Secure multiparty computation

Second part

Dáša Krasnayová

Oblivious transfer

- It is a functionality for two parties - sender and receiver.
- Sender's input is a pair of strings (z_0, z_1) and there is no output.
- Receiver's input is a bit b and output is z_b.

Oblivious transfer

- It is a functionality for two parties - sender and receiver.
- Sender's input is a pair of strings (z_0, z_1) and there is no output.
- Receiver's input is a bit b and output is z_b.

How to compute this securely? (Sender does not learn anything and receiver learns z_b only.)

Trapdoor permutation and enhanced trapdoor permutation

- A *trapdoor permutation* (TDP) is a 4-tuple of algorithms (I, D, F, F^{-1}), where
  - I samples a function f and a trapdoor t in a family,
  - $D(f)$ uniformly samples a value in a domain of f,
  - $F(f, x)$ computes $f(x)$,
  - $F^{-1}(f, y, t)$ computes $f^{-1}(y)$

  and it is hard to invert f given y but not t.
- An *enhanced trapdoor permutation* is a TDP for which it is hard to compute $f^{-1}(y)$ even given the random coins used to sample y (using D).

TDP vs. ETDP example

- RSA trapdoor function
  - I chooses random $((e,n),d)$, where $n = p \cdot q$ for some p,q prime and $e \cdot d = 1(\mod \varphi(n))$,
  - D chooses random value in $\mathbb{Z}_n$,
  - $F(f,x) = x^e \mod n$,
  - $F^{-1}(f,y,t) = y^d \mod n$.
- Rabin trapdoor function
  - I chooses random $(n,(p,q))$, such that $n = p \cdot q$, $p = q = 3 \pmod 4$, p,q prime,
  - D chooses random value in $\mathbb{Z}_n$ and squares it,
  - $F(f,x) = x^2 \mod n$,
  - $F^{-1}(f,y,t)$ as in Rabin cryptosystem.

Hard-core predicate

- A function $B(x)$ is a *hard-core predicate* if $B(x)$ is a bit and probability of guessing $B(x)$ given $y = f(x)$ is only negligibly larger than one half. (Given $y = f(x)$, the bit $B(x)$ is pseudorandom.)
- Example: Let r be a string of the same length as $f(x)$. Then function

$$b(x,r) = \bigoplus_j r_j x_j$$

is a hard-core predicate.

Construction of OT protocol for semi-honest adversary

Suppose that (I,D,F,F^{-1}) is an enhanced TDP and B is a hard-core predicate.
Sender's input is a pair of bits (z_0,z_1),
Receiver's input is a bit b.

- Sender chooses (f,t) using sampling algorithm I and sends f to the receiver.
- Receiver chooses x_b and computes $y_b = f(x_b)$. Receiver chooses random y_{1-b} using D and sends (y_0,y_1) to the sender.
- Sender inverts (y_0,y_1) getting (x_0,x_1). Sender computes $a_i = z_i \oplus B(x_i)$ for $i = 0,1$ and sends (a_0,a_1) to the receiver.
- Receiver computes $z_b = a_b \oplus B(x_b)$.

Construction is computationally secure

We will show that there is a simulator that generates a transcript indistinguishable from a transcript of a real protocol execution.

- Sender corrupted: Simulator is given sender's input (z_0, z_1) and there is no output. Simulator chooses (f, t) using I, y_0, y_1 using $D(f)$ and computes a_0, a_1. The transcript is exactly like in a real protocol execution because choosing x_b and computing $y_b = f(x_b)$ is identical to choosing y_b using $D(f)$.
- Receiver corrupted: Simulator is given receiver's input b and the output z_b. Simulator chooses (f, t) using I, x_b, y_{1-b} using $D(f)$ and computes $y_b = f(x_b)$, $a_b = z_b \oplus B(x_b)$. Simulator finally chooses a_{1-b} at random. Since B is a hard-core predicate and f is enhanced, $B(x_{1-b})$ is indistinguishable from random. Therefore this simulator output is indistinguishable from a real execution.

Malicious adversary – Ideal / Real paradigm

- In ideal world a trusted authority computes output.
- All parties just send their inputs to the TA through secure channels.

Malicious adversary – Security definition

Protocol π *securely computes* a function f, if for every non-uniform polynomial-time real-model adversary **A**, there exists a non-uniform polynomial-time ideal-model simulator **S**, such that for all input vectors and auxiliary inputs the joint outputs of **A** and the honest parties in real execution of π is indistinguishable from the joint outputs of **S** and the honest parties in an ideal execution where the trusted party computes the f.

Malicious adversary – Security definition

The following properties hold

- privacy - from adversary's output
- correctness - TA computes the functionality
- independence of inputs - ideal execution
- fairness and guaranteed output delivery - ideal execution

Relaxing the definition

- Sometimes this definition is too strong
- For example fairness cannot be guaranteed without an honest majority
  - We sometimes change the instructions of the trusted party

Secure computation of AND function

For parties **A** and **B** with input bits a and b we want to compute $a \cdot b$ securely.

- Let **A** be the sender with input $(0,a)$ and **B** be the receiver with input b.
- Now we execute the OT protocol and finally **B** sends output to **A**.
- Another option is to execute the protocol twice, exchanged roles in the second execution.

Claim: If OT protocol is secure, this computation is secure as well.

- Simulation is easy.

Feasibility of constructing OT

There is no OT protocol providing unconditional security for both parties.

- We will prove this by proving there is no unconditionally secure protocol computing AND function.
- I there was an OT protocol providing unconditional security for both parties, we could construct AND protocol with unconditional security.

Feasibility of constructing OT (cont.)

Suppose there is an AND protocol with unconditional security.

- Let T be a transcript of a real execution.
- Parties use random inputs R_A and R_B, given these inputs protocol is a deterministic function.
- Let **A** (sender) be the corrupted party and suppose that in a certain execution **A** has input 0.
  - If **B**'s input is 0, **B** must not learn **A**'s input. Hence there is an R' such that if **A** has input $a = 1$ and R', the transcript of the execution of the protocol would be T.
  - If **B**'s input is 1, there is no R' such that if **A** has input $a = 1$ and R', the transcript of the execution of the protocol would be T due to correctness.

Feasibility of constructing OT (cont.)

- **A** can therefore compute b by determining whether there is an R' such that the transcript of the execution of the protocol would be T if **A**'s input was $a = 1$ and R'.

Sequential modular composition

- In a protocol, secure protocols are run sequentially as subroutines, with arbitrary messages in between them.
- Formalization of the security - Hybrid model
  - A trusted party computes a sub-functionality

Sequential modular composition

- If subprotocols ρ_i securely computes functionalities f_i and a protocol π securely computes functionality g in a hybrid model where a trusted party is used to compute every f_i, than a real protocol π^ρ that uses real calls to each ρ_i instead of a trusted party, securely computes g.