Cryptographic Protocols

Spring 2021

MPC Part 5

Active Adversaries – High-Level Approach

Setting

- Information-theoretic security, active adversary, $t < n/3$.

Approach

- Values are Shamir-shared with degree t → no commitments!
- Reconstruction deals with faulty shares → error-correction codes
- Generating random double-sharings → hyper-invertible matrices 2.0
- Public reconstruction → new trick

Structure

1. Detectable MPC → security with abort (abort only in case of cheating)
2. Preprocessing phase → circuit randomization
3. Full security → player-elimination framework

Active Adversaries – Local Reconstruction

Goal: Reconstruct sharing $[s]_d$ towards P_i. $(d = t$ or $d = 2t)$

Protocol

1. $\forall P_j$: send s_j to P_i.
2. P_i: If $\exists\, g$ with $\deg(g) \le d$ and $\left|\{j : s_j = g(\alpha_j)\}\right| \ge d+1+t$ then
   output $s = g(0)$
   else
   ABORT

Correctness: $d+1+t$ shares on $g \Rightarrow d+1$ "honest" shares $\Rightarrow$ correct g.

Robustness: Robust if at least $d+1+t$ honest parties, i.e., if $d < n-2t$.

Efficiency: Berlekamp-Welch decoder $\Rightarrow$ find g efficiently.

Active Adversaries – Public Reconstruction

Goal: Publicly reconstruct $k+1$ sharings $[s_0]_d, \ldots, [s_k]_d$.

High-Level Protocol

1. Expand $[s_0]_d, \ldots, [s_k]_d$ to $[u_1]_d, \ldots, [u_n]_d$, with redundancy.
2. $\forall P_i$: locally reconstruct $[u_i]_d$ to P_i, send u_i to $\forall P_j$ (might **ABORT**).
3. $\forall P_j$: shrink $u_1, \ldots, u_n$ to $s_0, \ldots, s_k$ (might **ABORT**).

Expansion

- Interpret $s_0, \ldots, s_k$ as coefficients of polynomial g of degree k.
- $u_i = g(\alpha_i) = s_0 + s_1\alpha_i + \ldots + s_k\alpha_i^k$, $\quad [u_i]_d = [s_0]_d + \ldots + [s_k]_d\alpha_i^k$.
- Shrinking: Find coefficients of g s.t. $|\{i : u_i = g(\alpha_i)\}| \ge k+1+t$.

Correctness: $k+1+t$ values u_i on $g \Rightarrow$ correct g.

Robustness: Robust if $d < n-2t$ **and** $k < n-2t$.

Communication: $\mathcal{O}(n^2)$ fe for $k+1$ public reconstructions. ☺

Active Adversaries – Generate Random Sharings

Generate Random Sharings

1. $\forall P_i$: chose random s_i, share s_i with degree $t \to [s_i]$.
2. All: $\begin{bmatrix}[r_1]\\ \vdots \\ [r_n]\end{bmatrix} = \begin{bmatrix} & \text{HIM} & \end{bmatrix}\begin{bmatrix}[s_1]\\ \vdots \\ [s_n]\end{bmatrix}$
3. For $j = 1, \ldots, 2t$:
   i) Reconstruct $[r_j]$ towards P_j.
   ii) P_j: check that *all* shares of $[r_j]$ lie on polynomial of degree t. Otherwise: **ABORT**
4. Output $n-2t$ sharings $[r_{2t+1}], \ldots, [r_n]$.

Correctness: $n-t$ good $[s_i]$, t checked $[r_j]$, others are linear combinations.

Secrecy: Adv. knows t $[s_i]$ plus t $[r_j]$, any $n-2t$ sharings are random.

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

Active Adversaries – Generate Random Double-Sharings

Generate Random Double-Sharings

1. $\forall P_i$: chose random s_i, share s_i with degrees t and $2t \to [s_i]_{t,2t}$.
2. All: $\begin{bmatrix}[r_1]_{t,2t}\\ \vdots \\ [r_n]_{t,2t}\end{bmatrix} = \begin{bmatrix} & \text{HIM} & \end{bmatrix}\begin{bmatrix}[s_1]_{t,2t}\\ \vdots \\ [s_n]_{t,2t}\end{bmatrix}$
3. For $j = 1, \ldots, 2t$:
   i) Reconstruct $[r_j]_{t,2t}$ towards P_j.
   ii) P_j: check that *all* shares of $[r_j]_t$ lie on degree-t polynomial g,
   AND that *all* shares of $[r_j]_{2t}$ lie on degree-$2t$ polynomial g',
   AND that $g(0) = g'(0)$.
   Otherwise: **ABORT**
4. Output $n-2t$ double-sharings $[r_{2t+1}]_{t,2t}, \ldots, [r_n]_{t,2t}$.

Observe: Linear combination of (correct) random double-sharings are (correct) random double-sharings!
→ same analysis as for "normal" sharings.

Active Adversaries / Security with Abort – Summary

Model: $t < n/3$, active adversary, security with abort.

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, ☺
- 1 broadcast per input.

Circuit Randomization

Preparation

- Generate enough triples $([a],[b],[c])$ with a, b random and $c = ab$.

Observation

$$\begin{aligned} x \cdot y &= ((x-a)+a)\cdot((y-b)+b) \\ &= (x-a)(y-b) + (x-a)b + (y-b)a + ab \end{aligned}$$

Multiplication protocol: $[x] \cdot [y]$

1. Compute and publicly reconstruct $[u] = [x] - [a]$
   and $[v] = [y] - [b]$.
2. Compute $[x \cdot y] = uv + u[b] + v[a] + [c]$.

Communication: 2 public reconstructions per multiplication. ☺

Robustness: The protocol is robust! ☺☺

Player Elimination Framework

Structure

1. Non-Robust Computation: Run protocol, parties can abort.
2. Fault Detection: $\forall P_i$ broadcasts 1 if aborted, take OR.
3. Fault Localization
   - 3.1. Choose referee P_r (any party, e.g. P_1).
   - 3.2. $\forall P_i$: send all random values and all received messages to P_r.
   - 3.3. P_r: identify P_i, P_j disagreeing on m_k, broadcast $(i, j, k, m_k^{(i)}, m_k^{(j)})$.
   - 3.4. P_i, P_j: broadcast "agree" or "accuse".
   - 3.5. If P_i/P_j accuses, then $E = \{P_i, P_r\}/\{P_j, P_r\}$. Else $E = \{P_i, P_j\}$.
4. Player elimination: Eliminate E, repeat.

Obstacles

- Additional costs ⇒ divide computation into t blocks.
- Secrecy ⇒ use player-elimination only in preparation.
- Shrinking player set ⇒ all sharings of fixed degree t.

Prepare Multiplication Triples I

Prepare m Multiplication Triples

1. Initialize $\mathcal{P}' \leftarrow \{P_1, \ldots, P_n\}$, $t' \leftarrow t$, triples $\mathcal{T} \leftarrow \emptyset$.
2. Repeat until $|\mathcal{T}| \geq m$:
   - 2.1 Non-robustly generate block $\mathcal{B}$ of $\ell = m/t$ triples with degree t.
   - 2.2 On abort: $\mathcal{P}' \leftarrow \mathcal{P}' \setminus E$, $t' \leftarrow t' - 1$, discard block.
   - 2.3 On success: $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{B}$.

Communication: At most t aborts, i.e., at most $2m$ triples are generated.

Invariant: All sharings with degree t (among parties $\mathcal{P}'$).

New Problem

- Generate multiplication triples with degree t.
- Party set is $\mathcal{P}'$ with $|\mathcal{P}'| = n'$, t' corrupted, where ....

Prepare Multiplication Triples II

Non-Robustly Generate Block of ℓ Multiplication Triples

1. Generate ℓ random double-sharings $[a]_{t',t}$.
2. Generate ℓ random double-sharings $[b]_{t',t}$.
3. Generate ℓ random double-sharings $[r]_{t',2t'}$.
4. Compute and publicly reconstruct $[s]_{2t'} = [a]_{t'} \cdot [b]_{t'} - [r]_{2t'}$.
5. Locally compute $[c]_t = [r_t] + s$
6. Output triple $([a]_t, [b]_t, [c]_t)$.

Communication: $\mathcal{O}(n)$ per triple.

Active Adversaries / Full Security – Summary

Preparation

1. Initialize $\mathcal{P}' \leftarrow \{P_1, \ldots, P_n\}$, $t' \leftarrow t$, triples $\mathcal{T} \leftarrow \emptyset$.
2. Generate triples with degree t, in blocks of size $\ell = m/t$.
3. Player-Elimination, until t successful blocks.
4. Output triples $\mathcal{T}$, new party set $\mathcal{P}'$, new threshold t'.

MPC Protocol

- Input: Pick next triple, reconstruct $[a]_t$ to P_i, broadcast difference.
- Addition / Linear gates: same as passive.
- Multiplication: Pick next triple, reconstruct $[x]_t - [a]_t$ and $[y]_t - [b]_t$.
- Output: Use reconstruction protocol.

Communication

- $\mathcal{O}(n)$ fe per multiplication/output, ☺
- 1 broadcast per input.