n]$, compute the share $y_i := p(x_i)$ and send y_i to party i.
- $\underline{\mathsf{Recon}(\{(x_i, y_i)\}_{i\in[n]})}$: Pick any $(t+1)$ points. Without loss of generality, we pick (x_1, y_1), ..., (x_{t+1}, y_{t+1}). Then we use Lagrange interpolation to recover $p(x)$ as follows:

$$p(x) = \sum_{i=1}^{t+1} y_i \cdot \ell_i(x), \tag{12.1}$$

  where

$$\ell_i(x) = \prod_{j\neq i} \frac{x - x_j}{x_i - x_j} = \frac{x - x_1}{x_i - x_1} \cdot \dots \cdot \frac{x - x_{j-1}}{x_i - x_{j-1}} \cdot \frac{x - x_{j+1}}{x_i - x_{j+1}} \cdot \dots \cdot \frac{x - x_{t+1}}{x_i - x_{t+1}}. \tag{12.2}$$

  Once the polynomial $p(x)$ is recovered, we obtain the secret by computing $p(0)$.

Note that in equation (12.1), if we plug-in $p(x_i)$, then equation (12.2) evaluates to 1 and thus equation (12.1) evaluates to y_i. This matches that $p(x_i) = y_i$ for all $i \in [n]$.

Alternatively, you can think of recovering the polynomial $p(x)$ as computing its coefficients from a system linear equations.

Here we assume the shares are all correct and thus we can use Lagrange interpolation to recover the polynomial and thus obtain the secret. If some shares are incorrect (i.e. they are not evaluations of $p(x)$ at given positions), then we will need to use the Berlekamp-Welch algorithm to reconstruct the secret.

Example. Let finite field $\mathbb{F} = \mathbb{F}_7$. Suppose I want to share the secret $s = 3$ to $n = 5$ people such that no single person knows any information about the secret and any two people can recover the secret. How to apply Shamir secret sharing in this case?

Solution. We recognize that the corruption threshold $t = 1$. Therefore, we sample a random degree-1 polynomial $p(x) = s + ax$ where $a \leftarrow \mathbb{F}$. Suppose the uniformly sampled $a = 1$. Then we have $p(x) = x + 3$. We evaluate the polynomial at evaluation points $1, 2, 3, 4, 5$ as follows:

$$\begin{aligned} p(1) &= 1 + 3 = 4 \mod 7 \\ p(2) &= 2 + 3 = 5 \mod 7 \\ p(3) &= 3 + 3 = 6 \mod 7 \\ p(4) &= 4 + 3 = 0 \mod 7 \\ p(5) &= 5 + 3 = 1 \mod 7 \end{aligned}$$

Hence, the shares to party 1 to party 5 are $4, 5, 6, 0, 1$ respectively. To reconstruct the secret, we arbitrarily pick two points $(1, 4)$ and $(4, 0)$. Then we have

$$p(x) = 4 \cdot \ell_1(x) + 0 \cdot \ell_4(x) = 4 \cdot \frac{x - x_4}{x_1 - x_4} = 4 \cdot \frac{x - 4}{4} = x - 4 = x + 3 \mod 7.$$

Then the secret is $p(0) = 3$.

12.2 Secure Multi-Party Computation (MPC)

Secure multi-party computation (MPC) allows n parties to jointly compute a function $f(x_1, \ldots, x_n)$, where each party P_i for $i \in [n]$ holds a private input x_i, and no party learns anything beyond the function output $f(x_1, \ldots, x_n)$ and their own private input x_i. Here we assume semi-honest adversary, which means each party follows the protocol but tries to learn additional information from the messages it exchanges throughout the protocol. We assume the standard secure point-to-point communication channels and a broadcast channel.

Example. Suppose we have three people P_1, P_2, P_3 with private input bit $s_1, s_2, s_3 \in \{0, 1\}$ respectively. The function we want to compute is $f(s_1, s_2, s_3) = s_1 \oplus s_2 \oplus s_3$ (an XOR of the bits). How do we design a MPC protocol Π that computes f?

Approach 1: Additive Secret Sharing. We use the idea of additive secret sharing. The protocol Π_{add} runs as follows:

1. Each party additively shares its secret to all three parties including itself.
    - P_1 shares $s_1 = a_1 \oplus a_2 \oplus a_3$. Thus, P_1 gets a_1, and P_2 gets a_2, and P_3 gets a_3.
    - P_2 shares $s_2 = b_1 \oplus b_2 \oplus b_3$. Thus, P_1 gets b_1, and P_2 gets b_2, and P_3 gets b_3.
    - P_3 shares $s_3 = c_1 \oplus c_2 \oplus c_3$. Thus, P_1 gets c_1, and P_2 gets c_2, and P_3 gets c_3.
2. Each party locally sums up the shares, so P_1 obtains $S_1 := a_1 \oplus b_1 \oplus c_1$, P_2 obtains $S_2 := a_2 \oplus b_2 \oplus c_2$, P_3 obtains $S_3 := a_3 \oplus b_3 \oplus c_3$. Then, each party broadcast its own result to the other two paries. As a result, each party obtains all of S_1, S_2, S_3.
3. Each party locally sums up the values $S_1 \oplus S_2 \oplus S_3$ and outputs.

Indeed, the protocol output is $S_1 \oplus S_2 \oplus S_3 = s_1 \oplus s_2 \oplus s_3$, matching the function output.

Approach 2: Shamir Secret Sharing. We use the idea of Shamir secret sharing. Fix a large enough finite field $\mathbb{F}$ (e.g. $\mathbb{F} = \mathbb{F}_5$). Fix evaluation points $x_1 = 1, x_2 = 2, x_3 = 3$. We require that any two people do not know the secret but all three together can recover the secret. Thus, the corruption threshold $t = 2$.

The protocol Π_{Shamir} runs as follows:

1. For each $i \in \{1, 2, 3\}$, party P_i runs $\mathsf{Share}(s_i)$. Let p_i denote the degree-2 polynomial sampled in the Share algorithm. Thus, P_1 gets $p_i(x_1)$, and P_2 gets $p_i(x_2)$, and P_3 gets $p_i(x_3)$.
2. Each party locally sums up the shares, so P_1 obtains $y_1 := p_1(x_1) + p_2(x_1) + p_3(x_1)$, P_2 obtains $y_2 := p_1(x_2) + p_2(x_2) + p_3(x_2)$, P_3 obtains $y_3 := p_1(x_3) + p_2(x_3) + p_3(x_3)$. Then, each party broadcast its own result to the other two paries. As a result, each party obtains all of y_1, y_2, y_3.
3. Each party reconstructs polynomial $p^*(x)$ by interpolating points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. In the end, each party evaluates $p^*(0)$ and then map it back to $\mathbb{F}_2$ and outputs.

For correctness, let polynomials $p_1(x) = s_1 + a_1x + a_2x^2$, $p_2(x) = s_2 + b_1x + b_2x^2$, $p_3(x) = s_2 + c_1x + c_2x^2$. Essentially what's happening behind the scene is $p^*(x) = p_1(x) + p_2(x) + p_3(x) = (s_1 + s_2 + s_3) + (a_1 + b_1 + c_1)x + (a_2 + b_2 + c_2)x^2$. Therefore, we have $p^*(0) = s_1 + s_2 + s_3$.

Note that the functionality $f(s_1, s_2, s_3) = s_1 \oplus s_2 \oplus s_3$ computes in binary (equivalently $\mathbb{F}_2$), so in the MPC protocol Π_{Shamir}, we also need the final output of each party to be in binary not in $\mathbb{F}$. Note that we cannot pick the finite field $\mathbb{F} = \mathbb{F}_2$ for Shamir secret sharing as there are only two evaluation points 0 and 1 in $\mathbb{F}_2$, while we need at least three. Thus, we pick a bigger field like $\mathbb{F} = \mathbb{F}_5$ and when we run the Share algorithm, we treat the bits b_1, b_2, b_3 as elements in $\mathbb{F}_5$. When we obtain the final result, we map it back to binary.

Remark. We remark that for the above functionality $f(s_1, s_2, s_3) = s_1 \oplus s_2 \oplus s_3$, since it only involves addition operation, both additive secret sharing and Shamir secret sharing work well as they are both additively homomorphic. If the functionality involves computing AND gates or multiplication, Shamir secret sharing is more friendly for multiplication (e.g. the BGW protocol) as when you multiply the shares, the underlying secrets are also multiplied.

12.3 Quiz Time

We will take a 10-minutes quiz (despite today is April Fool's day ☺).