CS 355 Lecture 7 (4/23)

Logistics: HW2 due this Friday at 5pm (submit via Gradescope)
HW3 posted this Friday
See Piazza for office hour information for upcoming week (course staff is traveling)

Previous lecture: 2-party computation (Yao's garbled circuits)

Suppose Alice and Bob want to compute function f on joint input (x, y)

Alice (x) — Bob (y)

OT requests for x → (requires algebraic assumptions (more expensive))

OT responses (only requires cheap symmetric primitives):
- garbled circuit for f
- garbled inputs for y

← $f(x,y)$

HW2: Realize large number of OTs from a small ($\sim\lambda$) number of base OTs

Security: neither party learns more from protocol other than what is revealed by $f(x,y)$

e.g. $\{\text{View}_A(x,y)\} \stackrel{c}{\approx} \{S(1^\lambda, x, f(x,y)\}$

- $\text{View}_A(x,y)$: view of Alice in protocol execution on inputs x and y
- $S(1^\lambda, x, f(x,y))$: real protocol transcript can be simulated just given party's input and output of computation

Question: What if we have more than two parties?

This lecture: Secret sharing and MPC
Beaver triples and MPC in the preprocessing model

Secret sharing: Suppose we have a secret and want to distribute it among n parties such that any t of them can subsequently recover the secret and any $(t-1)$ subset cannot [e.g., Board of directors at Coca-Cola want to protect Coca-Cola recipe]

Def. A (t,n)-secret sharing scheme over a message space $\mathcal{M}$ and share space $\mathcal{S}$ consists of two efficient algorithms:

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

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

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 the secret

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

$\{(s_1, \ldots, s_n) \leftarrow \text{Share}(m_0) : s_i \text{ for all } i \in I\} \equiv \{(s_1, \ldots, s_n) \leftarrow \text{Share}(m_1) : s_i \text{ for all } i \in I\}$

(can relax to computational indistinguishability (in which case, Share takes in additional security parameter); ≡ → notion here is information-theoretic)

Examples: Additive secret sharing [n out of n]: $\mathcal{M} = \mathbb{Z}_p$ (any ring will work (p need not be prime)), $\mathcal{S} = \mathbb{Z}_p^n$

Provides information-theoretic security — check this!

- Share(m): Sample $r_1, \ldots, r_{n-1} \xleftarrow{R} \mathbb{Z}_p$ and set $r_n = m - \sum_{i=1}^{n-1} r_i \in \mathbb{Z}_p$
- Reconstruct$(r_1, \ldots, r_n)$: Output $\sum_{i=1}^{n} r_i$

Combinatorial secret sharing [t out of n]: Will use a symmetric encryption scheme (satisfying CPA-security) over $\{0,1\}^*$ (e.g. AES-CTR)

- Share(m): Sample n keys $k_1, \ldots, k_n$ for encryption scheme

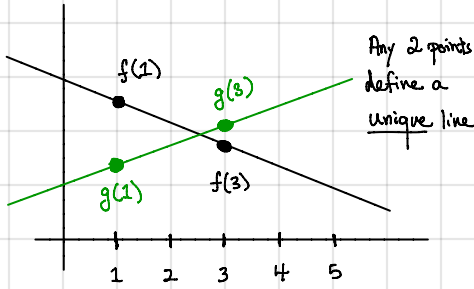
  For every t-subset $\{i_1, \ldots, i_t\} \subseteq [n]$, encrypt m using $k_{i_1}, \ldots, k_{i_t}$ (e.g. $\mathrm{Enc}(k_{i_1}, \mathrm{Enc}(k_{i_2}, \cdots, \mathrm{Enc}(k_{i_t}, m) \cdots))$)

  Let $\{ct\}$ be the collection of ciphertexts ← very large shares! Question: Can we do better?

  Output shares $\big((k_1, \{ct\}), \ldots, (k_n, \{ct\})\big)$
- Reconstruct $\big((k_1, \{ct\}), \ldots (k_t, \{ct\})\big)$: by construction, there is one $ct \in \{ct\}$ encrypted under $k_1, \ldots, k_t$ (relies on computational assumptions), so decrypt accordingly

Shamir secret sharing [t out of n]: $\mathcal{M} = \mathbb{F}_p$, $\mathcal{S} = \mathbb{F}_p^n$ (require $p > n$)

↳ beautiful construction based on polynomials (very useful mechanism for sharing data)



Key idea: Any d points uniquely define a degree-$(d-1)$ polynomial over a field

- e.g. 2 points define a line, 3 points define a parabola, etc.
- given d points, can efficiently obtain entire polynomial [Lagrange interpolation]

- Share(m): Choose $y_1, \ldots, y_{t-1} \xleftarrow{R} \mathbb{F}_p$

  Let $f: \mathbb{F}_p \to \mathbb{F}_p$ be the unique polynomial of degree $t-1$

  $f(0) = m$ and $f(i) = y_i \quad \forall i \in [t-1]$ [t points uniquely define the polynomial f]

  Output shares $(i, f(i)) \quad \forall i \in [n]$ Each share is just 2 field elements (independent of threshold t or # parties n).
- Reconstruct$\big((i_1, y_1), \ldots, (i_t, y_t)\big)$: Interpolate the unique polynomial f of degree $(t-1)$ defined by the points $(i_1, y_1), \ldots, (i_t, y_t)$

  Output $f(0)$

Correctness: Follows by uniqueness of the interpolating polynomial

Security: Given any subset of $(t-1)$ shares $(i_1, y_1), \ldots, (i_{t-1}, y_{t-1})$, and any message $m \in \mathbb{F}_p$, there is a unique polynomial f of degree $t-1$ where

$$f(i_1) = y_1, \ \ldots, \ f(i_{t-1}) = y_{t-1} \quad \text{and} \quad f(0) = m$$

Thus, any $(t-1)$ shares can be consistent with secret-sharing of any message m $\Rightarrow$ information-theoretic security

Efficiency: Secret sharing and secret reconstruction correspond to polynomial evaluation and interpolation (over $\mathbb{F}_p$). Both of these operations can be expressed as linear functions:

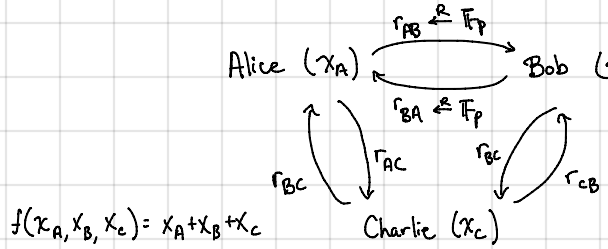
- Suppose $f(x) = \alpha_0 + \alpha_1 x + \cdots + \alpha_{t-1} x^{t-1}$
- Evaluation of f at points $x_1, x_2, \ldots, x_n$ can be expressed as matrix-vector product:

$$\underbrace{\begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & \cdots & x_1^{t-1} \\ 1 & x_2 & x_2^2 & x_2^3 & \cdots & x_2^{t-1} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & \cdots & x_n^{t-1} \end{pmatrix}}_{X \text{ ("Vandermonde matrix")}} \underbrace{\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix}}_{\alpha} = \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{y}$$

in particular, $y_i = f(x_i) \quad \forall i \in [n]$

- Polynomial evaluation corresponds to computing product $X \cdot \alpha$ and interpolation corresponds to computing the product $X^{-1} y$
- Both of these operations can be done in time $O(n \log n)$ using the Fast Fourier Transform (FFT)

Computing on secret-shared data: Another paradigm for 2PC (and MPC) – better-suited for evaluating arithmetic circuits (gates are addition and multiplication over $\mathbb{F}_p$)



Alice: Chooses $r_{AB}, r_{AC} \xleftarrow{R} \mathbb{F}_p$ and sends r_{AB} to Bob, r_{AC} to Charlie (Alice's share; Charlie's share)

↳ **Observation:** $(x_A - r_{AB} - r_{AC}, r_{AB}, r_{AC})$ is additive secret sharing of Alice's input x_A (r_{AB}: Bob's share)

We will write $[x_A]$ to denote additive secret sharing of x_A

Computing on shares: Given shares of x_A and x_B,

$$[x_A + x_B] = [x_A] + [x_B] \quad \text{(component-wise addition)}$$

Specifically if $[x_A] = (x_{A,1}, x_{A,2}, x_{A,3})$ where $x_{A,1} + x_{A,2} + x_{A,3} = x_A \in \mathbb{F}_p$

$[x_B] = (x_{B,1}, x_{B,2}, x_{B,3})$ where $x_{B,1} + x_{B,2} + x_{B,3} = x_B \in \mathbb{F}_p$

then $[x_A + x_B] = (x_{A,1} + x_{B,1}, x_{A,2} + x_{B,2}, x_{A,3} + x_{B,3})$ and $(x_{A,1} + x_{B,1}) + (x_{A,2} + x_{B,2}) + (x_{A,3} + x_{B,3}) = x_A + x_B \in \mathbb{F}_p$

More generally:

1. Share addition: $[x_A + x_B] = [x_A] + [x_B]$
2. Scalar multiplication: $[k x_A] = k \cdot [x_A]$
3. Addition by constant: $[x_A + k] = (x_{A,1} + k, x_{A,2}, x_{A,3})$

Question: How to multiply shared values?

Amazing insight due to Beaver: Suppose parties have a secret-sharing of a random product: $[a], [b], [c]$ where $c = ab \in \mathbb{F}_p$ (each party only has a share of a, b, c: no one knows actual values!)

$a, b \xleftarrow{R} \mathbb{F}_p$ (a, b are uniformly random values)

Then, given $[x]$ and $[y]$, we proceed as follows:

1. Each party computes $[x-a]$ and publishes their share of $x-a$
2. Each party computes $[y-b]$ and publishes their share of $y-b$
3. All of the parties compute non-interactively:

$$[z] = [c] + [x](y-b) + [y](x-a) - (x-a)(y-b)$$

Claim: $z = xy$. Follows by following calculation:

$$z = c + x(y-b) + y(x-a) - (x-a)(y-b)$$
$$= \cancel{ab} + xy - bx + \cancel{xy} - \cancel{ay} - \cancel{xy} + bx + \cancel{ay} - \cancel{ab}$$
$$= xy$$

Observe: Parties only see $x-a$ and $y-b$ in this protocol. Since a, b are uniformly random and unknown to the parties, $x-a$ and $y-b$ is a one-time pad encryption of x and y. Resulting protocol provides information-theoretic privacy for parties' inputs.

Assuming we have access to Beaver multiplication triples, we can evaluate any arithmetic circuit as follows (among n-parties):

1. Every party secret shares their input with every other party
2. For each addition gate in the circuit, parties locally compute on their shares
3. For each multiplication gate in the circuit, parties run Beaver's multiplication protocol (using different triple each time!) (What goes wrong if the same triple is used twice?)
4. Every party publishes share of the output; parties run share reduction to obtain output.

Comparison with Yao:

	Secret-Sharing	Yao
Type of computation	Arithmetic circuits ($\mathbb{F}_p$)	Boolean circuits
Number of parties	Arbitrary (n)	2
Round complexity	Depth of circuit	2
Communication	$\sim 2n \log p$ bits per multiplication gate*	~ 256 bits per AND gate*
Security	Information-theoretic (with Beaver triples)	Computational

* Can be improved further!

* leverages several optimizations (half-gates + free XOR)

Aside: Preprocessing is also possible for OT (OT correlations): gives fast, information-theoretic OT in the online phase

Question: Where do Beaver triples come from? preprocessing is input-independent!

Typically, generated in an offline "preprocessing" phase. Many techniques:

- Trusted dealer (e.g., Intel SGX) can generate them
- Using oblivious transfer (OT) - accelerate using OT extensions (HW2, Problem 4)
- Using somewhat homomorphic encryption (discussed later in the course)

Implication: OT is "complete" for MPC: any n-party functionality can be computed securely assuming existence of OT.

Question: What if parties are malicious? [So far, everything is only semi-honest secure.]

[Goldreich-Micali-Wigderson]

The GMW compiler transforms any protocol with semi-honest security to one with malicious security

High-level idea: Parties include a zero-knowledge proof that each step of the protocol was performed according to the protocol specification.

Corollary: Anything that can be computed with a trusted party can be computed without!

[Ben-Or-Goldwasser-Wigderson]

- For n-parties, if we have fewer than $n/3$ corrupted parties, this is even possible information-theoretically!
- With cryptographic assumptions (i.e. OT), we can support $n-1$ corrupted parties (with some caveats...)