Secure Multi-Party Computation

Boran Erol

February 2024

1 TO-DO

Summarize your notes from NPTEL.

Recall the different dimensions along which you can analyze MPC schemes.

Bar Ilan University Winter School

Resources for MPC

How To Simulate It by lindell

Goldreich Foundations of Crypto, Volume II, Chapter 7

2 Introduction and Definitions

Here's a simple example that demonstrates the need for joint distributions:

Adversarial models:

1. Semi-honest
2. Covert
3. Malicious

The following definition of security for 2PC is from "How To Simulate It":

Definition 1. Let $f = (f_1, f_2)$ be a functionality.

Notice that a malicious adversary can provide a fake input. This is an inherent lack of security in MPC protocols, since the same "attack" can be carried out in the ideal model.

How far are the implications of this?

The following is the definition from Goldreich:

Definition 2. Let f be an m-ary functionality and Π be an m-party protocol operating in the real model.

For a real model adversary A, controlling some minority of the parties, and an $m-$sequence $\bar{x}$, we denoted by $REAL_{\Pi,A}(\bar{x})$ the sequence of m outputs resulting from the execution of Π on input $\bar{x}$ under the attack of adversary A.

For an ideal model adversary A', controlling some minority of the parties, and an $m-$sequence $\bar{x}$, we denoted by $IDEAL\Pi, A'(\bar{x})$ the sequence of m outputs resulting from the ideal process on input $\bar{x}$ under the attack of adversary A'.

We say that Π **securely implemenets** f **with honest majority** if for every feasible real-model adversary A, there's an A' controlling the same parties such that

$$REAL_{\Pi,A}(\bar{x}) \approx_c IDEAL\Pi, A'(\bar{x})$$

MPC with a dishonest majority is possible iff premature suspension of the execution is not considered a breach of security. Notice that this is always the case in 2PC. *If the output of both parties depend on both inputs, one party can suspend the execution early to deny the other party their output.* Therefore, in the case of 2PC, we have to change the ideal model to allow aborts. Other than this modification, the security definition remains identical.

Notice that if we restrict our attention to *single-output functionalities*, premature suspension is no longer an issue.

3 Overview of Main Results

Theorem 3.1. Assuming enhanced trapdoor permutations exist, general MPC is possible in the following 3 models:

1. Semi-honest setting for any number of dishonest parties
2. Malicious setting with a strict minority
3. Malicious setting with premature suspension for any number of dishonest parties

In all ofthese cases, the adversary is computationally bounded and non-adaptive. The adversary may tap the communication lines between honest parties. The results in the malicious setting assume a broadcast channel.

Here are information-theoretic results:

Theorem 3.2. Assuming the existence of private channels, general MPC is possible in the following models:

1. Semi-honest setting with a strict minority
2. Malicious setting with less than 1/3 of the parties corrupt. If we assume a broadcast channel, we can bring this up to 1/2.

The adversary can also be adaptive.

4 Converting Semi-Honest Security to Malicious Security

4.1 Using Secret-Sharing

1. Each party commits to their input using a commitment scheme. *Using a ZKPOK, each party proves that they can decommit to the commitment they sent.* (This ensures that they can't run replay attacks.)
2. All parties jointly generate a sequence of random bits for each party such that only this party knows the outcome of the random sequence and everybody else gets a commitment of this outcome.
3. Each party shares their input and randomness using a VSS. *This helps prevent against premature suspension.*
4. During each computation step, each party provides a ZK proof that the message was computed according to the protocol. This can indeed be verified since all the information used to compute is public in the form of commitments.

4.2 Using Scrambled Circuits

This is mainly for 2PC. Yao's Garbled Circuits.

4.3 Secret-Sharing Circuits

These are BGW-like schemes where we compute the gates of a circuit in secret-shared fashion.

Goldreich implements AND using OT, which is interesting. More details in Section 7.3.

Goldreich covers BGW in Section 7.6.

5 2PC

There are certain complete functions out of which we can information theoretically build arbitrary functions.

One of the most famous examples is Yao's Millionaire Problem. It's basically evaluating $x \geq y$ without revealing x or y.

We'll first consider MPC in the semi-honest adversary setting. We can then upgrade this to the fully-malicious setting using **zero-knowledge proofs of honesty**.

We can consider 3 types of functionalities:

1. Symmetric deterministic functionalities
2. Input-oblivious randomized functionalities
3. Asymmetric functionalities

Here are some simplifying assumptions we make to simplify the exposition:

1. $|x| = |y|$
2. f is computable in polynomial-time.
3. Security is measured in terms of the length of the inputs, denoted n.

Observe that making no restriction on the relationship among the lengths of x and y only allows trivial protocols.

I don't understand how Goldreich justifies the other two assumptions. Read again.

We again let adversaries have auxiliary input since we want to compose secure protocols. We suppress their existence whenever they're irrelevant. Thus, we have a non-uniform formulation of security.

5.1 Security in the Semi-Honest Setting

We can either formulate security following in the footsteps of ZK or we can use the real-ideal paradigm.

5.2 Security in the Malicious Setting

Some preliminary comments:

1. Perfect fairness is not achievable in malicious 2PC, since one party can abort whenever. There are some partial fairness results. 2. The malicious party can provide fake input to the protocol.

6 Secret Sharing

6.1 Additive Secret Sharing

The idea behind additive secret sharing is simple. Let $x \in \mathbb{Z}/p\mathbb{Z}$. Just pick two random $x_1, x_2 \in \mathbb{Z}/p\mathbb{Z}$ such that $x_1 + x_2 = x$. Then, knowing x_1 and x_2 doesn't divulge anything about x but knowing both of them fully determines x.

Notice that when $p = 2$ this just corresponds to XOR.

7 Oblivious Transfer

1-out-of-2 Oblivious Transfer

I'll always call the sender Alice and receiver Bob.

Alice has bits b_0, b_1. Bob gets one of these bits. Alice doesn't learn which bit the receiver got, and Bob doesn't learn anything about the other bit.

Michael Rabin's OT

With probability 0.5, Bob gets the bit. With probability 0.5, Bob gets nothing.

Rabin's OT can be realized using quantum crap.

Lemma 7.1. These two primitives are equivalent.

Suppose we have 1-out-of-2 OT. Here's how we construct an AND function out of this in the **semi-honest** setting:

Suppose Alice has a bit x and Bob has a bit y. Alice calculates $b_0 := x \wedge 0$ and $b_1 := x \wedge 1$. Bob picks b_y.

Notice that if the function evaluates to 1, both parties learn the other party's input. Also, if one party has a 1 and the other party has a 0, the party with a 1 learns the other party's input. However, the party with a 0 doesn't learn anything.

Also notice that Bob can hang up and prevent Alice from learning the function output.

8 GMR Protocol

9 MPC using OT

Apart from assuming OT, this construction is fully information-theoretic.

Recall that every function f can be converted to a Boolean circuit. Some of the input to this Boolean circuit comes from Alice (call it x) and some of it comes from Bob (call it y). We'll additively secret share these values and then evaluate the circuit using the secret-shared values. Therefore, the problem reduces it evaluating any gate using additively secret shared values.

Let's first handle the NOT and XOR gates, which can be done in non-interactive fashion.

NOT Gates

Alice can just flip her input.

XOR gates

Both Alice and Bob XOR their secret shares.

Question: Is it proven that AND gates require interaction?

AND Gates

Let x, y be the inputs. Suppose Alice has x_1 and y_1 and Bob has x_2 and y_2. We'll write $+$ to mean XOR (think of it as addition modulo 2).

Notice that $xy = (x_1 + x_2)(y_1 + y_2) = x_1y_1 + x_1y_2 + x_2y_1 + x_2y_2$.

Alice can calculate x_1x_2 on her own and Bob can calculate y_1y_2 on his own. The only issue is with the cross terms, which we can handle using OT. However, we don't want either party to learn the cross terms, so we add some randomness to ensure that the cross terms are also additively secret-shared.

10 Yao's Garbled Circuits

This was the first MPC protocol for 2PC.

This is an asymmetric protocol. We need a garbler and an evaluator.

NPTEL has a good explanation.

The BMR protocol is an extension of Yao to n parties in $O(1)$ rounds.

11 GMW Protocol for 1-out-of-2 OT

Alice uses KeyGen to generate (f, f^{-1}).

12 Round Complexity of MPC

So far, both protocols we've covered had a round complexity that was linear in the depth of the circuit. We'll now aim to reduce the round complexity of these protocols.

We have two kinds of protocols: asynchronous and synchronous protocols.

Synchronous protocols have a global clock. At every tick of the clock, the players send a message.

We can have deadlock in asynchronous protocols. We won't cover this.

It turns out that we can create constant round protocols. In fact, Rafi has a sequence of papers that proves that four rounds is necessary and sufficient for computationally secure protocols.

Whether we can have information-theoretic constant round protocols is an open question. The best upper bound is $O(\frac{c}{\log c})$, and improving this would mean improving lower bounds on circuits, which is known to be difficult.

A randomized encoding takes a function f and some randomness r and maps this tuple $(f, r) \mapsto \hat{f}$ and $(x, r) \mapsto \hat{x}$ such that $f(x) = \hat{f}(\hat{x})$.

For all x, x' such that $f(x) = f(x')$,

$$(\hat{f}, \hat{x}) \approx_c (\hat{f}, \hat{x}')$$

It turns out that it is possible to achieve this with no assumptions for constant-depth circuits. For deeper circuits, however, we need an assumption such as one-way functions.

One player will play the role of the evaluator and another player will play the role of the garbler.

We should only be able to decrypt one row of the table.

How do we check whether we are able to encrypt?

One way is to add a hundred 100's before every string and check whether the decryption has a 100 zeros at the start of the string.

A more clever method is to use point and permute.

There are intel AES-accelerators to evaluate Yao gates. This uses the random oracle model and some additional assumptions since loading the key takes a long time but evaluating for a fixed key is simple.

Proving the security of this construction turns out to be delicate. We'll therefore carry out the rigorous proof.

There's a difference between first sending the garbled circuit and letting Evan pick his input and swapping the order. Obviously, first sending the garbled circuit is harder to prove secure. This is called adaptive Yao security. We won't cover this in class, but apparently it's doable.

In this class, we'll assume Ginny sends the garbled circuit after Evan gets his keys using OT. Notice that the other scenario might be useful in the following scenario: Suppose we have a startup and we let people evaluate a useful, expensive function using our garbled circuit. In this case, we have to publish the garbled circuit before Evan picks his keys.

The circuit that we're garbling might be a universal circuit, so we can even turn to circuit we want to run into an input and hide the circuit.

13 A Math Puzzle

Suppose we have n parties, each with a bit x_i. To calculate the XOR, we need a single random bit.

To calculate the AND, Rafi proved that we need at least 2 bits and at most 8 bits.

Open Problem: Can we do it using less than 8 bits?