Chapter 11

Secure Two-Party Computation Protocols

Stefan Dziembowski

www.crypto.edu.pl/Dziembowski

University of Warsaw

Plan

1. Introduction to two-party computation protocols
2. Definitions
3. Information-theoretic impossibility
4. Constructions
    1. oblivious transfer
    2. computing general circuits
5. Fully homomorphic encryption
6. Practical aspects
7. Private Information Retrieval

A love problem

$A :=\begin{cases} \mathbf{0} & \textbf{if Alice doesn't love Bob} \\ \mathbf{1} & \textbf{if Alice loves Bob} \end{cases}$

$B :=\begin{cases} \mathbf{0} & \textbf{if Bob doesn't love Alice} \\ \mathbf{1} & \textbf{if Bob loves Alice} \end{cases}$

They want to learn the value of
$f(A,B) := A$ **and** B

Solution?

A

B

computes
A* and *B
locally

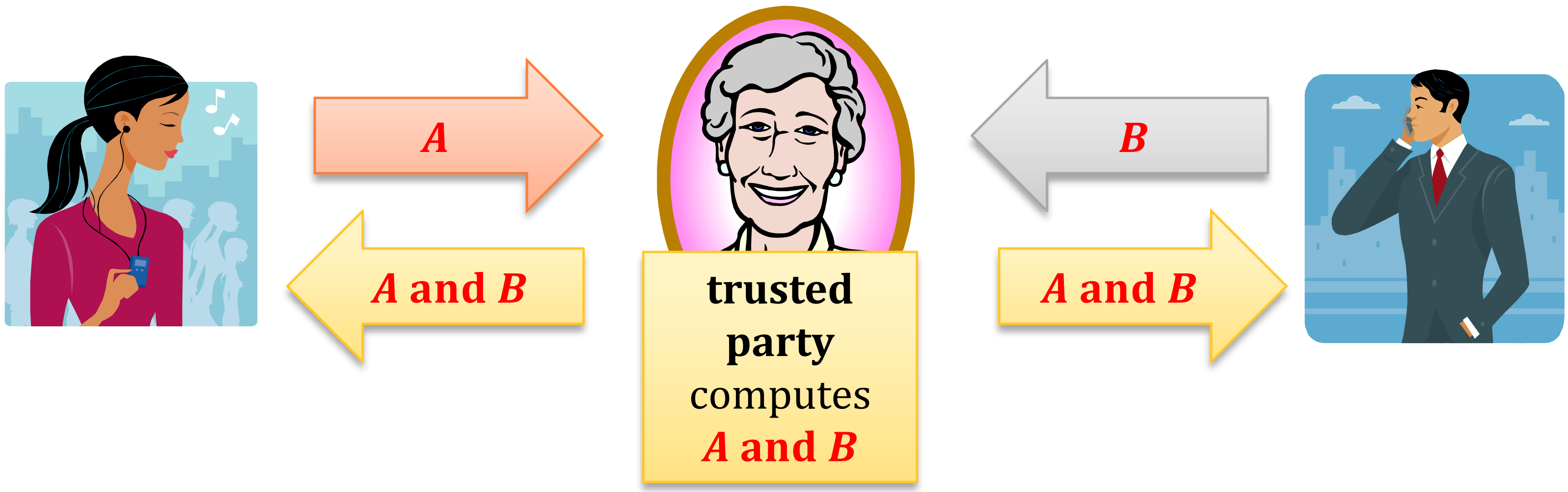
computes
A* and *B
locally

Problem

If $A = 0$ and $B = 1$ then **Alice** knows that **Bob** loves him while she doesn't!

If $A = 1$ and $B = 0$ then **Bob** knows that **Alice** loves him while he doesn't!

Solution?



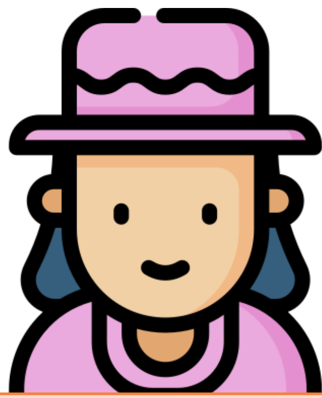
Alice and Bob learn **only** the value of $f(A, B) = A \textbf{ and } B$.

Of course: if $A = B = 1$ then $f(A, B) = 1$ and there is no secret to protect.

But, e.g., if $A = 0$ and $B = 1$ then $f(A, B) = 0$ then **Alice** will not know the value of B.

Question: Is it possible to compute f without a trusted party?

Another example: “the millionaire’s problem”



A := how much money **Alice** has

B := how much money **Bob** has

$$f(A,B) := \begin{cases} \text{“Alice” if } A > B \\ \text{“equal” if } A = B \\ \text{“Bob” if } A < B \end{cases}$$

How to solve this problem?

Can they compute ***f*** in a secure way?

(secure = “only the output is revealed”)

Of course, they **do not trust** any “third party”.

Answer

It turns out that:

in both cases, there exists a cryptographic protocol that allows *A* and *B* to compute *f* in a secure way.

Moreover:

In general, every poly-time computable function ***f*** can be computed securely by two-parties.

Of course, this has to be defined...

(assuming some problems are computationally hard)

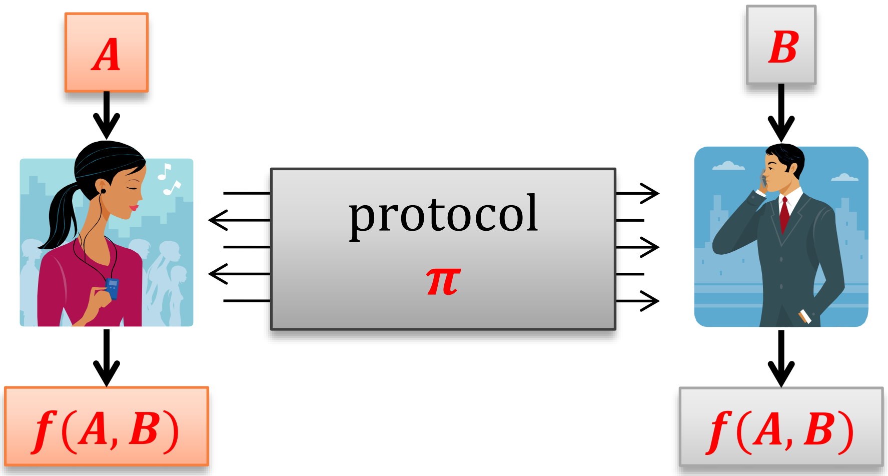
Plan

1. Introduction to two-party computation protocols
2. Definitions
3. Information-theoretic impossibility
4. Constructions
    1. oblivious transfer
    2. computing general circuits
5. Fully homomorphic encryption
6. Practical aspects
7. Private Information Retrieval

What do we mean by a "secure function evaluation"?

In general, the definition is complicated, and we'll not present it here.

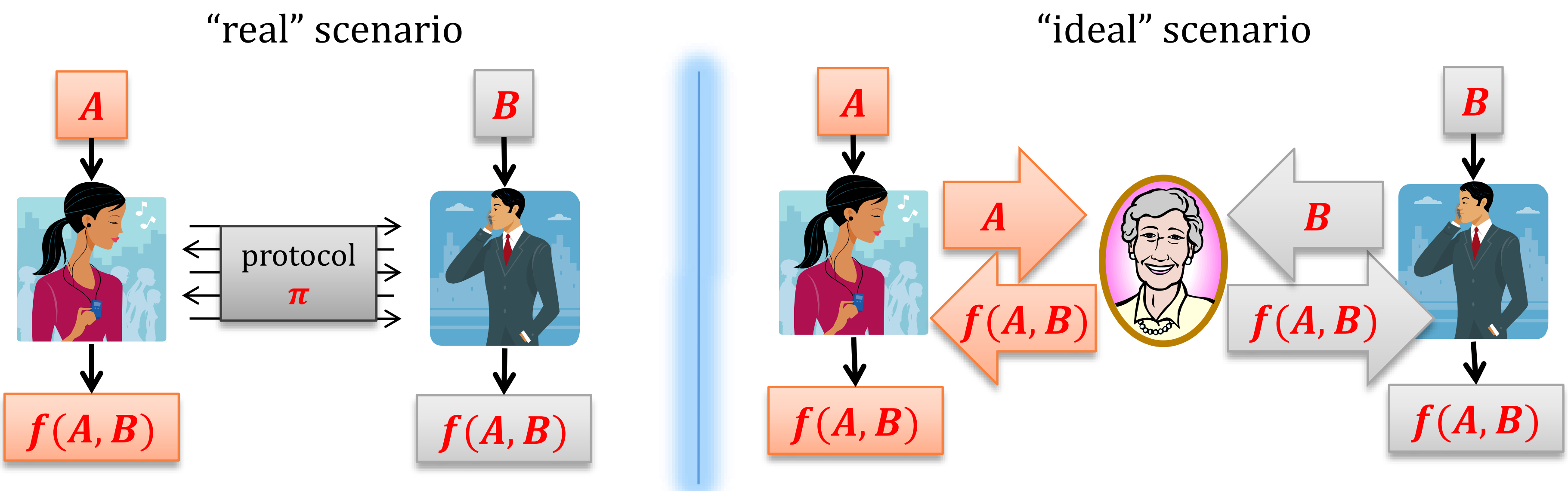
Main idea: suppose we have a function $f\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$



Each of the parties may try to:

- **learn something** about the input of the other party, or
- **disturb the output** of the protocol.

What do we mean by a “secure function evaluation”?



A malicious participant (**Alice** or **Bob**) should not be able to

- learn more information, or
- do more damage to the output

in the “**real” scenario**, than it can in the **“ideal”** one.

What do we mean by this?

For example:

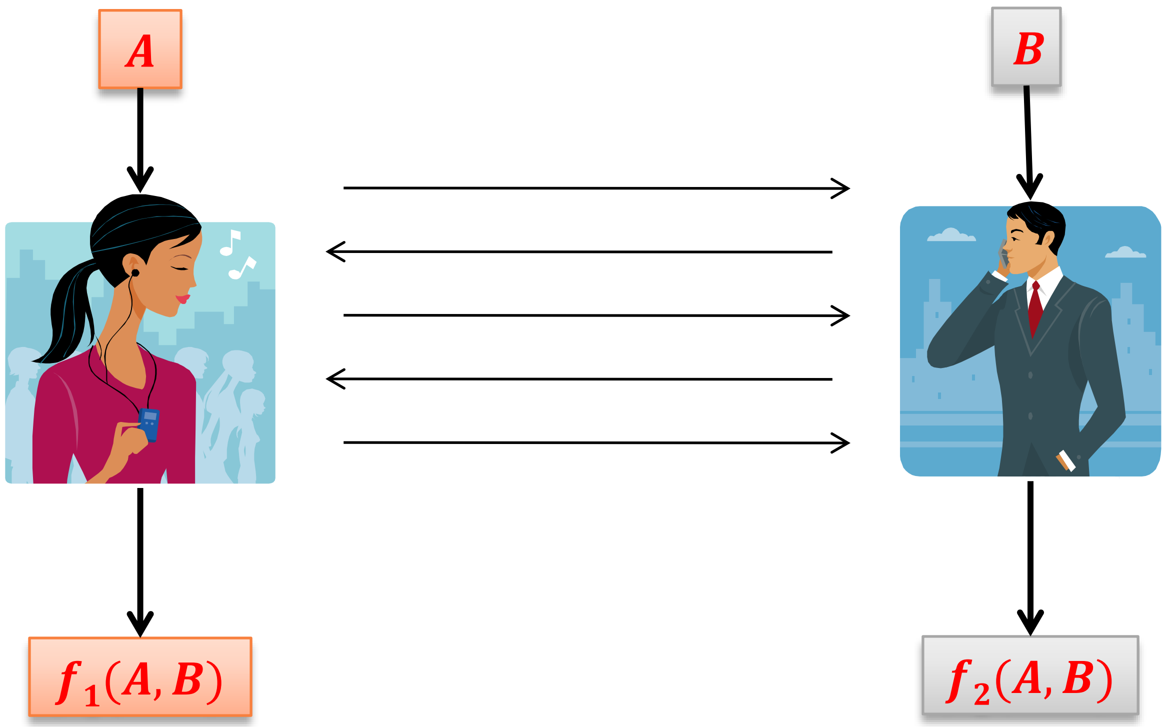
Alice can always declare that she loves **Bob**, while in fact she doesn't.

A **millionaire** can always claim to be poorer or reacher than he is...

But:

Bob cannot force the output of the protocol to be "equal" if he doesn't know the value of A.

Let's generalize it a bit:

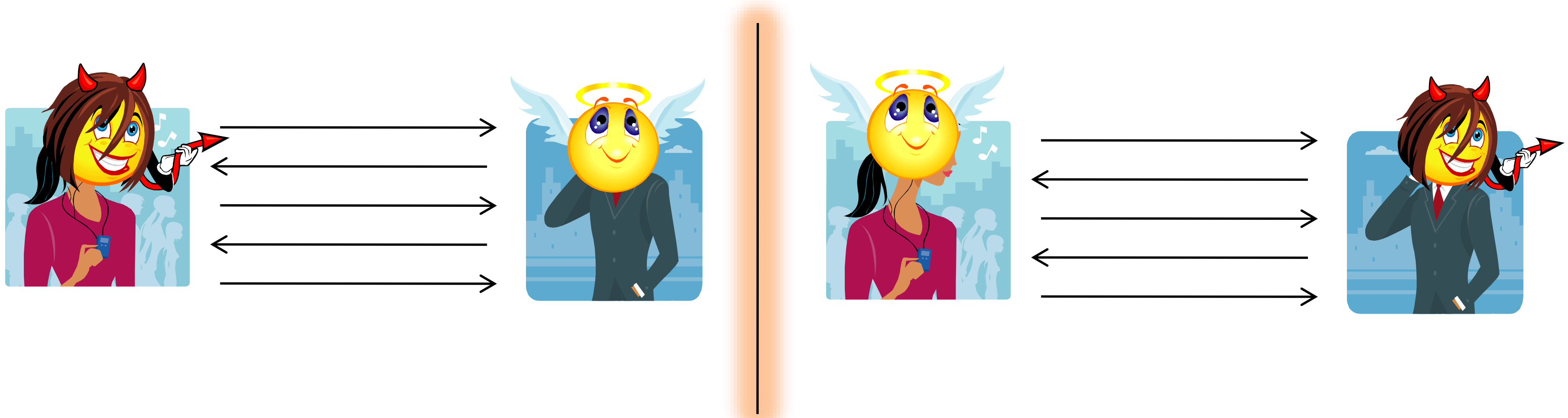


1. the outputs of **Alice** and **Bob** can be different
2. the function that they compute may be randomized

An adversary

It is convenient to thing about an adversary that **corrupts one of the players**.

(clearly if the adversary corrupts **both** players, there is no sense to talk about any security)



Two goals that the adversary may want to achieve

1. **learn** about the input of the other party "more than he would learn in the ideal scenario",

2. **change** the output of the protocol.

Two types of adversarial behavior

In general, we consider two types of adversarial behavior:

passive, also called: honest-but-curious:
a corrupted party follows the protocol

a protocol is **passively secure** if it is secure against one of the parties behaving maliciously **in a passive way.**

active, also called Byzantine
a corrupted party doesn't need to follow the protocol

a protocol is **actively secure** if it is secure against one of the parties behaving maliciously **in an active way.**

Problem with active security

In general, it is impossible to achieve a complete fairness.

That is: one of the parties may (after receiving her own output)

prevent the other party from receiving her output (by halting the protocol)

(remember the coin-flipping protocol?)

Fact

Let $\boldsymbol{\pi}$ be a **passively secure** protocol computing some function $\boldsymbol{f}$.

Then, we can construct a protocol $\boldsymbol{\pi}$' that is **actively secure**, and computes the same function $\boldsymbol{f}$.

How?

Using **Zero-Knowledge**!

(we skip the details)

Power of the adversary

The malicious parties may be

- computationally **bounded** (poly-time)
- computationally **unbounded**.

In this case we say that security is
information-theoretic

We usually allow the adversary to "break the security" with **some negligible probability**.

Plan

1. Introduction to two-party computation protocols
2. Definitions
3. Information-theoretic impossibility
4. Constructions
   1. oblivious transfer
   2. computing general circuits
5. Fully homomorphic encryption
6. Practical aspects
7. Private Information Retrieval

Some very natural functions cannot be computed by an **information-theoretically secure** protocol

Example

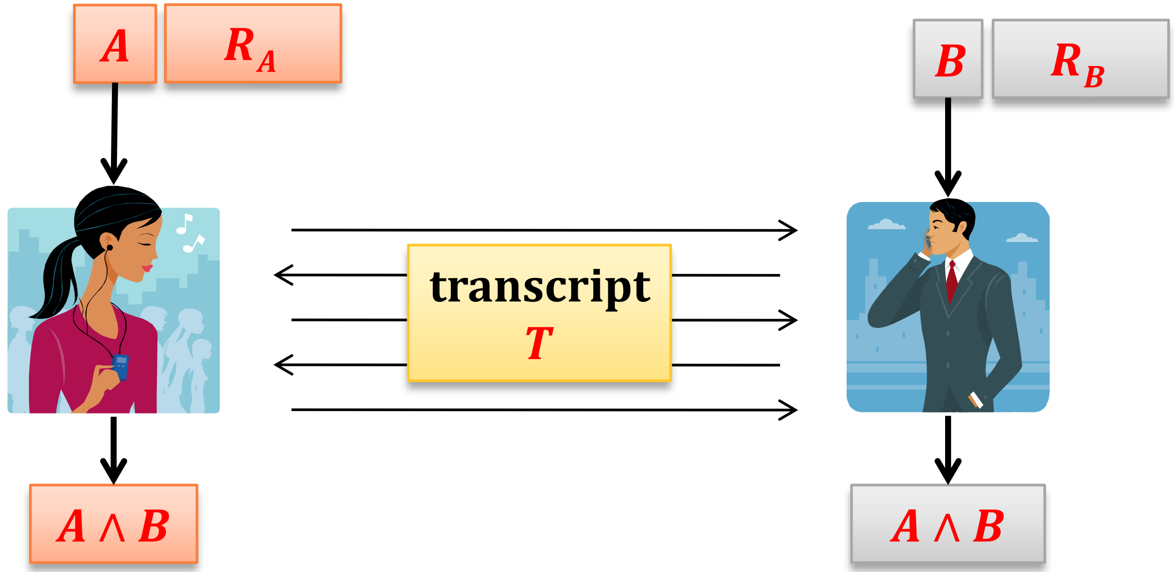
Consider a function

$$\boldsymbol{f(A, B) = A \wedge B}.$$

There exists an infinitely-powerful adversary that breaks **any protocol computing it**.

The adversary may even be passive.

A transcript



Definition

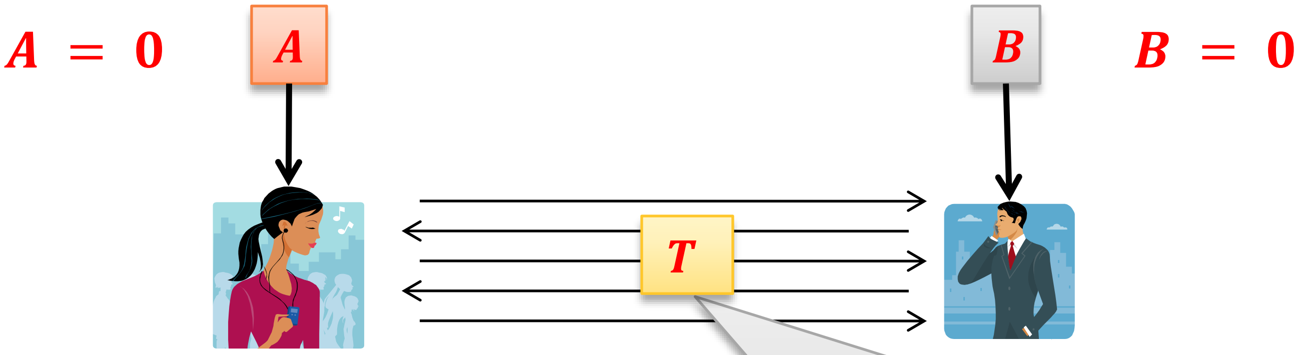
Transcript T is **consistent with input** $A = A_0$

for B – symmetric

if **there exist** random inputs R_A (for **Alice**) and (B, R_B) (for **Bob**) such that T is a transcript of the execution of the protocol with inputs

- (A_0, R_A) – for Alice
- (B, R_B) – for Bob.

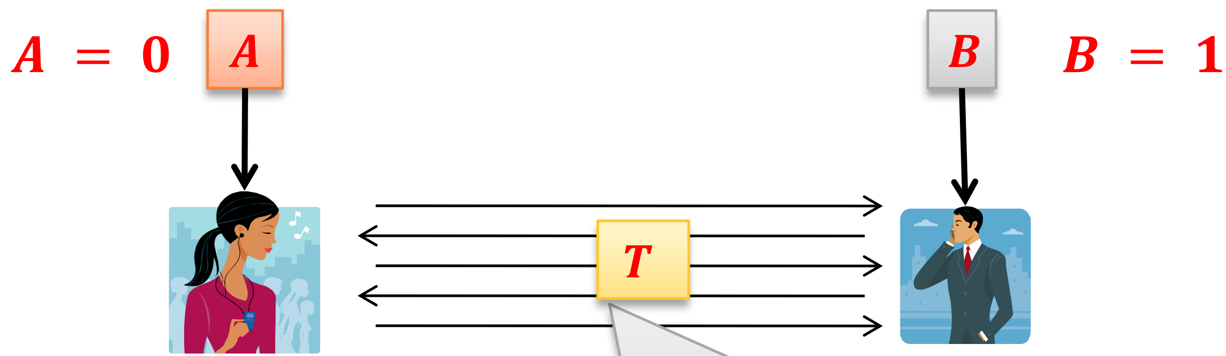
1. Suppose $\boldsymbol{A} = \mathbf{0}$ and $\boldsymbol{B} = \mathbf{0}$



has to be consistent with $\boldsymbol{A} = \mathbf{1}$

Otherwise a malicious Bob knows that $\boldsymbol{A} = \mathbf{0}$

2. Suppose $A = 0$ and $B = 1$

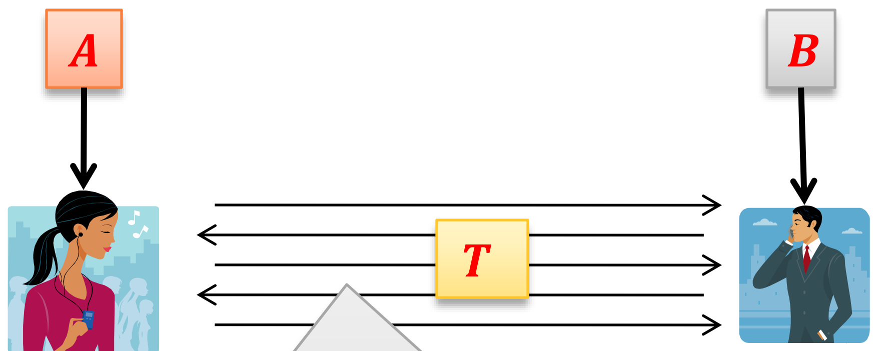


cannot be consistent with $A = 1$

Because the output of the protocol has to be different in these two cases:

- $A = 0$ and $B = 1$ and
- $A = 1$ and $B = 1$

So, if $A = 0$ then a malicious Alice has a way to learn what the input of Bob!



Alice checks if T is consistent with $A = 1$

If **yes** then she knows that $B = 0$

otherwise $B = 1$

Moral

If we want to construct a protocol for computing **AND**, we need to rely on computational assumptions.

Plan

1. Introduction to two-party computation protocols
2. Definitions
3. Information-theoretic impossibility
4. Constructions
    1. oblivious transfer
    2. computing general circuits
5. Fully homomorphic encryption
6. Practical aspects
7. Private Information Retrieval

A question

Does there exist a protocol π that is "complete for secure two-party computations"?

In other words:

We are looking for π such that:

if we have a secure protocol for π then we can construct a provably secure protocol for any function?

Answer

Yes!

A protocol like this is exists.

It is called **Oblivious Transfer (OT)**. There are two versions if it:

- **Rabin's Oblivious Transfer**

  M. O. Rabin. **How to exchange secrets by oblivious transfer**, 1981.

- **One-out-of-Two Oblivious Transfer**

  S. Even, O. Goldreich, and A. Lempel, **A Randomized Protocol for Signing Contracts**, 1985.

Rabin’s Oblivious Transfer

input bit $\boldsymbol{A}$

sender

receiver

The sender should have no information which was the case

outputs $\boldsymbol{B}$ such that

$$\boldsymbol{B} := \begin{cases} \boldsymbol{A} & \textbf{with probability } 1/2 \\ ? & \textbf{with probability } 1/2 \end{cases}$$

If $\boldsymbol{B} = ?$ then the receiver has no information on $\boldsymbol{A}$

One-out-of-two Oblivious Transfer

input bits (A_0, A_1)

sender

receiver

input bit B

The sender should have no information which was the case

outputs C such that

$$C := \begin{cases} A_0 & \textbf{if } B = 0 \\ A_1 & \textbf{if } B = 1 \end{cases}$$

then the receiver has no information on the other A_i

We will also write $C := \mathbf{OT}((A_0, A_1), B)$

Fact

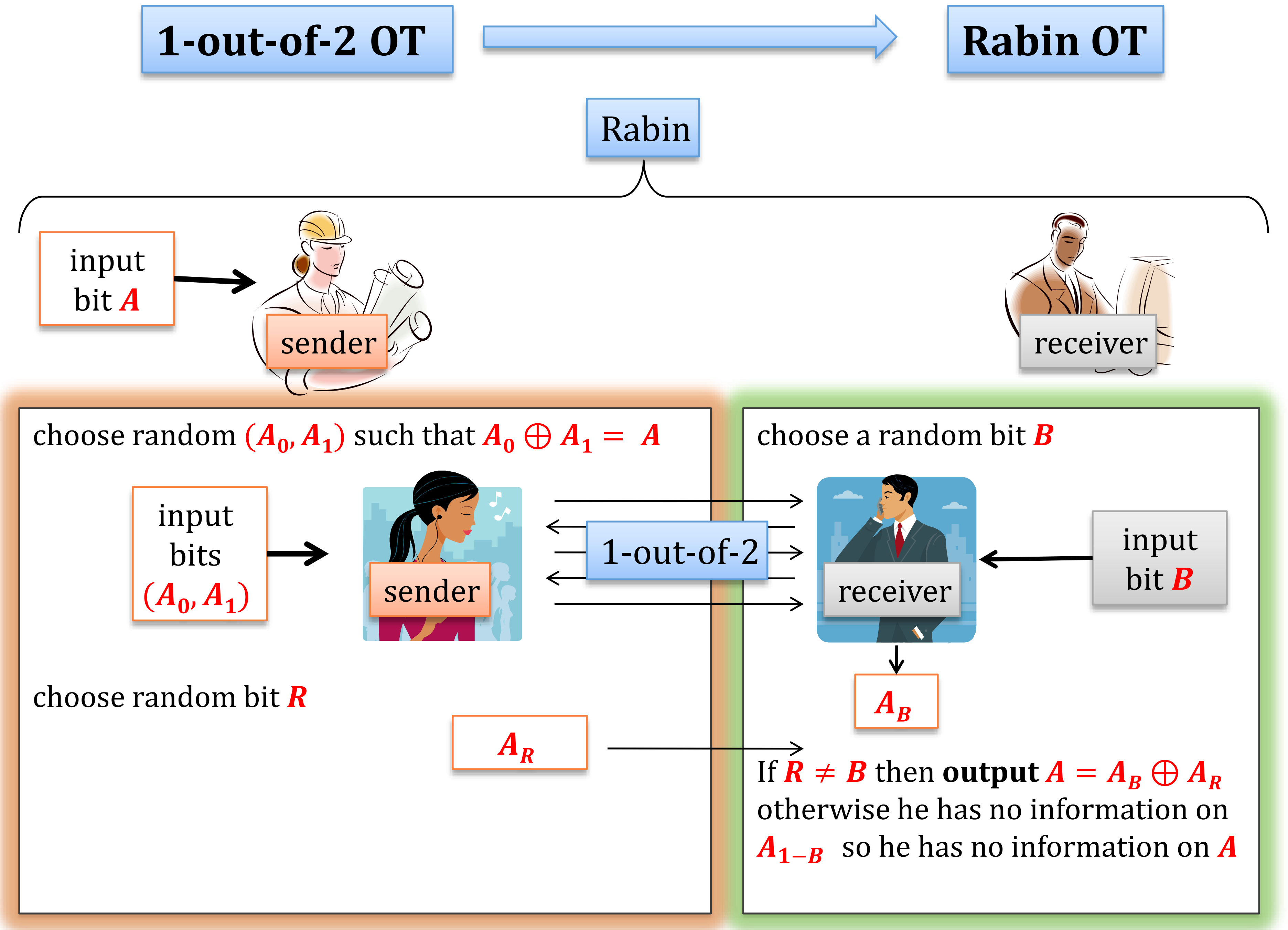
Rabin's Oblivious Transfer

and

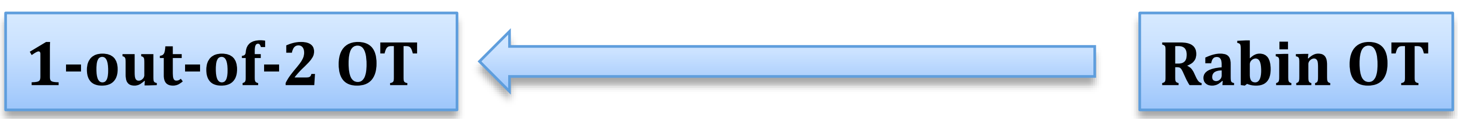
One-out-of-Two Oblivious Transfer

are "equivalent".

[Claude Crépeau. Equivalence between two flavours of oblivious transfer, 1988]


1-out-of-2 OT
Rabin OT
Rabin
input bit A
sender
receiver
choose random (A_0, A_1) such that $A_0 \oplus A_1 = A$
input bits (A_0, A_1)
sender
1-out-of-2
choose random bit R
A_R
choose a random bit B
receiver
input bit B
A_B
If $R \neq B$ then **output** $A = A_B \oplus A_R$
otherwise he has no information on
A_{1-B} so he has no information on A

It remains to show the opposite direction



input bits (A_0, A_1)

input bit B

α_1	α_2	α_3	α_4	α_5	α_6	α_7

random string of bits

k times Rabin OT

α_1	?	?	α_4	α_5	?	α_7

Let I be the set of indices of the bits that he "knows".
Let I^c be the complement of I.

if $B = 0$ send $(X_0, X_1) := (I, I^c)$
if $B = 1$ send $(X_0, X_1) := (I^c, I)$

the receiver knows only the indices in β_B

$$\beta_0 := \sum_{i \in X_0} \alpha_i$$

$$\beta_1 := \sum_{i \in X_1} \alpha_i$$

send
$(Z_0, Z_1) := (\beta_0 \oplus A_0, \beta_1 \oplus A_1)$

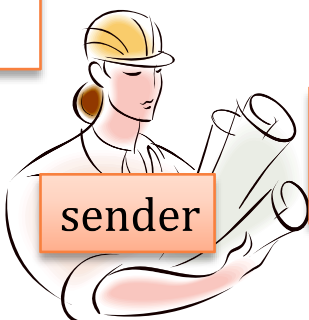
He outputs $\beta_B \oplus Z_B$

Security?

1. The learn $\boldsymbol{B}$ the **sender** would need to distinguish $\boldsymbol{I}$ from $\boldsymbol{I^c}$
2. To learn both $\boldsymbol{A_0}$ and $\boldsymbol{A_1}$ the **receiver** would need to know both $\boldsymbol{\beta_0}$ and $\boldsymbol{\beta_1}$
   This is possible only if he knows all $\boldsymbol{\alpha_i}$'s
   This happens with probability $\mathbf{0.5}^{\boldsymbol{k}}$.

An implementation of Rabin's OT

input bit A



a random **RSA** public key $pk := (N, e)$
$C := \mathbf{Enc}_{pk}(A)$

chooses a random x from Z_N^*

$y := x^2 \bmod N$

random z such that $z^2 = y \bmod N$

If $x = \pm z \bmod N$ output ?

otherwise $\mathbf{gcd}(x - z, N)$ is a non-trivial factor of N **hence** the receiver can decrypt A from C.
Output A

Remember the proof that computing square root is equivalent to factoring?
We used the reasoning:

1. with probability 0.5 we have $x \neq \pm z \bmod N$
2. if $x \neq \pm z \bmod N$ then $\mathbf{gcd}(x - z, N)$ is a non-trivial factor of N

Is it secure?

Against **passive cheating**?

YES!

Against **active cheating**?

Not so clear...

The sender acts as an oracle for computing square roots modulo N.

Does it can help him?

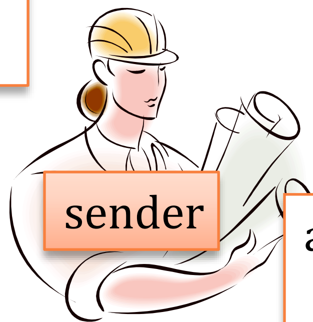
We don't know.

Solution

Add an intermediary step in which the receiver proves **in zero-knowledge** that he knows x.

How does it look now?

input bit A



sender

receiver

a random **RSA** public key $pk := (N, e)$

$C := \mathbf{Enc}_{pk}(A)$

chooses a random x from Z_N^*

$y := x^2 \bmod N$

receiver proves in **ZK** that he knows x

random z such that $z^2 = y \bmod N$

If $x = \pm y \bmod N$ output **?**

otherwise $\gcd(x - z, N)$ is a non-trivial factor of N **hence** the receiver can decrypt A from C.

Output A

Implementation of the 1-out-of-2 OT

$(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ – public key encryption scheme

(E, D) – private key encryption scheme

A_0

A_1

1. generates two pairs
(sk_0, pk_0)
(sk_1, pk_1)

pk_0 , pk_1

B

$X := \mathbf{Enc}(pk_B, K)$

2. generates a random symmetric key K

two cases:

	$B = 0$	$B = 1$
$K_0 =$	K	*"random"*
$K_1 =$	*"random"*	K

3. computes:

$K_0 := \mathbf{Dec}(sk_0, X)$
$K_1 := \mathbf{Dec}(sk_1, X)$

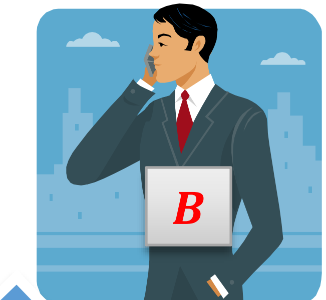
$C_0 := E(K_0, A_0)$
$C_1 := E(K_1, A_1)$

C_0 , C_1

4. computes A_B as:
$A_B = D(K, C_B)$

How to solve the love problem of Alice and Bob using OT?

Sets $(A_0, A_1) := (0, A)$



1-out-of-2 OT

the output of Bob is equal to $\mathbf{1}$
iff $A = B = \mathbf{1}$,
so it is equal to $A \wedge B$
Bob just outputs it

$A \wedge B$

output $A \wedge B$

works, because: $A \wedge B = \mathbf{OT}((\mathbf{0}, A), B)$

Oblivious Transfer for strings

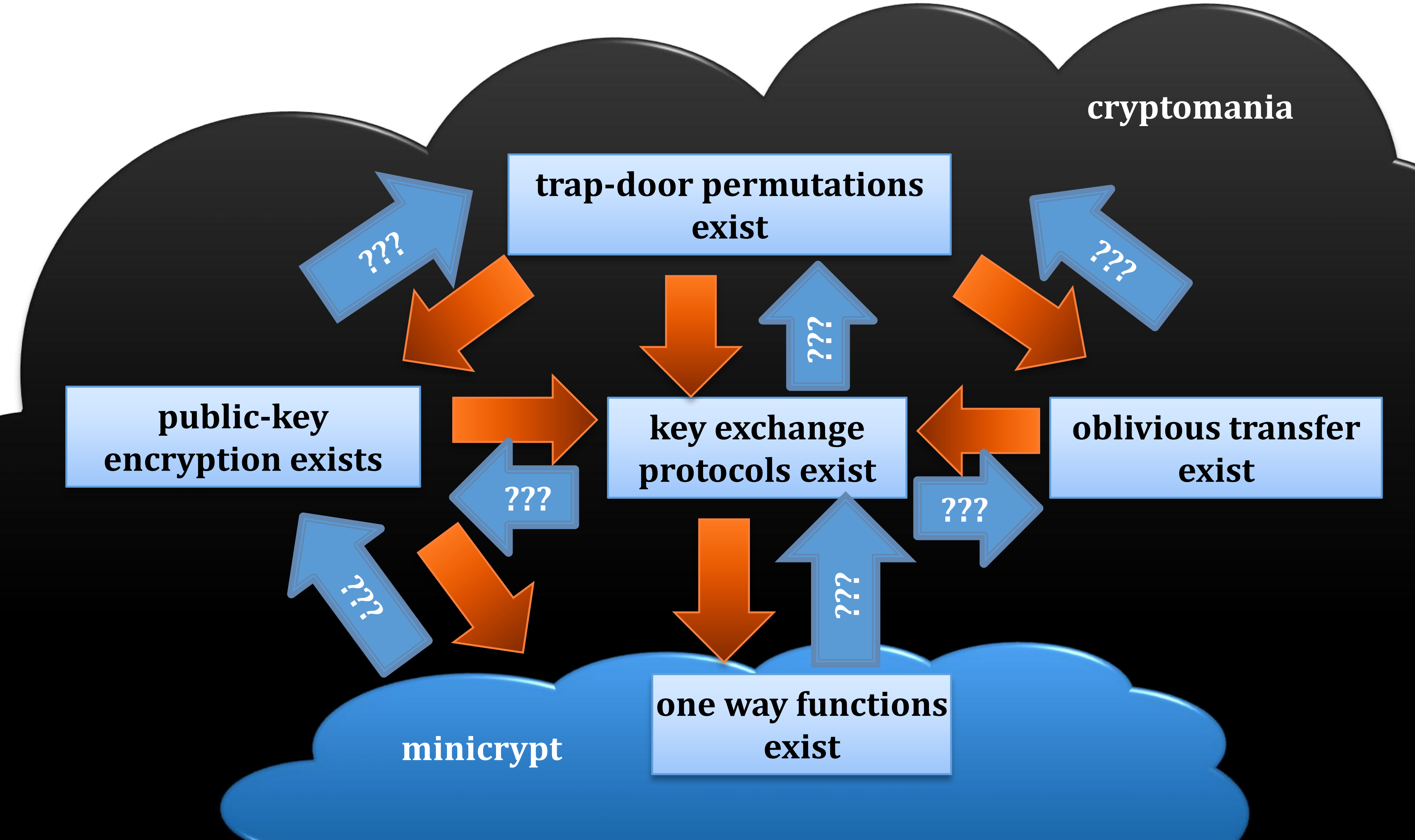
What if the sender's input $(\boldsymbol{A_0}, \boldsymbol{A_1})$ is such that each $\boldsymbol{A_i}$ is a bit-string $\left(\boldsymbol{A_i^0}, \dots, \boldsymbol{A_i^n}\right)$?

If the adversary is passive: just apply OT to each $(\boldsymbol{A_0^j}, \boldsymbol{A_1^j})$ separately (with the same $\boldsymbol{B}$).

If the adversary is active: it's more complicated, but a reduction also exists.

Is the oblivious transfer in Minicrypt?

As far as we know: **no!**



Plan

1. Introduction to two-party computation protocols
2. Definitions
3. Information-theoretic impossibility
4. Constructions
    1. oblivious transfer
    2. computing general circuits
5. Fully homomorphic encryption
6. Practical aspects
7. Private Information Retrieval

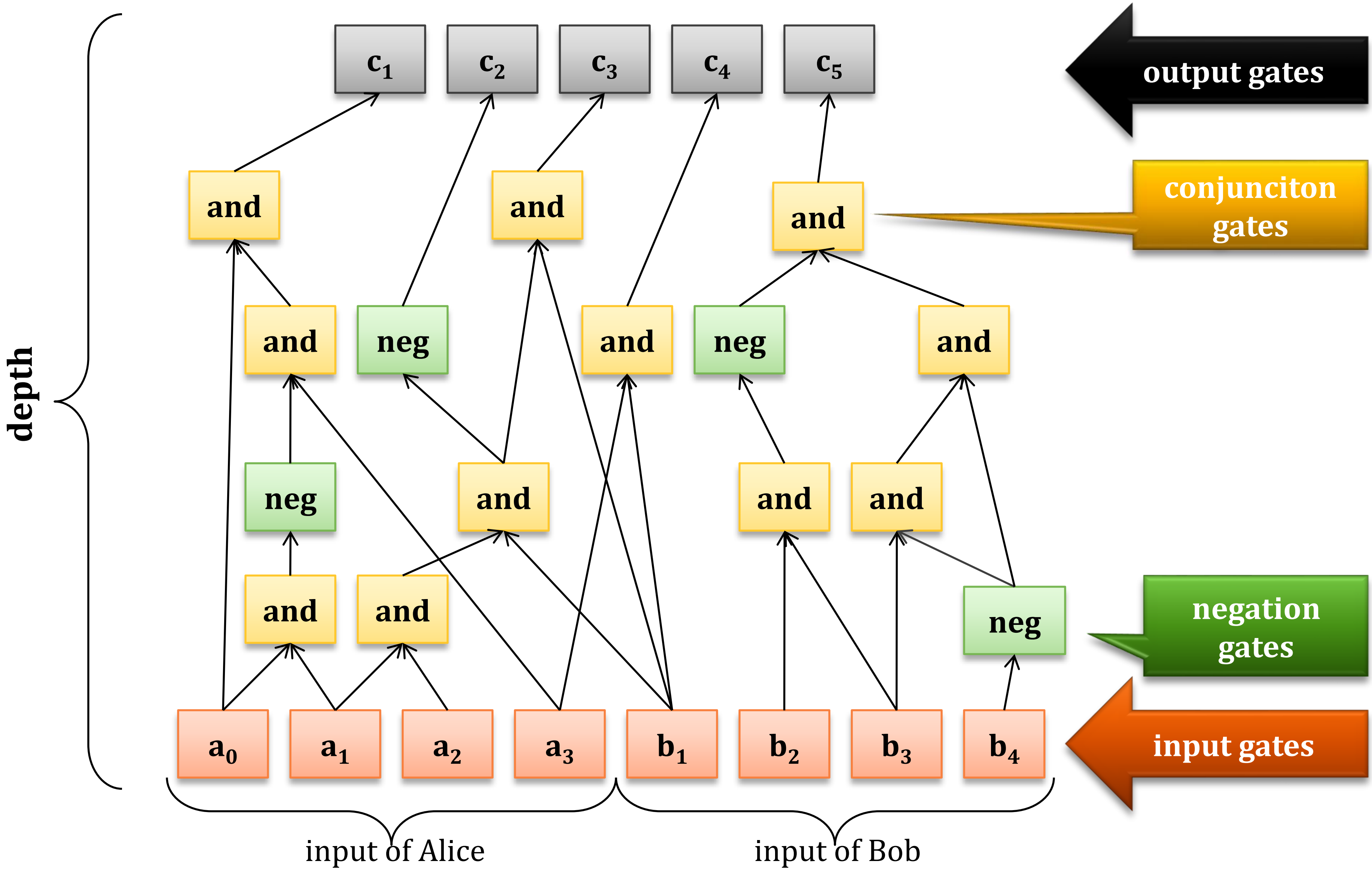
How to compute any function?

We will now show how Alice and Bob can securely compute any function ***f***.

More precisely: they can compute any function that can be computed by a **poly-time Boolean circuit**.

Boolean circuits

size: number of gates



Main idea

"Yao's Garbled circuits" (Andrew Yao, FOCS'86):

Alice "encrypts" the circuit together with her input and sends it to **Bob**.

Bob adds his input and computes the circuit **gate-by-gate**.

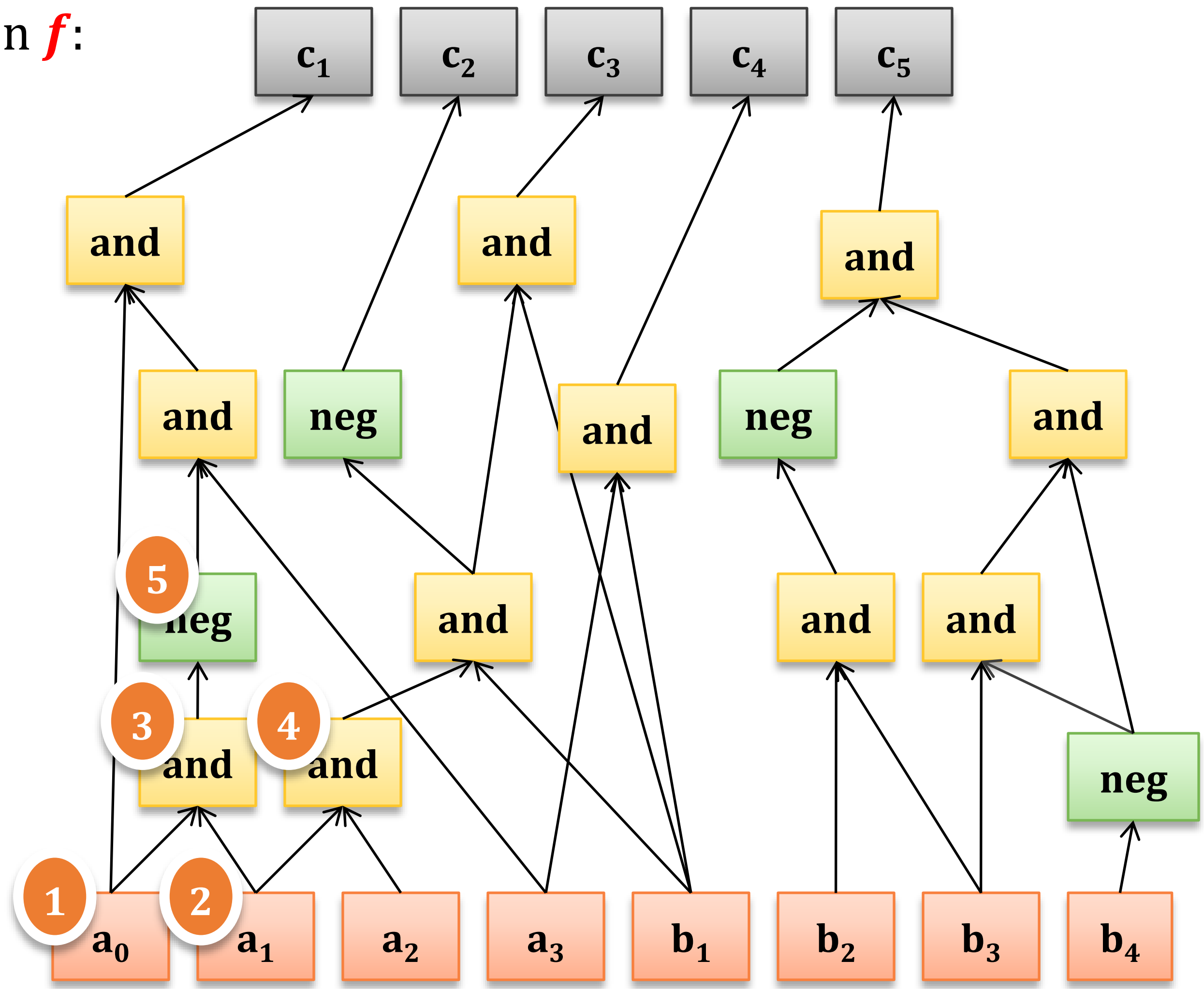
They do it in such a way that **the values on the gates remain secret** (except of the output gates)

Simplifying assumptions:

- Dishonest parties are *honest-but-curious*.
- Only Bob learns the output.

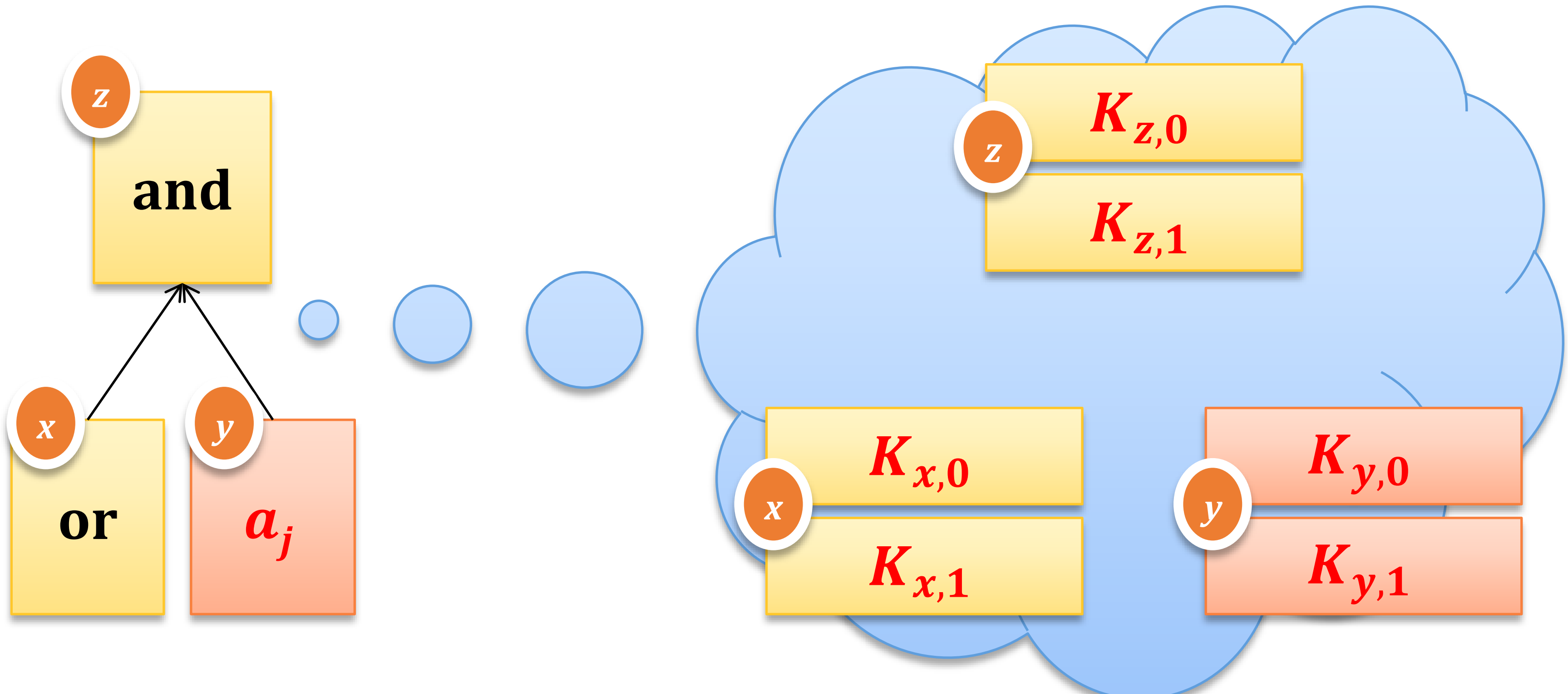
Let's number the gates

function ***f***:



Step 1: key generation

For every gate (except of the output) **Alice** chooses two random symmetric keys.



Alice does **not** send these keys to **Bob**.

Question

How to encrypt a message

$$\boldsymbol{M}$$

in such a way that in order to decrypt it one needs to know **two keys** $\boldsymbol{K_0}$ and $\boldsymbol{K_1}$?

Answer

encrypt twice:

$$\boldsymbol{E(K_0, E(K_1, M))}$$

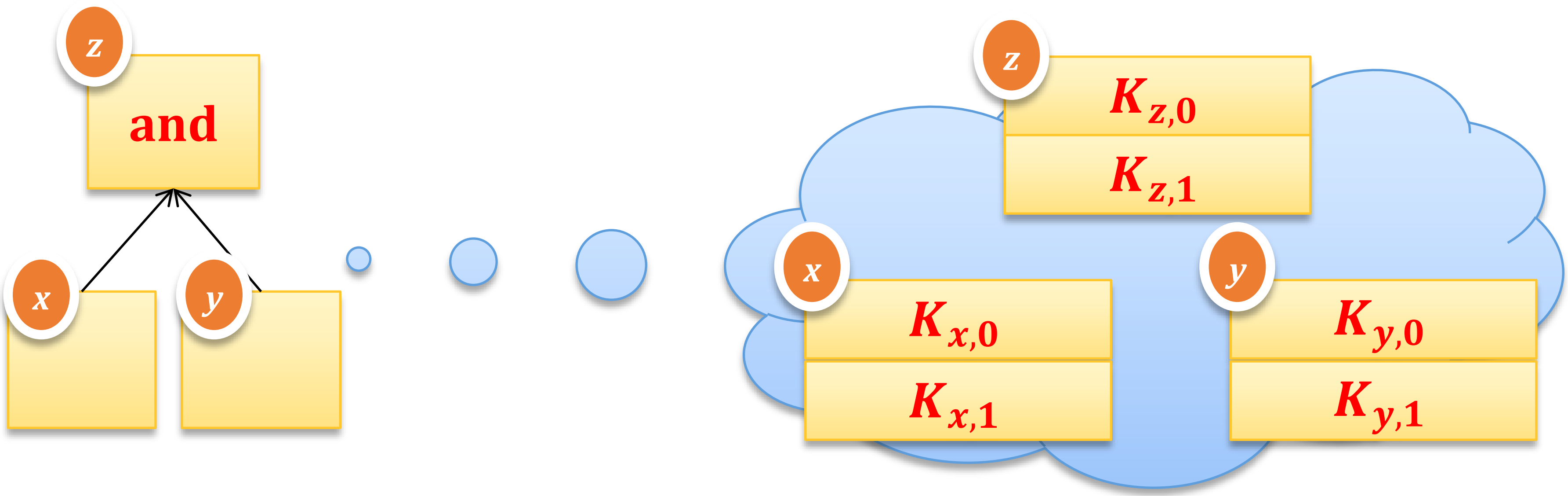
Another assumption

Let's assume that the encryption scheme $(\boldsymbol{E}, \boldsymbol{D})$ is such that decrypting

$$\boldsymbol{C} = \boldsymbol{E}(\boldsymbol{K}, \boldsymbol{M})$$

with a random key $\boldsymbol{K'}$ yields **error** ($\perp$) with overwhelming probability.

Step 2: encrypting keys



x	y	x and Y	encrypted keys
0	0	0	$E(K_{x,0}, E(K_{y,0}, K_{z,0}))$
0	1	0	$E(K_{x,0}, E(K_{y,1}, K_{z,0}))$
1	0	0	$E(K_{x,1}, E(K_{y,0}, K_{z,0}))$
1	1	1	$E(\mathrm{K}_{x,1}, E(K_{y,1}, K_{z,1}))$

analogously for the **xor** and **neg** gates

Main idea

x	y	x and Y	encrypted keys
0	0	0	$E(K_{x,0}, E(K_{y,0}, K_{z,0}))$
0	1	0	$E(K_{x,0}, E(K_{y,1}, K_{z,0}))$
1	0	0	$E(K_{x,1}, E(K_{y,0}, K_{z,0}))$
1	1	1	$E(\mathrm{K}_{\mathrm{x},1}, E(K_{y,1}, K_{z,1}))$

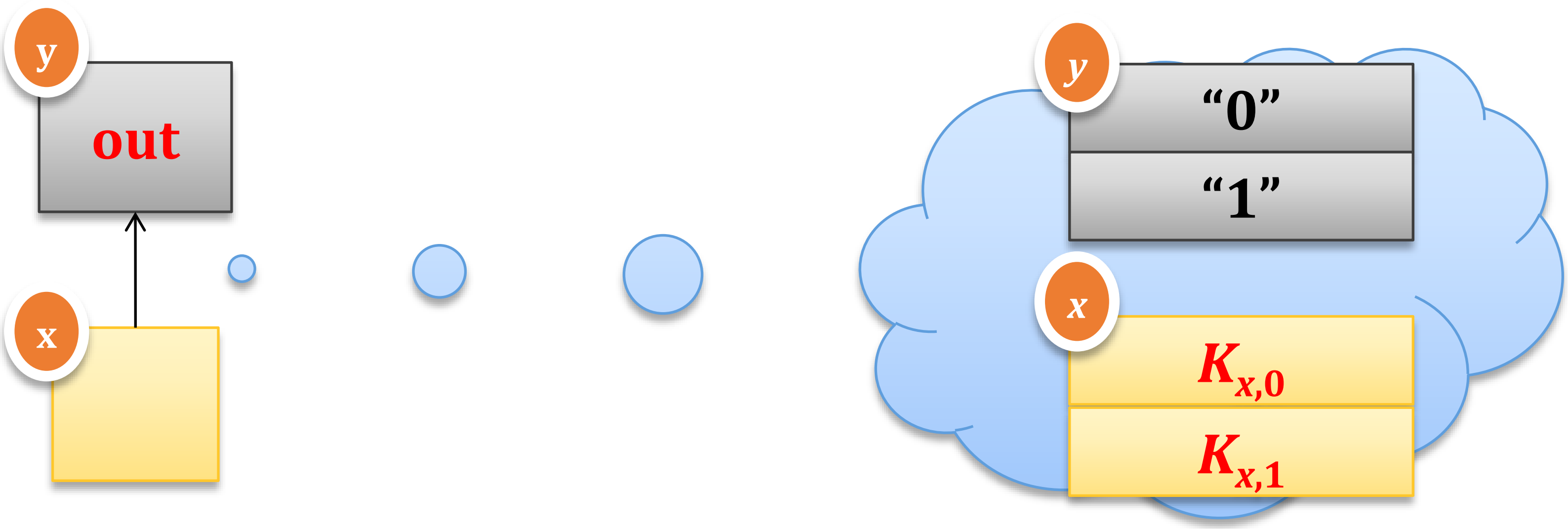
If one knows

$$K_{x,a} \text{ and } K_{y,b}$$

then one is able to decrypt **only** $K_{z,c}$ such that $c = a \wedge b$

(all the other $K_{z,i}$'s decrypt to $\perp$)

Output gates



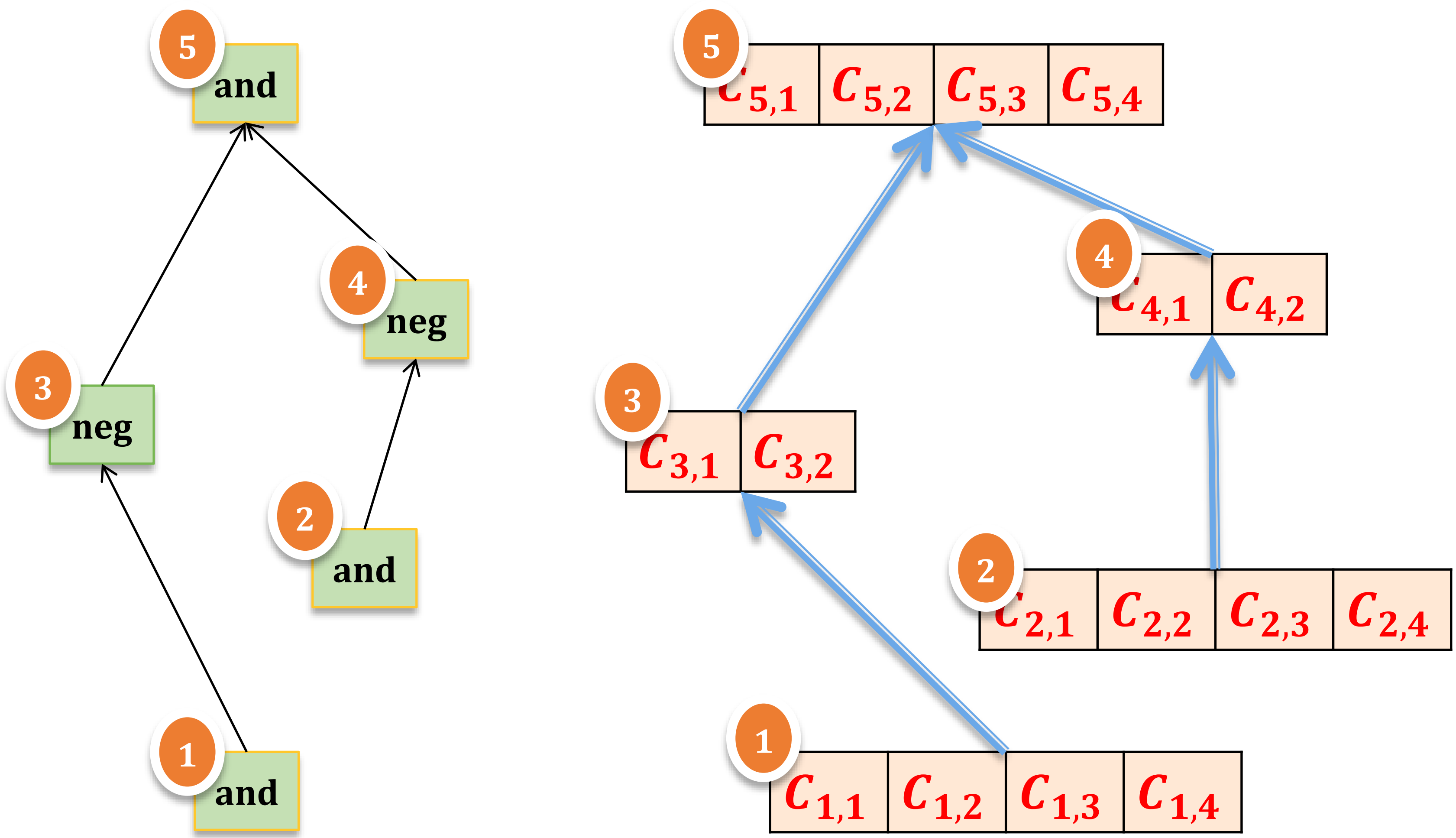
x	ciphertexts
0	$E(K_{x,0}, \text{"0"})$
1	$E(K_{x,1}, \text{"1"})$

Step 3: sending ciphertexts

For every gate **Alice** randomly permutes “encrypted keys” and sends them to **Bob**.

encrypted keys	
$E(K_{x,0}, E(K_{y,0}, K_{z,0}))$	$C_{z,1}$
$E(K_{x,0}, E(K_{y,1}, K_{z,0}))$	$C_{z,2}$
$E(K_{x,1}, E(K_{y,0}, K_{z,0}))$	$C_{z,3}$
$E(\mathrm{K}_{x,1}, E(K_{y,1}, K_{z,1}))$	$C_{z,4}$

The situation: Bob knows 2 or 4 ciphertexts for each gate

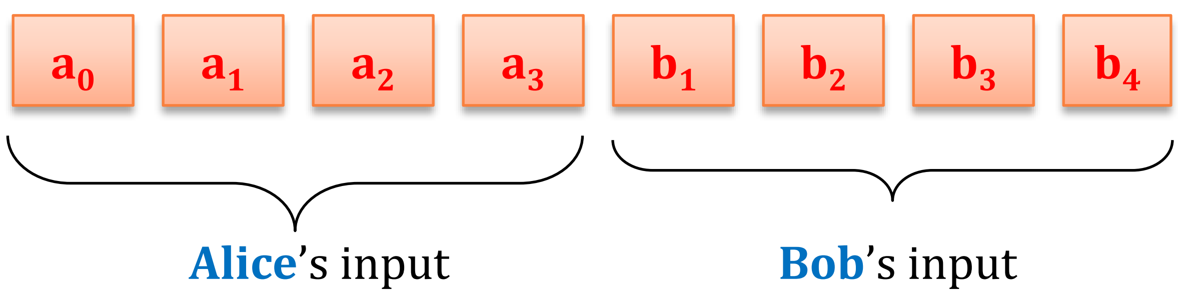


How can Bob compute the output?

Our method: decrypt the circuit “bottom up” to obtain the keys that decrypt the output.

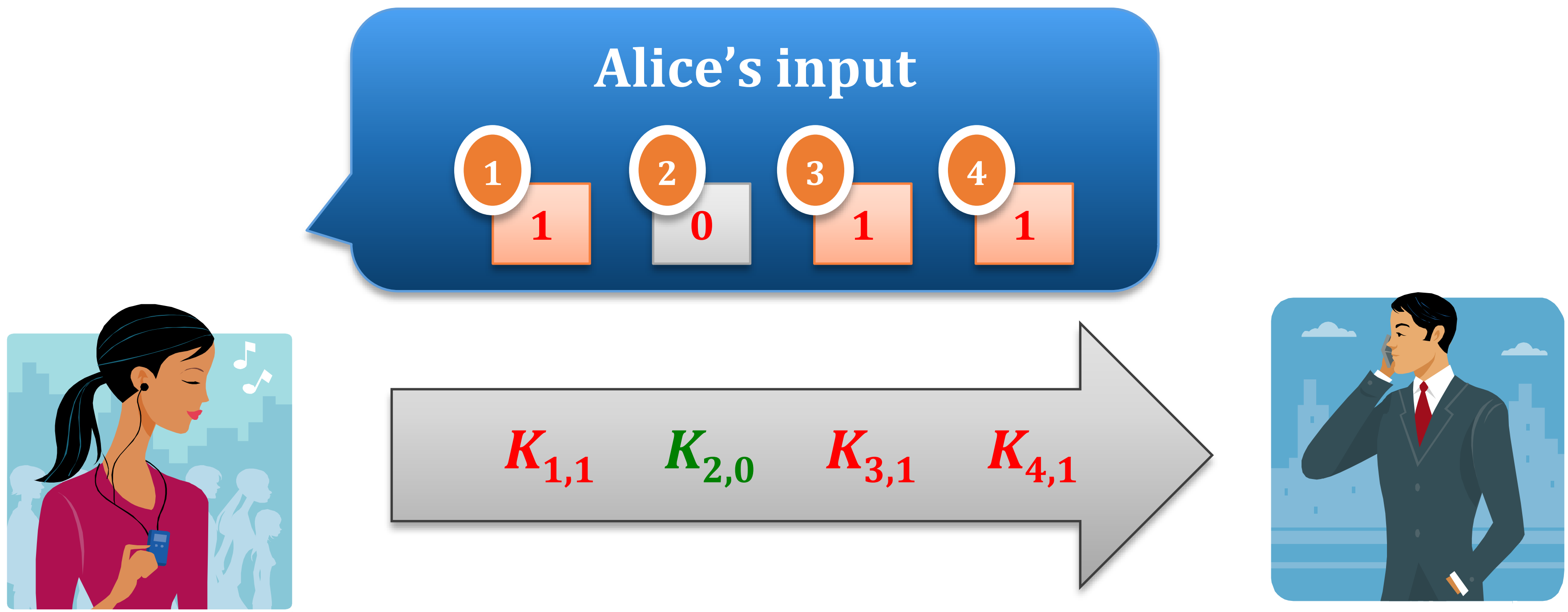
In order to start Bob needs to learn **the keys that correspond to the input gates**.

Recall that the input gates “belong” either to **Alice** or to **Bob**.



There is no problem with Alice's input

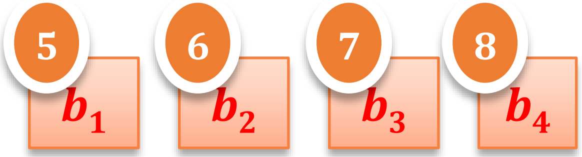
Step 4: **Alice** sends to Bob the keys that correspond to her input bits.



Note: since the gates are permuted **Bob** does not learn if he got a key that corresponds to **0** or to **1**.

How to deal with Bob's input?

$K_{5,0}$	$K_{6,0}$	$K_{7,0}$	$K_{8,0}$
$K_{5,1}$	$K_{6,1}$	$K_{7,1}$	$K_{8,1}$

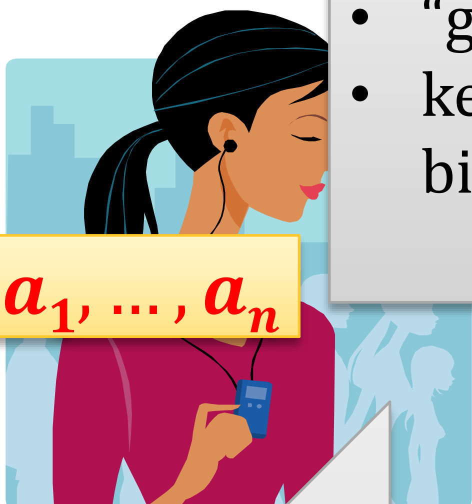


Problem: **Bob** cannot ask **Alice** to send him the keys that correspond to his input (because he would reveal his input to her).

On the other hand: **Alice** cannot send him both keys (because then he would he able to compute ***f*** on different inputs).

Solution: **1-out-of-2 Oblivious Transfer**!

Yao's method summarized



$a_1, \dots, a_n$

- "garbled" circuit computing f
- keys corresponding to input bits $a_1, \dots, a_n$

m times oblivious transfer (for each bit b_i)

$b_1, \dots, b_m$

computes the circuit bottom up and learns the output

Plan

1. Introduction to two-party computation protocols
2. Definitions
3. Information-theoretic impossibility
4. Constructions
5. Fully homomorphic encryption
6. Practical aspects
7. Private Information Retrieval

A problem

Yao's protocol has a high communication complexity:

Alice needs to send the entire encrypted circuit to **Bob**.

Can we do better?

An idea

If we could construct an encryption scheme

homomorphic with respect to field operations

then secure function evaluation would be simple.

Fully homomorphic encryption:
(assume that the set of messages is a field)

(sk, pk) – a key pair

does not know sk

can compute

$\mathrm{Enc}(pk, X)$

$\mathrm{Enc}(pk, Y)$

$\mathrm{Enc}(pk, X + Y)$

$\mathrm{Enc}(pk, X \times Y)$

How to compute f using such a cipher?

Assume that the field is Z_2.

Then **logical conjunction** is equal to **multiplication** and **negation** equals to "**adding 1**".

input: $a_1, \dots, a_n$

$pk, a_1, \dots, a_n$

input: $b_1, \dots, b_m$

1. generates a pair (sk, pk)
2. for every i computes:

$$c_i = \mathbf{Enc}(pk, a_i)$$

3. for every i computes:

$$d_i = \mathbf{Enc}(pk, b_i)$$

4. computes f using the homomorphism.

computes the ciphertext c that corresponds to the output

5. computes the result as $\mathbf{Dec}(sk, c)$

Do such ciphers exist?

Some well-known ciphers are homomorphic with respect to **one** field operation, e.g.:

- **RSA** is homomorphic with respect to multiplication,
- **Paillier encryption** is homomorphic with respect to addition.

Fully Homomorphic Encryption – see **Chapter 7**

Plan

1. Introduction to two-party computation protocols
2. Definitions
3. Information-theoretic impossibility
4. Constructions
5. Fully homomorphic encryption
6. Practical aspects
7. Private Information Retrieval

Practicality?

In practice this protocol is extremely inefficient.

But it shows that some things **in principle** can be done.

Research direction

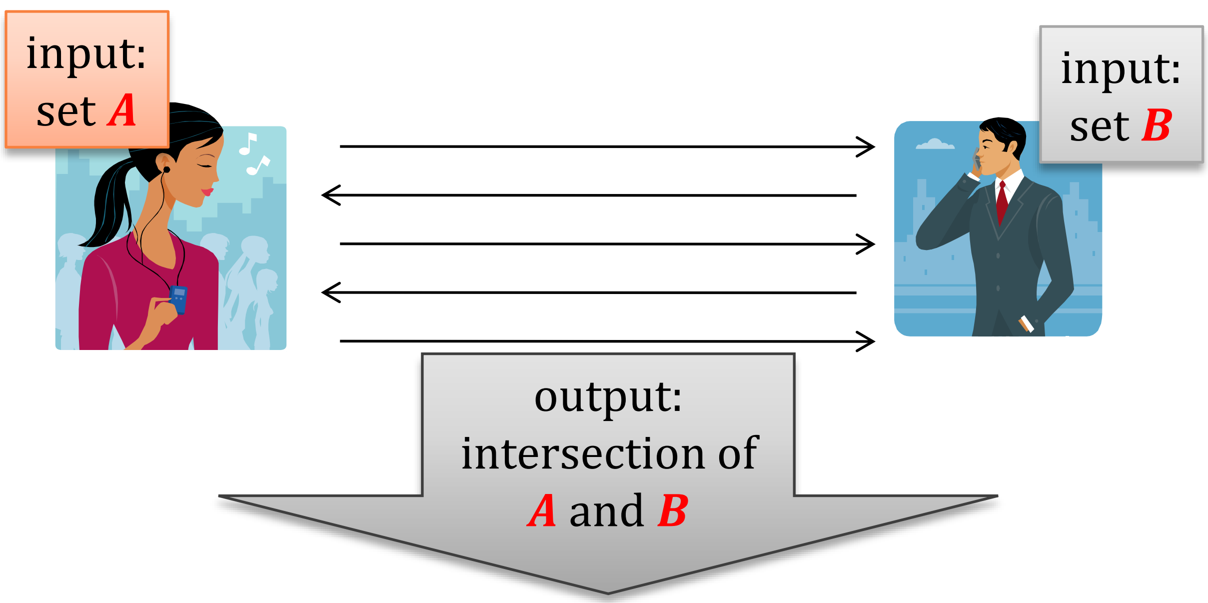
Construct protocols (for concrete problems) that are efficient.

Example

Michael J. Freedman, Kobbi Nissim, Benny Pinkas: **Efficient Private Matching and Set Intersection. EUROCRYPT 2004**

Set intersection:

Alice and Bob want to see which friends they have in common (without revealing to each other their lists of friends)

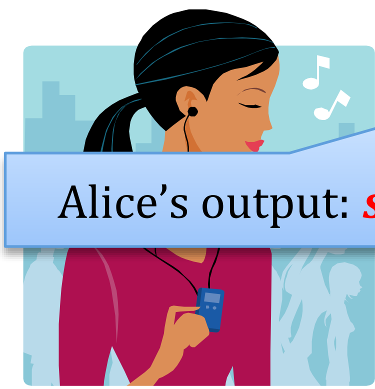


Another popular practical scenario

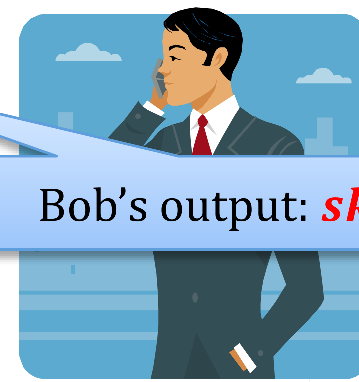
(Gen, Sign, Vrfy) – signature scheme

public output: pk

"distributed key generation"



Alice's output: sk_A



Bob's output: sk_B

such that

$sk_A \oplus sk_B = sk$

signing a message m

public output: $\mathbf{Sign}_{sk}(m)$

The same works for **public-key encryption**!

A natural question?

What if the number of parties is greater than **2**?

Solutions for this also exist!

(see **Chapter 12**)

Plan

1. Introduction to two-party computation protocols
2. Definitions
3. Information-theoretic impossibility
4. Constructions
5. Fully homomorphic encryption
6. Practical aspects
7. Private Information Retrieval
    1. introduction
    2. constructions

Private Information Retrieval (PIR)

In a nutshell:

a protocol that allows to access a database without revealing what is accessed.

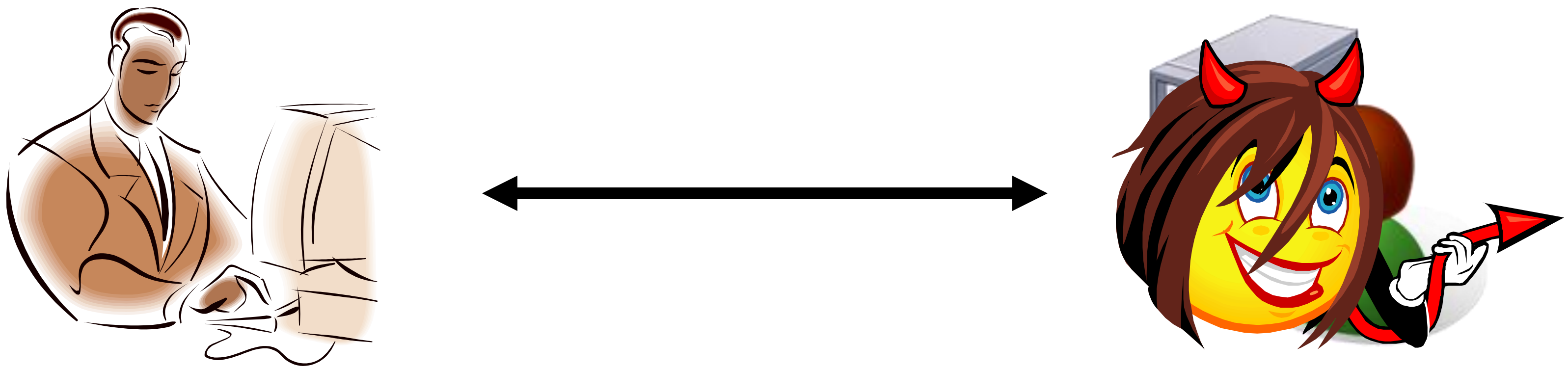
Main difference with the secure two-party computations:

1. secrecy of only one party is protected,
2. **on the other hand**: there is a restriction on **communication complexity**.

PIR was introduced in:

B. Chor, E. Kushilevitz, O. Goldreich and M. Sudan, **Private Information Retrieval**, Journal of ACM, 1998

Our settings

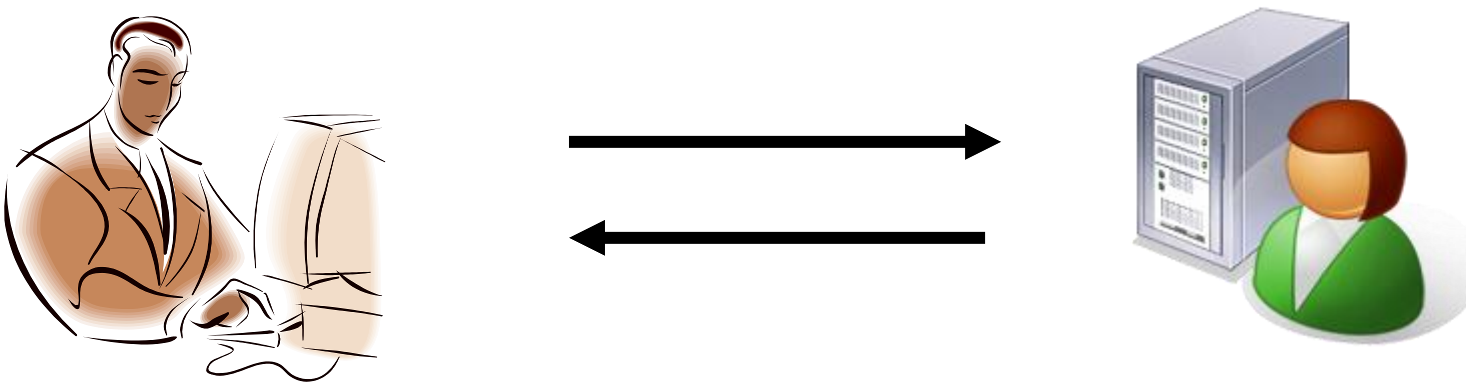


user ***U***

database ***D***

Question

How to protect privacy of queries?



user ***U***

database ***D***

wants to retrieve some data from ***D***	shouldn't learn what ***U*** retrieved

Let's make things simple!

?

index $\boldsymbol{i = 1, \dots, w}$

database $\boldsymbol{B}$:

$\boldsymbol{B_1}$	$\boldsymbol{B_2}$		$\boldsymbol{B_i}$		$\boldsymbol{B_w}$

the user should learn $\boldsymbol{B_i}$

each $\boldsymbol{B_i \in \{0, 1\}}$

(he may also learn other $\boldsymbol{B_i}$'s)

Trivial solution

The database simply sends everything to the user!

Non-triviality

The previous solution has a drawback:

the communication complexity is huge!

Therefore we introduce the following requirement:

"**Non-triviality**":

the number of bits communicated between *U* and *D* has to be smaller than *w*.

Private Information Retrieval

polynomial time randomized interactive algorithms

input:
index $\boldsymbol{i = 1, \ldots, w}$

input:

B_1	B_2	…	B_w

This property needs to be defined more formally

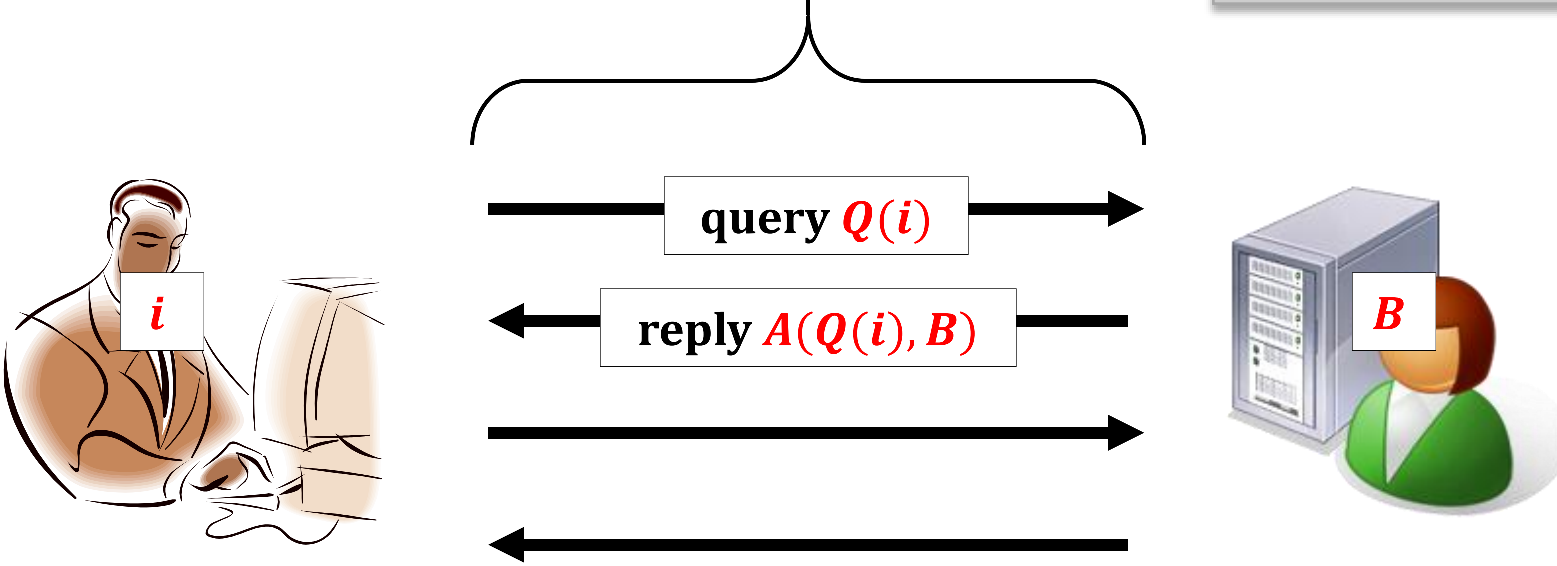
- at the end the user learns B_i ⟵ **correctness**
- the database does not learn i ⟵ **secrecy (of the user)**
- the total communication is $< w$ ⟵ **non-triviality**

<u>**Note**</u>: secrecy of the database is not required

How to define secrecy of the user [1/2]?

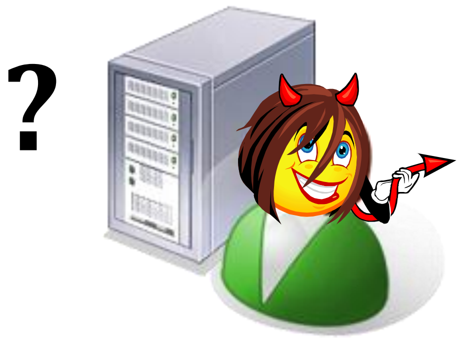
For fixed $\boldsymbol{i}$ and $\boldsymbol{B}$
$\boldsymbol{T}(\boldsymbol{i},\boldsymbol{B})$
is a **random variable**
(since the parties are randomized)

Def. $\boldsymbol{T}(\boldsymbol{i},\boldsymbol{B})$ – **transcript** of the conversation.



How to define secrecy of the user [2/2]?

Secrecy of the user: for every $\boldsymbol{i}, \boldsymbol{j} \in \{\mathbf{0}, \mathbf{1}\}$



single-round case:

it is impossible to distinguish between $\boldsymbol{Q}(\boldsymbol{i})$ and $\boldsymbol{Q}(\boldsymbol{j})$

multi-round case:

it is impossible to distinguish between $\boldsymbol{T}(\boldsymbol{i}, \boldsymbol{B})$ and $\boldsymbol{T}(\boldsymbol{j}, \boldsymbol{B})$

even if the adversary is malicious

depending on the settings: **computational** or **unconditional indistinguishability**

Computationally-secure PIR – formally

computational-secrecy:

For every $\boldsymbol{i, j} \in \{\mathbf{0}, \mathbf{1}\}$

it is impossible to distinguish
efficiently
between
$\boldsymbol{T(i, B)}$ and $\boldsymbol{T(j, B)}$

Formally: for every **polynomial-time** probabilistic algorithm $\boldsymbol{A}$ the value:

$$\left|\boldsymbol{P(A(T(i, B)) = 0) - P(A(T(j, B)) = 0)}\right|$$

should be **negligible.**

Is it possible?

Fact

Information-theoretically secure single-server **PIR** does not exist **[exercise]**.

What can be constructed is the following:

- **computationally-secure PIR** (we show it now)
- **information-theoretically secure multi-server PIR [exercise]**

PIR vs OT

PIR looks similar to the 1**-out-of-*w* OT**

Differences:

- **advantage of PIR**: **low communication complexity**
- **advantage of OT**: **privacy of the database is protected**

Can we combine both?

Yes! It’s called “**symmetric PIR**”.

Plan

1. Introduction to two-party computation protocols
2. Definitions
3. Information-theoretic impossibility
4. Constructions
5. Fully homomorphic encryption
6. Practical aspects
7. Private Information Retrieval
    1. introduction
    2. constructions

The construction

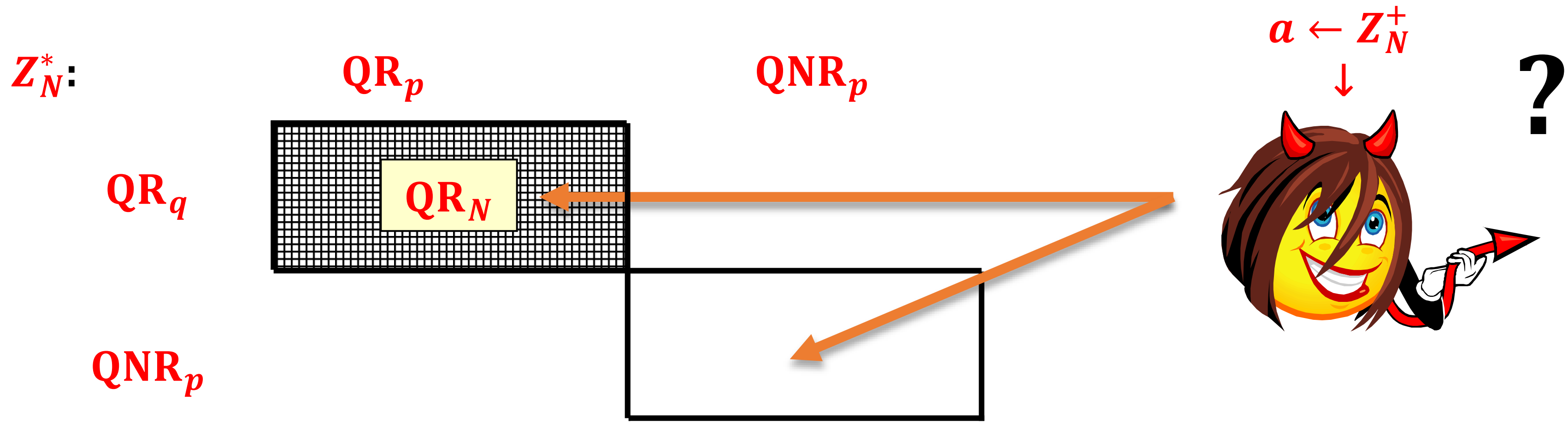
Kushilevitz and R. Ostrovsky **Replication Is NOT Needed: SINGLE Database, Computationally-Private Information Retrieval**, FOCS 1997

based on the **Quadratic Residuosity Assumption**.

Our presentation strategy:

1. we first present a **wrong** solution
2. then we **fix it**.

Quadratic Residuosity Assumption



Quadratic Residuosity Assumption (QRA):

For a random $a \leftarrow Z_N^+$ it is computationally hard to determine if $a \in \mathrm{QR}_N$.

Formally: for every **polynomial-time** probabilistic algorithm D the value:

$$\left| P\big(D(N,a) = Q_N(a)\big) - \frac{1}{2} \right|$$

(where $a \leftarrow Z_N^+$) is **negligible**.

Where a predicate $Q_N: Z_N^+ \rightarrow \{0,1\}$ is defined as follows:

$Q_N(a) = 0$ if $a \in \mathrm{QR}_N$

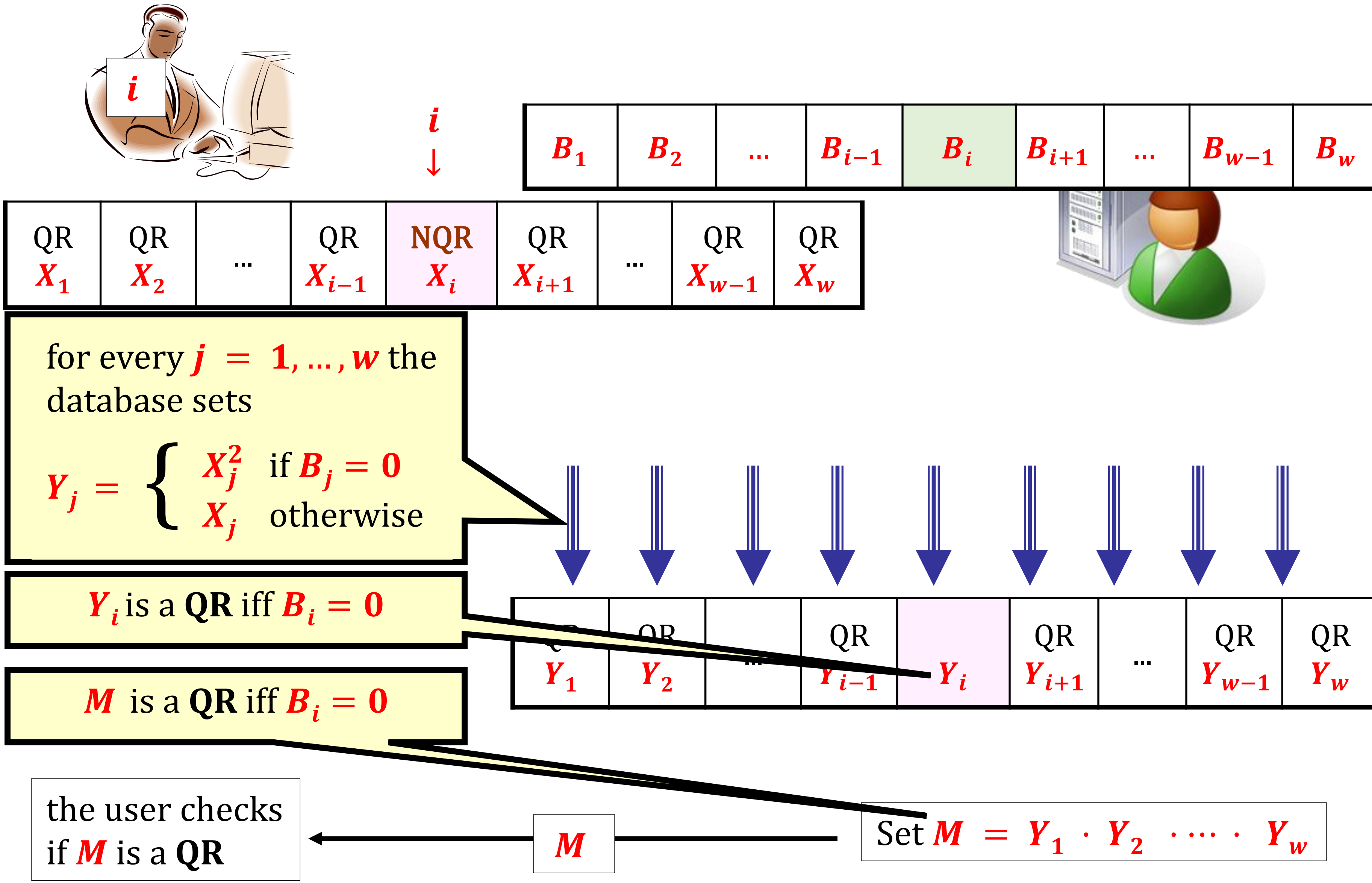
$Q_N(a) = 1$ otherwise

Homomorphism of $\boldsymbol{Q_N}$

For all $\boldsymbol{a, b \in Z_N^+}$

$$\boldsymbol{Q_N(ab) = Q_N(a) \oplus Q_N(b)}$$

First (wrong) idea



Problems!

PIR from the previous slide:

- **correctness** √
- **security**?

To learn $\boldsymbol{i}$ the database would need to distinguish **NQR** from **QR**. √

QR	QR	...	QR	NQR	QR	...	QR	QR
X_1	X_2		X_{i-1}	X_i	X_{i+1}		X_{w-1}	X_w

- **non-triviality**? doesn't hold!

<u>communication</u>:

user → database: $|B| \cdot |N|$

database → user: $|N|$

Call it:
$(|B|, 1)$–**PIR**

How to fix it?

Idea

Given:

$$(|\boldsymbol{B}|, \mathbf{1})\text{–PIR}$$

construct

$$\left(\sqrt{|\boldsymbol{B}|}, \sqrt{|\boldsymbol{B}|}\right)\text{–PIR}$$

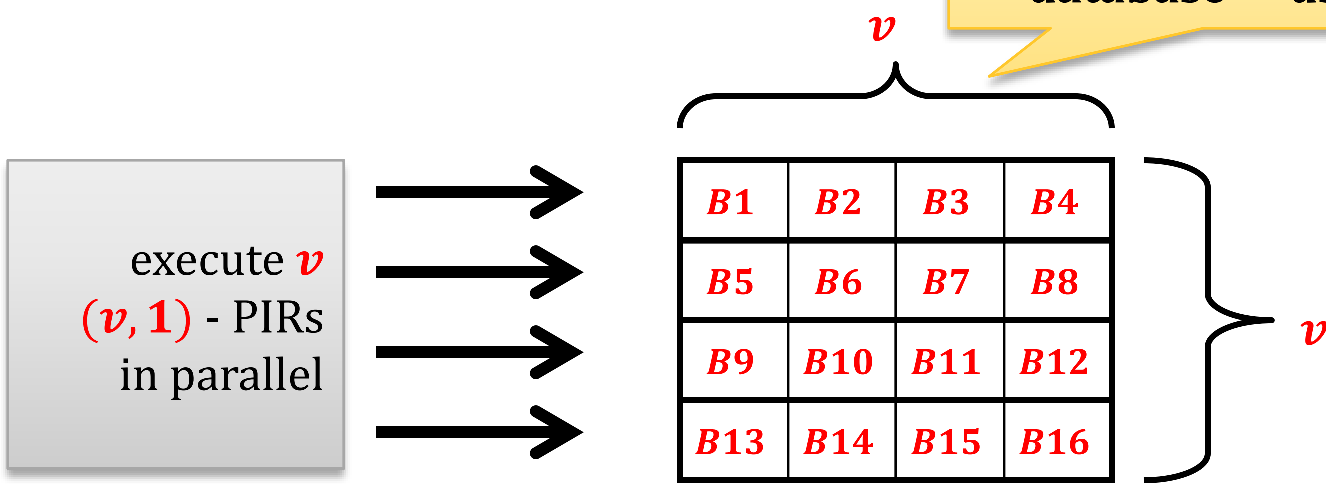
Suppose that $|\boldsymbol{B}| = \boldsymbol{v}^2$ and present $\boldsymbol{B}$ as a $\boldsymbol{v} \times \boldsymbol{v}$–matrix:

$B1$	$B2$	$B3$	$B4$	$B5$	$B6$	$B7$	$B8$	$B9$	$B10$	$B11$	$B12$	$B13$	$B14$	$B15$	$B16$

consider each row as a separate database

An improved idea

Looks even worse:
<u>communication</u>:
user → database: $\boldsymbol{v^2 \cdot |N|}$
database → user: $\boldsymbol{v \cdot |N|}$



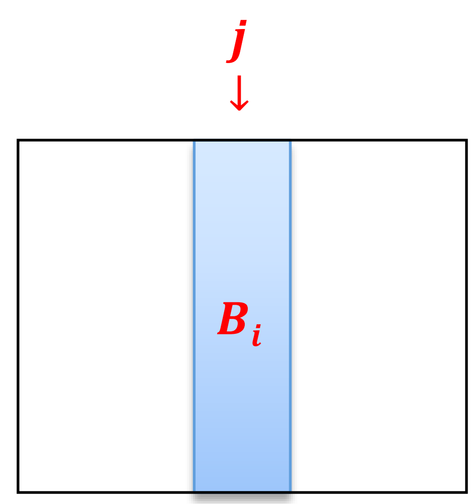
The method

Let j be the column where B_i is.

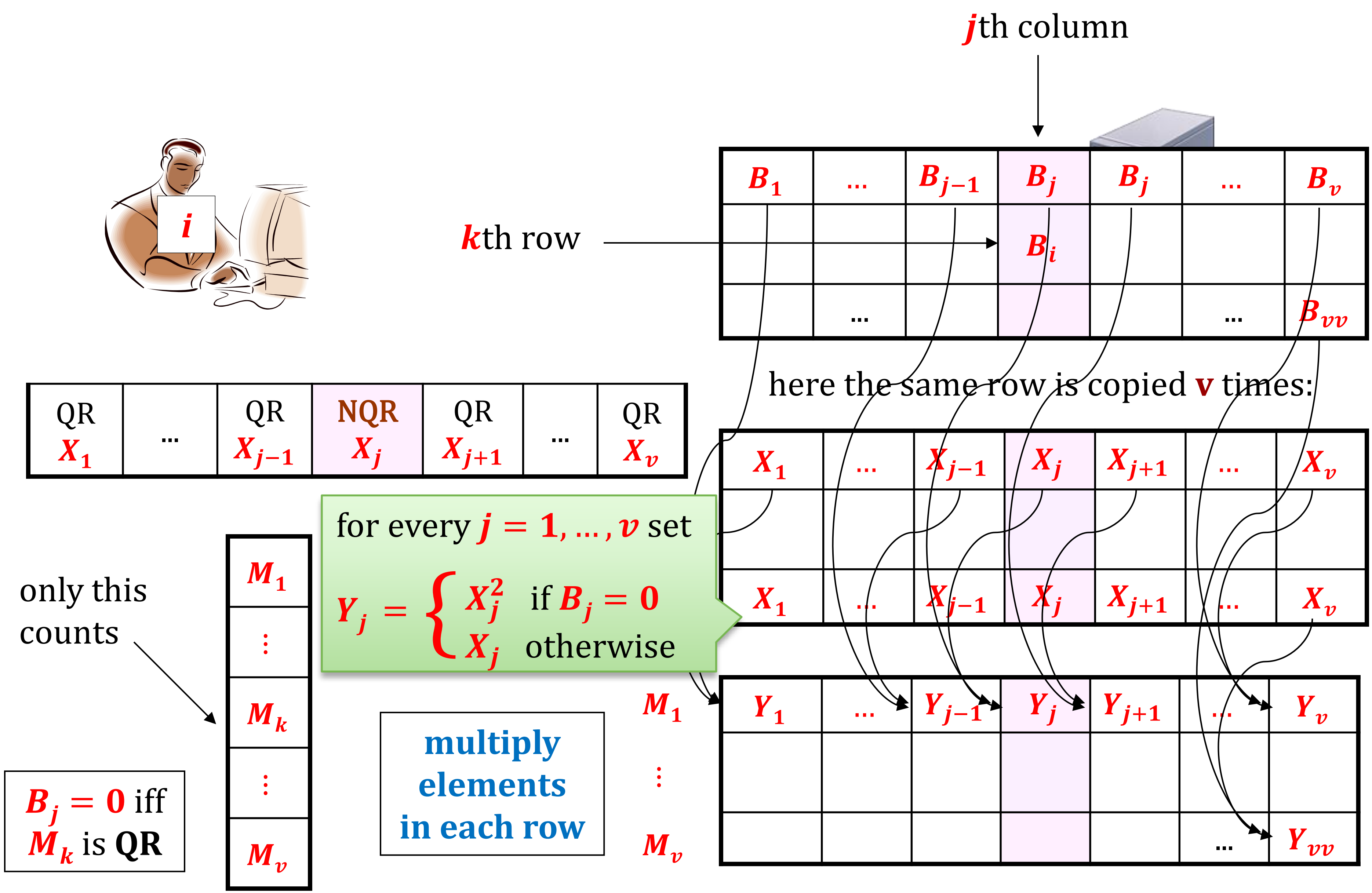
In every "row" the user asks for the jth element

So, instead of sending v queries the user can send one!

Observe: in this way the user learns
all the elements in the jth column!



Putting things together



So we are done!

PIR from the previous slide:

- **correctness** √
- **non-triviality:**
  communication complexity = $2\sqrt{|B| \cdot |N|}$ √
- **security**?
  To learn i the database would need to distinguish **NQR** from **QR**.

<u>**Formally**</u>:

from
any adversary that **breaks our scheme**
we can construct
an algorithm that **breaks QRA**

Improvements

$(X_1, \dots, Xv)$

$(M_1, \dots, M_v)$

user U

the user is interested just in one M_i.

database D

<u>Idea</u>: apply **PIR** recursively!

Extensions

- Symmetric PIR (also protect privacy of the database).

  [**Gertner, Ishai, Kushilevitz, Malkin**. 1998]
- Searching by key-words

  [**Chor, Gilboa, Naor**, 1997]
- Public-key encryption with key-word search

  [**Boneh, Di Crescenzo, Ostrovsky, Persiano**]