Secret Sharing

Today:
- Motivation & Definition
- A simple (n,n)- construction
- Shamir Secret Sharing
- 2-PC Using Secret Sharing

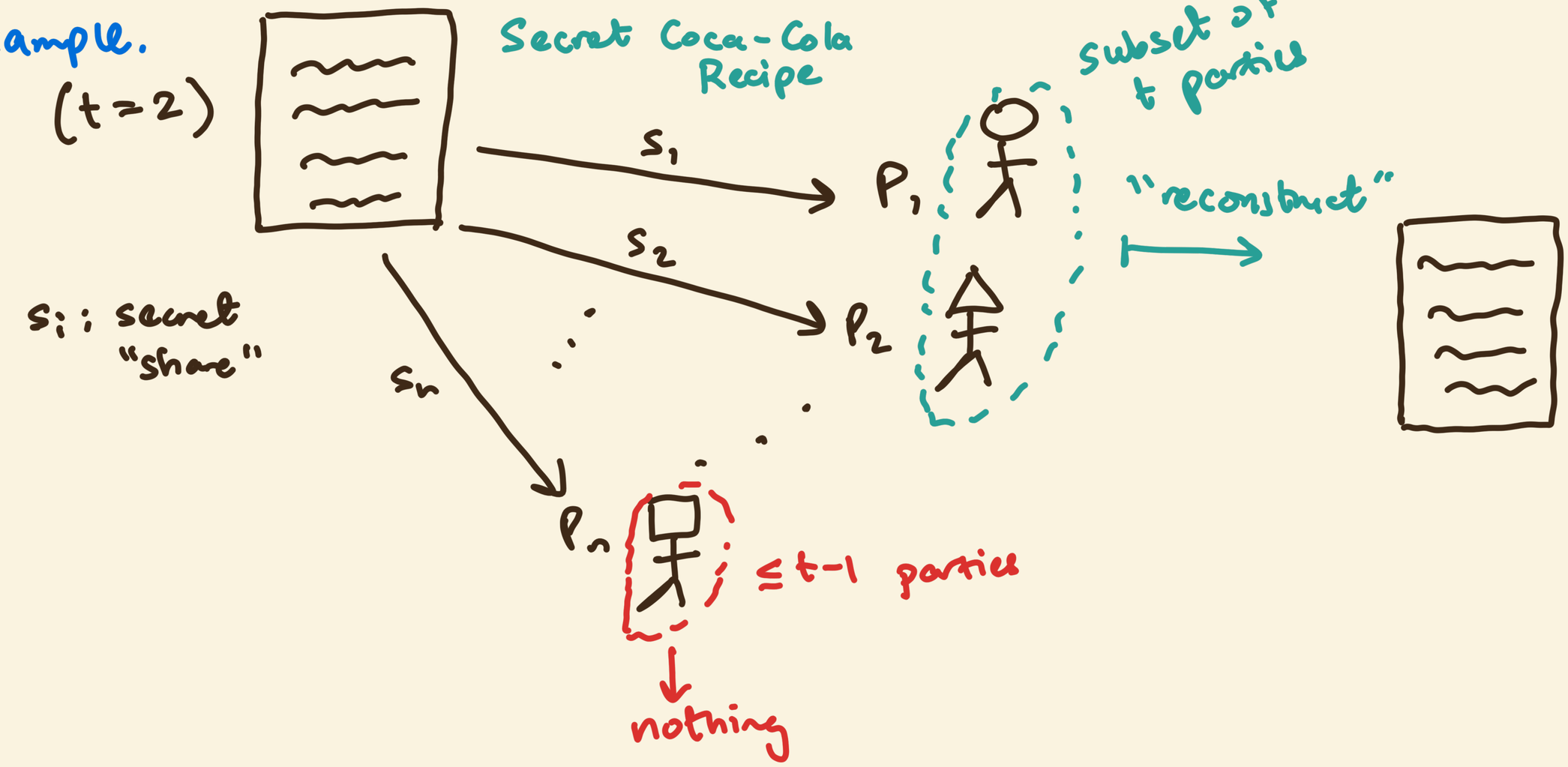
Motivation

There is some secret data...

We don't trust any single party with it. Can we share the data across multiple parties so no one single party can access it? We can relax it from single party to some threshold.

Example.

$(t=2)$



Formally, (ss)

Def. A (t,n)-secret sharing scheme over a message space $\mathcal{M}$ and share space S is a tuple of eff. algs.

$$\text{Share} : \mathcal{M} \longrightarrow S^n,$$
$$\text{Reconstruct} : S^t \longrightarrow \mathcal{M},$$

such that the following properties are satisfied:

Correctness. "any t shares can be used to reconstruct m"

$$\forall m \in \mathcal{M},\ (s_1, \ldots, s_n) \leftarrow \text{Share}(m),$$
$$\forall S \subseteq \{s_1, \ldots, s_n\} \text{ where } |S| = t,$$
$$\text{Reconstruct}(S) = m$$

Security. "less than t shares reveal nothing about m"

$$\forall m_0, m_1 \in \mathcal{M},\ \forall I \subseteq \{1, \ldots, n\} \text{ where } |I| < t,$$

denote $(s_1, \ldots, s_n) \leftarrow \text{Share}(m_0)$

$(s_1', \ldots, s_n') \leftarrow \text{Share}(m_1)$.

Then, $\{s_i : i \in I\} \approx \{s_i' : i \in I\}$

today, these distributions will be identical!

A Simple (n,n)-SS Construction

$(\mathcal{M} = \mathcal{S} = \mathbb{F})$ ← or any abelian group...

Share(m):

- Sample $r_1, \dots, r_{n-1} \xleftarrow{\$} \mathbb{F}$
- Define $r_n := m - \sum_{i=1}^{n-1} r_i$
- Output $(r_1, \dots, r_n)$

Reconstruct($\{r'_1, \dots, r'_n\}$):

- Output $m' := \sum_{i=1}^{n} r_i$

Correctness. $\sum_{i=1}^{n} r_i = \sum_{i=1}^{n-1} r_i + \left(m - \sum_{i=1}^{n-1} r_i\right) = m$

Security. $\forall m_0, m_1 \in \mathcal{M}$, distribution of any n-1 shares are identical!

Shamir Secret Sharing : (t,n)-SS

$(\mathcal{M} = \mathcal{S} = \mathbb{F}$ s.t. $|\mathbb{F}| > n)$

Polynomials are our best friend!

Intuition. A polynomial of degree t-1 can be uniquely determined by t points.
E.g. a line is uniquely determined by 2 points, a parabola by 3,...

Linear Algebra Viewpoint. Given a point $x \in \mathbb{F}$ and poly $f(X) = c_0 + c_1 X + \cdots + c_{t-1} X^{t-1}$, we can view the evaluation $f(x)$ as an inner product:

$$f(x) = \langle (1, x, \ldots, x^{t-1}), (c_0, \ldots, c_{t-1}) \rangle.$$

t distinct points $x_0, \ldots, x_{t-1}$ define a linear system

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{t-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{t-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{t-1} & x_{t-1}^2 & \cdots & x_{t-1}^{t-1} \end{bmatrix}}_{\text{known as the Vandermonde matrix (for } \{x_i\})} \underbrace{\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{t-1} \end{bmatrix}}_{\text{coefficients}} = \underbrace{\begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{t-1}) \end{bmatrix}}_{\text{evaluations}}$$

$V(x_0, \ldots, x_{t-1})$

So given evaluations, calculating the coefficients, i.e., interpolating $f(X)$, is equivalent to solving the linear system above.

Fact. For distinct $x_0, \ldots, x_{t-1}$, the Vandermonde matrix is invertible.

Thus, interpolating can be done by $(\text{Vandermonde})^{-1}(\text{evaluations})$

Proof Sketch of Invertibility.

Suppose not. Then cols are lin. dependent:

$\exists\, c_0, \dots, c_{t-1}$, not all zero, s.t.

$$c_0\begin{bmatrix}1\\ \vdots\\ 1\end{bmatrix} + c_1\begin{bmatrix}x_0\\ \vdots\\ x_{t-1}\end{bmatrix} + \dots + c_{t-1}\begin{bmatrix}x_0^{t-1}\\ \vdots\\ x_{t-1}^{t-1}\end{bmatrix} = \begin{bmatrix}0\\ \vdots\\ 0\end{bmatrix}$$

$\Rightarrow x_0, \dots, x_{t-1}$ are t distinct roots of $f(X) := c_0 + c_1 X + \dots + c_{t-1} X^{t-1}$.

But f is of degree t, which has at most $t-1$ roots unless $f = 0$.

$\Rightarrow c_i = 0$ ↯.

Construction. "Encode the secret as the constant coefficient c_0 and let the shares be evaluations"

Share(m):

- Sample $c_1, \dots, c_{t-1} \xleftarrow{\$} \mathbb{F}$
- Define $f(X) = m + \sum_{i=1}^{t-1} c_i X^i$
- Output $\left(s_i := (i, f(i)) : \forall\, i \in [n]\right)$ ← evaluations on f

Reconstruct($\{(x_i, y_i) : \forall i \in S, |S| = t\}$):

- Interpolate unique poly f of deg. $t-1$ defined by those points
- Output $f(0)$ ← constant coefficient

Correctness. Follows from the uniqueness of interpolation (t points define a poly of degree $\leq t-1$)

Security. We explicitly write out the distribution of shares for any message.

Let $m \in \mathbb{F}$ and $I \subseteq [n]$, s.t. $|I| = t-1$, be arbitrary.

Let $\{x_1, \ldots, x_{t-1}\} := I$.

Consider arbitrary $y_1, \ldots, y_{t-1} \in \mathbb{F}$.

The probability that the I-subset of shares output by Share(m) are $(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})$ is

$$\Pr\left[V(0, x_1, \ldots, x_{t-1}) \begin{bmatrix} m \\ c_1 \\ \vdots \\ c_{t-1} \end{bmatrix} = \begin{bmatrix} m \\ y_1 \\ \vdots \\ y_{t-1} \end{bmatrix} \right]$$

$$= \Pr\Bigg[\underbrace{\begin{bmatrix} m \\ c_1 \\ \vdots \\ c_{t-1} \end{bmatrix}}_{c_i \xleftarrow{\$} \mathbb{F}} = \underbrace{V^{-1} \begin{bmatrix} m \\ y_1 \\ \vdots \\ y_{t-1} \end{bmatrix}}_{\text{fixed arbitrary vector} \in \mathbb{F}^t} \Bigg] = \frac{1}{|\mathbb{F}|^{t-1}}$$

← independent of m

"For any choice of (t-1)-shares and any msg. m, there exists a unique polynomial of of deg. t-1 s.t. $f(x_i) = y_i$ and $f(0) = m$

So, any t-1 shares can be consistent with the sharing of m."

2-PC Using Secret Sharing

2-PC for computation described as arithmetic circuits.

We achieve semi-honest security ← parties follow the protocol honestly

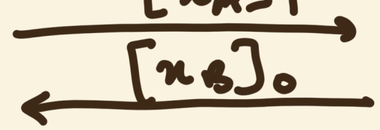
(it's possible to very easily get malicious security using "MACs")

Framework.

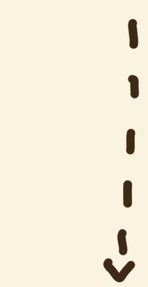
Alice (x_A) Bob (x_B)

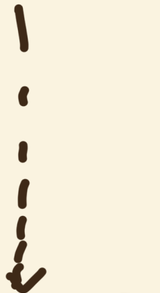
1. Share inputs

$[x_A]_0, [x_A]_1 \leftarrow \text{Share}(x_A)$ $[x_B]_0, [x_B]_1 \leftarrow \text{Share}(x_B)$



2. Compute on secret shares somehow





3. Output final shares

$[\text{output}]_0$

$[\text{output}]_1$

4. Reconstruct

$\text{Reconstruct}([\text{output}]_0, [\text{output}]_1) \rightarrow \text{output}$

For AC. Idea: derive shares of intermediate wires incrementally.

1. Both parties secret share their inputs
2. For each addition gate with inputs $[x] := ([x]_0, [x]_1)$, $[y] := ([y]_0, [y]_1)$, the parties derive shares of the output $[x+y] := ([x+y]_0, [x+y]_1)$.
3. Similarly for multiplication gates: derive $[x \cdot y]$.
4. Once final shares of output wire are derived, parties reveal their shares to reconstruct the output

Protocol. We will use the (n,n)-SS from before with $n = 2$. So, $[x]_0 = r$ $(r \xleftarrow{\$} \mathbb{F})$, $[x]_1 = x - r$

(also called "additive SS")

We show how to compute on SSs to derive $[x+y]$ and $[x \cdot y]$ from $[x]$ and $[y]$.

Addition. Easy! Just add the shares locally.

Set $[x+y]_0 = [x]_0 + [y]_0$

$[x+y]_1 = [x]_1 + [y]_1$

Then, $\underbrace{[x+y]_0 + [x+y]_1}_{\text{Reconstruct}} = [x]_0 + [y]_0 + [x]_1 + [y]_1 = x + y$

Can also easily add constants: $[c+x]$: $[c+x]_0 = c + [x]_0$, $[c+x]_1 = [x]_1$

Multiplication. By a constant is easy:

$[cx]$: Set $[cx]_0 = c \cdot [x]_0$

$[cx]_1 = c \cdot [x]_1$

Multiplying shares requires a bit more work

Today: Beaver's Trick (requires setup + interaction)

Beaver 1991:

Setup: Suppose parties already have shares of a product of random values

Let $a, b \xleftarrow{\$} \mathbb{F}$, and $c = ab$

Beaver triple : $([a], [b], [c])$

Interaction: To multiply shares $[x]$ and $[y]$ to obtain $[x \cdot y]$,

1. Locally compute $[x-a]$ and $[y-b]$ ← using addition mechanism described above
2. Reveal shares to reconstruct $\varepsilon := x-a$ and $\delta := y-b$

think of a and b as one-time pads on x and y (resp.)

3. Locally compute $[z] = [c] + [\delta x] + [\varepsilon y] - \frac{\varepsilon\delta}{2}$

Correctness. $[z]_0 + [z]_1 = [c]_0 + [\delta x]_0 + [\varepsilon y]_0 - \frac{\varepsilon\delta}{2}$
$+ [c]_1 + [\delta x]_1 + [\varepsilon y]_1 - \frac{\varepsilon\delta}{2}$

$= c + \delta x + \varepsilon y - \varepsilon\delta$

$= ab + (y-b)x + (x-a)y - \varepsilon\delta$

$= ab + xy - bx + xy - ay - (x-a)(y-b)$

$= ab + xy - bx + xy - ay - xy - ab + ay + bx$

$= xy$

Security. opening a bunch of random values... (informal)

So, how does setup work??

Requires public-key crypto or a trusted dealer

- OTs
- Garbled circuits
- Homomorphic Encryption

We need a new triple for each multiplication gate (reusing will break the one-time pad)

2-PC in the Preprocessing Model

Offline/Preprocessing stage. Parties generate B Beaver triples where B is an upper bound on the number of multiplication gates. This stage is expensive but independent of the circuit and parties' inputs.

e.g. one can generate many many triples in advance for use in the future; maybe waiting for data to be available or computation to be agreed on...

Online stage. no expensive PK operations but requires communication linear in the # of mult. gates and # of inputs.