Cryptographic Protocols

Spring 2021

MPC Part 2

MPC with an Active Adversary

Model

- Active adversary, corrupted parties can deviate from the protocol.
- For now: unbounded computation power, $t < n/2$.
- Synchronous, secure channels.
- Broadcast.

Corruption Levels

0. Passive corruption (adversary can read internal state).
1. Level 0 + corrupted parties send additional messages.
2. Level 1 + corrupted parties withhold messages.
3. Level 2 + corrupted parties send wrong messages (= active adversary).

Passive Protocol / Adv. can send additional Messages

Share input

0. P_i has input s.
1. P_i: select $r_1, ..., r_t$ at random.
2. P_i: comp. $\begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} = A \begin{pmatrix} s \\ r_1 \\ \vdots \\ r_t \end{pmatrix}$.
3. P_i: send s_j to every P_j.

Reconstruct Output

0. a is shared by $a_1, ..., a_n$.
1. $\forall P_j$: send a_j to P_i.
2. P_i: comp. $a = \mathcal{L}(a_1, ..., a_n)$.

Addition and Linear Functions f

0. $a, b, \ldots$ are shared by $a_1, ..., a_n,\ b_1, ..., b_n$, etc.
1. $\forall P_i$: compute $c_i = f(a_i, b_i, \ldots)$.

Multiplication

0. a, b are shared by $a_1, ..., a_n, b_1, ..., b_n$.
1. $\forall P_i$: compute $d_i = a_i b_i$.
2. $\forall P_i$: share $d_i \rightarrow d_{i1}, \ldots, d_{in}$.
3. $\forall P_j$: compute $c_j = \mathcal{L}(d_{1j}, \ldots, d_{nj})$.

Additional Messages:

Passive Protocol / Adv. can withhold Messages

Share input

0. P_i has input s.
1. P_i: select $r_1, ..., r_t$ at random.
2. P_i: comp. $\begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} = A \begin{pmatrix} s \\ r_1 \\ \vdots \\ r_t \end{pmatrix}$.
3. P_i: send s_j to every P_j.

Reconstruct Output

0. a is shared by $a_1, ..., a_n$.
1. $\forall P_j$: send a_j to P_i.
2. P_i: comp. $a = \mathcal{L}(a_1, ..., a_n)$.

Addition and Linear Functions f

0. $a, b, \ldots$ are shared by $a_1, ..., a_n,\ b_1, ..., b_n$, etc.
1. $\forall P_i$: compute $c_i = f(a_i, b_i, \ldots)$.

Multiplication

0. a, b are shared by $a_1, ..., a_n, b_1, ..., b_n$.
1. $\forall P_i$: compute $d_i = a_i b_i$.
2. $\forall P_i$: share $d_i \rightarrow d_{i1}, \ldots, d_{in}$.
3. $\forall P_j$: compute $c_j = \mathcal{L}(d_{1j}, \ldots, d_{nj})$.

Withholding Messages:

Case 1:

Case 2:

Case 3:

Passive Protocol / Adv. can send wrong Messages

Share input

0. P_i has input s.
1. P_i: select $r_1, ..., r_t$ at random.
2. P_i: comp. $\begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} = A \begin{pmatrix} s \\ r_1 \\ \vdots \\ r_t \end{pmatrix}$.
3. P_i: send s_j to every P_j.

Reconstruct Output

0. a is shared by $a_1, ..., a_n$.
1. $\forall P_j$: send a_j to P_i.
2. P_i: comp. $a = \mathcal{L}(a_1, ..., a_n)$.

Addition and Linear Functions f

0. $a, b, \ldots$ are shared by $a_1, ..., a_n,\ b_1, ..., b_n$, etc.
1. $\forall P_i$: compute $c_i = f(a_i, b_i, \ldots)$.

Multiplication

0. a, b are shared by $a_1, ..., a_n, b_1, ..., b_n$.
1. $\forall P_i$: compute $d_i = a_i b_i$.
2. $\forall P_i$: share $d_i \rightarrow d_{i1}, \ldots, d_{in}$.
3. $\forall P_j$: compute $c_j = \mathcal{L}(d_{1j}, \ldots, d_{nj})$.

Idea:

MPC with an Active Adversary – Commitment Scheme

Commitment Scheme

- Protocol COMMIT: P_i can commit to a value a.
- Protocol OPEN: P_i can open a (to all parties).
- Binding (P_i cannot open wrong value $a' \neq a$).
- Hiding (adversary learns nothing about committed value a).
- Homomorphic: P_i is committed to a and $b \Rightarrow P_i$ is committed to $a + b$.
- Protocol CTP (Commitment Transfer Protocol): P_i is committed to a, can transfer commitment to P_j.
- Protocol CMP (Commitment Multiplication Proof): P_i is committed to a, b, and c, can prove that $c = a \cdot b$.

Macro: Protocol CSP (Commitment Sharing Protocol): P_i is committed to $a \Rightarrow a$ is shared, parties are committed to shares.

MPC with an Active Adversary – Generic Protocol for $t < n/2$

Share input

0. P_i has input s.
1. P_i: select $r_1, ..., r_t$ at random.
2. P_i: comp. $\begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} = A \begin{pmatrix} s \\ r_1 \\ \vdots \\ r_t \end{pmatrix}$.
3. P_i: send s_j to every P_j.

Reconstruct Output

0. a is shared by $a_1, ..., a_n$.
1. $\forall P_j$: send a_j to P_i.
2. P_i: comp. $a = \mathcal{L}(a_1, ..., a_n)$.

Addition and Linear Functions f

0. $a, b, \ldots$ are shared by $a_1, ..., a_n$, $b_1, ..., b_n$, etc.
1. $\forall P_i$: compute $c_i = f(a_i, b_i, \ldots)$.

Multiplication

0. a, b are shared by $a_1, ..., a_n, b_1, ..., b_n$.
1. $\forall P_i$: compute $d_i = a_i b_i$.
2. $\forall P_i$: share $d_i \rightarrow d_{i1}, \ldots, d_{in}$.
3. $\forall P_j$: compute $c_j = \mathcal{L}(d_{1j}, \ldots, d_{nj})$.

Prerequisite:

MPC Active – Cryptographic Security

Model

- Active adversary
- Adversary is computationally bounded
- $t < n/2$

Goal

- Homomorphic Commitment Scheme with CTP and CMP

Type B vs. Type H

- Type B: perfect correctness, computational secrecy
- Type H: perfect secrecy, computational correctness?
- Everlasting Security

Recap: Commitment Schemes – Definition

Intuition

- Peggy P commits to a value x towards Vic V.
- Peggy can open x if she wants to.

Attempt 1: Hash function h, send $h(x)$ to COMMIT, send x to OPEN.

Definition: A commitment scheme is a pair of protocols (COMMIT, OPEN), where Peggy inputs x in COMMIT and Vic outputs x' in OPEN, s.t.

- Binding: After COMMIT, the value x is fixed.
- Hiding: In COMMIT, Vic does not learn x.
- Correctness: If Vic is honest, then $x' \in \{x, \perp\}$.
  If both are honest, then $x' = x$.

Attempt 2: Random r, send $h(r\|x)$ to COMMIT, send (x, r) to OPEN.

Recap: Commitment Schemes – Types

Non-interactive Commitment Scheme

- Function $C : (x, r) \rightarrow b$.
- COMMIT: Peggy computes and sends $b = C(x, r)$ (the blob).
- OPEN: Peggy sends (x, r), Vic checks that $b \stackrel{?}{=} C(x, r)$.

Type B

- Perfect Binding (even unbounded Peggy cannot open $x' \neq x$).
- (At least) computational Hiding.

Type H

- Perfect Hiding (even unbounded Vic obtains no information about x).
- (At least) computational Binding.

Lemma: Simultaneously Type B and Type H is not possible.

Recap: Pedersen Commitment Scheme

Setting

- Cyclic group H of prime order $q = |H|$.
- Generators g and h, i.e., $H = \langle g \rangle = \langle h \rangle$, $\mathrm{DL}_g(h)$ unknown.

Commitment

- Value $x \in \mathbb{Z}_q$, random value $r \in_R \mathbb{Z}_q$.
- $C(x, r) = g^x h^r$.

Analysis

- Perfect hiding: $r \in_R \mathbb{Z}_q \Rightarrow h^r \in_R H \Rightarrow g^x h^r \in_R H$.
- Comp. binding: given $g^x h^r = g^{x'} h^{r'} \rightarrow$ can compute $\mathrm{DL}_g(h)$.

Trapdoor Commitment Scheme

- If Vic knows Trapdoor $T = \mathrm{DL}_g(h)$, he can open both ways.
- Relevant in some zero-knowledge proofs.

ElGamal Commitment Scheme

Setting

- Cyclic group H of prime order $q = |H|$.
- Generators g and h, i.e., $H = \langle g \rangle = \langle h \rangle$, $\mathrm{DL}_g(h)$ unknown.

Commitment

- Value $x \in \mathbb{Z}_q$, random value $r \in_R \mathbb{Z}_q$.
- $C(x, r) = (g^r, g^x h^r)$.

Analysis

- $\rightarrow$ exercise

Homomorphic Commitment Schemes

Informally
- Homomorphic ⇒ can "add" blobs, results in blob for the sum.

Definition
- A commitment scheme is homomorphic if
  $C(x,r) \otimes C(x',r') = C(x \oplus x', r \oplus r')$.

Examples
- Pedersen: $g^x h^r \cdot g^{x'} h^{r'} = g^{x+x'} h^{r+r'}$.
- ElGamal: → exercise

Multi-Party Commitment Schemes

Informally
- P_i commits to a value x towards all parties.
- P_i can open x if she wants.
- Either all (honest) parties accept x, or all (honest) parties reject.

Multi-Party Commitments from Non-Interactive Commitments
- Given non-interactive commitment scheme C.
- Commit: Compute $b = C(x,r)$, broadcast b.
- Open: Broadcast (x,r), every P_j accepts x iff $b \stackrel{?}{=} C(x,r)$.

CMP for Pedersen Commitments

Given: Commitments $A = g^a h^\alpha$, $B = g^b h^\beta$, $C = g^c h^\gamma$

Assume: Peggy knows $a, b, c, \alpha, \beta, \gamma$ such that $c = a \cdot b$

Goal: Prove that $c = a \cdot b$

Idea:
- B^a is *some* commitment to ab
- Prove knowledge of a such that
  i) a "is contained in" A ii) B^a and C "contain" same value

Sketch: Prove knowledge of a, α, ξ s.t. $A = g^a h^\alpha$ and $C = B^a \cdot g^0 h^\xi$

Proof
- Define $f_B : Z_q^3 \mapsto H^2, (a, \alpha, \xi) \to (g^a h^\alpha, B^a \cdot g^0 h^\xi)$
- Observe: f_B is a group homomorphism!
- Proof knowledge of a pre-image of (A, C) w.r.t. f_B
- Note: $(a, \alpha, \gamma - a\beta)$ is such a pre-image …

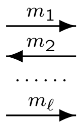
MPC Active – Cryptographic Security

Model: Active, crypto, $t < n/2$

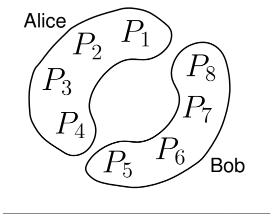
The Protocol
- Use generic protocol …
  … with a non-interactive, homomorphic Commitment Scheme $C(a, \alpha)$.
- Commit, Open via broadcast.
- CTP obvious.
- CMP: see previous slide (Pedersen) / exercise (ElGamal), …
  … challenge as a (linear) MPC, proof via broadcast.

MPC Active – Impossibility

Two Parties ($n = 2, t = 1$)

Alice		Bob
Input a		Input b
		
Output $c = ab$		Output $c = ab$

n Parties ($n, t \geq n/2$)



Analysis
- Consider *shortest* secure protocol. Hence, $m_1, \ldots, m_{\ell-1}$ is not secure!
- Corrupted Alice can drop last message.
- If ℓ is unknown (but poly-bounded): adversary can guess ℓ.