Lecture 23: Secret Sharing and Secure Multi-Party Computation

Notes by Yael Kalai

MIT - 6.5620
Lecture 23 (December 1, 2025)

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Outline

- Secret Sharing: Definition and construction.
- Secure Multi-Party Computation: Motivation and definition.

Secret Sharing

Intuitively, a secret sharing scheme allows a dealer to share a secret s among n parties $P_1, \ldots, P_n$ such that any *authorized* subset of parties can use all their shares to reconstruct the secret, while any other (non-authorized) subset learns nothing about the secret from their shares. Secret sharing has some direct applications, where we need to distribute a secret to several parties/servers, in order to distribute the required trust, as well as to allow reconstruction even if some of the parties fail.

Additionally, secret sharing is a useful tool in many larger cryptographic systems, notably, *secure multi-party computation* which we will discuss today and in the upcoming lectures.

In a secret sharing scheme we want to ensure that no information whatsoever is leaked to any unauthorized subset of parties. For example, simply giving some of the bits of the secret to each party certainly reveals information.

Secret sharing can be defined with respect to any access structure that specifies the set of authorized subsets, as long as that access structure is monotone (namely, if a subset is authorized, any larger subset should also be authorized). We will focus on a common access structure which is t-out-of-n or threshold secret sharing, where authorized subsets are all those of size at least t, while sets of size less than t are not authorize.

Definition

Definition 1. A t-out-of-n secret sharing scheme scheme over message space $\mathcal{M}$ consists of a pair of efficient algorithms $(\mathsf{Share}, \mathsf{Reconstruct})$ such that:

- Share is a randomized algorithm that takes as input a message $m \in \mathcal{M}$ and outputs a n-tuple of shares $(s_1, ..., s_n)$.
- $\mathsf{Reconstruct}$ is a deterministic algorithm that given a t-tuple of shares $\{(i, s_i)\}_{i \in I}$ for $|I| = t$, outputs a message $m \in \mathcal{M}$.

The following two properties are required to be satisfied.

Correctness: For every $m \in \mathcal{M}$ and every $I \subseteq \{1, \ldots, n\}$ of size t,

$$\Pr_{(s_1,\ldots,s_n) \leftarrow \mathsf{Share}(m)} [\mathsf{Reconstruct}(\{(i, s_i)\}_{i \in I}) = m] = 1$$

Security: For every $m, m' \in \mathcal{M}$ and for every $I \subseteq [n]$ such that $|I| < t$,

$$(s_i)_{i \in I} \equiv (s'_i)_{i \in I}$$

where $(s_i)_{i \in [n]} \leftarrow \mathsf{Share}(\mathsf{m})$ and $(s'_i)_{i \in [n]} \leftarrow \mathsf{Share}(\mathsf{m}')$.

n-out-of-n Secret Sharing Scheme

Suppose n parties wish to share a secret $m \in \{0,1\}^\ell$.

$\mathsf{Share}(\mathsf{m})$: Choose at random $s_1, \ldots, s_n \in \{0,1\}^\ell$ such that $\oplus_{i=1}^n s_i = m$, and output $(s_1, \ldots, s_n)$.

$\mathsf{Reconstruct}(s_1, \ldots, s_n)$ outputs $\oplus_{i=1}^n s_i$.

Note that the correctness property follows from the construction, and the security property follows from the fact that an equivalent way to compute $\mathsf{Share}(\mathsf{m})$ is to choose at random $\{s_i\}_{i \neq j}$ and set $s_j = m \oplus (\oplus_{i \neq j} s_i)$, which implies that any set of shares that excludes at least one share, are randomly distributed, independently of m.

t-out-of-n Secret Sharing Scheme

Take 1: For simplicity, suppose that $t = 2$. Then to share a secret m, for every pair of distinct parties (i, j) in $\{1, \ldots, n\}$ generate 2-out-of-2 shares of m. Then give each party all the shares generated for that party. Note that each party has $n - 1$ shares.

Problem: If we use this approach to construct a t-out-of-n secret sharing scheme this will result with shares of size $\binom{n}{t-1}$.

Shamir's idea [2]: Use polynomials! Suppose we wish to share a message $m \in \{0,1\}$. If we want to share ℓ bits we will use the single-bit secret sharing scheme ℓ times, one for each bit. Choose a prime p such that $p > n$.

Share(m): Choose a random degree $t-1$ polynomial $f : \mathsf{GF}[p] \to \mathsf{GF}[p]$ such that $f(0) = m$. This can be done by choosing $a_1, \ldots, a_{t-1}$ at random in $\mathsf{GF}[p]$, setting $a_0 = m$ and letting

$$f = \sum_{i=0}^{t-1} a_i x^i.$$

Recall that $\mathsf{GF}[p]$ denotes the field with elements $\{0, 1, \ldots, p-1\}$ where addition and multiplication are done modulo p.

For every $i \in [n]$ let $s_i = f(i)$ and output $(s_1, \ldots, s_n)$.

The shares can be the value of f on any n distinct points in the field $\mathsf{GF}[p]$ as long as none of these points is 0.

Reconstruct$(\{(\alpha_j, s_{\alpha_j})\}_{j \in J})$: Solve t linear equations in t variables. The variables are $a_0, a_1, \ldots, a_{t-1}$ and each player P_{α_j} holds a linear questions:

$$\sum a_i (\alpha_j)^i = s_j$$

We note that these t linear equations are always independent (this is a Vandermonde matrix, which is known to be invertible).

We emphasize that these t players can jointly recover not only the secret m, but also the entire degree $t-1$ polynomial that the dealer chose, by computing:

$$f(x) = \sum_{i=1}^{t} f_i(x) \cdot s_i$$

where f_i is the unique degree $t-1$ polynomial that satisfies that $f_i(x) = 1$ if $x = \alpha_i$ and $f_i(x) = 0$ for every $x \in \{\alpha_j\}_{j \in [t] \setminus \{i\}}$. Namely,

$$f_i(x) = \prod_{j \in [t] \setminus \{i\}} \frac{\alpha_j - x}{\alpha_j - \alpha_i}$$

Connection to Reed-Solomon Codes

Shamir's secret-sharing scheme is very similar to the Reed-Solomon error-correcting codes (1960), which is a beautiful coding scheme! In a Reed-Solomon code a message $m = (m_0, m_1, \ldots, m_{t-1}) \in \{0,1\}^t$ is viewed as the unique degree $t-1$ polynomial f_m such that $f_m(i) = m_i$ for every $i \in \{0, 1, \ldots, t-1\}$. The codeword corresponding to the message m is the evaluation of the polynomial f corresponding to m on all the points in the field:

$$\mathsf{ECC}(m) = (f_m(0), f_m(1), \ldots, f_m(p-1)).$$

What we have seen above is that this codeword can be uniquely decoded even if all but t of the coordinates in the codeword were erased.

It is also known how to decode if less than $\frac{p-t+1}{2}$ of the coordinates were maliciously corrupted, and this is known to be optimal! Decoding from malicious corruptions is slightly trickier than decoding from erasure, but it is not too hard (and is explained beautifully in these Lecture notes by Anup Rao). This means that in Shamir's secret sharing scheme if a secret is shared among n parties and then the n parties jointly try to decode, even if some of the parties try to foil the outcome and give malicious shares, still the parties will be able to jointly recover the secret, as long as less than $\frac{n-t+1}{2}$ of the parties are corrupted.

The main disadvantage of the Reed-Solomon ECC is that it is over a large alphabet. Often people want an ECC over the binary alphabet. For the application of cryptography and secret-sharing this is good enough.

We will next see how to use Shamir's secret-sharing scheme to do *secure multi-party computation*, defined below.

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 a function $f(x_1, \ldots, x_n)$ without revealing any information about their secret inputs beyond the function output $f(x_1, \ldots, x_n)$. This task can be achieved using a multi-party computation (MPC) scheme. MPC enables multiple parties to jointly compute a function over their inputs while keeping their inputs private. It has several practical applications across various domains. Some examples are:

- Privacy-Preserving Data Analysis: MPC allows multiple parties to analyze sensitive data collaboratively without revealing individual inputs. This is particularly useful in sectors like healthcare, finance, and market research, where data privacy is critical but collaborative analysis is necessary.
- Secure Machine Learning: MPC can be used to train machine learning models on combined datasets from multiple sources without sharing the raw data. This is beneficial in situations where data sharing is restricted due to privacy concerns or regulations.
- Secure Auctions and Bidding: In auction scenarios, MPC ensures fairness and privacy by allowing bidders to submit encrypted bids, preventing bid manipulation and preserving bidder anonymity.

Definition

We need to define the model of communication and security.

The Model. In the MPC model we assume that all n parties are connect via *private and authenticated point-to-point channels*, which means that each pair of parties P_i and P_j can send each other messages via their point-to-point channel in a way that only they can send and receive the bits that are communicated, and no other party can read or tamper with these messages. The reason we assume the existence of such channels is that we know how to implement them using encryption schemes and signature schemes, the former to make the channel private and the latter to make it authentic.

We also assume a synchronous network, where the protocols proceed in "rounds", and at the end of each round all the messages that were sent were received. This assumption is not necessarily realistic and is made to simplify protocol design. There are various techniques in the literature for dealing with the asynchronous setting (which we will not get into in this class).

Defining Security. As we saw in the course of the semester, defining security of cryptographic primitives is tricky. The case of MPC is no different and in some sense is the trickiest of them all.

Take 1: A MPC protocol is secure if by the end of the protocol no party learns any information about the other parties' inputs.

Problem: This is impossible to achieve since the output itself $f(x_1, \ldots, x_n)$ reveals information about the other parties' inputs.

Take 2: A MPC protocol is secure if by the end of the protocol no party learns any information about the other parties' inputs, except of $f(x_1, \ldots, x_n)$.

Problem: This does not provide the desired security guarantee. We want the inputs to be "independent." Consider for example the auction setting, and suppose we do an MPC to compute who is the party with the highest bid. It is not enough to say that I don't learn anyone's bid. We need to ensure that I cannot somehow outbid you by 1 (without knowing your bid).

Take 3: A MPC protocol should be as secure as an *"ideal world"* implementation, where each party P_i sends her secret input x_i to a trusted party and then the parties receive $f(x_1, \ldots, x_n)$. This is formalized below

Definition 2. An n-party protocol securely computes a function f in the presence of at most t corruptions, if for every (PPT) adversary $\mathcal{A}$

"controlling" at most t parties in the real world, there exists a (PPT) simulator $\mathcal{S}$ "controlling" the same subset of parties in the ideal world, such that for any set of inputs $x_1, \ldots, x_n$,

$$\mathsf{Real}_{\mathcal{A}}(x_1, \ldots, x_n) \equiv \mathsf{Ideal}_{\mathcal{S}}(\mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n})$$

where $\mathsf{Real}_{\mathcal{A}}(x_1, \ldots, x_n)$ is the output of all the parties after running the protocol, where the adversarial party output whatever they want; $\mathsf{Ideal}_{\mathcal{S}}(x_1, \ldots, x_n)$ is the output of all the parties after handing their inputs to the trusted party, who computes f on their input and returns the output y. The honest parties output y and the adversarial parties (controlled by the simulator) output whatever they want.

We consider two types of adversaries: *honest-but-curious* and *malicious*. A malicious adversary that controls t-parties completely controls them and can choose to send arbitrary messages on their behalf (and so can the simulator in the ideal world). In the honest-but-curious setting an adversary that controls t parties learns all their inputs and the messages that they receive and send (as well as their internal state), but cannot modify their messages.

In this class we present the MPC protocol due to Ben-Or, Goldwasser and Wigderson [1] (henceforth referred to as the BGW protocol). They constructed a version that is secure against an honest-but-curious adversary that controls less than $t = n/2$ parties. They also constructed a protocol that is secure against a malicious adversary that controls less than $t = n/3$ parties. We will only cover their protocol in the honest-but-curious setting.

We note that their protocol is information theoretically secure! It does not rely on any cryptography, and is secure even if the adversary is all powerful.

We note that if we rely on cryptography we can get security against any number of corruptions!

References

[1] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988.

[2] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.