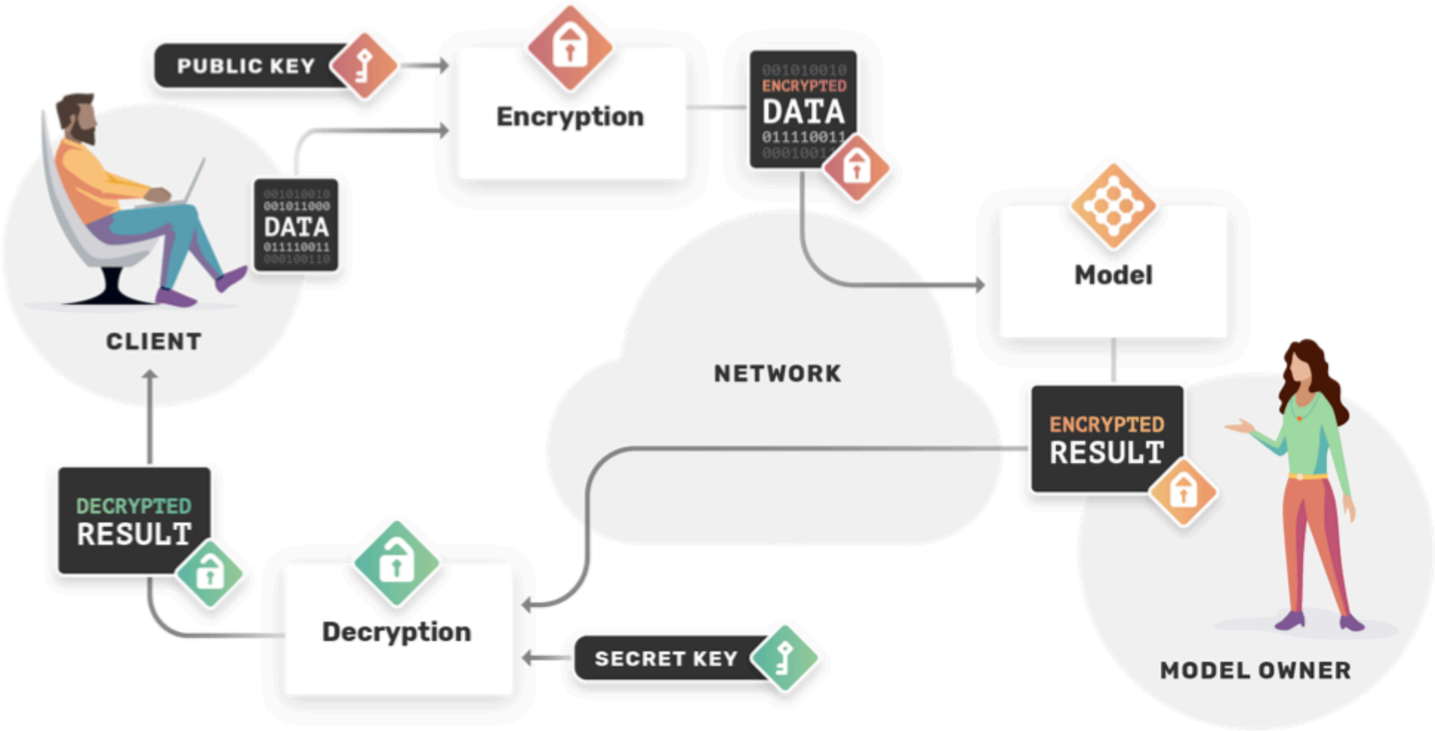
Qianying Liao

- Former Doctoral Researcher in University of Coimbra
- Current PhD student at DistriNet Research Group, KU Leuven
- My research area is Privacy Preserving Machine Learning.
- Previously, I was working on integrating Homomorphic Encryption into Machine Learning.
- I am now working the general threat models, frameworks and protocols for distributed collaborative machine learning.



November 23, 2023

Homomorphic Encryption

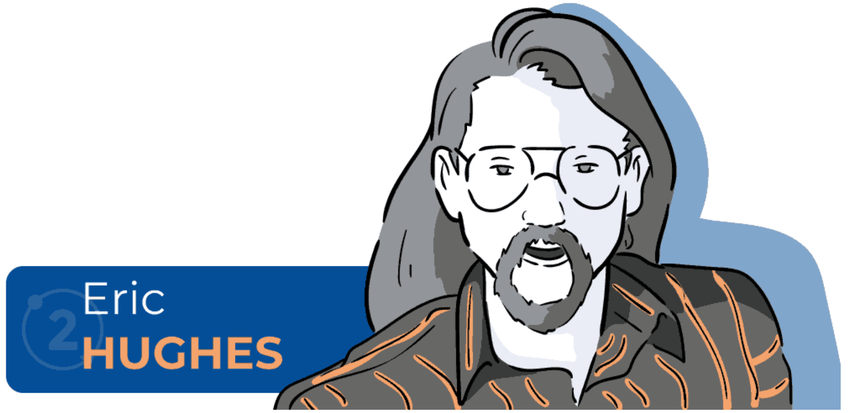
Privacy and Security Lecture

Outlines

- Introduction to Homomorphic Encryption
- Basic Applications
- Partially Homomorphic Encryption (PHE)
- Somewhat & Fully Homomorphic Encryption (SHE & FHE)
- Homomorphic Encryption Issues
- Applications in ML

Introduction to Homomorphic Encryption

Privacy in "A Cypherpunk's Manifesto" (1993)



Encryption Fits!

A **private** matter is something one doesn't want the whole world to know, but a **secret** matter is something one doesn't want anybody to know.

Privacy is the power to selectively reveal oneself to the world.

Basic satisfactions of privacy require:

Controllable Accessibility + Confidentiality

“Traditional” Encryption

- Syntax: (GEN, **ENC**, DEC)
- Correctness:
  - $(pk, sk) \leftarrow Gen$
  - $Dec_{sk}(Enc_{pk}(m)) = m$

“Traditional” Encryption

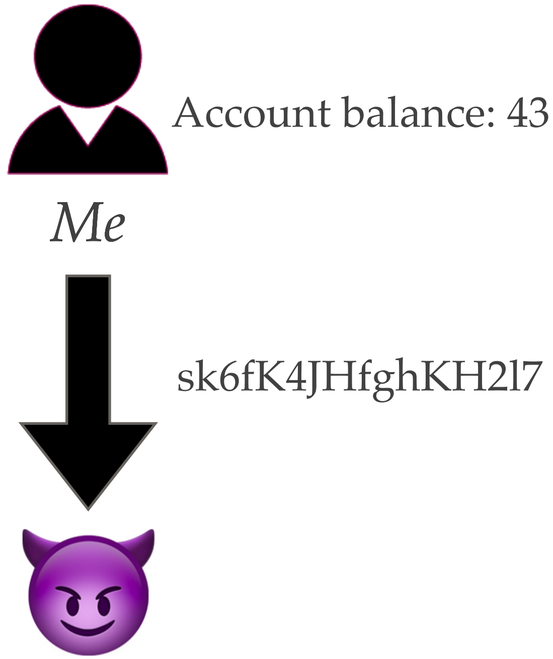
- Usually follows Kerckhoff ’s principle of Secure Encryption:
  - The attacker knows the encryption scheme
  - The only secret is the key
  - The key must be chosen at random; kept secret
- In "traditional" static encryption (e.g. AES), the only thing you can do with encrypted data is to decrypt it.

The Dream HE Promised

Homomorphic Encryption Enables that!

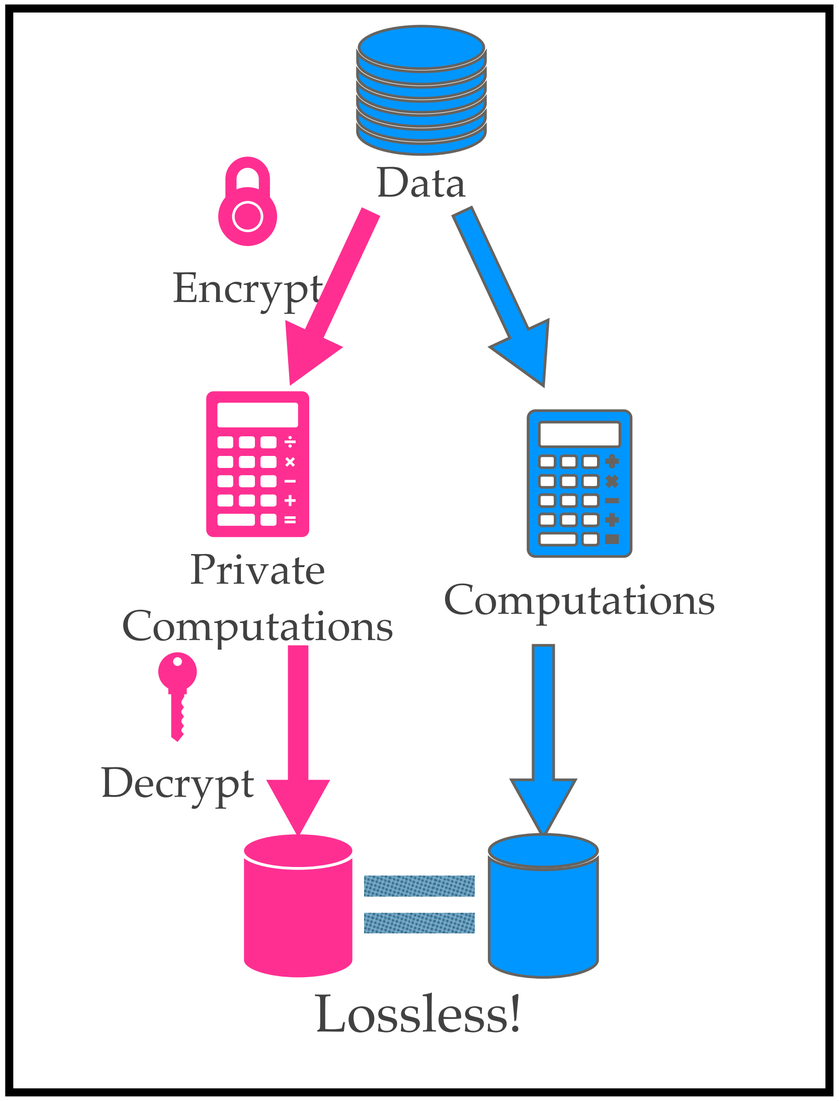
Wouldn't it be nice to be able to…

- Encrypt my data before sending it to the cloud
- While still allowing the cloud to process them
- Cloud returns encrypted answers
  - That I can decrypt



A Possibly Bad but Capable Entity

What is Homomorphic Encryption (HE)?



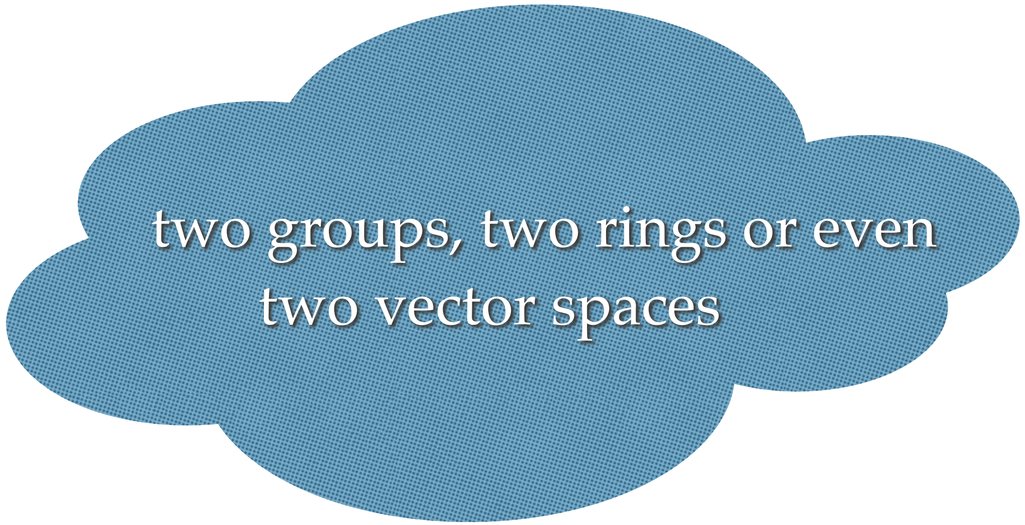
"Homomorphic encryption is a form of encryption that allows specific types of computations to be carried out on ciphertext and generate an encrypted result that, when decrypted, matches the result of operations performed on the plaintext."

By Yi, X., Paulet, R., & Bertino, E. (2014).

Must satisfy usual notion of Semantic Security.

What does Homomorphic mean?

- It is a structure-preserving map between two algebraic structures of the same type.



Homomorphism Group Definition

A homomorphism is a structure-preserving map between two algebraic structures, such as groups. [2]

$+$ *or* $\times$

Any function can be expressed by addtions and multiplications.

Given the Set G, Operation $\circ$ (group law of G), the Group is defined by $(G, \circ)$.

The Group $(G, \circ)$ must satisfy the following four axioms:

- Closure: $\forall a, b \in G \Rightarrow a \circ b \in G$
- Associativity: $\forall a, b, c \in G \Rightarrow (a \circ b) \circ c = a \circ (b \circ c) \in G$
- Identity element: $\exists! e \in G, \forall a \in G \Rightarrow e \circ a = a \circ e = a$
- Inverse element: $a \in G, \exists b \in G, a \circ b = b \circ a = e$, where e is the identity element.
- Commutativity* (Abelian Group)*: $\forall a \in G, \forall b \in G, a \circ b = b \circ a$

Operations are limited!

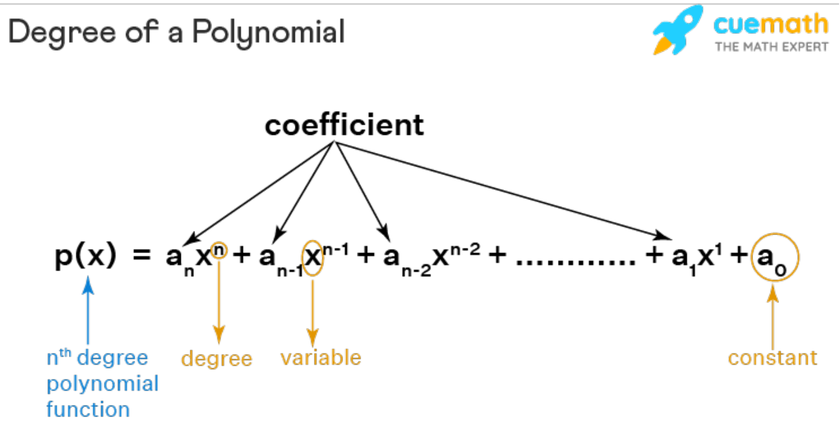
HE Timeline

- 1978 - Rivest, Adleman, Dertouzos
  - pose problem
- 2009 - Gentry
  - first candidate solution
  - bootstrapping technique
- 2011 - Brakerski, Vaikuntanathan
  - Lattice-based
- Even since then
  - new schemes
  - efficiency improvements
  - implementation: SEAL, HElib, ..
  - all based on lattice and bootstrapping

Categorizations of HE algorithms

$\circ$ denotes operations

d denotes the number of operations



- Partially HE (PHE): $\circ = \{+ \; or \; \times | \forall d < bound, d \in \mathbb{Z}\}$
- Somewhat HE (SHE): $\circ = \{+, \times | \forall d < N, d \in \mathbb{Z}\}$ ($5 \leq N \leq 15$ acc. ref. [3]).
- Fully HE (FHE): $\circ = \{+, \times | \forall d \in \mathbb{Z}\}$

Homomorphic Encryption Operations

($\langle . \rangle$ denotes encryption)

The operation of Homomorphic Encryption can be formally expressed as

- Supposed we have a bunch of ciphertexts,
$$m_1, m_2, \ldots, m_n \rightarrow \langle m_1 \rangle, \langle m_2 \rangle, \ldots, \langle m_n \rangle$$

- Ciphertext Calculation Function f1; Plaintext Calculation Function f2;
$$f_1(\langle m_1 \rangle, \langle m_2 \rangle, \ldots, \langle m_n \rangle) \rightarrow \langle f_2(m_1, m_2, \ldots, m_n) \rangle$$

Cryptographic methods for private and secure computations

Few rounds of interaction (typically Arithmetic Circuit, besides FHE)

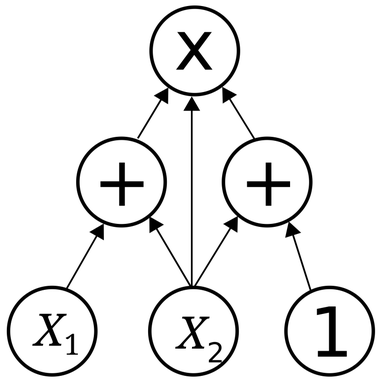
Many rounds of interaction (Typically Boolean Circuit)

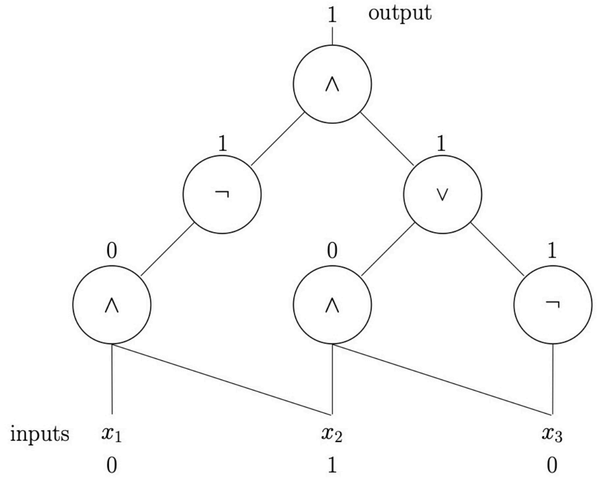
Homomorphic Encryption

Number of Interactions

GMW protocol & general MPC

Custom Protocols (MPC + HE)



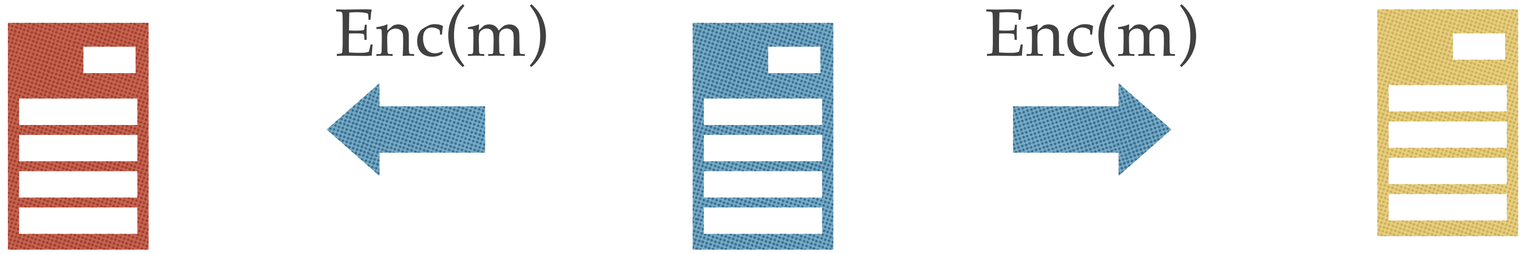


Basic Applications

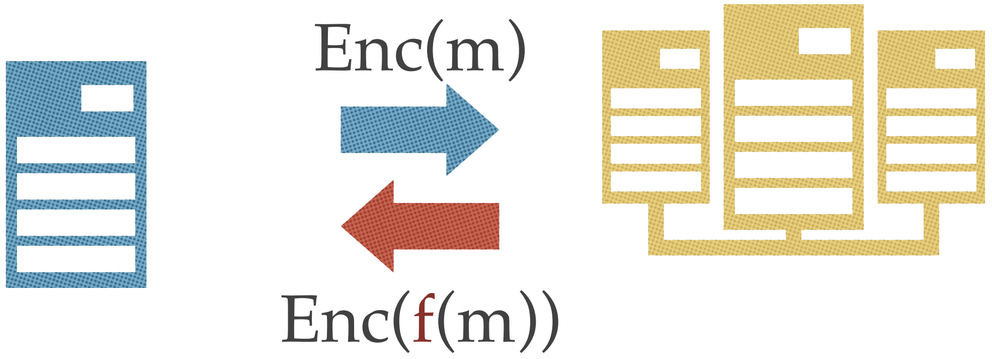
Being able to compute directly with ciphertexts is more powerful than you might think.

Homomorphic Encryption

- Encryption: used to protect data at rest or in transit



- Homomorphic Encryption: support computations on encrypted data



Example 1: Confidential Computation

- Do you want to compute something using your private data with someone's secret function in a secure environment?

I want to know how cute my cat is…

Cuteness points 100!

Me

Enc(features_my_cat)

Enc(points of cuteness)

Cloud

$$Cuteness = \alpha \times Hairiness + \beta \times Clumsiness + \delta \times Weight$$

Private Cuteness Computation

Example 2: Private Brith Parent Searching

Find my birth Parents

Other participants won't learn my DNA.

Birth Parents

Decrypt, Finds the Birthparent

Other participants won't learn extra citizen's information.

Me

(Holds pk, My DNA)

Citizen Database

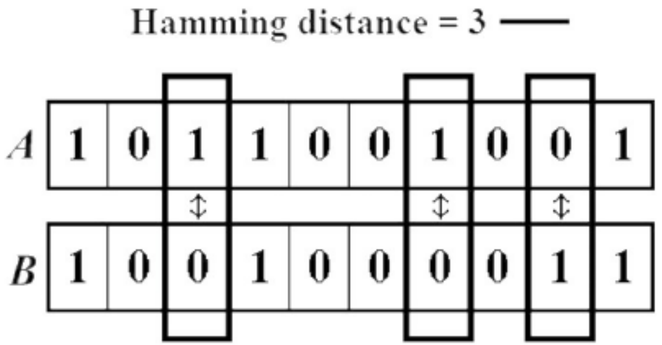
(Holds pr, citizen information database)

Enc(My DNA)

Other participants won't learn extra DNA information.

Enc(DNA_HD_1), Enc(citizen ID_1)
Enc(DNA_HD_2), Enc(citizen ID_2)
…,
Enc(DNA_HD_N), Enc(citizen ID_N)

DNA Database



Private Computations of Hamming distance

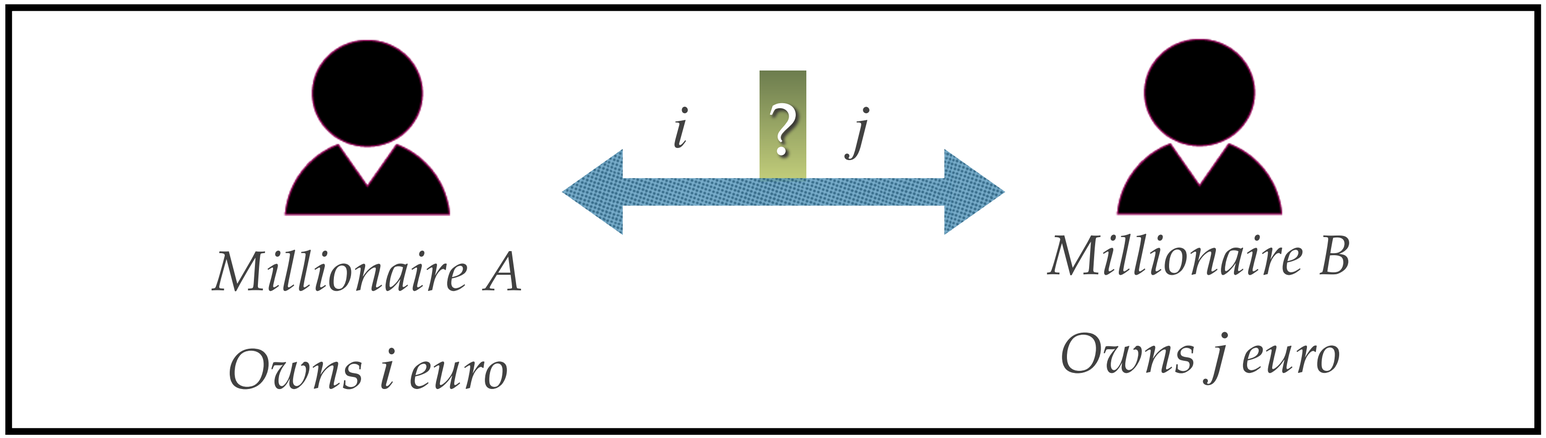
Example 3: Solution to Yao's Millionaire Problem

Andrew Chi-Chih Yao

Theoretical Computer Scientist

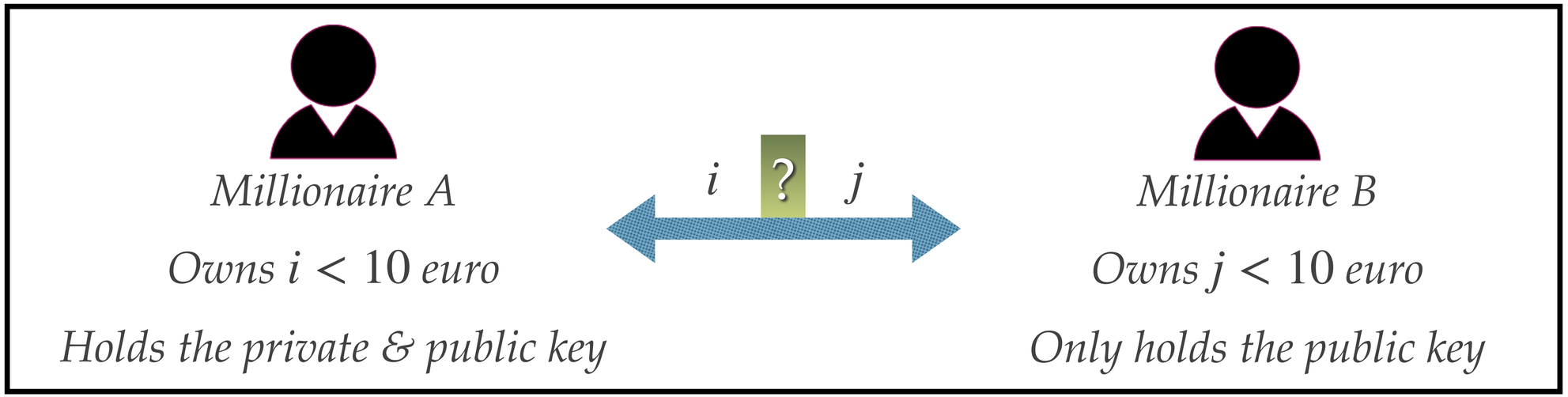
Prof of Tsinghua University, Beijing

Turing Award Winner (2000)

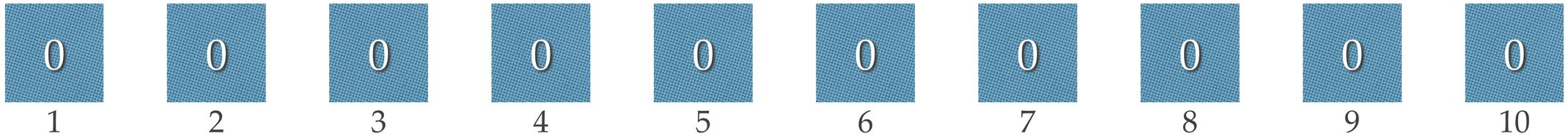


- Proposed by Andrew Chi-Chih Yao in 1982
- A Secure Multi-party Computation Problem

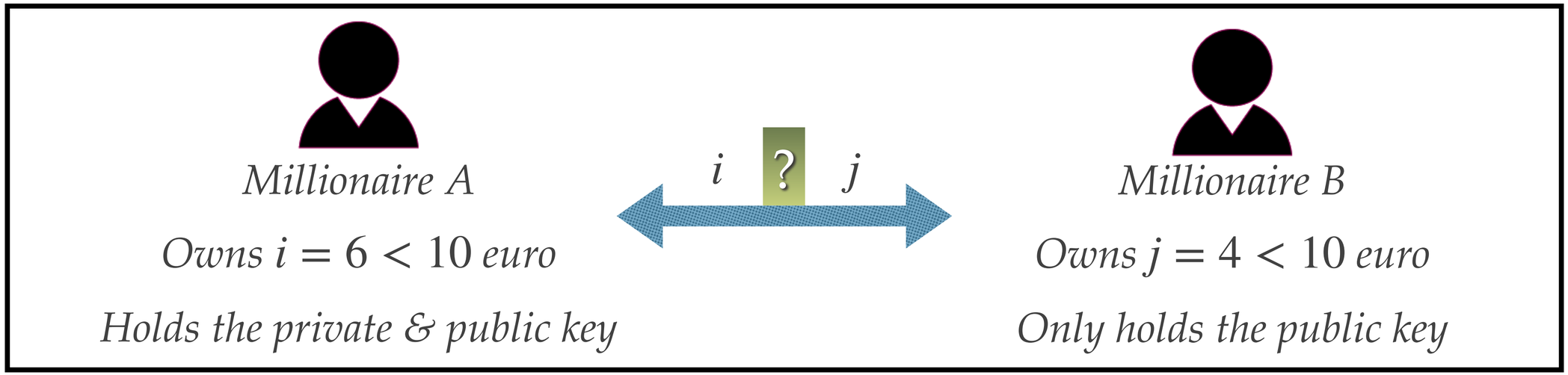
Yao's Millionaire Problem: Graphical Example



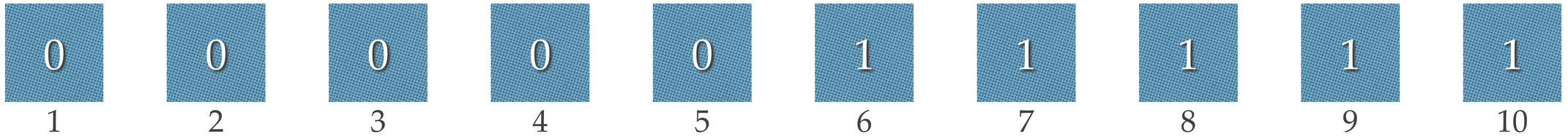
Supposed $i = 6$ and $j = 4$



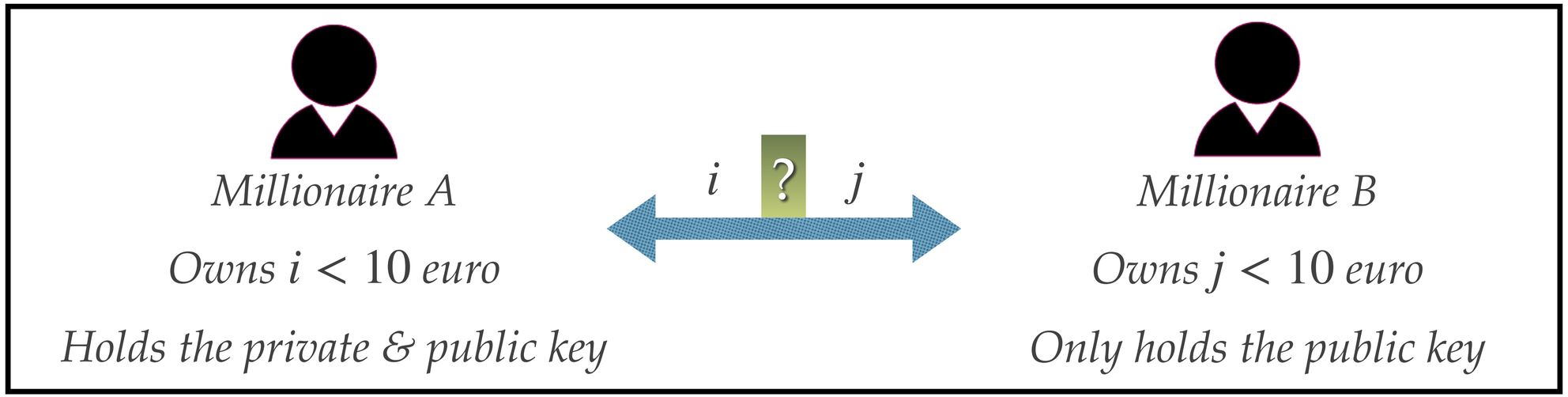
Yao's Millionaire Problem: Graphical Example



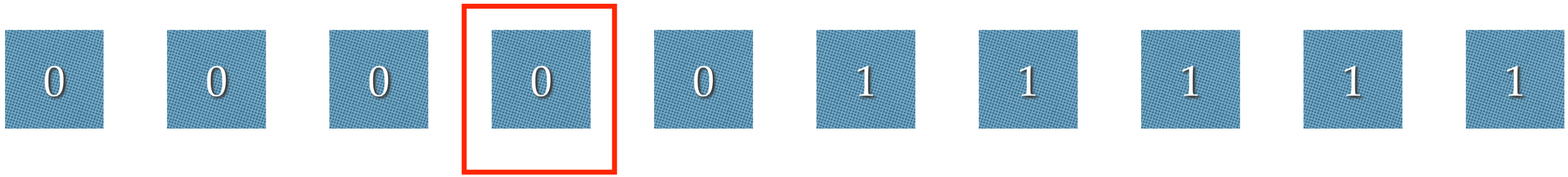
Millionaire A fills 1 to the box with an index greater than 6.



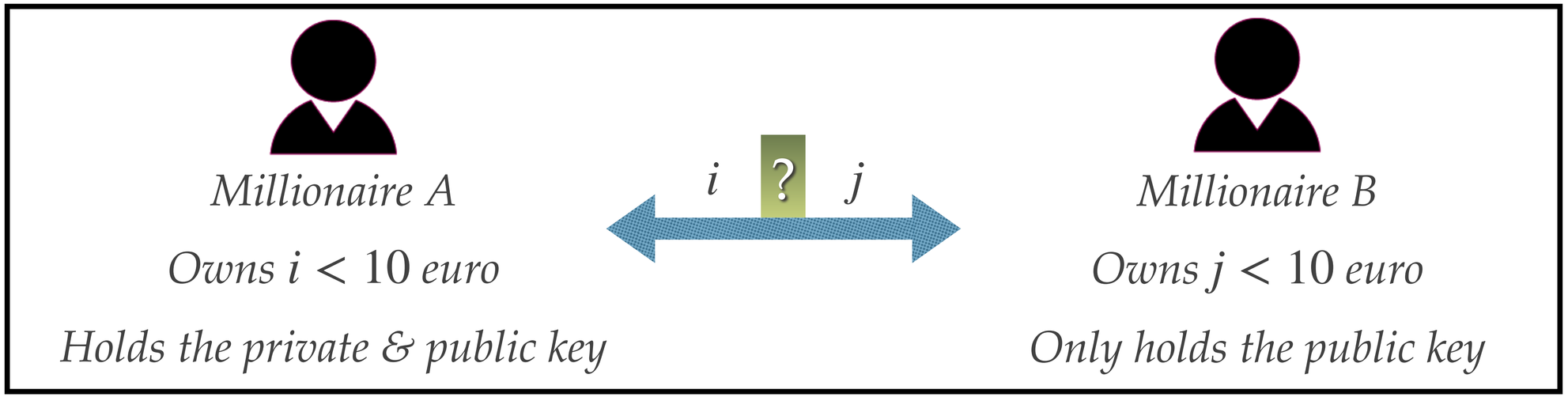
Yao's Millionaire Problem: Graphical Example



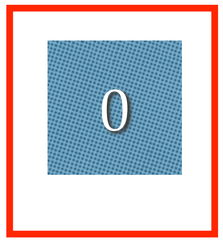
Millionaire B retrieves the 4-th box and removes all the boxes' index labels.



Yao's Millionaire Problem: Graphical Example



Millionaire B asks Millionaire A to open the box.



$$\text{Comparison Result} = \begin{cases} 0, & i > j \\ 1, & i \leq j \end{cases}$$

Yao's Millionaire Problem: Math

Millionaire A

Owns $i = 6 < 10$ *euro*

Holds the pr and pk

Millionaire B

Owns $j = 4 < 10$ *euro*

Only holds pk

Selects a big number X

$\langle X \rangle = Enc(X)$

$\langle X \rangle - j + 1 = \langle m \rangle$

$\langle m \rangle$

$\langle X \rangle = \langle X \rangle - j + j$

$\langle X \rangle - j + 1, \langle X \rangle - j + 2, \ldots, \langle X \rangle - j + 10$

$\mathbf{y} = \{y_u = Dec(\langle X \rangle - j + u) \mid \forall\ 1 < u \leq 10\}$

$y_j = X = D(\langle X \rangle)$

Yao's Millionaire Problem: Math

Millionaire A

Owns $i = 6 < 10$ *euro*

Holds the pr and pk

$$\langle X \rangle - j + 1,\ \langle X \rangle - j + 2,\ \ldots,\ \langle X \rangle - j + 10$$

$$\mathbf{y} = \{y_u = Dec(\langle X \rangle - j + u) \mid \forall\ 1 < u \leq 10\}$$

Select a prime P,

$$\mathbf{z} = \{z_u = y_u\ mod\ P \mid \forall\ 1 < u \leq 10\}$$

$$z_u = \begin{cases} z_u \mod p & i > u \\ z_u + 1 \mod p & i \leq u \end{cases}$$

The whole idea is $\begin{cases} X\ mod\ P = z_j\ , & i > j \\ X\ mod\ P \neq z_j\ , & i \leq j \end{cases}$

Millionaire B

Owns $j = 4 < 10$ *euro*

Only holds pk

Selects a big number X

$$\langle X \rangle = Enc(X)$$

$$\langle X \rangle - j + 1 = \langle m_1 \rangle$$

$\langle m_1 \rangle$

$\langle z_1 \rangle, \langle z_2 \rangle, \langle z_3 \rangle, \langle z_j \rangle, \langle z_5 \rangle, \langle z_i \rangle, \langle z_7 \rangle, \langle z_8 \rangle, \langle z_9 \rangle, \langle z_{10} \rangle$

P

$\langle m_2 \rangle$

$$\langle m_2 \rangle = (X \mod P) - \langle z_j \rangle$$

Decrypts and $\begin{cases} m_2 = 0, & i > j \\ m_2 \neq 0, & i \leq j \end{cases}$

Yao's Millionaire Problem: Math Example

Millionaire A

Owns $i = 6 < 10$ *euro*

Holds the pr and pk

Adds 0 to 9 to the received number, A gets

$\{\langle 17\rangle, \langle 18\rangle, \langle 19\rangle, \langle 20\rangle, \langle 21\rangle, \langle 22\rangle, \langle 23\rangle, \langle 24\rangle, \langle 25\rangle, \langle 26\rangle\}$

Decrypts the list of numbers and gets

$\mathbf{y} = \{17,18,19,20,21,22,23,24,25,26\}$

Choses prime p=5,

$\mathbf{z} = \{2,3,4,0,1,2,3,4,0,1\}$

Adds 1 to the number whose position is equal to and larger than 6 (mod p),

$\mathbf{z} = \{2,3,4,0,1,3,4,0,1,2\}$

Encrypts $\mathbf{z}$

Decrypts $\langle c\rangle$,

If $c = 0$, *A is richer, otherwise B is richer*

j and X hide each other.

$\langle X\rangle - j + 1 = \langle 17\rangle$

Millionaire B

Owns $j = 4 < 10$ *euro*

Only holds pk

Selects a big number $X = 20$

p

$\langle z_1\rangle, \langle z_2\rangle, \langle z_3\rangle, \langle z_j\rangle, \langle z_5\rangle, \langle z_i\rangle, \langle z_7\rangle, \langle z_8\rangle, \langle z_9\rangle, \langle z_{10}\rangle$

$\langle c\rangle$

"A is richer."

$X \mod p = 0$

$\langle c\rangle = \langle z_j\rangle - (X \bmod p)$

"What !?"

Example 4: Secure Voting System

- Application of additive HE: voting system
- Issue: Verifiability and Integrity

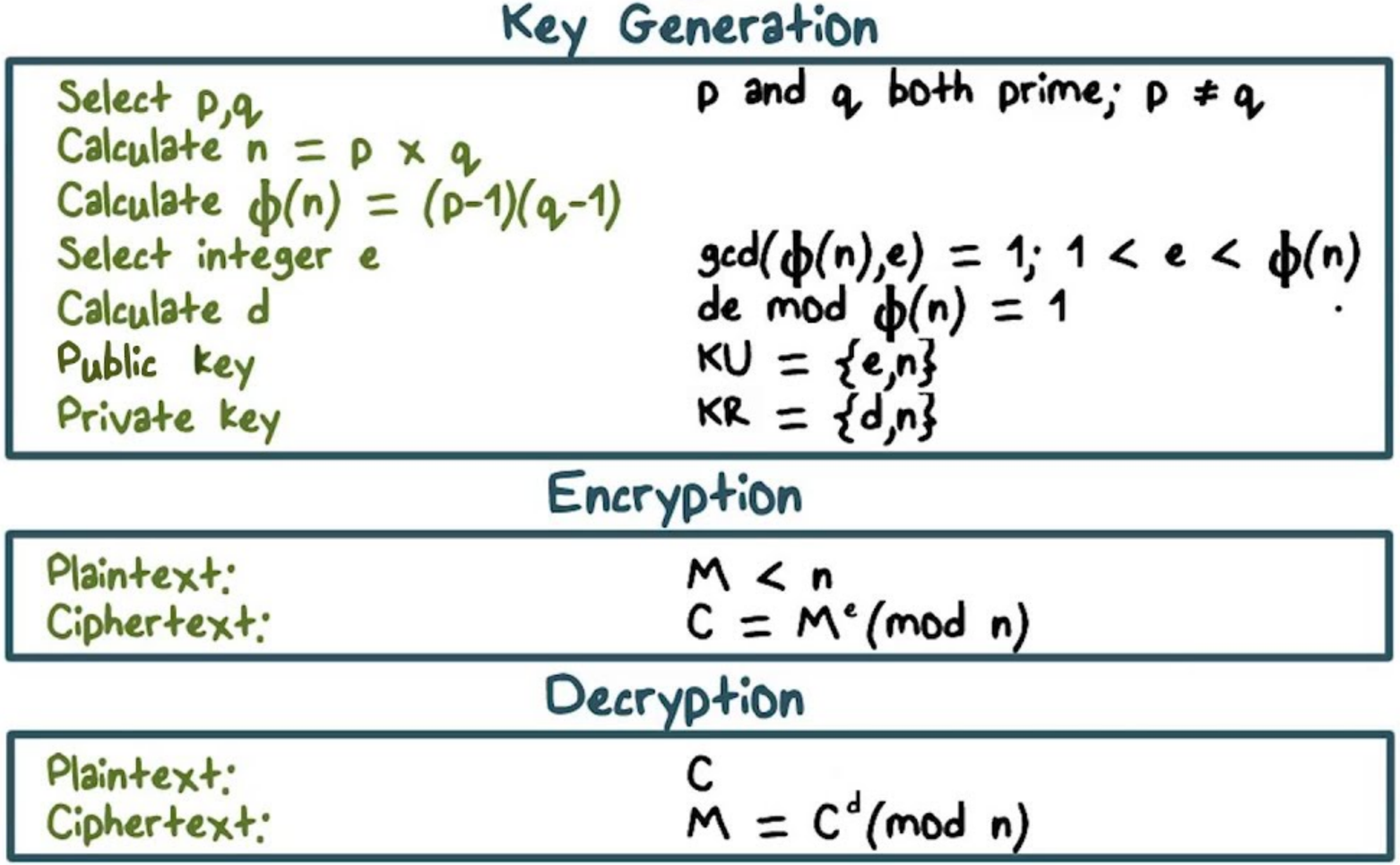
Partially Homomorphic Encryption (PHE)

RSA Basics

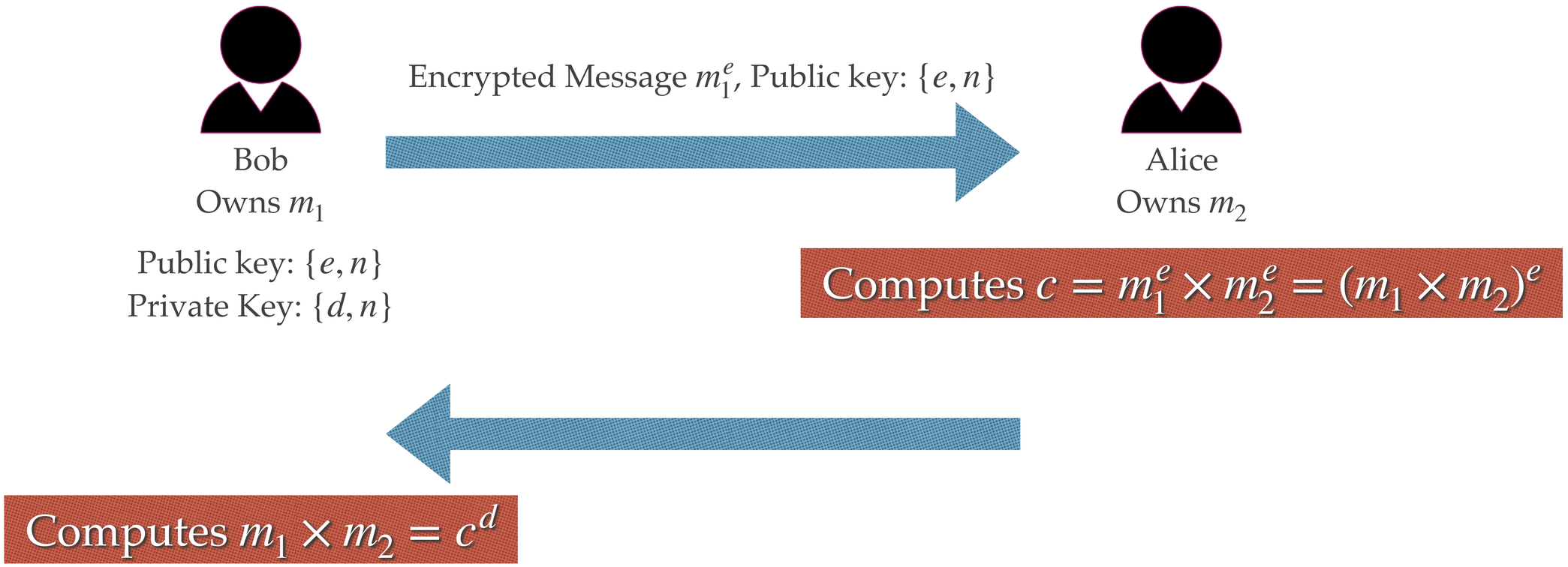
RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem.

The acronym "RSA" comes from the surnames of Ron Rivest, Adi Shamir and Leonard Adleman, who publicly described the algorithm in 1977.

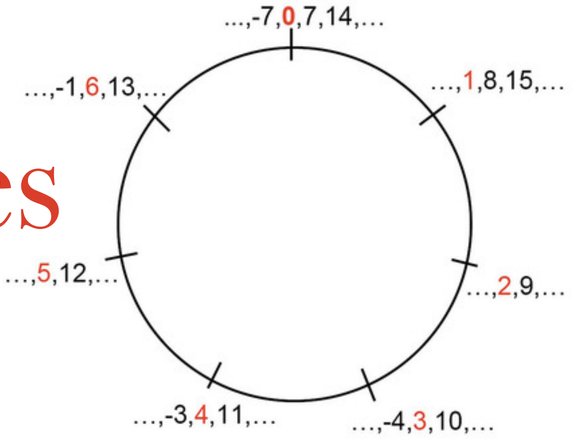
Security Found on the Integer Factorization Problem



RSA Homomorphic Computation Scenario



RSA Homomorphic Properties



Operation	Types	Check
Addition	Ciphertext vs. Ciphertext	✗
	Ciphertext vs. Plaintext	✗
Multiplication	Ciphertext vs. Ciphertext	✓
	Ciphertext vs. Plaintext	✗
Unlimited Times of Operation (d)		✗

Issue: the unpadded RSA is not secure (deterministic). Padding messes up the HE property.

$$\langle m_1 \rangle \times \langle m_2 \rangle = m_1^e \times m_2^e \ mod \ N = (m_1 \times m_2)^e \ mod \ N = \langle m_1 \times m_2 \rangle$$

Multiplicative Homomorphic

$$m_1 \times m_2 \times \ldots \times m_d \in Z_N,\ m_1, m_2, \ldots, m_d \in Z_N,\ Z_N = \{0,1,...,N-1\}$$

Goldwasser-Micali Basics

The Goldwasser–Micali (GM) encryption scheme [7] is the **first probabilistic public-key encryption algorithm** developed by Shafi Goldwasser and Silvio Micali in 1982. GM requires us to encode the message into a string of bits (XOR).

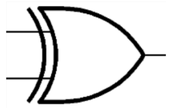
Its security found on quadratic residuosity problem

Bob	Alice
Key creation pk= {a, N} pr= {p, q}	
Choose secret primes p and q. Choose a with $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1$. Publish $N = pq$ and a.	
Encryption	
	Choose plaintext $m \in \{0, 1\}$. Choose random r with $1 < r < N$. Use Bob's public key (N, a) to compute $c = \begin{cases} r^2 \bmod N & \text{if } m = 0, \\ ar^2 \bmod N & \text{if } m = 1. \end{cases}$ Send ciphertext c to Bob.
Decryption	
Compute $\left(\frac{c}{p}\right)$. Decrypt to $m = \begin{cases} 0 & \text{if } \left(\frac{c}{p}\right) = 1, \\ 1 & \text{if } \left(\frac{c}{p}\right) = -1. \end{cases}$	

Boolean Circuit Truth Table

SUM

=

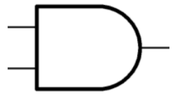


XOR

Inputs		Output
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

PRODUCT

=



AND

Inputs		Output
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

Goldwasser-Micali Homomorphic Computation Scenario

Bob
Owns m_1

Public key: $\{a, n\}$
Private Key: $\{p, q\}$

Encrypted Message $a^{m_1} \times r_1^2$, Public key: $\{a, n\}$

Alice
Owns m_2

Computes $c = a^{m_1} \times r_1^2 \times a^{m_1} \times r_2^2 = (r_1 \times r_2)^2 \times a^{m_1+m_2}$

Computes $m_1 + m_2$ through decryption

Goldwasser-Micali Homomorphic Properties

Operation	Types	Check
Addition	Ciphertext vs. Ciphertext	✓
	Ciphertext vs. Plaintext	✗
Multiplication	Ciphertext vs. Ciphertext	✗
	Ciphertext vs. Plaintext	✗
Unlimited Times of Operation (d)		✗

$$\langle m_1 \rangle \times \langle m_2 \rangle = \langle m_1 + m_2 \rangle$$

Additive Homomorphic

Issue: inefficient (PoC)
Encryption is bit by bit.
For each bit in the plaintext, the ciphertext is one number in Z_n^*, expansion factor is 1024 when using 1024 moduli

Elgamal Basics

The ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie–Hellman key exchange.

It was described by Taher Elgamal in 1985.

Security Found on the Discrete Logarithm Problem

- **Key Generation**
  - Select a large prime as a q
  - Select x *to* be a member of the group **G** = < Zq^*, X >, x must be "$1 \le x \le q - 1$"
  - Select g *to* be a primitive root (generator) in the group **G** = < Zq^*, X >
  - $y = g^x \bmod q$
  - Public key ← (g, y, q)
  - *Private key* ← x
- **Encryption**
  - Select a random integer r in the group **G** = < Zq^*, X >, r must be "$1 \le r \le q - 1$"
  - $C_1 = g^r \bmod q$
  - $C_2 = (p \,.\, y^r) \bmod q$ // p is the plaintext
- **Decryption**
  - $P = [\, C_2 (C_1{}^x)^{-1} \,] \bmod q$

Elgamal Homomorphic Computation Scenario

Bob
Owns m_1

Public key: $\{g, y, q\}$
Private Key: $\{x\}$

Encrypted Message $m_1 \times y^{r_1}$, Public key: $\{g, y, q\}$, Share key g^{r_1}

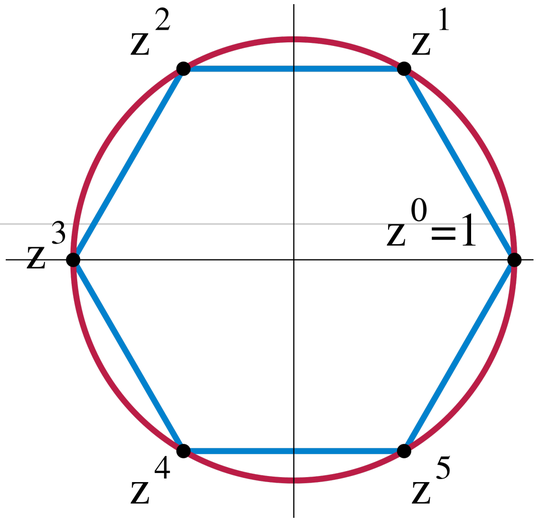
Alice
Owns m_2

Computes $c = m_1 \times y^{r_1} \times m_2 \times y^{r_2} = (m_1 m_2) y^{r_1 + r_2}$

Encrypted Message c, Share key: $g^{r_1 + r_2}$

Computes $m_1 m_2 = c \times ((g^{r_1 + r_2})^x)^{-1}$

Elgamal Basics



Multiplicative Homormorphic

Operation	Types	Check	
Addition	Ciphertext vs. Ciphertext	✓	$\langle g^{m_1} \rangle \times \langle g^{m_2} \rangle = \langle g^{m_1} \times g^{m_2} \rangle = \langle g^{m_1+m_2} \rangle$
	Ciphertext vs. Plaintext	✓	$\langle g^{m_1} \rangle \times g^{m_2} = \langle g^{m_1} \times g^{m_2} \rangle = \langle g^{m_1+m_2} \rangle$
Multiplication	Ciphertext vs. Ciphertext	✓	$\langle m_1 \rangle \times \langle m_2 \rangle = \langle m_1 \times m_2 \rangle$
	Ciphertext vs. Plaintext	✓	$\langle m_1 \rangle \times m_2 = \langle m_1 \times m_2 \rangle$
Unlimited Times of Operation (d)		✗	$m_1 \times m_2 \times \ldots \times m_d \in Z_q^*,\ m_1, m_2, \ldots, m_d \in Z_q^*,\ Z_q^* = \{1,...,N-1\}, q$ is prime.

To decrypt, we need to be able to solve the discrete logarithm.
This can be done efficiently if the value $x+y$ is known to be "not too large".
Is it a false idea?

Paillier Basics

The Paillier cryptosystem, invented by and named after Pascal Paillier in 1999, is a probabilistic asymmetric algorithm for public key cryptography.

Security Found on the Composite Residuosity Problem

Key Generation:

- $p, q \in P$ with equal length
- $n = pq$
- $g = 1 + n$
- $\phi(n) = (p-1) \cdot (q-1)$
- $\mu = \phi(n)^{-1} \; mod \; n$ (μ is used as *private-key*)

Encryption:

- Plaintext $m < n$ ($m \in Z_n$)
- Choose $r < n$ $\quad \gcd(r, n) = 1$ randomly ($r \in Z_n^*$)
- Ciphertext $c = g^m . r^n \; mod \; n^2$

Decryption:

- Ciphertext $c < n^2$
- Plaintext $m = L\left(c^\lambda \; mod \; n^2\right) \cdot \mu \; mod \; n$

Paillier Scenario

Bob

Owns m_1

Public key: $\{g, n\}$

Private Key: $\{\lambda, \mu\}$

Encrypted Message $g^{m_1} \times r_1{}^n$, Public key: $\{g, n\}$

Alice

Owns m_2

Computes $c = g^{m_1} \times r_1{}^n \times g^{m_2} \times r_2{}^n = g^{m_1+m_2}(r_1 r_2{}^n)$

Encrypted Message c, Share key: $g^{r_1+r_2}$

Computes $m_1 + m_2$ through the decryption function

Paillier Homomorphic Properties

Operation	Types	Check	
Addition	Ciphertext vs. Ciphertext	✓	$\langle m_1 \rangle \times \langle m_2 \rangle = \langle m_1 + m_2 \rangle$
	Ciphertext vs. Plaintext	✓	$\langle m_1 \rangle \times m_2 = \langle m_1 + m_2 \rangle$
Multiplication	Ciphertext vs. Ciphertext	✗	
	Ciphertext vs. Plaintext	✓	$\langle m_1 \rangle^{m_2} = \langle m_1 \times m_2 \rangle$
Unlimited Times of Operation (d)		✗	$m_1 \times m_2 \times \ldots \times m_d \in Z_N,\ m_1, m_2, \ldots, m_d \in Z_N,\ Z_N = \{0,1,...,N-1\}$

Additive Homomorphic

Somewhat & Fully Homomorphic Encryption (SHE & FHE)

FHE Intro

- Origin: The concept of FHE was introduced by Rivest [14] under the name privacy homomorphisms.
- Boneh-Goh-Nissim Algorithm (2005): One multiplications, many additions
- Gentry FHE (2009): Construction from lattices, turns SHE into FHE through the bootsrapping method.
- Dijk, Gentry, Halevi & Vaikuntanathan Integer Algorithm (2010)
- Smart–Vercauteren scheme (2010)

Ring Definition

- Fully homomorphic encryption can be considered as ring homomorphism.
- A ring is a set R equipped with operations $+$ and $\times$ satisfying the following ring axioms.
- We use $\circ = \{ +, \times \}$ to denote axioms for both operations $+$ and $\times$.

Ring Definition

- Associativity: $\forall a, b, c \in R \Rightarrow (a \circ b) \circ c = a \circ (b \circ c)$
- Identity element: $\exists! e \in R, \forall a \in R \Rightarrow e \circ a = a \circ e = a$ (multiplicative $e = 1$ & additive $e = 0$)
- Inverse element: $a \in R, \exists b \in R, a + b = b + a = e$, where $e = 0$ is the identity element.
- Commutativity (Abelian Group): $\forall a \in R, \forall b \in R, a + b = b + a$
- Distributivity: $\forall a, b, c \in R, a \times (b + c) = a \times b + a \times c$ (left distributivity) $(a + b) \times c = a \times c + b \times c$ (right distributivity)

R is an abelian group under addition

R is a monoid under multiplication

Ring homomorphism

- A ring homomorphism is a function between two rings which respects the structure. More explicitly, if R and S are two rings, then a ring homomorphism is a function $f : R \to S$ such that $\forall a, b \in R$, $f(a + b) = f(a) + f(b), f(a \times b) = f(a) \times f(b)$

- Given two rings $(P, +_p , \times_p)$ and $(C, +_c , \times_c)$, where P, C are the plaintext and ciphertext spaces, a encryption function $E : P \to C$ such that for all g and g' in P, it holds that $\langle g \ +_p \ g' \rangle = \langle g \rangle \ +_c \ \langle g' \rangle$ and $\langle g \ \times_p \ g' \rangle = \langle g \rangle \ \times_c \ \langle g' \rangle$

A Roadmap [6]

Secret-key SHE

Under the approximate GCD assumption

Public-key SHE

Under the approximate GCD assumption

Public-key FHE

Under the approximate GCD assumption + sparse subset sum

Integer-Based Secret Key Scheme

- Secret Key Generation:
  - a large odd number $p \in \{2^{\eta-1}, \ldots, 2^{\eta}\}$
- Encryption: For a binary message $m \in \{0,1\}$, select large integers q, r $(r < p/2)$.

  CipherBit=Key∗RandomIntegerA+2∗RandomIntegerB+PlainTextBit
  - $c = pq + 2r + m$
- Decryption:
  - $m = (c \ (mod \ p))(\ mod \ 2) = (pq + 2r + m \ (mod \ p))(\ mod \ 2) = (2r + m)(\ mod \ 2)$

Homomorphism

- $e_1 = 2 \times r_1$
- $e_2 = 2 \times r_2$
- $C_1 = Enc_p(m_1) = pq_1 + m_1 + e_1$
- $C_2 = Enc_p(m_2) = pq_2 + m_2 + e_2$
- $C_1 + C_2 = p(q_1 + q_2) + (m_1 + m_2) + (e_1 + e_2)$
- $C_1 \times C_2 = ?$

Questions

- $C_0 \times C_1 = p(\text{a bunch of stuff}) + (m_0 m_1) + (e_0 e_1 + e_0 m_1 + e_1 m_0)$

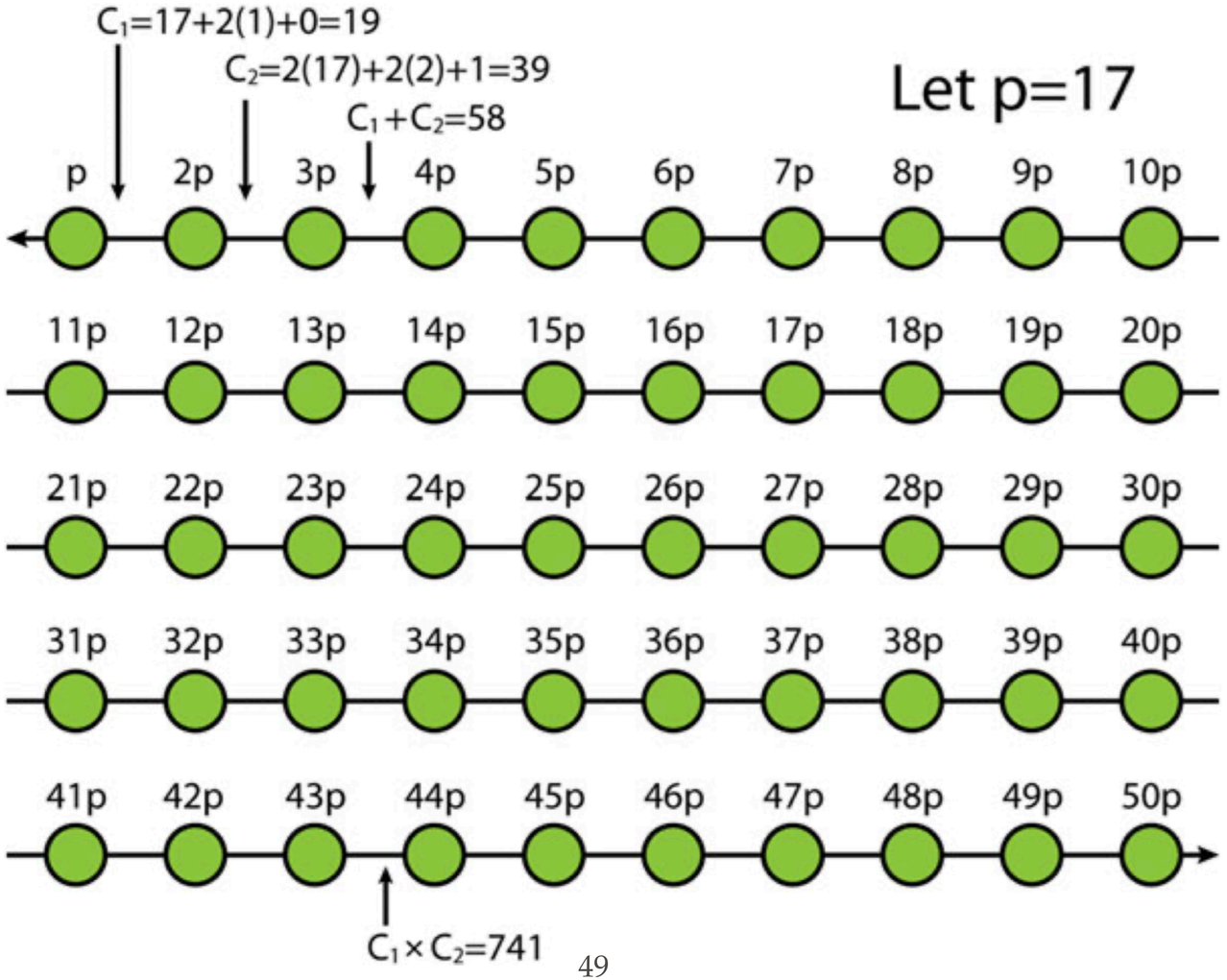
Integer-Based Secret Key Scheme Properties

Operation	Types	Check	
Addition	Ciphertext vs. Ciphertext	✓	$c_1 + c_2 = (q_1 + q_2)p + 2(r_1 + r_2) + (m_1 + m_2)$
	Ciphertext vs. Plaintext	✓	$c_1 + m_2 = q_1 p + 2r_1 + (m_1 + m_2)$
Multiplication	Ciphertext vs. Ciphertext	✓	$c_1 \times c_2 = (pq_1q_2 + 2q_1r_2 + 2q_2r_1 + m_1q_2 + m_2q_1)p + 2(2r_1r_2 + m_1r_2 + m_2r_1) + m_1m_2$
	Ciphertext vs. Plaintext	✓	$c_1 \times m_2 = q_1 p + 2r_1 + m_1m_2$
Unlimited Times of Operation (d)		✗	When $r_1 + r_2 < p/2$ or $2r_1r_2 + m_1r_2 + m_2r_1 < p/2$

Integer-Based Secret Key Scheme Noise

- First, the size of the ciphertext itself grows (the length roughly doubles when you multiply). Second, the noise grows too large when $r_1 + r_2 > p/2$ or $2r_1r_2 + m_1r_2 + m_2r_1 > p/2$.
- If we choose $r \approx 2^n, p \approx 2^{n^2}$, *and* $q \approx 2^{n^5}$ the somewhat encryption scheme can compute polynomials of degree n before the noise grows too large.

Integer-Based Secret Key Scheme Example



A Roadmap [6]

Under the approximate GCD assumption

Under the approximate GCD assumption

Under the approximate GCD assumption + sparse subset sum

SHE 2: Integer-Based Public Key Scheme

The scheme has many parameters, controlling the number of integers in the public key and the bit-length of the various integers.

- υ : is the bit-length of the integers in the public key;
- η : is the bit-length of the secret key (which is the hidden approximate GCD of all the public-key integers);
- ρ: is the bit-length of the noise (i.e., the distance between the public-key elements and the nearest multiples of the secret key).
- τ : is the number of integers in the public key.

Integer-Based Public Key Scheme

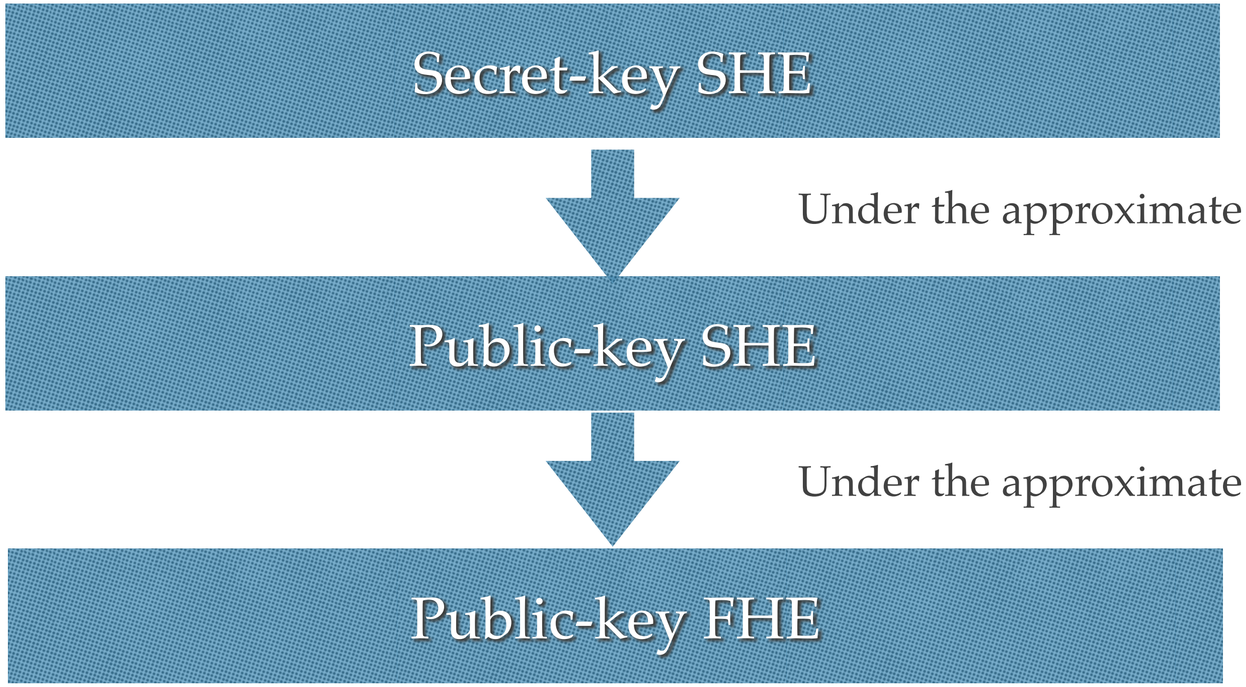
- Key generation $KeyGen(\lambda)$:
  - **Private key:** a random η-bit odd integer p.
  - **Public key:** $x_i = pq_i + r_i$, where $q_i \in \mathbb{Z} \cap \{0,...,(2^\gamma/p) - 1\}$, $r_i \in \mathbb{Z} \cap \{2^\rho + 1,...,2^\rho - 1\}$ are chosen randomly, for $i = 0,1,...,\tau$.
    - Note: relabel so that x_0 is the largest. Restart unless x_0 is odd and $x_0(\ mod\ p)$ is even. The public key is $pk = [x_0, x_1, \ldots, x_\tau]$
- Encrypt $Enc(pk, m)$:
  - Given $m \in \{0,1\}$, and public key pk, choose a random subset $S \subseteq \{1,2,...,\tau\}$ and a random integer $r_i \in \mathbb{Z} \cap \{2^{\rho'} + 1,...,2^{\rho'} - 1\}$
    - $c = (m + 2r + 2\sum_{i \in S} x_i)(mod\ x_0)$
- Decryption $Decrypt(sk, c)$:
  - $m = (c(mod\ p))(mod\ 2)$

Integer-Based Public Key Scheme Example

- Private key $P = 10001$
- $[q_0, q_1, q_2, q_3] = [36,27,34,6]$
- $[r_0, r_1, r_2, r_3] = [8,5,4,2]$
- Public Key $[x_0, x_1, x_2, x_3] = [360044,270032,340038,60008]$
- We now encrypt two messages $m_1 = 0$, $m_2 = 1$ using a random subset of the public key $S = \{1,3\}$
- $c_1 \equiv$
- $c_2 =$

- $c_1 + c_2 =$
- $c_1 \times c_2 =$

A Roadmap [6]


Secret-key SHE
Under the approximate GCD assumption
Public-key SHE
Under the approximate GCD assumption
Public-key FHE
Under the approximate GCD assumption + sparse subset sum

From SHE to FHE

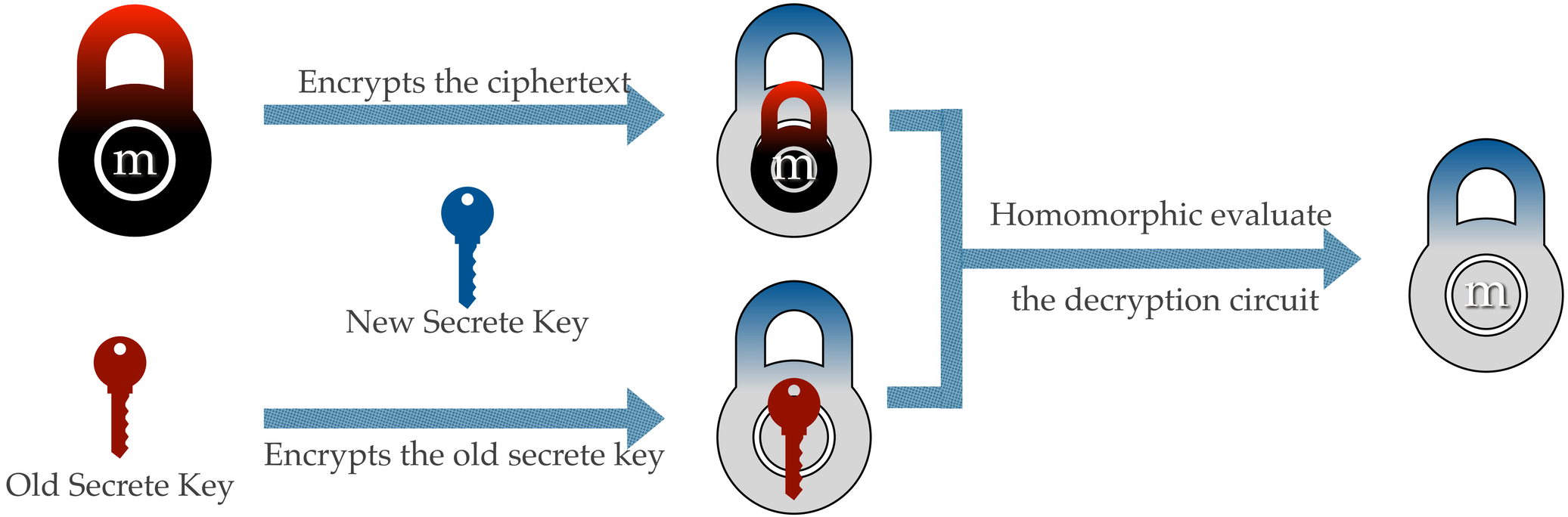
There are two classes of solutions to this problem.

First, in many somewhat homomorphic encryption schemes, there are clever tricks to make multiplication only increase the error by a constant factor (eg. 1000x) instead of squaring it.

Second, there is Craig Gentry's technique of "bootstrapping".

Noise Growing [7]

Bootstrapping (Recrypt)



Issues

- Efficiency
- Machine-in-the-middle attack (Change the content of the message)
- Comparison operation
- Noise (SHE, FHE)
- Message Encoding (negative, floating point number)

The story so far… [7]

- Simple FHE schemes exist.
- But…bootstrapping is expensive!
  - At 76 bits of security: each bootstrapping operation requires 320 seconds and 3.4 GB of memory [8]
  - Other implementations exist but are generally less flexible/efficient
- SHE (without bootstrapping) is closer to practical: can evaluate shallow circuits.

Applications in ML

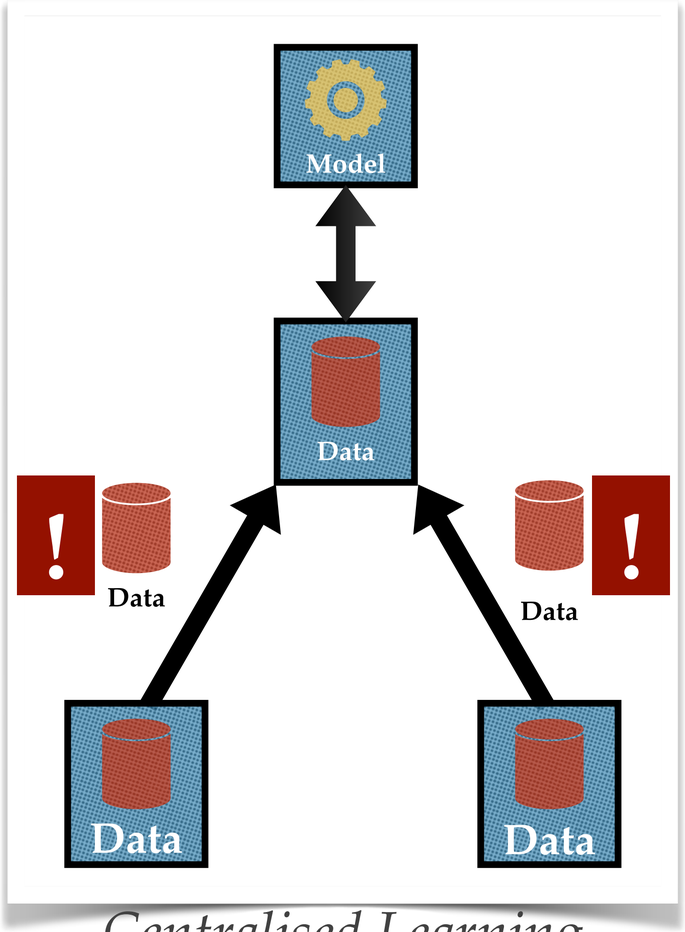
HE in PPML: Privacy-Preserving Machine Learning

The model is available but not visible

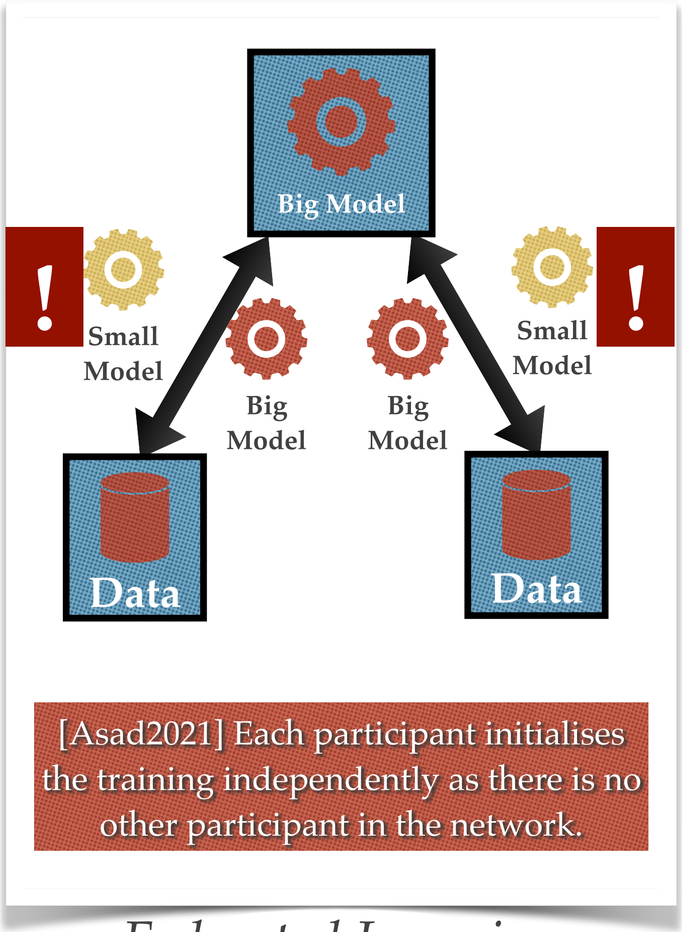
Source: https://ec.europa.eu/jrc/en/news/harnessing-new-data-sources-responsibly-effective-migration-policy

Data is available but not “visible”

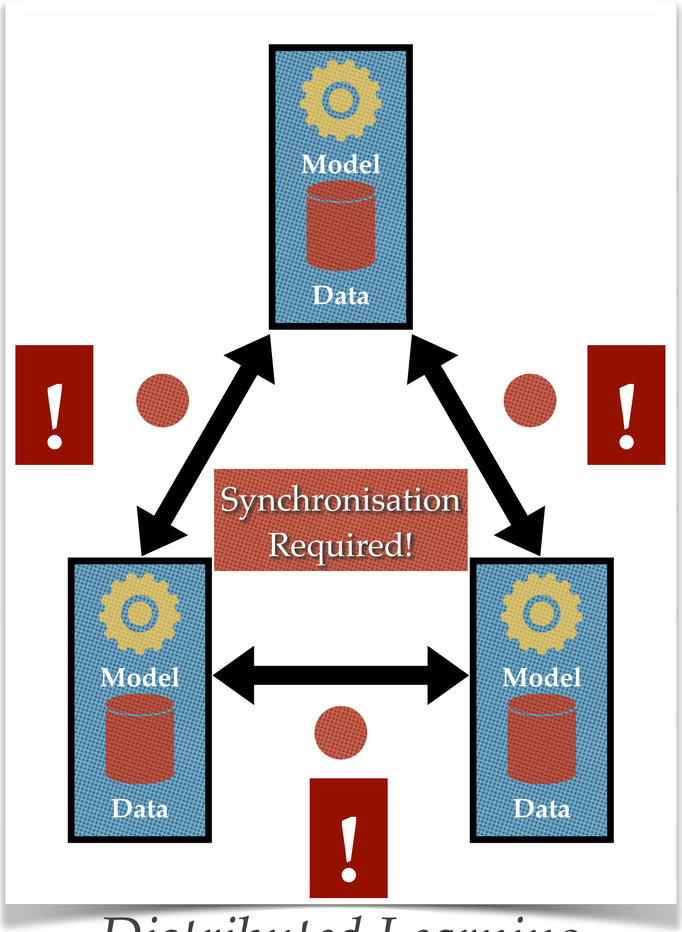
Model Training Methods



Centralised Learning (Data Aggregation)

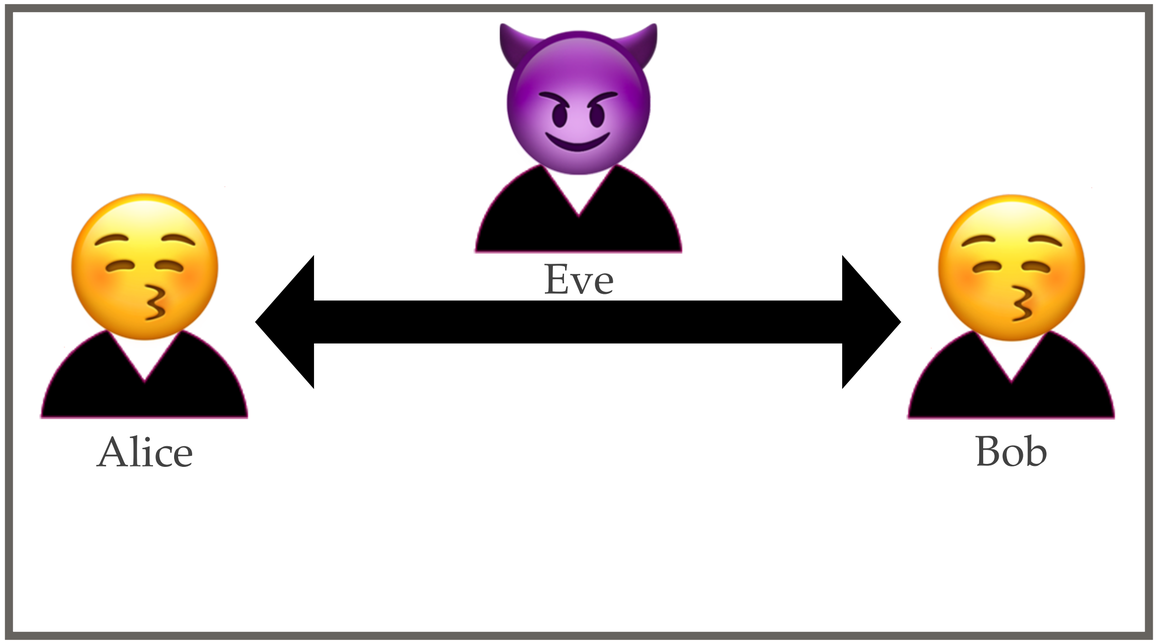


Federated Learning (Model Aggregation)

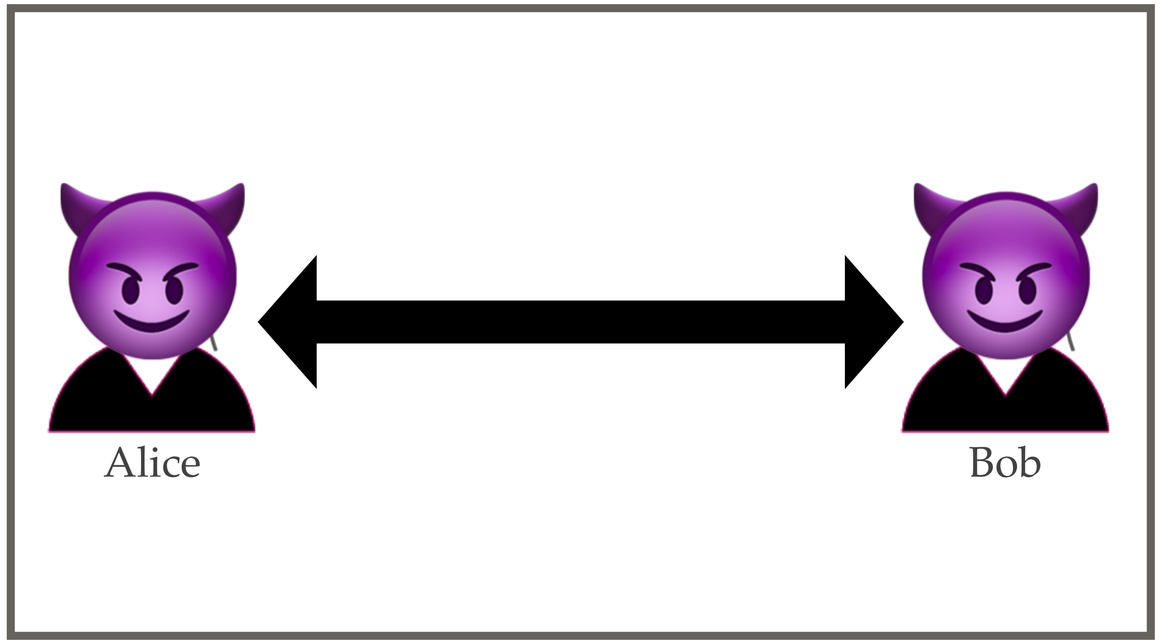


Distributed Learning (Multi-nodal system)

Adversary Model Property



External Threats



Internal Threats

K Malicious Users

Adversary Model (Other Properties) [Wagner2018]

- Internal vs. External
- Local vs. Global
- Prior Knowledge

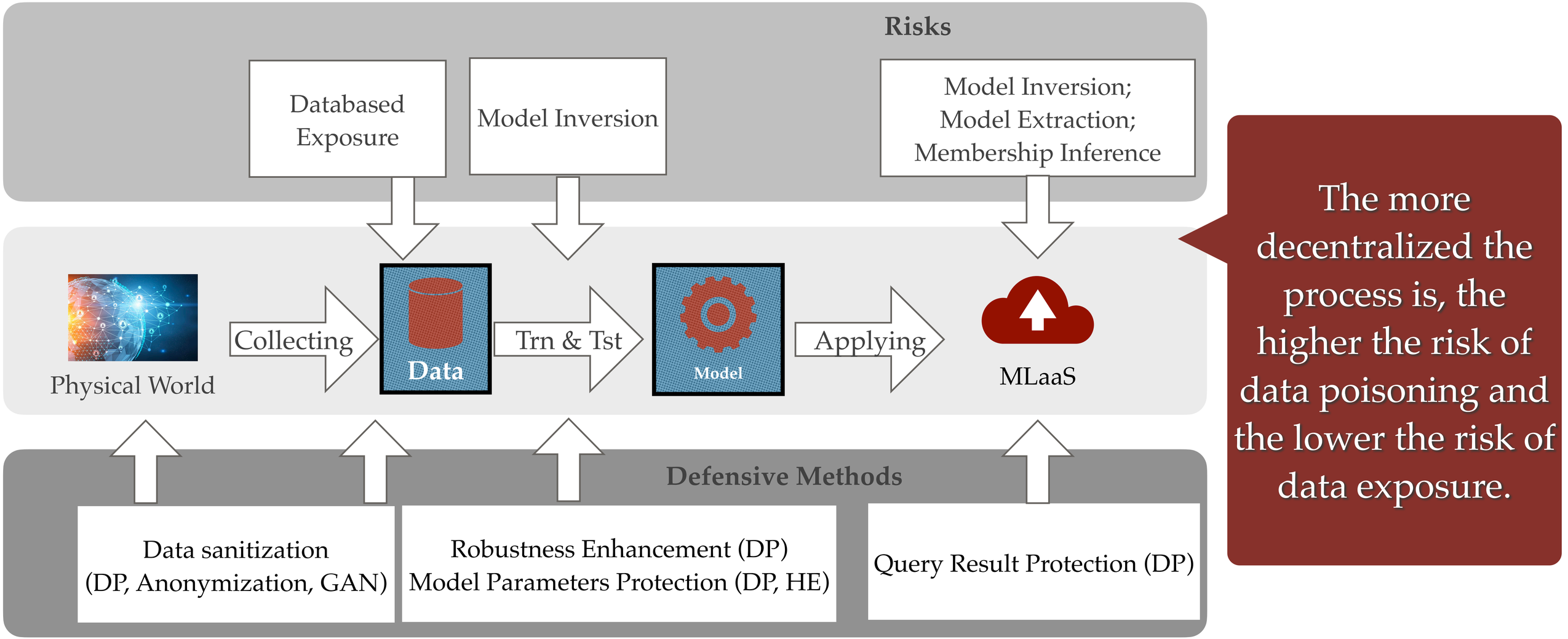
- Active vs. Passive (i.e., Honest-but-Curious)
- Static vs. Adaptive
- Resources (e.g., restricted or unlimited)
- Number of Adversaries

Established

Assumed

We need PETs to guarantee the privacy and security of the established conditions, and adjust the strength according to strength of the assumed conditions.

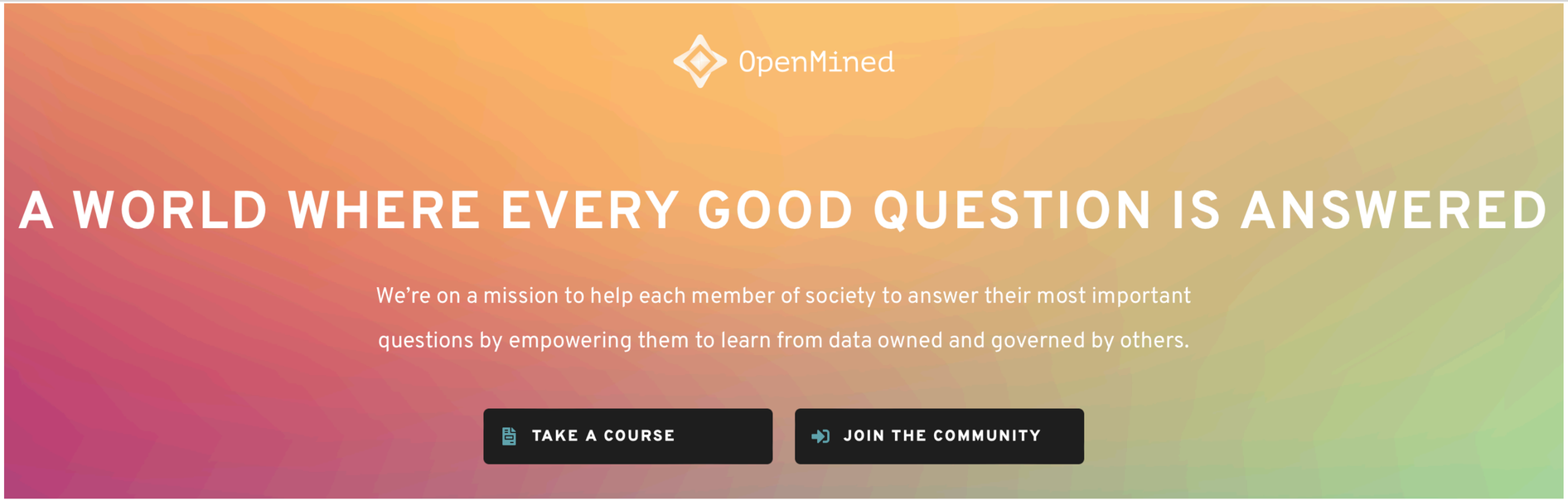
Core Privacy Risks and Defensive Techniques



In sum, using HE in ML, we can

- Encrypt the ML model before sharing (achieve model privacy)
- Encrypt the data before sharing (achieve data privacy)

Community



https://www.openmined.org

- [Asad2021] Asad, M., Moustafa, A., & Ito, T. (2021). Federated Learning Versus Classical Machine Learning: A Convergence Comparison. arXiv preprint arXiv:2107.10976.
- [Wagner2018] Wagner, I., & Eckhoff, D. (2018). Technical privacy metrics: a systematic survey. *ACM Computing Surveys (CSUR)*, *51*(3), 1-38.

Reference

- [1] https://www.iacr.org/conferences/crypto2011/slides/Halevi.pdf
- [2] Yi, X., Paulet, R., & Bertino, E. (2014). Homomorphic encryption. In *Homomorphic encryption and applications* (pp. 27-46). Springer, Cham.
- [3] https://vitalik.ca/general/2020/07/20/homomorphic.html
- [4]Dijk, M. V., Gentry, C., Halevi, S., & Vaikuntanathan, V. (2010, May). Fully homomorphic encryption over the integers. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 24-43). Springer, Berlin, Heidelberg.
- [5] Ízdemir, F., & Ídemiş Ízger, Z. (2021). Boneh-Goh-Nissim Algorithm. In *Partially Homomorphic Encryption* (pp. 123-133). Springer, Cham.
- [6]https://www.youtube.com/watch?v=Hl5lMDydMvs
- [7] https://www.youtube.com/watch?v=hK9ktrC9dSw&t=2135s
- [8] Jung, W., Kim, S., Ahn, J. H., Cheon, J. H., & Lee, Y. (2021). Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with GPUs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 114-148.
- [9] Liu, Q., Li, P., Zhao, W., Cai, W., Yu, S., & Leung, V. C. (2018). A survey on security threats and defensive techniques of machine learning: A data driven view. *IEEE access*, *6*, 12103-12117.