Cryptographic Protocols

Spring 2018

Part 8

Seminal MPC Protocols — Analysis

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

- $\forall P_j$: 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 $[a]_t$, $[b]_t$, $[r]_{t,2t}$ → 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:

⇒ 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

1. $\forall P_i$, chose random s_i, share it $\rightarrow [s_i]_t$.
2. $[r] = \Sigma_i\, [s_i]$.

Analysis

- Communication: $\mathcal{O}(n^2)$ ☹
- Very wasteful: $n-t$ good sharings $[s_i] \rightarrow$ only 1 output sharing $[r]$.

Free Wishes

1. Fairy tells us, which $n-t$ of the $[s_i]$'s are the good ones.
2. Fairy gives us $[r_1], \ldots, [r_{n-t}]$, as good as the good $[s_i]$'s.

Hyper-Invertible Matrices (HIM) – The Fairy Tale

$$\begin{bmatrix} [r_1] \\ \vdots \\ [r_m] \end{bmatrix} = \begin{bmatrix} \quad \\ \text{HIM} \\ \quad \end{bmatrix} \begin{bmatrix} [s_1] \\ \vdots \\ [s_n] \end{bmatrix}$$

Any k of the $[r_i]$'s are as good as *the best* k of the $[s_i]$'s

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

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

Passive Adversaries – Generate Random Sharings II

Generate Random Sharings

1. $\forall P_i$: chose random s_i, share s_i with degree $t \to [s_i]_t$.
2. All:
$$\begin{bmatrix} [r_1] \\ \vdots \\ [r_{n-t}] \end{bmatrix} = \begin{bmatrix} \quad \text{HIM} \quad \end{bmatrix} \begin{bmatrix} [s_1] \\ \vdots \\ [s_n] \end{bmatrix}$$
i.e., each P_i applies HIM on his respective shares.
3. Output $n - t$ sharings $[r_1], \ldots, [r_{n-t}]$.

Analysis

1. Assume sharings $\{[s_i]\}_{i \in G}$ are good, for $|G| = n - t$.
2. For fixed values $\{[s_i]\}_{i \notin G}$, consider $f : \{[s_i]\}_{i \in G} \to \{[r_j]\}_j$.
3. f is bijective! Given $\{[s_i]\}_{i \notin G}$ and $\{[r_j]\}_j$, can compute $\{[s_i]\}_{i \in G}$.
4. Output $\{[r_j]\}_j$ is bijective function from good inputs $\{[s_i]\}_{i \in G}$.
Adversary can choose bijection ;–)

Passive Adversaries – Generate Random Double-Sharings

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:
$$\begin{bmatrix} [r_1]_{t,2t} \\ \vdots \\ [r_{n-t}]_{t,2t} \end{bmatrix} = \begin{bmatrix} \quad \text{HIM} \quad \end{bmatrix} \begin{bmatrix} [s_1]_{t,2t} \\ \vdots \\ [s_n]_{t,2t} \end{bmatrix}$$
i.e., each P_i applies HIM on his respective shares.
3. Output $n - t$ sharings $[r_1]_{t,2t}, \ldots, [r_{n-t}]_{t,2t}$.

Analysis

- Bijection from "good" inputs $\{[s_i]_{t,2t}\}_{i \in G}$ 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

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

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

MPC Protocol

- Input: P_i wants to input s
  1. P_i: secret-share $s \to [s]$
- Addition / Linear gates: $[c] = \mathcal{L}([a], [b], \ldots)$
  1. $\forall P_i$: $c_i = \mathcal{L}(a_i, b_i, \ldots)$.
- 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 reconstruct $[s]_{2t} = [e]_{2t} - [r]_{2t}$ to all.
  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. ☺

Active Adversaries

Overview

- Share – secure for honest dealers
- Reconstruction – error correction codes
- Public reconstruction — new trick
- Generating random double-sharings — hyper-invertible matrices 2.0

Today: Only security with abort!

. . . can be pimped to full security with standard techniques (2x costs)

Active Adversaries – Share and Reconstruct

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

Share – Dealer P_i shares secret s

- $\forall P_j$: 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. *Abort if shares are inconsistent.*
- 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:~~

Active 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 $[a]_t$, $[b]_t$, $[r]_{t,2t}$ → Product $[c]_t$

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

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

Assumption:

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).

Active Adversaries – Generate Random Double-Sharings

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:
$$\begin{bmatrix}[r_1]_{t,2t}\\ \vdots \\ [r_n]_{t,2t}\end{bmatrix} = \begin{bmatrix} \quad \text{HIM} \quad \end{bmatrix} \begin{bmatrix}[s_1]_{t,2t}\\ \vdots \\ [s_n]_{t,2t}\end{bmatrix}$$
3. For $j = 1, \ldots, 2t$:
   i) Reconstruct $[r_j]_{t,2t}$ towards P_j.
   ii) P_j: check whether $[r_j]_{t,2t}$ is good (degrees t and $2t$, same secrets).
   iii) P_j: In case of fault: broadcast complaint, **ABORT**
4. Output $n - 2t$ sharings $[r_{2t+1}]_{t,2t}, \ldots, [r_n]_{t,2t}$.

Analysis: $n-t$ good $[s_i]_{t,2t}$, t checked $[r_j]_{t,2t}$, others are lin. combinations.

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

Active Adversaries – Public Reconstruction

Idea: Reconstruct k sharings $[a_1]_d, \ldots, [a_k]_d$ at once, $k \leq n - t$.

High-Level Protocol

1. Expand $[a_1]_d, \ldots, [a_k]_d$ to $[b_1]_d, \ldots, [b_n]_d$, with redundancy.
2. Reconstruct each $[b_i]_d$ to P_i.
3. $\forall P_i$: send b_i to $\forall P_j$.
4. $\forall P_j$: shrink $b_1, \ldots, b_n$ to $a_1, \ldots, a_k$, abort if inconsistent.

Assumption:

Expansion

- Interpret $a_1, \ldots, a_k$ as coefficients of polynomial $g(\cdot)$ of degree $< k$.
- $b_i = g(\alpha_i) = \sum_j \lambda_{ij} a_j$, $[b_i] = \sum_j \lambda_{ij} [a_j]$.
- Shrinking: Find $g(\cdot)$, given $b_1, \ldots, b_n$.

Analysis: Detects cheating of up to $n - k \geq t$ parties.

Communication: $\mathcal{O}(n^2)$ fe for $n - t$ reconstructions. ☺

Active Adversaries – Summary

Model: $t < n/3$, active, perfect security.

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

MPC Protocol

- Input: P_i wants to input s
  1. pick next prepared double-sharing $[r]_{t,2t}$.
  2. reconstruct $[r]_t$ towards P_i.
  3. P_i: broadcast $e = s - r$.
  4. Parties take $[s]_t = [r]_t + e$ as sharing of input.
- Addition / Linear gates: same as passive
- Multiplication: same as passive (with actively-secure public recons.)
- Output: Use reconstruction protocol.

Communication: $\mathcal{O}(n)$ fe per multiplication/output, ☺
$\mathcal{O}(n^2)$ fe per input.