CSCI471/971
Modern Cryptography

MPC

Fuchun Guo

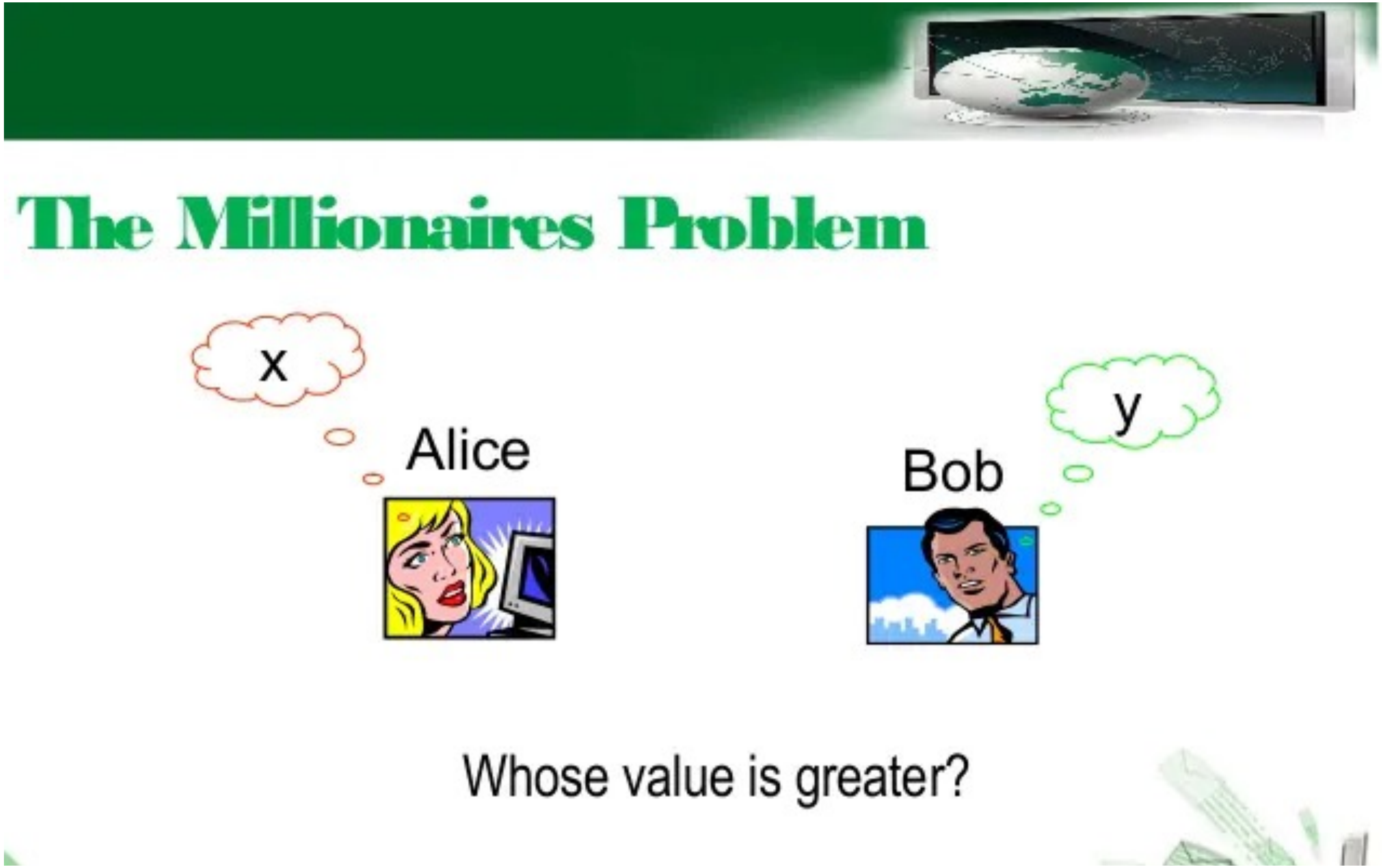
SCIT UOW

Motivation

The Millionaires Problem

x
Alice
y
Bob

Whose value is greater?



The Millionaires Problem

Whose value is greater?

Trivial Solution:

We can ask a **trusted third party** to compare and tell them the result.

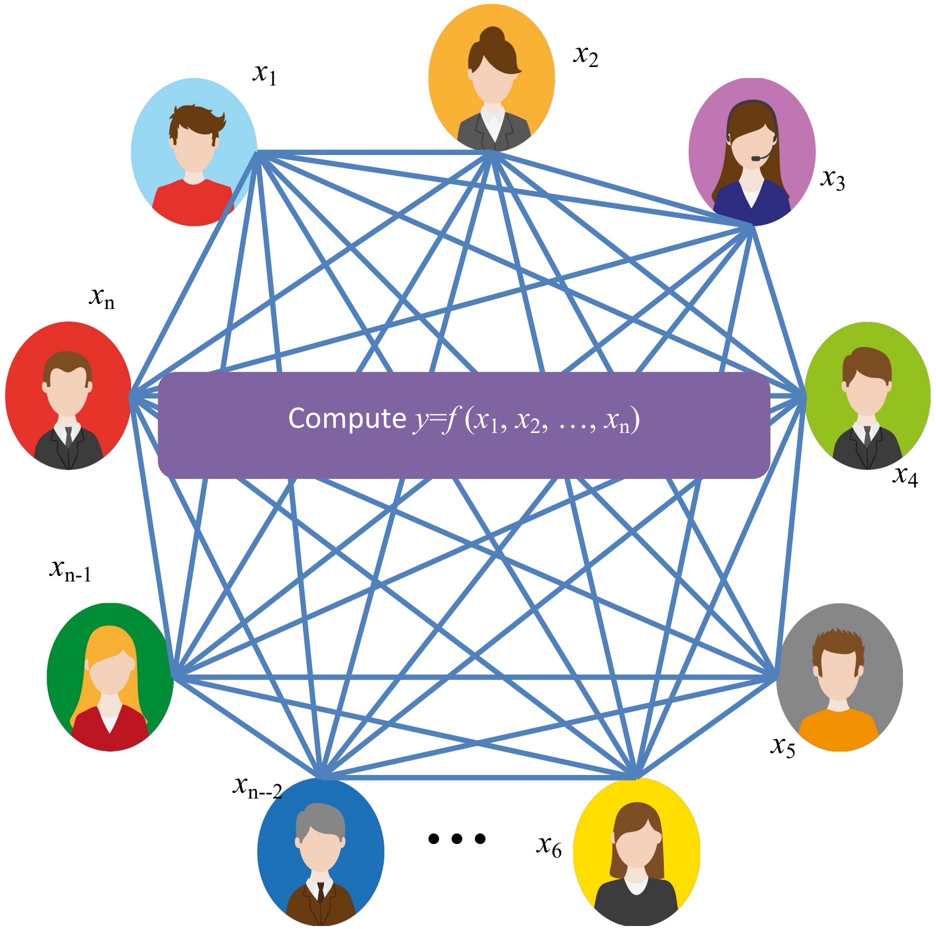
MPC

Secure Multiparty Computation

- A set of parties with private inputs wish to compute some joint function of their inputs.

- Parties wish to preserve some security properties. E.g., privacy and correctness.

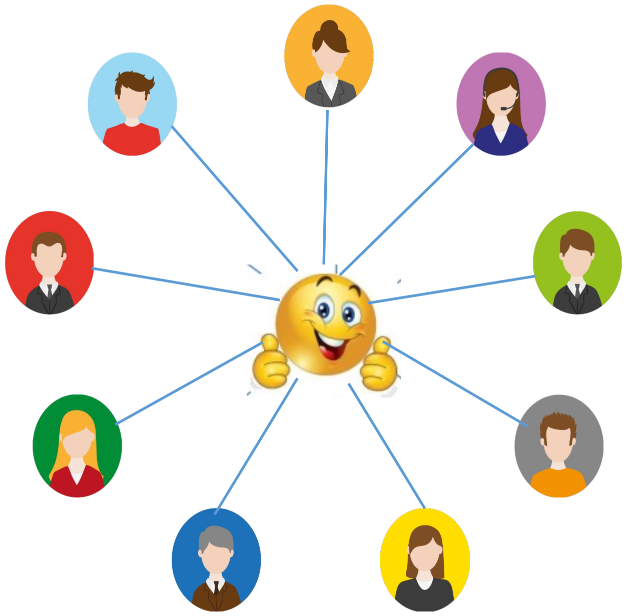
to calculate the sum of salaries



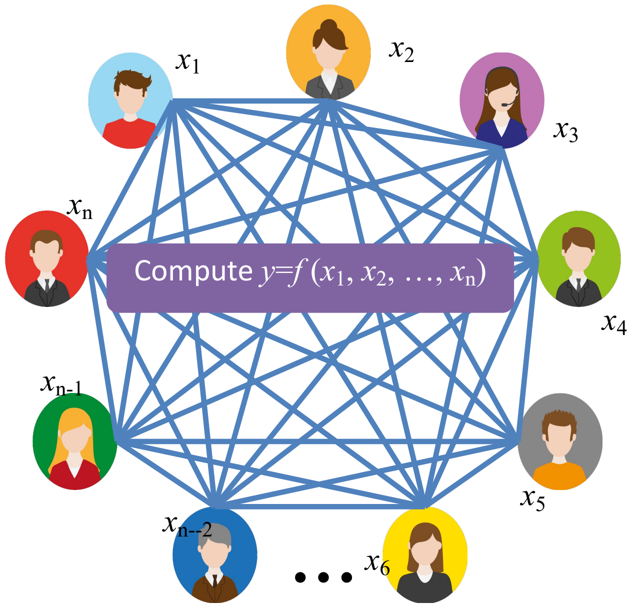
Compute $y=f(x_1, x_2, \ldots, x_n)$

Secure Multiparty Computation

- MPC is easy if there exists a trusted third party
- We want to use a protocol to replace the TTP while achieving the same security and correctness goals

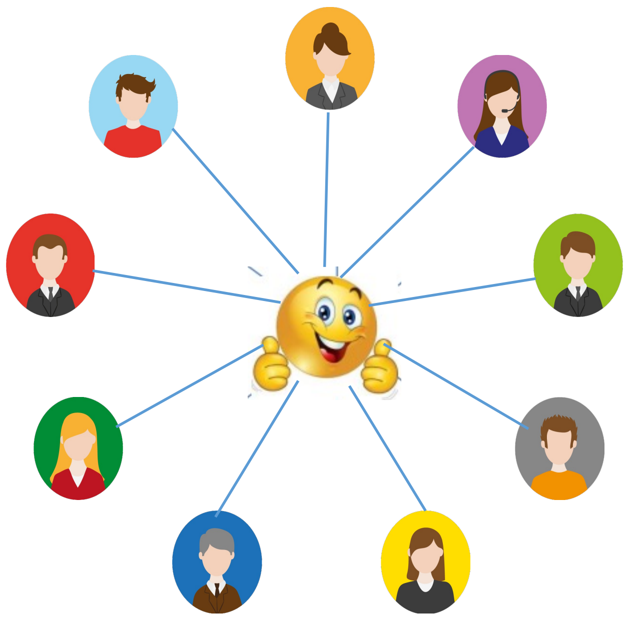


$\approx$

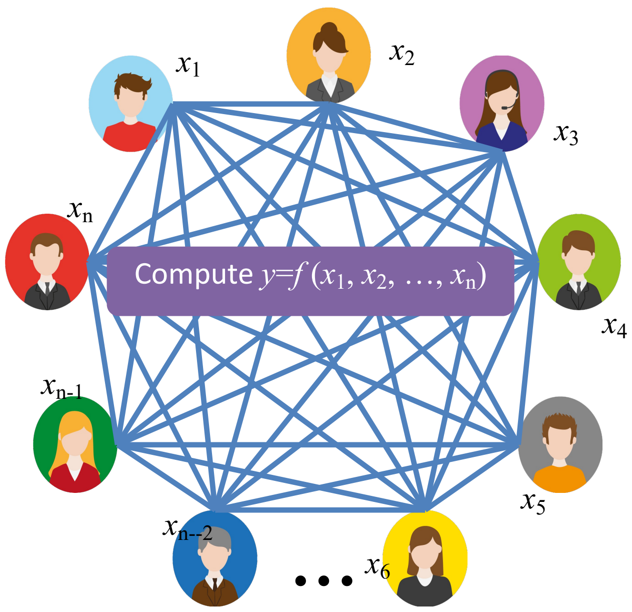


Compute $y=f(x_1, x_2, \ldots, x_n)$

Secure Multiparty Computation



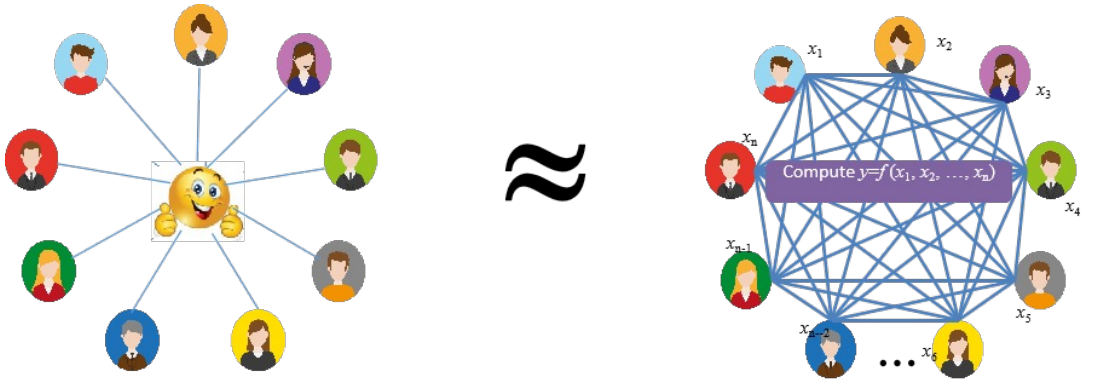
$\approx$



- Security should take account of insiders
  - Semi-honest (a.k.a. honest-but-curious)
  - Malicious
- If it is secure against insiders, it is secure against outsiders.

Secure Multiparty Computation

- Security should take account of insiders
  - Semi-honest (a.k.a. honest-but-curious)
  - Malicious



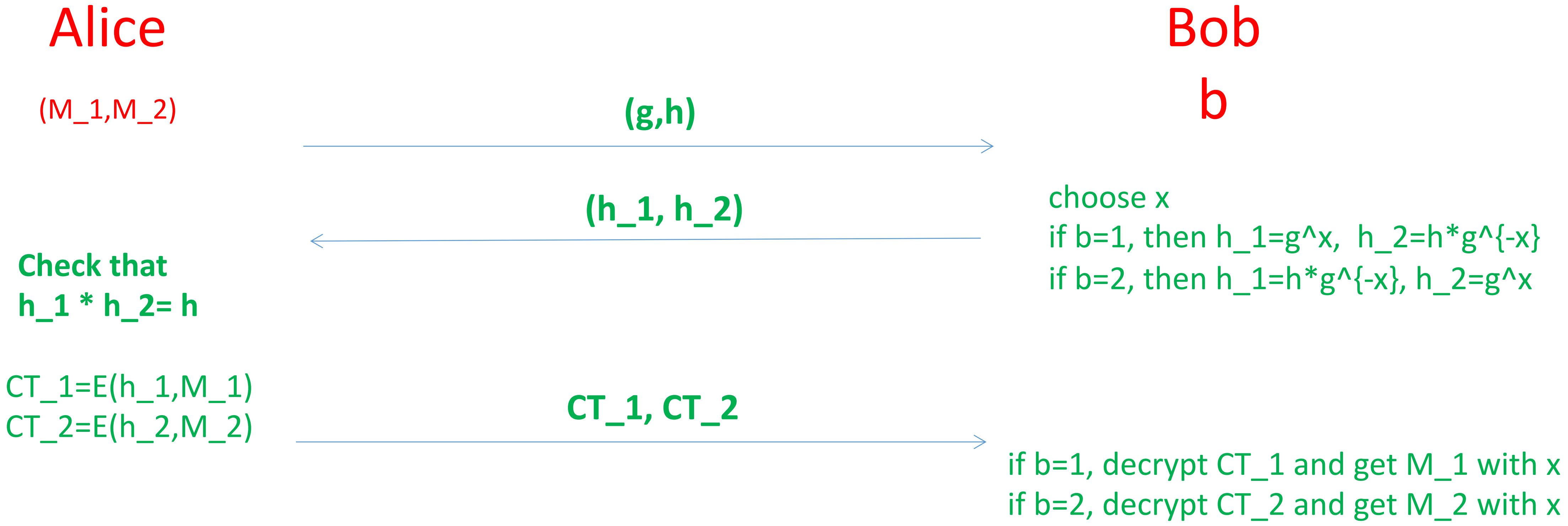
- Semi-honest (a.k.a. honest-but-curious)

The adversary follows the protocol algorithms and wants to learn more information.

- Malicious

The adversary changes the protocol algorithms to learn more information.

1-out-of-2 Oblivious Transfer (Revisit)



Alice

Bob
b

(M_1,M_2)

(g,h)

(h_1, h_2)

choose x
if b=1, then h_1=g^x, h_2=h*g^{-x}
if b=2, then h_1=h*g^{-x}, h_2=g^x

Check that
h_1 * h_2= h

CT_1=E(h_1,M_1)
CT_2=E(h_2,M_2)

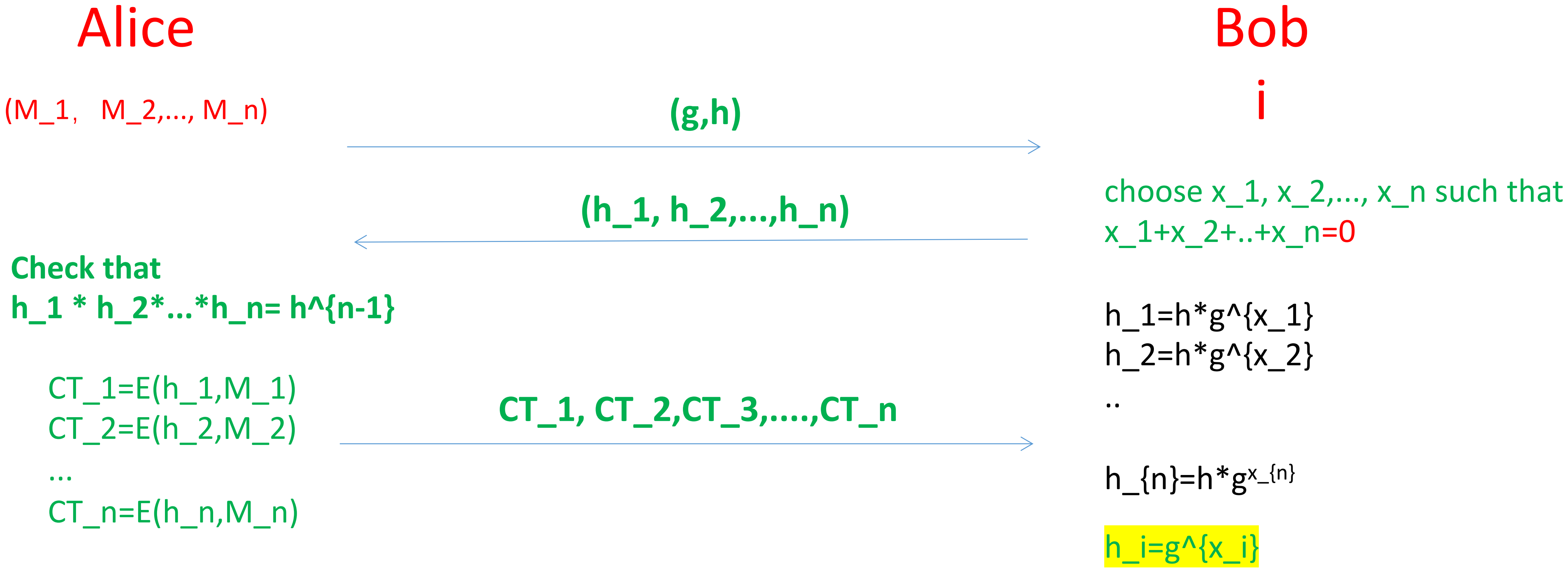
CT_1, CT_2

if b=1, decrypt CT_1 and get M_1 with x
if b=2, decrypt CT_2 and get M_2 with x

Semi-honest: if Bob follows the protocol, he cannot get two messages.

Malicious: Bob cannot get two messages no matter how h_1 and h_2 are computed.

1-out-of-n Oblivious Transfer(Weakness of Semi-Honest)



Alice

Bob

i

(M_1, M_2,..., M_n)

(g,h)

choose x_1, x_2,..., x_n such that
x_1+x_2+..+x_n=0

(h_1, h_2,...,h_n)

Check that
h_1 * h_2*...*h_n= h^{n-1}

h_1=h*g^{x_1}
h_2=h*g^{x_2}
..

h_{n}=h*g$^{\text{x\_\{n\}}}$

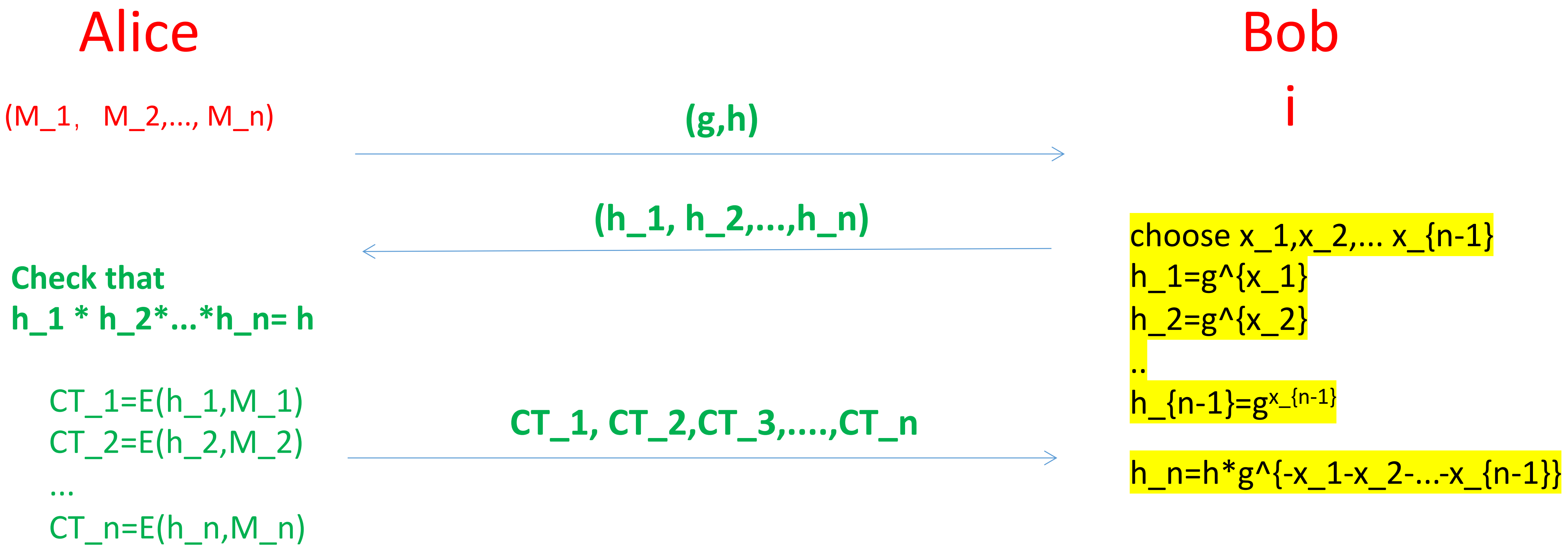
h_i=g^{x_i}

CT_1=E(h_1,M_1)
CT_2=E(h_2,M_2)
...
CT_n=E(h_n,M_n)

CT_1, CT_2,CT_3,....,CT_n

Semi-honest: if Bob follows the protocol, he can get one message only.

1-out-of-n Oblivious Transfer(Weakness of Semi-Honest)



Alice

Bob

(M_1, M_2,..., M_n)

(g,h)

i

(h_1, h_2,...,h_n)

choose x_1,x_2,... x_{n-1}
h_1=g^{x_1}
h_2=g^{x_2}
..
h_{n-1}=g$^{x_{n-1}}$

h_n=h*g^{-x_1-x_2-...-x_{n-1}}

Check that
h_1 * h_2*...*h_n= h

CT_1=E(h_1,M_1)
CT_2=E(h_2,M_2)
...
CT_n=E(h_n,M_n)

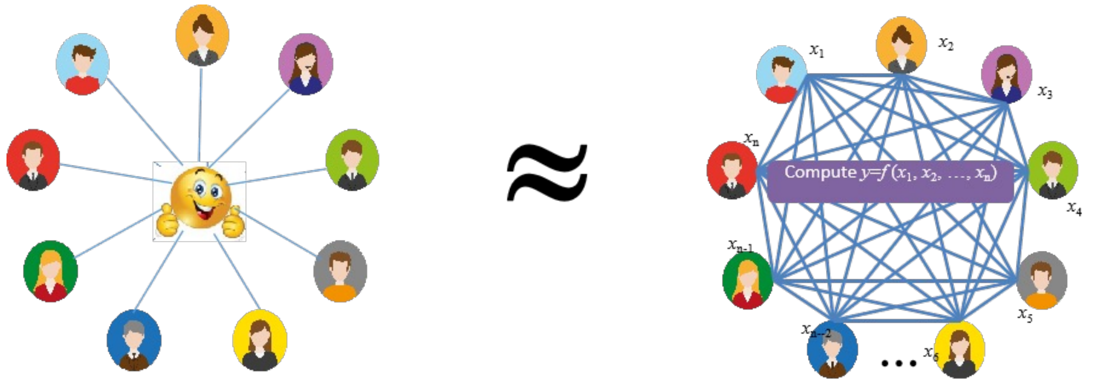
CT_1, CT_2,CT_3,....,CT_n

Malicious: if Bob changes the protocol, he will get n-1 messages.

Defining security

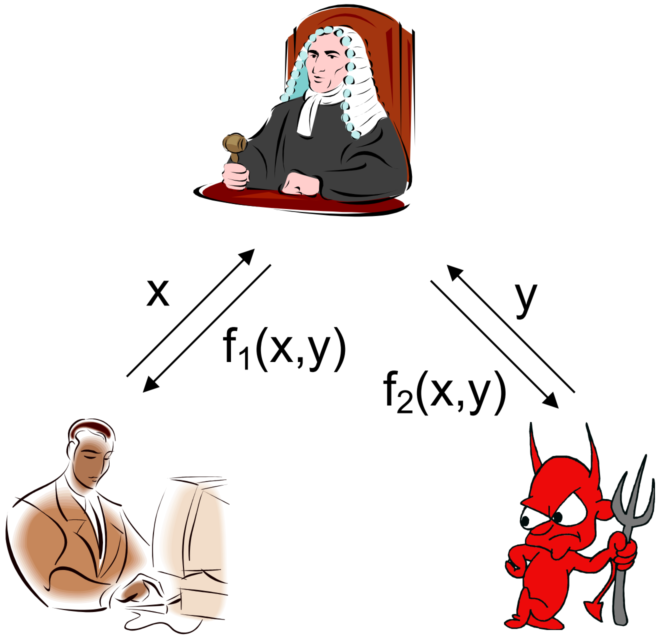
- Option 1: Could try to define security by listing several properties
  - Might depend on the function being computed
  - Might be complex to define in any case
  - How do we know we did not miss something?

- Option 2: ideal/real model
  - Define an *ideal-world* execution
  - Compare real world to ideal world



Ideal-world execution

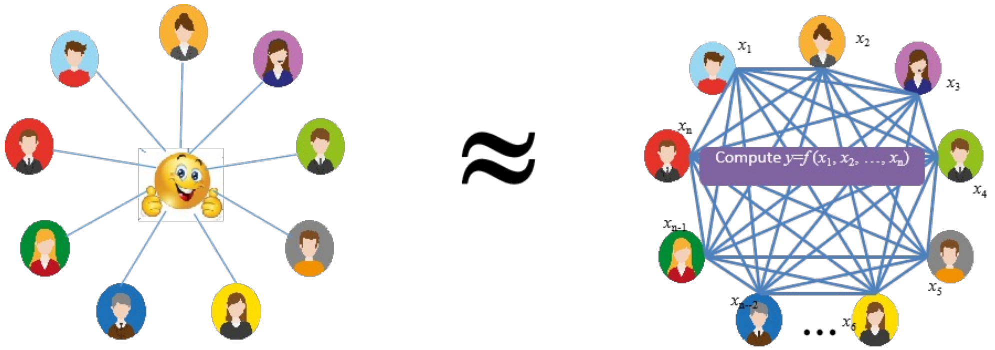
- Assume parties have access to a trusted party who does the computation for them



Defining Security – Option 2

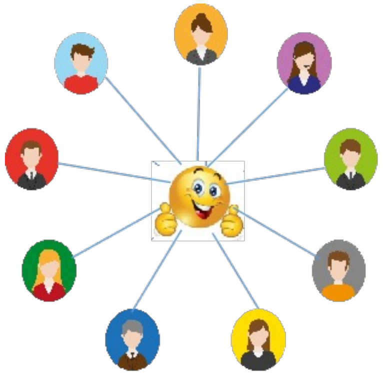
- The real/ideal model paradigm:
  - Ideal model: parties send inputs to a trusted party, who computes the function and sends the outputs.
  - Real model: parties run a real protocol with no trusted help.

- Informally: a protocol is secure if any attack on a real protocol can be carried out (or simulated) in the ideal model.
- Since essentially **no** attacks can be carried out in the ideal model, security is implied.

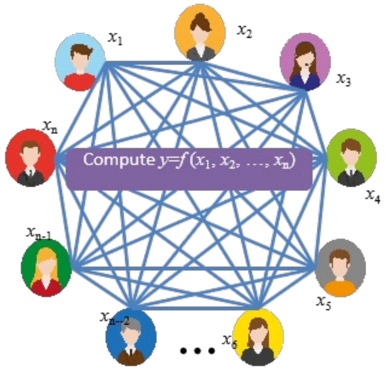


Observations

- In the ideal world:
  - *Privacy* is guaranteed (each party learns only its output)
  - *Correctness* is guaranteed (ideal party computes the correct function on inputs of the honest parties and some input from corrupted parties)
  - *Input independence* is guaranteed (corrupted parties choose inputs independently of the honest parties)
  - *True randomness* used (for randomized f's)

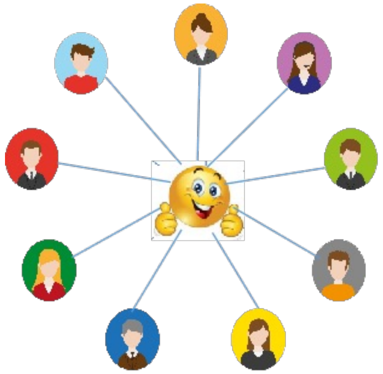


≈

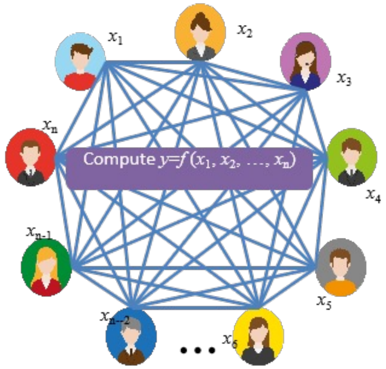


Defining security

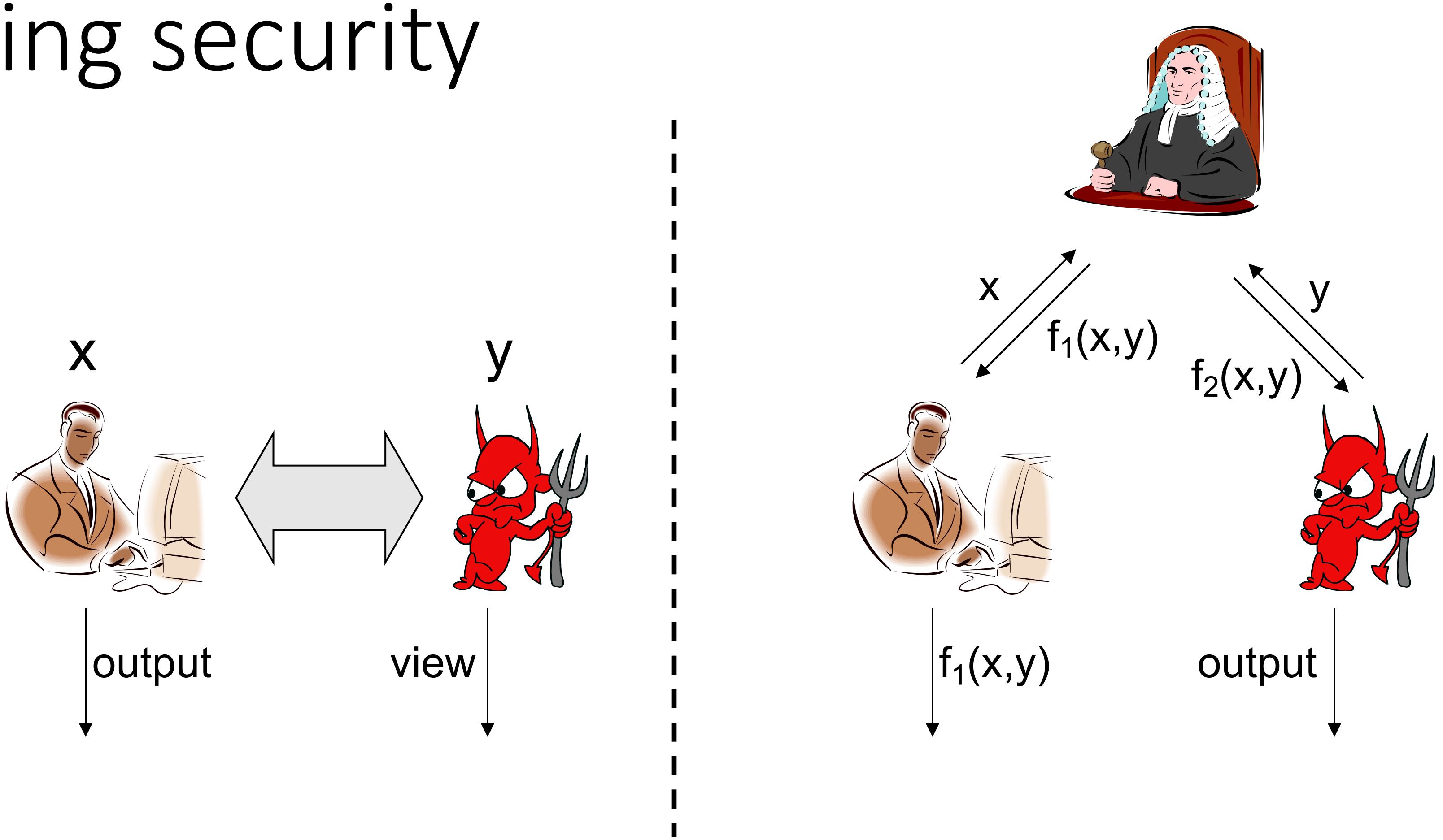
- “Real-world execution should be as secure as the ideal-world execution”
- “The only things the adversary can do in the real-world execution are things it can do in the ideal-world execution”
  - For every efficient attacker A in the real world, there is an “equivalent” (efficient) attacker B in the ideal world



$\approx$

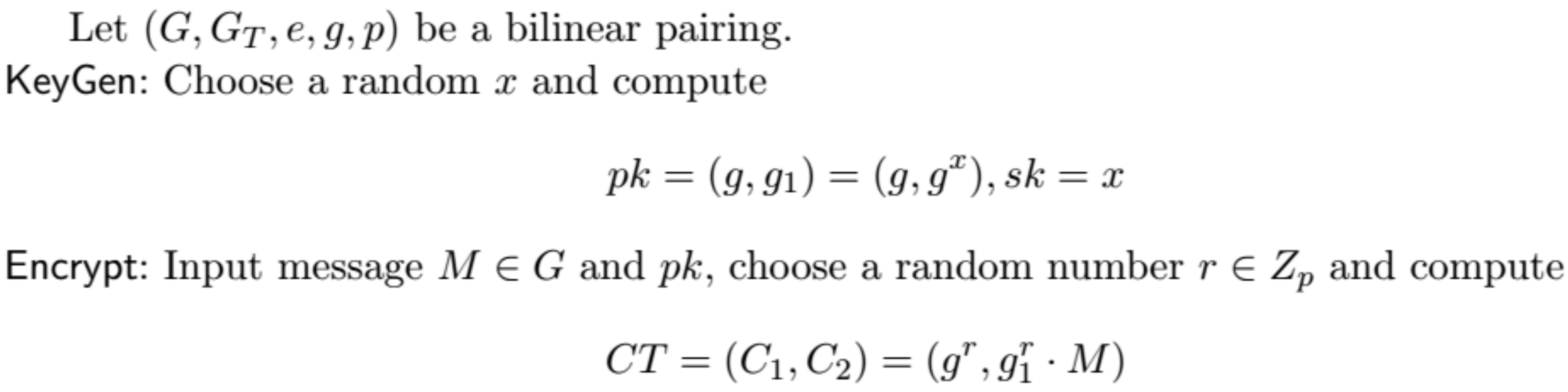


Defining security



A protocol is *secure* if for every (efficient) real-world adversary, there is an ideal-world adversary such that for all x, y the joint distributions of the above are computationally indistinguishable

ElGamal Encryption (Revisited)



Let (G, G_T, e, g, p) be a bilinear pairing.

KeyGen: Choose a random x and compute

$$pk = (g, g_1) = (g, g^x), sk = x$$

Encrypt: Input message $M \in G$ and pk, choose a random number $r \in Z_p$ and compute

$$CT = (C_1, C_2) = (g^r, g_1^r \cdot M)$$

Computationally indistinguishable:

Given (C_1, C_2), we don’t know

- It is computed from the ElGamal encryption using (pk,m), or
- Someone randomly chooses C_1 and C_2 from G.

(If the adversary can distinguish, then the ElGamal encryption is not IND-CPA secure)

ElGamal Encryption (Revisited)

Computationally indistinguishable:

Given (C_1, C_2), we don’t know

- It is computed from the ElGamal encryption using (pk,m), or
- Someone randomly chooses C_1 and C_2 from G.

(If the adversary can distinguish, then the ElGamal encryption is not IND-CPA secure)

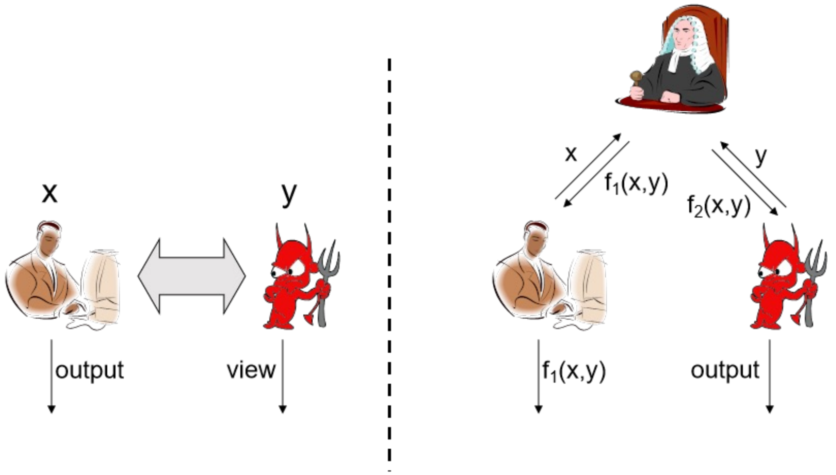
Proof:

We randomly choose two messages M_0 and M_1 for challenge.
Let (C_1, C_2) be the challenge ciphertext
1. It should be computed using (pk, M_0), or
2, computed using (pk, M_1), where (C_1, C_2) looks random when M_1 is randomly chosen.

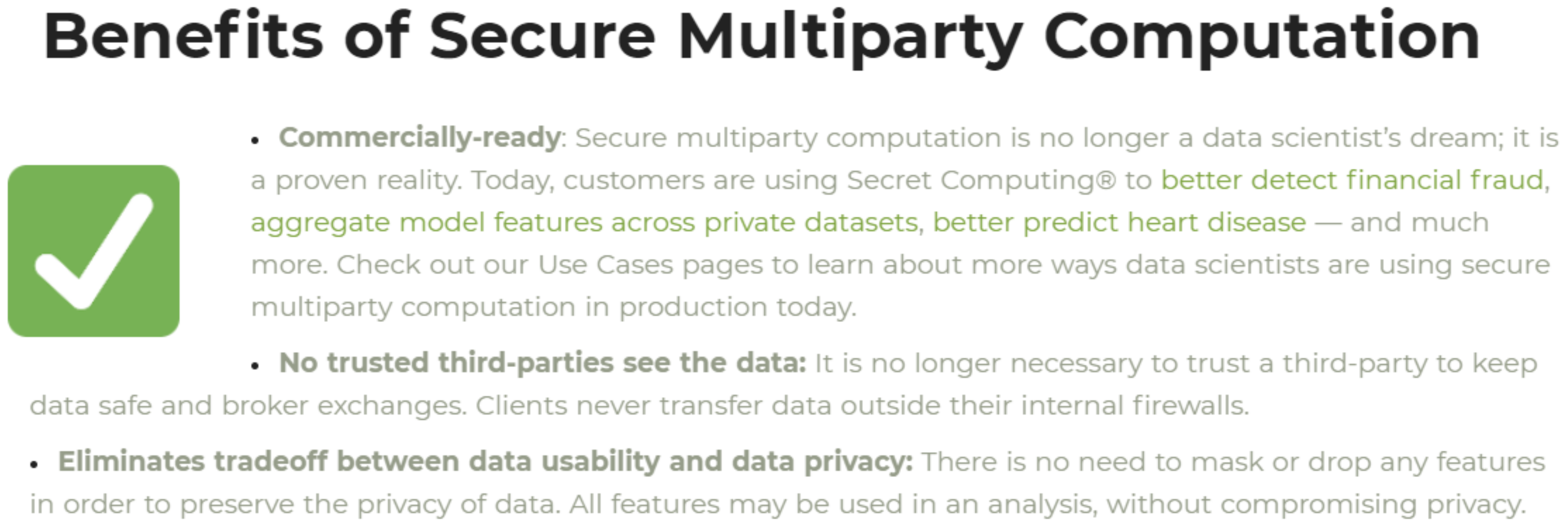
Simulation paradigm

- To prove security, we must take an arbitrary (efficient) real-world attacker A and construct an ideal-world attacker B for which the distributions are indistinguishable for all x, y

- B is called a *simulator* for A
  - B (running in the ideal world) will *simulate* the view of A in a real-world execution
  - View = randomness + messages



Secure Multiparty Computation

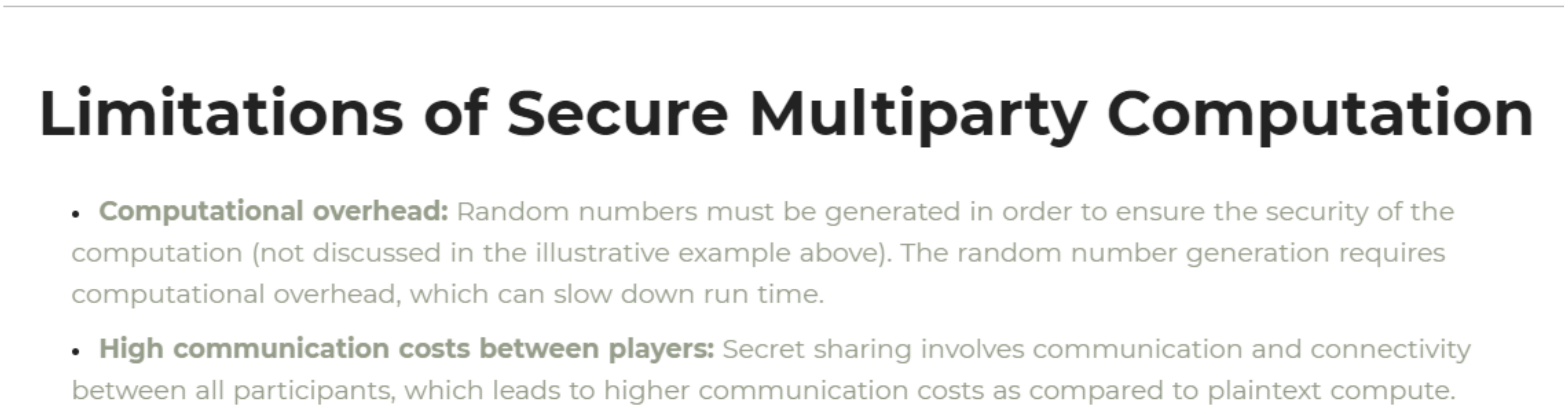


Benefits of Secure Multiparty Computation

- **Commercially-ready**: Secure multiparty computation is no longer a data scientist's dream; it is a proven reality. Today, customers are using Secret Computing® to better detect financial fraud, aggregate model features across private datasets, better predict heart disease — and much more. Check out our Use Cases pages to learn about more ways data scientists are using secure multiparty computation in production today.
- **No trusted third-parties see the data:** It is no longer necessary to trust a third-party to keep data safe and broker exchanges. Clients never transfer data outside their internal firewalls.
- **Eliminates tradeoff between data usability and data privacy:** There is no need to mask or drop any features in order to preserve the privacy of data. All features may be used in an analysis, without compromising privacy.

https://inpher.io/technology/what-is-secure-multiparty-computation/

Secure Multiparty Computation



Limitations of Secure Multiparty Computation

- **Computational overhead:** Random numbers must be generated in order to ensure the security of the computation (not discussed in the illustrative example above). The random number generation requires computational overhead, which can slow down run time.
- **High communication costs between players:** Secret sharing involves communication and connectivity between all participants, which leads to higher communication costs as compared to plaintext compute.

Example: Runing AES encryption with (K,m) input, the time cost is within millisecond. We can also let Alice input K and Bob imput m to compute a ciphertext for Bob using MPS, but it will take more than 5 mins.

https://inpher.io/technology/what-is-secure-multiparty-computation/

Roadmap

Secure Multiparty Computation

- Multi-Party Computation

↑

- Two-Party Computation F(x1,x2)=y_1 y_2 ..... y_n

↑

- Two-Party Computations with 0/1 as the computing result.

f(x1,x2)=y_i \in {0,1}

Secure Multiparty Computation

- Two-Party Computations with 0/1 as the computing result.

f(x1,x2)=y_i \in {0,1}

Suppose that Alice and Bob know y_i.

It partially leaks the information.

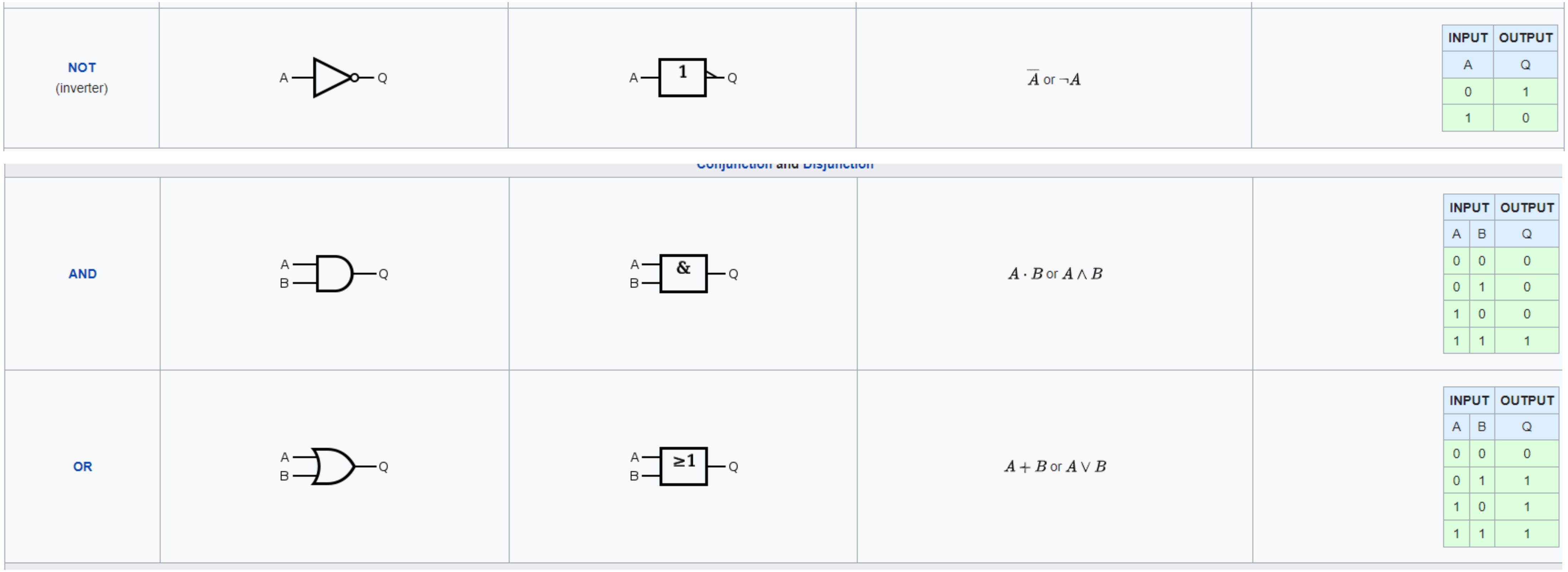
The range of x2.

If f(x1,x2’)=f(x1,x2), it should be hard for Alice to know whether Bob has x2 or x2’.

Construction

Logic Gates

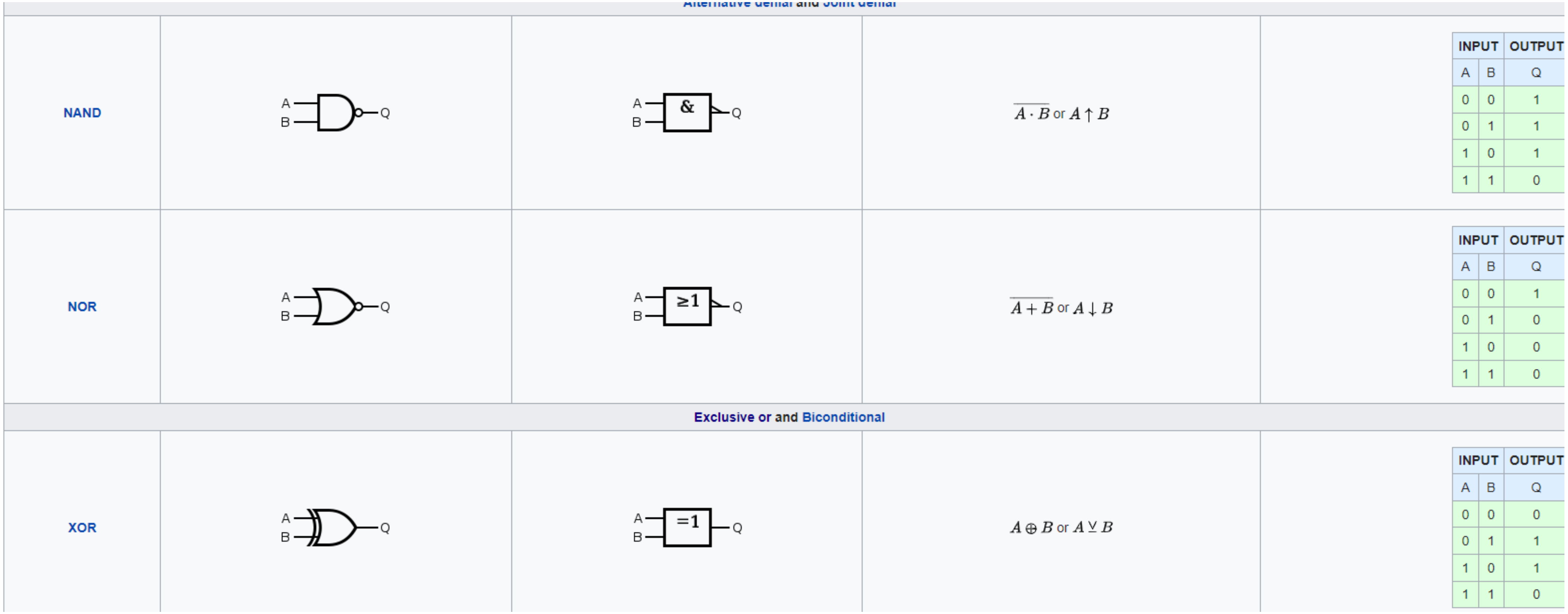
A logic gate is an idealized model of computation or physical electronic device implementing a Boolean function, a logical operation performed on one or more binary inputs that produces a single binary output.



<table>
<tr><td><b>NOT</b>
(inverter)</td><td>A Q</td><td>A 1 Q</td><td>$\overline{A}$ or $\neg A$</td><td></td><td><table><tr><th>INPUT</th><th>OUTPUT</th></tr><tr><td>A</td><td>Q</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table></td></tr>
<tr><td colspan="6">Conjunction and Disjunction</td></tr>
<tr><td><b>AND</b></td><td>A
B Q</td><td>A
B & Q</td><td>$A \cdot B$ or $A \wedge B$</td><td></td><td><table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><td>A</td><td>B</td><td>Q</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td></tr>
<tr><td><b>OR</b></td><td>A
B Q</td><td>A
B ≥1 Q</td><td>$A + B$ or $A \vee B$</td><td></td><td><table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><td>A</td><td>B</td><td>Q</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td></tr>
</table>

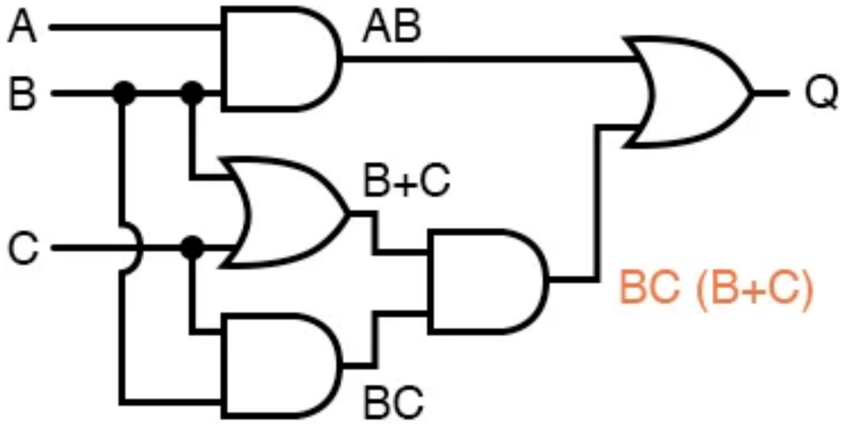
Logic Gates

A logic gate is an idealized model of computation or physical electronic device implementing a Boolean function, a logical operation performed on one or more binary inputs that produces a single binary output.



Alternative denial and Joint denial				
NAND		&	$\overline{A \cdot B}$ or $A \uparrow B$	INPUT OUTPUT A B Q 0 0 1 0 1 1 1 0 1 1 1 0
NOR		≥1	$\overline{A + B}$ or $A \downarrow B$	INPUT OUTPUT A B Q 0 0 1 0 1 0 1 0 0 1 1 0
Exclusive or and Biconditional				
XOR		=1	$A \oplus B$ or $A \veebar B$	INPUT OUTPUT A B Q 0 0 0 0 1 1 1 0 1 1 1 0

Boolean Circuit



f(x)=y \in {0,1}

Boolean circuit is a mathematical model for combinational digital logic circuits. Boolean circuits are defined in terms of the logic gates they contain. For example, a circuit might contain AND gate ,OR gate, NOT gate, XOR gate. Each gate input two bits and outputs a single bit.

A formal language can be decided by a family of Boolean circuits, one circuit for each possible input length. Boolean circuits are also used as a formal model for combinational logic in digital electronics.

Boolean Circuit

https://logic.ly/demo/

TPC to Boolean Circuit

f(x1,x2)=

- 0 if x_1 ≤ x2
- 1 if x_1> x2

f(x)=y \in {0,1}

- y=0 if the first half bits of x ≤ the second half bits of x
- y=1 if the first half bits of x > the second half bits of x

Secure Multiparty Computation

- Multi-Party Computation

↑

- Two-Party Computation F(x1,x2)=y_1 y_2 ..... y_n

↑

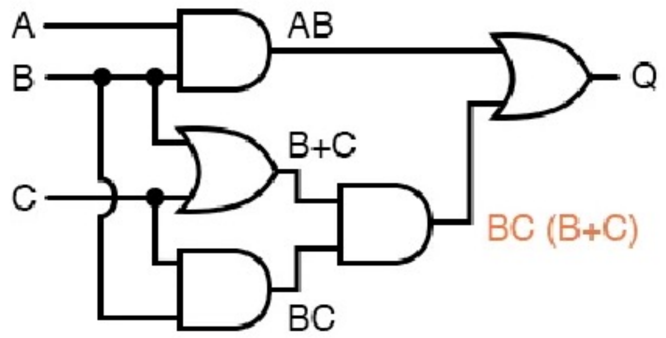
- Two-Party Computations with f(x1,x2)=y_i as the computing result.

↑

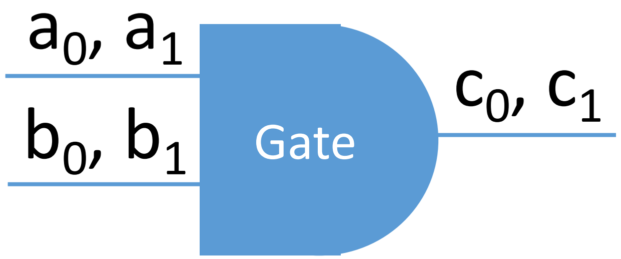
- Boolean function computation f(x)=y

↑

- Logic gate computation G(b1, b2)=b

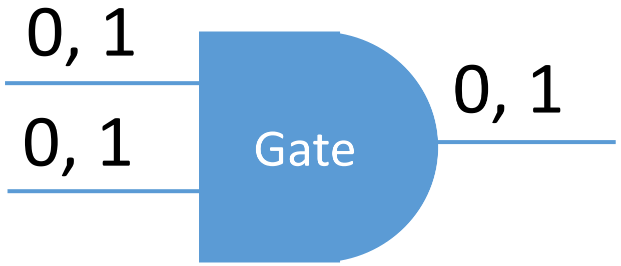


From general logic gate to garbled gate



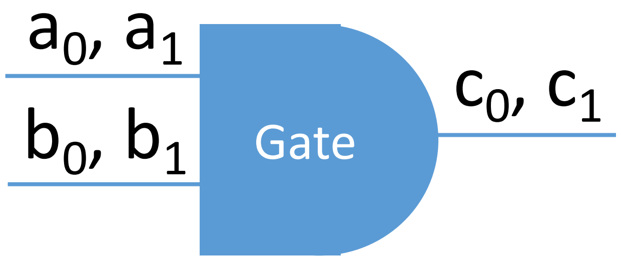
1st wire	2nd wire	Output wire
a_0	b_0	c_1
a_0	b_1	c_2
a_1	b_0	c_3
a_1	b_1	c_4

Gate
$Enc_{a1,\ b0}(c_3)$
$Enc_{a1,\ b1}(c_4)$
$Enc_{a0,\ b1}(c_2)$
$Enc_{a0,\ b0}(c_1)$



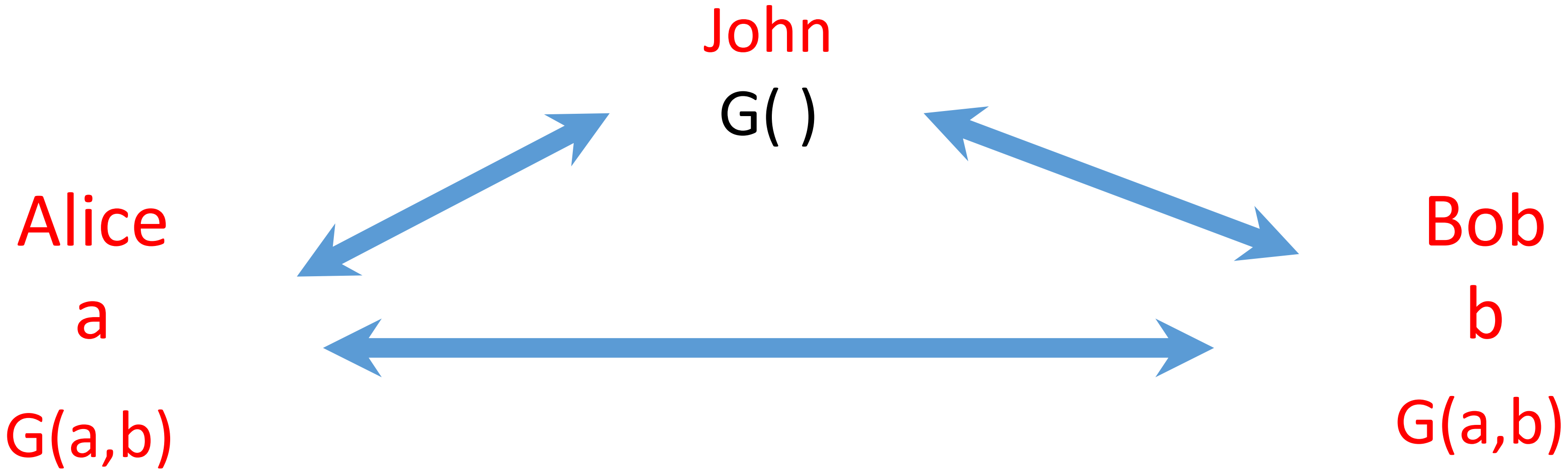
1st wire	2nd wire	Output wire
0	0	c_1
0	1	c_2
0	0	c_3
0	1	c_4

From general logic gate to garbled gate

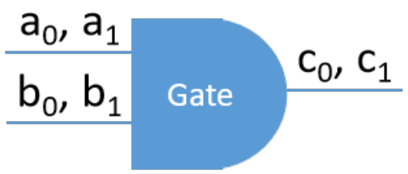


1st wire	2nd wire	Output wire
a_0	b_0	c_1
a_0	b_1	c_2
a_1	b_0	c_3
a_1	b_1	c_4

Gate
$Enc_{a1,\ b0}(c_3)$
$Enc_{a1,\ b1}(c_4)$
$Enc_{a0,\ b1}(c_2)$
$Enc_{a0,\ b0}(c_1)$

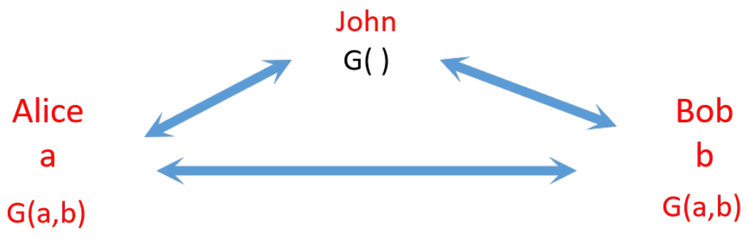


From general logic gate to garbled gate



1st wire	2nd wire	Output wire
a_0	b_0	c_1
a_0	b_1	c_2
a_1	b_0	c_3
a_1	b_1	c_4

Gate
$Enc_{a1,\ b0}(c_3)$
$Enc_{a1,\ b1}(c_4)$
$Enc_{a0,\ b1}(c_2)$
$Enc_{a0,\ b0}(c_1)$



- John publishes to Alice and Bob.

Gate
$Enc_{a1,\ b0}(c_3)$
$Enc_{a1,\ b1}(c_4)$
$Enc_{a0,\ b1}(c_2)$
$Enc_{a0,\ b0}(c_1)$

- John runs 1-out-of-2 OT protocol with Alice:

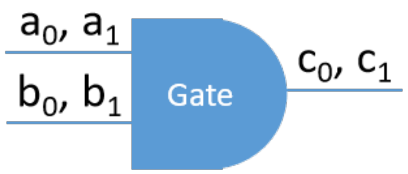
I have two keys (a_0, a_1), if you have a=0, pls run the protocol to get a_0. Otherwise, get a_1.

- John runs 1-out-of-2 OT protocol with Bob:

I have two keys (b_0, b_1), if you have b=0, pls run the protocol to get b_0. Otherwise, get b_1.

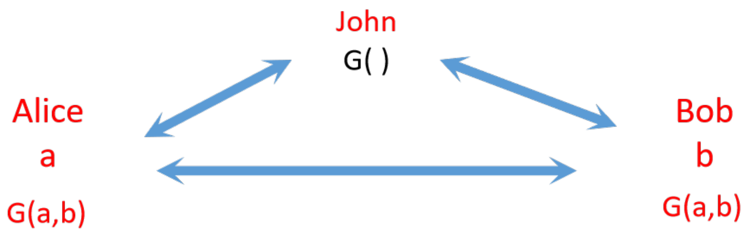
- Alice receives one key and Bob receives one key. They can reveal the keys and combine together to decrypt and get c

From general logic gate to garbled gate



1st wire	2nd wire	Output wire
a_0	b_0	c_1
a_0	b_1	c_2
a_1	b_0	c_3
a_1	b_1	c_4

Gate
$Enc_{a1,\, b0}(c_3)$
$Enc_{a1,\, b1}(c_4)$
$Enc_{a0,\, b1}(c_2)$
$Enc_{a0,\, b0}(c_1)$



- John knows nothing about Alice and Bob inputs.
- Alice cannot learn Bob’s bit b from b_i because (b_0,b_1) are randomly chosen.
- Bob cannot learn Alice’s bit a from a_i because (a_0,a_1) are randomly chosen.
- Alice and Bob cannot learn the gate G from c.

Note: Suppose they run and follow the protocol honestly. (semi-honest)
In the above example, we have to consider three parties. Otherwise, privacy will leaked.

Secure Multiparty Computation

- Multi-Party Computation

↑

- Two-Party Computation F(x1,x2)=y_1 y_2 ..... y_n

↑

- Two-Party Computations with f(x1,x2)=y_i as the computing result.

↑

- Boolean function computation f(x)=y

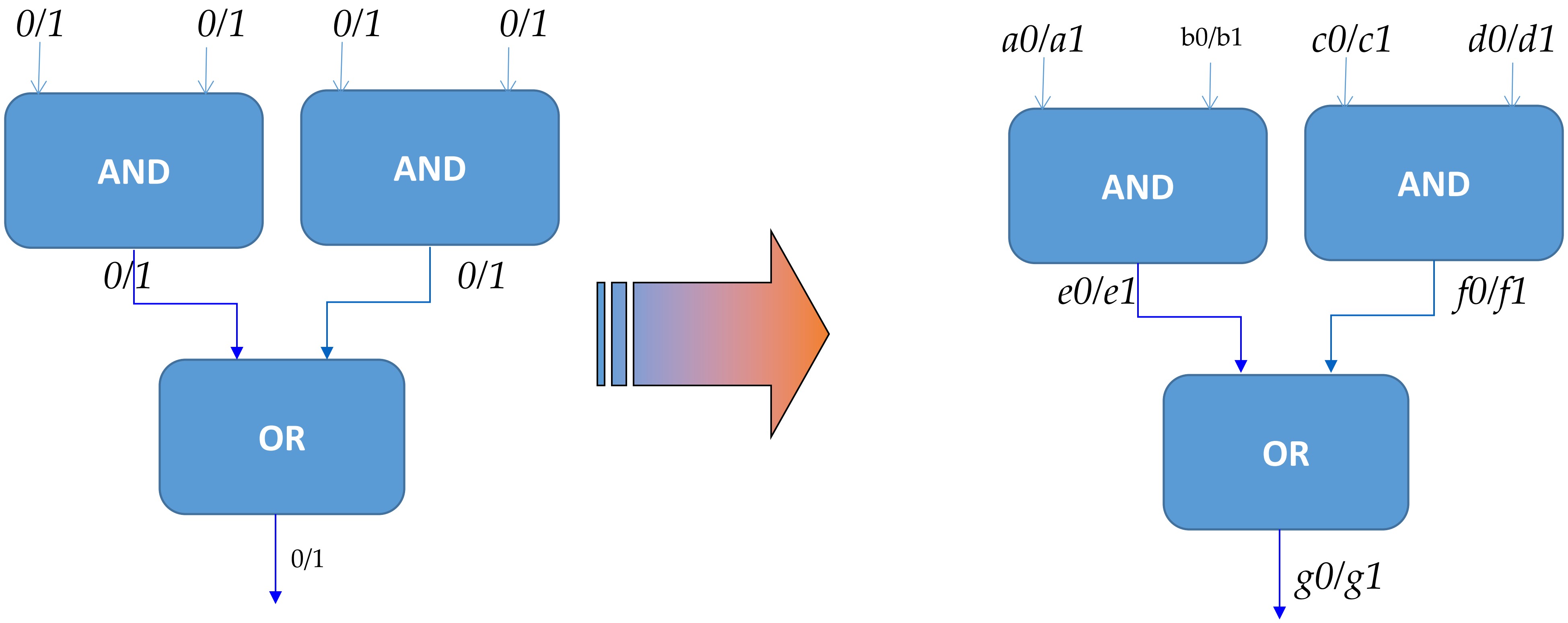
↑

- Logic gate computation G(b1, b2)=b

↑

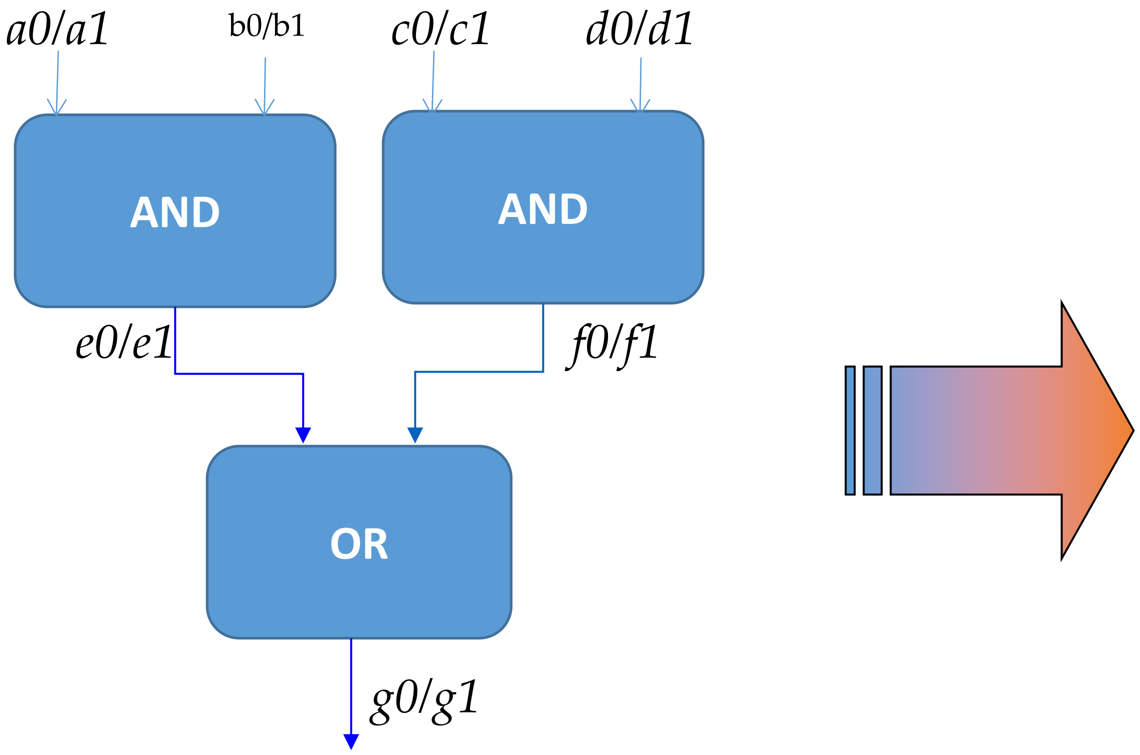
- Garbled gate computation G(b1, b2)=b (with privacy protection)

From boolean circuit to garbled circuit



Each wire will carry a bit value 0/1.
We map each bit value in wire to a random integer/string

From boolean circuit to garbled circuit

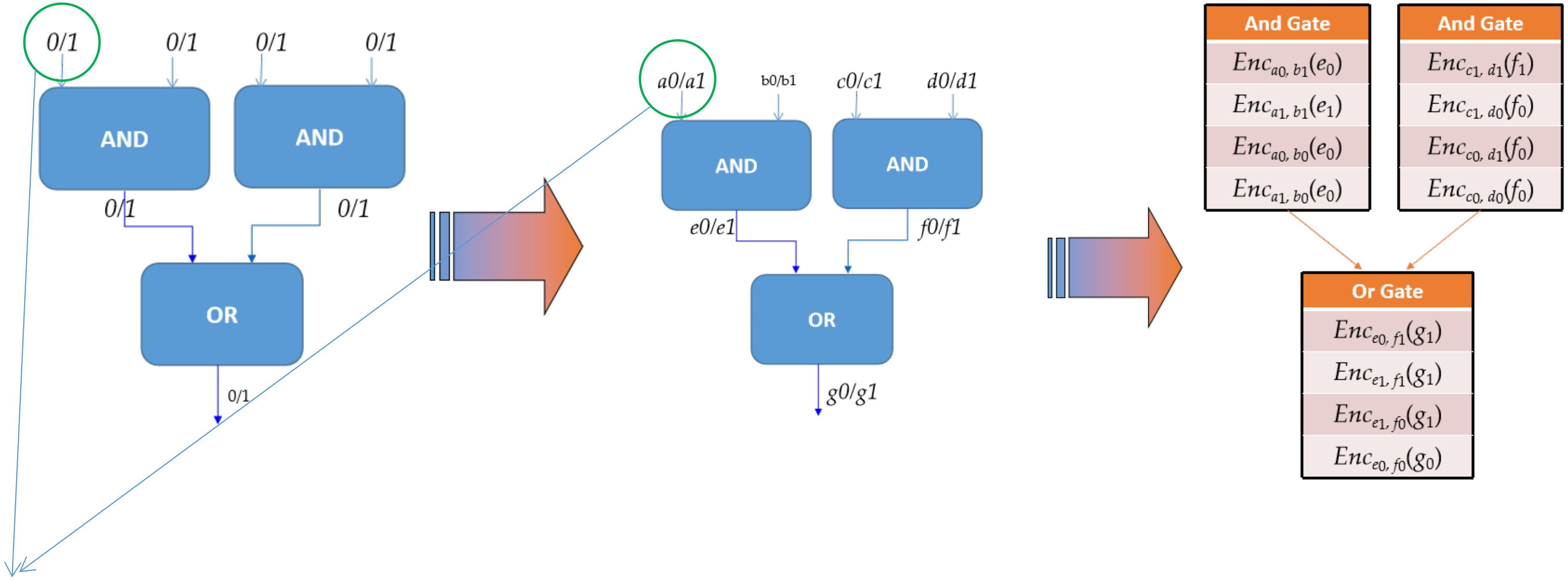


And Gate
$Enc_{a0,\ b1}(e_0)$
$Enc_{a1,\ b1}(e_1)$
$Enc_{a0,\ b0}(e_0)$
$Enc_{a1,\ b0}(e_0)$

And Gate
$Enc_{c1,\ d1}(f_1)$
$Enc_{c1,\ d0}(f_0)$
$Enc_{c0,\ d1}(f_0)$
$Enc_{c0,\ d0}(f_0)$

Or Gate
$Enc_{e0,\ f1}(g_1)$
$Enc_{e1,\ f1}(g_1)$
$Enc_{e1,\ f0}(g_1)$
$Enc_{e0,\ f0}(g_0)$

From boolean circuit to garbled circuit



And Gate
$Enc_{a0,\,b1}(e_0)$
$Enc_{a1,\,b1}(e_1)$
$Enc_{a0,\,b0}(e_0)$
$Enc_{a1,\,b0}(e_0)$

And Gate
$Enc_{c1,\,d1}(f_1)$
$Enc_{c1,\,d0}(f_0)$
$Enc_{c0,\,d1}(f_0)$
$Enc_{c0,\,d0}(f_0)$

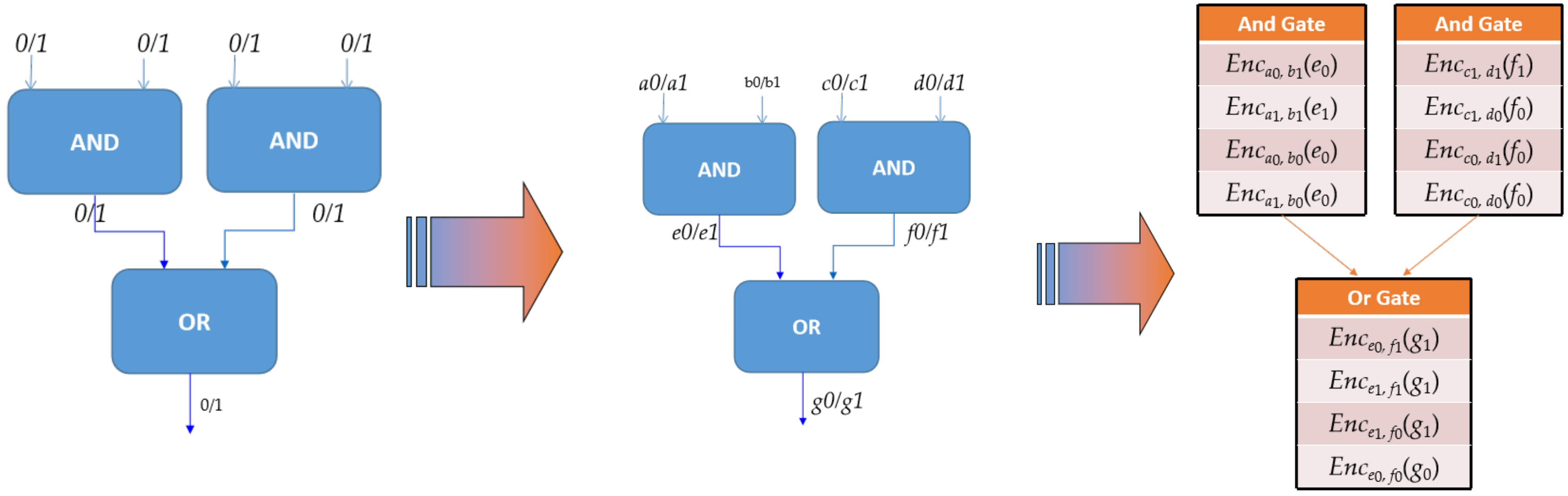
Or Gate
$Enc_{e0,\,f1}(g_1)$
$Enc_{e1,\,f1}(g_1)$
$Enc_{e1,\,f0}(g_1)$
$Enc_{e0,\,f0}(g_0)$

g_0=0
g_1=1

Suppose the garbled circuit is created by Alice.
Each input value must be either from Alice or Bob.

- If it belongs to Alice, Alice directly reveals a0 if the bit value is 0 and a1 if the bit value is 1.
- If it belongs to Bob, Bob runs the 1-out-of-2 OT protocol with Alice to get a0 or a1.
- After all input integer values are received, Bob can run the protocol to evalute the circuit to get g_0 or g_1.

From boolean circuit to garbled circuit



And Gate
$Enc_{a0,\, b1}(e_0)$
$Enc_{a1,\, b1}(e_1)$
$Enc_{a0,\, b0}(e_0)$
$Enc_{a1,\, b0}(e_0)$

And Gate
$Enc_{c1,\, d1}(f_1)$
$Enc_{c1,\, d0}(f_0)$
$Enc_{c0,\, d1}(f_0)$
$Enc_{c0,\, d0}(f_0)$

Or Gate
$Enc_{e0,\, f1}(g_1)$
$Enc_{e1,\, f1}(g_1)$
$Enc_{e1,\, f0}(g_1)$
$Enc_{e0,\, f0}(g_0)$

g_0=0
g_1=1

Suppose a0/a1 and c0/c1 should be revealed by Alice. Suppose Alice’s input=(0,0)
Suppose b0/b1 and d0/d1 should be revealed by Bob. Suppose Bob’s input =(0,1)

- Alice sends ALL garbled gates + (g_0,g_1)=(0,1) + a0 + c0 to Bob
- Bob runs the 1-out-of-2 OT protocol with Alice to receive b0 and d1.
- Bob can use (a0, b0) to get e0 and use (c0,d1) to get f0.
- Bob uses e0 and f0 to get g_0 from the third garbled gate.
- Bob returns the result 0.

Secure Multiparty Computation

- Multi-Party Computation

↑

- Two-Party Computation F(x1,x2)=y_1 y_2 ..... y_n

↑

- Two-Party Computations with f(x1,x2)=y_i as the computing result.

↑

- Boolean function computation f(x)=y

↑

- Boolean circuit f(x)=y

↑

- Garbled circuit (with privacy protection)

Workshop

A secure commitment scheme

Let (g,h) be two group elements chosen by the receiver.

Commit(m): Taking as input the value m in Z_p to be committed, choose a random integer r from Z_p and compute

$$C=g^{m}h^{r}\text{ , secret}=r$$

Open(C,m’,r’): Taking as input (C,m’,r’), accept if

$$C=g^{m'}h^{r'}$$

This commitment scheme is secure such that the receiver cannot know what m is.

How can Alice prove to Bob that she knows (m,r) such that C=g^m*h^r ?

(Zero Knowledge Proof)

Identify flaws in this protocol (Q1)

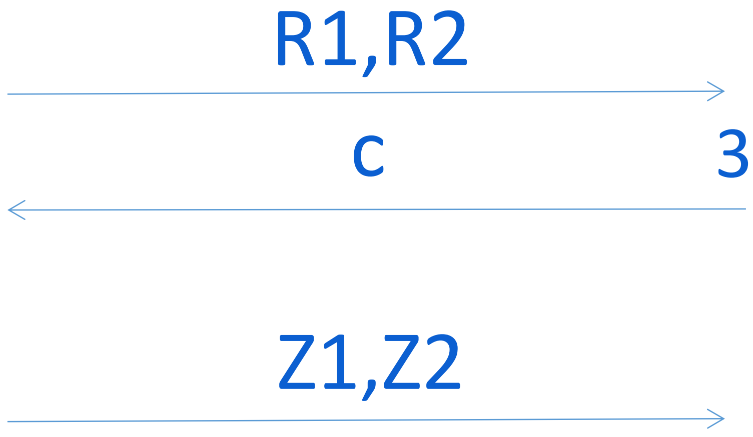
Prover

(C,m,r)

Verifier

(C)

1. Choose random r_1, r_2
2. R1=g^{r_1}
   R2=h^{r_2}



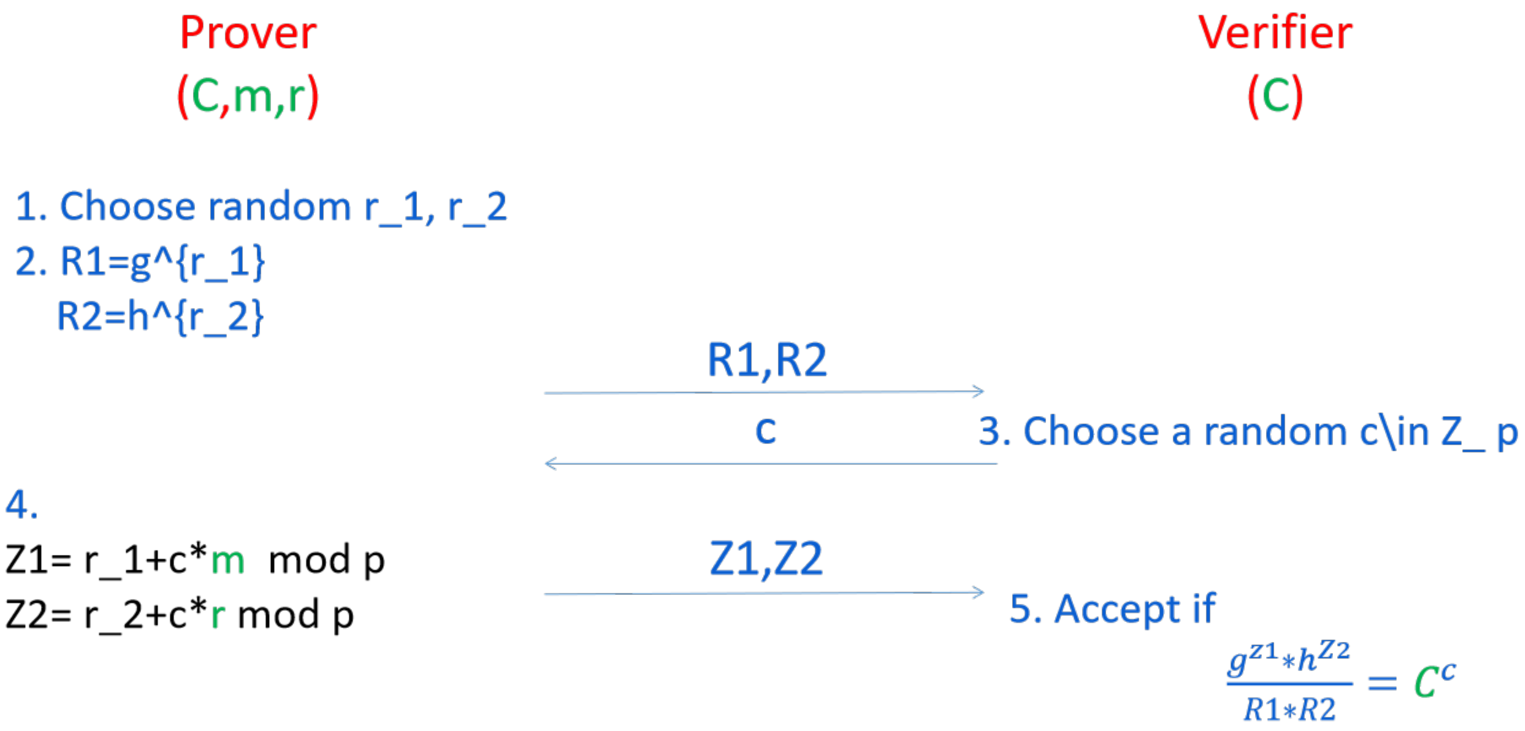
3. Choose a random c\in Z_ p

4.
Z1= r_1+c*m mod p
Z2= r_2+c*r mod p

5. Accept if

$$\frac{g^{z1}*h^{Z2}}{R1*2} = C^c$$

Identify flaws in this protocol



Prover

(C,m,r)

Verifier

(C)

1. Choose random r_1, r_2
2. R1=g^{r_1}

   R2=h^{r_2}

R1,R2

c

3. Choose a random c\in Z_ p

4.

Z1= r_1+c*m mod p

Z2= r_2+c*r mod p

Z1,Z2

5. Accept if

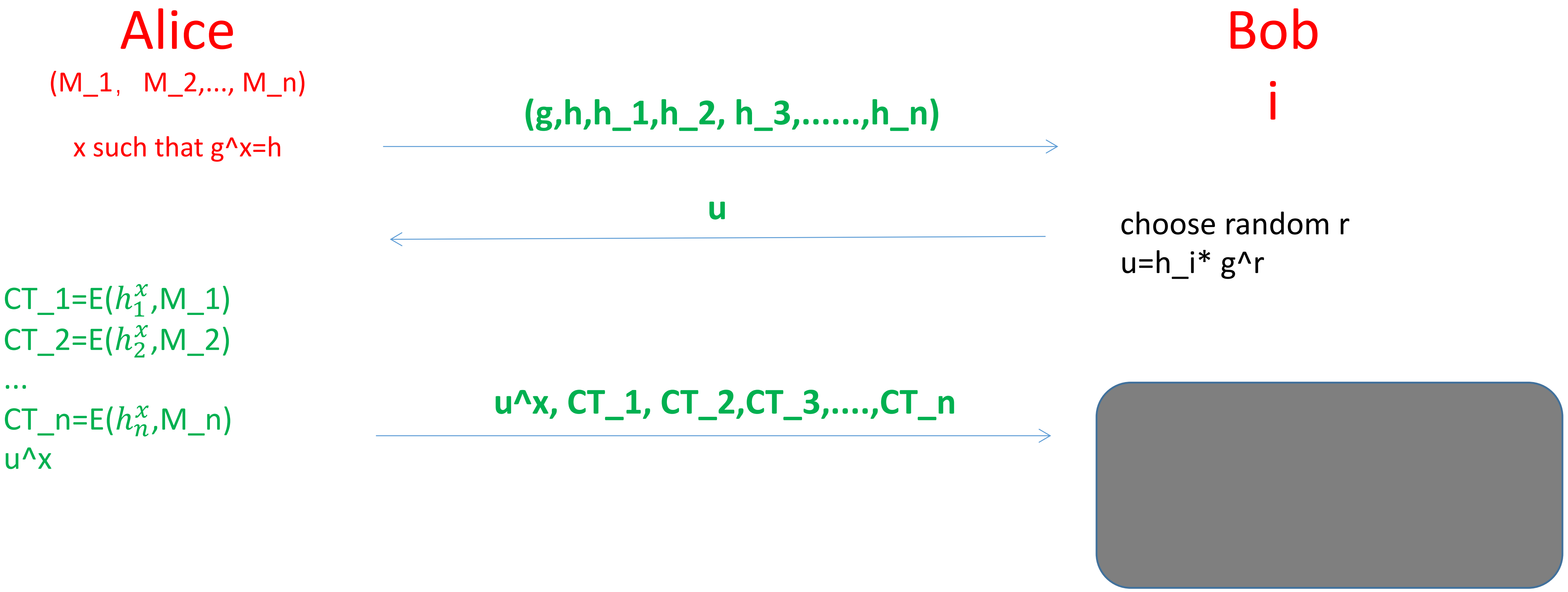
$$\frac{g^{Z1} * h^{Z2}}{R1 * R2} = C^c$$

The verifier can compute

$$\frac{g^{Z1}}{R1} = g^{mc}$$

As the verifier knows c, he/she can then compute g^m. Once m is in a small range, the message m will be known. So, it is not a zero-knowedge proof.

Clarify the purpose of protocol (Q2)



Alice

(M_1, M_2,..., M_n)

x such that g^x=h

(g,h,h_1,h_2, h_3,......,h_n)

Bob

i

u

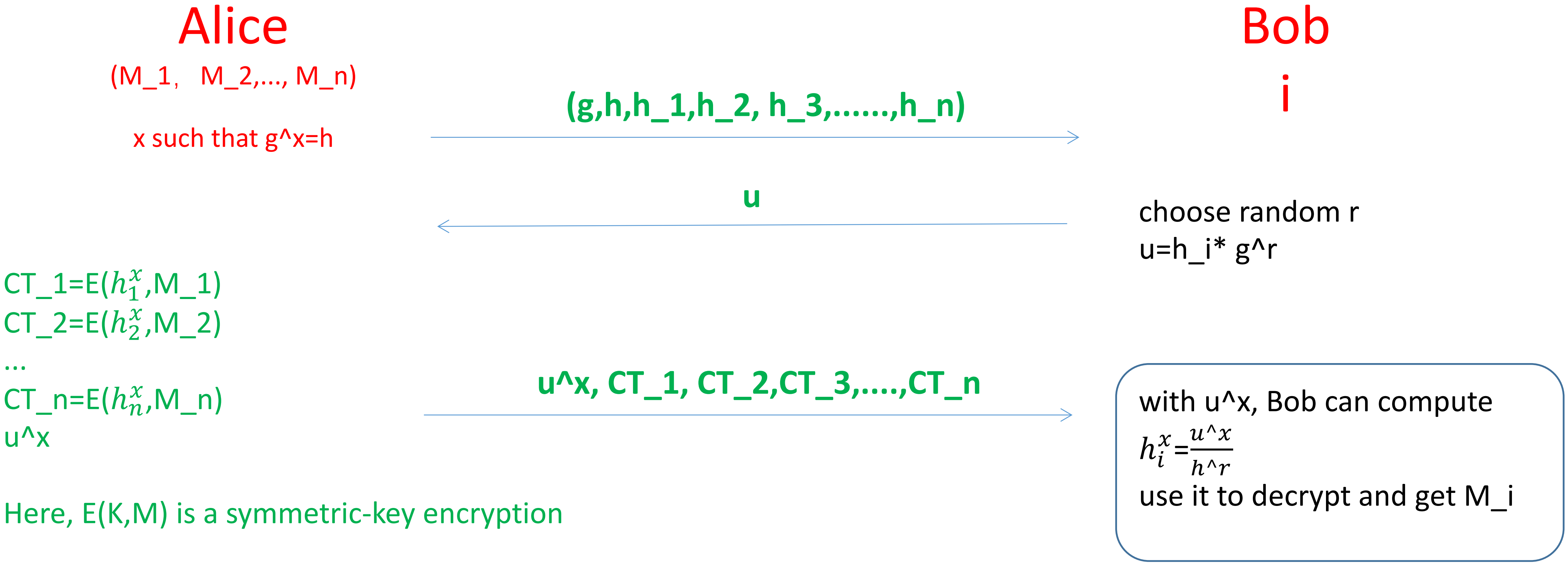
choose random r
u=h_i* g^r

CT_1=E(h_1^x,M_1)
CT_2=E(h_2^x,M_2)
...
CT_n=E(h_n^x,M_n)
u^x

u^x, CT_1, CT_2,CT_3,....,CT_n

What should Bob do next? (Q3)

Clarify the purpose of protocol (Q2)



Alice

(M_1， M_2,…, M_n)

x such that g^x=h

Bob

i

(g,h,h_1,h_2, h_3,……,h_n)

u

choose random r
u=h_i* g^r

CT_1=E(h_1^x,M_1)
CT_2=E(h_2^x,M_2)
…
CT_n=E(h_n^x,M_n)
u^x

u^x, CT_1, CT_2,CT_3,….,CT_n

with u^x, Bob can compute
$h_i^x = \frac{u\hat{}x}{h\hat{}r}$
use it to decrypt and get M_i

Here, E(K,M) is a symmetric-key encryption

This is a 1-out-of-n OT protocol. We can easily extend it to k-out-of-n OT protocol.