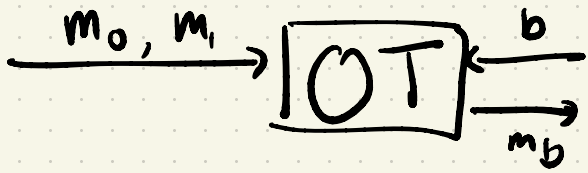
Secret Sharing

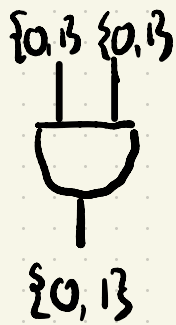
Last Time: Garbled Circuits

- Oblivious Transfer



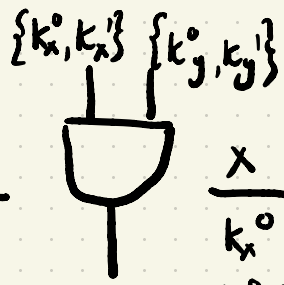
- Garbled Circuits Idea
  - Circuits over random pairs of symmetric keypairs

Normal Circuit



x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Garbled Circuit



x	y	z
k_x^0	k_y^0	$E(k_x^0, E(k_y^0, k_z^0))$
k_x^0	k_y^1	$E(k_x^0, E(k_y^1, k_z^0))$
k_x^1	k_y^0	$E(k_x^1, E(k_y^0, k_z^0))$
k_x^1	k_y^1	$E(k_x^1, E(k_y^1, k_z^1))$

$E(k_x^b, E(k_y^{b'}, k_z^{b \wedge b'}))$

- Garbled circuits protocol
  1. Alice sends "garbled" circuit to Bob
  2. Alice sends keys for her inputs to Bob
  3. Bob uses OT to get keys for his inputs
  4. Bob incrementally decrypts circuit to the ouput

Secure 2PC Definition:

correctness:

$$\Pr[\langle A(x), B(y)\rangle = f(x,y)] = 1$$

$$\begin{array}{ccc} A(x) & & B(y) \\ \downarrow & \rightleftarrows & \downarrow \\ f(x,y) & & f(x,y) \end{array}$$

semi-honest correctness:

$$\text{View}_A(\langle A(x), B(y)\rangle) \approx \text{Sim}_A(x, f(x,y))$$

$$\text{View}_B(\langle A(x), B(y)\rangle) \approx \text{Sim}_B(y, f(x,y))$$

Today: Towards <u>multi</u>-party computation...

$(n \geq 3)$

Primitive: secret sharing.

Secret Sharing

User holds a secret $\alpha \in \mathcal{X}$ wants to split into n shares:

$$(s_1, s_2, \ldots, s_n) \in \mathcal{S}^n$$

such that

$t \leq n$

- α is reconstructible from any t shares
- any $t-1$ shares reveal nothing about α.

Defn. A secret sharing scheme is a pair of algorithms:

- G (Generate) is randomized:

$$G(\alpha, t, n) \to (s_1, \ldots, s_n)$$

- C (reConstruct) is deterministic:

$$C(s_{i_1}, \ldots, s_{i_t}) \to \alpha$$

Properties:

- Correctness: $\forall \alpha \in \mathcal{X}$, $\forall n \in \mathbb{N}$, $\forall t \leq n$,
  $\forall$ distinct $i_1, \ldots, i_t \in [n]$
  $\{1, \ldots, n\}$

$$\Pr\left[\begin{array}{l} s_1, \ldots, s_n \leftarrow G(\alpha, n, t) \\ C(s_{i_1}, \ldots, s_{i_t}) = \alpha \end{array}\right] = 1$$

- Security: $\forall \alpha, \alpha' \in \mathcal{X}$, $\forall n \in \mathbb{N}$, $\forall t \leq n$
  $\forall S = \{i_1, \ldots, i_t\} \subset [n]$

$$\{\mathrm{View}_S(\alpha)\} \approx \{\mathrm{View}_S(\alpha')\}$$

where

$$\{\mathrm{View}_S(\alpha)\} = \{(s_{i_1}, \ldots, s_{i_t}) \mid s_1, \ldots, s_n \leftarrow G(\alpha, n, t)\}$$

Example: Additive Sharing

- A scheme for $t = n$.
- $\mathcal{X} = \mathbb{Z}_q$
- $G(\alpha, n, n)$:

  $s_2, \ldots, s_n \xleftarrow{\$} \mathbb{Z}_q^{\,n-1}$

  $s_1 \leftarrow \alpha - \sum_{i=2}^{n} s_i$

  output $(s_1, \ldots, s_n)$

- $C(s_1, \ldots, s_n)$:

  output $\sum_{i=1}^{n} s_i$

Example: Combinatorial Sharing

- Let (E, D) be a symmetric cipher with $\mathcal{X}$ as the message space
- $G(\alpha, n, t)$:
  - $k_1, \ldots, k_n \xleftarrow{\$} K$
  - for $S = \{i_1, \ldots, i_t\} \subset [n]$:

    $ct_S \leftarrow E(k_{i_1}, E(k_{i_2}, \ldots, E(k_{i_t}, \alpha) \ldots))$
  - $ct \leftarrow \{ct_S, \text{ for all } S\}$
  - output $s_i \leftarrow (ct, k_i)$ for $i \in [n]$
- $C(s_{i_1}, \ldots, s_{i_t})$: decrypt S.
- Note: computationally secure (because of the cipher)

Comparison:

Scheme:	Additive	Combinatorial
Supported t-values	$t = n$	$1 \le t \le n$
Security	Information Theoretic	Computational (cipher)
Share Size	$\lvert X \rvert$	$\binom{n}{t} \lvert C \rvert + \lvert K \rvert$ ↑ exponential in t

Can we support all $t \le n$ without exponential share size?

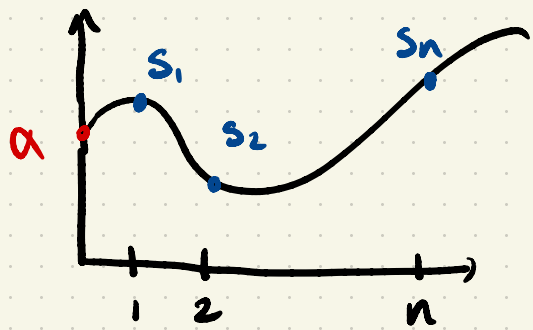
YES!

Do we need to give up info-theoretic security?

NO!

Shamir Secret Sharing

Idea



← random deg $< t$ poly p, with $p(0) = \alpha$

Generation: $\alpha \to$ polynomial $\to$ evaluations

Reconstruction: t evals $\to$ polynomial $\to \alpha$

$\to$ take $\mathbb{Z}_q$ to be a field (i.e., q is prime)

$G(\alpha, n, t)$: $a_1, \ldots, a_{t-1} \xleftarrow{\$} \mathbb{Z}_q$

define $p(X) = \alpha + a_1 X + \cdots + a_{t-1} X^{t-1}$

for $i \in [n]$: $s_i \leftarrow (i, p(i))$

output $(s_1, \ldots, s_n)$

$C(s_{i_1}, \dots, s_{i_t})$:

- we have

$$(x_1, y_1), \dots, (x_t, y_t)$$

with

$$x_i \neq x_j \quad \text{for all } i \neq j$$

- we want to recover

$$p(\underline{X}) \in \mathbb{F}^{<t}[\underline{X}]$$

such that

$$p(x_i) = y_i \quad \text{for } i \in [t]$$

(then, $\alpha = p(0)$).

This problem (finding a degree-$<t$ polynomial which agrees with t evaluations) is called

<u>Interpolation</u>

Interpolation, take 1: Linear Algebra

Let

$$p(\underline{x}) = \alpha + a_1 \underline{x}^1 + \cdots + a_{t-1} \underline{x}^{t-1},$$

then we want:

$$\left.\begin{aligned} p(x_1) &= \alpha + a_1 x_1 + \cdots + a_{t-1} x_1^{t-1} = y_1 \\ &\vdots \\ p(x_t) &= \underbrace{\alpha + a_1 x_t + \cdots + a_{t-1} x_t^{t-1}}_{t \text{ unknowns.}} = y_t \end{aligned}\right\} t \text{ eqns.}$$

Matrix form:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{t-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{t-1} \\ \vdots & \vdots & & \ddots & \\ 1 & x_{t-1} & x_{t-1}^2 & & x_t^{t-1} \end{bmatrix} \begin{bmatrix} \alpha \\ a_1 \\ \vdots \\ a_{t-1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_t \end{bmatrix}$$

"Vandermonde Matrix", $V(x_1, x_2, \ldots, x_t)$

- invertible for distinct x_i
- $O(n^2)$ time
- $O(n \log n)$ time if x_i are a multiplicative subgroup (use FFT)

not useful here ($t<n$) but good to know

Interpolation, take 2: Lagrange Bases

Given a set of inputs,

$$S = \{x_1, \dots x_t\}$$

the ith lagrange basis polynomial is

- zero at all $x_j \neq x_i$
- one at x_i

defined as:

$$\ell_i(X) = \frac{\prod_{j \in [t] \setminus \{i\}} (X - x_j)}{\prod_{j \in [t] \setminus \{i\}} (x_i - x_j)}$$

We compute

$$p(X) = \sum_{i=1}^{t} y_i \, \ell_i(X)$$

← polynomial sum!

Security of Shamir Secret Sharing

Let $\alpha \in \mathbb{Z}_q$, let $S \subset [n]$, $|S| = t-1$

$\{\text{View}_S(\alpha)\}$ is $\{(y_1, y_2, \dots, y_{t-1})\}$

Claim: $\{\text{View}_S(\alpha)\} = \text{uniform}(\mathbb{Z}_q^{t-1})$

Corollary: $\{\text{View}_S(\alpha)\} = \{\text{View}_S(\alpha')\}$

Proof: $\Pr[(y_1, \dots, y_{t-1}) = (\hat{y}_1, \dots, \hat{y}_{t-1})] = \frac{1}{q^{t-1}}$ want to show

over $a_1, \dots, a_{t-1}$

$$= \Pr[(\alpha, y_1, \dots, y_{t-1}) = (\alpha, \hat{y}_1, \dots, \hat{y}_{t-1})]$$

$$= \Pr\left[V(0, x_1, \dots, x_{t-1}) \begin{bmatrix} \alpha \\ a_1 \\ \vdots \\ a_{t-1} \end{bmatrix} = \begin{bmatrix} 0 \\ \hat{y}_1 \\ \vdots \\ \hat{y}_{t-1} \end{bmatrix} \right]$$

invertible

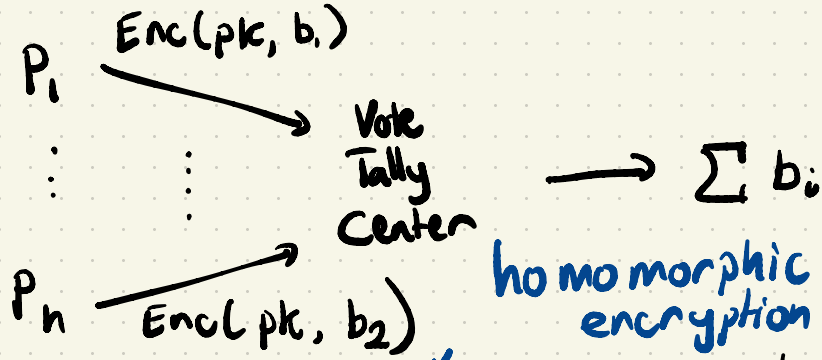
$$= \Pr\left[\begin{bmatrix} \alpha \\ a_1 \\ \vdots \\ a_{t-1} \end{bmatrix} = \underbrace{V^{-1}(\quad) \begin{bmatrix} \alpha \\ \hat{y}_1 \\ \vdots \\ \hat{y}_{t-1} \end{bmatrix}}_{\text{a fixed } \hat{a} \in \mathbb{F}^n} \right]$$

$$= \frac{1}{|\mathbb{Z}_q|^{t-1}}$$

uniform! $\boxtimes$

Application: Threshold Decryption for Private Polls

Reminder (lecture 10):



homomorphic encryption

Idea: sum votes, then decrypt.

→ Also, P_i <u>proves</u> $b_i \in \{0,1\}$

Assumption: VTC is honest

→ Doesn't decrypt single votes!

↳ can we relax this?

Idea: t-of-n <u>threshold encryption</u>.

→ secret key has n parts

→ split among n parties

→ encryption is normal

→ t parties interact to decrypt.

(similar: threshold signatures)

Recap: El - Gamal

$\text{Gen}() \to (pk, sk)$:

$sk \xleftarrow{\$} \mathbb{Z}_q$

$pk \leftarrow g^{sk}$

output (sk, pk)

$\text{Enc}(pk, m \in \mathbb{Z}_q) \to c$:

$r \xleftarrow{\$} \mathbb{Z}_q$

output $(g^r, \underbrace{pk^r g^m}_{\text{OTP}})$

can get OTP from with sk

$\text{Dec}(sk, c) \to m$:

$(g^r, pk^r g^m) \leftarrow c$

output $\log_g\left(\dfrac{pk^r g^m}{(g^r)^{sk}}\right)$

easy if m is small.

Threshold El-Gamal: n-of-n

- Addively share sk into $sk_1, \ldots, sk_n$

  $\left(\sum_i sk_i = sk \right)$

- Decryption Protocol

  All servers have $(g^r, pk^r g^m)$

  each publicizes:

$$(g^r)^{sk_i}$$

  then we compute:

$$(g^r)^{sk} \leftarrow \prod_i (g^r)^{sk_i}$$

$$m \leftarrow \log_g \left(\frac{pk^r g^m}{(g^r)^{sk}} \right)$$

t-of-n El Gamal

1. Secret-share sk into (x_i, y_i)
   (using Shamir sharing)

2. Decryption $(g^r, pk^r g^m)$:

   Each server publicizes:

$$(x_i, (g^r)^{y_i})$$

   One server determines
   $\vec{R}$, such that

$$\alpha = \langle \vec{R}, \vec{y} \rangle$$

   1st row of $V^{-1}(x_1, \ldots, x_t)$
   a function of $x_1, \ldots, x_t$

   computes:

$$w \leftarrow \prod_i \left((g^r)^{y_i}\right)^{R_i}$$

   (can show $= g^{r \cdot sk}$)

   output $m \leftarrow \log_g\left(\frac{sk^r g^m}{w}\right)$