An Introduction to Secure Multi-Party Computation

Giuseppe D'Alconzo
giuseppe.dalconzo@telsy.com

De Cifris Augustae Taurinorum, Politecnico di Torino

15 luglio 2019

Overview

Introduction: What is Secure Multiparty Computation

Introduction: What is Secure Multiparty Computation

- Computation: a known function is evaluated

Telsy

Introduction: What is Secure Multiparty Computation

- Computation: a known function is evaluated
- Multiparty: a set of parties want to evaluate this function using their (private) inputs

Telsy

Introduction: What is Secure Multiparty Computation

- Computation: a known function is evaluated
- Multiparty: a set of parties want to evaluate this function using their (private) inputs
- Secure: each party's input remains secret

Telsy

Introduction: What is Secure Multiparty Computation

- Computation: a known function is evaluated
- Multiparty: a set of parties want to evaluate this function using their (private) inputs
- Secure: each party's input remains secret

MPC allows a set of parties to joinly compute a function on their secret inputs.

Introduction: An Example (Yao 1982)

Two millionaires want to know who has more money without revealing their assets.

- Parties: two millionaires Alice and Bob
- Function: $X_A > X_B$?
- Inputs: assets X_A and X_B

Trivial solution: a trusted third party.
It gets X_A and X_B and announces who is the richer.

Introduction: Trusted Third Party

Unfortunately a trusted third party doesn't always exists. We would like a solution with the same security guarantees, but without using any trusted party.

Telsy

Introduction: Trusted Third Party

Unfortunately a trusted third party doesn't always exists. We would like a solution with the same security guarantees, but without using any trusted party.

To compute on private data there are two main solutions:

- Homomorphic encryption
- Multiparty Computation

Introduction: Formalization of the Model

An MPC protocol is a set of instructions for each party.
These instructions can be local computations and exchanges of data.
The output can be revealed to all or some of the players.

Telsy

Introduction: Formalization of the Model

An MPC protocol is a set of instructions for each party.
These instructions can be local computations and exchanges of data.
The output can be revealed to all or some of the players.

Formally:

- Parties: $P_1, \ldots, P_n$
- Inputs: $x_1, \ldots, x_n$
- Public function: f
- Output: $y = f(x_1, \ldots, x_n)$

Introduction: Security Properties

- Input privacy: the execution of the protocol should not give any information about the private data of the parties, except for what is revealed by the output of the function.

Introduction: Security Properties

- Input privacy: the execution of the protocol should not give any information about the private data of the parties, except for what is revealed by the output of the function.

- Correctness: depending on the MPC protocol, even if a subset of parties are colluding sharing information or deviating by the protocol, they should not be able to force honest parties to output an incorrect result.

Telsy

Introduction: Types of Security

MPC protocols can differ in the type of security guaranteed. There are 3 main types:

- Passive security: security properties are guaranteed with semi-honest (or "honest but curious“) adversaries, which follow the rules of the protocol but wanting to extract more information from their observed data.

Introduction: Types of Security

MPC protocols can differ in the type of security guaranteed. There are 3 main types:

- Passive security: security properties are guaranteed with semi-honest (or "honest but curious") adversaries, which follow the rules of the protocol but wanting to extract more information from their observed data.
- Active security: even if adversaries deviate from the protocol and try to obtain information about honest parties' inputs, the security properties are guaranteed. This type of adversaries are called malicious.

Introduction: Types of Security

MPC protocols can differ in the type of security guaranteed. There are 3 main types:

- Passive security: security properties are guaranteed with semi-honest (or ”honest but curious“) adversaries, which follow the rules of the protocol but wanting to extract more information from their observed data.
- Active security: even if adversaries deviate from the protocol and try to obtain information about honest parties' inputs, the security properties are guaranteed. This type of adversaries are called malicious.
- Covert security: between passive and active security. Quite informal definition: “secure enough”.

Introduction: Boolean MPC and Arithmetic MPC

There are two main classes of MPC protocols:

- Boolean MPC: the function is represented by a boolean circuit. There are two parties, the circuit constructor A and the circuit evaluator B. A encrypts or garbles the circuit and sends it to B that evaluates it with his input and learns the output. This protocol is called *Garbled Circuit*.

Telsy

Introduction: Boolean MPC and Arithmetic MPC

There are two main classes of MPC protocols:

- Boolean MPC: the function is represented by a boolean circuit. There are two parties, the circuit constructor A and the circuit evaluator B. A encrypts or garbles the circuit and sends it to B that evaluates it with his input and learns the output. This protocol is called *Garbled Circuit*.

- Arithmetic MPC: inputs are shared using some secret sharing scheme, then the function is computed on this sharings, using addition, multiplication and other arithmetic operations.

Telsy

Telsy

Boolean MPC: Garbled Circuits

Every function can be represented as a boolean circuit with AND, OR and NOT gates. Gates are connected by three types of wires: input wires, output wires and intermediate wires.

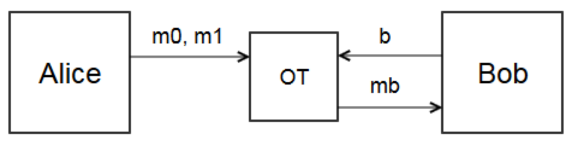
Boolean MPC: Garbled Circuits

Every function can be represented as a boolean circuit with AND, OR and NOT gates. Gates are connected by three types of wires: input wires, output wires and intermediate wires.

Ingredients:

- Double key simmetric encryption: given a plaintext m and two keys k_1, k_2, we denote by $E_{k_1,k_2}(m)$ the encryption of m with keys k_1, k_2. For example we can use $E_{k_1,k_2}(m) = AES_{k_1}(AES_{k_2}(m))$.
  In order to check the validity of a plaintext we can add some redundancy.
- Oblivious Transfer (OT)

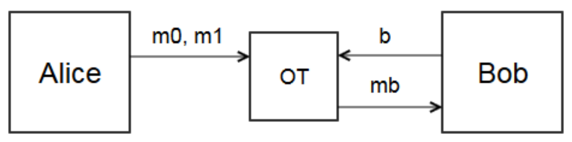
Boolean MPC: Oblivious Transfer



An oblivious transfer protocol is an interactive protocol between two parties: a Sender and a Receiver, each providing some inputs.

- The Sender inputs a couple of bits (or numbers) m_0, m_1

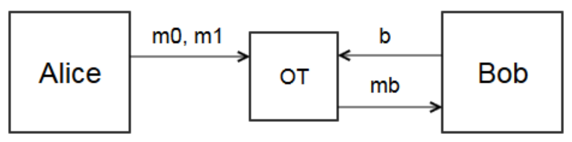
Boolean MPC: Oblivious Transfer



An oblivious transfer protocol is an interactive protocol between two parties: a Sender and a Receiver, each providing some inputs.

- The Sender inputs a couple of bits (or numbers) m_0, m_1
- The input of the Receiver is a single bit b.

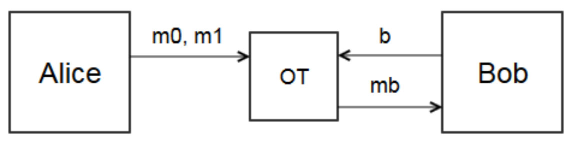
Boolean MPC: Oblivious Transfer



An oblivious transfer protocol is an interactive protocol between two parties: a Sender and a Receiver, each providing some inputs.

- The Sender inputs a couple of bits (or numbers) m_0, m_1
- The input of the Receiver is a single bit b.
- At the end of the protocol, the Receiver gets the value m_b

Boolean MPC: Oblivious Transfer



An oblivious transfer protocol is an interactive protocol between two parties: a Sender and a Receiver, each providing some inputs.

- The Sender inputs a couple of bits (or numbers) m_0, m_1
- The input of the Receiver is a single bit b.
- At the end of the protocol, the Receiver gets the value m_b
- The Sender knows nothing about b and the Receiver knows nothing about the other value m_{1-b}

Boolean MPC: Circuit Construction

Alice constructs the "plain" boolean circuit for the function f. Then Alice garbles it:

Boolean MPC: Circuit Construction

Alice constructs the "plain" boolean circuit for the function f. Then Alice garbles it:

- For each wire W_i she randomly chooses two secret values: w_i^0 for the value 0 and w_i^1 for 1. These are called *garbled values* for 0 and 1.

Boolean MPC: Circuit Construction

Alice constructs the "plain" boolean circuit for the function f. Then Alice garbles it:

- For each wire W_i she randomly chooses two secret values: w_i^0 for the value 0 and w_i^1 for 1. These are called *garbled values* for 0 and 1.
- Given a truth table for the gates G Alice construct a *garbled truth table* encrypting the garbled value of the output wire using the two garbled values of the inputs as keys.

Boolean MPC: Circuit Construction

Alice constructs the "plain" boolean circuit for the function f. Then Alice garbles it:

- For each wire W_i she randomly chooses two secret values: w_i^0 for the value 0 and w_i^1 for 1. These are called *garbled values* for 0 and 1.
- Given a truth table for the gates G Alice construct a *garbled truth table* encrypting the garbled value of the output wire using the two garbled values of the inputs as keys.
- For every output wire, Alice announces the correspondence between w_j^i and i so Bob can learn the output.

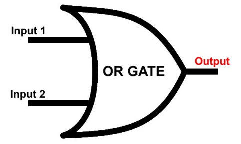
Boolean MPC: Circuit Construction

Alice constructs the “plain” boolean circuit for the function f. Then Alice garbles it:

- For each wire W_i she randomly chooses two secret values: w_i^0 for the value 0 and w_i^1 for 1. These are called *garbled values* for 0 and 1.
- Given a truth table for the gates G Alice construct a *garbled truth table* encrypting the garbled value of the output wire using the two garbled values of the inputs as keys.
- For every output wire, Alice announces the correspondence between w_j^i and i so Bob can learn the output.
- Alice sends the garbled values of every truth table to Bob.

Boolean MPC: Example of Garbled Truth Table

OR gate



W_0	W_1	W
0	0	0
0	1	1
1	0	1
1	1	1

W_0	W_1	W	Garbled value
w_0^0	w_1^0	w^0	$\mathbb{E}_{w_0^0, w_1^0}(w^0)$
w_0^0	w_1^1	w^1	$\mathbb{E}_{w_0^0, w_1^1}(w^1)$
w_0^1	w_1^0	w^1	$\mathbb{E}_{w_0^1, w_1^0}(w^1)$
w_0^1	w_1^1	w^1	$\mathbb{E}_{w_0^1, w_1^1}(w^1)$

Boolean MPC: Evaluating the Circuit

- Using an oblivious transfer, Bob asks Alice the garbled values for its secret inputs.

Telsy

Boolean MPC: Evaluating the Circuit

- Using an oblivious transfer, Bob asks Alice the garbled values for its secret inputs.
- Starting from the input wires, for each gate Bob decrypts the values in the garbled truth table, finding the valid plaintext related to the output wire of the gate.

Telsy

Boolean MPC: Evaluating the Circuit

- Using an oblivious transfer, Bob asks Alice the garbled values for its secret inputs.
- Starting from the input wires, for each gate Bob decrypts the values in the garbled truth table, finding the valid plaintext related to the output wire of the gate.
- Bob continues with the next gate.

Boolean MPC: Evaluating the Circuit

- Using an oblivious transfer, Bob asks Alice the garbled values for its secret inputs.
- Starting from the input wires, for each gate Bob decrypts the values in the garbled truth table, finding the valid plaintext related to the output wire of the gate.
- Bob continues with the next gate.
- When Bob has a garbled value Z for an output wire, he learns the output bit looking at values (Z_0, Z_1) and checking if $Z = Z_0$ or $Z = Z_1$.

Boolean MPC: Example of Evaluation of a Gate

- Gate values: X_1, X_2, X_3, X_4.
- Bob has the two garbled values of the inputs of this gate: w_1, w_2.
- He computes $D_{w_1,w_2}(X_i)$ for each i.
- When he finds a valid plaintext, he gets the garbled value associated to the output wire of this gate.

Telsy

Arithmetic MPC

If the function to be evaluated can be easily expressed in an arithmetic form, then it is convenient to use an MPC protocol based on *secret sharing*. This kind of protocols usually works on finite algebraic structures (finite fields or rings).

Arithmetic MPC

If the function to be evaluated can be easily expressed in an arithmetic form, then it is convenient to use an MPC protocol based on *secret sharing*. This kind of protocols usually works on finite algebraic structures (finite fields or rings).

There are a lot of secret sharing schemes, the most used are:

- Additive secret sharing;
- Shamir secret sharing.

Arithmetic MPC: Additive Secret Sharing

Suppose n players $P_1, \ldots, P_n$.
If a player P wants to share its secret input x, he randomly generates n shares $x^{(j)}$ such that

$$x = \sum_{j=1}^{n} x^{(j)}$$

Then P sends $x^{(j)}$ to player P_j.
The shared value of x is denoted as

$$[[x]] = (x^{(1)}, \ldots, x^{(n)})$$

This means that every player has a little part of x but nobody knows the actual value.

Telsy

Arithmetic MPC: Example of Additive Secret Sharing

Suppose 5 players $P_1, \dots, P_5$. A dealer wants to share the secret $s = 6 \in \mathbb{F}_{11}$.

- He generates 4 random elements of $\mathbb{F}_{11}$

$$s^{(1)} = 5, s^{(2)} = 3, s^{(3)} = 8, s^{(4)} = 0$$

Arithmetic MPC: Example of Additive Secret Sharing

Suppose 5 players $P_1, \dots, P_5$. A dealer wants to share the secret $s = 6 \in \mathbb{F}_{11}$.

- He generates 4 random elements of $\mathbb{F}_{11}$

$$s^{(1)} = 5, s^{(2)} = 3, s^{(3)} = 8, s^{(4)} = 0$$

- Then he sets

$$s^{(5)} = s - \sum_{i=1}^{4} s^{(i)} = 6 - 5 = 1$$

and distributes $s^{(i)}$ to P_i.

Arithmetic MPC: Example of Additive Secret Sharing

Suppose 5 players $P_1, \dots, P_5$. A dealer wants to share the secret $s = 6 \in \mathbb{F}_{11}$.

- He generates 4 random elements of $\mathbb{F}_{11}$

$$s^{(1)} = 5, s^{(2)} = 3, s^{(3)} = 8, s^{(4)} = 0$$

- Then he sets

$$s^{(5)} = s - \sum_{i=1}^{4} s^{(i)} = 6 - 5 = 1$$

  and distributes $s^{(i)}$ to P_i.
- All players can reconstruct the secret sharing their values to get

$$s = \sum_{i=1}^{5} s^{(i)}$$

Arithmetic MPC: Shamir Secret Sharing Scheme

Suppose n players $P_1, \ldots, P_n$ and $t \leq n$. A secret x of the player P can be shared as follow:

- P secretly chooses a random polynomial f of degree $t-1$ such that $f(0) = x$.

Telsy

Arithmetic MPC: Shamir Secret Sharing Scheme

Suppose n players $P_1, \ldots, P_n$ and $t \leq n$. A secret x of the player P can be shared as follow:

- P secretly chooses a random polynomial f of degree $t-1$ such that $f(0) = x$.
- P gives to the player P_i the couple $(i, f(i))$.

Arithmetic MPC: Shamir Secret Sharing Scheme

Suppose n players $P_1, \ldots, P_n$ and $t \leq n$. A secret x of the player P can be shared as follow:

- P secretly chooses a random polynomial f of degree $t-1$ such that $f(0) = x$.
- P gives to the player P_i the couple $(i, f(i))$.
- If at least t player share their shares, using Lagrange interpolation they can reconstruct f and compute $x = f(0)$.

Arithmetic MPC: Shamir Secret Sharing Scheme

Suppose n players $P_1, \dots, P_n$ and $t \leq n$. A secret x of the player P can be shared as follow:

- P secretly chooses a random polynomial f of degree $t - 1$ such that $f(0) = x$.
- P gives to the player P_i the couple $(i, f(i))$.
- If at least t player share their shares, using Lagrange interpolation they can reconstruct f and compute $x = f(0)$.
- The number t is called threshold.

Arithmetic MPC: Shamir Secret Sharing Scheme

Suppose n players $P_1, \ldots, P_n$ and $t \leq n$. A secret x of the player P can be shared as follow:

- P secretly chooses a random polynomial f of degree $t-1$ such that $f(0) = x$.
- P gives to the player P_i the couple $(i, f(i))$.
- If at least t player share their shares, using Lagrange interpolation they can reconstruct f and compute $x = f(0)$.
- The number t is called threshold.

The shared value of x is denoted as

$$[[x]] = ((1, f(1)), \ldots, (n, f(n)))$$

Telsy

Arithmetic MPC: Lagrange Interpolation

Given a set of points $\{(x_1, y_1), \ldots, (x_r, y_r)\}$ such that $x_i \neq x_j$ for every $i \neq j$, then exists a unique polinomial f of degree $\leq r - 1$ such that $f(x_i) = y_i$ for each i.

The polynomial f can be constructed as follows:

- Define
$$\delta_i(x) = \frac{\prod_{\substack{j=1 \\ j \neq i}}^{r} (x - x_j)}{\prod_{\substack{j=1 \\ j \neq i}}^{r} (x_i - x_j)}$$
we see that for each i $\delta_i(x_i) = 1$ and $\delta_i(x_j) = 0$ if $i \neq j$.
- Then set
$$f(x) = \sum_{i=1}^{r} \delta_i(x) y_i$$

Telsy

Arithmetic MPC: Example of Shamir Secret Sharing

Suppose 5 players $P_1, \ldots, P_5$, we work on $\mathbb{F}_{11}$. A dealer want to share the secret $s = 3$ with the threshold $t = 3$.

- He chooses a random polynomial such that $f(0) = s = 3$:

$$f(x) = 5x^2 + 10x + 3$$

- He gives $(i, f(i))$ to the i-th player:

$$(1,7),(2,10),(3,1),(4,2),(5,2)$$

- If P_1, P_4, P_5 want to learn the secret, they use Lagrange interpolation and find f and then $s = 3$.

Arithmetic MPC: Arithmetic with Additive Secret Sharing

Suppose n players have $[[x]] = (x^{(i)})_{i=1}^{n}$ and $[[y]] = (y^{(i)})_{i=1}^{n}$. They want to compute the shared value of the sum: $[[z]] = [[x + y]]$

- Each player P_i sets $z^{(i)} = x^{(i)} + y^{(i)}$.
- In fact:

$$z = \sum_{i=1}^{n}(x^{(i)} + y^{(i)}) = \sum_{i=1}^{n} x^{(i)} + \sum_{i=1}^{n} y^{(i)} = x + y$$

- This is an operation without communication.

Arithmetic MPC: Arithmetic with Additive Secret Sharing

If players have $[[x]]$ and they want to compute $[[z]] = [[cx]]$ for any public c:

- Each player P_i sets $z^{(i)} = cx^{(i)}$
- In fact:

$$z = \sum_{i=1}^{n} cx^{(i)} = c \sum_{i=1}^{n} x^{(i)} = cx$$

- This is another communication-free operation.

Arithmetic MPC: Arithmetic with Additive Secret Sharing

To perform a multiplication there are some different methods. For example the SPDZ protocol uses the Beaver's trick with some precomputed "multiplication triples":

$$([[a]], [[b]], [[c]]) \quad \text{such that} \quad ab = c$$

Arithmetic MPC: Arithmetic with Additive Secret Sharing

To perform a multiplication there are some different methods. For example the SPDZ protocol uses the Beaver's trick with some precomputed "multiplication triples":

$$([[a]], [[b]], [[c]]) \quad \text{such that} \quad ab = c$$

How those triples are generated depends on the MPC protocol used, SPDZ bases his triple generation on homomorphic encryption.

Arithmetic MPC: Arithmetic with Additive Secret Sharing

Suppose parties have $[[x]]$ and $[[y]]$.
To compute $[[z]] = [[xy]]$:

- Players compute $[[\rho]] = [[x]] - [[a]]$ and reveal ρ

Arithmetic MPC: Arithmetic with Additive Secret Sharing

Suppose parties have $[[x]]$ and $[[y]]$.
To compute $[[z]] = [[xy]]$:

- Players compute $[[\rho]] = [[x]] - [[a]]$ and reveal ρ
- Players compute $[[\sigma]] = [[y]] - [[b]]$ and reveal σ

Telsy

Arithmetic MPC: Arithmetic with Additive Secret Sharing

Suppose parties have $[[x]]$ and $[[y]]$.
To compute $[[z]] = [[xy]]$:

- Players compute $[[\rho]] = [[x]] - [[a]]$ and reveal ρ
- Players compute $[[\sigma]] = [[y]] - [[b]]$ and reveal σ
- The output is $[[z]] = [[c]] + [[\rho b]] + [[\sigma a]] + \rho\sigma$

Arithmetic MPC: Arithmetic with Additive Secret Sharing

Summarizing:

- additions and scalar multiplications are "free" operations in terms of communication
- performing a multiplication costs 1 round of communication
- The complexity of a function to be evaluated in MPC is linked to the number of multiplications

Arithmetic MPC: A Simple Example (1)

Two parties, P_1 and P_2 want to compute $f(x_1, x_2) = x_1 x_2 + x_1$ in $\mathbb{F}_7$. Suppose $x_1 = 2$ and $x_2 = 5$.

Telsy

Arithmetic MPC: A Simple Example (1)

Two parties, P_1 and P_2 want to compute $f(x_1, x_2) = x_1 x_2 + x_1$ in $\mathbb{F}_7$. Suppose $x_1 = 2$ and $x_2 = 5$.

- They share their inputs. P_1 generates a random $x_1^{(1)} = 3$ and sets $x_1^{(2)} = 2 - 3 = 6$. P_1 sends $x_1^{(2)}$ to P_2, then we have:

$$[[x_1]] = [[2]] = (3, 6)$$

Arithmetic MPC: A Simple Example (1)

Two parties, P_1 and P_2 want to compute $f(x_1, x_2) = x_1 x_2 + x_1$ in $\mathbb{F}_7$. Suppose $x_1 = 2$ and $x_2 = 5$.

- They share their inputs. P_1 generates a random $x_1^{(1)} = 3$ and sets $x_1^{(2)} = 2 - 3 = 6$. P_1 sends $x_1^{(2)}$ to P_2, then we have:

$$[[x_1]] = [[2]] = (3, 6)$$

- P_2 does the same, he generates $x_2^{(1)} = 1$ and sets $x_2^{(2)} = 5 - 1 = 4$ and sends $x_2^{(1)}$ to P_1. Then

$$[[x_2]] = [[5]] = (1, 4)$$

Telsy

Arithmetic MPC: A Simple Example (2)

Now they want to compute $[[x_1 x_2]]$. They pick a precomputed multiplication triple:

$$([[a]], [[b]], [[c]]) = ([[2]], [[6]], [[5]])$$

such that:

$$[[2]] = (1, 1), \ [[6]] = (4, 2), \ [[5]] = (0, 5)$$

Telsy

Arithmetic MPC: A Simple Example (3)

Multiplication subprotocol:

- P_1 computes

$$\rho^{(1)} = x_1^{(1)} - a^{(1)} = 2$$
$$\sigma^{(1)} = x_2^{(1)} - b^{(1)} = 4$$

Arithmetic MPC: A Simple Example (3)

Multiplication subprotocol:

- P_1 computes

$$\rho^{(1)} = x_1^{(1)} - a^{(1)} = 2$$
$$\sigma^{(1)} = x_2^{(1)} - b^{(1)} = 4$$

- P_2 computes

$$\rho^{(2)} = x_1^{(2)} - a^{(2)} = 5$$
$$\sigma^{(2)} = x_2^{(2)} - b^{(2)} = 2$$

Telsy

Arithmetic MPC: A Simple Example (3)

Multiplication subprotocol:

- P_1 computes

$$\rho^{(1)} = x_1^{(1)} - a^{(1)} = 2$$
$$\sigma^{(1)} = x_2^{(1)} - b^{(1)} = 4$$

- P_2 computes

$$\rho^{(2)} = x_1^{(2)} - a^{(2)} = 5$$
$$\sigma^{(2)} = x_2^{(2)} - b^{(2)} = 2$$

- They reveal the shares of $\rho = 0$ and $\sigma = 6$.

Arithmetic MPC: A Simple Example (3)

Multiplication subprotocol:

- P_1 computes

$$\rho^{(1)} = x_1^{(1)} - a^{(1)} = 2$$
$$\sigma^{(1)} = x_2^{(1)} - b^{(1)} = 4$$

- P_2 computes

$$\rho^{(2)} = x_1^{(2)} - a^{(2)} = 5$$
$$\sigma^{(2)} = x_2^{(2)} - b^{(2)} = 2$$

- They reveal the shares of $\rho = 0$ and $\sigma = 6$.
- P_1 sets $z^{(1)} = c^{(1)} + \rho b^{(1)} + \sigma a^{(1)} + \rho\sigma = 6$

Arithmetic MPC: A Simple Example (3)

Multiplication subprotocol:

- P_1 computes

$$\rho^{(1)} = x_1^{(1)} - a^{(1)} = 2$$
$$\sigma^{(1)} = x_2^{(1)} - b^{(1)} = 4$$

- P_2 computes

$$\rho^{(2)} = x_1^{(2)} - a^{(2)} = 5$$
$$\sigma^{(2)} = x_2^{(2)} - b^{(2)} = 2$$

- They reveal the shares of $\rho = 0$ and $\sigma = 6$.
- P_1 sets $z^{(1)} = c^{(1)} + \rho b^{(1)} + \sigma a^{(1)} + \rho\sigma = 6$
- P_2 sets $z^{(2)} = c^{(2)} + \rho b2^{(2)} + \sigma a^{(2)} + \rho\sigma = 4$

Arithmetic MPC: A Simple Example (4)

Now we have $[[z]] = (6, 4) = [[3]] = [[2 \cdot 5]]$. To obtain the output of $f(2, 5)$ we need to compute $[[z + x_1]]$.

- P_1 sets $w^{(1)} = z^{(1)} + x_1^{(1)} = 2$

Telsy

Arithmetic MPC: A Simple Example (4)

Now we have $[[z]] = (6, 4) = [[3]] = [[2 \cdot 5]]$. To obtain the output of $f(2, 5)$ we need to compute $[[z + x_1]]$.

- P_1 sets $w^{(1)} = z^{(1)} + x_1^{(1)} = 2$
- P_2 sets $w^{(2)} = z^{(2)} + x_1^{(2)} = 3$

Telsy

Arithmetic MPC: A Simple Example (4)

Now we have $[[z]] = (6, 4) = [[3]] = [[2 \cdot 5]]$. To obtain the output of $f(2, 5)$ we need to compute $[[z + x_1]]$.

- P_1 sets $w^{(1)} = z^{(1)} + x_1^{(1)} = 2$
- P_2 sets $w^{(2)} = z^{(2)} + x_1^{(2)} = 3$
- Now they excange their shares and learn the output $w = 2 + 3 = 5$, in fact $f(2, 5) = 5$.

Telsy

Arithmetic MPC: Offline Phase vs Online Phase

Some protocols split computation in two parts:

- A preprocessing phase that depends on the function and is independent on the inputs. It is called "offline phase".
- An evaluation phase: players uses their inputs and compute the function, this is called "online phase".

Arithmetic MPC: Offline Phase vs Online Phase

Some protocols split computation in two parts:

- A preprocessing phase that depends on the function and is independent on the inputs. It is called “offline phase”.
- An evaluation phase: players uses their inputs and compute the function, this is called “online phase”.

For example in the SPDZ protocol, the offline phase is represented by the triples generation, while the online phase by the actual computation of the function f.

Telsy

Active Security: How to Prevent Active Attacks?

How to deal with malicious adversaries that can deviate from the protocol? When the protocol says "send x " they could send y or some crafted values.

There are some solutions, we see how the SPDZ protocol solves this problem.

Active Security: MAC Keys

Each player P_i generates a MAC key $\Delta^{(i)}$. We define

$$\Delta = \sum_{i=1}^{n} \Delta^{(i)}$$

Active Security: MAC Keys

Each player P_i generates a MAC key $\Delta^{(i)}$. We define

$$\Delta = \sum_{i=1}^{n} \Delta^{(i)}$$

Now shares of the value $x \in \mathbb{F}$ are of the form

$$[[x]] = (\underbrace{x^{(1)}, \ldots, x^{(n)}}_{\text{shares}}, \underbrace{m(x)^{(1)}, \ldots, m(x)^{(n)}}_{\text{MAC shares}}, \underbrace{\Delta^{(1)}, \ldots, \Delta^{(n)}}_{\text{MAC keys}})$$

Such that:

$$x = \sum_{i=1}^{n} x^{(i)}, \qquad x \cdot \Delta = \sum_{i=1}^{n} m(x)^{(i)}$$

Telsy

Active Security: MAC Keys

If a malicious player sends the wrong values for $x^{(i)}$, he can't modify his MAC shares $m(x)^{(i)}$ to be consistent with the new value since he has not other MAC shares and Δ.

Telsy

Active Security: MAC Keys

If a malicious player sends the wrong values for $x^{(i)}$, he can't modify his MAC shares $m(x)^{(i)}$ to be consistent with the new value since he has not other MAC shares and Δ.

When the function is evaluated and players hold the shared output, before revealing it to all parties, there is a general MAC check on all the values opened during the protocol.
If this check passes, then the output is revealed and accepted.

Telsy

Telsy

Real-world Applications

Since 2008 there were a lot of real-world applications of MPC, for example:

- Danish sugar beet auction
- Benchmarking
- Satellite collisions
- Machine learning on private data

Telsy

Libraries

There are a lot of libraries that implement some MPC functionalities. Some examples:

- SCALE-MAMBA
- MP-SPDZ
- libSCAPI
- Fresco
- ...and many other

References

- Damgård, Ivan, et al. *Multiparty computation from somewhat homomorphic encryption.* Annual Cryptology Conference. Springer, Berlin, Heidelberg, 2012
- Bogetoft, Peter, et al. *Secure multiparty computation goes live.* International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2009
- Kamm, Liina, and Jan Willemson. *Secure floating point arithmetic and private satellite collision analysis.* International Journal of Information Security 14.6 (2015): 531-548

Telsy

Thank you.
Questions?

Telsy