Secure Multiparty Computation

All primitives we've seen so far allow 2 distrustful parties to securely compute some function of their respective data

Ex:

FHE

client: data x — server data: $\emptyset$

$f(x) \leftarrow f(x, \emptyset) \rightarrow 1$

ZK

client: x — prover: w

1 if witness valid, 0 otherwise $\leftarrow f(x, w) \rightarrow 1$

Main result: Any function that can be computed with the help of a trusted 3rd party can also be computed without it.

Formal model:

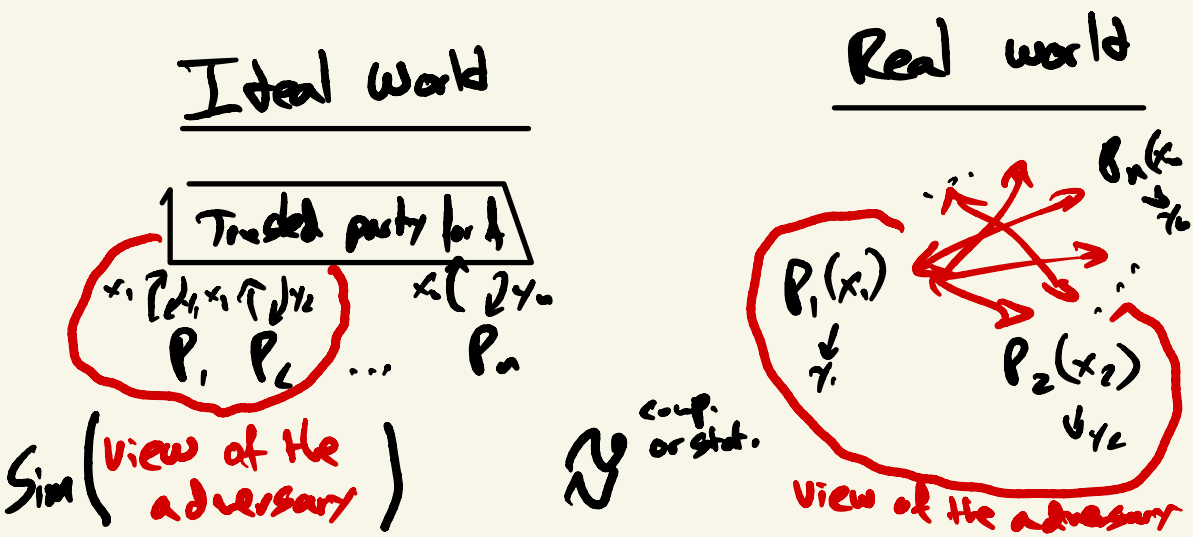
n parties $P_1, \dots, P_n$ with inputs $x_1, \dots, x_n$

who want to compute

$$y_1, \dots, y_n \leftarrow f(x_1, \dots, x_n)$$

Security (informally): Consider an Alr that "corrupts" some number of parties and should learn no more than it could have learned if we were using a trusted third party

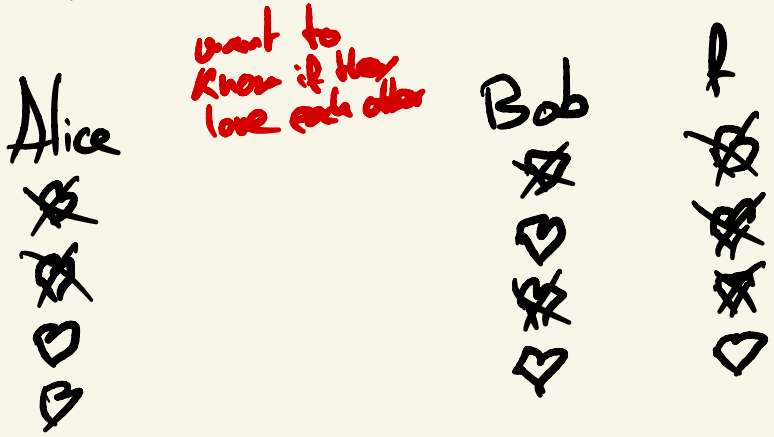
the real vs ideal paradigm (simulation paradigm)



What about what the function leaks?

This is fine in the MPC model

Eg.



The function leaks something

for ex: if Alice loves Bob, she learns Bob's input

the privacy: if Alice does not love Bob she learns nothing

Security models for MPC

1. Semi-honest security: the Adv follows the protocol, but it tries to learn as much as possible from the messages it receives.

2. Malicious security: the Adv can deviate from the protocol as it wants

Formalize semi-honest MPC

Def: A protocol securely computes f in the presence of a semi-honest adversary if $\exists$ Sim s.t. $\forall x_1, \dots, x_n$ and any set $C \subseteq [n]$ of corrupt parties:

$$\text{Sim}\left(C, \{x_i : i \in C\}, y \in f(x_1, \dots, x_n)\right) \overset{c}{\approx} \{\text{view}_i : i \in C\}$$

ideal world — real world

GMW Protocol

Idea: parties "secret share" their data and then we compute over secret shares

Model: $x_i \in \mathbb{F}_p$, f is an arithmetic circuit over $\mathbb{F}_p$ $y \leftarrow C(x_1, \ldots, x_n)$

Def: Additive Secret Sharing: To share $x \in \mathbb{F}_p$

sample $s_1, \ldots, s_{n-1} \xleftarrow{R} \mathbb{F}_p$ and set $s_n = x - \sum_{i=1}^{n-1} s_i$

party i: gets share s_i

Notation: $[x] = (s_1, \ldots, s_n)$ where $\sum_{i=1}^{n} s_i = x$

Lemma: Given any subset of $n-1$ shares you learn nothing about x (info. theoretic)

Proof: the $n-1$ parties can sum up their shares to get $x - s_j$, which is uniformly random over $\mathbb{F}_p$

The protocol:

1. Each party samples shares of their input $[x_i]$ and distributes $n-1$ shares to the other parties

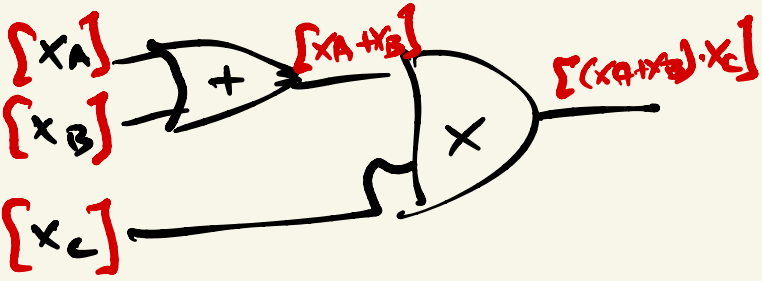
2. For each addition gate, run a sub protocol
   $[x], [y] \Rightarrow [x+y]$

3. For each mult. gate, run a sub protocol
   $[x], [y] \Rightarrow [xy]$

   each party has a share of the output

4. Each party publishes their share of the output and everyone reconstructs it

Ex: $f(x_A, x_B, x_C) = (x_A + x_B) \cdot x_C$



A	B	C	Σ
x_{AA}	x_{AB}	x_{AC}	x_A
x_{BA}	x_{BB}	x_{BC}	x_B
x_{CA}	x_{CB}	x_{CC}	x_C
s_1	s_2	s_3	x_{A+B}
s_1'	s_2'	s_3'	$(x_A+x_B)\cdot x_C$

we split

Addition

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

$[x] = (x_1, \ldots, x_n)$ $\quad \sum x_i = x$

$[y] = (y_1, \ldots, y_n)$ $\quad \sum y_i = y$

$[x+y] = (x_1+y_1, \ldots, x_n+y_n)$ $\quad \sum x_i+y_i = \sum x_i + \sum y_i = x + y$

Constants

for a public constant k,

$[kx] = (kx_1, \ldots, kx_n)$ $\quad \sum kx_i = k \cdot x$

$[x+k] = (x_1+k, x_2, \ldots, x_n)$ $\quad (\sum x_i) + k = k+x$

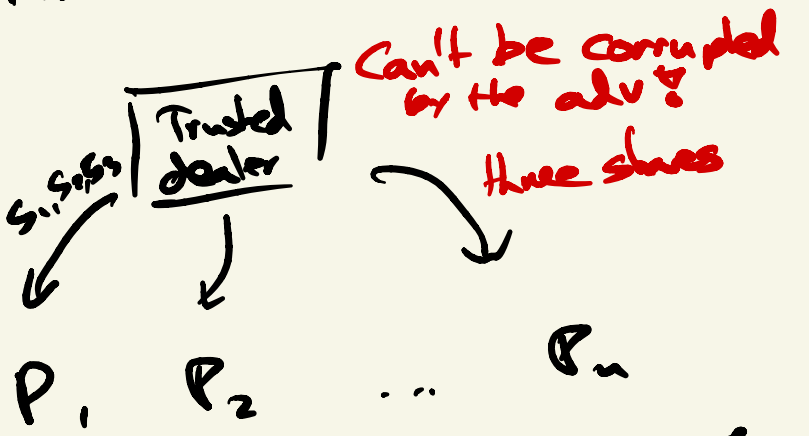
Multiplication gates

$$[xy] \neq [x] \cdot [y] = [x_1y_1, \ldots, x_ny_n]$$

$$\sum x_iy_i \neq (\sum x_i)(\sum y_i)$$

all cross terms are missing

Warm-up: Assume a "trusted dealer party"



parties have shares $[a], [b], [c]$ — multiplication triple

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

$a \cdot b = c$

a, b, c are unknown to all parties

Insight (Beaver '92): multiplying $[x], [y]$ can be reduced to multiplying random values $[a], [b]$, $a, b \xleftarrow{\$} \mathbb{F}$

In math, if you don't know what to do add and substract the same value

$$\frac{x}{y} \Rightarrow \frac{x-a+a}{y}$$

AHA! I don't understand this

(*) $xy = ((x-a)+a)((y-b)+b)$

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

Insight: if parties have $(x-a), (y-b)$ and $[a], [b], [ab]$

they can compute $[xy]$

Protocol:

1. from $[x], [y], [a], [b]$ parties create $[x-a], [y-b]$
2. parties open $x-a, y-b$ AHA
3. parties compute $[xy]$ using (*)

Why is releasing $x-a$, $y-b$ ok?

These are one-time pads ▽₀

Corollary: you can only use a mult. triple $[a],[b],[c]$ once ▽₀

How to get rid of the trusted dealer?

- MPC for general circuits
  $\Rightarrow$ reduced to MPC for a special "low-degree" circuit
- How to generate multiplication triples?
  ⊗ Needs public-key crypto

In modern MPC protocols (eg. SPDZ) we use somewhat homomorphic encryption

Analysis of GMW: (in the semi-honest setting)

* Security (information theoretic in the online phase)
  against any $n-1$ corrupt parties

* offline phase: we use PKC to create the Beaver triples

Malicious security

What can an adversary do?

* reveal incorrect output shares
* use incorrect shares at arbitrary points
* drop messages / replay messages
* can wait until all honest parties reveal their output share and then go on vacation!? "fairness"

* "Input independence": the adv. can't choose their input as a function of someone else's

Blueprint: run GMW (semi-honest) with "integrity checks"

* originally: ZK proofs
* "modern": "authenticated shares"

Beyond Additive Secret Sharing

- Additive: n-out-of-n secret-sharing
  - ↳ all parties need to be online to reconstruct
  - ↳ a single party can mess things up

Def: t-out-of-n secret sharing

$Gen(n, t, x) \rightarrow$ shares $s_1, \ldots, s_n$

$Recons(s_{i_1}, \ldots s_{i_t})$ takes t shares and outputs x

Properties

- Correctness: $\forall \alpha \in \mathcal{Z}$, any set of n shares output by Gen, and any subset of t shares, $Recons \to \alpha$
- Privacy: $\forall \alpha, \alpha' \in \mathcal{Z}$ and any subset $S \subseteq [n]$ of size $t-1$:

$$Gen(n,t,\alpha)[S] \equiv Gen(n,t,\alpha')[S]$$

Shamir's secret sharing

Let p be a prime, $\alpha \in \mathbb{Z}_p$

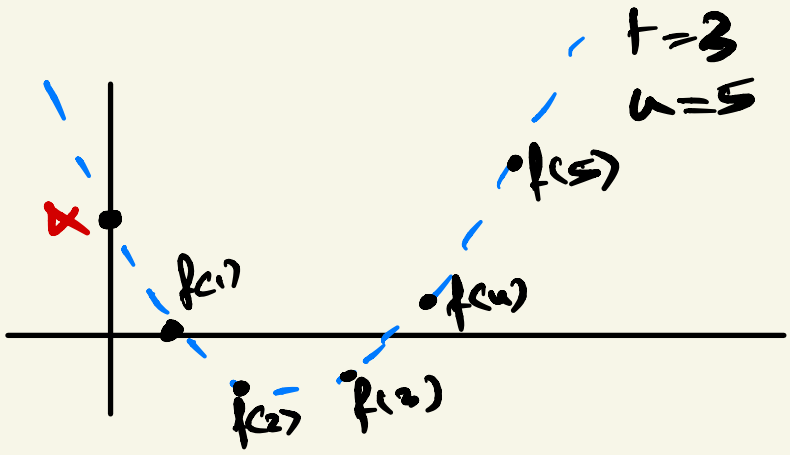
- $Gen(n,t,\alpha)$: choose coeffs $a_1, \dots, a_{t-1} \xleftarrow{\$} \mathbb{Z}_p$ and define: $f(x) = \alpha + a_1 x + \dots + a_{t-1} x^{t-1} \in \mathbb{Z}_p[x]$

$\to \deg(f) \le t-1$

$\to f(0) = \alpha$

Shares $(s_1, \dots, s_n)$: $s_i = (i, f(i))$

Recons($s_{i_1}, \ldots, s_{i_t}$) : interpolate a poly. f' of degree $t-1$ through these t points and output $f'(0)$



Lemma: For any set of d points $(x_1, y_1), \ldots, (x_d, y_d) \in \mathbb{Z}_p^2$ s.t $x_i \neq x_j$ there exists a unique polynomial of degree $\leq d-1$ s.t $f(x_i) = y_i \ \forall i = 1, \ldots, d$

(if p is prime)

Proof

let $f(x) = a_0 + a_1 x + \dots + a_{d-1} x^{d-1}$

be a poly of degree $d-1$

we want d unknowns

d eqns:

$$\begin{aligned} f(x_1) &= a_0 + a_1 x_1 + \dots + a_{d-1} x_1^{d-1} = y_1 \\ f(x_2) &= a_0 + a_1 x_2 + \dots + a_{d-1} x_2^{d-1} = y_2 \\ &\vdots \\ f(x_d) &= a_0 + a_1 x_d + \dots + a_{d-1} x_d^{d-1} = y_d \end{aligned}$$

$$Az = b$$

rewrite:

(*)
$$\underbrace{\begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{d-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{d-1} \\ \vdots & & & & \\ 1 & x_d & x_d^2 & \dots & x_d^{d-1} \end{pmatrix}}_{V(x_1, \dots, x_d)} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{d-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{pmatrix}$$

"Vandermonde"

Fact: $\det(V(x_1, \dots, x_d)) = \prod_{1 \le i < j \le d} (x_j - x_i)$

Linear Algebra 101 :

$\circledast$ has a unique solu $\iff$ $\det(V) \neq 0$

$\iff$ x_i are all distinct

in particular : $(a_0, a_1, \ldots, a_{n-1}) = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \cdot V^{-1}(x_1, \ldots, x_n)$

- Correctness ✓
- Privacy (proof in lecture notes)
- Neat : Polynomial interpolation is linear in the shares

  $\Rightarrow$ can compute "homomorphically" over shares