SEC595 Encrypted Computing

Lecture 16: Secure Multiparty Computation

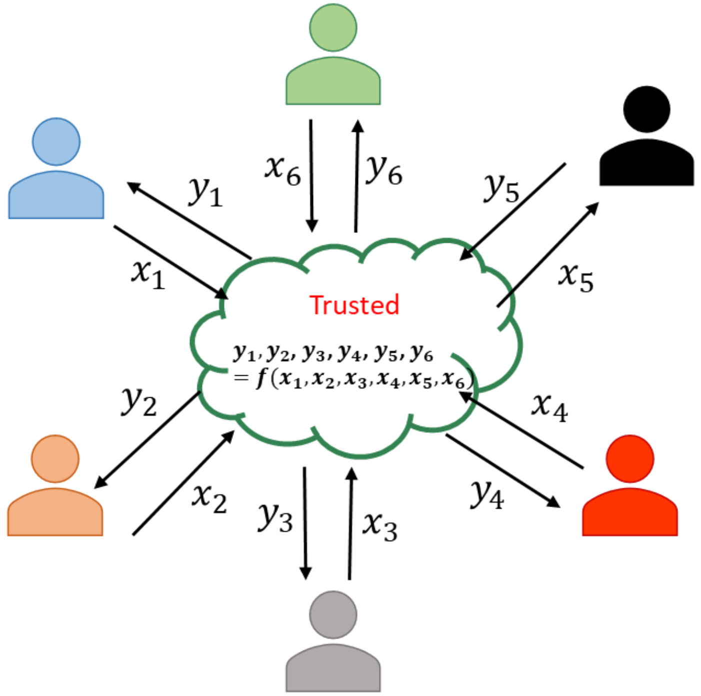
Instructor: Muhamad Felemban

Computing on Encrypted Data

- Basic idea
  - Client encrypts his data x and sends encryption $E(x)$ to the server
  - The server performs some computation (evaluate function f) and returns the encrypted result to the client
- The client decrypts the result to find out the answer, but the server learns nothing about the data
  - Homomorphic encryption

Secure Multiparty Computation (SMC)

- General framework for describing computation between parties who do not trust each other
- A set of parties with private inputs wish to compute some joint function of their inputs
- Parties wish to preserve some security properties. e.g., **privacy** and **correctness**
- Security must be preserved in the face of adversarial behavior by
  - One (or some) of the participants, or
  - An external party

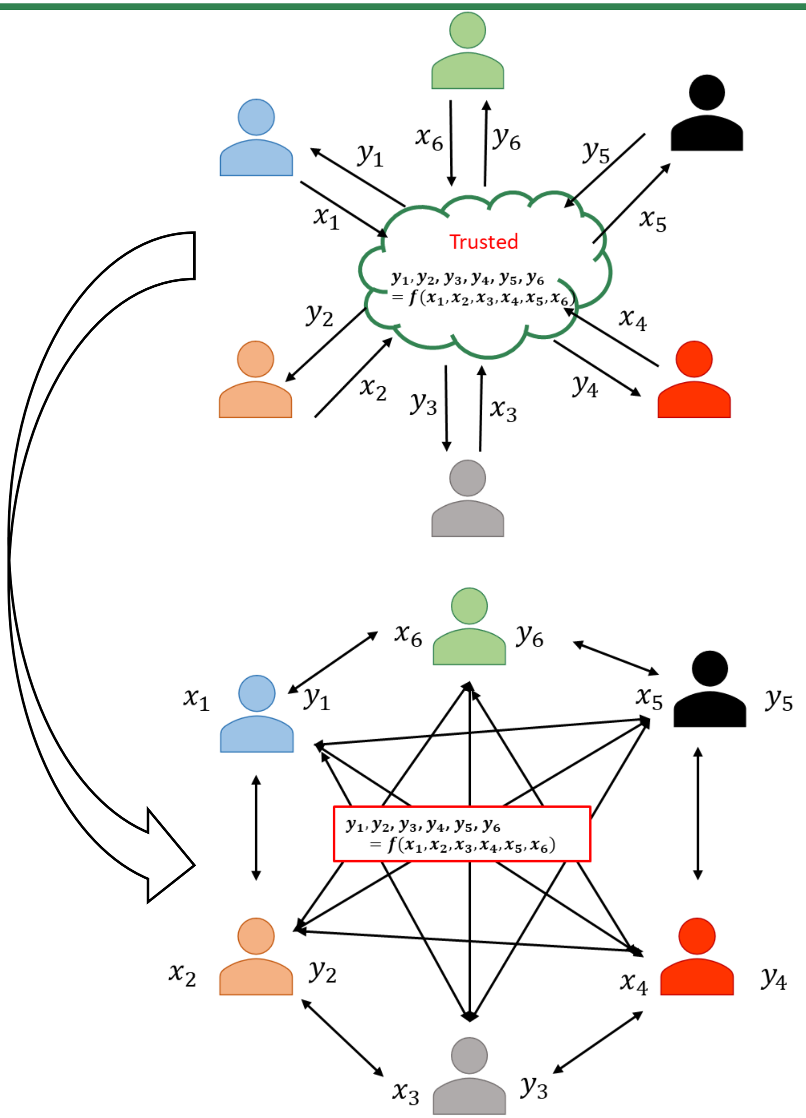


SMC Examples

- Elections
  - N parties, each one has a "Yes" or "No" vote
  - **Goal:** determine whether the majority voted "Yes", but no voter should learn how other people voted
- Auctions
  - Each bidder makes an offer
    - Offer should be committing! (can't change it later)
  - **Goal:** determine whose offer won without revealing losing offers
- Business intelligence
  - Two companies want to compare their datasets without sharing them
  - **Goal:** determine common business interest without revealing too much information
- Database privacy
  - Database holder wants to share analytics of data without sharing the date
    - Similar to differential privacy
  - Analysts doesn't want to reveal his query to data base holder!!!
  - **Goal:** Evaluate a query on the database without revealing the query to the database owner

A Couple of Observations

- In all cases, we are dealing with distributed multi-party protocols
  - A protocol describes how parties are supposed to exchange messages on the network
- All these tasks can be easily computed by a trusted third party
- The goal of secure multi-party computation is to achieve the same result without involving a trusted third party



Security Requirements

- **Privacy:** only the relative output is revealed
  - An adversary shouldn't learn the bids of all parties
- **Correctness:** the function is computed correctly
  - An adversary shouldn't win with a lower bid than the highest
- **Independence of inputs:** parties cannot choose inputs based on others' inputs
  - The adversary shouldn't be ensured that he always gives the highest bid
- **Fairness:** if one party receives output, all receive output
  - An adversary shouldn't be able to abort the execution if its bid is not the highest

How to Define Security?

- Must be mathematically rigorous

- Must capture all realistic attacks that a malicious participant may try to stage

- Should be "abstract"
  - Based on the desired "functionality" of the protocol, not a specific protocol
  - **Goal:** define security for an entire class of protocols

Heuristic Approach to Security

Approach 1

1. Build a protocol
2. Try to break the protocol
3. Fix the break
4. Return to (2)

- Problems
    1. Real adversaries won't tell you that they have broken the protocol
    2. You can never be really sure that the protocol is secure

Approach 2

1. Design a protocol
2. Provide a list of attacks that (provably) cannot be carried out on the protocol
3. Reason that the list is complete

- Problem: often, the list of attacks is not complete
    - Zero-day attacks!!

A Rigorous Approach for Security

- Provide an exact problem definition
  - Adversarial power
  - Network model
  - Meaning of security

- Prove that the protocol is secure
  - Often by reduction to an assumed hard problem, like factoring large composites

SMC Functionality

- SMC problem model
  - K mutually distrustful parties want to jointly carry out some task
  - The input to the function is K (private) input, the output is K outputs
  - F can be either a probabilistic or a deterministic map function from inputs to outputs
- In general, we can model F as

$$F: (\{0,1\}^*)^K \rightarrow (\{0,1\}^*)K$$

K inputs (one per party); each input is a bitstring

K outputs

- Assume that F is computable in polynomial time, for example,

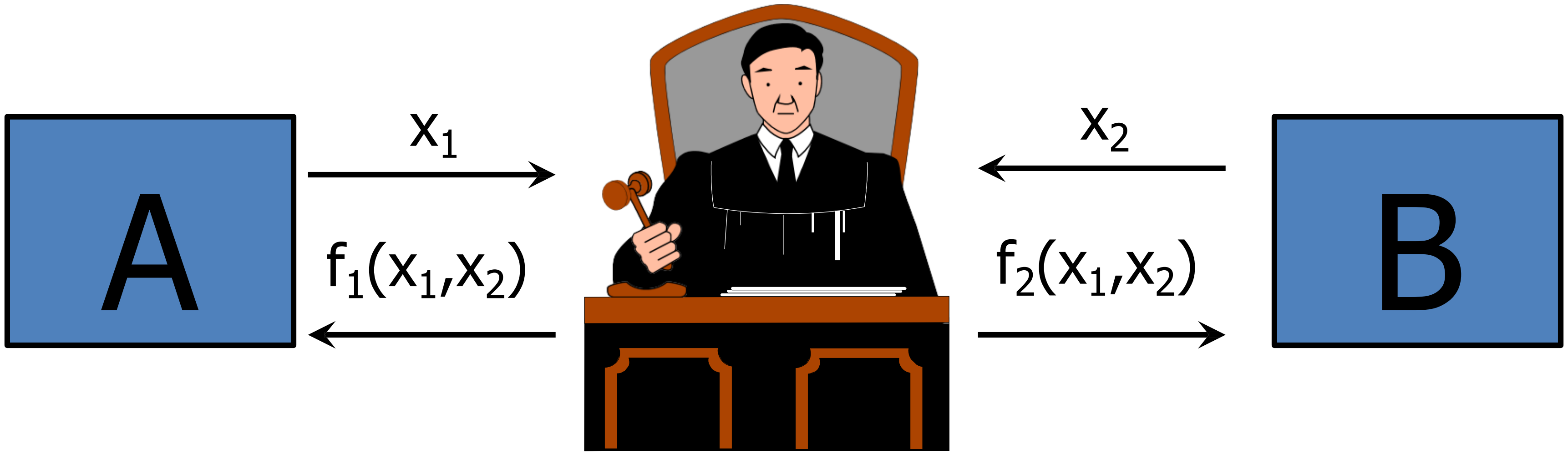
$$F(A, B) = ((A \geq B), (A \geq B))$$

Defining Security

- The real/ideal (simulation) model paradigm for defining security
  - **Ideal model:** parties send inputs to a trusted party, who computes the function for them
  - **Real model:** parties run a real protocol with no trusted help
- The security of a protocol can be checked by comparing what an adversary can do in a real protocol execution to what it can do in an ideal scenario (simulation)
- A protocol is secure if any adversary in the real model cannot do more harm than if it was involved in the ideal model
- In other words, an adversary should have an indistinguishable view of the protocols

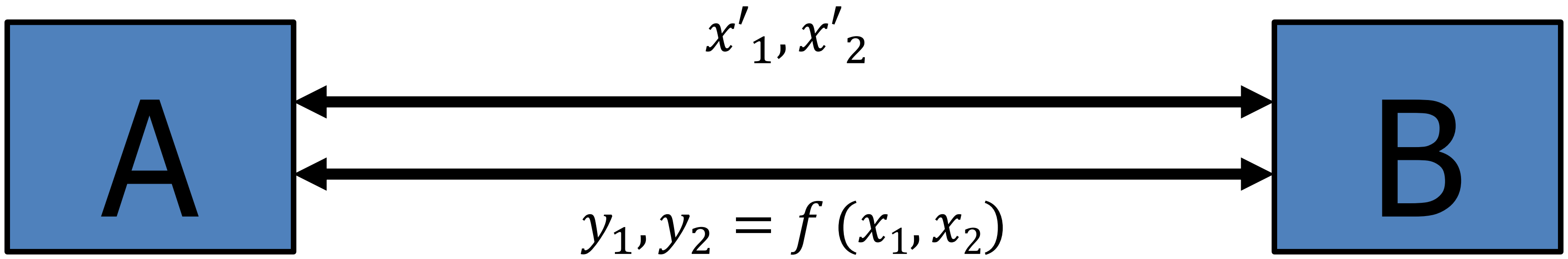
Ideal Model

- Intuitively, we want the protocol to behave "as if" a trusted third party collected the parties' inputs and computed the desired functionality
- Computation in the ideal model is secure by definition!



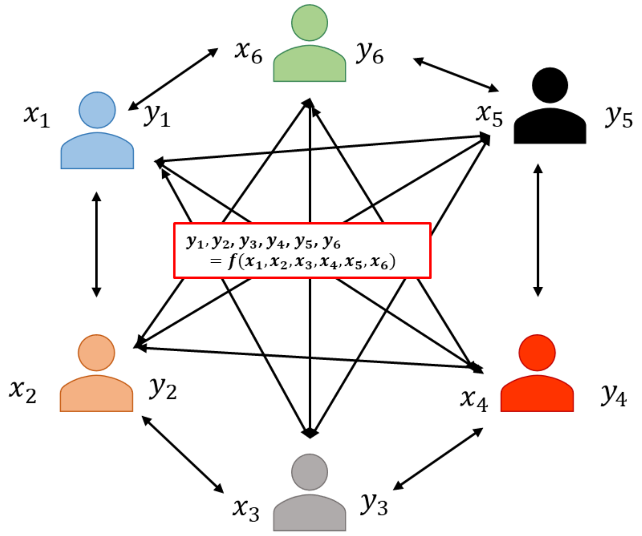
Real world

- No trusted third party
- Participants run some protocol amongst themselves without any help
- Despite that, secure protocol should emulate an ideal setting



Adversary Models

- Some participants may be dishonest (corrupt)
  - If all were honest, we would not need secure multi-party computation
- Semi-honest (aka passive; honest-but-curious)
  - Follows protocol, but tries to learn more from received messages than he would learn in the ideal model
- Malicious
  - Deviates from the protocol in arbitrary ways, lies about his inputs, may quit at any point
- For now, focus on semi-honest adversaries and two-party protocols



Properties of the Definition

- How do we argue that the real protocol "emulates" the ideal protocol?

- Correctness
  - All honest participants should receive the correct result of evaluating function F
    - Because a trusted third party would compute f correctly

- Privacy
  - All corrupt participants should learn no more from the protocol than what they would learn in ideal model

Yao’s Millionaire Problem

- Two millionaires, Alice and Bob want to know which of them is richer without revealing their actual wealth

- This problem is analogous to a more general problem where there are two numbers A and B and the goal is to solve the inequality without revealing the actual values of A and B

I am more rich

I am rich

Binary Numbers and Prefixes

- Given a binary number $\mathrm{X} = \mathrm{x_n x_{n-1} \ldots x_1}$
  - Example, $X = 110$ and $Y = 001$
- The set of all prefixes of X is the set

$$P = \{x_n, x_n x_{n-1}, x_n x_{n-1} x_{n-2}, \ldots., \mathrm{x_n x_{n-1} \ldots x_1}\}$$

- Example
- $X = 110, P = \{1,11,110\}$
- $Y = 001, P = \{0,00,001\}$

0-Encoding and 1-Encoding of Binary Numbers

- We define two types of encoding

1. 0-encoding of X is a set $S_X^0 = \{x_n x_{n-1} \dots x_{i+1} 1 | x_i = 0 \quad \forall\, 1 \le i \le n\}$
   - Invert the least significant bit of all prefixes of X tailing 0
   - Example,
     - $X = 110, P = \{1,11,110\} \rightarrow S_X^0 = \{111\}$
     - $Y = 011, P = \{0,01,011\} \rightarrow S_Y^0 = \{1\}$
2. 1-encoding of X is a set $S_X^1 = \{x_n x_{n-1} \dots x_{i+1} x_i | x_i = 1 \quad \forall\, 1 \le i \le n\}$
   - All prefix of X tailing 1
   - Example,
     - $X = 110, P = \{1,11,110\} \rightarrow S_X^1 = \{1,11\}$
     - $Y = 011, P = \{0,01,011\} \rightarrow S_Y^1 = \{01,011\}$

HE-based Solution: Setup

- Assume Alice has $X = \$7$ and Bob has $Y = \$2$
- Represent X and Y using binary numbers of length n
  - $X=7=111_2$, $Y=2=010_2$
- Find 1-encoding of X and 0-encoding of Y
  - $S_X^1 = \{1,11,111\}$
  - $S_Y^0 = \{1,011\}$

- $X > Y$ if and only if S_X^1 and S_Y^0 has a common element, i.e., $S_X^1 \cap S_Y^0 \neq \emptyset$
  - $\{1,11,111\} \cap \{1,011\} = \{1\}$

HE-based Solution: Protocol

1. Alice sends a matrix $T_{2\times n}$ to Bob, where $T[x_i, i] = E(1), T[\neg x_i, i] = E(r_i), r_i \neq 1$ is random

2. Bob does the following
   A. Finds S_y^0

   B. Computes $c_t = T[t_n, n] \cdot T[t_{n-1}, n-1] \cdot \ldots \cdot T[t_i, i]$ for each $t = t_n t_{n-1} \ldots t_i \in S_y^0$

   C. Chooses another $\mathrm{n} - |S_y^0|$ random ciphertexts forming a new set $\{c_1, c_2, \ldots, c_n\}$ where $c_i \neq 1$ and sends them back to Alice after random permutation

3. Alice decrypts all c_i, check whether some of them are 1 which indicates x>y

Lin, Hsiao-Ying, and Wen-Guey Tzeng. "An efficient solution to the millionaires' problem based on homomorphic encryption." In *Applied Cryptography and Network Security: Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005. Proceedings 3*, pp. 456-466. Springer Berlin Heidelberg, 2005.

HE-based Solution: Protocol

Alice (X=7=$(111)_2$)

Step 1

	0	1	2
0	$E(r_2)$	$E(r_1)$	$E(r_0)$
1	$E(1)$	$E(1)$	$E(1)$

Step 2

	0	1	2
0	$E(r_2)$	$E(r_1)$	$E(r_0)$
1	$E(1)$	$E(1)$	$E(1)$

$S_y^0 = \{1{,}011\}$

$c_{\{1\}} = T[1,0]= E(1)$

$c_{\{011\}} = T[0,2]T[1,1]T[1,0] = E(r_0)E(1)E(1)$

$\{E(r_0)E(1)E(1), E(1), E(c_1)\}$

Step 3

$\{E(r_0)E(1)E(1), E(1), c_1\}$

$D\big(E(r_0.1.1)\big) = r_0$

$D\big(E(1)\big) = 1$

$D\big(E(c_1)\big) = c_1$

Result:
X>Y

Bob (Y=2= $(010)_2$)

HE-based Solution: Another example

Alice (X=7=$(111)_2$)

Step 1

	0	1	2
0	$E(r_2)$	$E(r_1)$	$E(r_0)$
1	$E(1)$	$E(1)$	$E(1)$

Step 2

	0	1	2
0	$E(r_2)$	$E(r_1)$	$E(r_0)$
1	$E(1)$	$E(1)$	$E(1)$

$S_y^0 = \{111\}$

$c_{\{111\}} = T[1,2]T[1,1]\ T[1,0] = E(1)E(1)\ E(1)$

$\{E(1)E(1)E(1), E(c_2), E(c_1)\}$

Step 3

$\{E(1)E(1)E(1), E(c_2), E(c_1)\}$

$D\big(E(1.1.1)\big) = 1$

$D\big(E(c_2)\big) = c_2$

$D\big(E(c_1)\big) = c_1$

Result:
X>Y

Bob (Y=6= $(110)_2$)

HE-based Solution: Another example

Alice (X=5=$(101)_2$)

Step 1

	0	1	2
0	$E(r_2)$	$E(1)$	$E(r_0)$
1	$E(1)$	$E(r_1)$	$E(1)$

Step 2

	0	1	2
0	$E(r_2)$	$E(1)$	$E(r_0)$
1	$E(1)$	$E(r_1)$	$E(1)$

$S_y^0 = \{111\}$

$c_{\{111\}} = T[1,2]T[1,1]\ T[1,0] = E(1)E(r_1)\ E(1)$

$\{E(1)E(r_1)E(1), E(c_2), E(c_1)\}$

Step 3

$\{E(1)E(r_1)E(1), E(c_2), E(c_1)\}$

$D\big(E(1.r_1.1)\big) = r_1$

$D\big(E(c_2)\big) = c_2$

$D\big(E(c_1)\big) = c_1$

Result:
X<Y

Bob (Y=6= $(110)_2$)

HE-based Solution: Another example

Alice (X=10=$(1010)_2$)

Step 1

	0	1	2	3
0	$E(r_3)$	$E(1)$	$E(r_1)$	$E(1)$
1	$E(1)$	$E(r_2)$	$E(1)$	$E(r_0)$

Step 2

	0	1	2	3
0	$E(r_3)$	$E(1)$	$E(r_1)$	$E(1)$
1	$E(1)$	$E(r_2)$	$E(1)$	$E(r_0)$

$S_y^0 = \{1,01\}$

$c_{\{1\}} = T[1,0] = E(1)$

$c_{\{01\}} = T[0,0]T[1,1] = E(r_3)E(r_2)$

$\{E(1), E(r_3)E(r_2), E(c_1), E(c_2)\}$

Step 3

$\{E(1), E(r_3)E(r_2), E(c_1), E(c_2)\}$

$D(E(1)) = 1$

$D(E(r_3)E(r_2)) = r_3 r_2$

$D(E(c_1)) = c_1$

$D(E(c_2)) = c_2$

Result:
X>Y

Bob (Y=3= $(0011)_2$)

Private Set Intersection

Do we have common contacts on our phones?

Maybe…

We could see if we have common friends?

I don't care and I don't really like to give you my personal contacts

Have you heard of "secure multiparty computation" ?

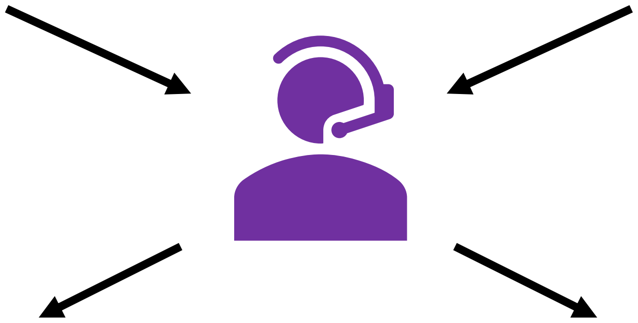
No, I am from KBS…

Typical Solution

Input: X Y

Output: $F(x,y) = X \cap Y$ set of common friends

As if… X Y



$F(x,y)$ $F(x,y)$

Our Specific Scenario

Input:	$X = x_1 \dots x_k$	$Y = y_1 \dots y_k$
Output:	$X \cap Y$ only	nothing

The Protocol

- Alice defines a polynomial of degree k whose roots are his inputs $x_1, \dots, x_k$
- $P(x) = (x_1 - x)(x_2 - x) \dots (x_k - x) = a_0 + a_1 x + \cdots + a_k x^k$
- Alice sends to server (S) homomorphic encryptions of polynomial's coefficients

$$Enc(a_0), \dots, Enc(a_k)$$

The Protocol

- For each input $y_i \in \ y_1, \dots, y_k$, Bob evaluates the polynomial using Paillier's additive homomorphic properties $(Enc(X + Y) = Enc(X).Enc(Y))$

$$Enc\big(\ P(y)\big) = \ Enc(\ a_0\ +\ a_1\ \cdot\ y^1\ +\ \dots + \ a_k\ \cdot\ y^k\ )$$
$$= \ Enc(a_0)\ \cdot\ Enc\ (a_1)^{\ y}\ \cdot\ \dots\ \cdot\ Enc\ (a_k)^{\ y^k}$$

- Bob sends $Enc(\ r_i\ \cdot\ P(y_i)\ +\ y_i\ )$

- S sends (permuted) results back to C

The Protocol

- Alice receives $Enc(\, r_i \;\cdot\; P(y_i) \;+\; y_i \,)$ and decrypts each one

if y ∈ X ∩ Y → Enc (y)

otherwise → Enc (random)

Example

- Alice {3, 4, 5}, Bob {5, 6}
- Alice creates polynomial

$$P(x) = (3-x)(4-x)(5-x) = 60 - 46x + 12x^2 - x^3$$

- Alice sends: $Enc(60), Enc(-46), Enc(12), Enc(-1)$
- Bob evaluates
  - $Enc(P(5)) = Enc(60) \cdot Enc(-46)^5 \cdot Enc(12)^{5^2} \cdot Enc(-1)^{5^3}$
  - $Enc(P(6)) = Enc(60) \cdot Enc(-46)^6 \cdot Enc(12)^{6^2} \cdot Enc(-1)^{6^3}$
- Bob sends **permuted** results $\{(r_2.Enc(P(6)) + 6), (r_1.Enc(P(5)) + 5)\}$
- Alice decrypts the items in the list
  - $Dec\left((r_2.Enc(P(6)) + 6)\right) \Rightarrow$ Random
  - $Dec\left((r_1.Enc(P(5)) + 5)\right) \Rightarrow 5$ (i.e.

Efficiency

- Communication is O(k)
  - Alice sends k coefficients
  - Bob sends k evaluations on polynomial

- Computation
  - Alice encrypts and decrypts k values
  - Bob:
    - $\forall y \in Y,$ computes $Enc(ry \cdot P(y) + y)$,
    - Using k exponentiations
    - Total $O(k^2)$ exponentiations

Summary

- Secure multiparty computation is a framework for secure distributed computing
- If a trusted third-party exists, no need for SMC
- Security and privacy of SMC can be proven using simulation of real/ideal model
- Example of SMC: Yao's Millionaire problem
- Solution to Yao's Millionaire problem using Homeomorphic encryption
- Next, a solution for general Yao's millionaire problem
  - K parties
  - Any arbitrary function