Cryptographic Protocols

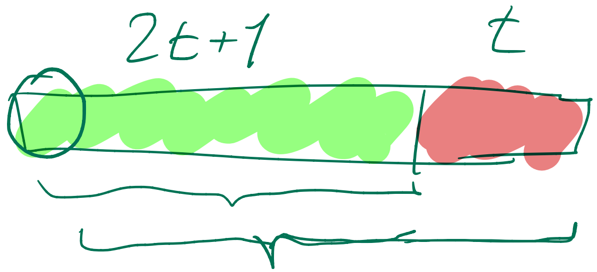
Spring 2021

MPC Part 5 /2

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! ☺☺

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 $(i, j, k, m_k^{(i)}, m_k^{(j)})$.
      $\neq$
    - 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.

$n > n' > n''$
$t > t' > t''$

Obstacles

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

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. ~~$\mathcal{B}$~~
   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 $t + 2t' < n'$

$3t < n$

$n-2, t-1$

$[a]_t\ [b]_t\ [c]_t$ from local recons: $t+1+2t' < n'$

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 $([a]_t, [b]_t, [c]_t)$.

t, not t'!

among P', with |P'| = n'

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

parties $\mathcal{P}'$, with t' corruptions, all degree t, req. $t+2t'<n$

- 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.