Secret Sharing and Secure Multiparty Computation

Secret sharing

Motivation

- Secret information - secret (for example, safe code, password, key, treasure map, rocket launch code,. . .).
- Not secure if it is stored in one device/person
- Solution - Distribute the secret into shares: The shares are stored independently, and the secret can be reconstructed if needed.

Secret sharing

Motivation

- Secret information - secret (for example, safe code, password, key, treasure map, rocket launch code,. . .).
- Not secure if it is stored in one device/person
- Solution - Distribute the secret into shares: The shares are stored independently, and the secret can be reconstructed if needed.
  - Bad way: Cut it into pieces, e.g. 35749134 $\rightarrow$ 35,74,91,34

Secret sharing

Motivation

- Secret information - secret (for example, safe code, password, key, treasure map, rocket launch code,. . .).
- Not secure if it is stored in one device/person
- Solution - Distribute the secret into shares: The shares are stored independently, and the secret can be reconstructed if needed.
  - Bad way: Cut it into pieces, e.g. 35749134 $\rightarrow$ 35,74,91,34
  - Good way: secret $s \in \mathbb{F}$ and shares $a, b, c, s - a - b - c \in \mathbb{F}$.

Shamir secret sharing (SSS)

Shamir secret sharing (k-threshold) scheme:

- P: set of participants, D : dealer
- s secret
- $A \subseteq P$ can recover the secret if $|A| \geq k$

Shamir secret sharing (SSS)

Shamir secret sharing (k-threshold) scheme:

- P: set of participants, D : dealer
- s secret
- $A \subseteq P$ can recover the secret if $|A| \geq k$

Construction (Shamir)

1. $\mathbb{F}$ *finite field (e.g.,* $\mathbb{F}_p$*) such that* $|\mathbb{F}| > |P|$. $\mathbb{F}$ *is public.*
2. D *chooses randomly* $p \in \mathbb{F}[x]$, *with* $\deg p \leq k-1$ *and* $p(0) = s$.
3. D *sends* $s_i = p(i)$ *to* P_i.

1. If $A \subseteq P, |A| \geq k$: Lagrange-interpolation
2. If $A \subseteq P, |A| < k \Rightarrow$ cannot compute anything about p

Lagrange-interpolation

Lagrange-interpolation

- $\mathbb{F}$ field, $p \in \mathbb{F}[x]$ polynomial, $\deg p \leq k - 1$
- $y_i = p(x_i)$ is known for $i = 1, \ldots, k$
- Determine the polynomial p

Lagrange-interpolation

Lagrange-interpolation

- $\mathbb{F}$ field, $p \in \mathbb{F}[x]$ polynomial, $\deg p \leq k-1$
- $y_i = p(x_i)$ is known for $i = 1, \ldots, k$
- Determine the polynomial p

1. Construct basis polynomials: $\ell_i(x_i) = 1$ and $\ell_i(x_j) = 0$, if $i \neq j$.

$$\ell_i(x) = \prod_{\substack{1 \leq j \leq k \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$$

2. Construct p:

$$p(x) = \sum_{i=1}^{k} p(i)\ell_i(x)$$

Remark

To reconstruct the secret, it is enough to compute $p(0)$.

Example - Lagrange-interpolation

Example

Compute the polynomial $p \in \mathbb{F}_7$ such that $\deg(p) \leq 2$ and $p(1) = 2, p(3) = 6$ and $p(4) = 1$.

1. Compute the base polynomials:

$$\ell_1(x) = \frac{(x-3)(x-4)}{(1-3)(1-4)} = \frac{x^2 - 7x + 12}{6} = 6x^2 + 2$$

$$\ell_3(x) = \frac{(x-1)(x-4)}{(3-1)(3-4)} = \frac{x^2 - 5x + 4}{-2} = 3x^2 + 6x + 5$$

$$\ell_3(x) = \frac{(x-1)(x-3)}{(4-1)(4-3)} = \frac{x^2 - 4x + 3}{3} = 5x^2 + x + 1$$

2. Compute p:

$$2\ell_1(x) + 6\ell_3(x) + 4\ell_4(x) = 50x^2 + 40x + 38 = x^2 + 5x + 3$$

Example

Suppose that we want to generate a 3-out-of-4 secret sharing.

Secret generation

- Setup: $P = \{P_1, P_2, P_3, P_4\}$, $k = 3$, $\mathbb{F} = \mathbb{F}_7$, $s = 3$
- Polynomial: $p(x) = x^2 + 5x + 3$
- Shares: P_1 : 2, P_2 : 3, P_3 : 6 P_4 : 4

Example

Suppose that we want to generate a 3-out-of-4 secret sharing.

Secret generation

- Setup: $P = \{P_1, P_2, P_3, P_4\}$, $k = 3$, $\mathbb{F} = \mathbb{F}_7$, $s = 3$
- Polynomial: $p(x) = x^2 + 5x + 3$
- Shares: $P_1 : 2$, $P_2 : 3$, $P_3 : 6$ $P_4 : 4$

Suppose P_1, P_3, and P_4 want to reconstruct the secret from their shares.

Secret reconstruction

They collect their shares (2, 6, 4) and compute

$$2\frac{(0-3)(0-4)}{(1-3)(1-4)} + 6\frac{(0-1)(0-4)}{(3-1)(3-4)} + 4\frac{(0-1)(0-3)}{(4-1)(4-3)} =$$
$$= 2 \cdot 2 + 6 \cdot (-2) + 1 \cdot 4 = 3$$

Shamir Secret sharing - Properties

Notation

$[s]$: s is a value shared with SSS. The share of P_i is s_i.

- k-out-of-n secret sharing: SS with n parties and threshold k.
- The size of the share is $\log_2 |F|$.
- $|F| > |P|$ is important, as all participants must receive a different value of the polynomial.

Remark

Every polynomial $p \in \mathbb{F}[x], \deg p < k$, $p(0) = s$ determines a valid SSS of the secret s.

MPC motivation - Private Dating

Private dating

Alice and Bob meet at a bar.

- If both of them want to date together - they will find out
- If Alice doesn't want to date - she won't learn his intentions
- If Bob doesn't want to date - he won't learn her intentions

MPC motivation - Private Dating

Private dating

Alice and Bob meet at a bar.

- If both of them want to date together - they will find out
- If Alice doesn't want to date - she won't learn his intentions
- If Bob doesn't want to date - he won't learn her intentions

- Solution with *trusted third party*: Both Alice and Bob tell their intention to a trusted third party (bartender, dating app, friend)
- What if a trusted third party is not available?

MPC motivation - Private Auction

Private Auction

Many parties wish to execute a private auction

- The highest bid wins
- Only the highest bid (and bidder) is revealed

- Solution with *trusted third party*: Every bidder shares their bidding with a trusted third party (auctioneer, computer).
- What if a trusted third party is not available?

MPC motivation - Private Set Intersection

Private Set Intersection (PSI)

Intelligence agencies hold lists of potential terrorists (MI5, FBI)

- They would like to compute the intersection
- Any other information must remain secret

- The solution with a trusted third party is unacceptable. Is there any other way?

MPC motivation - Secure shuffling

Secure shuffling

- Clients: hold sensitive data.
- Shuffler: collects, shuffles, and sends client data to the analyzer.
- Analyzer: analyzes the data.

How do we protect client data if there is a small probability that the shuffler and analyzer collide?

Trusted Third Party

- All the previous challenges can be solved with the help of a trusted third party.
- Trusting a third party is a powerful assumption (in most cases, it is not available).
- Can we do this without any trusted party?

MPC - definition

MPC (Secure Multiparty Computation)

- Parties: $P_1, \ldots, P_N \in \mathcal{P}$.
- Private inputs: $x_1, \ldots, x_N$. ($x_i \to P_i$)
- Function: f (N variable)
- Goal: jointly compute $(y_1, \ldots, y_N) = f(x_1, \ldots, x_N)$ where y_i is only known to P_i (in some cases $y_1 = \ldots y_N$).

MPC - definition

MPC (Secure Multiparty Computation)

- Parties: $P_1, \ldots, P_N \in \mathcal{P}$.
- Private inputs: $x_1, \ldots, x_N$. ($x_i \rightarrow P_i$)
- Function: f (N variable)
- Goal: jointly compute $(y_1, \ldots, y_N) = f(x_1, \ldots, x_N)$ where y_i is only known to P_i (in some cases $y_1 = \ldots y_N$).

Examples

1. Private dating: Inputs: 0 (no),1 (yes). Function: $\wedge$ (logical AND)
2. Private auction: Inputs: whole number. Function: maximum
3. Terrorist Inputs: sets. Function: Intersection

Requirements

1. Correctness: parties obtain correct output (even if some parties misbehave. It is possible that the protocol halts with no outputs)
2. Privacy: Only the output is learned
3. Independence of inputs: parties cannot choose their inputs as a function of other parties' inputs
4. Fairness: if one party learns the output, then all parties learn the output

Sum

Example

The input of P_i is x_i, compute $\sum_{i=1}^{N} x_i \mod M$.

Sum

Example

The input of P_i is x_i, compute $\sum_{i=1}^{N} x_i \mod M$.

1. P_1: choose $r \in_R \mathbb{F}_M$ and send $m_1 = x_1 + r$ to P_2,
2. P_2: send $m_2 = x_2 + m_1$ to P_3;

   $\ldots$
3. P_i: send $m_i = x_i + m_{i-1}$ to P_{i+1};

   $\ldots$
4. P_N send $m_N = x_N + m_{N-1}$ to P_1
5. P_1 broadcast $y = m_N - r$.

$$m_i = r + \sum_{j=1}^{i} x_j \Rightarrow y = m_N - r = \sum_{j=1}^{N} x_i$$

Adversary

Adversaries (inner and outer) might attack the protocol (corrupt parties) by trying to recover private input information. Based on the attack, the parties can be:

- Honest: Follows the protocol, does not compute anything else.
- Semi-honest: Follows the protocol but tries to learn as much as possible.
- Fail stop: Semi-honest, but can halt at any moment.
- Malicious: Can deviate from the protocol in any way.

The adversary can corrupt more participants at the same time.

Attacks against SUM

- Semi-honest participants:
  - P_i (with m_{i-1}, x_i and y) unable to compute anything vulnerable.
  - P_i and P_{i+2} together:

$$m_{i+1} - m_i = \left(\sum_{j=1}^{i+1} x_j + r\right) - \left(\sum_{j=1}^{i} x_j + r\right) = x_{i+1}$$

- Fail stop: Halts the protocol.
- Malicious: Wrong result (unable to detect)

SUM with SSS

Construction

$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i$

1. *$\forall i$ P_i shares its input as a dealer with others, sending $p_i(j)$ to P_j.*
2. *$\forall i$ P_i locally computes $\sum_{j=1}^{n} p_j(i)$*
3. *k parties (using Lagrange-interpolation) jointly compute $\sum_{j=1}^{n} p_j(0)$.*

Remark

- *If $x_i = 0$ $\forall i \Rightarrow$ the parties can jointly generate $[0]$.*
- *If x_i is random $\forall i \Rightarrow$, the parties can jointly generate a shared random (unknown to them).*
- *If $x_1 = t$ és $x_i = 0$ $\forall i, i \neq 1 \Rightarrow$ parties can jointly generate $[t]$ for any t.*

Operations with shared secrets

Lemma

Given $[a]$, $[b]$ (SSS of a and b) and a constant c, participants can locally calculate an Shamir secret sharing of $a + b$ and ca.

If the shares of P_i in $[a]$ and $[b]$ are a_i and b_i respectively, then

- $[a] + [b]$: secret sharing s.t. the shares of P_i is $a_i + b_i$.
- $c[a]$: secret sharing s.t. the shares of P_i is ca_i
- $[a] \to p$, $[b] \to q$,
- $[a] + [b]$ is a SSS of $a + b$: $r = p + q$. $r(i) = p(i) + q(i) = a_i + b_i \; \forall i = 0, 1, \ldots, n$.
- $c[a]$ is a SSS of ca: $r' = c \cdot p$. $r'(i) = c \cdot p(i) = c \cdot a_i \; \forall i = 0, 1, \ldots, n$.

SUM with SSS example

Example

A 2-out of-3 SSS over $\mathbb{F}_{11}$.

Name	Alice (1)	Bob (2)	Cloe (3)
Input	5	2	7
Polynomial	$3x+5$	$9x+2$	$x+7$
Alice's shared input	8	0	3
Bob's shared input	0	9	7
Cloe's shared input	8	9	10
Sum of shared values	5	7	9

Reconstruction of the Secret (Alice and Cloe):

$$s = 5\frac{0-3}{1-3} + 9\frac{0-1}{3-1} = 2+1 = 3.$$

SUM against attacks

Security:

- Semi-honest: if at most $k-1$ parties collude $\Rightarrow$ cannot compute anything vulnerable
- Halting: If at least k parties remain, they can compute SUM.
- Malicious: Suppose that every party is honest in step 1 and the number of malicious parties is at most $t \leq k-1$.
    - Attack is detectable if at least $k+t$ parties participate in the interpolation.
    - The correct result can be computed if at least $k+2t$ parties participate in the interpolation.

Multiplication

- Multiplication: Given $[a]$ and $[b]$, compute $[ab]$.
- If $r = pq$, then $r(0) = p(0)q(0) = a \cdot b$.
- P_i locally computes $a_i\dot{b}_i$.

Multiplication

- Multiplication: Given $[a]$ and $[b]$, compute $[ab]$.
- If $r = pq$, then $r(0) = p(0)q(0) = a \cdot b$.
- P_i locally computes $a_i\dot{b}_i$.

Problems

- Problem 1: The degree of r can be $2k - 2 \Rightarrow 2k - 1$ parties needed to restore the secret
- Problem 2: r not random

- Degree reduction algorithm - It takes extra communication.
- Randomize r with $[0]$.