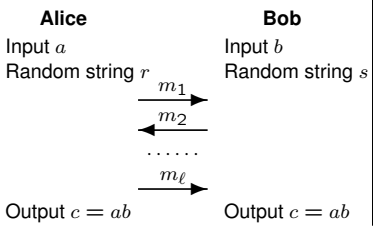
Cryptographic Protocols

Spring 2020

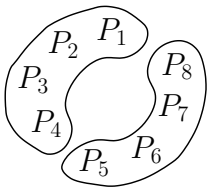
MPC Part 2

Passive MPC – Impossibility

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



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



Analysis

- $b = 0$: Transcript contains no info about a, i.e. $I(T; a) = 0$.
- $b = 1$: Transcript contains full info about a, i.e. $I(T, a) = H(a)$.
- Alice can check: Is there (a', r') s.t. same transcript is produced?
  (exhaustive search requires unbounded computing power)

MPC with Active Adversaries

Model

- Active adversary, corrupted parties can deviate from the protocol.
- So far: 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 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

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

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

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

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:

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.

Commitment Schemes – Types

Non-interactive Commitment Scheme

- Function $C : (x, r) \to b$.
- 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.

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 $g^x h^r = g^{x'} h^{r'} \to$ 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 (later ...).

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

- $\to$ exercise

Homomorphic Commitment Schemes

Informally

- Homomorphic $\Rightarrow$ 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: $\to$ exercise

Multi-Party Commitment Scheme

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 $b \stackrel{?}{=} C(x, r)$.

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

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

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 $d_i \to d_{i1}, \ldots, d_{in}$.
3. $\forall P_j$: compute $c_j = \mathcal{L}(d_{1j}, \ldots, d_{nj})$.

Commitment Scheme with: