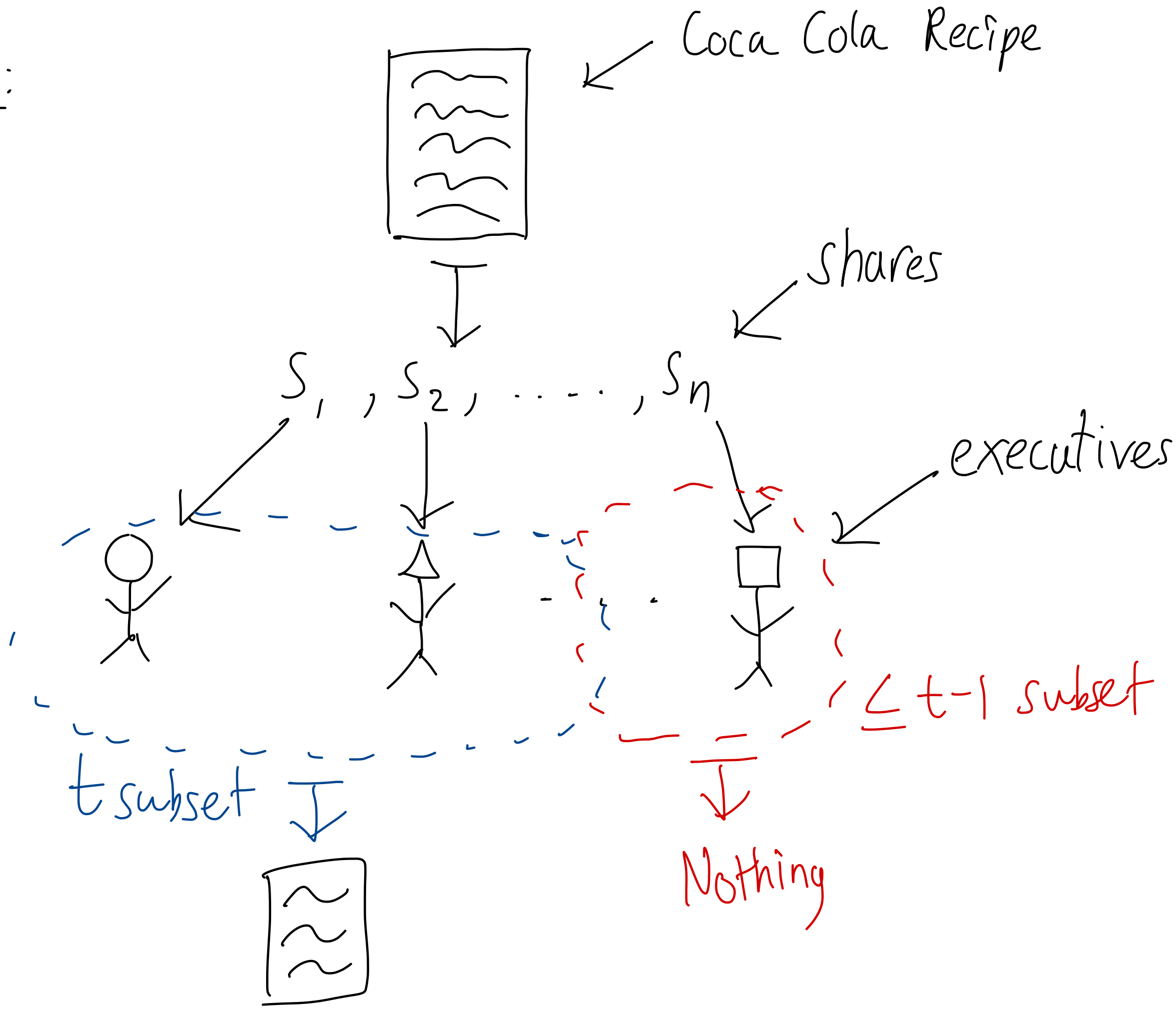
Secret Sharing

Suppose I have a secret that I want to share across n parties such that any t subset can recover my secret, but any $\le t-1$ subset learn nothing about my secret.

Example:



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

$$\text{Share}: \mathcal{M} \to S^n$$

$$\text{Reconstruct}: S^t \to \mathcal{M}$$

With the following properties:

Correctness: Any t shares can be used to reconstruct m.

$\forall m \in \mathcal{M}$, $(s_1, \ldots, s_n) \leftarrow \text{Share}(m)$

$\forall S \subseteq \{s_1, \ldots, s_n\}$ where $|S| = t$,

$\text{Reconstruct}(S) = m$

Security: Need at least t shares to learn anything about m.

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

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

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

$$\{s_i \mid i \in I\} \approx \{s_i' \mid i \in I\}$$

Today, these dist will be identical.

Construction of (n,n)-secret sharing:

For message space $\mathcal{M} = \mathbb{F}$ and $S = \mathbb{F}$, can actually just be an abelian group

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

Output $(r_1, \ldots, r_n)$

Reconstruct$(r_1, \ldots, r_n)$: 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: $\forall m_0, m_1 \in \mathcal{M}$, the $(n-1)$ share distributions are actually identical!

* See if you can convince yourselves

Shamir Secret Sharing: ((t,n)-secret sharing scheme)

For $M = \mathbb{F}$, $S = \mathbb{F}$ s.t. $|\mathbb{F}| > n$.

Intuition: a polynomial of degree $t-1$ can be uniquely determined by t points. For example, a line by two points, a parabola by 3.

Linear Algebra Viewpoint:

Given a point $x \in \mathbb{F}$ and a poly $f(X) = c_0 + c_1 X + c_2 X^2 + \ldots + c_{t-1} X^{t-1}$.

We can view the evaluation $f(x)$ as an inner product

$$f(x) = \langle (1, x, x^2, \ldots, x^{t-1}), (c_0, \ldots, c_{t-1}) \rangle$$

Thus, given t distinct points, we can describe a linear system.

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{t-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{t-1} \\ & \vdots & & \ddots & \\ 1 & x_{t-1} & x_{t-1}^2 & \cdots & x_{t-1}^{t-1} \end{bmatrix}}_{\text{Vandermonde Matrix}} \underbrace{\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{t-1} \end{bmatrix}}_{\text{Coeffs}} = \underbrace{\begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{t-1}) \end{bmatrix}}_{\text{evals}}$$

Interpolating $f(X)$ is equivalent to solving the linear system above for the coeffs. For distinct $x_0, \ldots, x_{t-1}$, the Vandermonde matrix is invertible; thus, interpolating requires a matrix mul $V^{-1} \cdot \text{evals}^T$.

Proof Sketch:

The cols linearly independent:

$$c_0 \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} + c_1 \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{t-1} \end{pmatrix} + \ldots + c_{t-1} \begin{pmatrix} x_0^{t-1} \\ x_1^{t-1} \\ \vdots \\ x_{t-1}^{t-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Implies $\overbrace{x_0, \ldots, x_{t-1}}^{t}$ are roots of a poly of deg $\le t-1$, but a poly can have at most $t-1$ roots. Thus, $c_0, \ldots, c_{t-1} = 0$.

Construction

Share(m):
- Sample random coeffs $c_1, \dots, c_{t-1} \xleftarrow{\$} \mathbb{F}$.
- Define $f(X) := m + \sum_{i<t} c_i X^i$.
- Output n points on f: $(S_i := (i, f(i)) \; \forall i \in [1,n])$

Reconstruct $((x_i, y_i) \; \forall i \in [t])$:
- Interpolate the unique poly f of deg $t-1$ defined by those t points.
- Output $f(0)$

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

Security: Consider an arbitrary message $m \in \mathbb{F}$ and $I \subseteq [1,n]$ s.t. $|I| = t-1$. Define $\{x_i \mid \forall i \in [1,t-1]\} := I$. Consider arbitrary elts $y_1, \dots, y_{t-1} \in \mathbb{F}$. What's the probability the shares for this subset are $(x_1, y_1), \dots, (x_{t-1}, y_{t-1})$?

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

$$= \Pr\left[\begin{bmatrix} m \\ c_1 \\ \vdots \\ c_{t-1} \end{bmatrix} = V^{-1}(\dots) \begin{bmatrix} m \\ y_1 \\ \vdots \\ y_{t-1} \end{bmatrix} \right] = \frac{1}{|\mathbb{F}|^{t-1}}$$

independent of m

Another way to interpret, for any choice of $(t-1)$ shares and any message m, there exist a unique poly f of deg $t-1$ s.t.

$$\forall i \in [1, t-1], \; f(x_i) = y_i \text{ and } f(0) = m$$

Thus, any $(t-1)$ shares can be consistent with the sharing of any message m.

Now, we will describe a protocal for 2-PC (two-party MPC) for functions expressible by Arithmetic Circuits.

We will show an elegant construction from secret sharing that targets Semihonest Security, where we restrict corrupt parties to follow the protocol specification exactly, but try to extract info about the honest parties' input.

☆ Though there is an elegant way to make the protocol maliciously secure by using "info-theoretic macs"

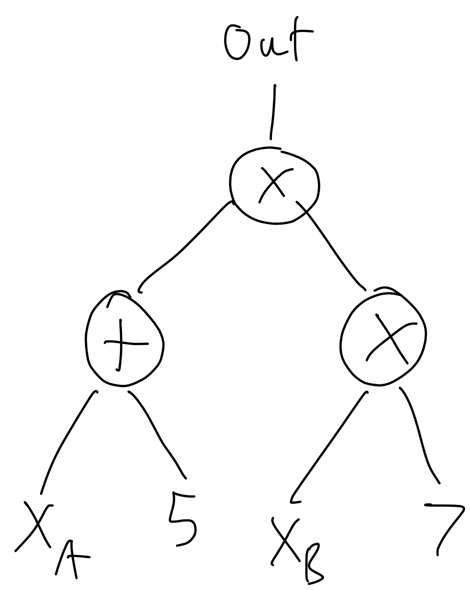
2-Party Computation for Arithmetic Circuits

	Alice(x_A)		Bob(x_B)
Share Inputs	$[x_A]_0, [x_A]_1 \leftarrow \text{Share}(x_A)$		$[x_B]_0, [x_B]_1 \leftarrow \text{Share}(x_B)$
		$\xrightarrow{[x_A]_1}$	
		$\xleftarrow{[x_B]_0}$	
Computation on Secret shares	$\vdots$		$\vdots$
	$[\text{Output}]_0$		$[\text{Output}]_1$
Opening final output shares		$\xrightarrow{[\text{Output}]_0}$	
		$\xleftarrow{[\text{Output}]_1}$	
Reconstruct Output		$[\text{Output}]_0 + [\text{Output}]_1 = \text{Output}$	

Protocol

Idea: Derive shares to intermediate wires incrementally

1. Both parties secret share their input elts with each other.
2. For each addition gate in the circuit with inputs $[x]$, $[y]$, the parties jointly derive shares to $[x+y]$ (the output share)
3. For each multiplication gate with inputs $[x]$, $[y]$ parties jointly derive shares to $[x \cdot y]$
4. Once share of circuit outputs is derived, each party sends their share of the output



Alice		Bob
$[x_A]_0, [x_A]_1 \leftarrow Share(x_A)$	$\xrightarrow{[x_A]_1}$	$[x_B]_0, [x_B]_1 \leftarrow \ldots$
$[x_A]_0, [x_B]_0$	$\xleftarrow{[x_B]_0}$	$[x_A]_1, [x_B]_1$
$\downarrow$		$\downarrow$
$[x_A+5]_0, [7x_B]_0$		$[x_A+5]_1, [7x_B]_1$
$\downarrow$	$\rightleftarrows$	$\downarrow$
$[(x_A+5)(7x_B)]_0$		$[(x_A+5)(7x_B)]_1$
$\parallel$		$\parallel$
$[out]_0$	$\xrightarrow{[out]_0}$ $\xleftarrow{[out]_1}$	$[out]_1$

$[out]_0 + [out]_1 = out$

Computation on Secret Shares

Here we assume the (2,2) scheme above $[X]_0 = r \xleftarrow{\$} \mathbb{F}$, $[X]_1 = X - r$.

Operations that do not require interaction

Adding Shares: $[X_A]_0 + [X_B]_0 = [X_A + X_B]_0$

$$[X_A]_1 + [X_B]_1 = [X_A + X_B]_1$$

$$[X_A + X_B]_0 + [X_A + X_B]_1 = [X_A]_0 + [X_A]_1 + [X_B]_0 + [X_B]_1 = X_A + X_B$$

Adding a constant c: $\frac{c}{2} + [X_B] = [c + X_B]$

Multipling by a constant c: $c[X_B] = [cX_B]$

Multipling Shares: Will require setup + interaction

Beaver's Trick 91

Suppose parties already have secret shares of a random product:

$\underbrace{[a], [b], [c]}_{\text{beaver triple}}$ where $a, b \xleftarrow{\$} \mathbb{F}$ and $c = ab$

To multiply shares $[x]$ and $[y]$:

from adding shares

1. Locally compute shares to $[x-a]$ and $[y-b]$.
2. Send shares to jointly reconstruct $\varepsilon = x - a$ and $\delta = y - b$

One time pad encryptions of x and y

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

Correctness

$$[z]_0 + [z]_1 = \begin{array}{l} [c]_0 + \delta[x]_0 + \varepsilon[y]_0 - \frac{\varepsilon\delta}{2} \\ \quad + \\ [c]_1 + \delta[x]_1 + \varepsilon[y]_1 - \frac{\varepsilon\delta}{2} \end{array}$$

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

$$= ab + (y-b)x + (x-a)y - (x-a)(y-b)$$

$$= \cancel{ab} + yx - \cancel{bx} + \cancel{xy} - \cancel{ay} - \cancel{xy} + \cancel{ay} + \cancel{bx} - \cancel{ab}$$

$$= xy$$

Security: Information-Theoretic!

How do the parties obtain beaver triples?

☆ Requires public-key cryptography or a trusted dealer.

- ↳ Oblivious Transfer
- ↳ Garbled Circuits
- ↳ Somewhat homomorphic Enc

} In HW!

Need to generate one beaver triple per mult gate (cannot reuse triples; o/w break OTP)

2-PC in the Preprocessing Model

Preprocessing / offline stage: Parties generate M beaver triples where M is an upper bound on multiplication gates. Note that this process is expensive, but independent of the future circuit / party input.

- Save these beaver triples for use in the future. Maybe waiting for data to be available or a computation to be agreed on.

Online Stage: No expensive PK operations, but requires communication linear in the # of mult gates / inputs

- parties secret share inputs
- parties perform the 2-PC protocol using pregenerated triples (as long as # mult gates $< M$)
- parties reconstruct output