Multi-Party Computation

Hoeteck Wee

May 14, 2018

Today  MPC with information-theoretic / unconditional security

• “everlasting” security, no computational assumptions, quantum-safe
• very efficient, simple field arithmetic, no exponentiations (cf. Unbound Tech, …)
• requires honest majority, or trusted set-up

\((n,t)\)-threshold secret sharing

A secret \(s\) is shared among \(n\) parties:

• (reconstruction) any subset of \(t+1\) parties (or more) can recover \(s\)
• (\(t\)-privacy) any subset of \(t\) parties has no information about \(s\)

Shamir: Suppose \(s \in \mathbb{F}_q\) and \(q \geq n + 1\).

• random polynomial \(p\) of degree \(t\) s.t. \(p(0) = s\), party \(i\) gets \(p(i)\).
• can compute \(\lambda_1, \ldots, \lambda_{t+1} \in \mathbb{F}_q\) s.t. \(p(0) = \sum_{i=1}^{t+1} \lambda_i p(i)\)
• \((p(1), \ldots, p(t))\) is identically distributed to the uniform distribution over \(\mathbb{F}_q^t\).

Preliminaries  Fix a prime \(q\). Are the following distributions identically distributed to the uniform distribution?

1. \((u_1, u_1 + u_2), \) where \(u_1, u_2 \leftarrow \mathbb{Z}_q\)
2. \((u_1 + u_2, u_2 + u_3, u_3 + u_1), \) where \(u_1, u_2, u_3 \leftarrow \mathbb{Z}_q\)
3. \((u_1 + u_2, u_2 + u_3, u_3 + u_4, u_4 + u_1), \) where \(u_1, u_2, u_3, u_4 \leftarrow \mathbb{Z}_q\)

Multi-Party Computation (MPC)  \(n\) parties \(P_1, \ldots, P_n\) want to jointly evaluate \(f(x_1, \ldots, x_n)\)

• input: \(x_i\) is the private input of \(P_i\)
• output: everyone learns \(f(x_1, \ldots, x_n)\)
• privacy: … and nothing else about the other inputs
MPC: security

- honest-but-curious: all parties follow the protocol honestly
  - do not change their inputs, do not abort pre-maturely
- t-privacy: if t parties collude, they learn nothing beyond their inputs and the output
  - \text{view}(P_i) = \text{input, randomness, messages received}
  - view of any t parties depend only on their inputs & output
  - exists “simulator” \text{sim} s.t. \forall S \subseteq [n], |S| \leq t and \forall x_1, \ldots, x_n

\text{view}(P_i : i \in S) and \text{sim}((x_i : i \in S), f(x_1, \ldots, x_n))

are identically distributed.

Examples for 1-privacy

1. \( n = 2, f(x_1, x_2) = x_1 + x_2 \) over \( \mathbb{Z}_q \)
   - \( P_i \) sends \( x_i \) to \( P_j \) for all \( i \neq j \)
   - each party computes \( x_1 + x_2 \)
2. \( n = 2, f(x_1, x_2) = x_1 \cdot x_2 \mod q \) over \( \mathbb{F}_q \)
   - \( P_i \) sends \( x_i \) to \( P_j \) for all \( i \neq j \)
   - each party computes \( x_1 \cdot x_2 \)

Examples for 1-privacy

3. \( n = 3, f(x_1, x_2, x_3) = x_1 + x_2 + x_3 \mod q \)
   - (round 1) \( P_1 \) picks \( r \sim \mathbb{Z}_q \) and sends \( a_1 = x_1 + r \mod q \) to \( P_2 \)
   - (round 2) \( P_2 \) sends \( a_2 = a_1 + x_2 \mod q \) to \( P_3 \)
   - (round 3) \( P_3 \) sends \( a_3 = a_2 + x_3 \mod q \) to \( P_1 \)
   - (round 4) \( P_1 \) sends \( a_4 = a_3 - r \mod q \) to \( P_2, P_3 \)

- \textbf{correctness}. \( a_3 = x_1 + x_2 + x_3 + r \mod q \)
- 1-privacy.
  - \text{view}(P_1) = (x_1, r, x_1 + x_2 + x_3 + r)
  - \text{view}(P_2) = (x_2, x_1 + r, x_1 + x_2 + x_3)
  - \text{view}(P_3) = (x_3, x_1 + x_2 + r, x_1 + x_2 + x_3)

- 2-privacy?
Examples for 1-privacy

4. \( n = 4, f(x_1, x_2, x_3, x_4) = x_1 + x_2 + x_3 + x_4 \mod q \)
   - (round 1) \( P_1 \) picks \( r \in \mathbb{Z}_q \) and sends \( a_1 = x_1 + r \mod q \) to \( P_2 \)
   - (round 2) \( P_2 \) sends \( a_2 = a_1 + x_2 \mod q \) to \( P_3 \)
   - (round 3) \( P_3 \) sends \( a_3 = a_2 + x_3 \mod q \) to \( P_4 \)
   - (round 4) \( P_4 \) sends \( a_4 = a_3 + x_4 \mod q \) to \( P_1 \)
   - (round 5) \( P_1 \) broadcasts \( a_5 = a_4 - r \mod q \).

- correctness? 1-privacy?
- 2-privacy?
  - \( \text{view}(P_1, P_2) = (x_1, r, x_1 + \cdots + x_4 + r, x_2, x_1 + r, ...) \)
  - \( \text{view}(P_1, P_3) = (x_1, r, x_1 + \cdots + x_4 + r, x_3, x_1 + x_2 + r, \ldots) \)

### MPC for honest majority (BGW)

(Ben-Or, Goldwasser, Wigderson, 1988) \( n \) players and \( t \)-privacy. Fix a field \( \mathbb{F}_q \) s.t. \( q > n \).

Three results:

1. \( f \) is any polynomial of degree \( d \) and \( n \geq dt + 1 \)
2. \( f \) is a polynomial of any degree and \( n \geq 2t + 1 \)
3. (malicious) \( f \) is a polynomial of any degree and \( n \geq 3t + 1 \)

- Degree. \( f(x_1, x_2, x_3) = x_1^2 x_2 + x_1 x_3 + x_2 \) has degree 3.
- Circuits. Boolean circuits of depth \( D \) are a special case of polynomials of degree \( 2^D \). Can model \( f \) as “arithmetic” circuits over \( \mathbb{F}_q \) where the gates are + and \( \times \) (fan-in two).
- Round complexity. Protocol 1 is constant-round, whereas 2, 3 are \( O(\log d) \) rounds.

### MPC protocol 1

**Goal.** securely compute polynomial \( f \) of degree \( d \) over \( \mathbb{F}_q \)

- (round 1) \( P_i \) shares \( x_i \) using polynomial \( p_i \) of degree \( t \), i.e. sends \( a_{i,j} = p_i(j) \) to \( P_j \).
- (round 1) i.e., \( P_i \) receives \( p_1(i), \ldots, p_n(i) \).
- (round 2) \( P_i \) broadcasts \( b_i = f(p_1(i), \ldots, p_n(i)) \).

- correctness. define \( F(X) := f(p_1(X), \ldots, p_n(X)) \)
  - claim 1: \( F(0) = f(x_1, \ldots, x_n) \)
  - claim 2: \( F(i) = b_i \)
  - claim 3: \( F \) has degree at most \( dt \)
  - recover \( F(0) \) via interpolation at \( n \geq dt + 1 \) points

- \( t \)-privacy. how to simulate \( b_1, \ldots, b_n \) or \( F \)?
  - e.g. \( n = 2, t = 1 \) and \( F(t) = (u_1 t + x_1)(u_2 t + x_2) \)
MPC protocol 1

Goal. securely compute polynomial $f$ of degree $d$ over $\mathbb{F}_q$

- (round 1) $P_i$ shares $x_i$ using polynomial $p_i$ of degree $t$, i.e. sends $a_{i,j} = p_i(j)$ to $P_j$.
- (round 1) $P_1$ shares 0 using polynomial $p_0$ of degree $dt$, i.e. sends $\tilde{a}_j = p_0(j)$ to $P_j$.
- (round 2) $P_i$ broadcasts $b_i = f(p_1(i), \ldots, p_n(i)) + p_0(i)$.

• correctness. define $F(X) := f(p_1(X), \ldots, p_n(X)) + p_0(X)$
  - $F(0) = f(x_1, \ldots, x_n), F(i) = b_i$ and $F$ has degree at most $dt$

• $t$-privacy. fix $S \subset [n], |S| \leq t$
  - $F$ random polynomial of deg $dt$ s.t. $F(0) = f(x_1, \ldots, x_n)$
  - pick $\{a_{i,j} : i \in [n], j \in S\}$ at random
  - compute $\tilde{a}_j = F(j) - f(a_{1,j}, \ldots, a_{n,j})$
  - note. need all players to contribute shares of 0.

MPC protocol 2

• Goal. reduce requirement on $n \geq dt + 1$ to $n \geq 2t + 1$

• Idea. look at each addition and multiplication in $f$
  - start with shares of $x_1, \ldots, x_n$
  - if addition: just add the shares, yields polynomial of degree $t$
  - if multiplication: multiply the shares, yields polynomial of degree $2t$, then do degree reduction

• Degree reduction.
  - input: $P_i$ holds $F(i)$ where $F$ has degree $2t$
  - output: $P_i$ holds $G(i)$ where $G$ has degree $t$ and $G(0) = F(0)$
  - key idea:
    \[ G(0) = \sum_{i=1}^{2t+1} \lambda_i F(i) \]
  - run protocol 1 for $d = 1$ to compute $f(x_1, \ldots, x_n) = \sum_{i=1}^{2t+1} \lambda_i x_i$

Private simultaneous messaging

PSM (Feige-Kilian-Naor, Ishai-Kushilevitz). goal: securely compute $f$

• $n$ parties $P_1, \ldots, P_n$ share private randomness $r$
• input: $x_i$ is the private input of $P_i$
• output: referee learns $f(x_1, \ldots, x_n)$ (doesn’t see $r$)
• non-interactive: $P_i$ sends a single message to referee
security. exists \textbf{sim} s.t. \( \forall x_1, \ldots, x_n \)

\[ (P_1(x_1; r), \ldots, P_n(x_n; r)) \text{ and } \textbf{sim}(f(x_1, \ldots, x_n)) \]

are identically distributed.

**PSM: examples**

\( f(x_1, \ldots, x_n) = x_1 + \cdots + x_n \text{ over } \mathbb{Z}_q \)

- shared randomness is \( r_1, \ldots, r_n \leftarrow \mathbb{R} \mathbb{Z}_q \)
- \( P_i \) sends \( x_i + r_i \), and \( P_1 \) also sends \( r_1 + \cdots + r_n \)
- \textbf{sim}: \( P_i \) sends \( r'_i \) and ...

**PSM: examples**

\( P_1 \) holds \( i \in [n] \), \( P_2 \) holds \( D_1, \ldots, D_n \in \{0, 1\} \), referee learns \( D_i \)

- \textbf{Attempt 1}:
  - shared randomness is random permutation \( \pi \)
  - \( P_1 \) sends \( i' = \pi(i) \) and \( P_2 \) sends \( D'_j = D_{\pi^{-1}(j)} \)
  - referee outputs \( D'_{i'} = D_i \)
- \textbf{Attempt 2}:
  - shared randomness is \( \pi \) and \( r_1, \ldots, r_n \)
  - \( P_1 \) sends \( i' = \pi(i) \) and \( P_2 \) sends \( D'_j = D_{\pi^{-1}(j)} \oplus r_j \)
  - referee outputs \( D'_{i'} = D_i \oplus r_{\pi(i)} \)
  - \textbf{fix}. \( P_1 \) also sends \( r_{\pi(i)} \). how to simulate?