EXPLORING CRYPTOGRAPHY PROTOCOLS

WITH LIMITED EMPHASIS ON MATHEMATICS ☺

Presented By
@7eRoM

ATTENTION

THESE SLIDES HAVE BEEN CRAFTED USING THE FOUNDATION OF MY MSC COURSE IN CRYPTOGRAPHY PROTOCOLS AT THE UNIVERSITY OF ISFAHAN.

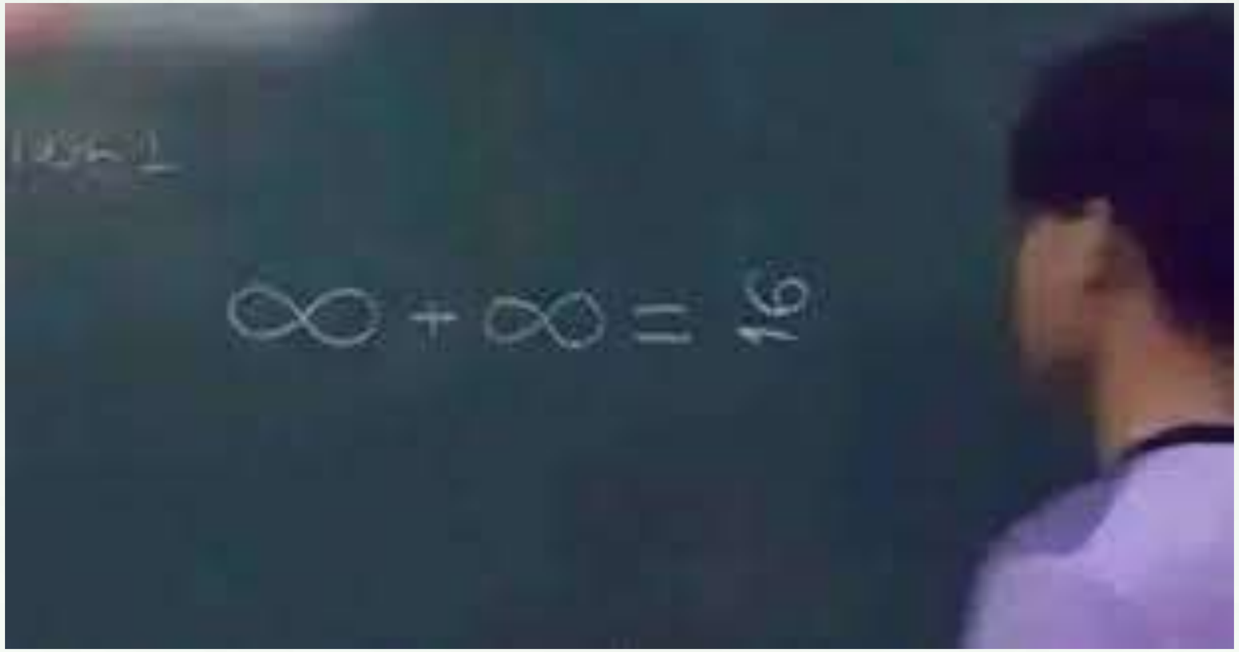
I'VE MADE ADJUSTMENTS TO THE CONTENT TO ALIGN WITH THE SPECIFIC OBJECTIVES OF THIS PRESENTATION.

ALSO, MY INTENTION HAS BEEN TO MINIMIZE THE USE OF MATHEMATICAL CONCEPTS, WHICH MAY RESULT IN SOME CONCEPTS BEING SIMPLIFIED OR LESS PRECISE.

Agenda

1. Identification and Entity Authentications Protocols
2. Zero Knowledge Protocols
3. Key Establishment Protocols
4. Threshold Cryptography and Secret Sharing Protocols
5. Special Purpose Protocols (like simultaneous contract signing, mental poker, fair exchange)
6. Identity Based Cryptography
7. Types of Digital Signatures
8. Secure Multiparty Computations

Secure Multiparty Computations



Secure Multiparty Computations

Content

- Introduction
- Yao's Garbled Circuit
- Median
- k^{th} Element
- Greater Than
- Private Set Intersection
- Private Bidding

Secure Multiparty Computations

Introduction

- **Computation between parties who do not trust each other**
- Example: 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
- Example: auctions
  - Each bidder makes an offer
    - Offer should be committing! (can’t change it later)
  - Goal: determine whose offer won without revealing losing offers

Secure Multiparty Computations

Introduction

- Example: distributed data mining
  - Two companies want to compare their datasets without revealing them
    - For example, compute the intersection of two lists of names
- Example: database privacy
  - Evaluate a query on the database without revealing the query to the database owner
  - Evaluate a statistical query on the database without revealing the values of individual entries
  - Many variations

Secure Multiparty Computations

Introduction

- Also known as Secure Multiparty Computation

- 2-party SFE: Alice has x, Bob has y, and they want to compute two functions $f_A(x,y)$, $f_B(x,y)$. At the end of the protocol
  - Alice learns $f_A(x,y)$ and nothing else
  - Bob learns $f_B(x,y)$ and nothing else

Secure Multiparty Computations

Introduction

- n-party SFE: n parties each have a private input, and they join compute functions

- How to let n parties, $P_1,..,P_n$ compute a function $F(x_1,..,x_n)$
  - Where input x_i is known to party P_i
  - Party P_i learns nothing more than what he can learn from his own input
  - and the final output $F(x_1,..,x_n)$

Secure Multiparty Computations

Introduction

Secure Multiparty Computations

Yao's Garbled Circuit for 2-party SFE

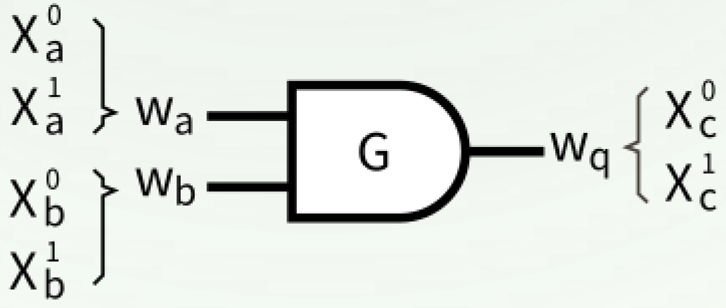
- For simplicity, assume that Alice has x, Bob has y, Alice learns f(x,y), and Bob learns nothing
  - represent f(x,y) using a boolean circuit
  - Alice encrypts the circuit and sends it to Bob
    - in the circuit each wire is associated with two random values
  - Alice sends the values corresponding to her input bits
  - Bob uses OT to obtain values for his bits
  - Bob evaluates the circuits and sends the result to Alice

Secure Multiparty Computations

Yao's Garbled Circuit for 2-party SFE

- Alice holds a bit b_A and Bob holds a bit b_B. They want to jointly compute the AND of their private bits $b_A \wedge b_B$.

- How can they do this privately?



Secure Multiparty Computations

Yao's Garbled Circuit for 2-party SFE

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

1. Alice replaced 0 and 1 with randomly generated strings called *labels*: X_A^0 , X_A^1 , X_B^0 , X_B^1
2. Then, Alice generates 4 ciphertexts according to the truth table of the AND gate.
3. Alice sends over the 4 ciphertexts c_{00}, c_{01}, c_{10}, c_{11} in permuted order to Bob.
4. Alice also sends the corresponding key for its own input bit $X_A^{b_A}$ to Bob.
5. Alice and Bob proceeds in an oblivious transfer protocol where Alice plays the sender and Bob plays the receiver.
   - Alice's input: (X_B^0, X_B^1)
   - Bob's input: b_B

A	B	AND
X_A^0	X_B^0	$c_{00} = E_{X_A^0}(E_{X_B^0}(0))$
X_A^0	X_B^1	$c_{01} = E_{X_A^0}(E_{X_B^1}(0))$
X_A^1	X_B^0	$c_{10} = E_{X_A^1}(E_{X_B^0}(0))$
X_A^1	X_B^1	$c_{11} = E_{X_A^1}(E_{X_B^1}(1))$

Secure Multiparty Computations

Yao's Garbled Circuit for 2-party SFE

6. At the end of the OT protocol, Bob receives: $X_B^{b_B}$

7. Now Bob has 4 ciphertexts $c_{00}, c_{01}, c_{10}, c_{11}$ (in permuted order) and a pair of keys $(X_A^{b_A}, X_B^{b_B})$. Bob tries to decrypt each ciphertext $D_{X_B^{bB}}(D_{X_A^{bA}}(c))$. Then, 3 out of the 4 ciphertexts should decrypt to some random garbage. 1 ciphertext should decrypt to either 0 or 1.

8. Either Alice can share her information to Bob or Bob can reveal the output to Alice such that one or both of them learn the output.

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

A	B	AND
X_A^0	X_B^0	$c_{00} = E_{X_A^0}(E_{X_B^0}(0))$
X_A^0	X_B^1	$c_{01} = E_{X_A^0}(E_{X_B^1}(0))$
X_A^1	X_B^0	$c_{10} = E_{X_A^1}(E_{X_B^0}(0))$
X_A^1	X_B^1	$c_{11} = E_{X_A^1}(E_{X_B^1}(1))$

Secure Multiparty Computations

Median ($k = n/2$)

Set S_A

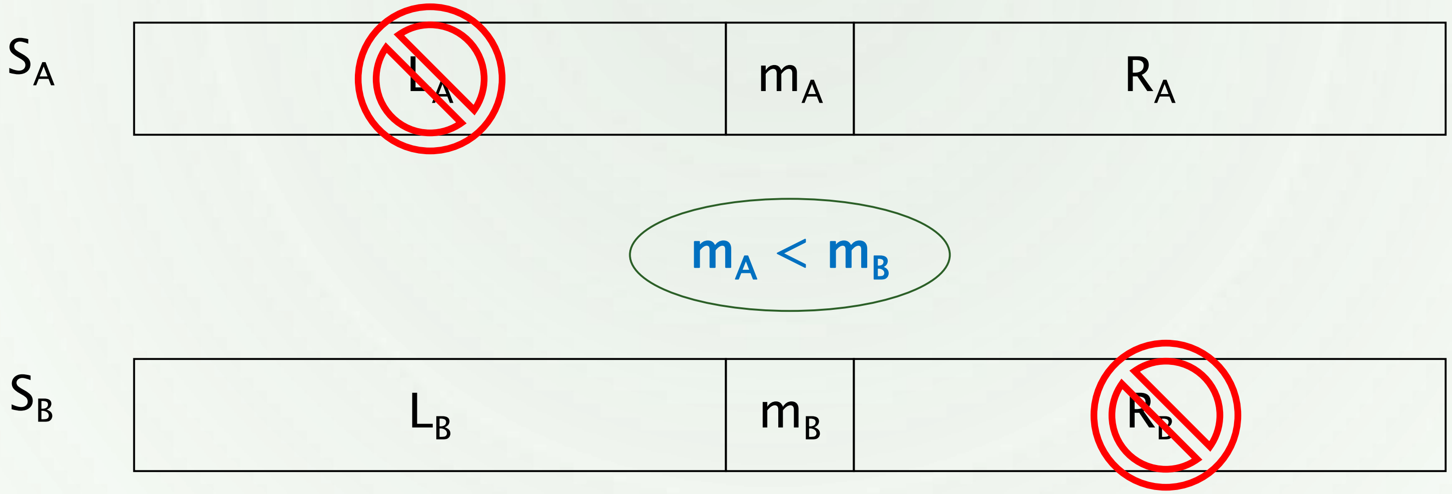
Set S_B

Assume $|S_A| = |S_B|$

How can we find the median of $S_A \cup S_B$

Secure Multiparty Computations

Median (k = n/2)



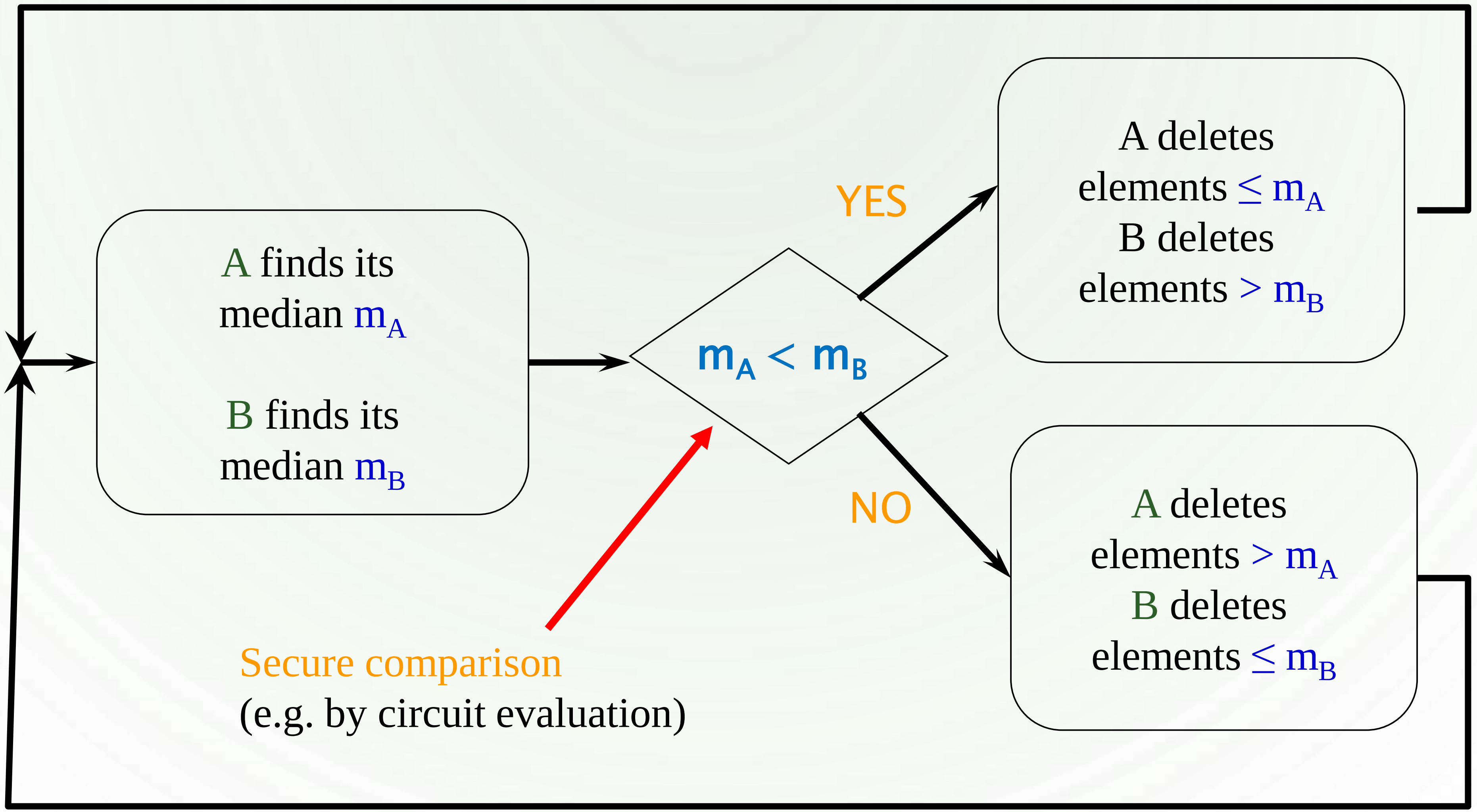
L_A lies below the median, R_B lies above the median.

New median is same as original median.

Recursion → Need `log n` rounds
(assume each set contains `n=2`^{i} items)

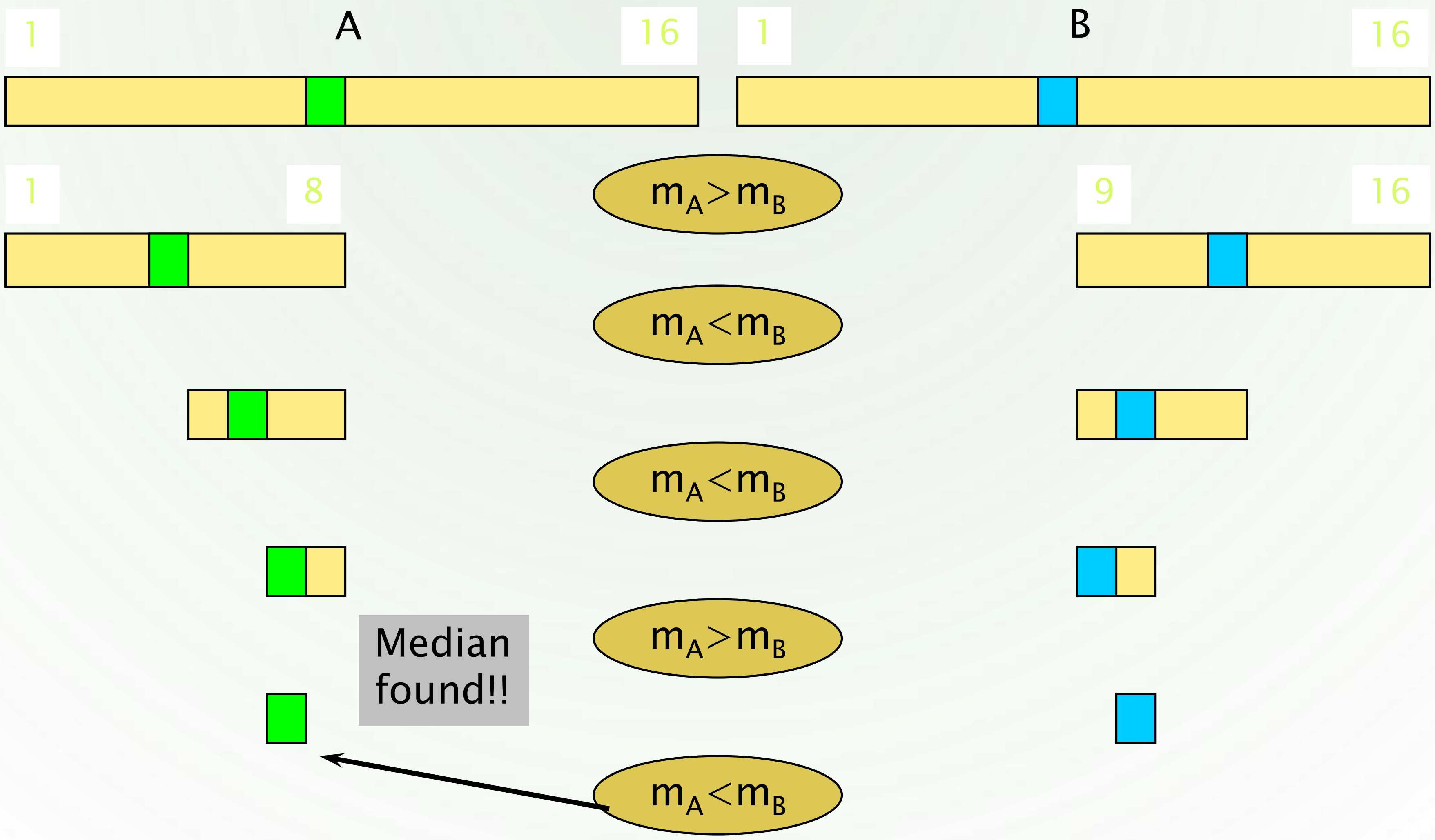
Secure Multiparty Computations

Median (k = n/2)



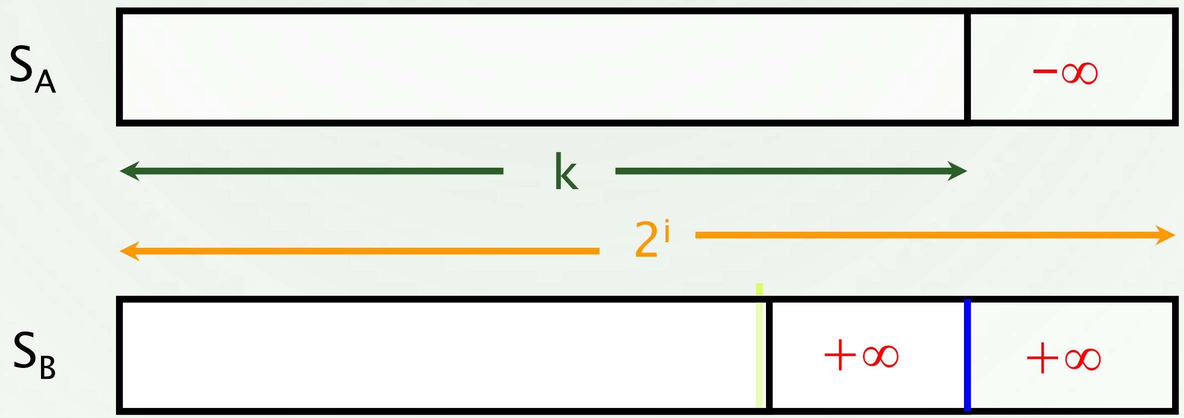
Secure Multiparty Computations

Median – Example



Secure Multiparty Computations

Median – Arbitrary input size, arbitrary k



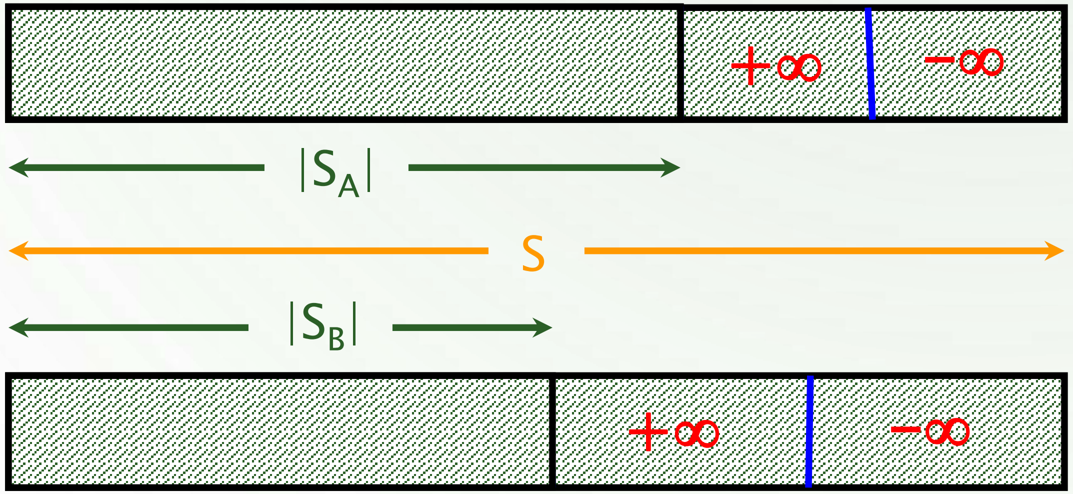
Size should be a power of 2

$\boxed{\text{median of new inputs} = k^{th} \text{ element of original inputs}}$

Secure Multiparty Computations

k^{th} element – Hiding size of inputs

- Can search for k^{th} element without revealing size of input sets.
- However, $k=n/2$ (median) reveals input size.
- Solution: Let $S=2^i$ be a bound on input size.



Median of new datasets is same as median of original datasets.

Secure Multiparty Computations

Greater Than

$$x_i = x_{i,\ell} x_{i,\ell-1} \cdots x_{i,1}$$

Define two sets of prefix strings:

$$X_i^1 = \{x_{i,\ell} x_{i,\ell-1} \ldots x_{i,j+1} \mid x_{i,j} = 1\}$$

$$X_i^0 = \{x_{i,\ell} x_{i,\ell-1} \ldots x_{i,j+1} \mid x_{i,j} = 0\}$$

$$GT(x_1, x_2) = \begin{cases} 1 & X_1^1 \cap X_2^0 \neq \phi \\ 0 & X_1^1 \cap X_2^0 = \phi \end{cases}$$

Compare $x_1 = 234$

$x_2 = 228$

$$x_1 = 234 \Rightarrow x_1 = 11101010$$

$$x_2 = 228 \Rightarrow x_2 = 11100100$$

$$X_1^1 = \{\lambda, 1, 11, \boxed{1110}, 111010\}$$

$$X_2^0 = \{111, \boxed{1110}, 111001, 1110010\}$$

Secure Multiparty Computations

Private Set Intersection

- One-way PSI

Client: $X = \{x_1, \ldots, x_n\}$

Server: $Y = \{y_1, \ldots, y_n\}$

Output:

Client learns $X \cap Y$

Server learns nothing

Secure Multiparty Computations

Homomorphic Encryption

- Allows computations to be performed on encrypted data without first having to decrypt it

- Enables mathematical computations to be performed directly on the encrypted data
- Enables organizations to store encrypted data in a public cloud for future process
- Enable new services by removing privacy barriers inhibiting data sharing or increasing security to existing services (for example, predictive analytics in health care)

Secure Multiparty Computations

Homomorphic Encryption

- Given $\mathrm{Enc}(M1)$, $\mathrm{ENC}(M2)$, can compute (without knowing the decryption key)
  - $\mathrm{Enc}(M1+M2)$
  - $\mathrm{Enc}(c \cdot M1)$ for any constant c
  - I.e. $\mathrm{Enc}(a_0) + \mathrm{Enc}(a_1)x + \ldots + \mathrm{Enc}(a_n)x^n = \mathrm{Enc}(P(x))$

- Examples: El Gamal, Paillier

We could have discussed 1 or 2 sessions on Homomorphic Encryption.

However, we need to skip it for now.

Secure Multiparty Computations

Private Set Intersection

Client: $X = \{x_1, \dots, x_n\}$

Server: $Y = \{y_1, \dots, y_n\}$

- Client defines a polynomial of degree n whose roots are $x_1, \dots, x_n$
  - $P(y) = (x_1 - y) \cdot (x_2 - y) \cdot \dots \cdot (x_n - y) = a_n y^n + \dots + a_1 y + a_0$
- Sends to server homomorphic encryptions of coefficients
  - $Enc(an), \dots, Enc(a_0)$
- Server uses homomorphic properties to compute

  $\forall y \; Enc(r \cdot P(y) + y)$ (r is random)
- If $y \in X \cap Y$ result is $Enc(r \cdot 0 + y) = Enc(y)$, otherwise result is $Enc(random)$
- Server sends permuted results to Client
- Client decrypts, compares to his list

Special Purpose Protocols

Private Bidding

$$GT(x_1, x_2) = \begin{cases} 1 & X_1^1 \cap X_2^0 \neq \phi \\ 0 & X_1^1 \cap X_2^0 = \phi \end{cases}$$

$X_A = X_A^1 = \{X_{A,1}^1, X_{A,2}^1, \dots, X_{A,n}^1\}$
$a_i = H(X_{A,i}^1)$
$A = \{a_1, a_2, \dots, a_n\}$

$1 \leq \text{random } u \leq q-1$
$m_i = H(a_i^u)$
$M = \{m_1, m_2, \dots, m_n\}$

$$\xrightarrow{\textbf{M}}$$

$$\xleftarrow{\textbf{T , E}}$$

$X_B = X_B^0 = \{X_{B,1}^0, X_{B,2}^0, \dots, X_{B,n}^0\}$
$b_i = H(X_{B,i}^0)$
$B = \{b_1, b_2, \dots, b_n\}$

$1 \leq \text{random } v \leq q-1$
$\mathrm{t}_i = H(b_i^v)$
$T = \{t_1, t_2, \dots, t_n\}$

$f_i = H(t_i^v)$
$F = \{f_1, f_2, \dots, f_n\}$

$e_i = H(m_i^v)$
$E = \{e_1, e_2, \dots, e_n\}$

Check if there are similar elements in both $\boldsymbol{E}$ and $\boldsymbol{F}$

Secure Multiparty Computations

Learning Cryptography straightforward!

Ref

1. Cryptography Protocols Course, Dr. Hamid Mala, University of Isfahan
2. https://crypto.stanford.edu/cs355/18sp/lec6.pdf
3. https://en.wikipedia.org/wiki/Garbled_circuit
4. https://en.wikipedia.org/wiki/Homomorphic_encryption
5. https://www.techtarget.com/searchsecurity/definition/homomorphic-encryption
6. https://www.researchgate.net/figure/Millionaires-problem_fig1_320290997
7. https://www.iconfinder.com/UsersInsights
8. https://www.iconfinder.com/Chanut-is
9. https://www.iconfinder.com/iconsets/softwaredemo