Secure Multiparty Computation: Introduction

Ran Cohen (Tel Aviv University)
Scenario 1: Private Dating

Alice and Bob meet at a pub

- If both of them want to date together – they will find out
- If Alice doesn’t want to date – she won’t learn his intentions
- If Bob doesn’t want to date – he won’t learn her intentions
Scenario 1: Private Dating

Alice and Bob meet at a pub

- If both of them want to date together – they will find out
- If Alice doesn’t want to date – she won’t learn his intentions
- If Bob doesn’t want to date – he won’t learn her intentions

Solution: use a trusted bartender
Scenario 2: Private Auction

Many parties wish to execute a private auction

• The highest bid wins
• Only the highest bid (and bidder) is revealed
Scenario 2: Private Auction

Many parties wish to execute a private auction

- The highest bid wins
- Only the highest bid (and bidder) is revealed

Solution: use a trusted auctioneer
Scenario 3: Private Set Intersection

Intelligence agencies hold lists of potential terrorists
- The would like to compute the intersection
- Any other information must remain secret

MI5
FBI
Mossad
Scenario 3: Private Set Intersection

Intelligence agencies hold lists of potential terrorists
• The would like to compute the intersection
• Any other information must remain secret

Solution: use a trusted party
Scenario 4: Online Poker

Play online poker reliably
Scenario 4: Online Poker

Play online poker reliably

**Solution:** use a trusted party
Secure Multiparty Computation

- In all scenarios the solution of an external trusted third party works
- Trusting a third party is a very strong assumption
- Can we do better?
- We would like a solution with the same security guarantees, but without using any trusted party
Secure Multiparty Computation

Goal: use a protocol to emulate the trusted party
The Setting

• Parties $P_1, \ldots, P_n$ (modeled as interactive TM)
• Party $P_i$ has private input $x_i$
• The parties wish to jointly compute a (known) function $y = f(x_1, \ldots, x_n)$
• The computation must preserve certain security properties, even if some of the parties collude and maliciously attack the protocol
• Normally, this is modeled by an external adversary $\mathcal{A}$ that corrupts some parties and coordinates their actions
Auction Example – Security Requirements

- **Correctness**: $\mathcal{A}$ can’t win using lower bid than the highest
- **Privacy**: $\mathcal{A}$ learns an upper bound on all inputs, nothing else
- **Independence of inputs**: $\mathcal{A}$ can’t bid one dollar more than the highest (honest) bid
- **Fairness**: $\mathcal{A}$ can’t abort the auction if his bid isn’t the highest (i.e., after learning the result)
- **Guaranteed output delivery**: $\mathcal{A}$ can’t abort (stronger than fairness, no DoS attacks)
Security Requirements

- **Correctness**: parties obtain correct output (even if some parties misbehave)
- **Privacy**: only the output is learned (nothing else)
- **Independence of inputs**: parties cannot choose their inputs as a function of other parties’ inputs
- **Fairness**: if one party learns the output, then all parties learn the output
- **Guaranteed output delivery**: all honest parties learn the output
Example – Computing Sum

- Each $P_i$ has input $x_i < M$ (work modulo $M$)
- Want to compute $\sum x_i$
- Is the protocol secure facing one corruption (semi-honest)?

\[
\begin{align*}
m_3 &= x_3 + m_2 \\
m_2 &= x_2 + m_1 \\
m_1 &= x_1 + r \\
m_4 &= x_4 + m_3 \\
m_5 &= x_5 + m_4 \\
m_6 &= x_6 + m_5 \\
r &\leftarrow \mathbb{Z}_M
\end{align*}
\]
Example – Computing Sum

- Each $P_i$ has input $x_i < M$ (work modulo $M$)
- Want to compute $\sum x_i$
- Is the protocol is secure facing one corruption (semi-honest)?
- What about two corruptions?

$$r \leftarrow \mathbb{Z}_M$$

1. $m_1 = x_1 + r$
2. $m_2 = x_2 + m_1$
3. $m_3 = x_3 + m_2$
4. $r \leftarrow \mathbb{Z}_M$
5. $m_4 = x_4 + m_3$
6. $m_5 = x_5 + m_4$
7. $m_6 = x_6 + m_5$
How to Define Security

**Option 1:** property-based definition
- Define a list of security requirements for the task
- Used for Byzantine agreement, coin flipping, etc.
- Difficult to analyze complex tasks
- How do we know if all concerns are covered?

**Option 2:** the real/ideal paradigm
- Whatever an adversary can achieve by attacking a real protocol can also be achieved by attacking an ideal computation involving a trusted party
- Formalized via a simulator
Ideal World

1) Each party sends its input to the trusted party
2) The trusted party computes $y = f(x_1, ..., x_n)$
3) Trusted party sends $y$ to each party
Real World

Parties run a protocol $\pi$ on inputs $(x_1, \ldots, x_n)$
Simulation-Based Security
Simulation-Based Security

\[ \approx \]

Distinguisher \( D \)
Simulation-Based Security

Distinguisher $\mathcal{D}$

Adversary $\mathcal{A}$

≈
Simulation-Based Security

Simulator $S$  Distinguisher $D$  Adversary $A$
Simulation-Based Security

The distinguisher $\mathcal{D}$:
- Gives inputs to parties
- Gets back output from parties and from adversary/simulator
- Guesses which world it is real/ideal

Protocol $\pi$ securely computes $f$ if $\forall \mathcal{A} \exists \mathcal{S} \forall \mathcal{D}$ distinguishing success is “small”
Sanity check

- Correctness
- Privacy
- Independence of inputs

- Fairness
- Guaranteed output delivery
Advantages of this Approach

- Very general – captures any computational task
- The security guarantees are simple to understand
  Simply imagine a trusted party computes the task
- No security requirements are ”missed”
- Supports sequential modular composition
  - Security remains when secure protocols run sequentially
  - A single execution at a time
  - Arbitrary messages can be sent between executions
- Useful for modular design of protocols
Sequential Modular Composition

• Design a protocol in a **hybrid model**
  – Similar to the stand-alone real world
  – A trusted party helps to compute some functionality \( f \)
  – In rounds with calls to \( f \) no other messages are allowed

• **Theorem** (informal)
  – Protocol \( \pi \) securely computes \( g \) in the \( f \)-hybrid model
  – Protocol \( \rho \) securely computes \( f \)
  – Then, protocol \( \pi \rho \) securely computes \( g \) in the real world

Replace ideal calls to \( f \) with real protocol \( \rho \)
The Definition Cont’d

A definition of an MPC task involves defining:

• **Functionality**: what do we want to compute?

• **Security type**: how strong protection do we want?

• **Adversarial model**: what do we want to protect against?

• **Network model**: in what setting are we going to do it?
The Functionality

• The code of the trusted party
• Captures inevitable vulnerabilities
• Sometimes useful to let the functionality talk to the ideal-world adversary (simulator)

We will focus on secure function evaluation (SFE), the trusted party computes \( y = f(x_1, \ldots, x_n) \)
  • Deterministic vs. randomized
  • Single public output vs. private outputs
  • Reactive vs. non-reactive
Security Type

- **Computational**: a PPT distinguisher
  - The real & ideal worlds are computationally indistinguishable

- **Statistical**: all-powerful distinguisher, negligible error probability
  - The real & ideal worlds are statistically close

- **Perfect**: all-powerful distinguisher, zero error probability
  - The real & ideal worlds are identically distributed
Adversarial Model (1)

• Adversarial behavior
  – **Semi honest**: honest-but-curious. corrupted parties follow the protocol honestly, $A$ tries to learn more information. Models inadvertent leakage
  – **Fail stop**: same as semi honest, but corrupted parties can prematurely halt. Models crash failures
  – **Malicious**: corrupted parties can deviate from the protocol in an arbitrary way
Adversarial Model (2)

- Adversarial power
  - *Polynomial time*: computational security, normally requires cryptographic assumptions, e.g., encryption, signatures, oblivious transfer
  - *Computationally unbounded*: an all-powerful adversary, information-theoretic security
Adversarial Model (3)

- **Adversarial corruption**
  - **Static**: the set of corrupted parties is defined before the execution of the protocol begins. Honest parties are always honest, corrupted parties are always corrupted.
  - **Adaptive**: $\mathcal{A}$ can decide which parties to corrupt during the course of the protocol, based on information it dynamically learns.
  - **Mobile**: $\mathcal{A}$ can “jump” between parties. Honest parties can become corrupted, corrupted parties can become honest again.
Adversarial Model (4)

• Number of corrupted parties
  – Threshold adversary:
    Denote by $t \leq n$ an upper bound on # corruptions
    ➢ No honest majority, e.g., two-party computation
    ➢ Honest majority, i.e., $t < n/2$
    ➢ Two-thirds majority, i.e., $t < n/3$
  – General adversary structure:
    Protection against specific subsets of parties
Communication Model (1)

• **Point-to-point**: fully connected network of pairwise channels.
  – Unauthenticated channels
  – **Authenticated channels**: in the computational setting
  – **Private channels**: in the IT setting
  ➢ Partial networks: star, chain

• **Broadcast**: additional broadcast channel
Communication Model (2)

• Message delivery:
  – **Synchronous**: the protocol proceeds in rounds. Every message that is sent arrives within an known time frame
  – **Asynchronous (eventual delivery)**: the adversary can impose arbitrary (finite) delay on any message
  – **Fully Asynchronous**: the adversary has full control over the network, can even drop messages
Execution Environment

- **Stand alone:**
  - A single protocol execution at any given time (isolated from the rest of the world)

- **Concurrent general composition:**
  - Arbitrary protocols are executed concurrently
  - An Internet-like setting
  - Requires a strictly stronger definition
    Captured by the universal composability (UC) framework
  - Impossible in general without a trusted setup assumption (e.g., common reference string)
Relaxing the Definition

• Recall the ideal world (with guaranteed output delivery)
  1) Each party sends its input to the trusted party
  2) The trusted party computes $y = f(x_1, \ldots, x_n)$
  3) Trusted party sends $y$ to each party
• This ideal world is overly ideal
• In general, fairness cannot be achieved without an honest majority [Cleve’86]
• A relaxed definition is normally considered
Security with Abort

• Ideal world without fairness and guaranteed output delivery:

1) Each party sends its input to the trusted party
2) The trusted party computes \( y = f(x_1, ..., x_n) \)
3) Trusted party sends \( y \) to the adversary
4) The adversary responds with continue/abort
5) If continue, trusted party sends \( y \) to all parties
   If abort, trusted party sends \( \bot \) to all parties

• Correctness, privacy, independence of inputs are satisfied
Prevalent Models

• In the seminar we will consider:
  – Adversary: semi honest / malicious with static corruptions
  – Synchronous P2P network with a broadcast channel
  – Stand-alone setting

• Computational setting
  – PPT adversary & distinguisher (computational security)
  – Arbitrary number of corruptions $t < n$
  – Authenticated channels

• Information-theoretic setting
  – All powerful adversary & distinguisher (perfect/statistical)
  – Honest majority $t < n/2$ (if $t < n/3$ no need for broadcast)
  – Secure channels
Oblivious Transfer

$m_0, m_1 \xrightarrow{\text{}} b \in \{0, 1\} \xleftarrow{\text{}} m_b$
Feasibility Results

- Malicious setting
  - For $t < n/3$, every $f$ can be securely computed with perfect security [BGW’88, CCD’88]
  - For $t < n/2$, every $f$ can be securely computed with statistical security [RB’89]
  - For $t < n$, assuming OT, every $f$ can be securely computed with abort and computational security [GMW’87]

- Semi-honest setting
  - For $t < n/2$, every $f$ can be securely computed with perfect security [BGW’88, CCD’88]
  - For $t < n$, assuming OT, every $f$ can be securely computed with computational security [GMW’87]
Outline of the Seminar

• Lecture 2: definitions
• Lectures 3-7: semi-honest setting
  – Yao’s garbled circuit
  – Oblivious transfer
  – GMW protocol [Goldreich, Micali, Wigderson’87]
  – BGW protocol [Ben-Or, Goldwasser, Wigderson’88]
  – BMR protocol (constant-round MPC) [Beaver, Micali, Rogaway’90]
• Lectures 8-11: malicious setting
  – GMW compiler
  – IKOS zero-knowledge proof
  – Cut and choose (Yao’s protocol for malicious)
  – Sigma protocols
• Lecture 12: specific functionalities (median, PSI)
Summary

• Secure multiparty protocols emulate computations involving a trusted party

• Impressive feasibility results: every task that can be computed can also be computed securely

• Many different models and settings

• Exciting and active field – many open questions