Privacy Preserving Signal Processing

Prof. Inald Lagendijk

(R.L.Lagendijk@TUDelft.nl)

Zeki Erkin, Michael Beye, Thijs Veugen
EU SPEED partners
Sentinels Kindred Spirits partners

Information Security and Privacy Lab

Delft University of Technology

About Me …

- Full professor TU Delft, the Netherlands.
- Fellow IEEE.
- Appointed member of the Royal Netherlands Academy of Arts and Sciences.

- >100 papers, 1100 citations, Hirsch factor: 30.
- Acquired over 30 M€ indirect research funding.
- >100 M.Sc. Students, ~ 25 Ph.D students.
- Signal processing, content retrieval, privacy.

- Research department of 120 people in Signal Processing, Pattern Recognition, Visualization, Man-Machine Interaction.
- Application domains: Media, Telecom, Health.

- Scientific director Delft Institute for Research on ICT.
- Director of national COMMIT research program (100 M€).

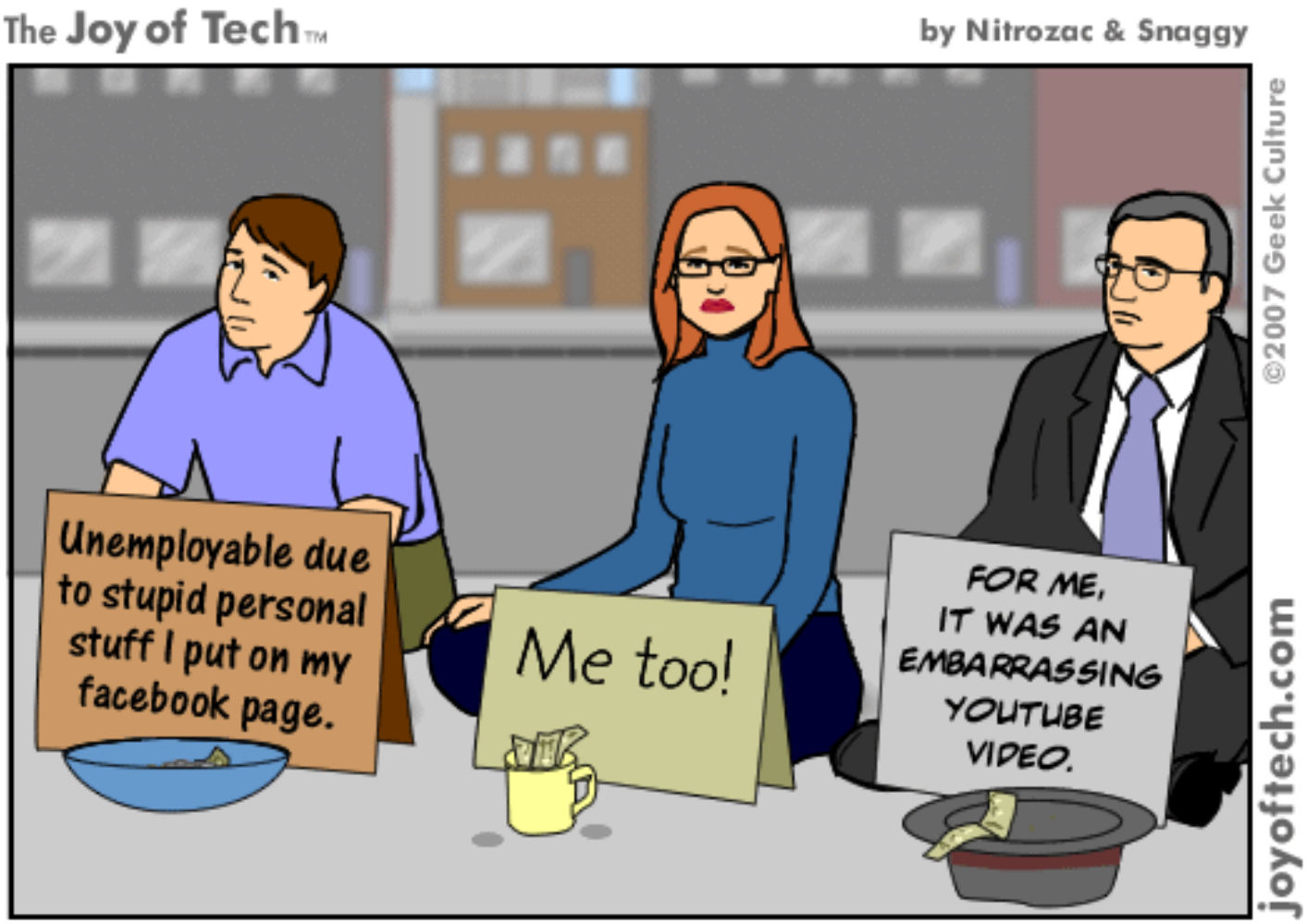
Do I need to convince you that Privacy on the Internet is an issue?

☑ No. Skip next sheets / Go into sleep mode.

☑ Yes. Pay attention to some illustrations.

We are Becoming Increasingly Reliant on …

Privacy of Sensitive Personal Information (1)

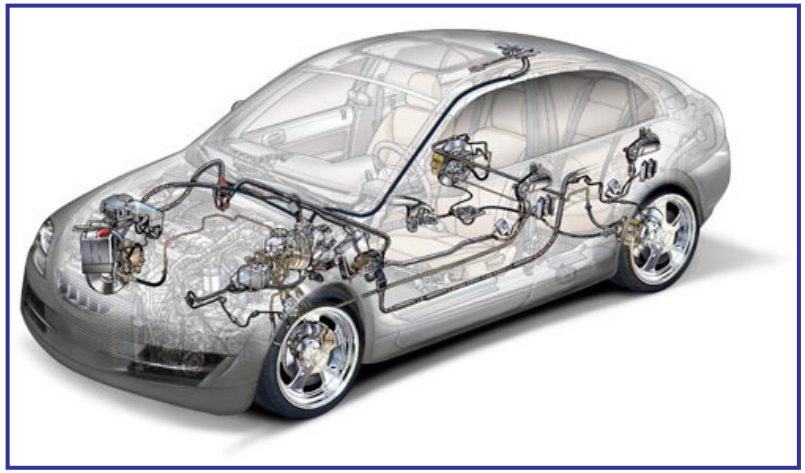


Privacy of Sensitive Personal Information (2)

Privacy of Sensitive Personal Information (3)

Privacy of Sensitive Personal Information (4)

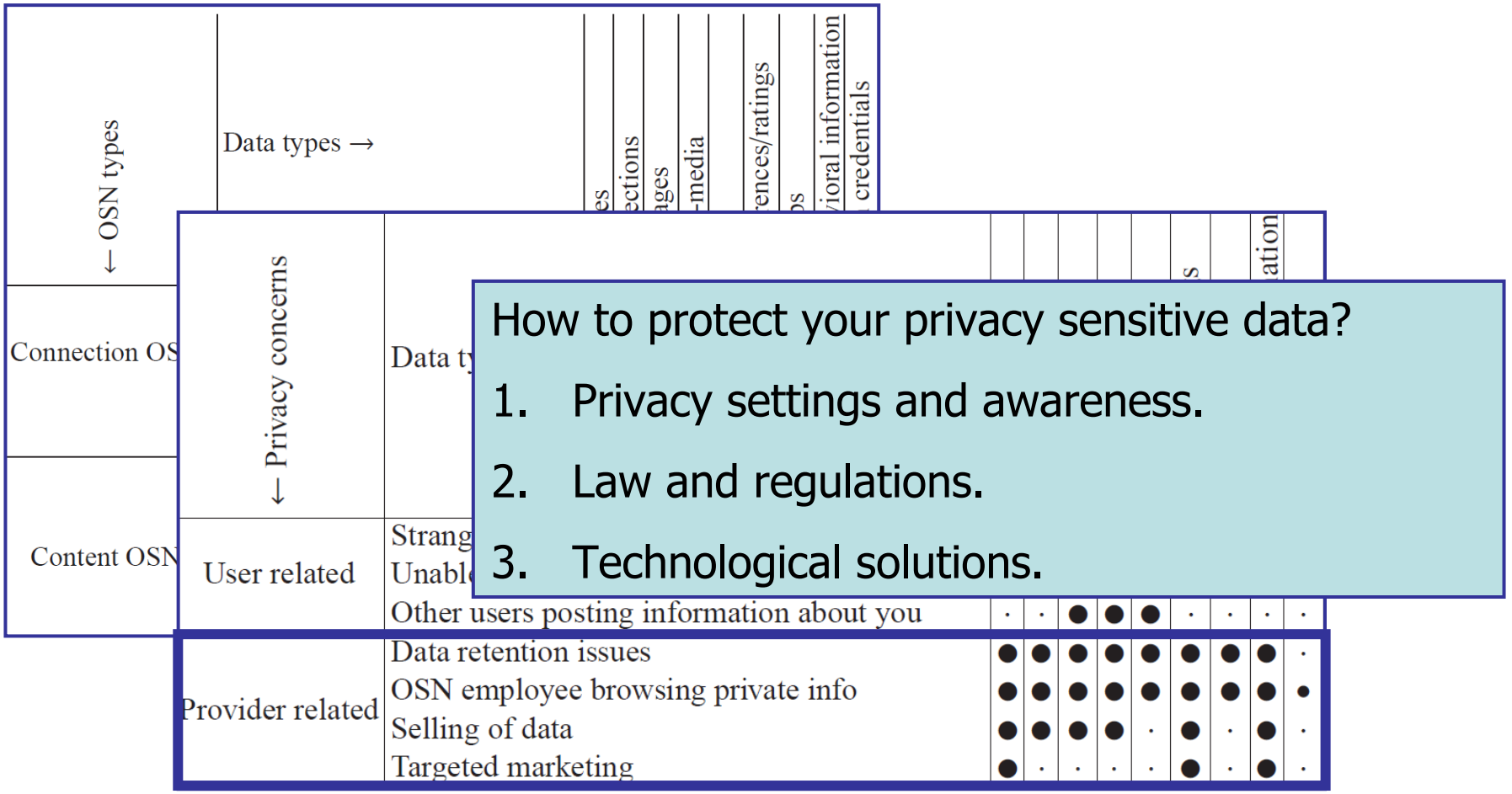
- Beyond the Internet



Privacy versus Functionality

- Increasingly, the provision of desirable personalized services such as recommendation (books, navigation, health advices, …) require sensitive personal information.
- Without that personal information, the personalized service cannot work.
- What data, what privacy risks?

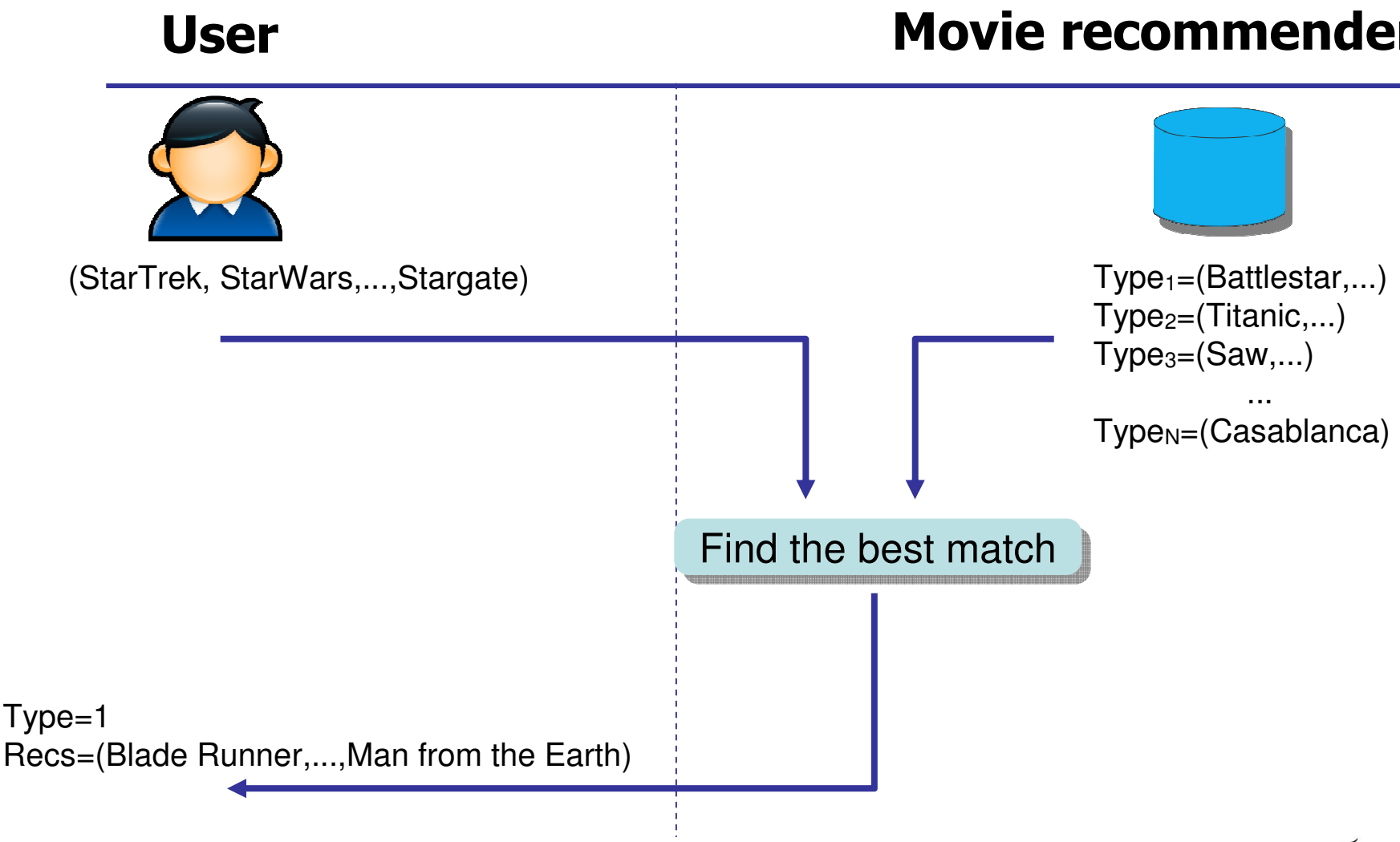
Data and Privacy Risk for OSNs



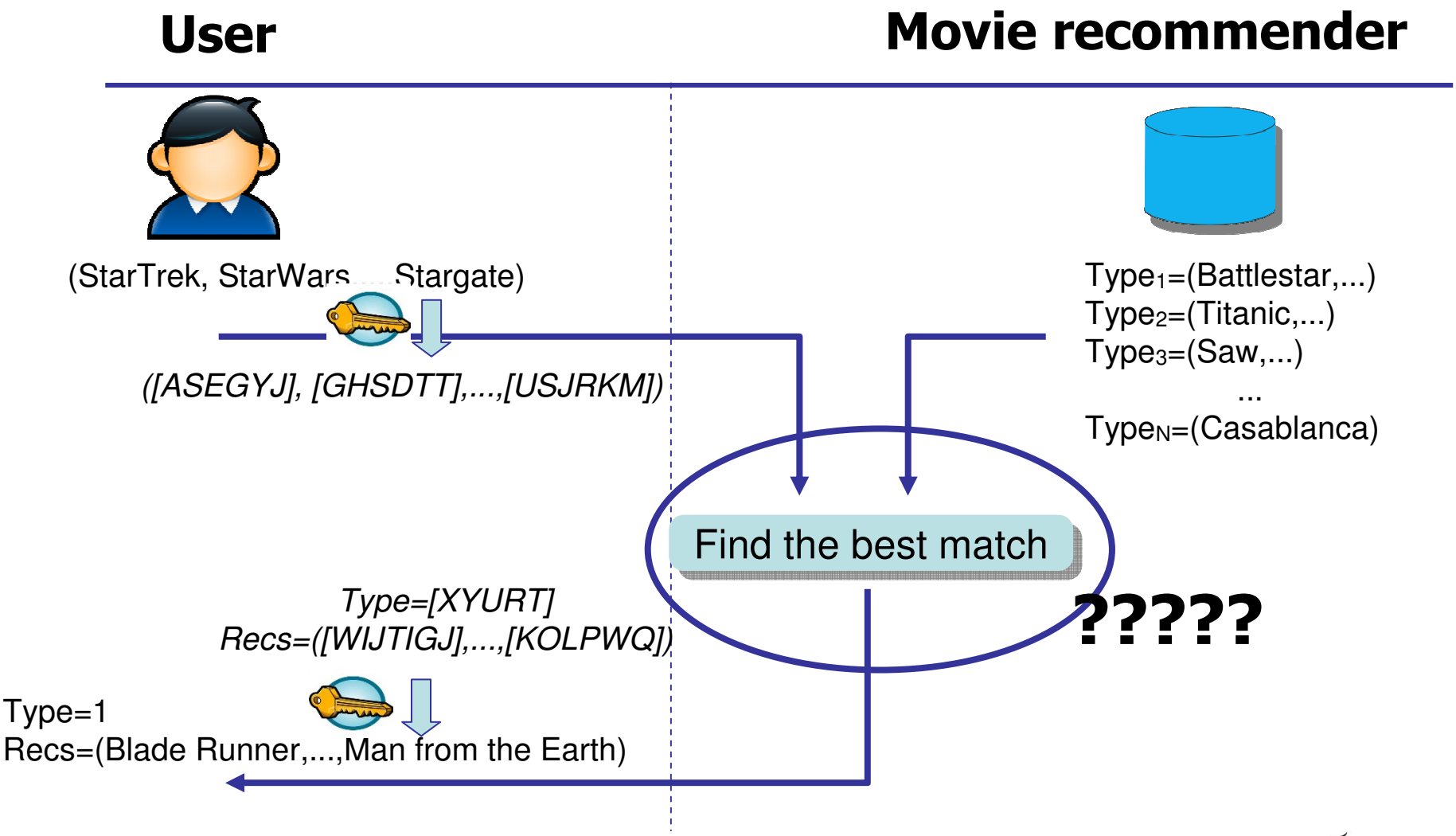
Privacy versus Functionality

- Increasingly, the provision of desirable services such as recommendation (books, navigation, health advices, …) require sensitive personal information.
- Without that personal information, the service cannot work.
- What data, what privacy risks?
- Focus in this lecture is on …
  - … privacy protection (PP) …
  - … against the *provider of the service* …
  - … guaranteed by technological (cryptographic) means.

Example of a PP Recommender System (1)



Example of a PP Recommender System (2)



Take Home Message

- Privacy preserving (secure) signal processing is a new branch on the multimedia research tree.

- Solutions require truly interdisciplinary research …
- … and the results are of high societal relevance.

- Multimedia (signal processing) community faces a wealth of challenges.

Content of the Lecture

- Introduction (done)
- Cryptography 2.0. (until 12:30 pm)
  - Homomorphic cryptography.
  - Secure multiparty computation.
- Secure face recognition. (start at 2pm)
- Secure recommender system.
- Challenges. (wrap up around 3:30 pm)

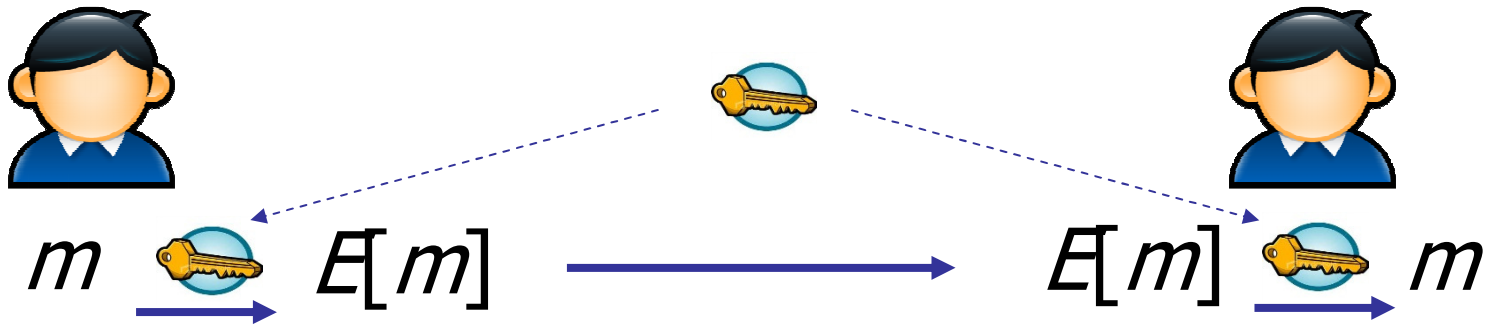
The State of … Cryptography

- Crypto 1.0
  - Communication between 2 (or more) parties.
  - Security model: protect communication against an outsider, the malicious adversary.
  - Private key cryptography*: DES, AES, …
  - Public key cryptography*: RSA, Paillier, El Gamal, …
  - Hashes: MD*, RIPEMD, SHA*
  - Secure communication, signature schemes, integrity, …

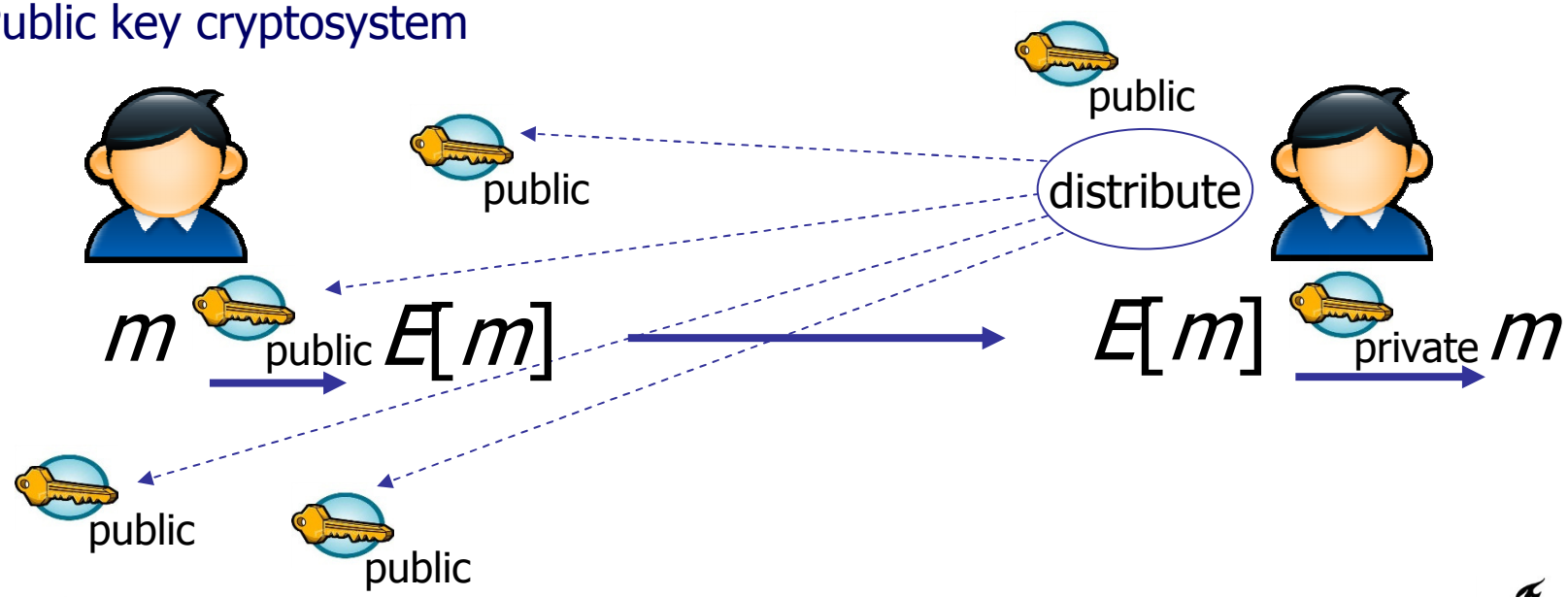
- Crypto 2.0
  - Communication and computing with a second (multi) party.
  - Security model: protect against curious or malicious behavior of certain parties.
  - Homomorphic cryptosystems
  - Secure multiparty computation

Private versus Public Key Cryptosystem

Private key cryptosystem



Public key cryptosystem



Crypto 2.0: The Best-known Example (1)

A(lice)
has € x

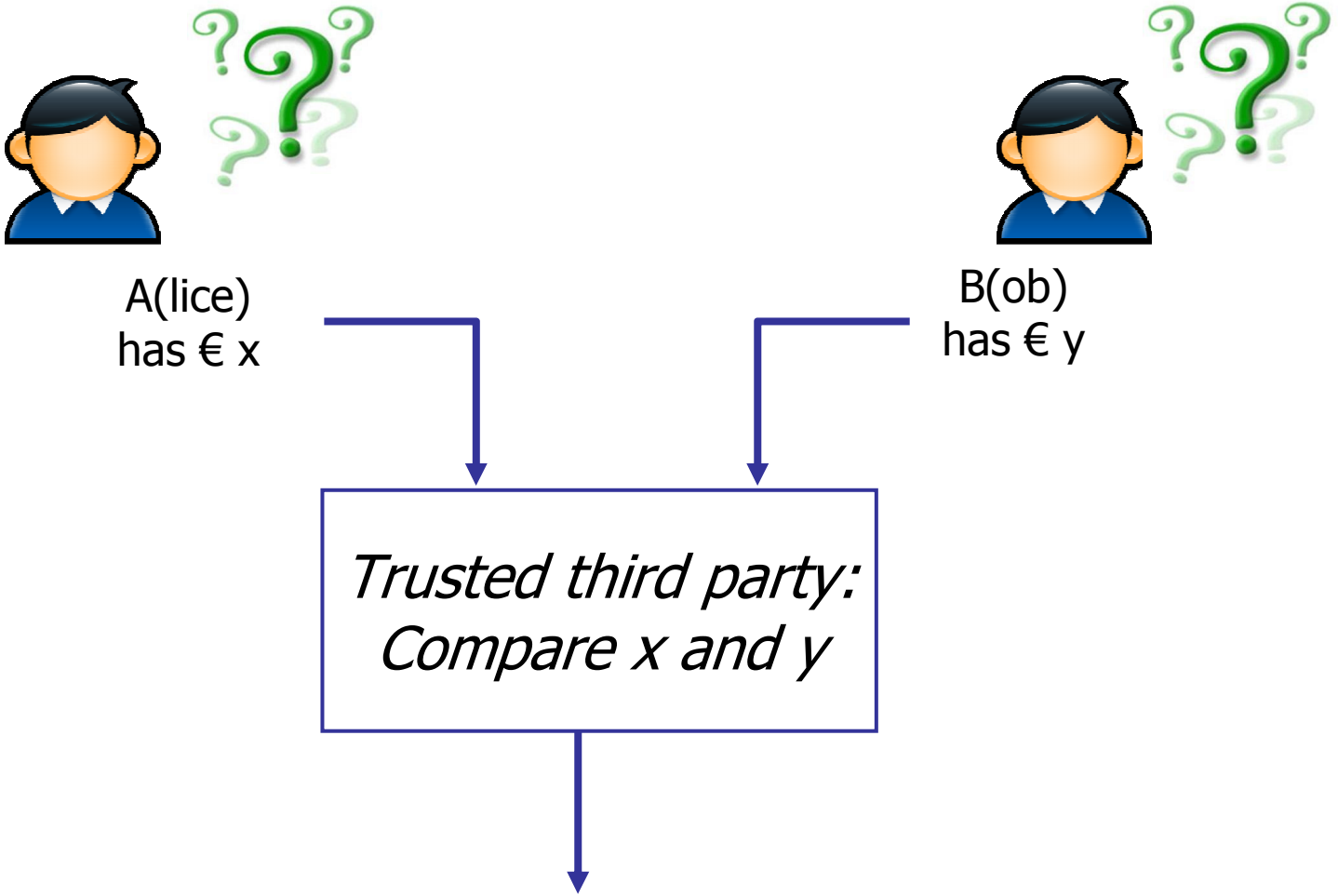
B(ob)
has € y

Who is the richest?

Is x > y or y > x

(*aka the Millionaire Problem*)

Crypto 2.0: The Best-known Example (2)



(a TTP solution is always possible)

Crypto 2.0: The Best-known Example (3)

$$d(x,y) = |x - y| \overset{?}{<} T = 0$$

$E[x]$ versus $E[y]$

A(lice)
has € x

B(ob)
has € y

Prepare a boolean logic circuit

garble (*encrypt*) circuit
garble x

garble y

evaluate circuit

ungarble result

share result

(This is an example of Multiparty Computation (MPC))

Homomorphic Encryption (1)

- Some public cryptosystems preserve some structure after encryption.
- Efficient way of evaluating linear functions on encrypted data (add, multiply).
- One form is additive homomorphism (Paillier, 1999):

$$E[a] * E[b] = E[a + b]$$

$$E[a]^C = E[Ca]$$

- This allows for calculating additions and product with a constant C directly on encrypted data without decrypting the data first.

Homomorphic Encryption (2)

- Example

$$\omega = 2x_1 + 9x_2 + 5x_3$$

$$E[\omega] = E[2x_1 + 9x_2 + 5x_3]$$

$$= E[2x_1].E[9x_2].E[5x_3]$$

$$= E[x_1]^2.E[x_2]^9.E[x_3]^5$$

- *Note*. If the numbers are not integers, scaling and rounding is necessary (fixed point data type).

Paillier Cryptosystem: Additive Homomorphic (1)

- Based on difficulty to factor a large number in primes

$$n = p \cdot q$$

public key (g, n)

private key (p, q)

$$E_{pk}(m) = [m] = g^m \cdot r^n \bmod n^2$$

$$[m_1] \times [m_2] = g^{m_1} \cdot r_1^n \times g^{m_2} \cdot r_2^n \bmod n^2$$

$$[m_1 + m_2] = g^{m_1 + m_2} \cdot (r_1 \times r_2)^n \bmod n^2$$

- Encryption of the plain text sum is the product of the two encrypted values.

Paillier Cryptosystem: Additive Homomorphic (2)

- Paillier: $E[m] = [m] = g^m r^n \bmod n^2$

- Properties:
  - Semantic security. A computationally bounded adversary cannot derive meaningful information from observing plain text and cipher text.
  - Random parameter *r* plays an important role because in signal processing inputs are usually of small range (8-16 bits).
  - Example:
    - $[5] = 3^6 7^{15} \bmod 225 = 99$ ($225 = 15^2 = (3 \times 5)^2$)
    - $[5] = 3^6 8^{15} \bmod 225 = 126$
    - Cipher text is uniform over range $(0, n^2 - 1)$ as random number changes.
  - In practice, *p*, *q*, *n* are 512, 1024 or more bits long.
  - Observe and remember the data expansion.

RSA Cryptosystem: Multiplicative Homomorphic

- Based on difficulty to factor a large number in primes.

$n = p.q$

$d.e \equiv 1 \bmod (p\text{-}1)(q\text{-}1)$

public key (n, e)

private key (d)

$E[m] = c = m^e \bmod n$

$D[c] = m = c^d \bmod n$

- $[m_1] \times [m_2] = m_1^e\, m_2^e \bmod n = (m_1\, m_2)^e \bmod n = [m_1 \times m_2]$

- Multiplicative homomorphic.
- Not semantically secure (no randomness factor).

What Can We Do With This?

- Additive homomorphism

$$E[a] * E[b] = E[a+b] \qquad [a] \text{ x } [b] = [a+b]$$

$$E[a]^C = E[Ca] \qquad [a]^C = [Ca]$$

- This can be used for linear operations of the following kind:

$$f(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{N} x_i \; y_i$$

$$[f(\boldsymbol{x}, \boldsymbol{y})] = \langle [\boldsymbol{x}], \boldsymbol{y} \rangle = \prod_{i=1}^{N} [x_i]^{y_i}$$

➔ Inner products, projections, linear signal transformations (DCT).

What Can We Not Do With This?

- If both **x** and **y** are encrypted …

$$[f(\boldsymbol{x},\boldsymbol{y})] = \langle [\boldsymbol{x}], [\boldsymbol{y}] \rangle = \underbrace{[x_1] \times [y_1]}_{\textit{multiplication}} + \underbrace{[x_2] \times [y_2] + \cdots + [x_N] \times [y_N]}_{\textit{addition}}$$

- … a fully algebraically homomorphic cryptosystem is needed.

- Solution exist (C. Gentry, 2009) but completely impractical.
- Even Paillier and RSA are computationally intense due to power raising operations.
- Also: Non-linear operations require secure MPC solutions.

Blinding is Useful too

- Alice calculates the product of [a] and [b] to [a.b], but Bob has the secret key *sk*.

[*a*], [*b*], *pk*

sk

$$[\tilde{a}] = [a + r_a] = [a] \times [r_a]$$

$$[\tilde{b}] = [b + r_b] = [b] \times [r_b]$$

$[\tilde{a}]$ $[\tilde{b}]$ →

decrypt

$$\tilde{a}.\tilde{b} = (a + r_a).(b + r_b)$$

$$= ab + ar_b + br_a + r_a r_b$$

← $[\tilde{a}.\tilde{b}]$

encrypt

$$[a.b] = [\tilde{a}.\tilde{b}] \times [a]^{-r_b} \times [b]^{-r_a} \times [r_a . r_b]^{-1}$$

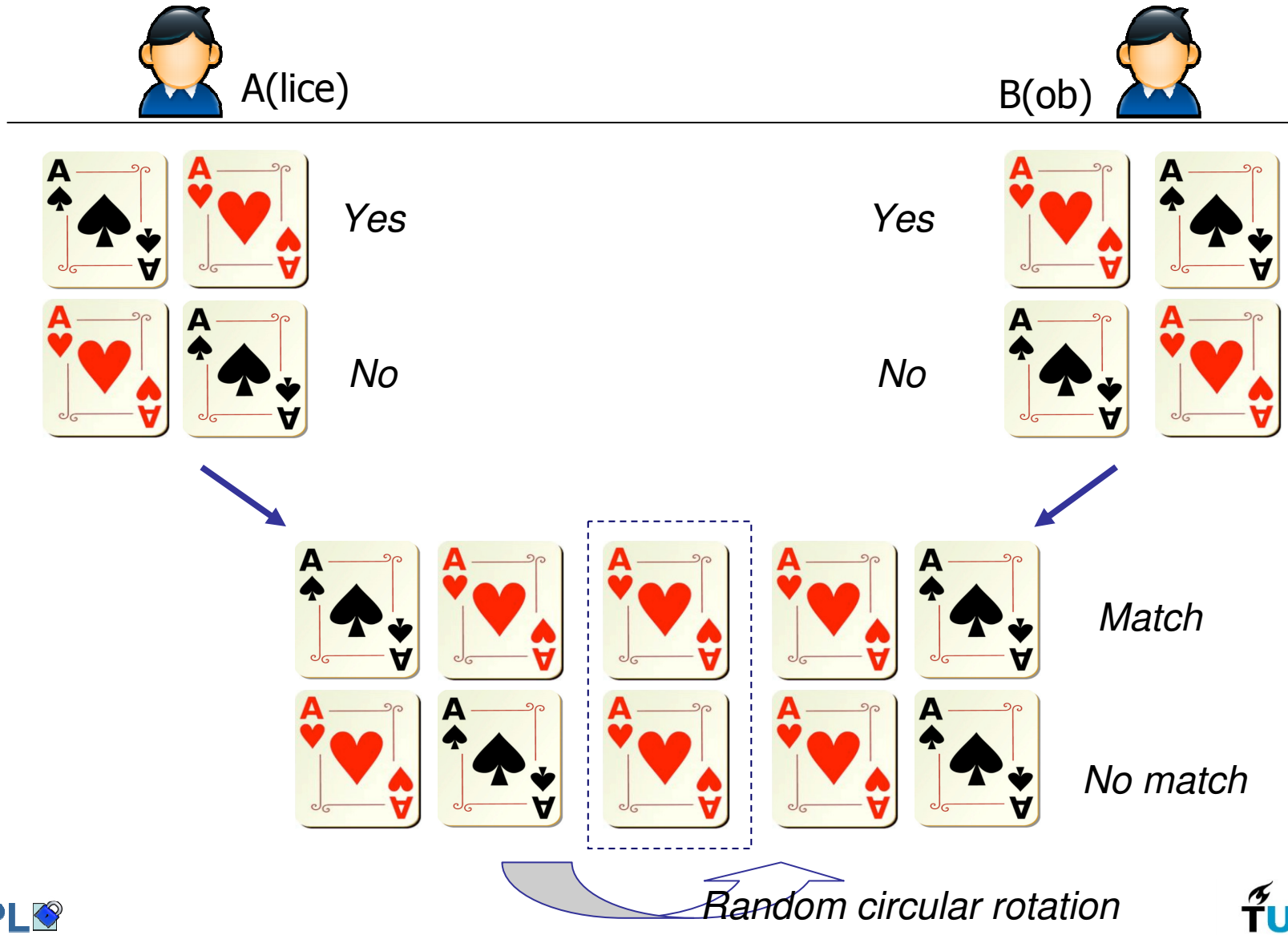
Secure Multiparty Computation (MPC)

- Procedures to compute a function $f(x_1,x_2,...,x_N)$
  - Public function.
  - Secret inputs.
  - No one should see the inputs of the others.
  - Joint result, known to one or many.

- Example: The "love" game.
  - A(lice) and B(ob), do they like each other?
  - Results:
    - Both like each other, learn the input of the other (yes/no).
    - If one likes the other, learn if the other agrees (yes/no).
    - If one does not like the other, keep opinion of the other hidden.

- Note: this is the Boolean function: Result = A *and* B

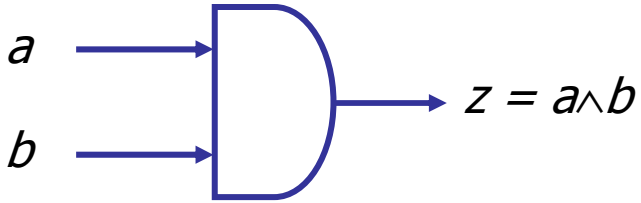
The "Love" Game Protocol



Garbled Circuits for Two-party MPC

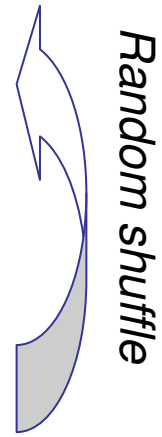
- Procedure to compute a function $f(x_1,x_2)$.
- Alice:
  - Creates a Boolean circuit of the function $f(a,b)$.
  - Garbles the truth table of every gate
    - Assign 2 keys for each input (usually called: *input wire*): (K_{a0}, K_{a1}) and (K_{b0}, K_{b1}).
    - Shuffle (randomize) the order of the truth table rows.
  - Send the key and the circuit to Bob.
- Bob:
  - Get the circuit and input of Alice (key K_{a0} or K_{a1}).
  - Needs Oblivious Transfer (OT) to get correct key K_{b0} or K_{b1}.
  - Evaluate the circuit using his own inputs (key K_{b0} or K_{b1})
  - Report outcome to Alice.

Garbled Circuit AND Function



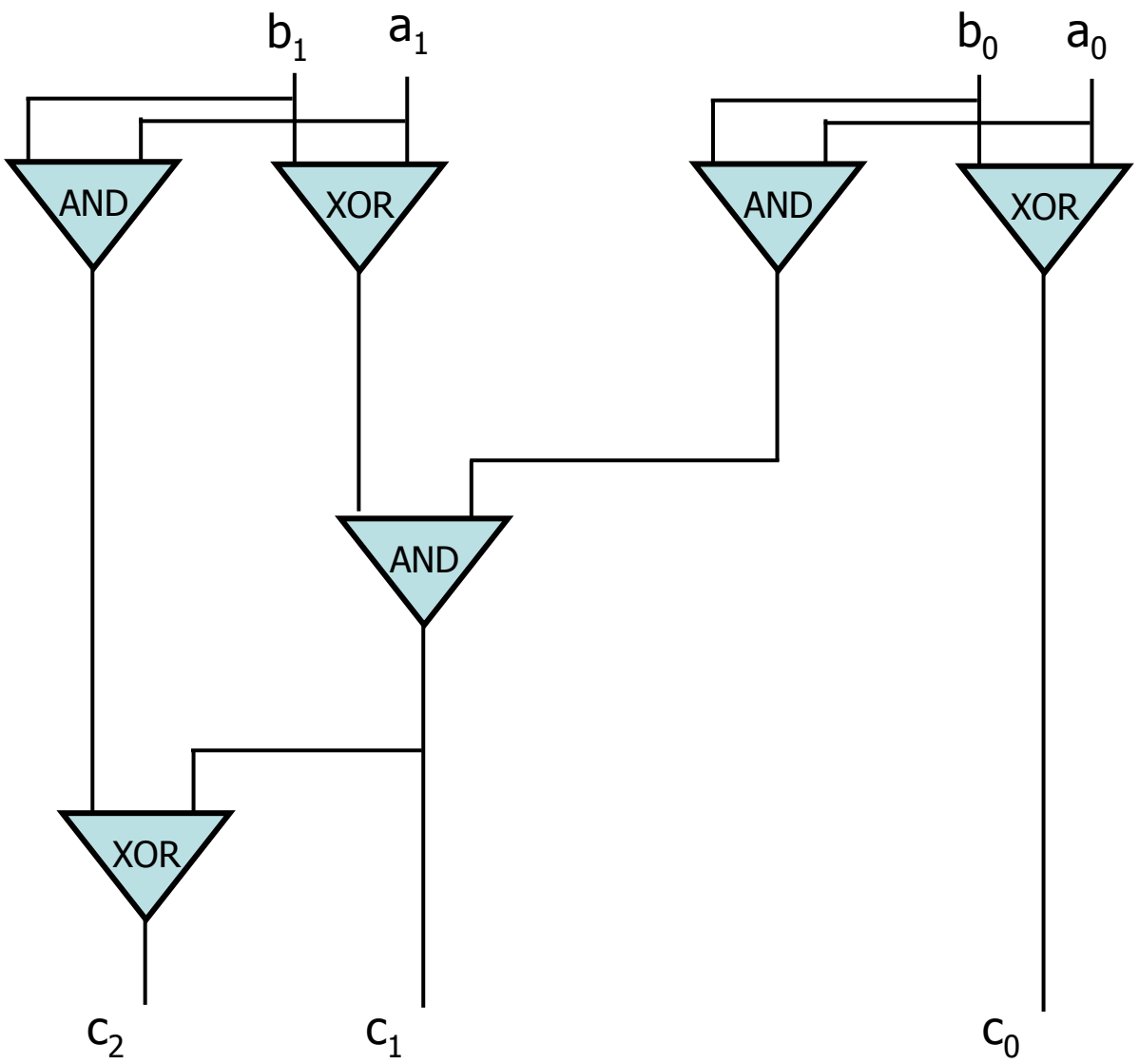
a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

a	b	$a \wedge b$
K_{a0}	K_{b0}	$E_{K_{a0}}(E_{K_{b0}}(K_{z0}))$
K_{a0}	K_{b1}	$E_{K_{a0}}(E_{K_{b1}}(K_{z0}))$
K_{a1}	K_{b0}	$E_{K_{a1}}(E_{K_{b0}}(K_{z0}))$
K_{a1}	K_{b1}	$E_{K_{a1}}(E_{K_{b1}}(K_{z1}))$



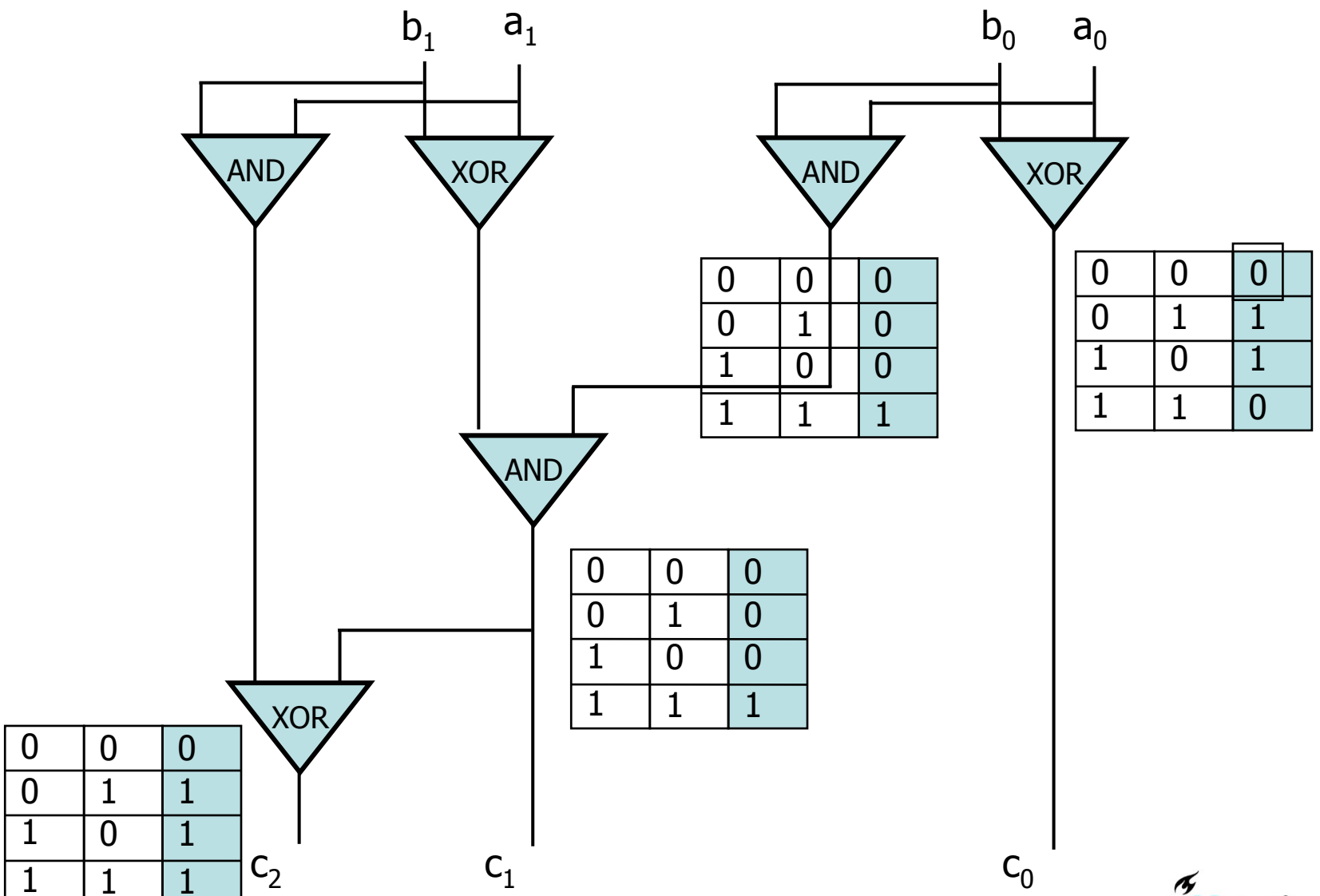
Function: $c_2c_1c_0 = f(a_1a_0, b_1b_0)$

This is the function Alice and Bob wish to evaluate



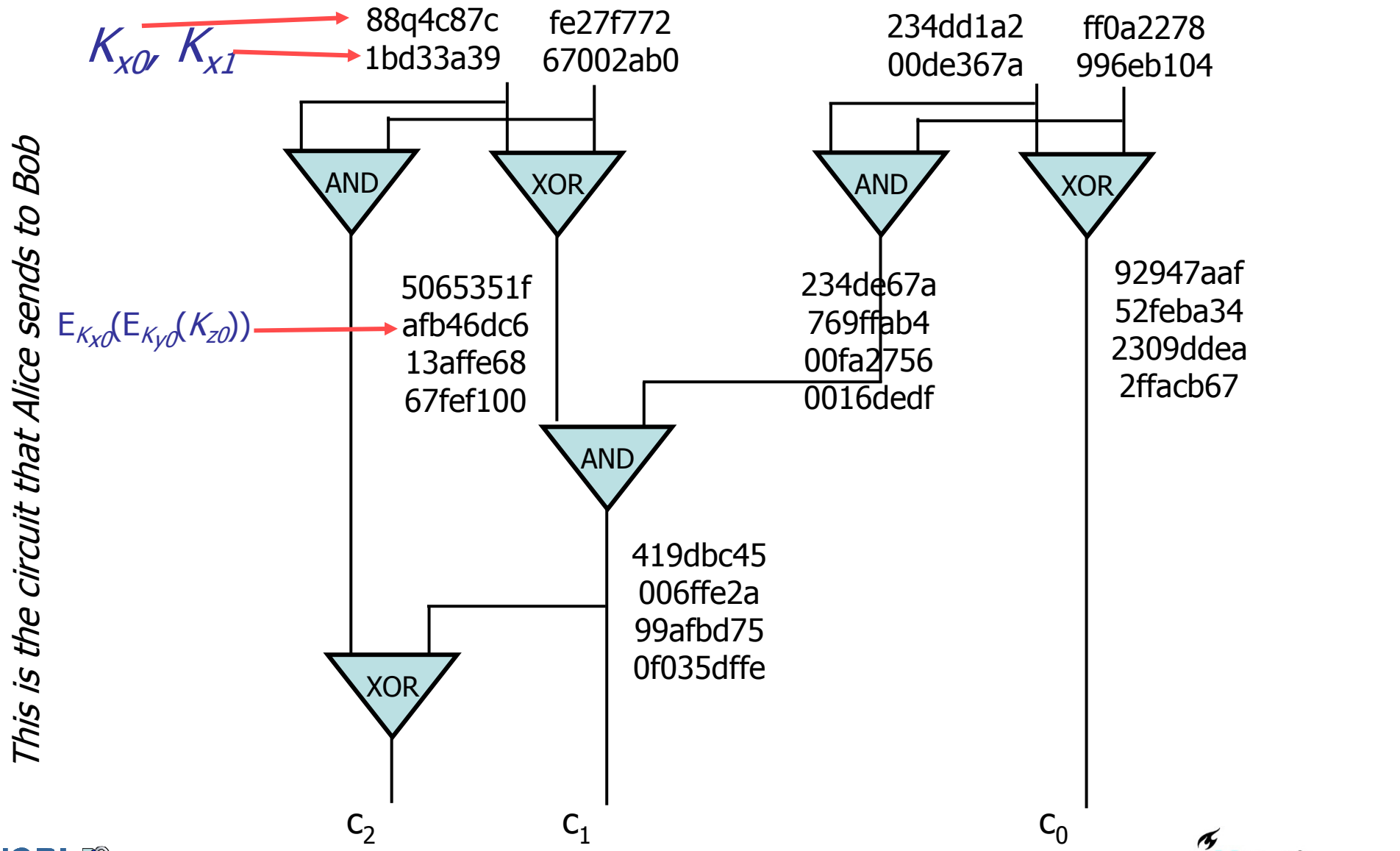
Function: $c_2c_1c_0 = f(a_1a_0, b_1b_0)$

This is the function Alice and Bob wish to evaluate

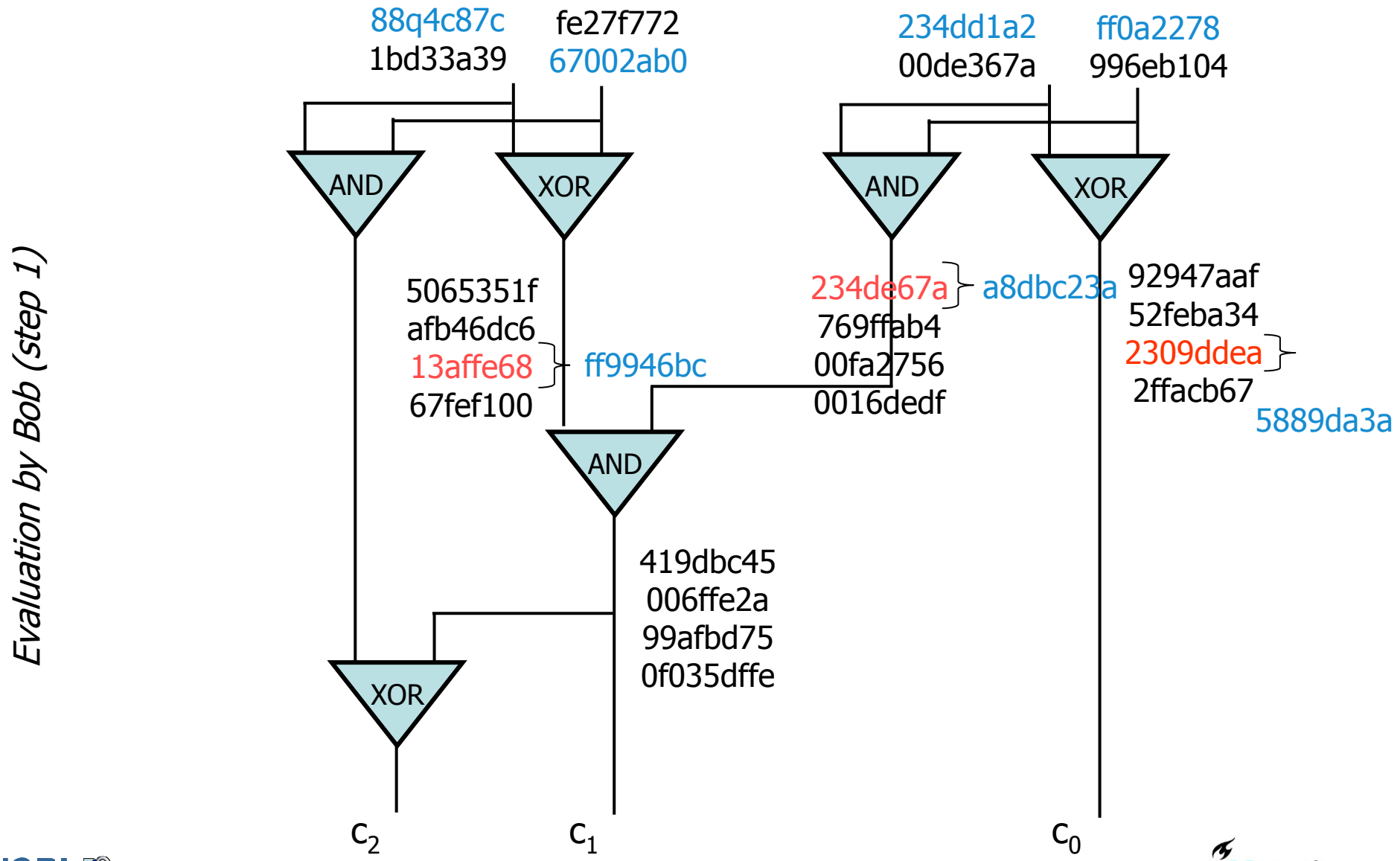


(*32 bit keys*)

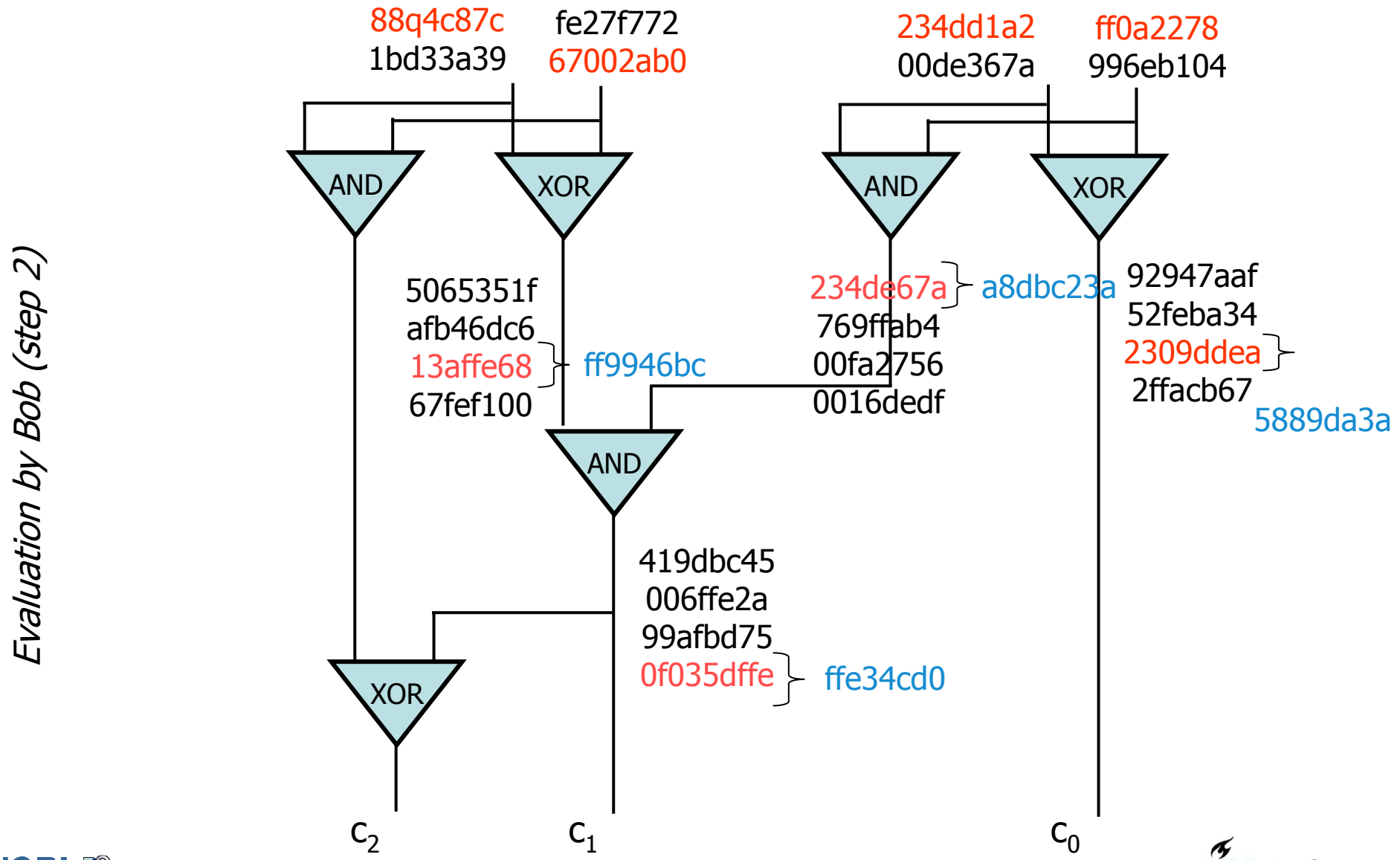
Function: $c_2c_1c_0 = f(a_1a_0, b_1b_0)$



Function: $c_2c_1c_0 = f(a_1a_0, b_1b_0)$



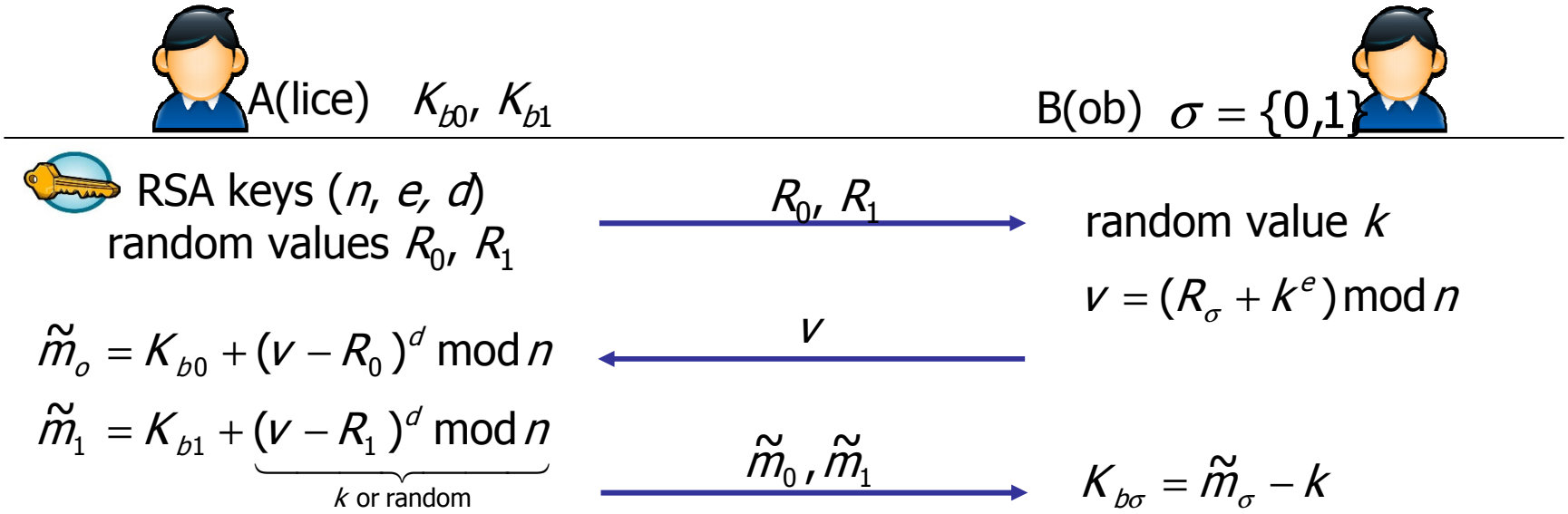
Function: $c_2c_1c_0 = f(a_1a_0, b_1b_0)$



Oblivious Transfer (One-out-of-Two)

- Bob needs the key K_{b0} or K_{b1} corresponding to his input 0/1.
- Cannot plainly ask Alice, or Alice would know input of Bob.
- Need oblivious transfer (in this case: one-out-of-two).

- OT: Procedure to let Alice select one value (key in this case) out of two without knowing which one was selected.



Oblivious Transfer and Garbled Circuits

- Garbled circuits are relatively efficient.

- But the OT requires:
  - for each wire (bit) ...
  - key generation ...
  - public key operations (RSA encryption and decryption) ...
  - data transmission ...

- Even with precomputation (e.g. keys, random numbers) this is fairly computationally expensive.

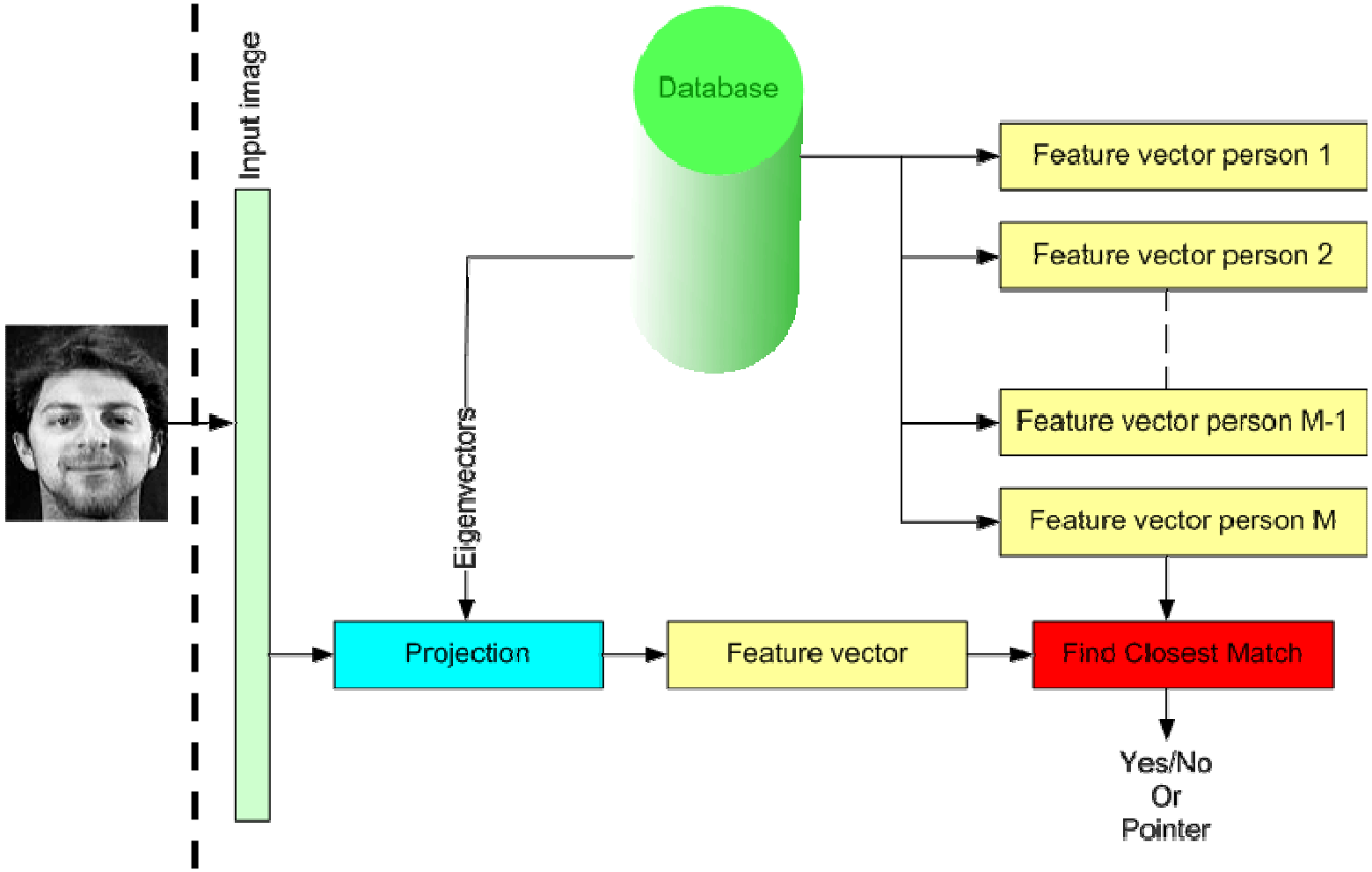
- Hybrid approaches:
  - Linear ➔ homomorphism.
  - Non-linear ➔ garbled circuits.

Content of the Lecture

- Introduction (done)
- Cryptography 2.0. (done)
  - Homomorphic cryptography.
  - Secure multiparty computation.
- Secure face recognition. (start at 2pm)
- Secure recommender system.
- Challenges. (wrap up around 3:30 pm)

Face Recognition: Blessing or Threat?

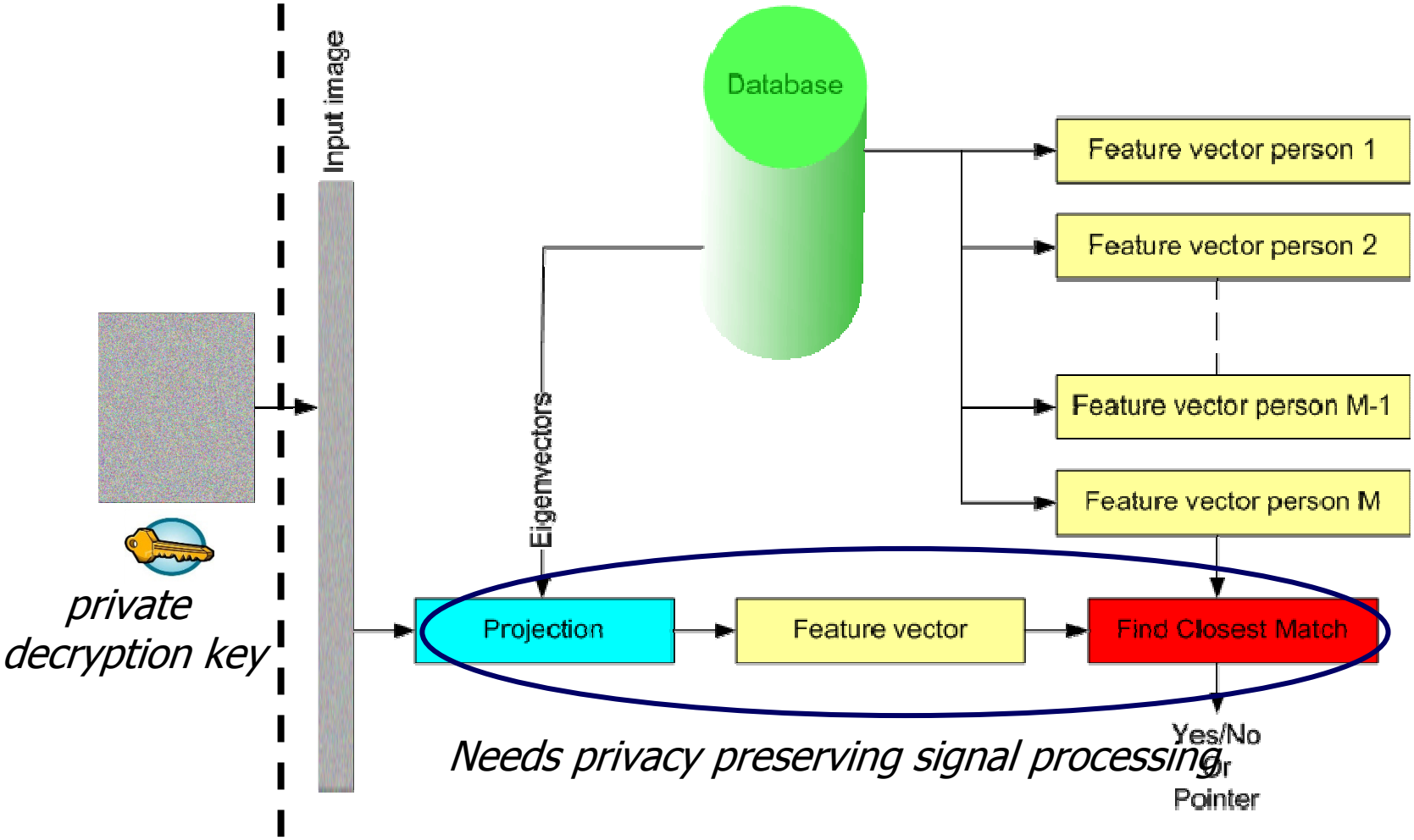
A 'Normal' Face Recognition System



Query (Alice)

Face database and recognition (Bob)

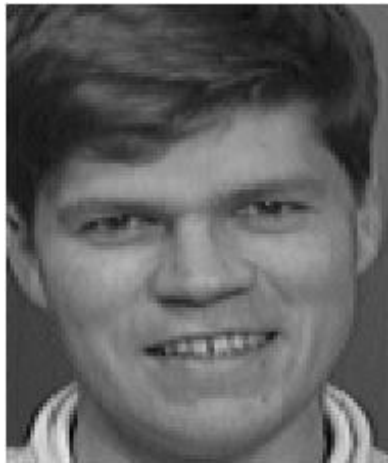
Desired Secure Version



Query (Alice)

Face database and recognition (Bob)

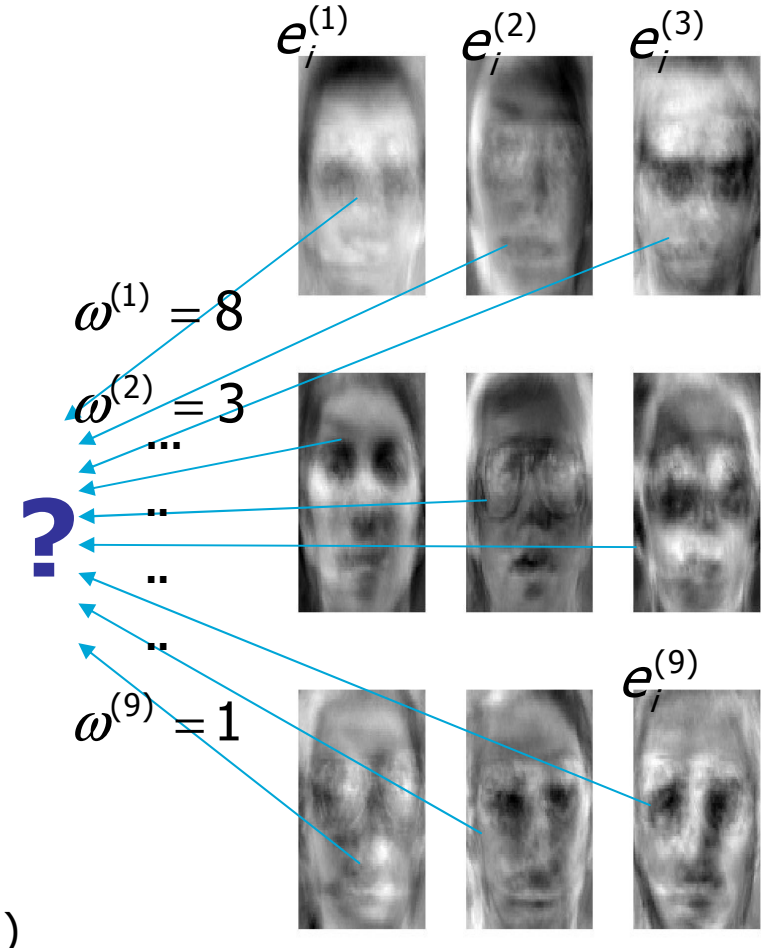
Face Recognition using Eigenfaces



Query face x

Projection

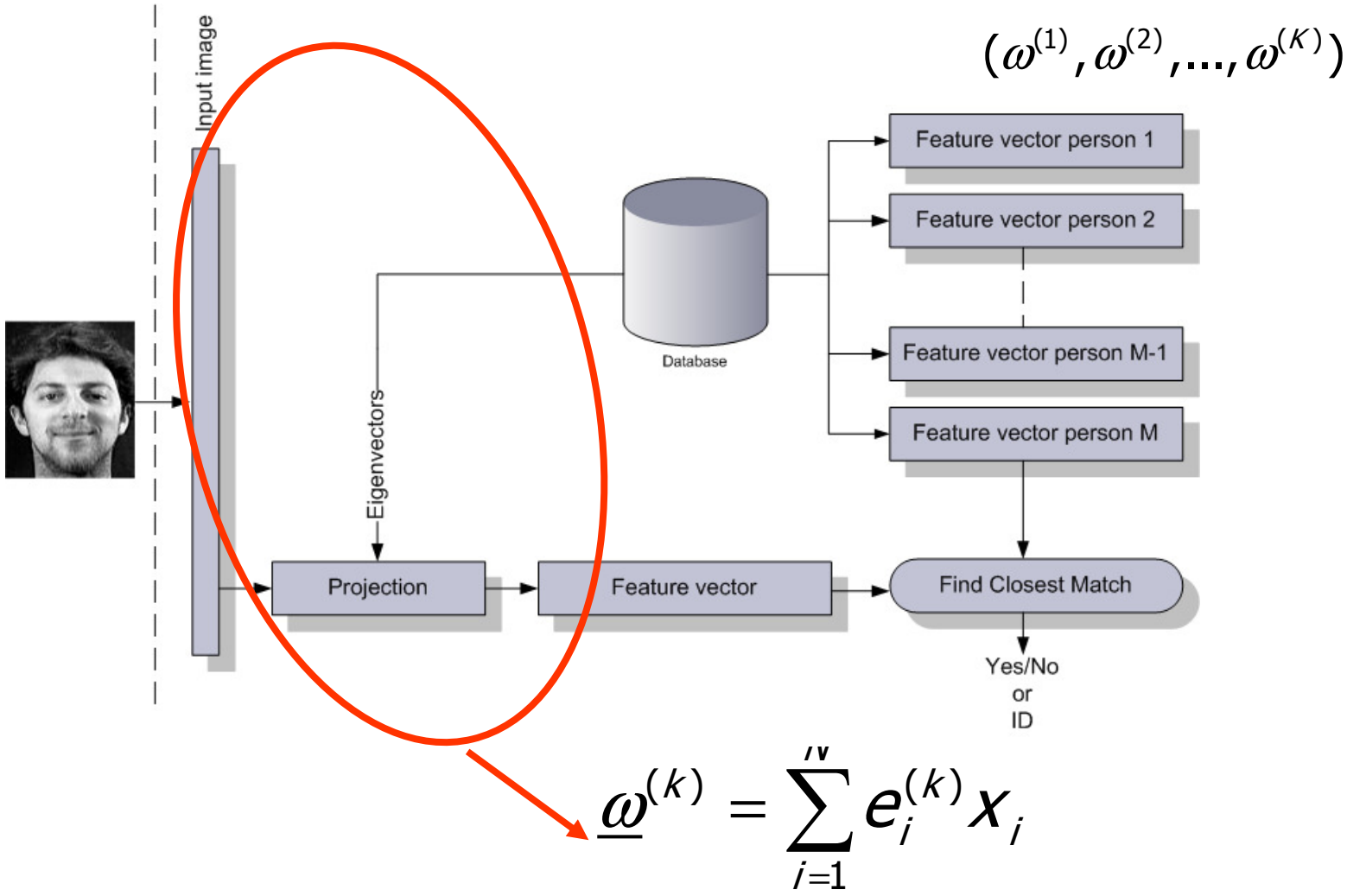
Projected face



$$x \approx \sum_{k=1}^{K} \omega^{(k)} e^{(k)}$$

The ω-values are the eigenvalues, weights, or feature values in the database

Work load in SFR: Linear Operations



Query (Alice)

Face database and recognition (Bob)

Linear Operations on Encrypted Data (1)

- Exploit homomorphism for these additive operation (Paillier)

$$[a] \times [b] = [a+b]$$
$$[a]^C = [C\,a]$$

- This allows the calculation of features $\omega^{(k)}$ -- the projection on eigenvectors $\boldsymbol{e}^{(k)}$ -- on *encrypted images* $\boldsymbol{x}$:

$$\overline{\omega}^{(k)} = \sum_{i=1}^{N} e_i^{(k)} x_i \quad \Leftrightarrow \quad \left[\overline{\omega}^{(k)}\right] = \prod_{i=1}^{N} \left[x_i\right]^{e_i^{(k)}}$$

- Allows for calculating encrypted features directly on encrypted image pixels x_i.

But …

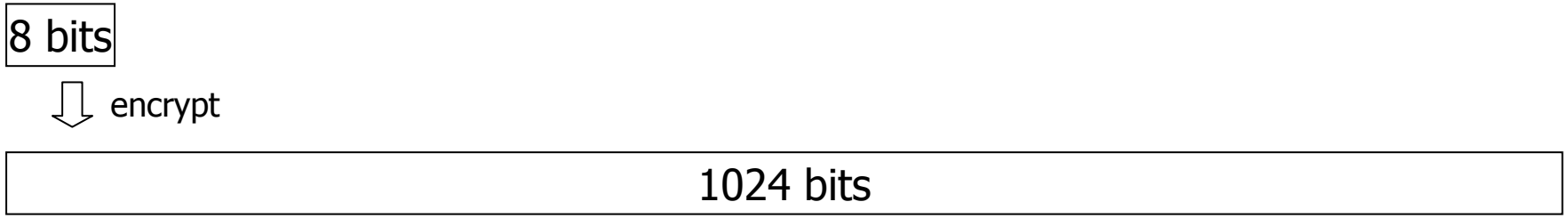
1. Public key encryption (8 bit pixel ➔ 1024 bit encrypted value)
   - Counteract by stacking multiple pixels before encryption

2. Only integer values of $\boldsymbol{e}^{(k)}$ and images $\boldsymbol{x}$.
   - Counteract by scaling (e.g. factor 1000) to work with fixed point data type instead of rational numbers).

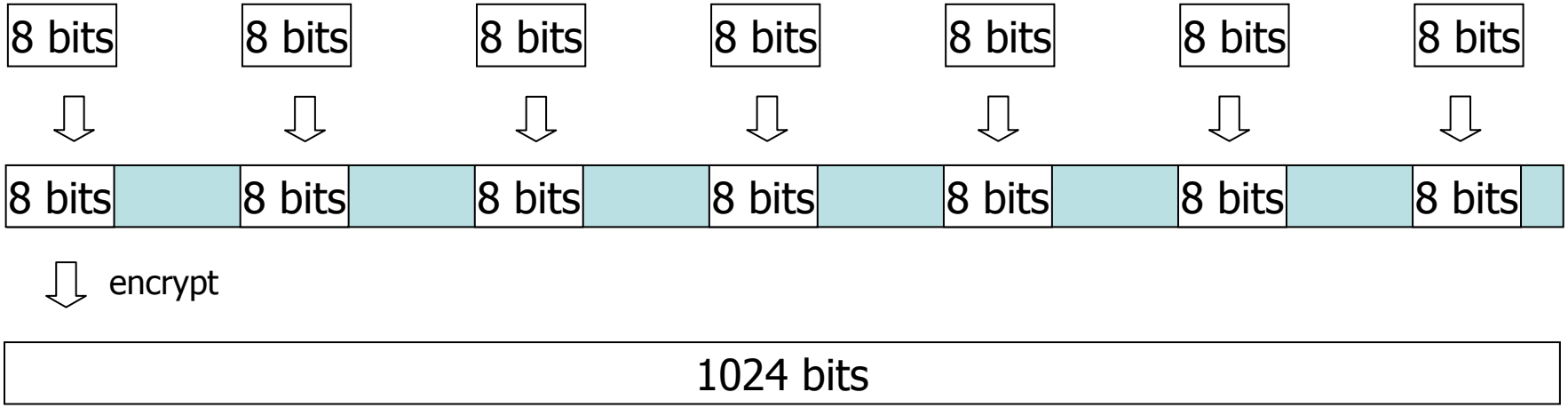
3. Matching is a non-linear operation.
   - Euclidean distance calculation.
   - Find the "closest" face, and determine if it is "close" enough.

Packing of Pixel Values

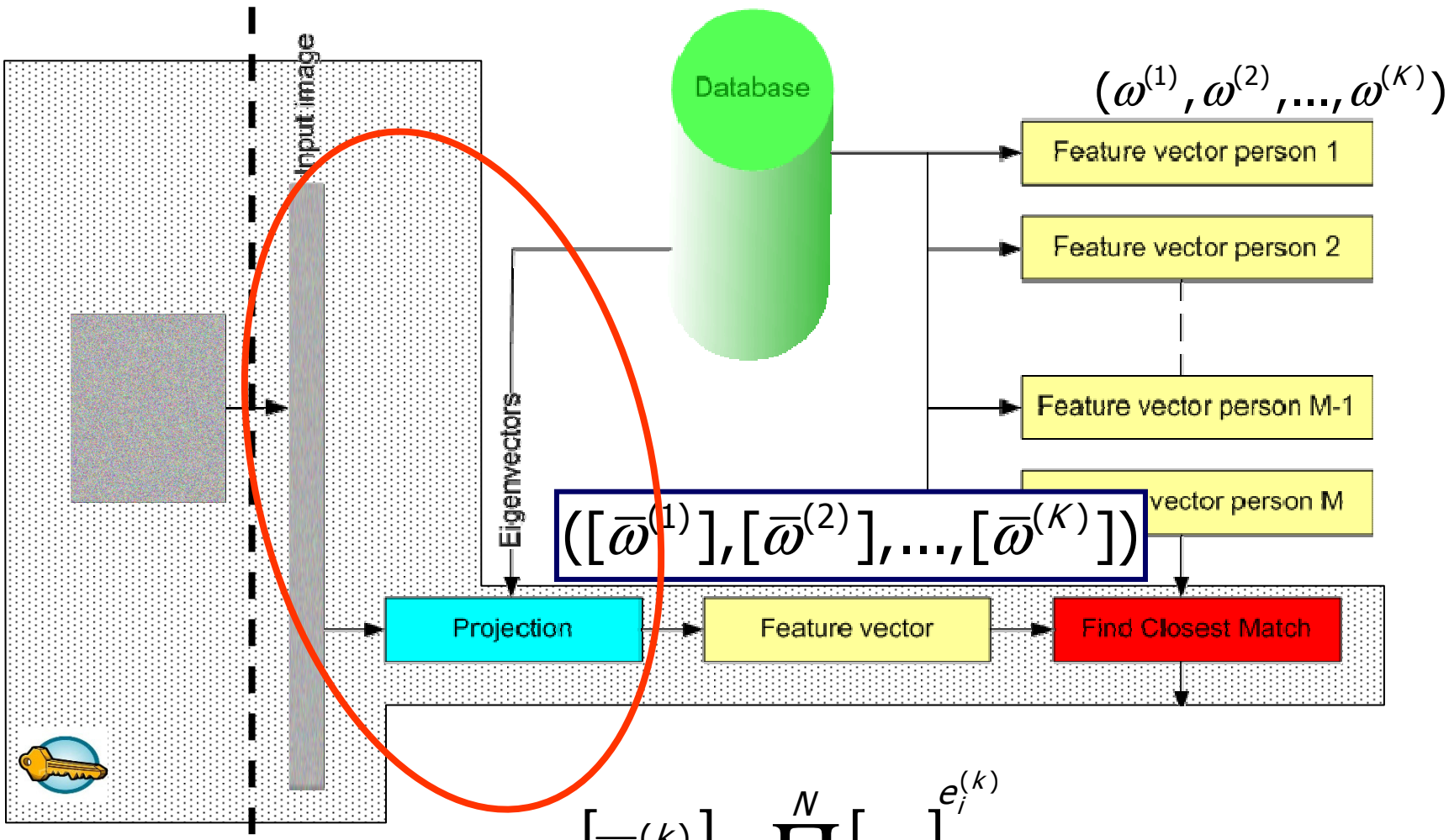
- Naïve approach: pixel-by-pixel



- Pixel packing



Secure Version of Face Recognition



$$\left[\bar{\omega}^{(k)}\right] = \prod_{i=1}^{N} \left[x_i\right]^{e_i^{(k)}}$$

Calculating (Encrypted) Distances (1)

- Euclidean distance between
  - Feature vector (encrypted) for input image:

$$[\bar{\Omega}] = ([\bar{\omega}^{(1)}],[\bar{\omega}^{(2)}],\ldots,[\bar{\omega}^{(K)}])$$

  - And feature vector from the database

$$\Omega = (\omega^{(1)}, \omega^{(2)}, \ldots, \omega^{(K)})$$

- Let us start with the case K=1 …

Calculating (Encrypted) Distances (2)

- In case K=1 (remember, only $[\overline{\omega}]$ is encrypted).

$$\left\|\Omega-\overline{\Omega}\right\|^2=(\omega-\overline{\omega})^2=\omega^2-2\omega\overline{\omega}+\overline{\omega}^2$$

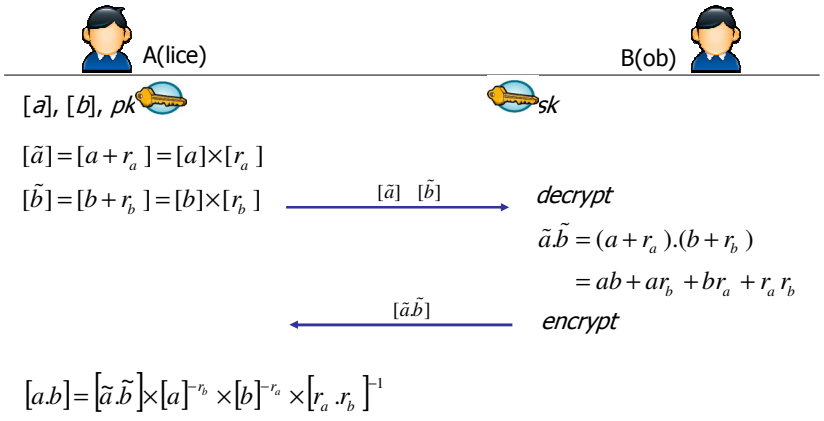
$$\left[\left\|\Omega-\overline{\Omega}\right\|^2\right]=\left[(\omega-\overline{\omega})^2\right]=\left[\omega^2-2\omega\overline{\omega}+\overline{\omega}^2\right]$$

$$=\left[\omega^2\right]\left[\overline{\omega}\right]$$

- Easy: $\left[\omega^2\right]$ Database se… o encrypt ω^2.
- Easy: $\left[\overline{\omega}\right]^{-2\omega}$ Bob raised e… power -2ω
- Harder: $\left[\overline{\omega}\right]^2$ Multiplicatio… . Remember the protocol based on blinding?

Blinding is Useful too

- Alice calculates the product of [a] and [b] to [a.b], but Bob has the secret key *sk*.



Calculating (Encrypted) Distances (3)

- Euclidean distance between
  - Feature vector (encrypted) for input image:

$$[\bar{\Omega}] = ([\bar{\omega}^{(1)}],[\bar{\omega}^{(2)}],\ldots,[\bar{\omega}^{(K)}])$$

  - And feature vector from the database

$$\Omega = (\omega^{(1)}, \omega^{(2)}, \ldots, \omega^{(K)})$$

$$\begin{aligned} ||\,\Omega - \bar{\Omega}\,||^2 &= (\omega^{(1)} - \bar{\omega}^{(1)})^2 + (\omega^{(2)} - \bar{\omega}^{(2)})^2 + \ldots + (\omega^{(K)} - \bar{\omega}^{(K)})^2 \\ &= (\omega^{(1)})^2 + (\omega^{(2)})^2 + \ldots + (\omega^{(K)})^2 \\ &\quad -2\omega^{(1)}\bar{\omega}^{(1)} - 2\omega^{(2)}\bar{\omega}^{(2)} - \ldots - 2\omega^{(K)}\bar{\omega}^{(K)} \\ &\quad + \underbrace{(\bar{\omega}^{(1)})^2 + (\bar{\omega}^{(2)})^2 + \ldots + (\bar{\omega}^{(K)})^2}_{} \end{aligned}$$

only encrypted values $\bar{\omega}^{(k)}$ available => need interactive protocol

Calculating (Encrypted) Distances (4)


Query (Alice)	Database server (Bob)
	Generate random: r
	$[x] = [\bar{w}_i] \cdot [r]$
	$[x]$ ⟵
	Remark: $[x^2] = [(\bar{w}_i + r)^2] = [\bar{w}_i{}^2 + 2\bar{w}_i r + r^2]$
$x \Leftarrow [x]$	
Calculate $x^2 \Rightarrow [x^2]$	
	$[x^2]$ ⟶
	$[\bar{\omega}_i{}^2] = [x^2] \cdot [\bar{\omega}_i]^{(-2r)} \cdot [-r^2]$

- For the calculation of the distance under encryption we find:

$$\left[|| \Omega - \bar{\Omega} ||^2 \right] = \underbrace{\left[\sum_{i=1}^{K} (\omega^{(i)})^2 \right]}_{\text{easy}} \cdot \underbrace{\left(\prod_{i=1}^{K} [\bar{\omega}^{(i)}]^{-2\omega^{(i)}} \right)}_{\text{easy}} \cdot \underbrace{\left(\prod_{i=1}^{K} [(\bar{\omega}^{(i)})^2] \right)}_{\text{above protocol}}$$

Find the Minimum of the Encrypted Distances

- Several encrypted values:
  - Find the minimum.
  - But database server (Bob) should not know which feature vectior (person) yields the minimum.

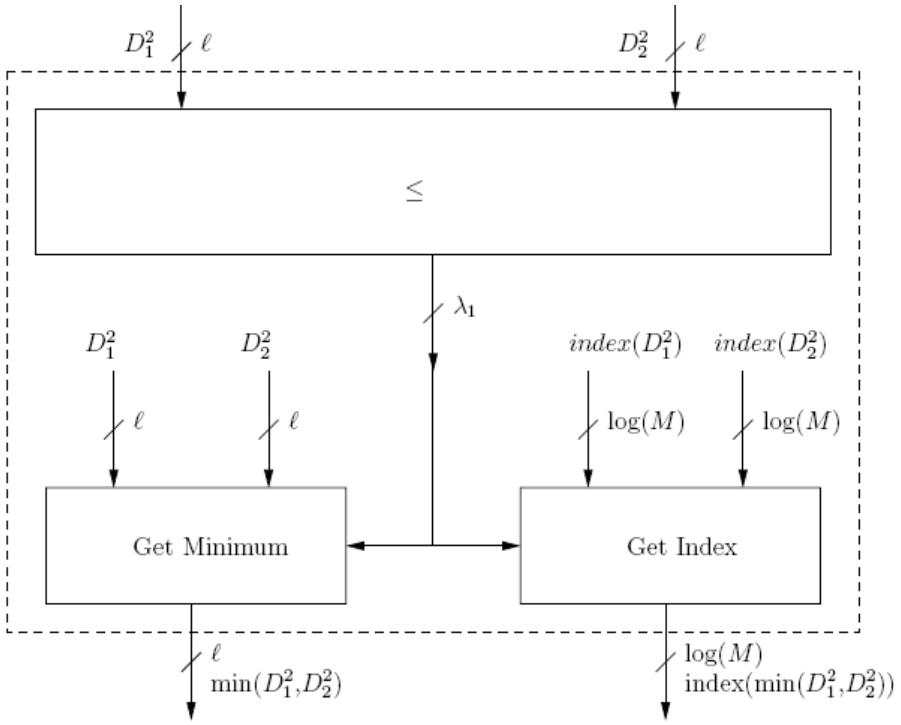
- Much like the Millionaire Problem.
  - Use garbled circuit to solve.
    (see next slide, no details).
  - Use dedicated protocol.
    (details next).
  - Based on comparing pairs, expand to a binary tree comparison.

- Secure Face Recognition solution becomes hybrid.

Find the Minimum of the Encrypted Distances

Circuit: 32ML x 128 bits = 32 . 320 . 50 . 128 = 6 MByte
Oblivious transfer: 5ML.1024 = 10 MByte

But tricks such as elliptic, #entries per table less,
reduced size of garbled circuit with factor 2-3,
oblivious transfer factor of 10



- A garbled circuit require several MBytes of communication for 320 images of size 112 x 90

Comparison Protocol: $a < b$?

- Setting:
  - Alice (query) has the private decryption key.
  - Bob (database server) has [a] and [b] (ℓ bit each).


A(lice), sk

B(ob), [a] [b]


$$[z] = [2^{\ell} + a - b] = [2^{\ell}].[a].[b]^{-1}$$

Look at the MSB: a>b ➔ "1"
a<b ➔ "0"

$$MSB = \underbrace{(z - z \bmod 2^{\ell})}_{\text{"sign bit" only}} \underbrace{2^{-\ell}}_{\text{shift to right}}$$

Comparison Protocol: $a < b$?

- Example A
  - $a = 100$ (4)
  - $b = 101$ (5)
  - $\ell = 3$

- $z = 2^{\ell} + a - b = 8 + 4 - 5 = 7 = 0111$
- MSB = $(z - z \bmod 2^{\ell}) 2^{-\ell} = (7 - 7 \bmod 8) 2^{-3} = (0000)\ 2^{-3} = 0$

- Example B
  - $a = 101$ (5)
  - $b = 100$ (4)
  - $\ell = 3$

- $z = 2^{\ell} + a - b = 8 + 5 - 5 = 9 = 1001$
- MSB = $(z - z \bmod 2^{\ell}) 2^{-\ell} = (9 - 9 \bmod 8)\ 2^{-3} = (1000)\ 2^{-3} = 1$

Comparison Protocol: $a < b$?

- Setting:
  - Alice (query) has the private decryption key.
  - Bob (database server) has [a] and [b] (ℓ bit each).


A(lice), *sk*

B(ob), [a] [b] 

$$[z] = [2^{\ell} + a - b] = [2^{\ell}].[a].[b]^{-1}$$

$$MSB = \underbrace{(z - z \bmod 2^{\ell})}_{\text{"sign bit" only}} \underbrace{2^{-\ell}}_{\text{shift to right}}$$

Cannot be done, only in encrypted form available.
Alice can decrypt, but should not know z.

Comparison Protocol: $a < b$? (Details I)

- Setting:
  - Alice (query) has the private decryption key.
  - Bob (database server) has $[a]$ and $[b]$ (ℓ bit each).

A(lice), *sk*

B(ob), $[a]$ $[b]$

$$[z] = [2^{\ell} + a - b] = [2^{\ell}].[a].[b]^{-1}$$

$$[c] = [z + r] = [z].\underbrace{[r]}_{\text{B uses pk}}$$

$[c]$ ←

Decrypt $[c]$
encrypt each bit c_i

$[c_0], [c_1] \cdots [c_{\ell-1}]$ →

Compare c with r.
Find first bit r_i =1 ***and*** *c_i=0,*
while all higher up bits are identical.
Result is still encrypted π_i.

Property of π_i (i=0…N-1): if contains zero, than r > c ➔ *result of b < a*

Comparison Protocol: $a < b$? (Details II)

- Setting:
  - Alice (query) has the private decryption key.
  - Bob (database server) has $[a]$ and $[b]$ (ℓ bit each).

A(lice), sk

B(ob), $[a]$ $[b]$

$$[z] = [2^{\ell} + a - b] = [2^{\ell}].[a].[b]^{-1}$$

$[c]$ ←

$$[c] = [z + r] = [z].\underbrace{[r]}_{\text{B uses pk}}$$

Decrypt $[c]$
encrypt each bit c_i

$[c_0], [c_1] \cdots [c_{\ell-1}]$ →

$$[\pi_i] = \left[1 + c_i - r_i + 3\sum_{j=i+1}^{N} c_j \oplus r_j\right]$$

$$= [1].[c_i].[r_i]^{-1}.\prod_{j=i+1}^{N}[c_j]^{3r_j}$$

$[\pi_i]$ ←

Shuffle, multiplicative blinding $[\pi_i]$

Example (Details III)

A(lice), c=01110

B(ob), r=01010

Encrypt each bit c_i

$[c_i]$ →

$[\pi_0] = [1 + 0 - 0 + 0] = [1]$

$[\pi_1] = [1 + 1 - 1 + 0] = [1]$

$[\pi_2] = [1 + 0 - 1 + 0] = [0]$

$[\pi_3] = [1 + 1 - 1 + 1] = [2]$

$[\pi_4] = [1 + 0 - 0 + 1] = [2]$

← $[\pi_i]$

Shuffle, multiplicative blinding $[\pi_i]$

Decrypt each π_i

Zero exists: k=0

Otherwise: k=1

Encrypt k → $[k]$

Interpret "k" (smaller vs larger?)

(Example of comparison protocol, variations exist)

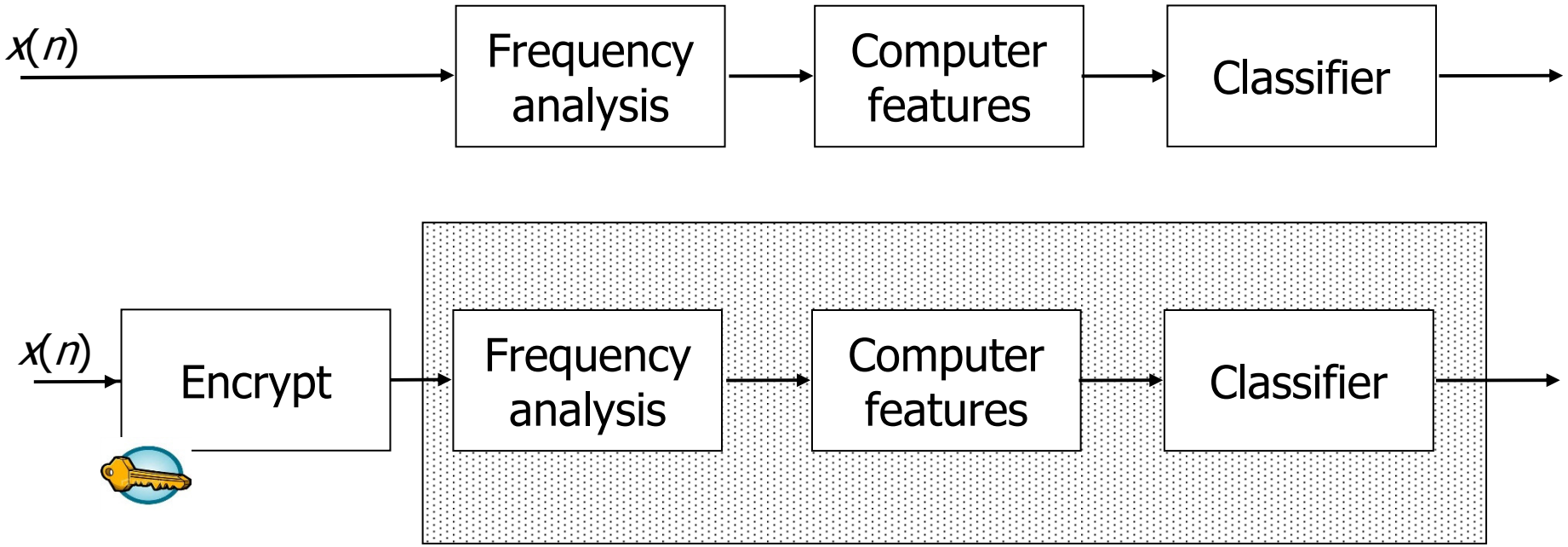
Evaluation

- Query image is encrypted pixel-by-pixel.
- Recognition algorithm is run by server on encrypted image.
- Feature calculation 'easy' thanks to homomorphic properties.
- Additional tweaking: scaling, packing, precomputation, DGK encryption for small message space.
- Interactive protocols required for:
  - Distance calculation (square of encrypted number)
  - Finding minimum of distances (and compare to threshold)

- Timing:
  - Integer arithmetic
    - 400 images (112x92): 18 seconds.
  - Hybrid approach
    - 1000 images: 13 seconds.

Sidetrack a Little Bit …

- Feature vectors are computed via a linear transform.
- Another well-known linear transform is the Discrete Fourier Transform.



Carry out on encrypted samples?

Popular Transform in DSP

- Fourier transform

$$X(k) = \sum_{n=0}^{M-1} x(n) W^{nk} \qquad k = 0 \ldots M-1$$

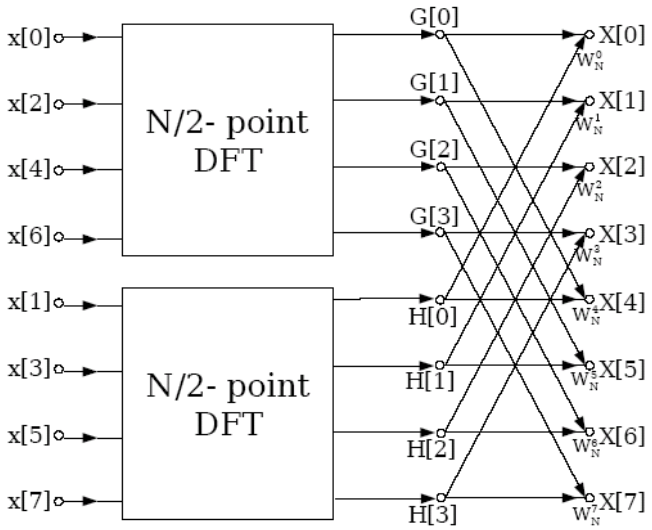
$$W = e^{-j2\pi/M}$$

- What if … the signal samples are encrypted under a homomorphic crypto system?
- Straightforward yet naïve:

$$[X(k)] = \left[\sum_{n=0}^{M-1} x(n) W^{nk} \right] = \prod_{n=0}^{M-1} [x(n)]^{W^{nk}}$$

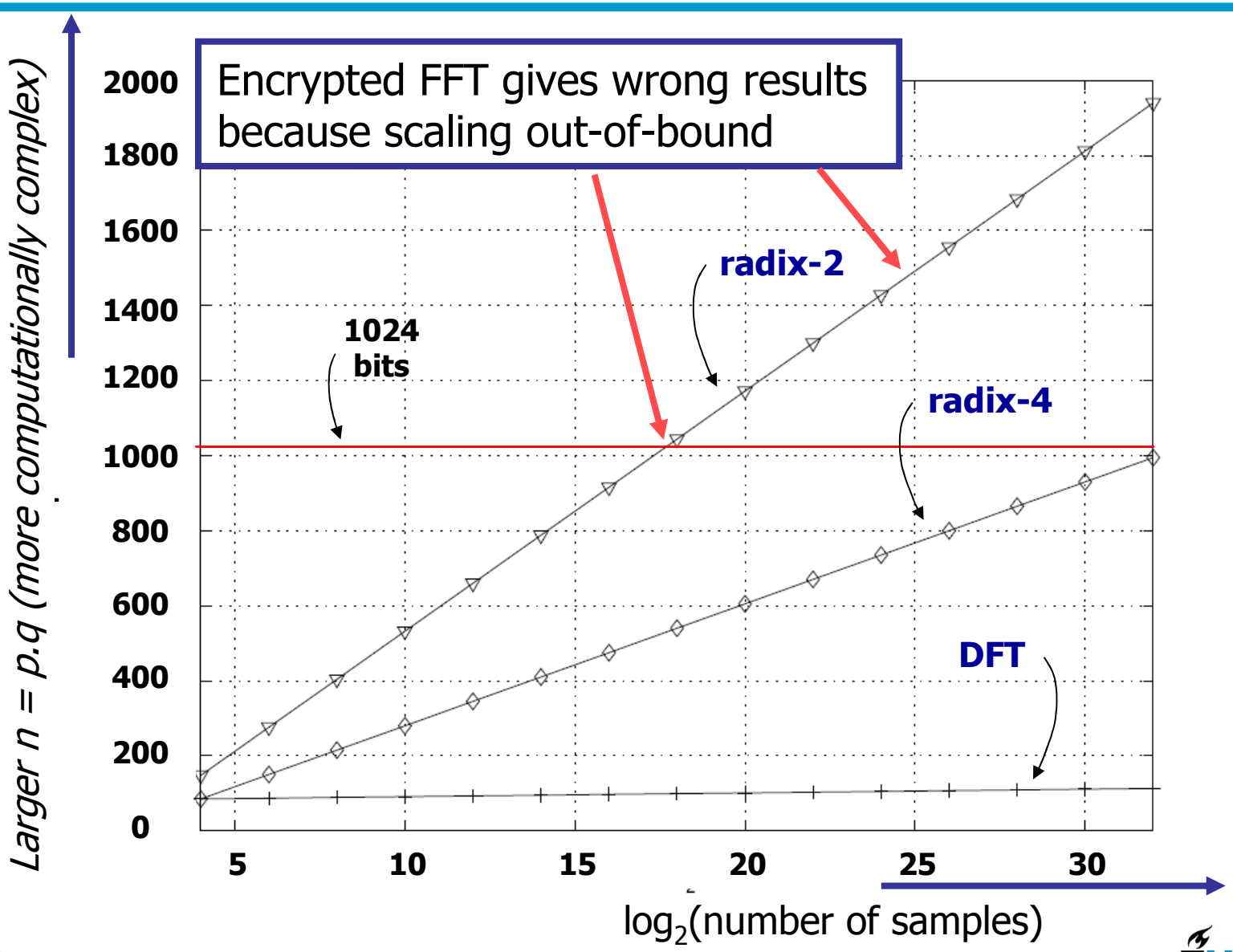
Radix-2 Implementation

- DFT is usually implemented as Fast Fourier Transform (FFT).

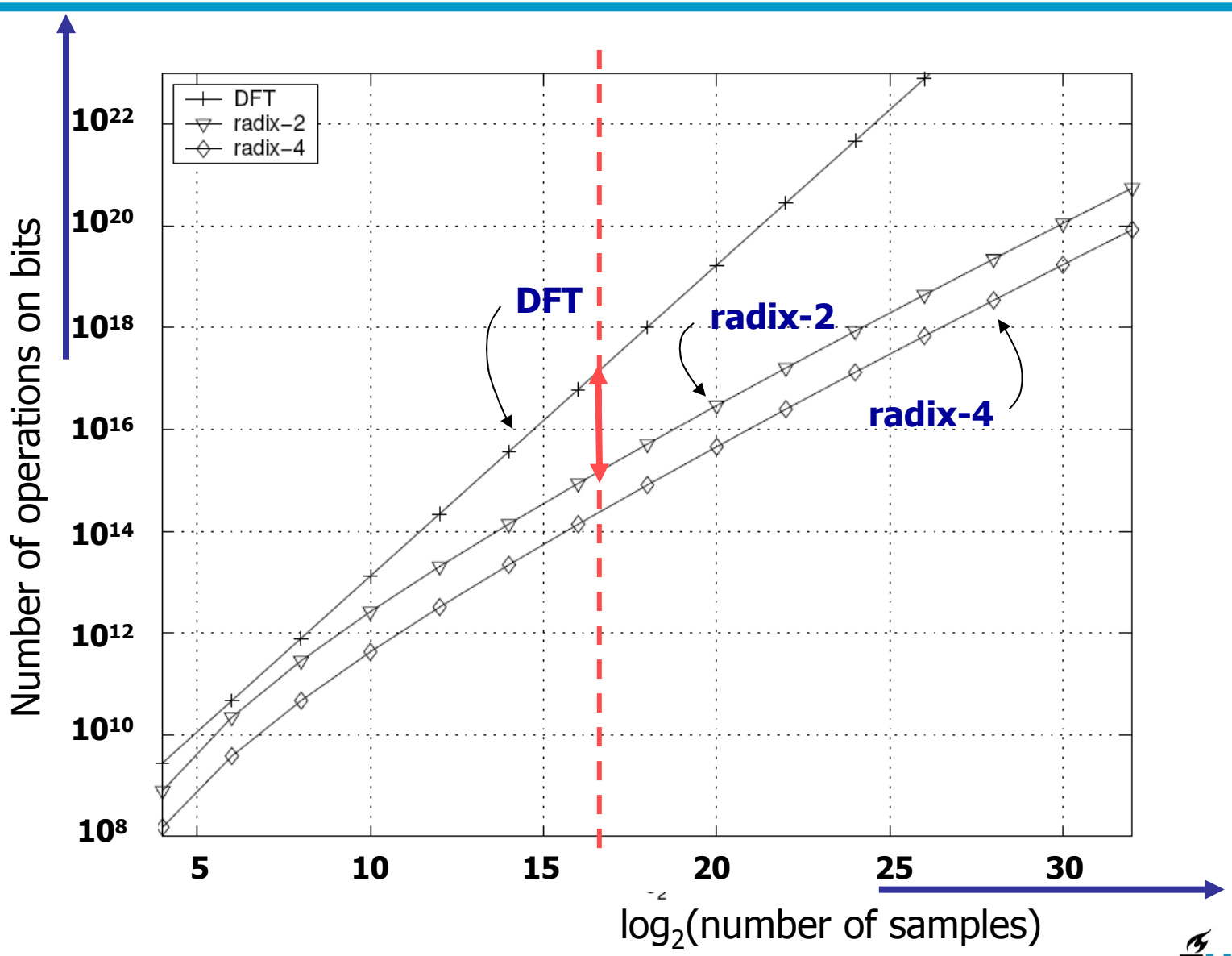


- How does this work out for encrypted signals?
  - Need to scale the powers W^{nk} ➔ *truncation and scaling*
  - Finite field should not disrupt the algebraic properties needed: concatenated scaling leads to larger modulus, or modulus limits scaling
  - DFT and FFT need different truncation and scaling steps

Given Accuracy, What DFT/FFT Size Fits?



Given Accuracy, What is Complexity?



Maybe Surprising ....

- The maximum allowable DFT size depends on
  - the modulus of the cryptosystem,
  - the DFT/FFT implementation,
  - the required precision.

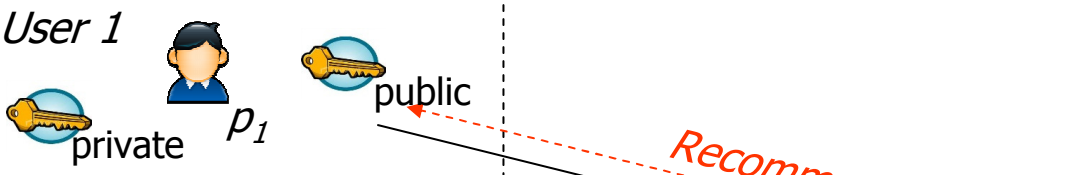
- There is a tradeoff between feasible but less efficient implementations and efficient but sometimes unfeasible ones.

- If the number of DFT points M is very large (e.g. multidimensional signals), only DFT may be feasible
  - This is counter intuitive.
  - Huge implications for computational complexity.

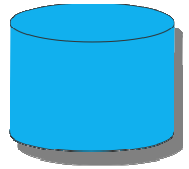
Content of the Lecture

- Introduction (done)
- Cryptography 2.0. (done)
  - Homomorphic cryptography.
  - Secure multiparty computation.
- Secure face recognition. (done)
- Secure recommender system.
- Challenges. (wrap up around 3:30 pm)

Recommender Systems



Server



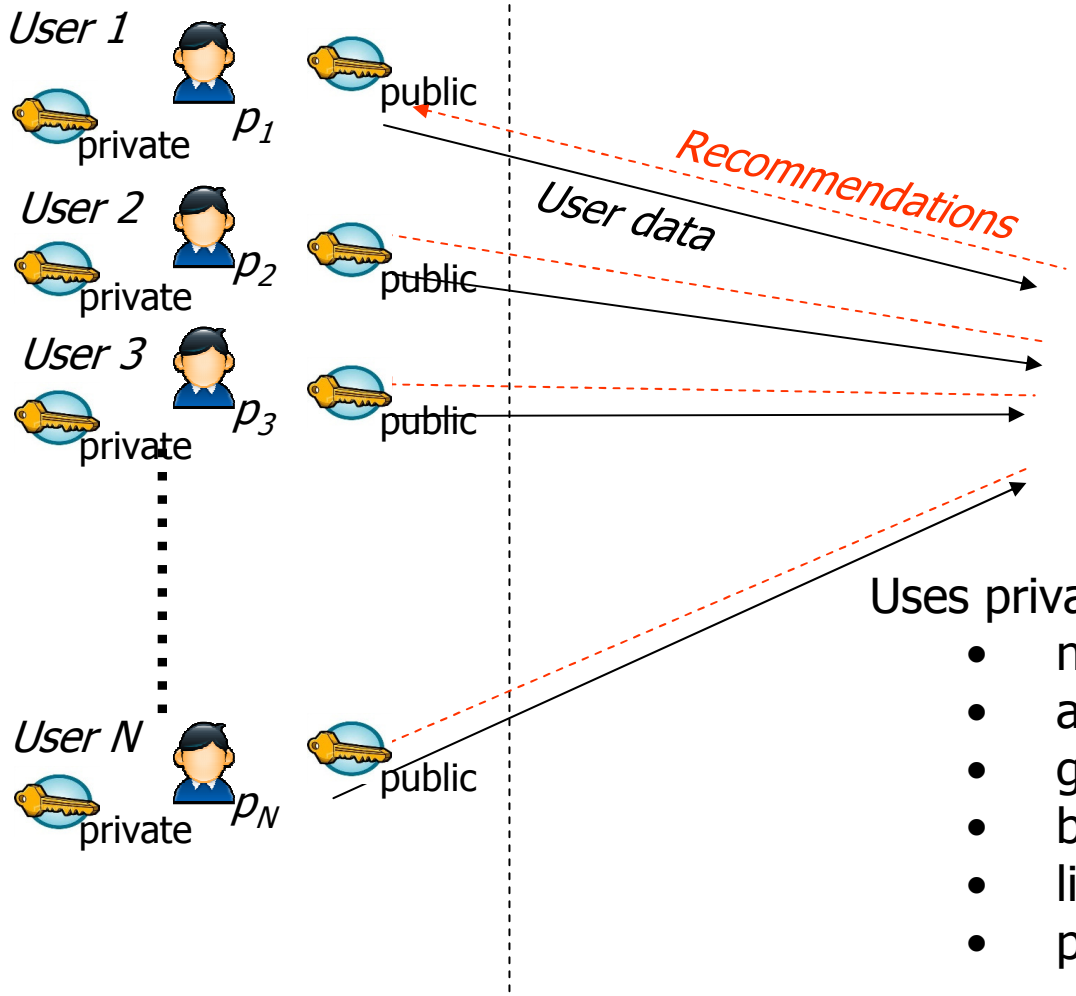
Uses privacy-sensitive data:

- name,
- age,
- gender,
- birth date,
- likes-dislikes (food, movie, cities),
- previous buys, sites visited, ratings,

…

to carry out clustering, collaborative filtering, recommendation,

Recommender Systems



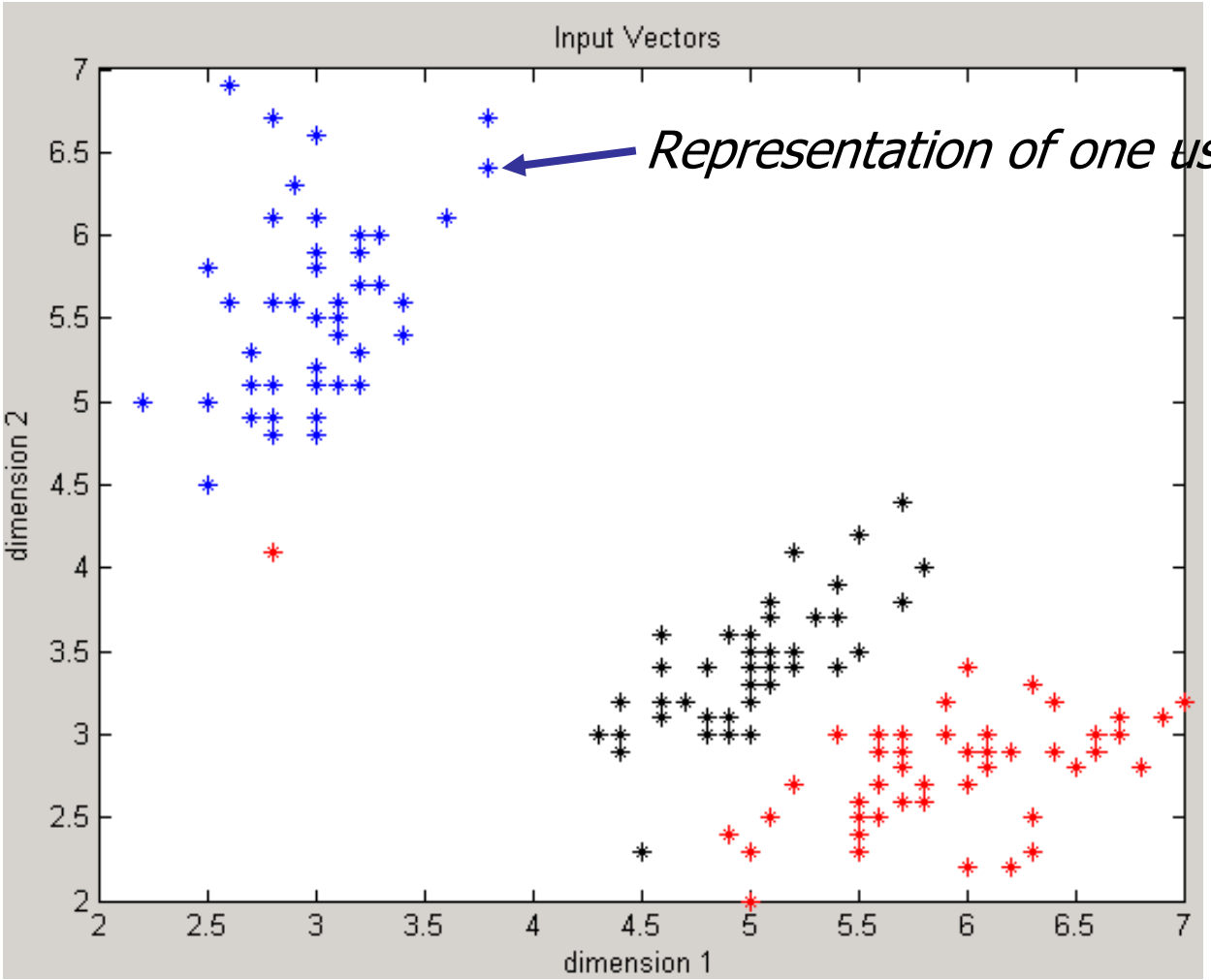
Uses privacy-sensitive data:

- name,
- age,
- gender,
- birth date,
- likes-dislikes (food, movie, cities),
- previous buys, sites visited, ratings,

...

to carry out clustering, collaborative filtering, recommendation,

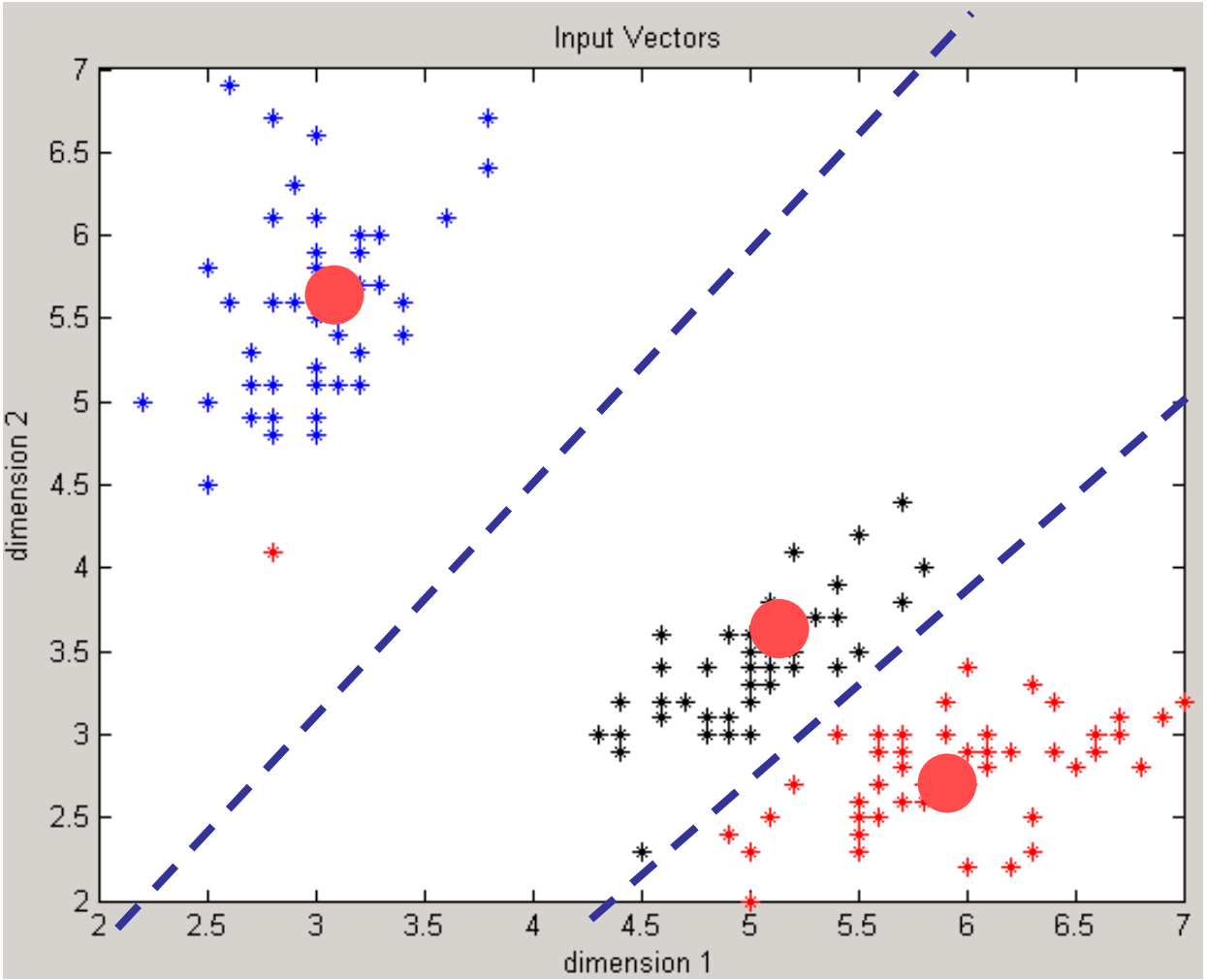
Pattern Recognition 101; *N* users in *K* groups



Dimension R=2, #users N = 100

Pattern Recognition 101; 100 users in 3 groups

K-Means Algorithm



Dimension R=2, #users N = 100, #centroids K=3

K-Means Algorithm

- The server carries out the clustering procedure:
  1. Start with arbitrary cluster centers (centroids).
  2. Assign users (user preferences vectors) to clusters based on the provided data.
  3. Recalculate cluster centers.
  4. Goto 2 and repeat until convergence.

- Note that:
  - Step 2 is similar to the last steps of the secure face recognition method
  - Namely:
    - A server needs to find the minimum distance.
    - In a secure version the user's vector is encrypted.

Secure K-Means Algorithm

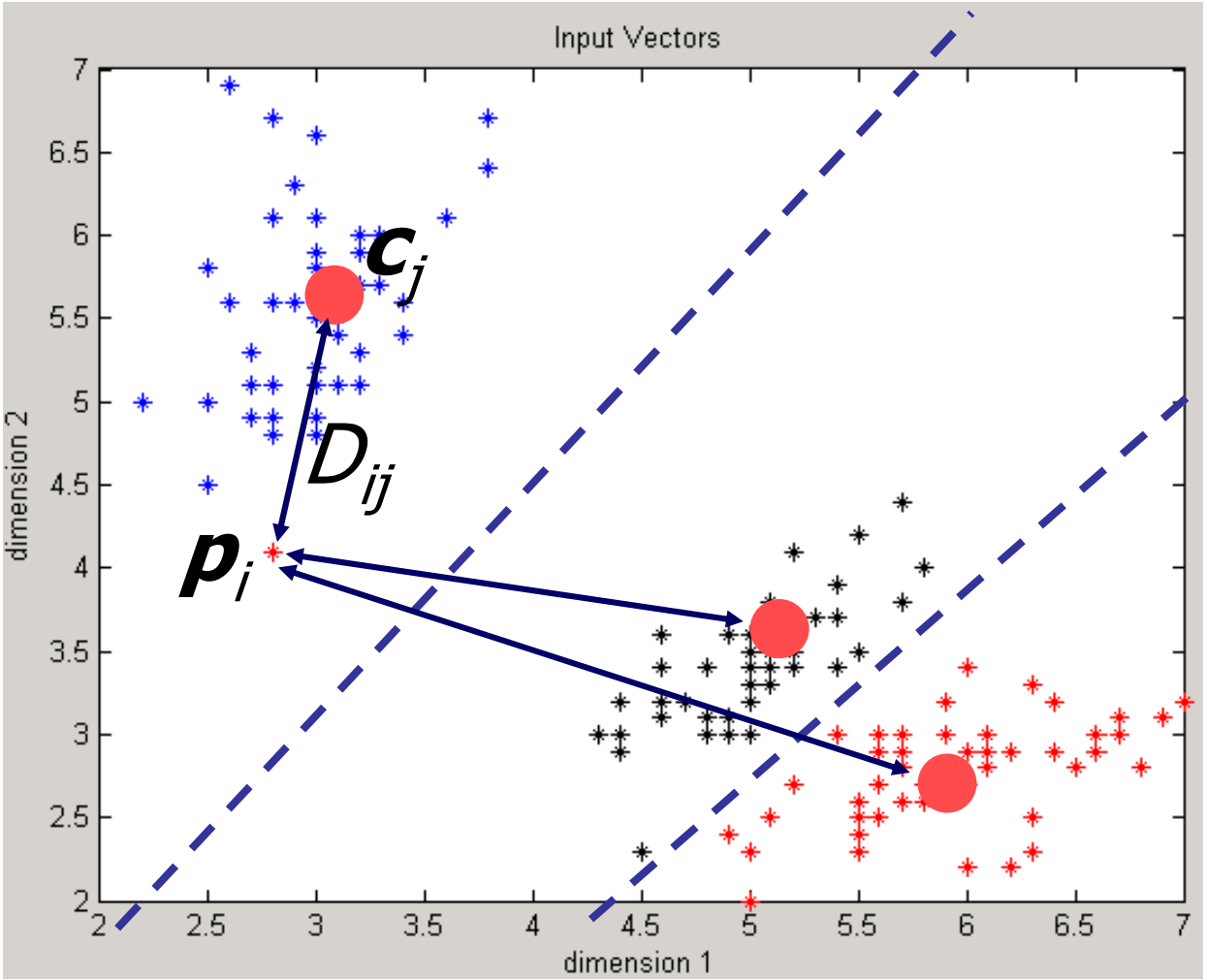
- Assume the user provides only encrypted vectors to the server.
- Server should only know cluster centers (centroids) $\boldsymbol{c}_j$, but
  - *not* the input of individual users $\boldsymbol{p}_i$,
  - *not* the cluster to which an individual user belongs.

- Users should only know their own data $\boldsymbol{p}_i$ and (eventually) their classification, but
  - *not* the cluster centers $\boldsymbol{c}_j$.

- (In a more extended version, also the cluster will not know the centroids).

Pattern Recognition 101; 100 users in 3 groups

K-Means Algorithm



Dimension R=2, #users N = 100 (i), #centroids K=3 (j)

Secure Clustering: Compute Distances (1)

- Each user *i* calculates (if plaintext vectors):

$$D_{i,j}^2 = \sum_{n=1}^{R}(c_{j,n} - p_{i,n})^2 = \underbrace{\sum_{n=1}^{R} c_{j,n}^2}_{\text{provided by server}} + \underbrace{\sum_{n=1}^{R} -2p_{i,n}c_{j,n}}_{c_{j,n}\text{ encrypted}} + \underbrace{\sum_{n=1}^{R} p_{i,n}^2}_{\text{available to user}}$$

(n runs over elements in the vector, 1 … R)

Secure Clustering: Compute Distances (2)

- Each user i calculates (if plaintext vectors):

$$D_{i,j}^2 = \sum_{n=1}^{R}(c_{j,n} - p_{i,n})^2 = \underbrace{\sum_{n=1}^{R} c_{j,n}^2}_{\text{provided by server}} + \underbrace{\sum_{n=1}^{R} -2p_{i,n}c_{j,n}}_{c_{j,n}\text{ encrypted}} + \underbrace{\sum_{n=1}^{R} p_{i,n}^2}_{\text{available to user}}$$

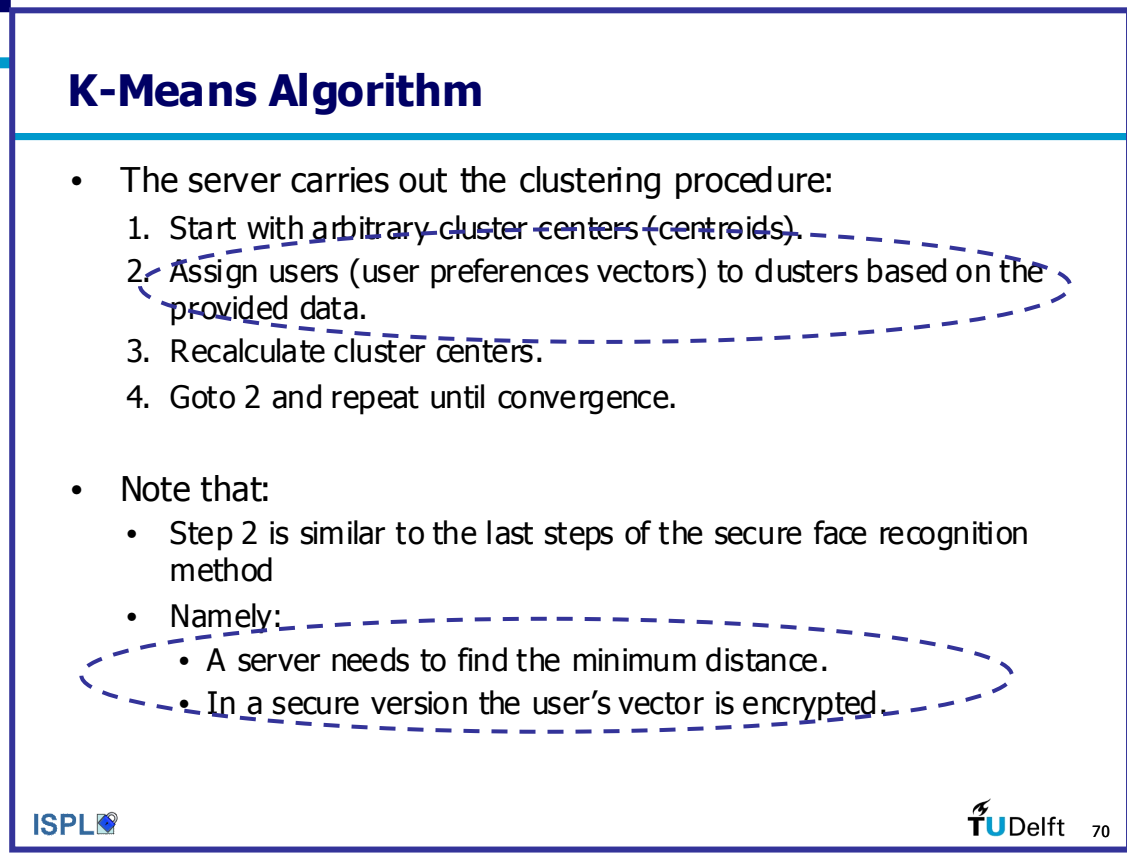
- When server data (centroid) $\boldsymbol{c}_j$ is encrypted, user i calculates:

$$[D_{i,j}^2] = \left[\sum_{n=1}^{R}(c_{j,n} - p_{i,n})^2\right]$$

$$= \underbrace{\left[\sum_{n=1}^{R}(c_{j,n})^2\right]}_{\text{easy by server (computed by the SP)}} \cdot \underbrace{\left(\prod_{n=1}^{R}[c_{j,n}]^{-2p_{i,n}}\right)}_{\text{easy by user (homomorphism)}} \cdot \underbrace{\left(\left[\sum_{n=1}^{R}(p_{i,n})^2\right]\right)}_{\text{easy by user (computed by the user)}}$$

K-Means Algorithm

- Centroids are encrypted by server.
- Each user can calculate its (encrypted) distance to each (encrypted) centroid, $[D_{ij}^2]$.



- To do:
  1. Find minimum distance: result is encrypted.
  2. Update the centroids.

Find (Position of) the Minimum

- Each user i has $\left([D^2_{i,1}], [D^2_{i,2}], \ldots, \underbrace{[D^2_{i,j}]}_{\text{smallest value}}, \ldots, [D^2_{i,K}] \right)$

$$\Downarrow$$

  and calculates $\left([0], [0], \ldots, \underbrace{[1],}_{\text{at position } j} \ldots, [0] \right)$

- Simplify $([D^2_{i,1}], [D^2_{i,2}])$ and run repeatedly in binary tree fashion.

$$\Downarrow$$

$$([0], [1])$$

- Similar to secure face recognition, use same algorithm.

Update the Centroids

- Per user $\boldsymbol{p}_i$, server find cluster $\boldsymbol{c}_j$ with minimum distance
- This information is given in *encrypted* form:
  - Indicated by encrypted vector $\boldsymbol{x}_i$:

$$[\mathbf{x}_i] = \underbrace{\left([0],[0], \quad \ldots \quad \underbrace{[1]}_{\text{closest centroid}} \quad ,\ldots[0],[0] \right)}_{\textit{Dimension K}}$$

Update the Centroids

- Per user $\boldsymbol{p}_i$, server find cluster $\boldsymbol{c}_j$ with minimum distance
- This information is given in *encrypted* form:
  - Indicated by encrypted vector $\boldsymbol{x}_i$:

$$[\mathbf{x}_i] = \left([0],[0], \quad \ldots \quad \underbrace{[1]}_{\text{closest centroid}} \quad ,\ldots[0],[0] \right)$$

- Update the centroids:

$$c_{j,n} = \frac{1}{\left(\begin{array}{l}\#\text{users who have} \\ \text{centroid } j \text{ as closest}\end{array}\right)} \sum_{\substack{\text{Those users who have} \\ \text{centroid } j \text{ as closest}}} p_{i,n} \qquad \text{(for all } j = 1 \ldots K\text{)}$$

Update the Centroids

- Per user $\boldsymbol{p}_i$, server find cluster $\boldsymbol{c}_j$ with minimum distance
- This information is given in *encrypted* form:
  - Indicated by encrypted $1 \times K$ vector $\boldsymbol{x}_i$:

$$[\mathbf{x}_i] = \left([0],[0], \quad \ldots \quad \underbrace{[1]}_{\text{closest centroid}} \quad ,\ldots[0],[0] \right)$$

- Update the centroids:

$$(c_{1,n}, c_{2,n}, \ldots, c_{K,n}) = \sum_{i=1}^{N} x_{i,n} p_{i,n}$$

$$= \sum_{i=1}^{N} \underbrace{(0,0,\ldots,1,\ldots 0)}_{\text{"1" at } j\text{-th position for } i\text{-th user}} \quad p_{i,n}$$

Update the Centroids (1)

- Per user $\boldsymbol{p}_i$, server find cluster $\boldsymbol{c}_j$ with minimum distance
- This information is given in *encrypted* form:
  - Indicated by encrypted vector $\boldsymbol{x}_i$:

$$[\mathbf{x}_i] = \Big([0],[0], \quad \ldots \quad \underbrace{[1]}_{\text{closest centroid}} \quad ,\ldots[0],[0] \Big)$$

- Update the centroids:

$$[c_{i,n}] = \left[\sum_{i=1}^{N} \underbrace{x_{i,n}}_{0/1} p_{i,n} \right] = \prod_{i=1}^{N} \underbrace{[x_{i,n} p_{i,n}]}_{\substack{x_{i,n} \text{ available only} \\ \text{to user in} \\ \text{encrypted form}}} = \prod_{i=1}^{N} \underbrace{[x_{i,n}]^{p_{i,n}}}_{\substack{\text{can be evaluated} \\ \text{by user}}}$$

Update the Centroids (2)

- Each user calculates $[x_{i,n}]^{p_{i,n}}$.
- How to combine via product: $\prod_{i=1}^{N} [x_{i,n}]^{p_{i,n}}$
  - User cannot do this because it has only its own $[x_{i,n}]^{p_{i,n}}$
  - Server can but undesirable because it can decrypt individual $[x_{i,n}]^{p_{i,n}}$
- Solutions:
  - Collaborative protocol amongst all users using *blinding*.
  - Involve semi-trusted computing party.
  - The latter also makes it possible to deny server access to centroids (i.e. server sees only encrypted centroids).
  - Protocols are not very efficient.
- There is room for improvement.

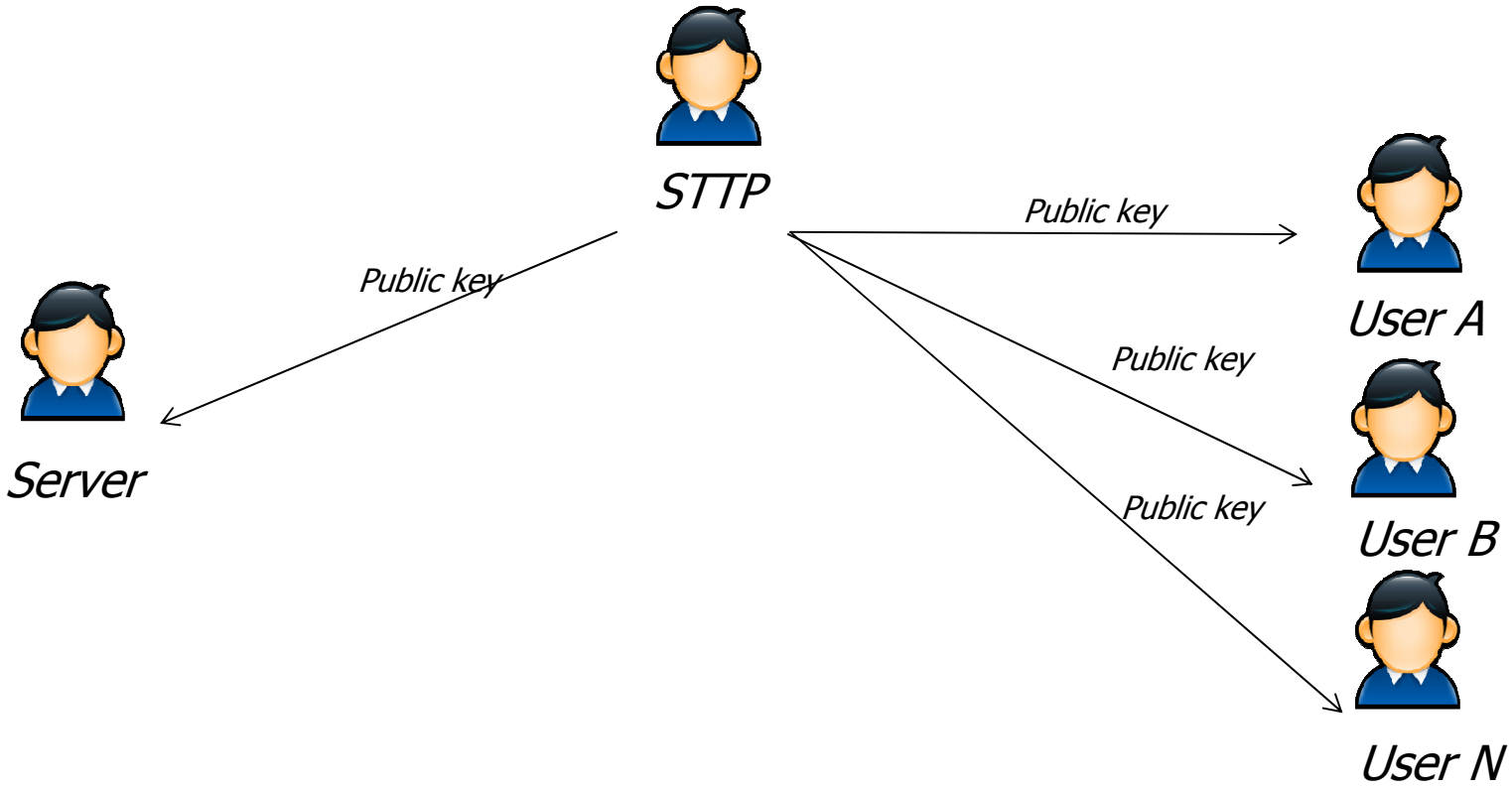
Improvements ...

- More restricted access by the server:
  - Centroids are unknown.
  - The number of users in each cluster is unknown.

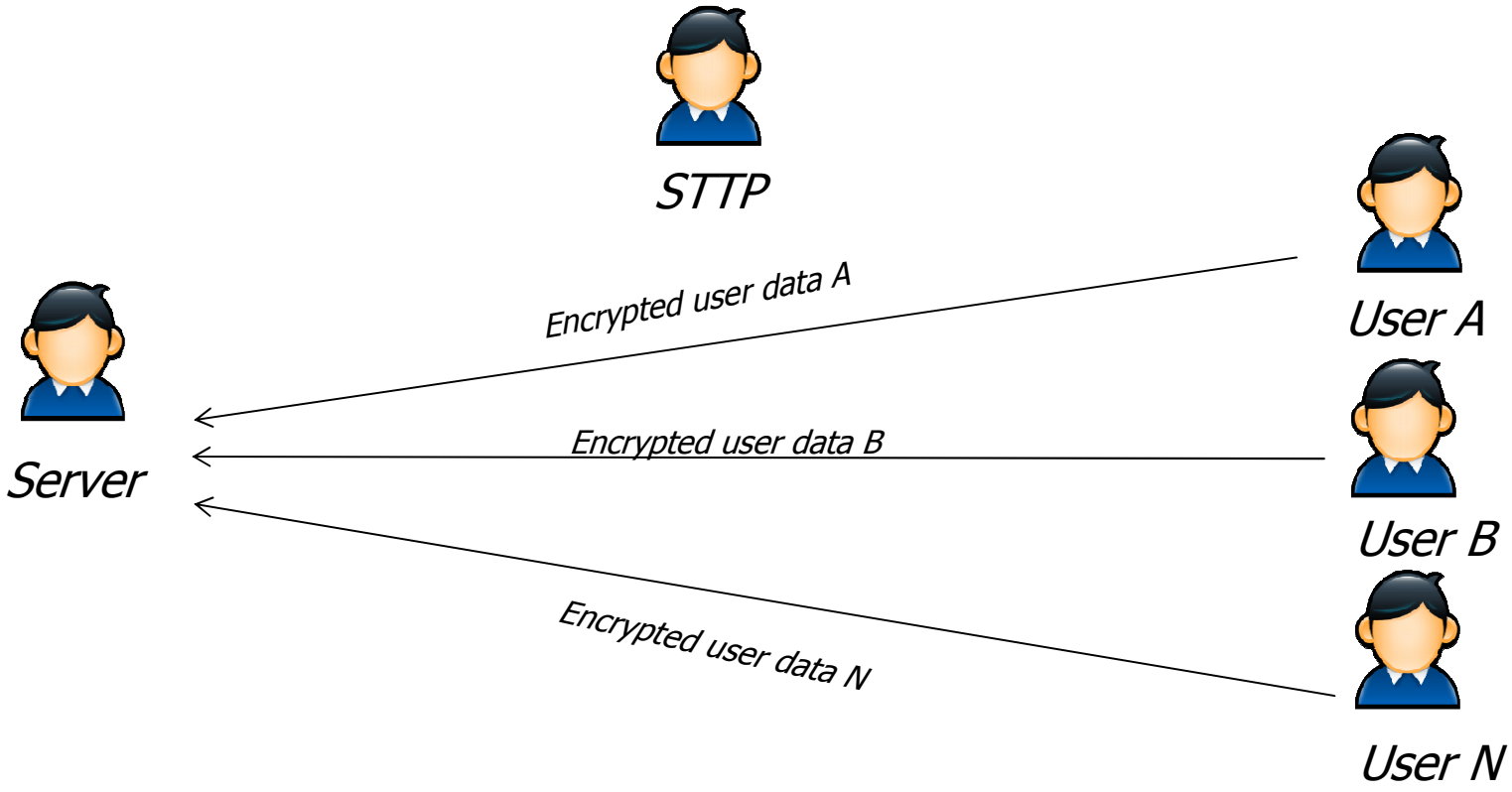
- By introducing a new player: Semi-trusted Third Party (STTP)
  - Trusted with computations.
  - Not trusted with the data.
  - Creates the key pairs.
  - Overtakes the responsibility of users for data processing.

- STTP is not a trusted third party (TTP) since it has no access to the data.

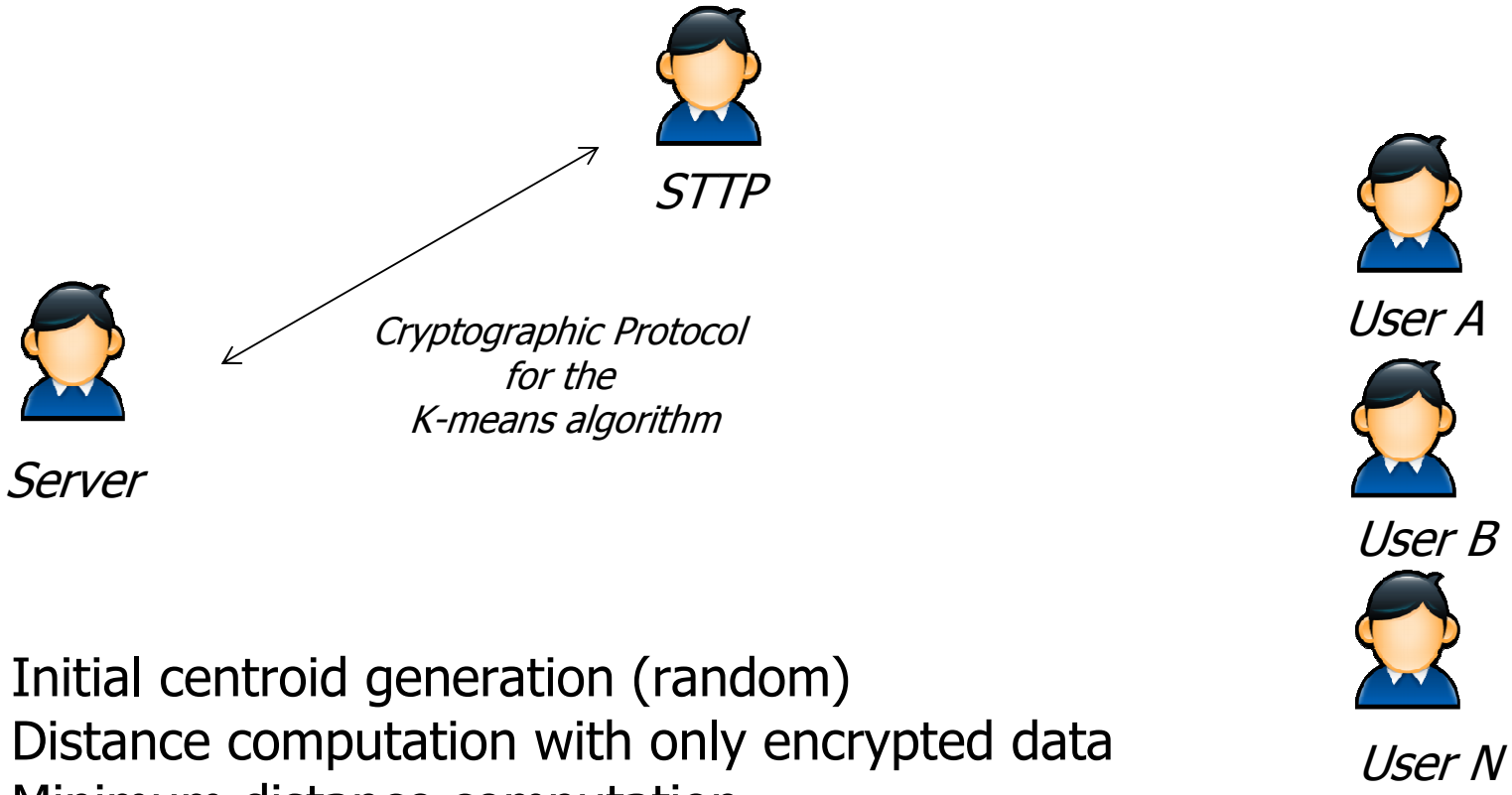
Extension with a STTP (1)



Extension with a STTP (2)

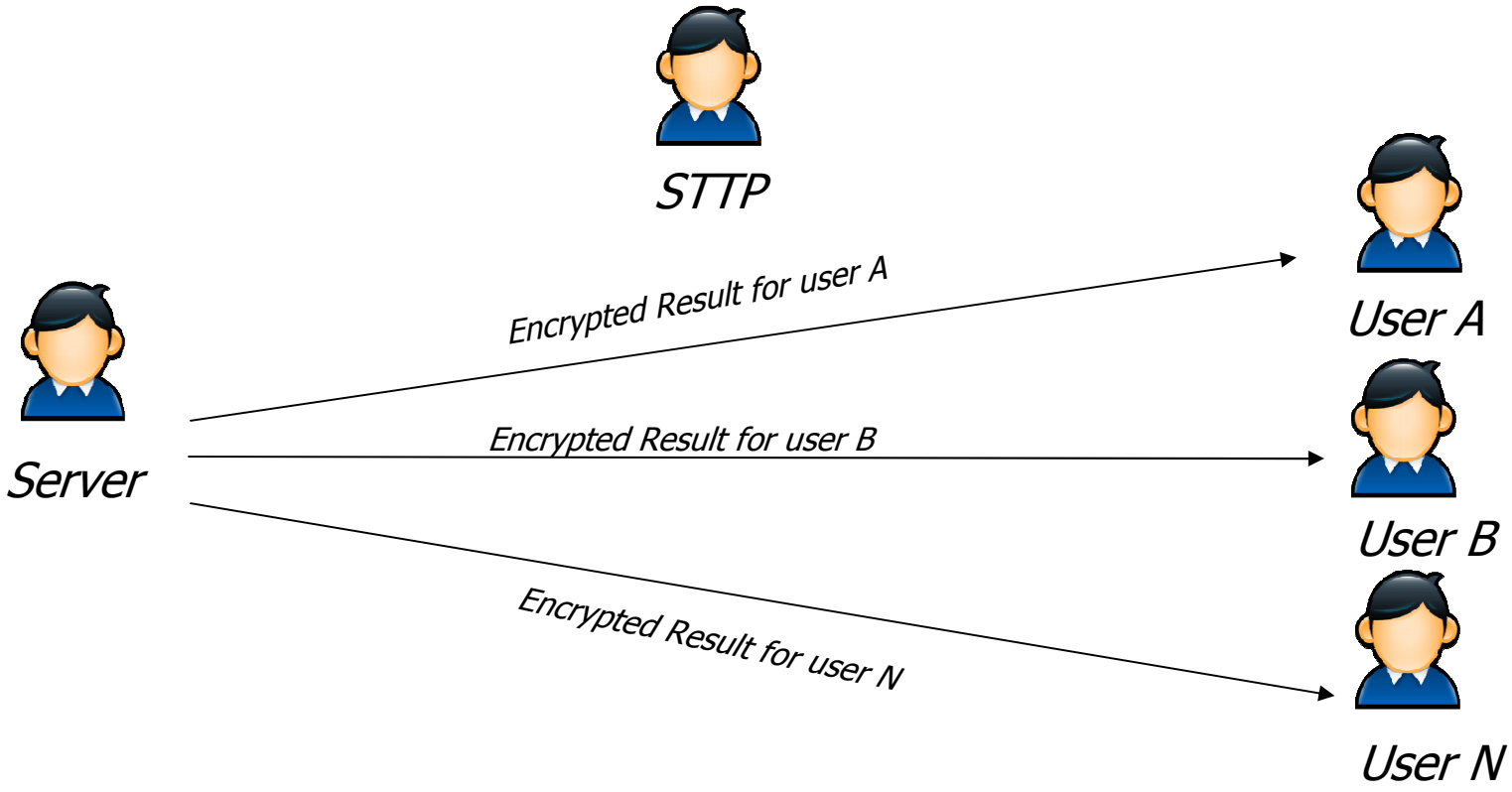


Extension with a STTP (3)

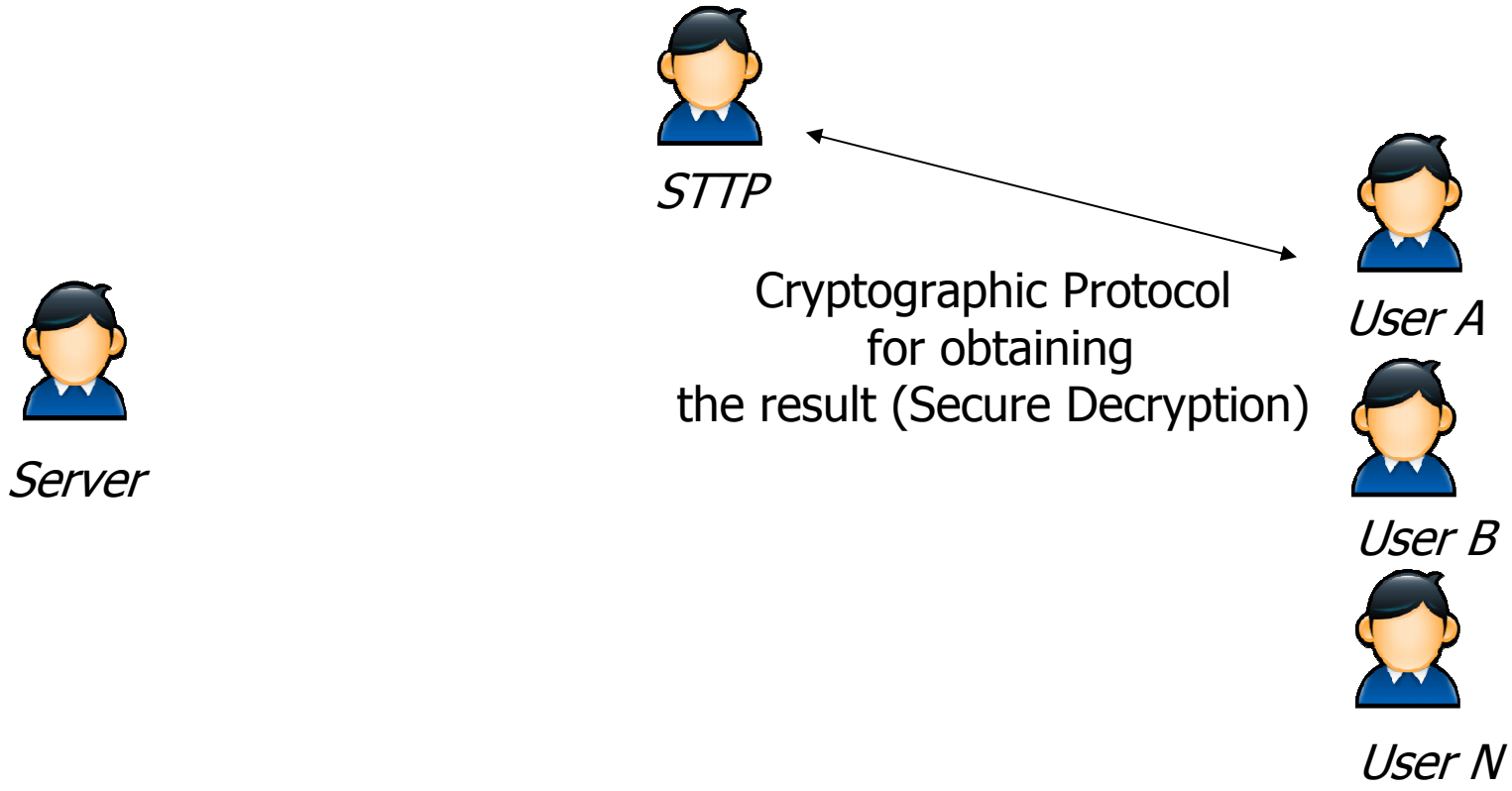


- Initial centroid generation (random)
- Distance computation with only encrypted data
- Minimum distance computation
- Centroid update procedure

Extension with a STTP (4)



Extension with a STTP (5)



Evaluation (1)

- User preference data is encrypted value-by-value
- Clustering algorithm is run by server on encrypted data
- Distance calculation 'easy' thanks to homomorphic properties
- Interactive protocols required for:
  - Finding minimum of distances
  - Updating centroids

- 1000 users, K=10 clusters, 10 iterations, 12-dimensional space
  - Timing:
    - Several minutes (versus few seconds on plaintext)

  - Communication costs:
    - 480 kBytes (each user)

Evaluation (2)

- More privacy preserving with STTP
  - Initial centroids locations are random and jointly created
  - Centroids and number of users in each cluster are unknown

- Cryptographic protocols are different
  - Exponentiations are now Secure Multiplication Protocols
  - Working with encrypted data all the time (no public data)
  - Dedicated algorithm for updating centroids without division
  - Data packing where possible

- Efficiency
  - Less computation for the users
  - More computation for the STTP and the server
  - But scalable because they are computationally powerful

Content of the Lecture

- Introduction (done)
- Cryptography 2.0. (done)
  - Homomorphic cryptography.
  - Secure multiparty computation.
- Secure face recognition. (done)
- Secure recommender system. (done)
- Challenges. (wrap up around 3:30 pm)

Challenges (1)

- In an increasing number of applications, secure signal processing is needed.
- Some (linear ops, distances) but not all signal processing can be done efficiently using homomorphic operations.
- For other operations we need
  - interactive protocols,
  - garbled circuits.
- Data expansion: stacking data into longer vectors.
- Additional players (such as STTP) can simplify the protocols considerably.

Challenges (2)

- Security model should be reconsidered.
  - Semi-honest is too simple.
  - Active adversary model is too complex.

- Exploit signal processing properties for efficient secure implementations.
- Reformulating or approximating signal processing algorithms so that they translate into more efficient secure versions.

- Signal processing is "inexact" (noise is tolerated). How to exploit in order to make crypto protocols more efficient?