Cryptographic Protocols

Spring 2021

MPC Part 4 (shortened)

MPC Protocols — Communication Complexities

Notation: n parties, m multiplications, “fe” = field element.

Passive		**i.t., $t < n/2$**
Share		$\mathcal{O}(n)$ fe
Mult		$\mathcal{O}(n)$ Share $= \mathcal{O}(n^2)$ fe

Active	**crypto., $t < n/2$**	**i.t., $t < n/3$**
Broadcast	$\mathcal{O}(n^4)$ [not seen]	$\mathcal{O}(n^2)$ [seen: $\mathcal{O}(n^3)$]
Commit	1 Broadcast $= \mathcal{O}(n^4)$ fe	$\mathcal{O}(n^2)$ Broadcast $= \mathcal{O}(n^4)$ fe
Share	$\mathcal{O}(n)$ Commit $= \mathcal{O}(n^5)$ fe	
Mult	$\mathcal{O}(n)$ Share $= \mathcal{O}(n^6)$ fe	

Example: $n = 100$, 1fe $= 32$ bit, $m = 10^6$ multiplications.
Passive: ca. 40 Gigabyte / Active: ca. 4000 Petabyte of communication.

Today: Active, i.t., 400 Megabyte of communication. (10^{10} x faster!).

Passive Adversaries – Share and Reconstruct

Goal: Everything linear (i.e., $\mathcal{O}(n)$ fe per operation)

Share – Dealer P_i shares secret s with degree d

- $\forall P_j$: P_i sends share s_j on degree-d polynomial to P_j.
- Communication: $\mathcal{O}(n)$ fe. ☺

Assumption:

Reconstruction – Reconstruct sharing $[s]$ towards P_j

- $\forall P_i$: send s_i to P_j; P_j interpolates s (degree d).
- Communication: $\mathcal{O}(n)$ fe. ☺

Assumption:

Public Reconstruction – Reconstruct sharing $[s]$ to all

- 1) $\forall P_i$: send s_i to P_1; 2) P_1 interpolates s and sends it to $\forall P_j$.
- Communication: $\mathcal{O}(n)$ fe. ☺

Assumption:

Passive Adversaries – The Multiplication

Notation

- $[s]_d$ – a sharing of s with degree d.
- $[s]_{d,d'}$ – two (independent) sharings of (same) s with degrees d and d'.

Idea: Assume $[r]_{t,2t}$ for random r is given – *random double-sharing*.

Multiplication – Sharings $[a]_t$, $[b]_t$, $[r]_{t,2t} \rightarrow$ Product $[c]_t$

1. $\forall P_i$: compute $e_i = a_i \cdot b_i \rightarrow [e]_{2t}$.
2. Compute and reconstruct to all $[s]_{2t} = [e]_{2t} - [r]_{2t}$.
3. Compute $[c]_t = [r]_t + s$ (locally).

Assumption:

$\Rightarrow$ Communication: 1 Public Reconstruction $= \mathcal{O}(n)$ fe. ☺

New Problem: Generate random double-sharings, efficiently!
Step 1: Generate random sharings (communication: $\mathcal{O}(n)$ fe).
Step 2: Generate random double-sharings (same costs).

Passive Adversaries – Generate Random Sharings I

Approach 1 (sum of sharings)

- Protocol: Every P_i shares random s_i, let $[r] = [s_1] + \ldots + [s_n]$.
- Communication: $\mathcal{O}(n^2)$. ☹

Approach 2 (use less sharings)

- Protocol: Only $P_1, \ldots, P_{t+1}$ share random s_i, let $[r] = [s_1] + \ldots + [s_{t+1}]$.
- Communication: $\mathcal{O}(nt)$, for $t \approx n/2$, this is $\mathcal{O}(n^2)$. ☹

Approach 3 (extract multiple values)

- Protocol: Every P_i shares random s_i, let

$$[r_1] = 3[s_1] + 2[s_2] + \ldots + 4[s_n] \quad [r_2] = 1[s_1] + 8[s_2] + \ldots + 1[s_n] \quad [r_3] = 7[s_1] + \ldots$$

$$\begin{bmatrix} [r_1] \\ \vdots \\ [r_m] \end{bmatrix} = \begin{bmatrix} & M & \end{bmatrix} \begin{bmatrix} [s_1] \\ [s_2] \\ \vdots \\ [s_n] \end{bmatrix}$$

- Communication: $\mathcal{O}(n^2)$, but for m random sharings.
- Ideally: $m = n - t$. ☺ $\boxed{\text{Requirements on } M\text{?}}$

Hyper-Invertible Functions and Matrices

Def.: A function $f : \mathcal{F}^n \rightarrow \mathcal{F}^m, (x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_m)$ is **hyper-invertible** iff for any $k \leq n$ inputs x_i and any $n - k$ outputs y_j, all other inputs and outputs are uniquely defined.

Example: $f : \mathcal{F}^5 \rightarrow \mathcal{F}^3$ is hyper-invertible, then there exist

- $f_1 : (x_1, x_2, x_3, x_4, y_1) \mapsto (x_5, y_2, y_3)$,
- $f_2 : (x_1, x_3, x_5, y_2, y_3) \mapsto (x_2, x_4, y_1)$,
- $f_3 : (x_1, x_2, y_1, y_2, y_3) \mapsto (x_3, x_4, x_5)$,
- ...

Def.: A matrix M over $\mathcal{F}$ is **hyper-invertible** iff every non-trivial square sub-matrix of M is invertible.

$$\begin{bmatrix} \cdot & \times & \cdot & \cdot & \times & \times \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \times & \cdot & \cdot & \times & \times \\ \cdot & \times & \cdot & \cdot & \times & \times \end{bmatrix}$$

Lemma: Let M be an m-by-n matrix over $\mathcal{F}$ and f be the induced linear function $f : \mathcal{F}^n \rightarrow \mathcal{F}^m$. Then, f is hyper-invertible iff M is hyper-invertible.

Linear Hyper-Invertible Function – Construction I

7

$g : \mathcal{F}^n \to \mathcal{F}^m, (x_1, \dots, x_n) \mapsto (y_1, \dots, y_m)$, linear & hyper-invertible

Linear Hyper-Invertible Function – Construction II

8

Given: Points $(\alpha_1, x_1), \dots, (\alpha_n, x_n)$.
Wanted: $y_1, \dots, y_m$, s.t. $(\beta_1, y_1), \dots, (\beta_m, y_m)$ on same polynomial.

A little bit of Lagrange

- Reminder: $\lambda_j(x)$ is polynomial with $\lambda_j(\alpha_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$
- $g(x) = \lambda_1(x) \cdot x_1 + \lambda_2(x) \cdot x_2 + \dots + \lambda_n(x) \cdot x_n$.
- $y_1 = \lambda_1(\beta_1) \cdot x_1 + \lambda_2(\beta_1) \cdot x_2 + \dots + \lambda_n(\beta_1) \cdot x_n$ (a weighted sum!)
- $y_2 = \lambda_1(\beta_2) \cdot x_1 + \lambda_2(\beta_2) \cdot x_2 + \dots + \lambda_n(\beta_2) \cdot x_n$ (a weighted sum!)
- $\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & \cdots & m_{1n} \\ m_{21} & m_{22} & m_{23} & \cdots & m_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ m_{m1} & m_{m2} & m_{m3} & \cdots & m_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$, where $m_{ij} = \lambda_j(\beta_i)$

Passive Adversaries – Generate Random Sharings II

9

Generate Random Sharings

1. $\forall P_i$: chose random s_i, share s_i with degree $t \to [s_i]_t$.
2. All: $\begin{bmatrix} [r_1] \\ \vdots \\ [r_{n-t}] \end{bmatrix} = \begin{bmatrix} & \text{HIM} & \end{bmatrix} \begin{bmatrix} [s_1] \\ \vdots \\ [s_n] \end{bmatrix}$

   i.e., each P_i applies HIM on his respective shares.
3. Output $n - t$ sharings $[r_1], \dots, [r_{n-t}]$.

Analysis

1. Adversary corrupts parties B with $|B| = t$ and chooses $\{[s_i]\}_{i \in B}$.
2. Consider mapping $f : \{[s_i]\}_{i \notin B} \to \{[r_j]\}_j$.
3. f is bijective! Given $\{[s_i]\}_{i \in B}$ and $\{[r_j]\}_j$, can compute $\{[s_i]\}_{i \notin B}$.
4. Outputs $\{[r_j]\}_j$ is bijective function from good inputs $\{[s_i]\}_{i \notin B}$.
   Adversary can choose bijection ;–)

Passive Adversaries – Generate Random Double-Sharings

10

Generate Random Double-Sharings

1. $\forall P_i$: chose random s_i, share s_i with degree $t \to [s_i]_t$, and
   share s_i with degree $2t \to [s_i]_{2t}$
2. All: $\begin{bmatrix} [r_1]_{t,2t} \\ \vdots \\ [r_{n-t}]_{t,2t} \end{bmatrix} = \begin{bmatrix} & \text{HIM} & \end{bmatrix} \begin{bmatrix} [s_1]_{t,2t} \\ \vdots \\ [s_n]_{t,2t} \end{bmatrix}$

   i.e., each P_i applies HIM on his respective shares.
3. Output $n - t$ double-sharings $[r_1]_{t,2t}, \dots, [r_{n-t}]_{t,2t}$.

Analysis

- Bijection from “good” inputs $\{[s_i]_{t,2t}\}_{i \notin B}$ onto outputs $\{[r_j]_{t,2t}\}_j$.
- Double-sharing property “survives” HIM.

Communication: $\mathcal{O}(n^2)$ fe for $n-t$ double sharings,
amortized $\mathcal{O}(n)$ fe per double sharing. ☺

Passive Adversaries – Summary

11

Model: $t < n/2$, passive, perfect security.

Preparation: Generate enough random double-sharings $[r]_{t,2t}, \dots$

MPC Protocol

- Input: P_i wants to input s
  1. P_i: secret-share $s \to [s]$
- Addition / Linear gates: $[c] = f([a], [b], \dots)$
  1. $\forall P_i$: $c_i = f(a_i, b_i, \dots)$.
- Multiplication: $[c] = [a] \cdot [b]$
  1. pick random double-sharing $[r]_{t,2t}$.
  2. $\forall P_i$: compute $e_i = a_i \cdot b_i \to [e]_{2t}$.
  3. Compute and publicly-reconstruct $[s]_{2t} = [e]_{2t} - [r]_{2t}$.
  4. $[c]_t = [r]_t + s$.
- Output: P_i shall receive output $[s]$
  1. Reconstruct $[s]$ towards P_i.

Communication: $\mathcal{O}(n)$ fe per input/multiplication/output. ☺