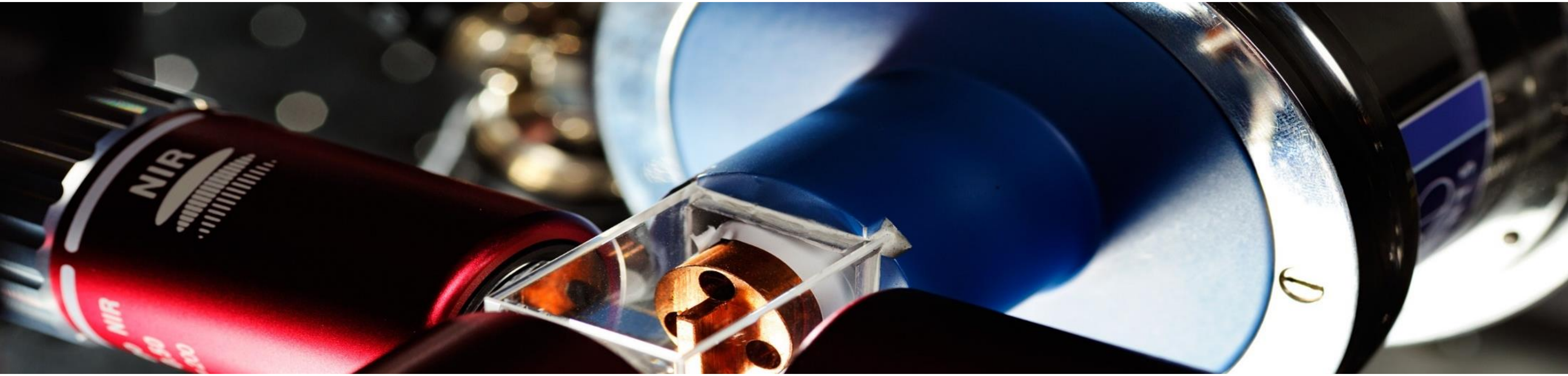


Secure Multi-Party Computation

- Mina Sheikhalishahi

m.Sheikhalishahi@tue.nl

Scenario 1: Love Game

- ❖ Alice and Bob meet at a pub
- ❖ If both want to date , they will find out
- ❖ If Alice doesn't want to date, she won't learn his intention
- ❖ If Bob doesn't want to date, he won't learn her intention
- One solution: a trusted person

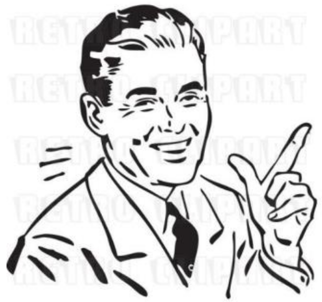
Scenario 2: Finding Potential Terrorists

Intelligence agencies holds lists of potential terrorists

❖ They like to compute the intersection

❖ Any other information must remain secret

One solution: use a trusted party



MI5 Mossad FBI

Scenario 3: Face Recognition

- ❖ Bob owns a list of face images, and Alice owning one face image is interested in knowing whether this image belongs to Bob list or not.
- ❖ The output will be 1 (or ID of corresponding match), or zero (if no match).
- one solution: use a trusted party.

Scenario 4: Auction

- ❖ Several parties wish to execute a private auction.
- ❖ The highest bid win.
- ❖ Only the amount of highest bid (and bidder) is revealed.
- One solution: use trusted auctioneer

Scenario 5: Distributed Data Clustering

- ❖ Two (or more) agents are willing to construct a data clustering algorithm on whole of their data, without revealing their datasets.
- ❖ All agents obtain the structure of clustering algorithm built on whole data, but not the inputs of others.
- One solution: use trusted party.

Secure Multiparty Computation

- ❖ In all scenarios the solution of an external trusted third party works.
- ❖ Trusting a third party is a very strong assumption.
- ❖ Can we do better?
- ❖ We would like a solution with the same security guarantees, but without using any trusted party.

Goal: use a protocol which works without the presence of trusted party

FBI

MI5

Mossad

Setting

- ❖ Party Pi (1 ≤ i ≤ n) has private input xi.
- ❖ The parties wish to jointly compute a (known) function $y = f\ (x1, \dots, xn)$.
- ❖ The computation must preserve certain security properties, even is some of the parties collude and maliciously attack the protocol.
- ❖ This is generally modeled by an external adversary $\mathcal{A}$ that corrupts some parties and coordinates their actions.

Security Requirements

- **Correctness**: parties obtain correct output .
- **Privacy**: only the output is learned.
- **Independence of inputs**: parties cannot choose their inputs as a function of other parties' inputs.
- **Fairness**: if one party learns the output, then all parties learn the output.
- **Guaranteed output delivery**: all honest parties learn the output.

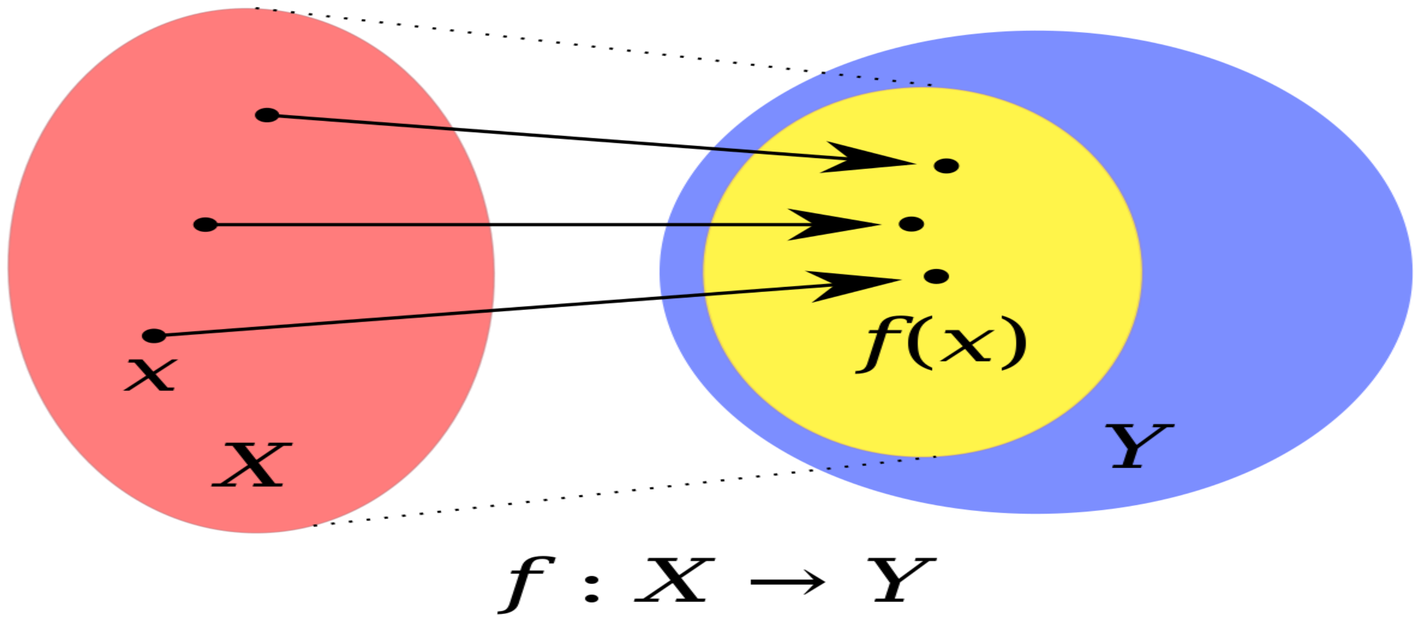
The Security Requirements of Auction Example

- **Correctness**: $\mathcal{A}$ can’t win using lower bid than the highest.
- **Privacy**: $\mathcal{A}$ learns the highest bid out of all inputs, nothing else.
- **Independence of inputs**: $\mathcal{A}$ can’t bid one dollar more than the highest (honest) bid.
- **Fairness**: $\mathcal{A}$ can’t abort the auction if his bid isn’t the highest (i.e., after learning the result).
- **Guaranteed output delivery**: $\mathcal{A}$ can’t abort (stronger than fairness, no DoS attacks).

Attack Scenarios

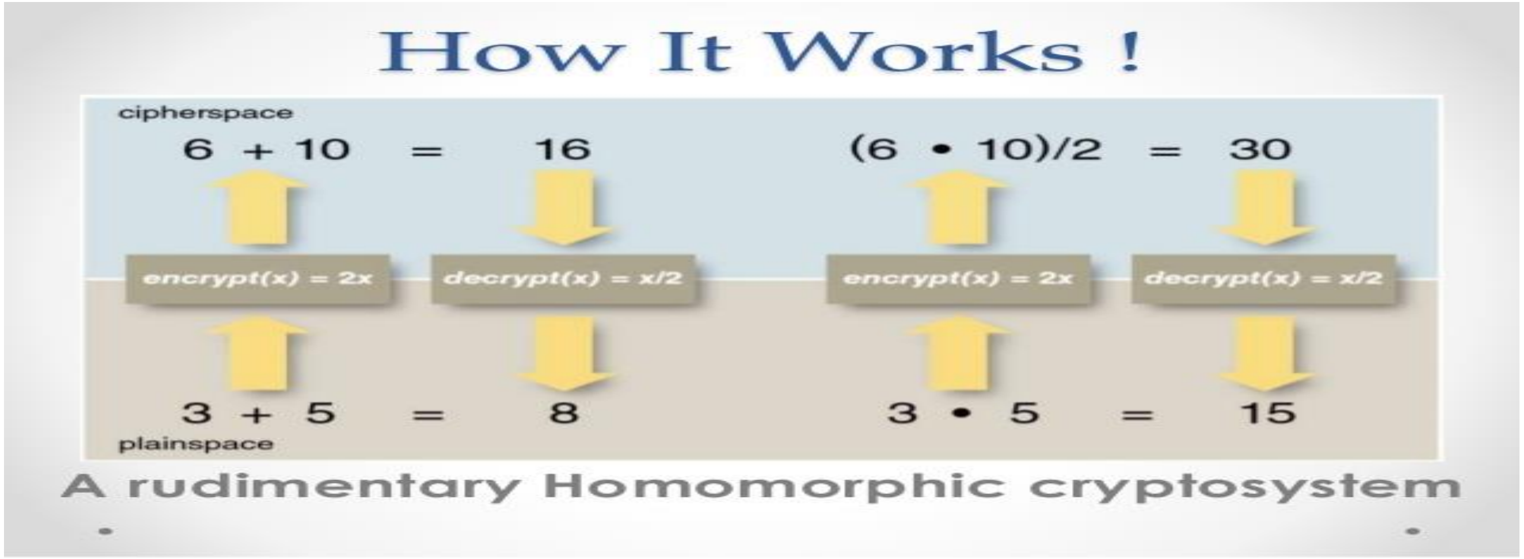
- **Chosen-Plaintext Attack (CPA)**: In this attack, the adversary has the ability to obtain the encryption of any plaintext(s) of its choice. It then attempts to determine the plaintext that was encrypted to give some other ciphertext.
- **Chosen-Ciphertext Attack (CCA)**: The other type of attack is one where the adversary is even given the capability to obtain the decryption of any ciphertext(s) of its choice. The adversary's aim, once again, is then to determine the plaintext that was encrypted to give some other ciphertext (whose decryption the adversary is unable to obtain directly).

Homomorphic Encryption



Homomorphic Encryption

Homomorphic encryption is a form of encryption that allows computation on ciphertexts, generating an encrypted result that when decrypted it matches the result of the operations as if they had been performed on the plaintext.



Number Theory

For positive integer n:

$\mathbb{Z}_n =$ “set of integers modulo n”

$\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n : \gcd(x, n) = 1\}$ “integers with multiplicative inverse modulo n”

$\phi(n) = |\mathbb{Z}_n^*| = n \prod_{p|n}(1 - 1/p)$ “Euler’s phi function”

Hence, $\phi(n)$ can be computed efficiently given the prime divisors of n.

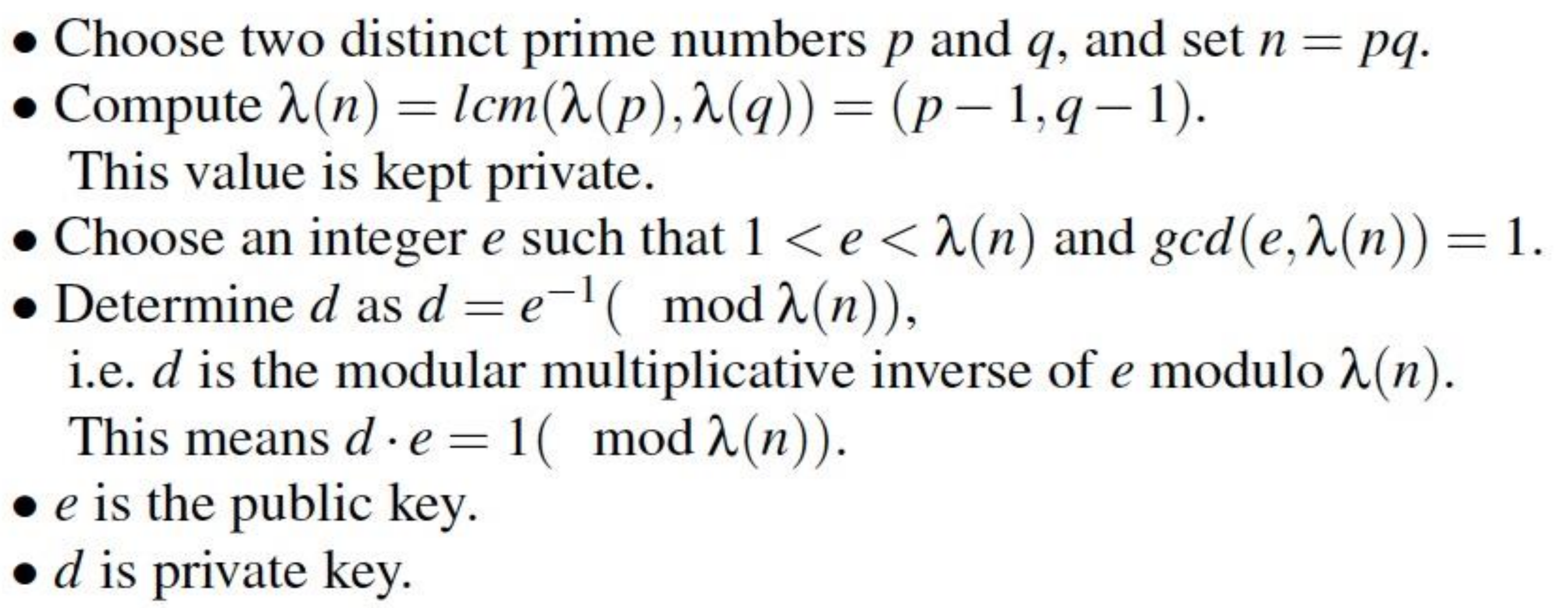
RSA: Multiplicative Homomorphism

RSA is a public key in modulus m, where the encryption of a message x is given by

$$\mathscr{E}(x) = x^e \mod m.$$

$$\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = x_1^e x_2^e \mod m = (x_1 x_2)^e \mod m = \mathcal{E}(x_1 \cdot x_2)$$

Key Generation:



- Choose two distinct prime numbers p and q, and set $n = pq$.
- Compute $\lambda(n) = lcm(\lambda(p), \lambda(q)) = (p-1, q-1)$.
  This value is kept private.
- Choose an integer e such that $1 < e < \lambda(n)$ and $gcd(e, \lambda(n)) = 1$.
- Determine d as $d = e^{-1}(\mod \lambda(n))$,
  i.e. d is the modular multiplicative inverse of e modulo $\lambda(n)$.
  This means $d \cdot e = 1(\mod \lambda(n))$.
- e is the public key.
- d is private key.

Decryption

Given the encrypted message m^e, the owner of private key d can decrypt the message by computing:

$$(m^e)^d = m^{e \cdot d} = m \pmod{n}$$

Security of RSA

- CPA: RSA encryption is deterministic encryption algorithm (i.e. has no random component). Hence, an attacker can successfully launch a chosen plaintext attack by encrypting likely plaintexts under the public key and test if they are equal to the ciphertext.

- CCA: Because of multiplicative property of RSA, it is not secure against CCA attack as well.

ElGamal: Multiplicative Homomorphism

G is cyclic group of order q with generator g. (G, q, g, h) is the public key, where $h = g^x$, and x is the secret key.
The encryption of a message m is $\mathcal{E}(m) = (g^r, m \cdot h^r)$ for some random $r \in \{0, , q-1\}$, with the following homomorphic property:

$$\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) = (g^{r_1}, m_1 \cdot h^{r_1})(g^{r_2}, m_2 \cdot h^{r_2})$$

$$= (g^{r_1+r_2}, (m_1 \cdot m_2)h^{r_1+r_2}) = \mathcal{E}(m_1 \cdot m_2).$$

Decryption

Given an encrypted message m as $\mathcal{E}(m) = (g^r, m \cdot h^r) = (c1, c2)$, it is decrypted as the following:

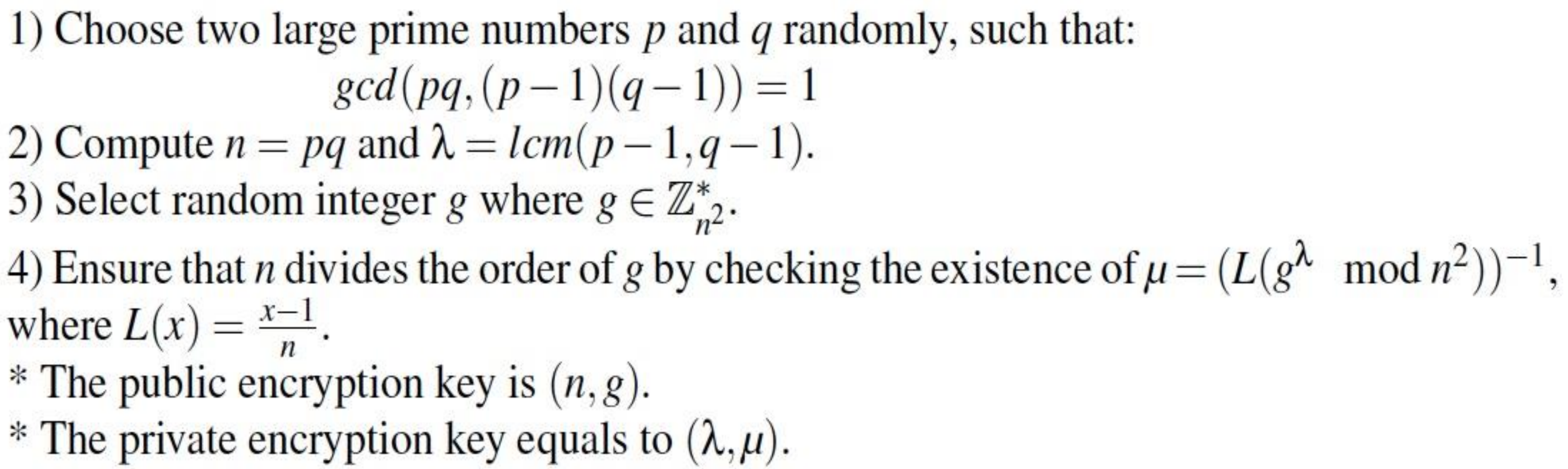
$$\frac{c_2}{c_1^x} = \frac{m \cdot (g^x)^r}{(g^r)^x} = m$$

Security of ElGamal

- CPA: Elgamal encryption scheme provides semantic security against chosen-plaintext attack resulting from random component.

- CCA: Because of multiplicative property of ElGamal, it is not secure against CCA attack. For example, given an encryption $(c1, c2)$ of some message m, one can easily construct a valid encryption $(c1, 2c2)$ of message $2m$.

Paillier: Additive Homomorphism



1) Choose two large prime numbers p and q randomly, such that:

$$gcd(pq,(p-1)(q-1)) = 1$$

2) Compute $n = pq$ and $\lambda = lcm(p-1, q-1)$.

3) Select random integer g where $g \in \mathbb{Z}^*_{n^2}$.

4) Ensure that n divides the order of g by checking the existence of $\mu = (L(g^\lambda \mod n^2))^{-1}$, where $L(x) = \frac{x-1}{n}$.

* The public encryption key is (n, g).

* The private encryption key equals to (λ, μ).

Encryption

Let m be a message to be encrypted where $0 \leq m \leq n$.
Select random r where $0 < r < n$, and $r \in Z^*_{n^2}$ (i.e. $gcd(r,n) = 1$).
Encrypt m as : $c = g^m \cdot r^n \mod n^2$.

Decryption
Number Theory

Carmichael's Theorem:

Let $n = pq$ where p and q are large numbers, $\phi(n)$ is Euler's totient function and $\lambda(n) = lcm(p-1, q-1)$. For any $w \in \mathbb{Z}^*_{n^2}$,

$$w^{\lambda} = 1 \bmod n$$

$$w^{n\lambda} = 1 \bmod n^2 \; .$$

Paillier Encryption Scheme
Key generation and Encryption

$$m \in \mathbb{Z}_n \text{ and } r \in_R \mathbb{Z}_n^*$$
$$E_{pk}(m) = g^m r^n \bmod n^2$$

How to find g?

$$\begin{aligned}
(1+n) &\equiv 1+n \bmod n^2\\
(1+n)^2 &= 1+2n+n^2 \equiv 1+2n \bmod n^2\\
(1+n)^3 &= 1+3n+3n^2+n^3 \equiv 1+3n \bmod n^2\\
\ldots(1+n)^k &\equiv 1+kn \bmod n^2\\
\text{then}(1+n)^n &= 1+n\cdot n \equiv 1 \bmod n^2\\
g &= 1+n
\end{aligned}$$

Decryption

$$D_{sk}(c) = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n \text{ where } L(u) = \frac{u-1}{n}$$

$$\begin{aligned} c^\lambda \bmod n^2 &= (g^m r^n)^\lambda = g^{m\lambda} r^{n\lambda} \bmod n^2 \\ &= (1+n)^{m\lambda} \bmod n^2 = 1 + nm\lambda \end{aligned}$$

$$g^\lambda \bmod n^2 = (1+n)^\lambda = 1 + n\lambda$$

$$\frac{L(1+nm\lambda)}{L(1+n\lambda)} = \frac{m\lambda}{\lambda} \bmod n \equiv m \bmod n$$

Homomorphic Properties

$$\mathscr{E}(m_1,r_1)\cdot\mathscr{E}(m_2,r_2) = g^{m_1}\cdot {r_1}^n \cdot g^{m_2}\cdot {r_2}^n \pmod{n^2}$$

$$= g^{m_1+m_2}\cdot (r_1r_2)^n \pmod{n^2} = \mathscr{E}(m_1+m_2, r_1r_2)$$

$$\mathscr{E}(m_1,r)\cdot g^{m_2} = g^{m_1}\cdot {r_1}^n\cdot g^{m_2} \pmod{n^2} = g^{m_1+m_2}\cdot {r_1}^n \pmod{n^2} = \mathscr{E}(m_1+m_2, r_1)$$

$$(\mathscr{E}(m_1,r_1))^k = (g^{m_1}\cdot {r_1}^n)^k \pmod{n^2} = g^{km_1}\cdot ({r_1}^k)^n \pmod{n^2} = \mathscr{E}(km_1, {r_1}^k)$$

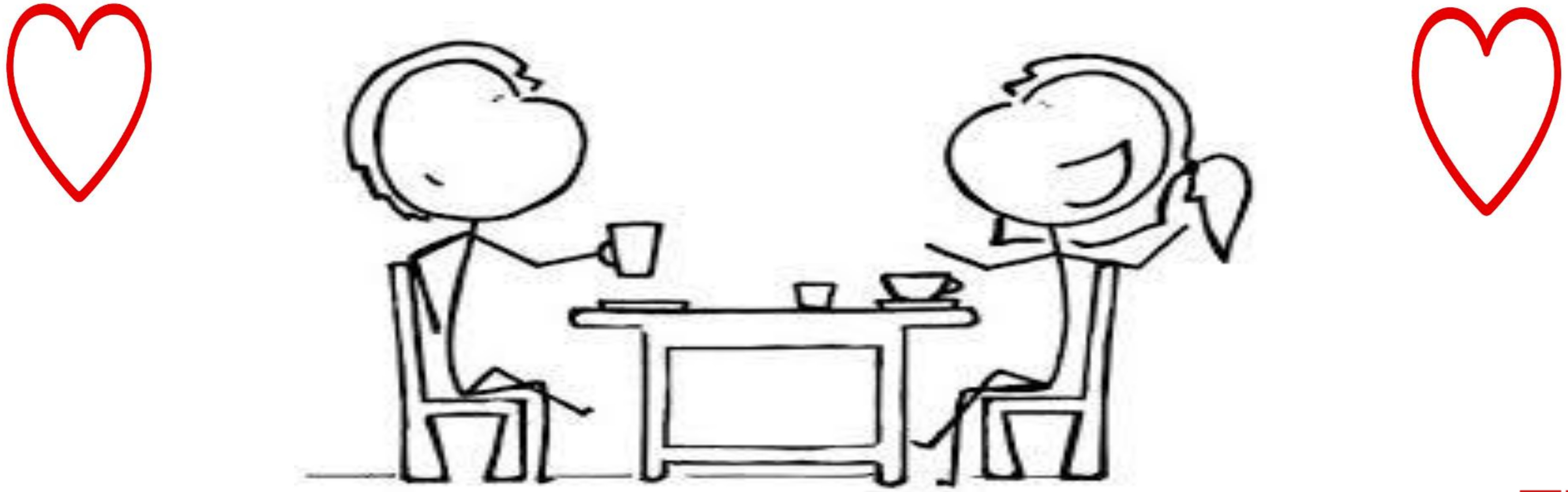
Security of Paillier

- **CPA** : Paillier encryption scheme provides semantic security against chosen-plaintext attack. The ability to successfully distinguish the challenge ciphertext essentially amounts to the ability to decide composite residuosity. The semantic security of the Paillier encryption scheme was proved under the decisional composite residuosity (DCR) assumption—the DCR problem is intractable.

- **CCA**: Paillier encryption is malleable and therefor does not protect against chosen-ciphertext attack.

Secure Multi-party Computation Protocols

Scenario 1:Love Game



Love Game

- The game is actually multiplication.

x	y	xy
0	0	0
1	0	0
0	1	0
1	1	1

$\cong$

Alice	*Bob*	*match?*
no	*no*	-
yes	*no*	-
no	*yes*	-
yes	*yes*	♡

Additive Homomorphic ElGamal Cryptosystem

$$\mathcal{E}_{pk}(m_1) \times \mathcal{E}_{pk}(m_2) = \mathcal{E}_{pk}(m_1 + m_2)$$
$$\mathcal{E}_{pk}(m)^c = \mathcal{E}_{pk}(m \cdot c)$$

Additively homomorphic ELGamal

$$x \in \mathbb{Z}_n, u \in_R \mathbb{Z}_n$$
$$E(x) = (g^u, h^u g^x)$$

Secure Multiplication with plaintext input

$$\begin{array}{ccc} \text{Party } \mathcal{A} & & \text{Party } \mathcal{B} \\ (x \in \{0,1\}) & & (y \in \{0,1\}) \\ u \in_R \mathbb{Z}_n & & \\ A \leftarrow g^u & & \\ B \leftarrow h^u g^x & \xrightarrow{\qquad (A,B) \qquad} & v \in_R \mathbb{Z}_n \\ & & C \leftarrow g^v A^y \\ & \xleftarrow{\qquad (C,D) \qquad} & D \leftarrow h^v B^y \\ & \xleftrightarrow{\text{jointly decrypt } (C,D)} & \\ \text{output } xy & & \text{output } xy \end{array}$$

Note: $(C, D) = (g^{v+uy}, h^{v+uy} g^{xy})$.

References

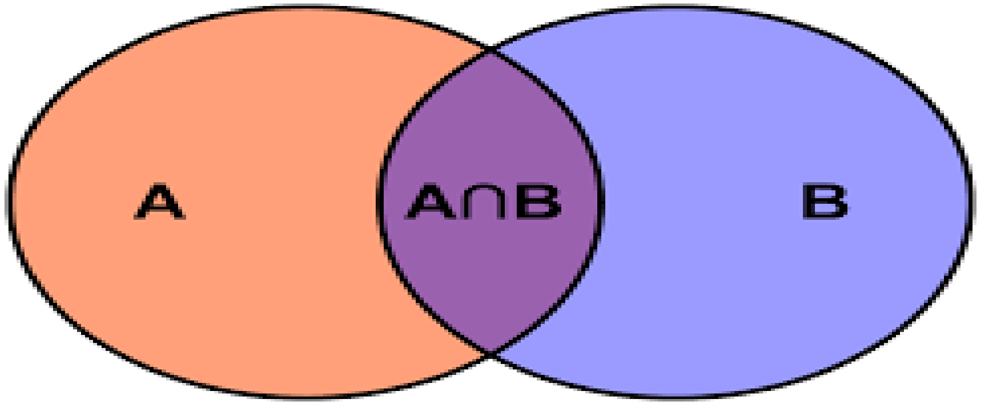
- Essential-reading:
  - Berry Schoenmakers, Lecture Notes Cryptographic Protocols, Chapter 7.
- Recommended reading
  - Oded Goldreich, Foundations of Cryptography, volume 2 Basic Applications, Chapter 7, Cambridge University Press, 2004.

Scenario 2: Finding Potential Terrorists

MI5 Mossad FBI

Finding Potential Terrorists:

- The problem is actually a set intersection.

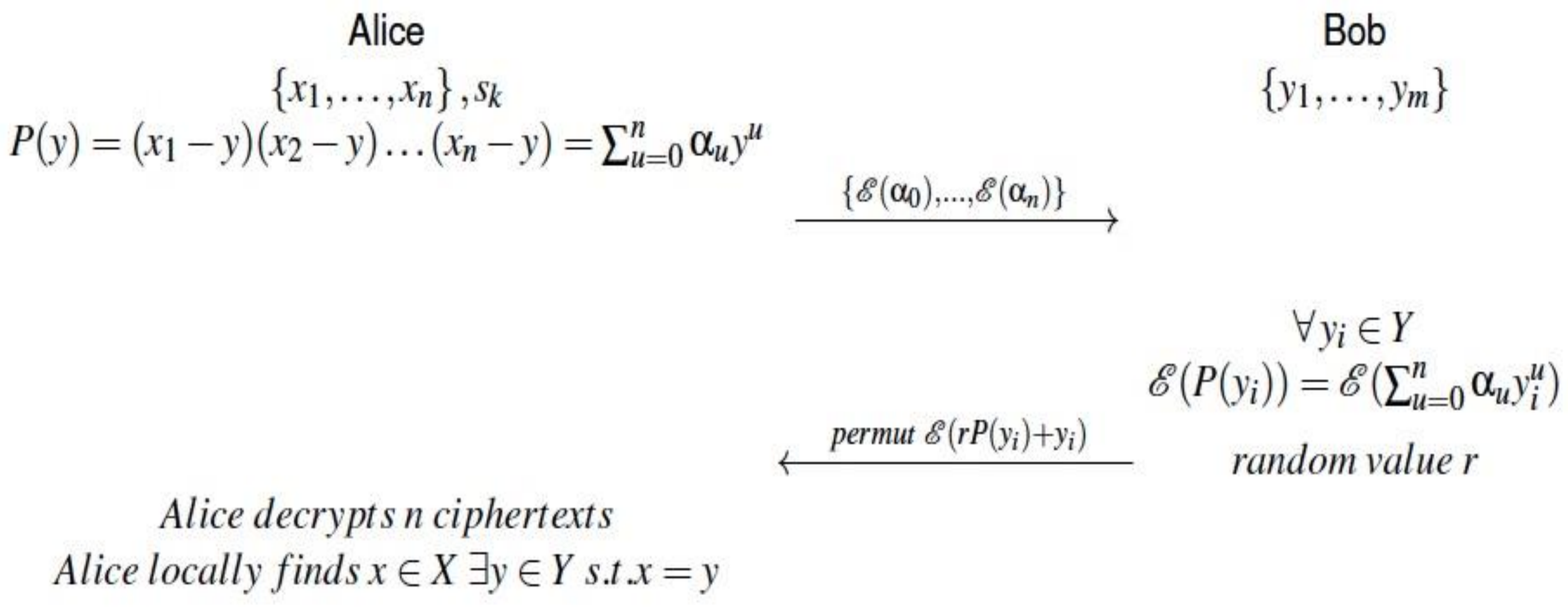


Secure Equality Test Protocol

Alice		Bob
x, p_k, s_k		y, p_k
	$\xrightarrow{\mathscr{E}(x)}$	
		$random\ value\ r$
	$\xleftarrow{(\mathscr{E}(x)+\mathscr{E}(-y))\cdot r=\mathscr{E}((x-y)\cdot r)}$	

$$Alice\ decrypts\ \mathscr{E}((x-y)\cdot r)$$
$$if\ it\ equals\ to\ zero\ it\ means\ x=y$$
$$otherwise\ Alice\ finds\ a\ random\ number$$

Secure Set Intersection Protocol



Alice

$\{x_1,\ldots,x_n\}, s_k$

$P(y) = (x_1 - y)(x_2 - y)\ldots(x_n - y) = \sum_{u=0}^{n} \alpha_u y^u$

$\xrightarrow{\{\mathscr{E}(\alpha_0),\ldots,\mathscr{E}(\alpha_n)\}}$

Bob

$\{y_1,\ldots,y_m\}$

$\forall y_i \in Y$

$\mathscr{E}(P(y_i)) = \mathscr{E}(\sum_{u=0}^{n} \alpha_u y_i^u)$

$random\ value\ r$

$\xleftarrow{permut\ \mathscr{E}(rP(y_i)+y_i)}$

$Alice\ decrypts\ n\ ciphertexts$

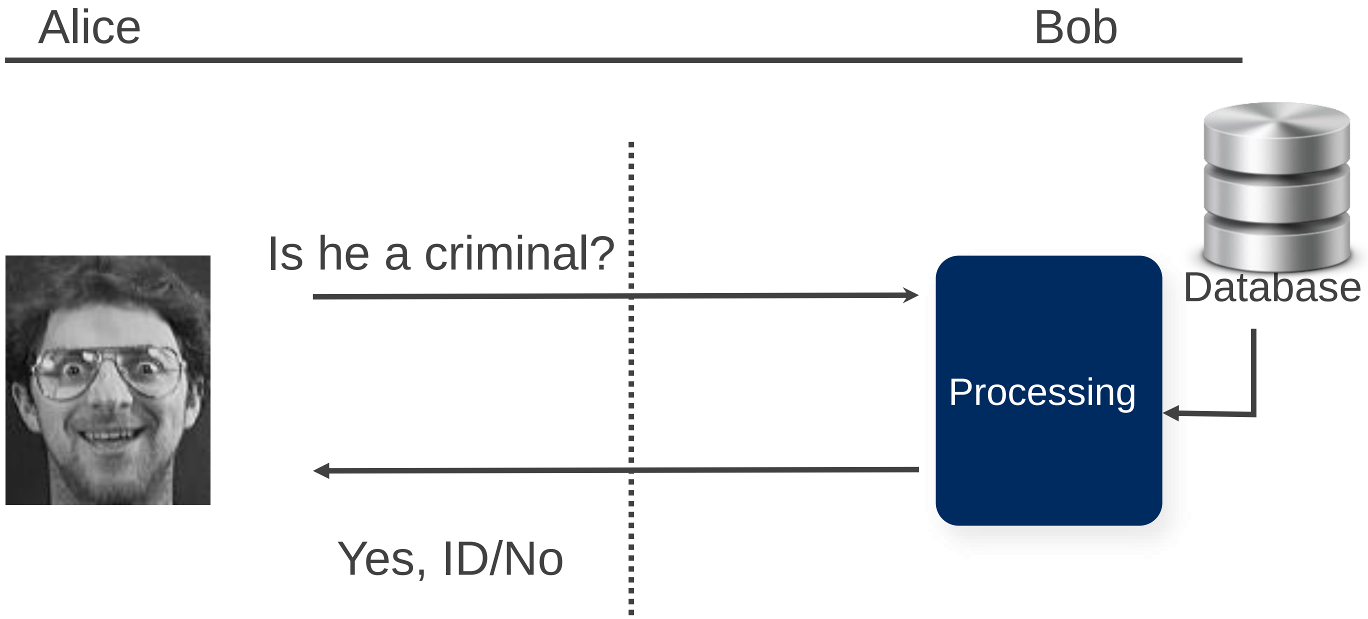
$Alice\ locally\ finds\ x \in X\ \exists y \in Y\ s.t.\ x = y$

References:

- Michael Freeman, Kobbi Nissim, Benny Pinkas, Efficient Private Matching and Set Intersection, Eurocrypt 2004, pp 1-19.

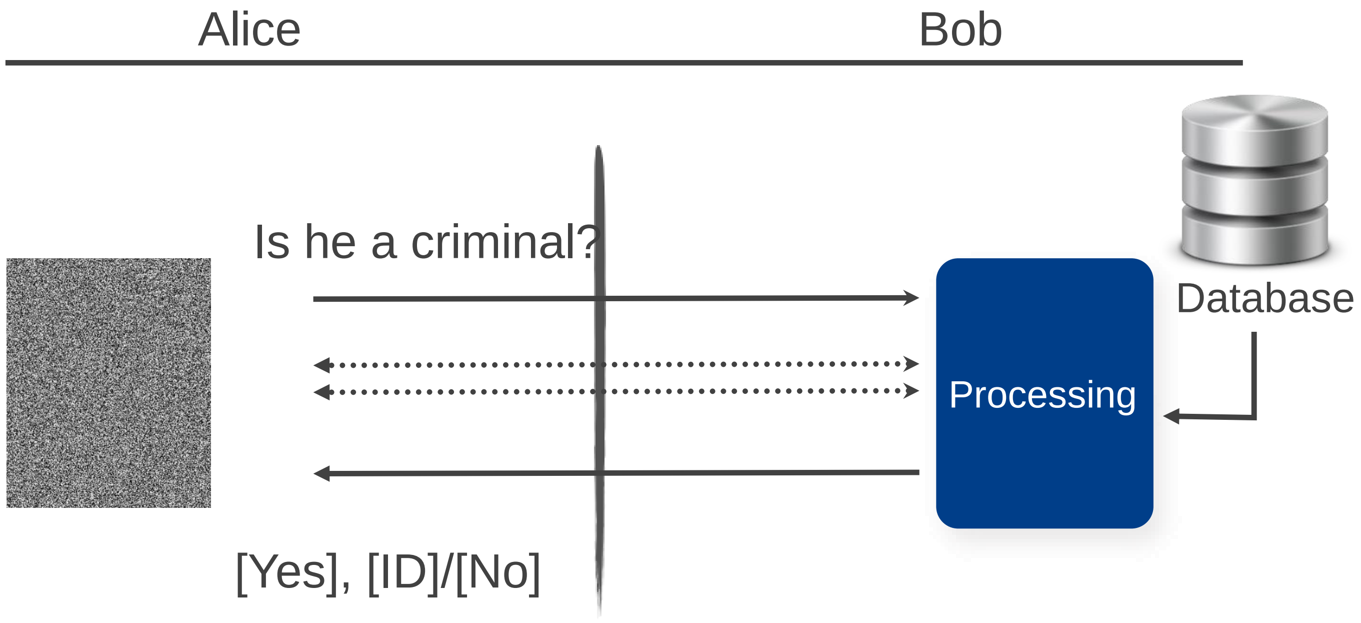
Sceneario 3:Face Recognition

Face Recognition

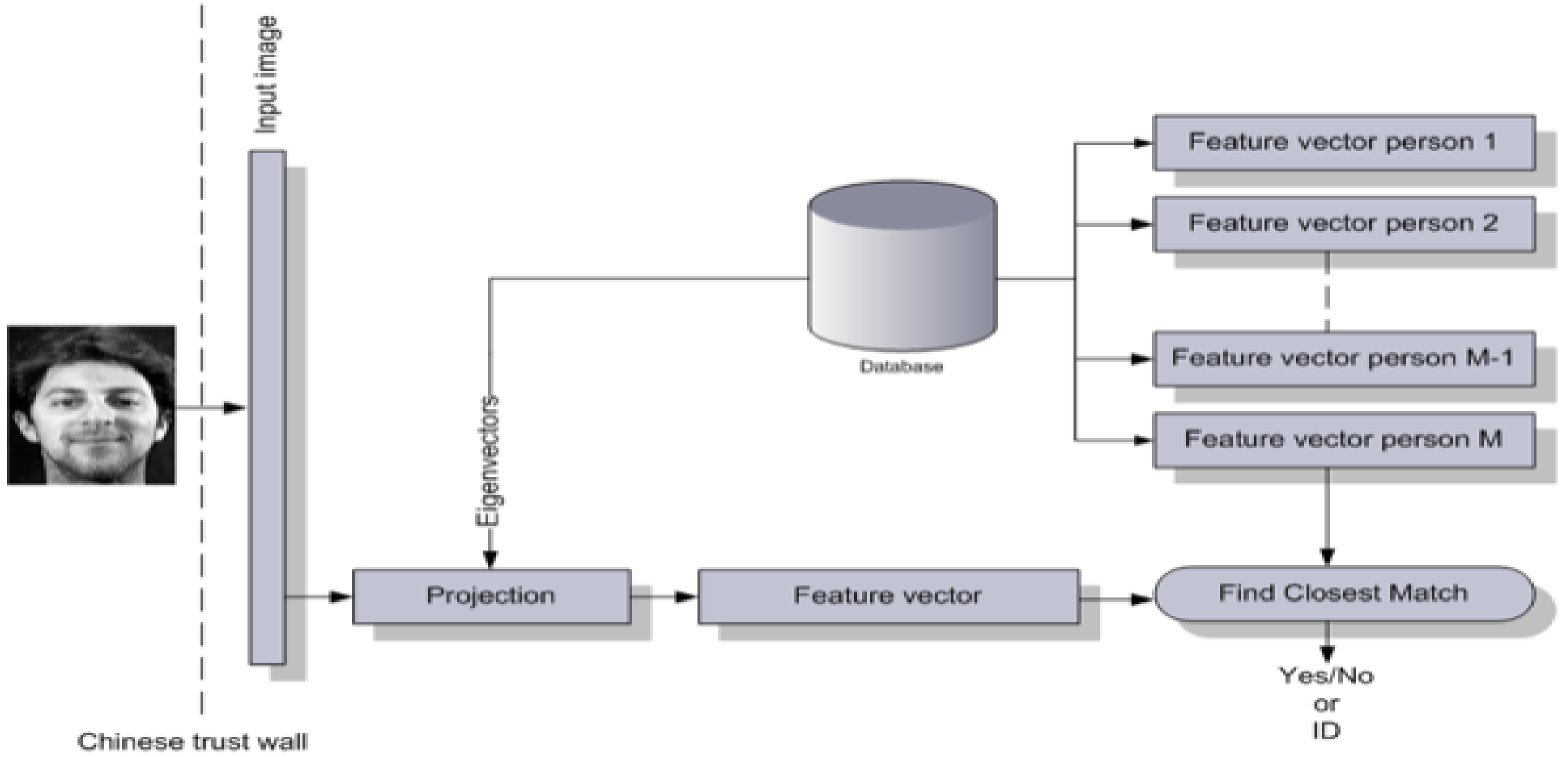


- Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, R. L. Lagendijk and T. Toft, Privacy- Preserving Face Recognition, 9th International Symposium on Privacy Enhancing Technologies, LNCS 5672, pp. 235-253, August 2009.

considering Privacy



Eigenface Algorithm



Secure Face Recognition

- Setting:
  - Customer (Alice) has an input image and her encryption key.
  - Service Provider (Bob) has an image database of criminals. Bob represents each criminal with a feature vector and has the eigenvectors to produce the feature vector of a new input image.
- Security assumptions:
  - Alice and Bob are semi-honest.

Projection in the encrypted domain

Alice (sk) | Bob (pk)

Input image

Feature vectors in a database

$$\xrightarrow{\text{Encrypted pixel values}}$$

Apply projection and obtain the feature vector of the input image.

$$F_a = (f_{(a,1)}, \ldots, f_{(a,K)})$$

$$f_{(a,k)} = \sum_{i=0}^{N} p_i \times e_i^{(k)} \longrightarrow [f_{(a,k)}] = \prod_{i=0}^{N} [p_i]^{e_i^{(k)}}$$

Euclidean Distance

Alice (sk) | Bob (pk)

$$[F_x] = ([f_{(x,1)}], \ldots, [f_{(x,K)}])$$

$$F_y = (f_{(y,1)}, f_{(y,2)}, \ldots, f_{(y,K)})$$

$$D^2(F_x, F_y) = \sum_{i=0}^{K} (f_{x,i} - f_{y,i})^2 = \sum_{i=0}^{K} (\underbrace{f_{x,i}^2}_{\text{Secure Multiplication Protocol!}} - \underbrace{2 \cdot f_{x,i} \cdot f_{y,i} + f_{y,i}^2}_{\text{Homomorphism}})$$

$$[D^2(F_x, F_y)] = \prod_{i=0}^{K} SMP(f_{x,i}) \cdot [f_{x,i}]^{-2 \cdot f_{y,i}} \cdot [f_{y,i}^2]$$

Secure Multiplication Protocol

Alice $[a],[b],pk$ Bob sk

$$[\tilde{a}] := [a+r_1] = [a] \times [r_1]$$
$$[\tilde{b}] := [b+r_2] = [b] \times [r_2]$$

$$\xrightarrow{[\tilde{a}],[\tilde{b}]}$$

$$\tilde{a}\cdot\tilde{b} = (a+r_1)\cdot(b+r_2)$$
$$= (ab + ar_2 + br_1 + r_1r_2)$$

$$\xleftarrow{[\tilde{a}\cdot\tilde{b}]}$$

$$[a\cdot b] = [\tilde{a}\cdot\tilde{b}]\cdot[a]^{-r_2}\cdot$$
$$[b]^{-r_1}\cdot[r_1r_2]^{-1}$$

Finding the minimum

Alice (sk)	Bob (pk)
	$[D^2(F_x, F_y)], [D^2(F_x, F_w)], \ldots, [D^2(F_x, F_z)]$
	Find the minimum squared distance!

But…

$$
\begin{aligned}
[D^2(F_x, F_y)] &= g^{D^2(F_x, F_y)} r_1^n \bmod n^2 \\
&= 154894318447855\ldots.4848948974897
\end{aligned}
$$

$$
\begin{aligned}
[D^2(F_x, F_w)] &= g^{D^2(F_x, F_w)} r_2^n \bmod n^2 \\
&= 956814894149\ldots.123484987163
\end{aligned}
$$

Secure Comparison

Which one has more money?!

A = how much money user1 has

B = how much money user2 has

$$f(A,B)=\begin{cases}\text{User 1} & \text{if } A>B\\ \text{Equal} & \text{if } A=B\\ \text{User2} & \text{if } A<B\end{cases}$$

Finding the Minimum: Comparison

$$10 \longleftarrow a = 1010$$
$$14 \longleftarrow b = 1110$$

a < b

$$2^4 + a - b = 10000 + 1010 - 1110 = \boxed{0}1100$$

$$a = 1110$$
$$b = 1010$$

a > b

$$2^4 + a - b = 10000 + 1110 - 1010 = \boxed{1}0100$$

$$z = 2^4 + a - b$$

$$0 \xleftarrow[a < b]{} \lambda = (z - z \bmod 2^4)2^{-4} \xrightarrow[a > b]{} 1$$

Interactive Game

$$\begin{tikzcd}
\text{Alice} & & \text{Bob} \\
 & & {[z] = [2^\ell + a - b]} \\
 {} & & {[c] = [z + r]} \arrow[ll, "{[c]}"'] \\
 {D([c]) = c} \arrow[rr, "{[c_\ell]}"] & & {[c_0], [c_1], \ldots, [c_{\ell-1}]}
\end{tikzcd}$$

$$z \bmod 2^\ell = (c \bmod 2^\ell - r \bmod 2^\ell) \bmod 2^\ell$$

$$z \bmod 2^\ell = (c \bmod 2^\ell - r \bmod 2^\ell) + 2^\ell \cdot k$$

$$k = r > c \; ? \; 1 : 0$$

Comparison

$$c = 000011$$
$$r = 001111$$

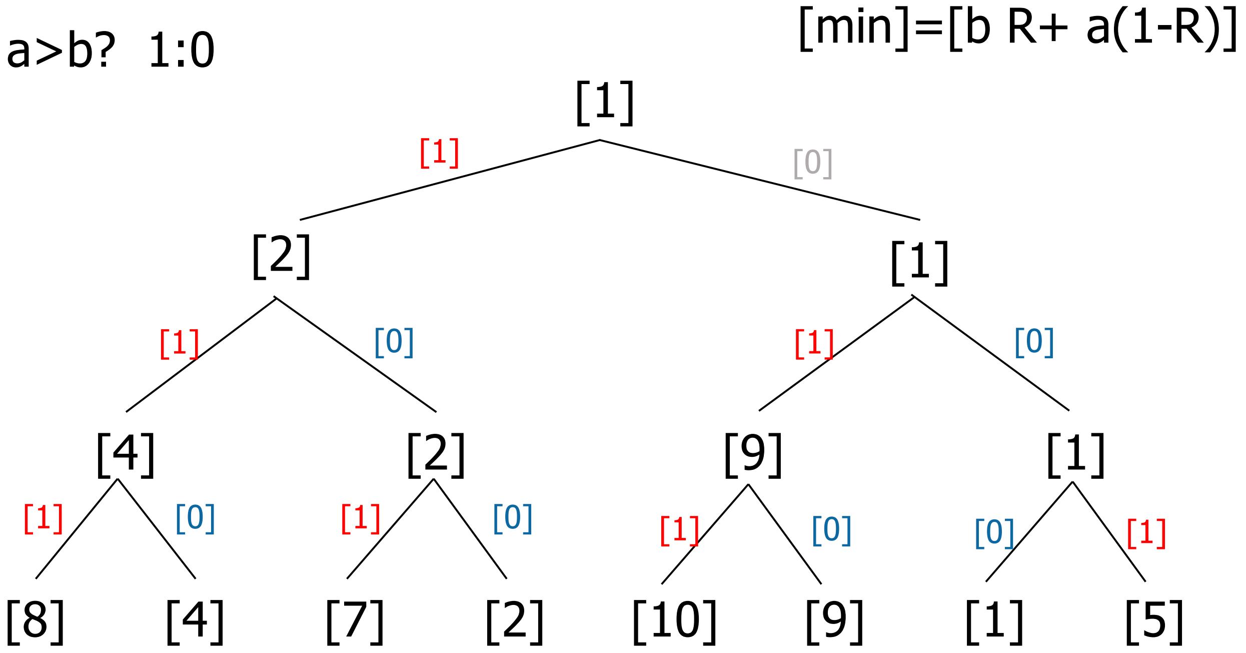
$$e_i = 1 + c_i - r_i + \sum_{j=i+1}^{\ell-1} c_j \oplus r_j$$

$$e_0 = 1 + 1 - 1 + (2) = 3$$
$$e_1 = 1 + 1 - 1 + (2) = 3$$
$$e_2 = 1 + 0 - 1 + (1) = 1$$
$$e_3 = 1 + 0 - 1 + (0) = 0$$
$$e_4 = 1 + 0 + 0 + (0) = 1$$
$$e_5 = 1 + 0 + 0 + (0) = 1$$

Comparison Protocol

- Alice computes the e values in the encrypted domain and sends them to Bob after shuffling.
- Bob decrypts e values and checks if there is any zero among them: if yes, sends [1], otherwise [0].
- Alice uses the response from Bob for the final step to compute the outcome of the comparison.

Finding the minimum



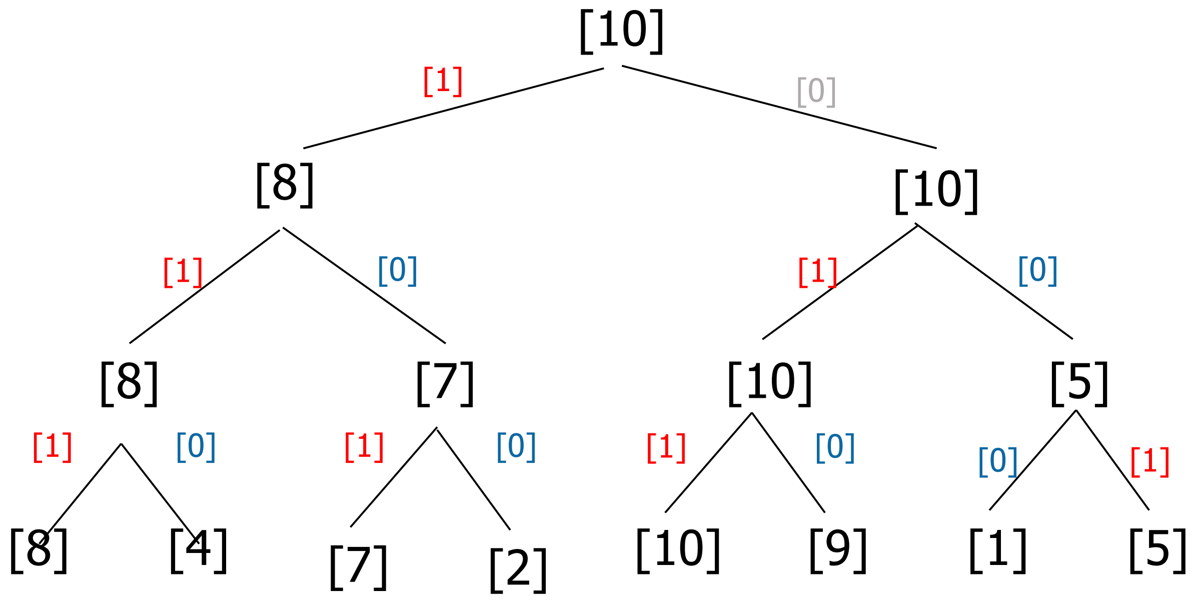
References

- Essential reading:
  - Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, R. L. Lagendijk and T. Toft, Privacy- Preserving Face Recognition, 9th International Symposium on Privacy Enhancing Technologies, LNCS 5672, pp. 235-253, August 2009.
  - P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In J. Stern, editor, Advances in Cryptology — EUROCRYPT ’99, volume 1592 of LNCS, pages 223–238. Springer, May 2-6, 1999.
  - Nigel Smart’s book, chapter 1.
- Recommended reading
  - Vladimir Kolesnikov, Ahmad-Reza Sadeghi, Thomas Schneider: Improved Garbled Circuit Building Blocks and Applications to Auctions and Computing Minima. CANS 2009: 1-20
  - Ahmad-Reza Sadeghi, Thomas Schneider, Immo Wehrenberg: Efficient Privacy-Preserving Face Recognition. IACR Cryptology ePrint Archive 2009: 507 (2009)

Scenario 4: Private Auction

Auction: Finding the maximum

a>b?1:0



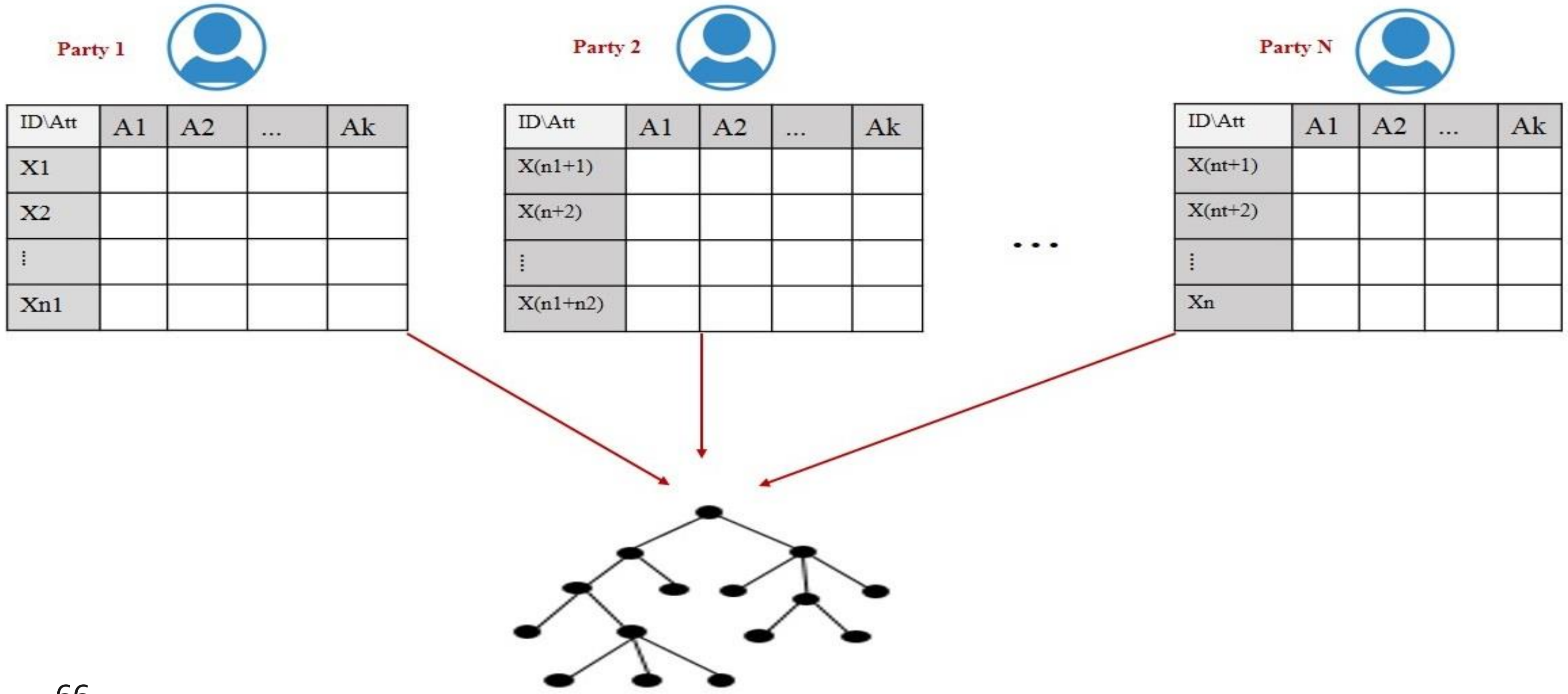
Scenario 5: Distributed Data Mining

Data Mining Operations

Collaborative Analysis

Horizontal Data Distribution:

Example: Hospital and Health Center.



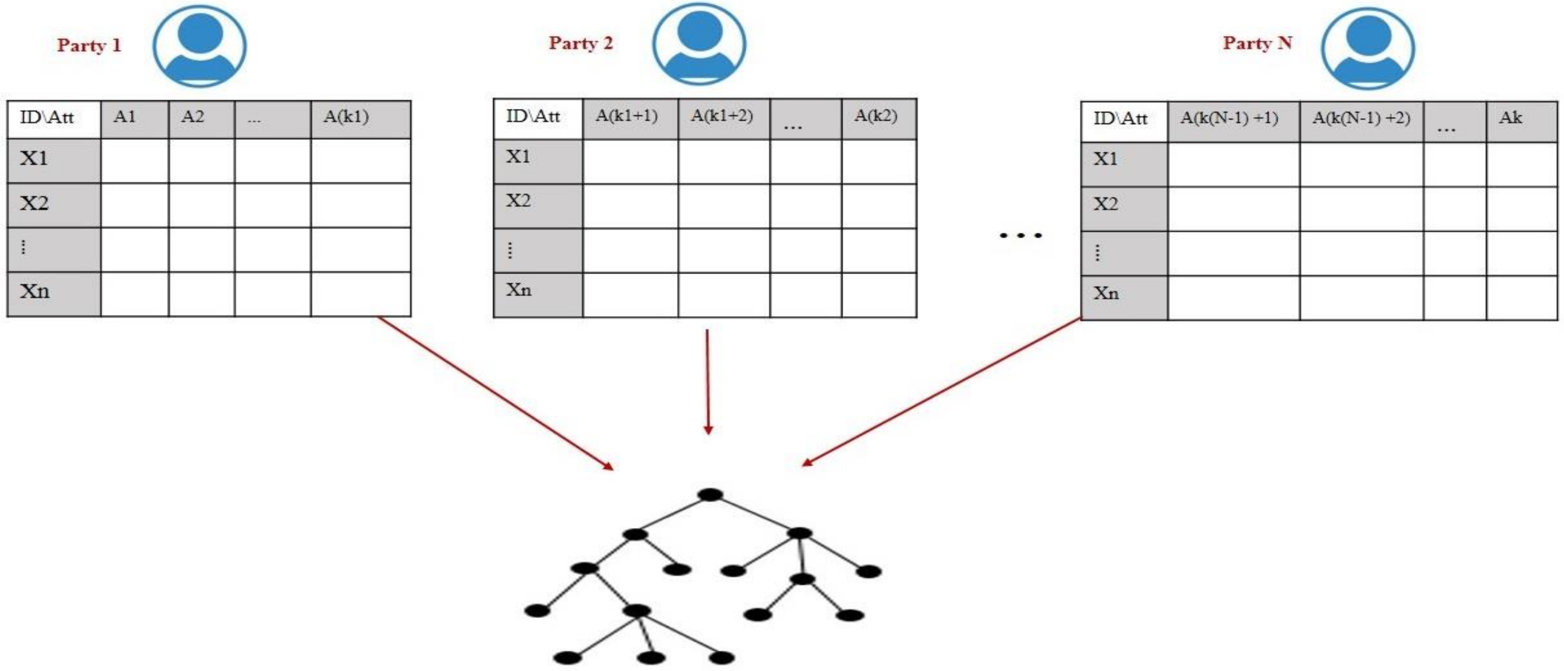
ID\Att	A1	A2	...	Ak
X1				
X2				
⋮				
Xn1				

ID\Att	A1	A2	...	Ak
X(n1+1)				
X(n+2)				
⋮				
X(n1+n2)				

ID\Att	A1	A2	...	Ak
X(nt+1)				
X(nt+2)				
⋮				
Xn				

Vertical Data Distribution:

Example: Facebook and Google accounts.

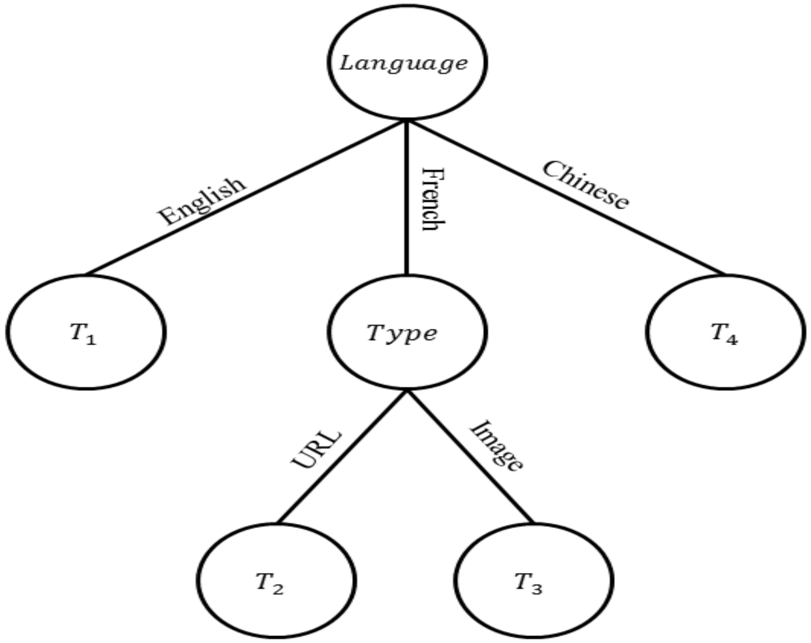


ID\Att	A1	A2	...	A(k1)
X1				
X2				
⋮				
Xn				

ID\Att	A(k1+1)	A(k1+2)	...	A(k2)
X1				
X2				
⋮				
Xn				

ID\Att	A(k(N-1) +1)	A(k(N-1) +2)	...	Ak
X1				
X2				
⋮				
Xn				

Categorical Clustering Tree (CCTree)



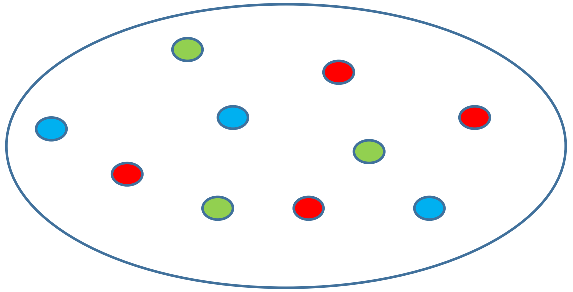
CCTree Criteria

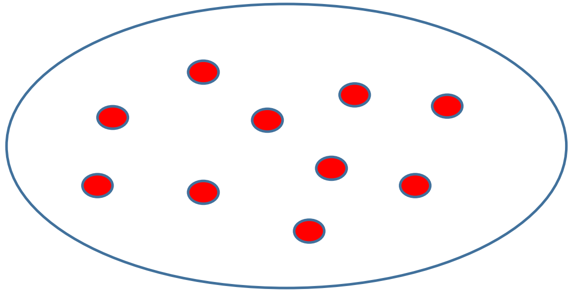
❖ **Stop Conditions:**

1) Number of elements in a node
2) Node purity

❖ **Split Attribute**

Shannon Entropy

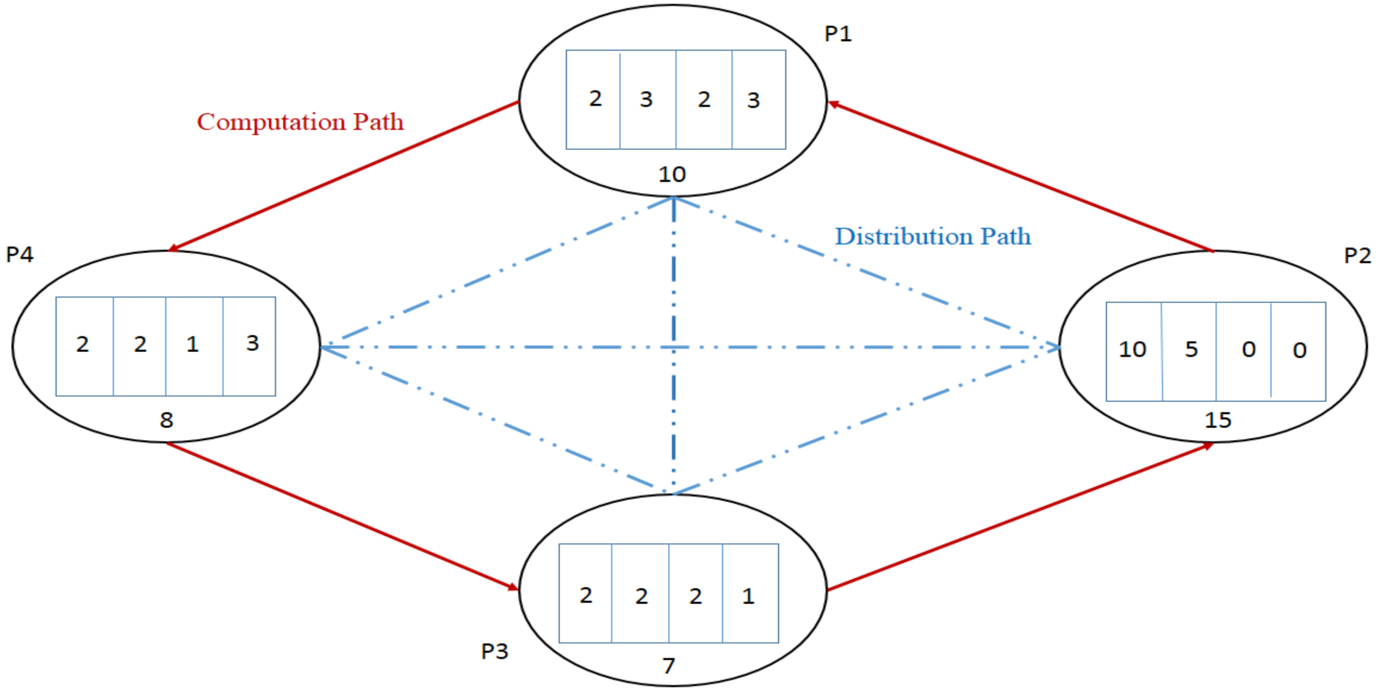




$$-\ (0.4 \log (0.4) + 0.3 \log (0.3) + 0.3 \log (0.3)\ ) = 0.4729$$

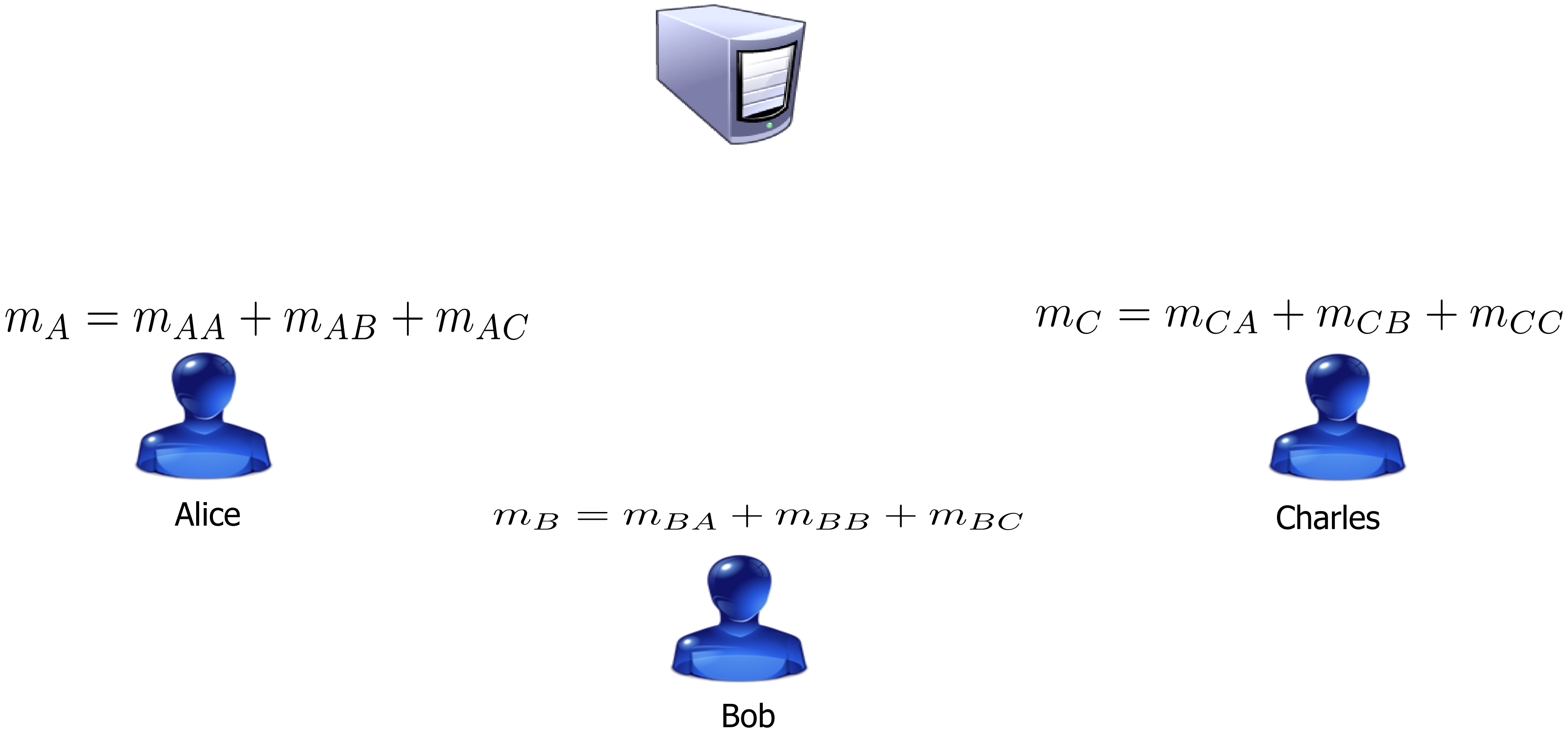
$$-\ \frac{10}{10} \log (\frac{10}{10}) = 0$$

Secure Sum (1): Distributed Secure Sum Protocol



Secure Sum (2): Secret Sharing and HE

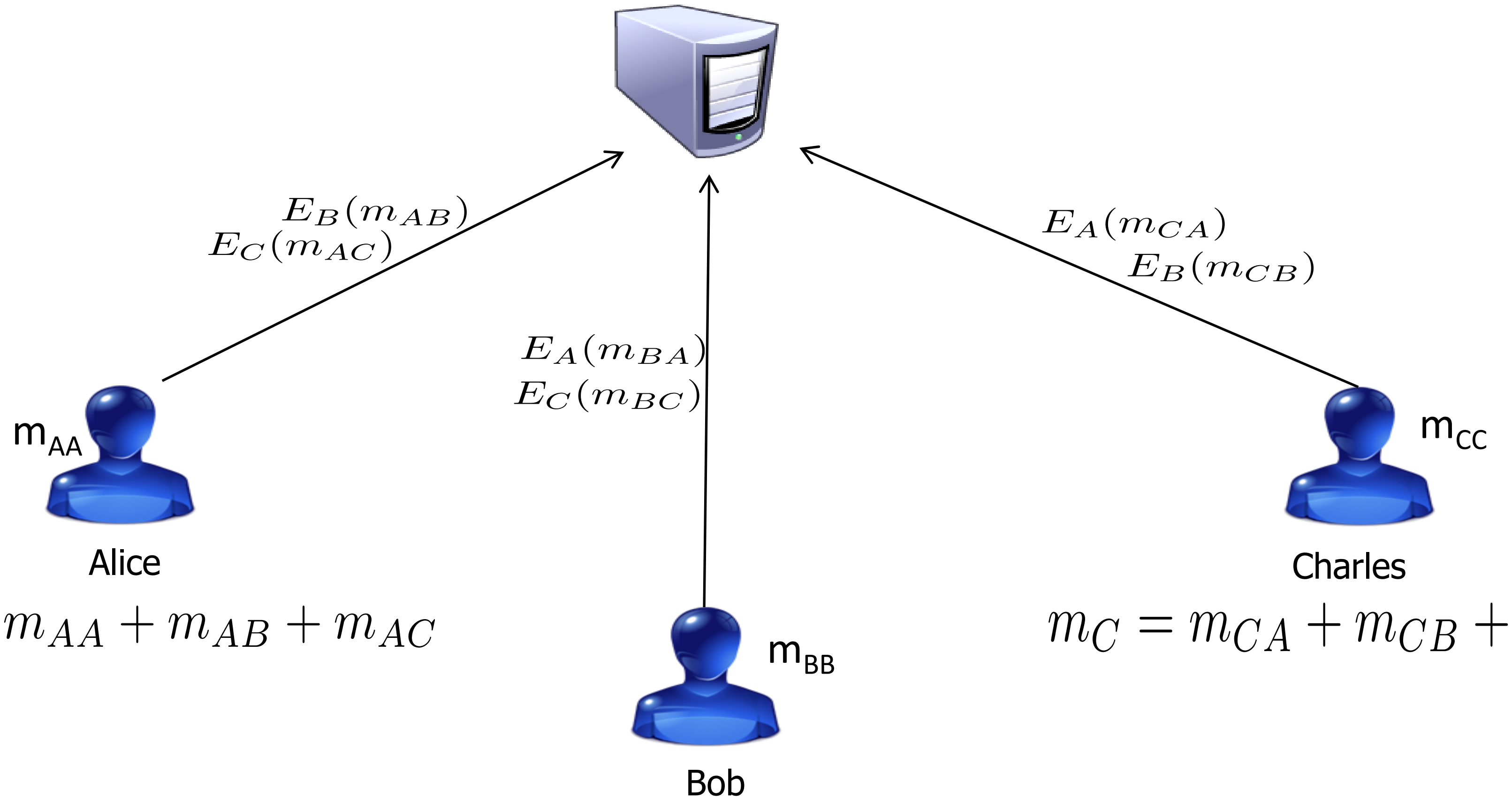
- First protocol for data aggregation
  - Additive homomorphic encryption
  - Secret sharing over mod n
- Interactive



$$m_A = m_{AA} + m_{AB} + m_{AC}$$

$$m_B = m_{BA} + m_{BB} + m_{BC}$$

$$m_C = m_{CA} + m_{CB} + m_{CC}$$



$$m_A = m_{AA} + m_{AB} + m_{AC}$$

$$m_C = m_{CA} + m_{CB} + m_{CC}$$

$$m_B = m_{BA} + m_{BB} + m_{BC}$$

$$\begin{matrix} E_A(m_{BA}) & E_A(m_{CA}) \\ E_B(m_{AB}) & E_B(m_{CB}) \\ E_C(m_{AC}) & E_C(m_{BC}) \end{matrix} \longrightarrow \begin{matrix} E_A(m_{BA}) \times E_A(m_{CA}) = E_A(m_{BA} + m_{CA}) \\ E_B(m_{AB}) \times E_B(m_{CB}) = E_B(m_{AB} + m_{CB}) \\ E_C(m_{AC}) \times E_C(m_{BC}) = E_C(m_{AC} + m_{BC}) \end{matrix}$$

Alice

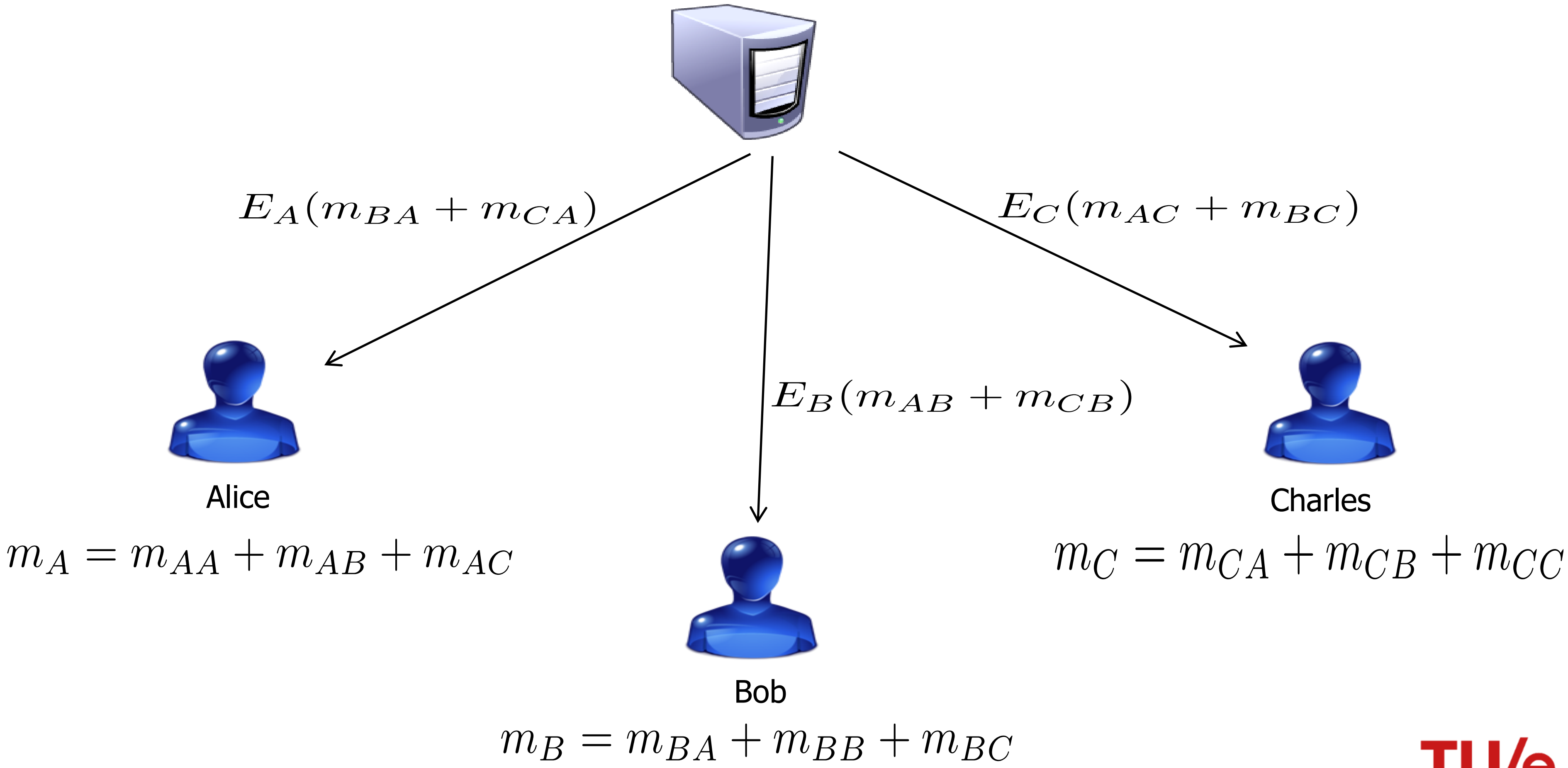
$$m_A = m_{AA} + m_{AB} + m_{AC}$$

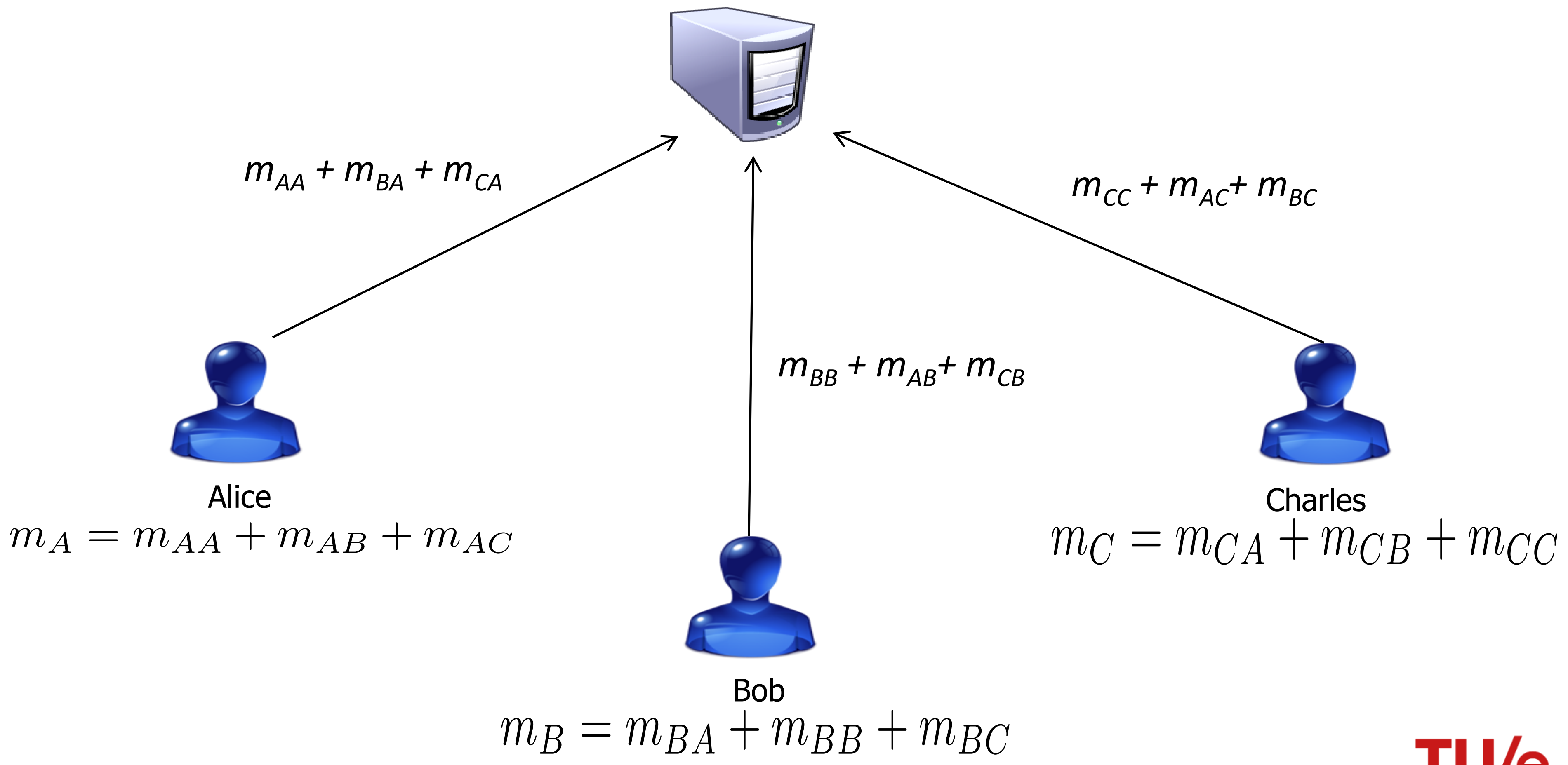
Charles

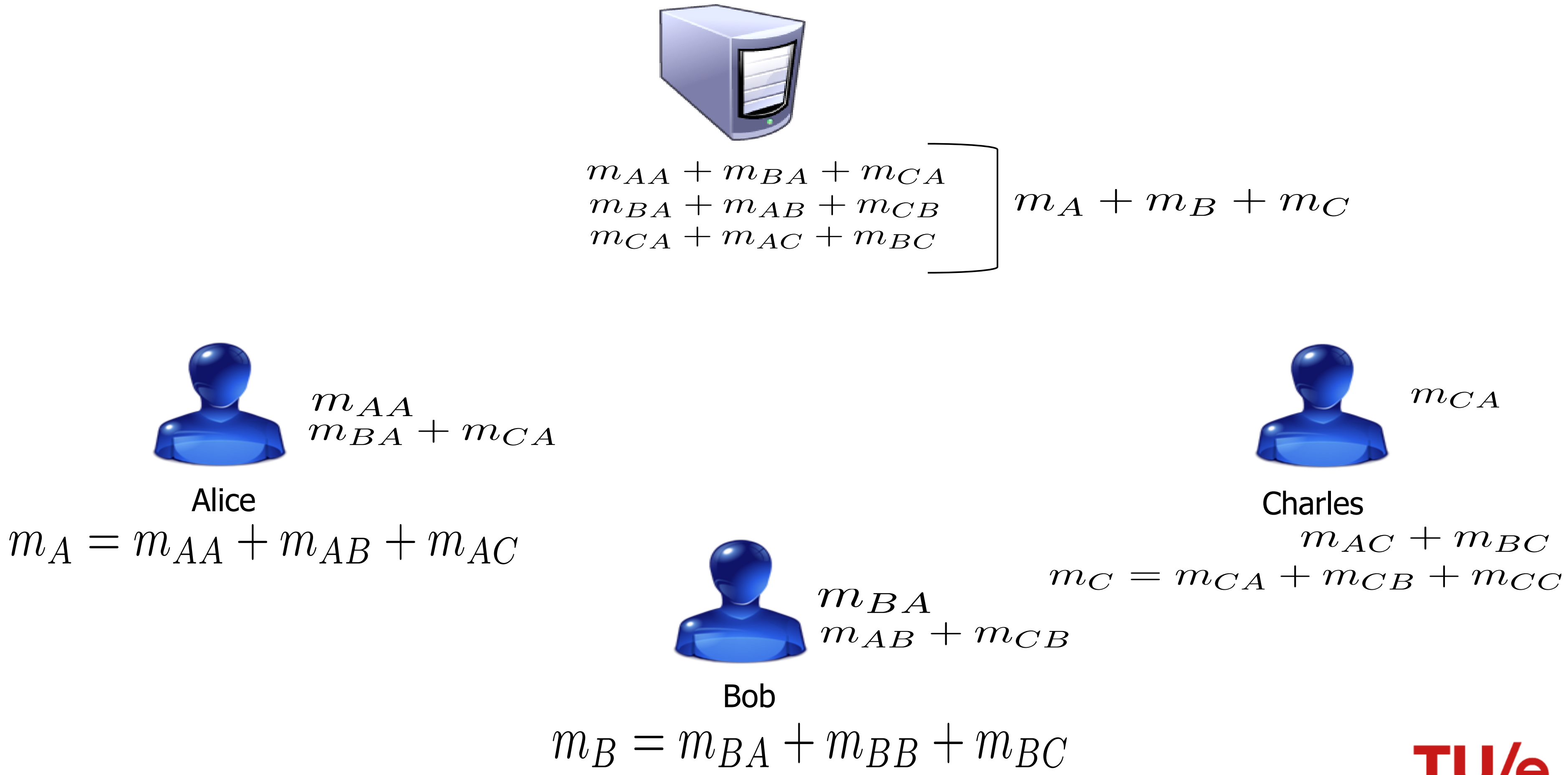
$$m_C = m_{CA} + m_{CB} + m_{CC}$$

Bob

$$m_B = m_{BA} + m_{BB} + m_{BC}$$


$E_A(m_{BA} + m_{CA})$
$E_B(m_{AB} + m_{CB})$
$E_C(m_{AC} + m_{BC})$
Alice
Bob
Charles
$m_A = m_{AA} + m_{AB} + m_{AC}$
$m_B = m_{BA} + m_{BB} + m_{BC}$
$m_C = m_{CA} + m_{CB} + m_{CC}$


$m_{AA} + m_{BA} + m_{CA}$
$m_{BB} + m_{AB} + m_{CB}$
$m_{CC} + m_{AC} + m_{BC}$
Alice
$m_A = m_{AA} + m_{AB} + m_{AC}$
Bob
$m_B = m_{BA} + m_{BB} + m_{BC}$
Charles
$m_C = m_{CA} + m_{CB} + m_{CC}$


$\left.\begin{array}{l} m_{AA} + m_{BA} + m_{CA} \\ m_{BA} + m_{AB} + m_{CB} \\ m_{CA} + m_{AC} + m_{BC} \end{array}\right] m_A + m_B + m_C$
m_{AA}
$m_{BA} + m_{CA}$
Alice
$m_A = m_{AA} + m_{AB} + m_{AC}$
m_{CA}
Charles
$m_{AC} + m_{BC}$
$m_C = m_{CA} + m_{CB} + m_{CC}$
m_{BA}
$m_{AB} + m_{CB}$
Bob
$m_B = m_{BA} + m_{BB} + m_{BC}$

TU/e

CCTree/ Horizontal/ Multi-Party

❖ **Split Attribute** (Secure Sum)

❖ **Stop Conditions:**

1) Number of elements (Secure Sum)
2) Node purity (Secure Entropies Addition)

CCTree/ Vertical/ Multi-Party

- ❖ **Split Attribute** (Secure Comparison)

- ❖ **Stop Conditions:**
  1) Number of elements (Secure Sum)
  2) Node purity (Secure Entropies Addition)

References

- Kursawe, K., Danezis, G., Kohlweiss, M.: Privacy-Friendly Aggregation for the Smart-Grid. In: Fischer-Hubner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 175–191. Springer, Heidelberg (2011)
- Garcia, F.D., Jacobs, B.: Privacy-Friendly Energy-Metering via Homomorphic Encryption. In: Cuellar, J., Lopez, J., Barthe, G., Pretschner, A. (eds.) STM 2010. LNCS, vol. 6710, pp. 226–238. Springer, Heidelberg (2011)
- Rashid Sheikh, Beerendra Kumar, Durgesh Kumar Mishra, A distributed k-secure sum protocol for secure Multi Party Computation, Journal of Computing, vol.2, issue 3, 2010
- Sheikhalishahi, M., Martinelli, F., Privacy Preserving Clustering over Horizontal and Vertical Partitioned Data, ISCC2017, 2017