1 Multi Party Computations

In this part of the lecture

1.1 Cryptography

Cryptography provides tools for secure communication between two parties using un-secure environment for message transfers.

We will learn how to design and analyze protocols that overcome the influence of adversaries. Cryptography protocols can ensure:

- confidentiality
- integrity
- authenticity
- non-repudiation

1.2 Cryptography primitives

<table>
<thead>
<tr>
<th>crypto primitive</th>
<th>useful for</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>encryption schemes</td>
<td>confidentiality</td>
<td>AES RSA</td>
</tr>
<tr>
<td>signature schemes</td>
<td>authenticity and non-repudiation</td>
<td>ElGamal signature GHR</td>
</tr>
<tr>
<td>MACs</td>
<td>authenticity and integrity</td>
<td>authenticity and integrity</td>
</tr>
<tr>
<td>hash functions</td>
<td>integrity</td>
<td>SHA-256</td>
</tr>
</tbody>
</table>

1.3 Security

How to prove that the protocol is secure?

- Heuristic approach
- Rigorous approach
1.4 Heuristic approach I

1. Build a protocol.
2. Try to break the protocol.
3. Fix the break.
4. Go to 2.

Problems:
- Never can be sure that the protocol is secure.
- Real adversaries don’t tell you their breaks.

Example: GSM protocol. This was private protocol. Here you can read why it’s not secure any more.

1.5 Heuristic approach II

1. Build a protocol.
2. Provide a list of attacks that provably cannot be launched on the protocol.
3. Reason that the list is complete.

Problems:
- Often the list is not complete

1.6 Rigorous approach

1. Provide an exact problem definition.
   - meaning of security
   - adversarial power
   - capabilities of the network
2. Prove that the protocol is
   - perfectly secure, e.g. one-time pad
   - computationally secure, e.g. RSA

Note: ! Randomness is expensive.
1.7 Computational security

1. Concrete approach

- A scheme is $(t, \epsilon)$ secure if every adv. running in time at most $t$ succeeds in breaking the scheme with probability at most $\epsilon$

2. Asymptotic approach

- A scheme is secure if every PPT adv. succeeds in breaking the scheme with only negligible probability

Example: Scheme with 60 bit key. $t$ computer cycles to break the system with probability $\frac{t}{2^{60}}$

2 Ghz (2 * $10^9$ cycles/sec) $\frac{2^{60}}{2^{10^9}} \approx 18$ years

1.8 Asymptotic approach

A scheme is secure if every PPT adversary succeeds in breaking the scheme with only negligible probability.

**Definition 1** Efficient algorithm is algorithm, s.t

- is probabilistic
- polynomial time: $\exists$ const $a, c$ and the running time is $a \ast n^c$

**Definition 2** Negligible function is such a function with small probability of success. (Smaller than any inverse polynomial) $\forall$ constants $c$ the adversary success probability is smaller that $n^{-c}$ for large enough values of $n$.

Example: Adversary run in time $n^3$ minutes. He can break the scheme with probability $\frac{2^{40}}{2^{n^3}}$

- $n \leq 40$, success $\frac{2^{40}}{2^{n^3}} = 1 \approx 44$ days
- $n \leq 50$, success $\frac{2^{40}}{2^{n^3}} = \frac{1}{2^{10}} \approx 3$ months
- $n = 500$, success $\frac{2^{40}}{2^{n^3}} = \frac{1}{2^{160}} \approx 2040$ years

1.9 Adversaries

- cipher text only - passive adversary
- known plain text - adv. knows (part of) the message, that is exchanged
- chosen plain text - adv. can play with the encryption mechanism; minimum requirement of PKC
- chosen cipher text - adv. can play with the decryption mechanism
- adaptive chosen cipher text - adv. can play with the decryption mechanism and can adapt the queries
1.10 Multi Party Computations

MPC is a subfield of the Cryptography. It is related to zero-knowledge proof systems. Formally introduced by A. C. Yao in 1982.

We want to compute

- Parties or players are denoted $P_1, P_2, \ldots, P_n$
- Each party holds a secret input $x_i$ and the players agree on some $n$-input function $f$.
- Multi output case:
  $$(y_1, y_2, \ldots, y_n) = f(x_1, x_2, \ldots, x_n)$$
- Single output case:
  $$y = f(x_1, x_2, \ldots, x_n)$$
- Single output case with randomness:
  $$y = f(x_1, x_2, \ldots, x_n; r)$$

Example: Tao’s Millionaires’ Problem

- Two millionaires wish to compute who is richer without revealing their wealth.

\[
f(x_1, x_2) = \begin{cases} 
1 & \text{if } x_1 < x_2 \\
0 & \text{if } x_1 \geq x_2 
\end{cases}
\]

$x_1$ and $x_2$ are the amounts of money which millionaires hold

Example: Voting

- There are two candidates $C_0$ and $C_1$.
- There are $n$ voters
- To vote for $C_0$ submit $x_i = 0$
- To vote for $C_1$ submit $x_i = 1$
- Who is the winner?

\[
f(x_1, \ldots, x_n) = \begin{cases} 
C_0 & \text{if } \sum_{i=0}^{n} x_i < \frac{n}{2} \\
C_1 & \text{otherwise}
\end{cases}
\]

- How many votes do the candidates have?

\[
f(x_1, \ldots, x_n) = (\#C_0, \#C_1) = (n - \sum_{i=0}^{n} x_i, \sum_{i=0}^{n} x_i)
\]

Example: Sealed Bid Action

- $n$ bidders
- $x_i$ is the bid of the $i$-th bidder
- Announce the winner and price to $f(x_1, x_2, \ldots, x_n; r) = (\max_{i} x_i, i)$
- Tell the bidders whether they won or lost the bidding $f(x_1, x_2, \ldots, x_n; r) = (\ldots, l, w, l, \ldots)$
1.11 Challenges

- Keep private data private:
  - Millionaires do not want to tell how much money they have.
  - Voters do not want to tell their vote.
  - Auctioneers do not want to reveal their bid.

- Compute function correctly
  - Who guarantees the common function is computed correctly

[Image goes here]

1.12 Adversaries

- Malicious vs. semi-honest adversary
  - Semi-honest (passive): the adversary behaves as specified, but he tries to learn additional information
  - Malicious (active): the adversary does not behave as specified

- Static vs. adaptive
  - Static: the adversary corrupts a number of parties, that is fixed from the beginning
  - Adaptive: the adversary corrupts parties as he sees fit

- Complexity: Most of the time PPT

- Monolithic adversary: one adversary controls a subset of parties.

1.13 Network model

- Authenticated channels
- All parties share an authenticated channel
- All parties are connected point to point
- Synchronous / asynchronous
- Message delivery guaranteed?
- Are there other protocols executed in the environment? Broadcasting: who guarantees that all parties receive the same?
- Consensus broadcast: all honest parties receive the same, even if sender is malicious

**Definition 3** (informal) A real protocol that is run by the parties (in a world where no TTP exists) is secure if an adversary cannot profit more in a real execution than in an execution that takes place in the ideal world.

**Definition 4** For any adversary that launches a successful attack on the real protocol there exists an adversary that can carry out the same attack in the ideal world.
1.14 Ideal world

- Given an ideal functionality F (judge) all parties can send their inputs to and receive outputs from F.
- send/receive privately
- F executes a certain number of commands.
- F is incorruptible, always correct, nothing leaks.

1.15 Secure addition

$n = 3$ players

$P_1 : x$
$P_2, P_3$ - they together can revert the secret

$x_1 \in \{0, ..., p - 1\}$, $x_1 \in Z_p$

$P_1$ chooses $r_1, r_2 \in \mathbb{Z}_p$

$r_3 = x_1 - r_1 - r_2 \mod p$

Example for secret sharing of one player:

<table>
<thead>
<tr>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_2$</td>
<td>$r_1$</td>
<td>$r_1$</td>
</tr>
<tr>
<td>$r_3$</td>
<td>$r_3$</td>
<td>$r_2$</td>
</tr>
</tbody>
</table>

Let $P_1 : x_1, P_2 : x_2, P_3 : x_3$ and $x_1, x_2, x_3 \in Z_p$

$S = x_1 + x_2 + x_3 \mod p$

$r_{1,3} = x_1 - r_{1,1} - r_{1,2} \mod p$, where $r_{1,1}, r_{1,2} \in \mathbb{Z}_p$

$r_{2,3} = x_2 - r_{2,1} - r_{2,2} \mod p$, where $r_{2,1}, r_{2,2} \in \mathbb{Z}_p$

$r_{3,3} = x_3 - r_{3,1} - r_{3,2} \mod p$, where $r_{3,1}, r_{3,2} \in \mathbb{Z}_p$

Step 1: Exchange values following the protocol discussion above

<table>
<thead>
<tr>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{1,2}$</td>
<td>$r_{1,1}$</td>
<td>$r_{1,1}$</td>
</tr>
<tr>
<td>$r_{1,3}$</td>
<td>$r_{1,3}$</td>
<td>$r_{1,2}$</td>
</tr>
<tr>
<td>$r_{2,2}$</td>
<td>$r_{2,1}$</td>
<td>$r_{2,1}$</td>
</tr>
<tr>
<td>$r_{2,3}$</td>
<td>$r_{2,3}$</td>
<td>$r_{2,2}$</td>
</tr>
<tr>
<td>$r_{3,2}$</td>
<td>$r_{3,1}$</td>
<td>$r_{3,1}$</td>
</tr>
<tr>
<td>$r_{3,3}$</td>
<td>$r_{3,3}$</td>
<td>$r_{3,2}$</td>
</tr>
</tbody>
</table>

Step 2: Everyone computes

<table>
<thead>
<tr>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_2$</td>
<td>$S_1$</td>
<td>$S_1$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$S_3$</td>
<td>$S_3$</td>
</tr>
</tbody>
</table>

$S_1 = r_{1,1} + r_{2,1} + r_{3,1} \mod p$

$S_2 = r_{1,2} + r_{2,2} + r_{3,2} \mod p$

$S_3 = r_{1,3} + r_{2,3} + r_{3,3} \mod p$

$x_1 + x_2 + x_3 = S_1 + S_2 + S_3 = S$