Cryptographic Protocols

Spring 2017

Part 9

MPC as a Service

Zürcher Kantonalbank

DIE POST

RAIFFEISEN

SwissLife

Adversary Structure

Notation

- Party set $\mathcal{P}$, $|\mathcal{P}| = n$
- Monotone adversary structure $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_\ell\} \subseteq 2^{\mathcal{P}}$
  (Monotone: $Z \in \mathcal{Z},\ Z' \subseteq Z \Rightarrow Z' \in \mathcal{Z}$)

Adv. chooses one of them

Definitions

- $Q^2(\mathcal{P}, \mathcal{Z}) :\Leftrightarrow \forall Z_1, Z_2 \in \mathcal{Z} : Z_1 \cup Z_2 \neq \mathcal{P}$ (no two sets add up to $\mathcal{P}$)
- $Q^3(\mathcal{P}, \mathcal{Z}) :\Leftrightarrow \forall Z_1, Z_2, Z_3 \in \mathcal{Z} : Z_1 \cup Z_2 \cup Z_3 \neq \mathcal{P}$
  (no three sets add up to $\mathcal{P}$)

Results

	Threshold	Gen.Adv.
• i.t. passive:	$t < n/2$	$Q^2(\mathcal{P}, \mathcal{Z})$ [HM97, Mau02]
• i.t. active:	$t < n/3$	$Q^3(\mathcal{P}, \mathcal{Z})$ [HM97, Mau02]
• crypto. active:	$t < n/2$	$Q^2(\mathcal{P}, \mathcal{Z})$ [HM97, Mau02]

Sharing

Addition / Linear Functions

Sharing State

$\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_\ell\}$

Sharing State:

A value s is shared if:

- s is split into random summands $s_1, \ldots, s_\ell$, where $\sum s_q = s$.
- All parties in $\overline{Z}_q$ know s_q.

A shared value is denoted by $[s]$.

Linearity

- $[a + b] = [a] + [b]$ locally add up summands
- $c \cdot [a] = [c \cdot a]$

Passive Protocol

Share Input

- P_d has input a.
- P_d selects random summands $a_1, ..., a_\ell$ s.t. $\sum a_q = a$.
- P_d sends a_q to every P_i in $\overline{Z}_q$

Reconstruct Output

- a is shared by $a_1, ..., a_\ell$.
- For all q: a fixed $P_i \in \overline{Z}_q$ sends a_q to P_r.
- P_r computes $a = \sum a_q$.

Addition

- a, b shared by $a_1, ..., a_\ell,\ b_1, ..., b_\ell$.
- For all q: Every $P_i \in \overline{Z}_q$ computes $c_q = a_q + b_q$.

Multiplication

- a, b shared by $a_1, ..., a_\ell,\ b_1, ..., b_\ell$.
- For all p, q: A fixed $P_i \in \overline{Z}_p \cap \overline{Z}_q$ computes and shares $a_p b_q$.
- Compute $[c] = \sum\sum[a_p b_q]$.

Multiplication: Idea

Goal: Given $[a]$,$[b]$ compute $[c] = [ab]$.

Observation:

$$\begin{aligned} ab &= (a_1 + \ldots + a_\ell)(b_1 + \ldots + b_\ell) \\ &= a_1 b_1 + a_1 b_2 + \ldots + a_\ell b_\ell = \sum_{p=1}^{\ell} \sum_{q=1}^{\ell} a_p b_q \end{aligned}$$

Idea: Its is enough to compute sharings of all $a_p b_q$:

$$[ab] = \sum_{p=1}^{\ell} \sum_{q=1}^{\ell} [a_p b_q]$$

Observation:

$$Q^2(\mathcal{P}, \mathcal{Z}) \quad \Leftrightarrow \quad \forall p, q\ Z_p \cup Z_q \neq \mathcal{P} \quad \Leftrightarrow \quad \forall p, q\ \exists P_i \in \overline{Z}_p \cap \overline{Z}_q$$

Active Sharing Protocol

Goal: Share input value s of party P_d.

Share(P_d, s)

1. P_D selects random summands $s_1 + \ldots + s_\ell$ s.t. $s = \sum s_q$
2. For all $1 \leq q \leq \ell$ do:
   a) P_D sends s_q to all parties in $\overline{Z_q}$.
   b) The parties in $\overline{Z_q}$ exchange the received values.
   If $P_i \in \overline{Z_q}$ sees different values: complain using broadcast.
   c) If any $P_i \in \overline{Z_q}$ complained: P_D broadcasts s_q.
   Otherwise each $P_i \in \overline{Z_q}$ takes the value received in step 2a) for s_q.

Active Reconstruction

Goal: Reconstruct $[s]$ towards P_r

Share-Reconstruction($P_r, [s], q$)

1. Every party $P_i \in \overline{Z}_q$ sends s_q to P_r.
2. Let v_i be the value P_r received from $P_i \in \overline{Z}_q$.
   P_r outputs v such that $\{P_i \mid v_i \neq v\} \in \mathcal{Z}$.

Reconstruction($P_r, [s]$)

1. For all q invoke $s_q \leftarrow$ Share-Reconstruction$(P_r, [s], q)$.
2. P_r outputs $s = s_1 + \ldots + s_\ell$.

Active Multiplication Protocol

Goal: Compute $[c] = [ab]$

Multiplication($[a], [b]$)

1. For all $1 \leq q, p \leq \ell$ do:
   a) Every party $P_i \in \overline{Z}_p \cap \overline{Z}_q$ shares $a_p b_q$ as $[v_{pq}^k]$.
   b) Let P_i be a fixed party in $\overline{Z}_p \cap \overline{Z}_q$.
   Compute and open $[v_{pq}^i] - [v_{pq}^j]$ for all $P_j \in \overline{Z}_p \cap \overline{Z}_q$.
   c) If all differences are zero: Set $[v_{pq}] = [v_{pq}^1]$ as a sharing for $a_p b_q$.
   Otherwise, reconstruct a_p and b_q (using Share-Reconstruction).
   Define $[v_{pq}]$ as the sharing with summands $(a_p b_q, 0, \ldots, 0)$.
2. Compute $[c] = \sum[v_{pq}]$.