Privacy Enhancing Technologies

2. MPC

Recap on commitments

$$Commit : M \times R \to C$$

- $\{Commit(m_0, r)\} \approx \{Commit(m_1, r)\}$ — Hiding
- Can't find $m_0 \neq m_1$ s.t. $Commit(m_0, r_0) = Commit(m_1, r_1)$ — Binding

Pedersen

$g, h \in G$ (prime order p), $g^x = h$ (unknown)

$$Commit(m, r) = g^m h^r$$

$m, r \in \mathbb{Z}_p$

Security

- **Hiding** (perfectly)

  given $c = g^{\alpha} = g^{m} h^{r} = g^{m+xr}$

  pick msg m_0, then $\exists r_0$ s.t
  $Commit(m_0, r_0) = c$ ↳ $\frac{\alpha - m}{x}$

- **Binding** (computational)

Hardness of DLog:

Given $\mathbb{G}$, generator g, $h \in \mathbb{G}$
hard to find x s.t $g^x = h$

How to prove security?

Hardness reduction

"if I can break binding
then I can break dLog

Assume break binding.

I can find $m_0 \neq m_1, r_0, r_1$
s.t $g^{m_0} h^{r_0} = g^{m_1} h^{r_1} = c$

$$g^{m_0 + x \cdot r_0} = g^{m_1 + x \cdot r_1} = c$$

$$m_0 + x \cdot r_0 = m_1 + x \cdot r_1$$

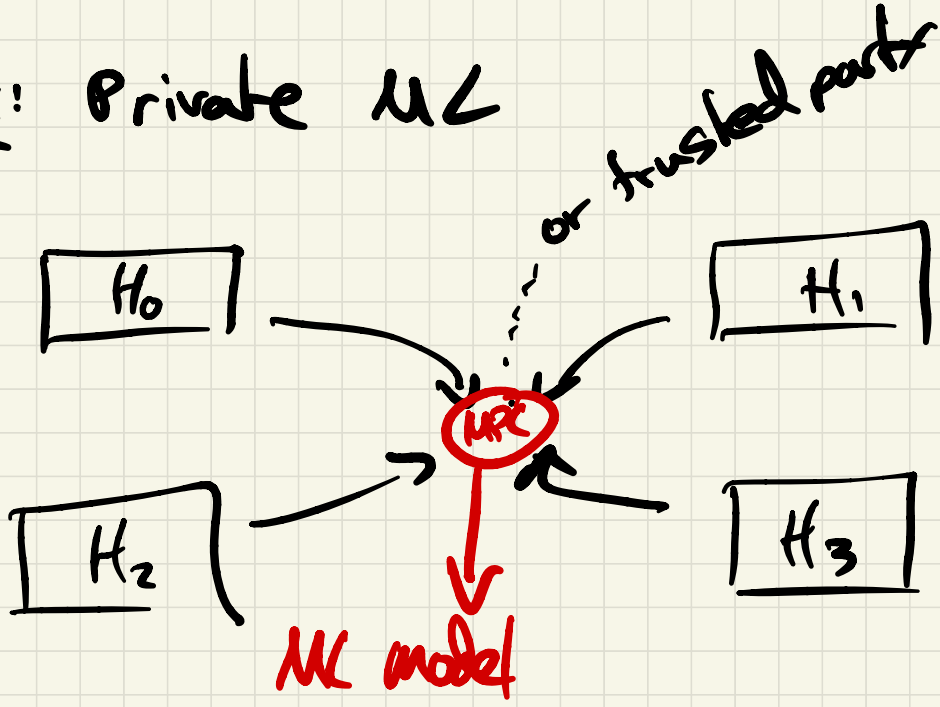
$$x = \frac{m_1 - m_0}{r_0 - r_1}$$

I wrote this the wrong way around in class. Fixed it.

Secure Multiparty Computation (MPC)

"Any function that can be computed securely with a trusted 3rd party, can also be computed securely without"

Ex: Private ML



Defining MPC

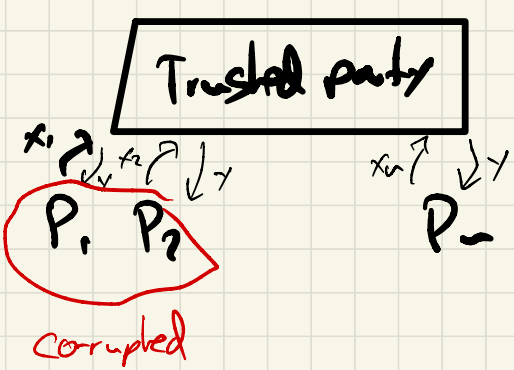
$y \leftarrow f(x_1, \ldots, x_n)$

private input of party 1

very general setup:

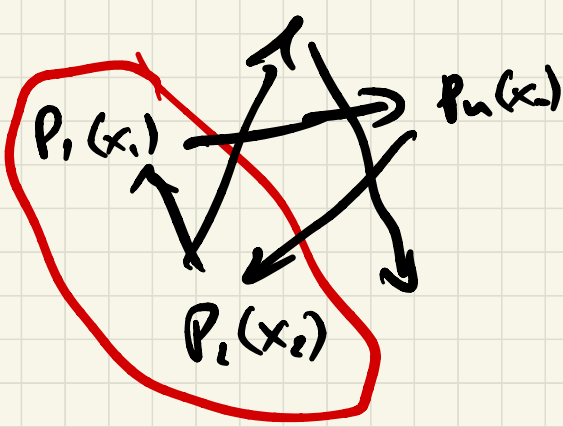
* e-voting
* secure group messaging
* private auctions
* ...

Ideal world



$\approx$

Real World



$$\mathrm{Sim}\left(C, \{x_i : i \in C\}, y = f(x_1, \ldots)\right) \approx \{\mathrm{View}_i : i \in C\}$$

- Sim: simulator
- $\{x_i : i \in C\}$: inputs of adv
- $y = f(x_1, \ldots)$: output of function
- $\{\mathrm{View}_i : i \in C\}$: corrupt parties

everything the adv sees in the ideal world

Security for MPC

- Semi-honest: the protocol is secure as long as all parties follow the protocol
- Malicious: the adv. can do whatever they want !!

When we build MPC, we start with

semi-honest and then add checks & balances

↳ very efficient, simple

for malicious security

↳ slow, complicated

Building a semi-honest secure MPC

Paradigm:

1. Parties "secret-share" their inputs
2. Parties compute over secret-shared data

Additive Secret Sharing

- secret $s \in \mathbb{F}_p$ (say $\mathbb{Z}_p$) to be shared among n parties
- sample $r_1, \ldots, r_{n-1} \xleftarrow{\$} \mathbb{F}_p$ and set

$$r_n = s - \sum_{i=1}^{n-1} r_i$$

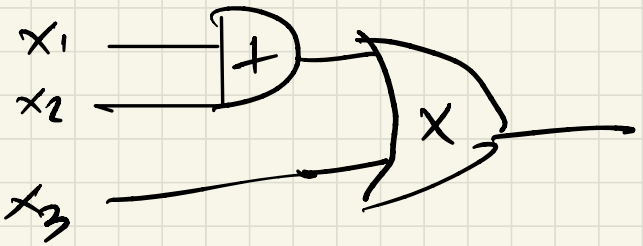
$$s = \sum_{i=1}^{n} r_i$$

$$[s] = (r_1, \ldots, r_n)$$

any subset of $n-1$ shares leaks nothing about s

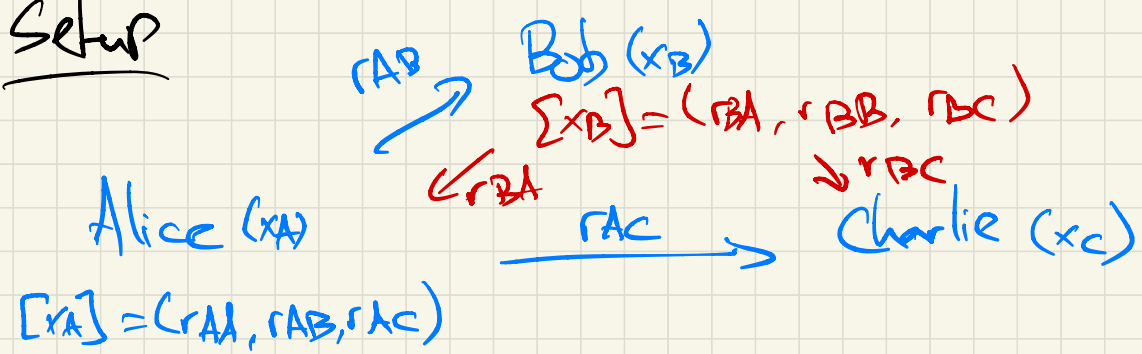
What are we going to compute?

Def: Arithmetic circuits, inputs $x_i \in \mathbb{F}_p$

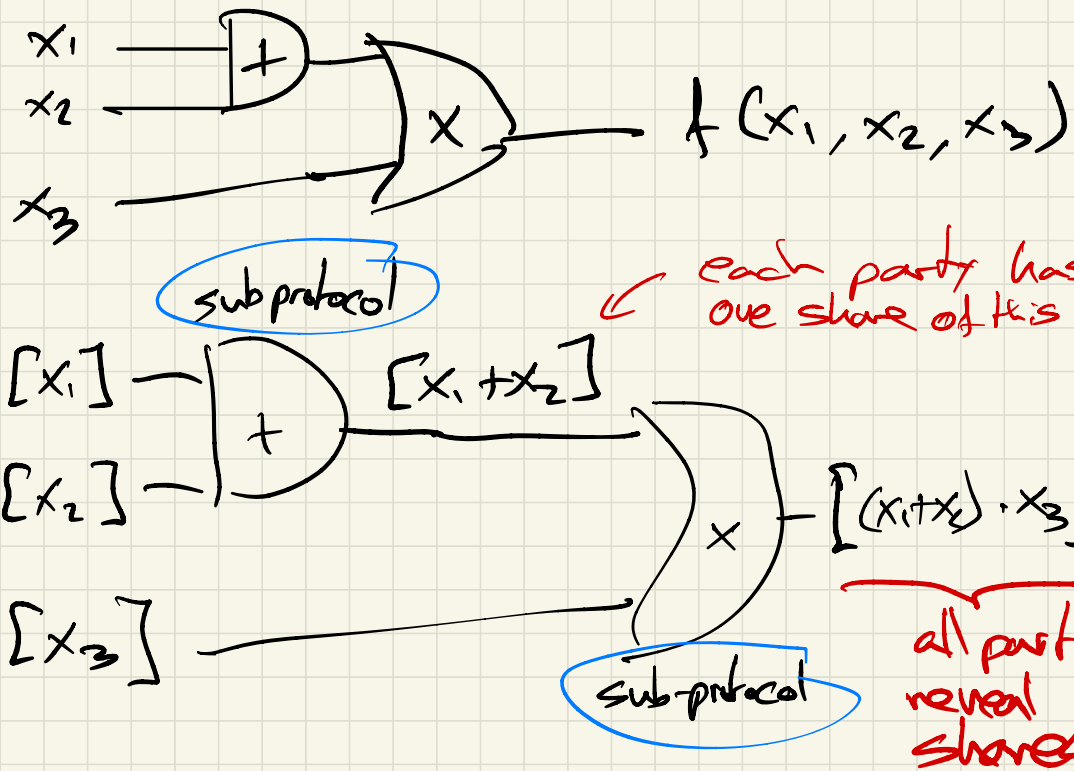


- Any function f can be represented as an arithmetic circuit
- Very convenient repr. for crypto

Setup



Observation: To get semi-honest MPC, we just need to do additions & mult. over secret-shared data



"Bad" example

suppose $x_2 = 0$, then we don't want $[x_1 + x_2] = [x_1]$

Addition: $[x]$, $[y]$ → $(+)$ → $[x+y]$

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

$(r_1, \dots, r_n)$ $(x_1, \dots, x_n)$ $(y_1, \dots, y_n)$

$$\hookrightarrow \sum_{i=1}^{n} r_i = \sum_{i=1}^{n} (x_i + y_i) = \underbrace{\sum_{i=1}^{n} x_i}_{x} + \underbrace{\sum_{i=1}^{n} y_i}_{y}$$

Linear ops

↙ constant known to all parties

- $[k \cdot x] = k \cdot [x]$
- $[x + k] = (x_1 + k, \dots, x_n)$

Multiplication

BAD WRONG

$$[x \cdot y] = [x] \circ [y]$$

$$= (x_1 \cdot y_1, x_2 \cdot y_2, \ldots, x_n \circ y_n)$$

$$\sum x_i \cdot y_i \neq \left(\sum x_i\right) \cdot \left(\sum y_i\right)$$

We cannot do this without interaction

How to compute $[x \cdot y]$?

1) information-theoretically
   - less than $n/2$ parties are corrupt

"honest majority"

⇒ Semi-honest information-theoretic MPC

2) Computational
   - what if $n = 2$?
   - need public-key cryptography

⇒ semi-honest MPC with $n-1$ corrupted parties

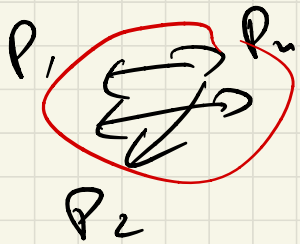
What does a "real" MPC protocol look like?

1) "Offline" phase
   - doesn't depend on inputs $x_1, \ldots, x_n$
   - "expensive" (uses crypto)

2) Online phase
   - depends on inputs
   - "cheap", information-theoretic

---

offline phase: generate "mult. triples"



$\Rightarrow$

$P_1\ (a_1, b_1, c_1)$
$P_2\ (a_2, b_2, c_2)$
$\vdots$
$P_n\ (a_n, b_n, c_n)$

$a, b \xleftarrow{\$} \mathbb{F}_p$

s.t. $\left(\underbrace{\sum a_i}_{a}\right) \cdot \left(\underbrace{\sum b_i}_{b}\right) = \underbrace{\sum c_i}_{c}$

Online: when we need to mult. $[x], [y]$ we can reduce this to mult. $[a], [b]$ for random a, b

we already have this !

"Beaver's trick"

$[x] = (x_1, \dots, x_n)$, $[a] = (a_1, \dots, a_n)$

$[y] = (y_1, \dots, y_n)$, $[b] = (b_1, \dots, b_n)$

$[c] = (c_1, \dots, c_n)$

1. $[\delta] = [x - a]$ all parties have shares of δ and ε
   $[\varepsilon] = [y - b]$

2. All parties reveal shares of δ, ε
   $\Rightarrow$ All parties know $\delta = x - a$, $\varepsilon = y - b$

3. locally compute: $z_i = \frac{\delta \cdot \varepsilon}{n} + b_i \cdot \delta + a_i \cdot \varepsilon + c_i$

$$z_i = \frac{\delta \cdot \varepsilon}{n} + b_i \cdot \delta + a_i \cdot \varepsilon + c_i$$

– garbage

Claim: $[z] = (z_1, \ldots, z_n)$ is a secret-share of $[xy]$

$$\sum_{i=1}^{n} z_i = \sum \frac{\delta\varepsilon}{n} + \sum b_i \cdot \delta + \sum a_i \cdot \varepsilon + \sum c_i$$

$$\delta\varepsilon + b \cdot \delta + a \cdot \varepsilon + c$$

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

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

$$\varepsilon \cdot \varepsilon = (x-a)(y-b)$$

$$= xy + \text{garbage}$$

- Computation : $O(|C|)$
- Communication : $O(\#\text{ of mult. gates})$