CSA E0 312: Secure Computation 11 Aug 2015

[Lecture 1]

Instructor: Arpita Patra *Submitted by: Ajith*

1 Secure Multi-party Computation (MPC): Introduction

We are in the age of information. Every human being, organization (public/private), nation holds/ is attached with a huge body of information. Some information is made available in public whereas the rest demands privacy. A few examples are tabulated in Table 1.

Source of data	**Public data**	**Private Data**
Individual	Identity details (passport no., PAN card, Voter ID, AADHAR id), Bank Details (netbanking login), Income Tax Details, Your vehicle details (cycle, two wheeler, car)	Age, Salary Bank Details (balance, password),Medical data: diseases, biometric traits (face, fingerprint, iris, speech), genome signature, minimum age of watching porn/taking drug, Child adoption details.
Educational Organization (IISc/IITs/IIITs/IISERs /NISERs/NITs)	List of Employees and Students, awards, recognitions, scientific publications, products	Employee details, student details ,drop-out details, drug addicts, suicides, sexual harassments
Profitable Organization (MS/IBM/TCS/Infosys)	CEO, Board of Managers, Names of the employes	Employees and their details, Profit, loss, turnover, salaries.
Hospitals	List of patients, doctors, nurses	Patient's medical history and diagnosis details, Details of Doctors and nurses.
Security Agencies (RAW/IB/CBI/NIA)	List of criminals and details, list of incidents and details	List of employees and details , list of intercepted messages and their details
Military Organizations (Army/Air Force/Navy)	List of soldiers, colonels and details, list of operations	Operation details, list of intercepted messages etc and their details
Country	List of citizens and details, prime minister, president, MLA, MPs, celebrities, under-privileged	Satellites / Nuclear weapons / Submarines information

Table 1: Types of data and their resources

The problem of secure communication is formulated to enable secure transfer of sensitive and private information. In a secure communication problem, a sender wants to send its private information to a receiver in a way that no third party gets the information by eavesdropping over the communication medium. Though it is non-trivial to solve the problem of secure communication, the goal is well-stated and well-understood. We know how to solve the problem, namely with the help of *encryption schemes*. What about computing on private information? There are numerous situations out there in practice where computing on private information can be of great interest. A large amount of added value can be obtained by combining confidential information from several sources and from this computing some result that holds an interest for all parties. In what follows, we consider a few examples/applications that demand protecting privacy of data while they undergo computation:

Secure Auction: Every party possesses bid amount as his private input. The goal is to compute the maximum bid and the corresponding bidder ensuring nothing beyond the winner and the winning bid are revealed.

Satellite Collision: The orbital information of a satellite is a private data and is possessed by a country which owns it. In order to prevent the collision among different satellites, the goal is to compute the collision probability without leaking the high accuracy positional information of the satellites.

Privacy Preserving Data Mining: Hospitals possesses patient records, which are private data. Several leading hospitals in India would like to find the number of patients who have been affected with a particular disease between say, 2-14 and 2015 without revealing their patient database to each other.

There are many scenarios like above that demand both privacy and computation on private data simultaneously such as secure benchmarking, secure set intersection, private information retrieval etc. The area of *Secure Computation* or more commonly known as *Secure Multi-party Computation (MPC)* has evolved naturally to tackle the above problem. The examples when abstracted out will lead to an (informal) definition of MPC presented next.

1.1 MPC: Informal Definition

Let's try to formalize the problem of MPCn. Our goal is to create methods for parties to jointly compute a function over their inputs, and keeping these inputs private.

Definition 1 Secure Multi-party Computation (informal): There are n parties $P_1, P_2, ..., P_n$ who do not trust each other. Each party P_i has its own private input x_i and there is a common n-input function, f which every party wants to compute on their private data. The goals l are as follows:

- **Correctness:** Every party must output $y = f(x_1, x_2, ..., x_n)$.
- **Privacy:** Nothing about the inputs of the parties must be leaked.

♢

Can the privacy requirement be achieved? To make the question clear, let there be two parties, say P_1 and P_2, with their respective private inputs, x_1 and x_2 and they want to calculate the sum of their inputs, $y = x_1 + x_2$, using MPC. Assume we have such a protocol for calculating the sum. Then the player P_1, upon getting the output y, can deduce the value of x_2 from it. Similar case for P_2 also. Thus if the input and function output of a party P leaks information about the inputs of other party, then we cannot prevent such information leakage via MPC. After all the goal of MPC is to let the parties join hand to compute and know the function output. So the guarantee that MPC provides is that *'Nothing beyond the function of y is revealed'*. The privacy goal above thus needs to be re-stated as "Nothing beyond the function of y is revealed".

1.2 Trusted Third Party (TTP)

The problem of MPC can be solved easily if we have a third party whom the participating parties can trust. The parties can send their inputs to this party and he can return the output after performing the computation.

Secure Multi-party computation using TTP

1. Party P_i sends it's private input x_i to TTP.
2. TTP computes $y = f(x_1, x_2, ..., x_n)$ and sends to every party.

Do you see a problem with the solution of MPC using TTP. First, such a solution results in a single point of failure. That means if TTP fails or is corrupted by a bad entity, the entire protocol fails and the private inputs of all the parties are leaked. Second, the reason we started taking in interest is lack of trust. Had the participating parties trust each other, solving MPC is easy, namely the parties can exchange their inputs with each other and then locally they compute the function of interest on the inputs. The lack of trust leads us to do MPC. So if there is no trust, how suddenly we will get a TTP who can be cent percent trusted. So solving MPC via TTP is not a satisfactory solution. Looks like we are stuck? How can we do computation on data that are located with parties who are far away, yet without leaking the data? Is it a feasible task at all? By no means, the answer is yes and with right kind of tools, we can do computation on private data while keeping the data private. In what follows, we start with one such tool known as secret sharing. We will see how using secret sharing, we can compute addition securely. Computing addition securely has application in e-voting. Later we will introduce another tool called Oblivious Transfer (OT) and use it to build MPC for bit multiplication that has application in match-making.

2 Primitive I: Secret Sharing Schemes and it's application in Secure Addition

In this section, we will introduce the concept of secret sharing. *Secret sharing refers to methods for distributing a* secret *amongst a group of participants, each of whom is allocated*

a share of the secret. The secret can be reconstructed only when a sufficient number of shares are combined together; individual shares are of no use on their own.

Definition 2[Secret Sharing (Informal)] There are n parties,say $P_1, \cdots, P_n$ and a party designated as Dealer who possesses a secret S. Dealer wants to share S among the parties in such a way that *individual parties doesn't possesses any information on secret S and all the parties together possesses full information on secret S.* . A secret sharing scheme is a a two phase process.

1. **Sharing Phase:** Dealer splits his secret S into n parts, say $S_1, \cdots, S_n$ and share it among n parties such that party P_i receives his share S_i.
2. **Reconstruction Phase:** Parties communicate their shares among themselves to reconstruct the secret S from the shares.

$\diamondsuit$

2.1 An instantiation of Secret Sharing

The secret sharing scheme is defined on a finite field. We start with the proof that $\mathbb{F}_p$ is a field and we will use this field for building our scheme. Consider the set $\mathbb{Z}_p = \{0, 1, \cdots, p-1\}$ where p is prime.

Theorem 1 $\mathbb{F}_p$ *= ($\mathbb{Z}_p$, $+ \bmod p$, $\cdot \bmod p$) is a field.*

Proof : To prove the above theorem, we need to show that $\mathbb{Z}_p$ satisfies the axioms for field with respect to addition and multiplication operations.

Addition:

- Closure under addition is inherited from the closure of the integers under addition
- Commutativity of addition is also inherited from the integers
- Associativity of addition is inherited from the integers
- $\mathbb{Z}_p$ contains an additive identity, 0 which acts in the same way it does in the integers
- Each element has an additive inverse: for each $x \in \mathbb{Z}_p$, take $-x(\bmod p)$. Then by the properties of modular arithmetic : $x + (-x) \equiv 0 \bmod p$

Multiplication:

- We get closure under multiplication from the integers since for $x, y \in \mathbb{Z}_p, xy \bmod p$ is in $\mathbb{Z}_p$.
- Multiplication is commutative since its commutative in the integers.
- Multiplication is associative since it is in the integers.

- We have a multiplicative identity, 1 which acts the same as it does in the integers.
- We need to show that multiplicative inverses exists for each element. Let $x \in \mathbb{Z}_p$. Note that we cannot just say that the inverse is $1/x$ since this does not look like an element of $\mathbb{Z}_p$. Since p is prime, it is relatively prime to every number, including x. Therefore by the hint we know that $\exists\ a, b \in \mathbb{Z}$ such that $ax + bp = 1$. Modding out by p gives us : $ax \equiv 1 \bmod p$. Therefore a is the multiplicative inverse of x.

The distributive law is inherited from the integers since modular arithmetic behaves well. So this is a field.

A simple secret sharing protocol (secret $s \in \mathbb{F}_p$)

1. **Sharing Phase:** Dealer chooses random shares, $s_1, \cdots, s_n \in \mathbb{F}_p$ such that $s_1 + s_2 + \cdots + s_n = s$. Let $\mathcal{S} = \{s_1, \cdots, s_n\}$ denotes the set of shares chosen. Dealer sends the share $S_i = \mathcal{S} \setminus s_i$ to party P_i.
2. **Reconstruction Phase:** Here parties exchange their shares and obtains the set S by taking union of all S_i's. $\mathcal{S} = \bigcup_{i=1}^{n} S_i$ and then computing $s_1 + s_2 + \cdots + s_n$ to retrieve the secret s.

In this protocol, after the sharing phase each party P_i possesses all shares except s_i. Since the shares are chosen randomly, this gives the party P_i no information about secret s. Thus the probability of guessing s before secret sharing is same as the probability of guessing s after secret sharing (irrespective of the computing power of the parties). Also together all the parties know the secret s as described in the protocol. In fact, any two parties together knows full information of secret s here. That is, parties P_i and P_j can exchange the values s_j and s_i with each other and thus both will get the set $\mathcal{S}$.

2.2 Secure addition using Secret Sharing

The problem of *Secure Addition* can be stated as follows: *There are n parties, say $P_1, \cdots, P_n$, where each party P_i is possessing his secret, x_i. The goal is to compute the sum of shares, $y = \sum_{i=1}^{n} x_i$, without revealing anything beyond y.*

Secure Addition Protocol

1. Every party P_i secret shares his share x_i (according to protocol in section 2.1). That is it finds $x_{i1}, \ldots, x_{in}$ randomly from $\mathbb{F}_p$ such that $x_i = \sum_{j=1}^{n} x_{ij}$. To party P_j, it sends all the shares $x_{i1}, \ldots, x_{in}$ barring x_{ij}.
2. Every party P_i computes $n-1$ values $s_j,\ j = 1, \cdots, n\ \&\ j \neq i$ where $s_j = \sum_{k=1}^{n} x_{kj}$.
3. Ever party P_i sends its $n-1$ s_j to other parties and computes the sum $y = \sum_{j=1}^{n} s_j$ on receiving s_i from other parties.

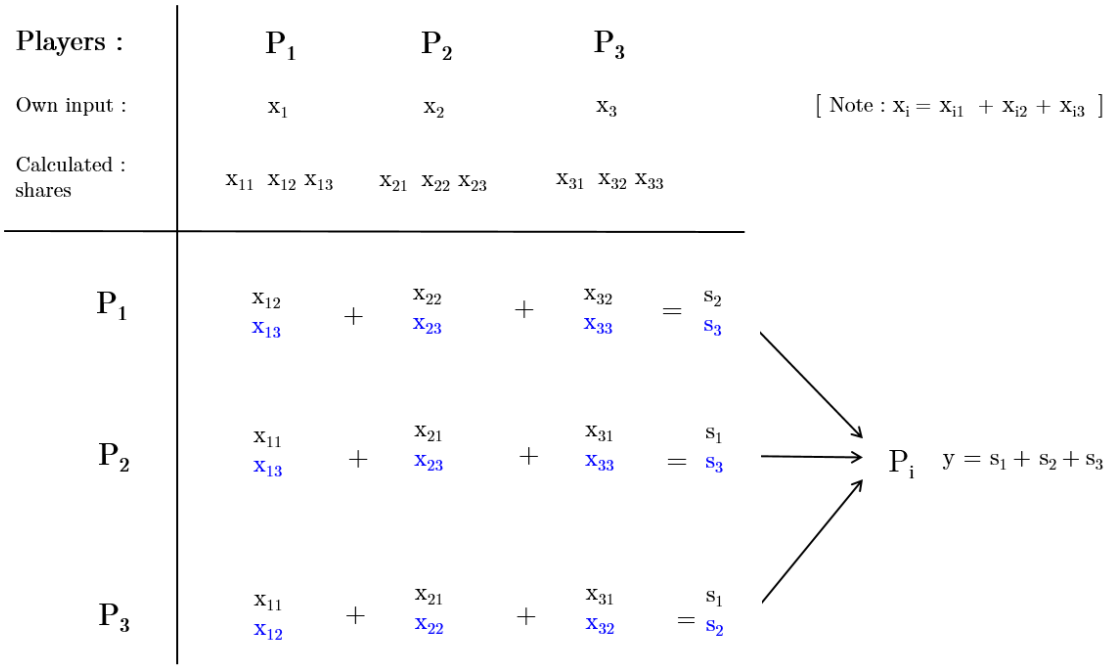

Figure 1: Secure addition for $n = 3$ parties

Figure 1 shows the secure addition protocol for 3 parties. The security of the protocol lies mainly in the underlying secret sharing scheme. After the sharing phase, Party P_i possesses all values except s_i. Since the values s_i's are random due to the randomness of underlying shares, x_{ij}'s, this gives the party P_i no information about output y until they exchange the s_i. On exchanging s_i's the parties learn only y and nothing beyond.

3 Primitive II: Oblivious Transfer and Secure bit Multiplication (Match-making)

Let's now try to compute bit multiplication security. The problem of *Secure Multiplication* which is stated as follows : *There are* 2 *parties, say* P_1 *and* P_2*, where each party* P_i *possesses his secret bit,* x_i*. The goal is to compute the product ,* $y = \prod_{i=1}^{n} x_i$*, without revealing* x_i*'s.* Note that if P_1's bit is 1, the the product will always leaks information about the input bit of P_2. But on the other hand, if the input bit of P_1 is 0, then the function output does not leak any information about the input of P_2. Our protocol must ensure to save privacy of P_2's bit when P_1's bit is 0. This problem has application in match-making.

This problem looks very similar to the Secure Addition protocol. Thus our first approach will be one similar to what we have seen in section 2.2. Consider the case of two parties, say P_1 and P_2, with their private inputs x_1 and x_2 respectively. As you can see in Figure

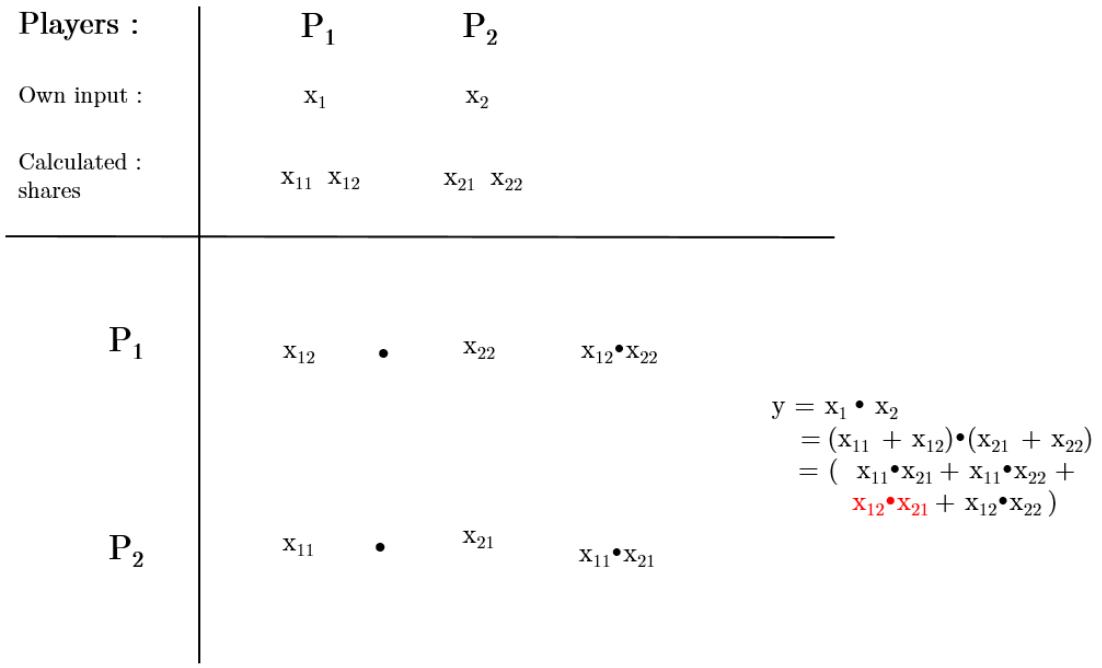

Figure 2: Secure Multiplication for $n = 2$ parties (Approach)

2, the output y can be written in terms of the secret shares as

$$y = x_{11} \cdot x_{21} + x_{11} \cdot x_{22} + x_{12} \cdot x_{21} + x_{12} \cdot x_{22}$$

In order to calculate $x_{11} \cdot x_{21}$ and $x_{12} \cdot x_{21}$, the parties need to exchange their shares. To make it more clear, take the case of $x_{11} \cdot x_{21}$. Here x_{11} is possessed by P_1 and x_{21} by P_2. If they are exchanging their shares, then P_1 will get x_{21} from which he can calculate the value of x_2. Similarly P_2 can calculate the value x_1. Now we are in a problem and we can't simply rely on the Secret Sharing Scheme. We need another primitive which we will see in next section.

3.1 1 out of 2 Oblivious Transfer (OT)

In a 1 out of 2 oblivious transfer protocol, Sender has two messages m_0 and m_1, and the Receiver has a bit b, and the Receiver wishes to receive m_b, without the Sender learning b, while the Sender wants to ensure that the Receiver receives no information on m_{1-b}.

Figure 3 shows the black-box representation of a 1 out of 2 OT. Sender gives his messages m_0 and m_1 to the box, while Receiver gives the choice bit b. The box outputs m_b to Receiver and nothing to Sender. Here the output, say y, can be rewritten as

$$y = (1 - b) \cdot m_0 + b \cdot m_1$$

We can easily see that when $b = 0$, output $y = m_0$ and when $b = 1$, $y = m_1$.

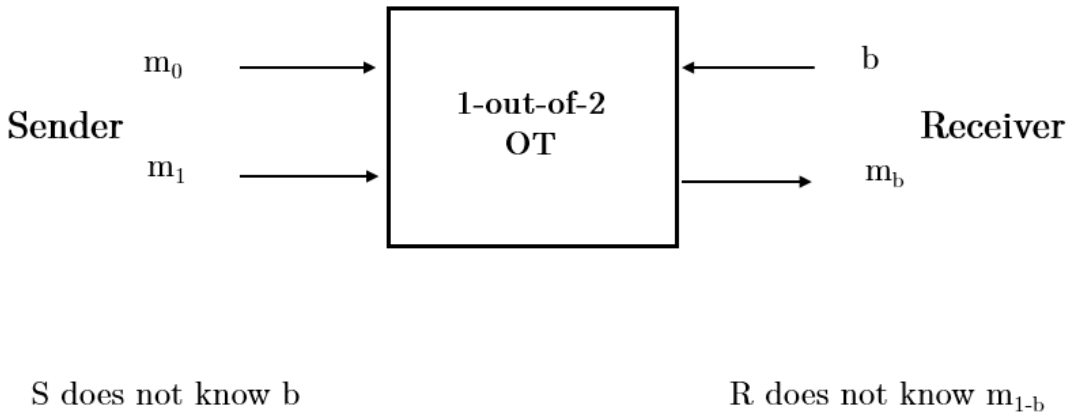

Figure 3: 1 out of 2 Oblivious Transfer

3.2 Secure Bit Multiplication using OT

Now that we have OT in our hands, the problem of Secure Multiplication [ref. Section 3] becomes very simple for the case of bits. The protocol is as follows:

1. P_1 acts as Sender and sets $m_0 = 0$ and $m_1 = x_1$.
2. P_2 acts as Receiver and sets the bit $b = x_2$.
3. Parties performs the OT protocol.
4. P_2 sends the output y to P_1.

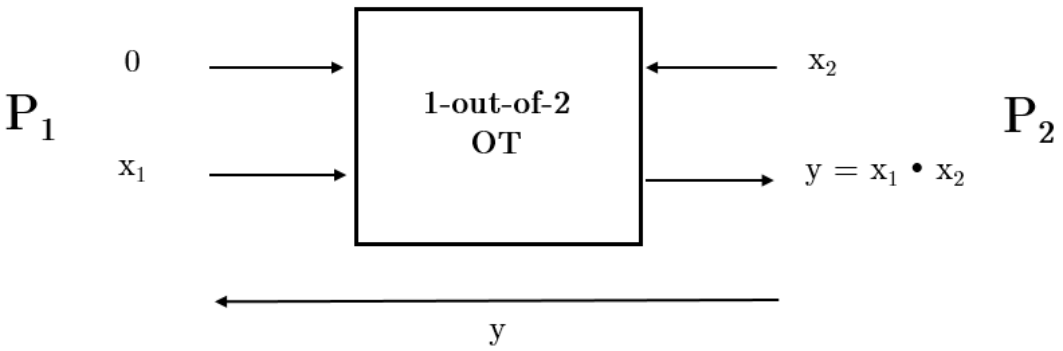

Figure 4: Secure Bit Multiplication using OT

Regarding privacy of the above protocol, when $x_2 = 0$, the output is 0 and hence P_2 learns nothing about x_1. When $x_2 = 1$, the output is x_1, which is same as $x_1 \cdot x_2$ in this case, which is the allowed output. We have seen two magic tools: secret sharing and oblivious transfer to accomplish the task of MPC of two simple functions. We will later see that these tools will allow us to do MPC for any arbitrary polynomially computable functions. In the next couple of lectures, we will see why MPC is omnipresent, all-pervasive and considered to be the holy-grail of cryptography. In other words, we will see Vishwaroop of MPC...

References

[1] Ronald Cramer, Ivan Bjerre Damgrd, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing - An Information Theoretic Approach*. Cambridge University Press, 2015.

[2] Arpita Patra. *http://drona.csa.iisc.ernet.in/ arpita/SecureComputation15.html* . Course Materials.