Yao's Millionaires' Problem

Protocols for Secure Computations (Extended Abstract). FOCS 1982: 160-164

Turing award winner Andrew Yao

Yao's millionaires' problem

₹ X

?

<

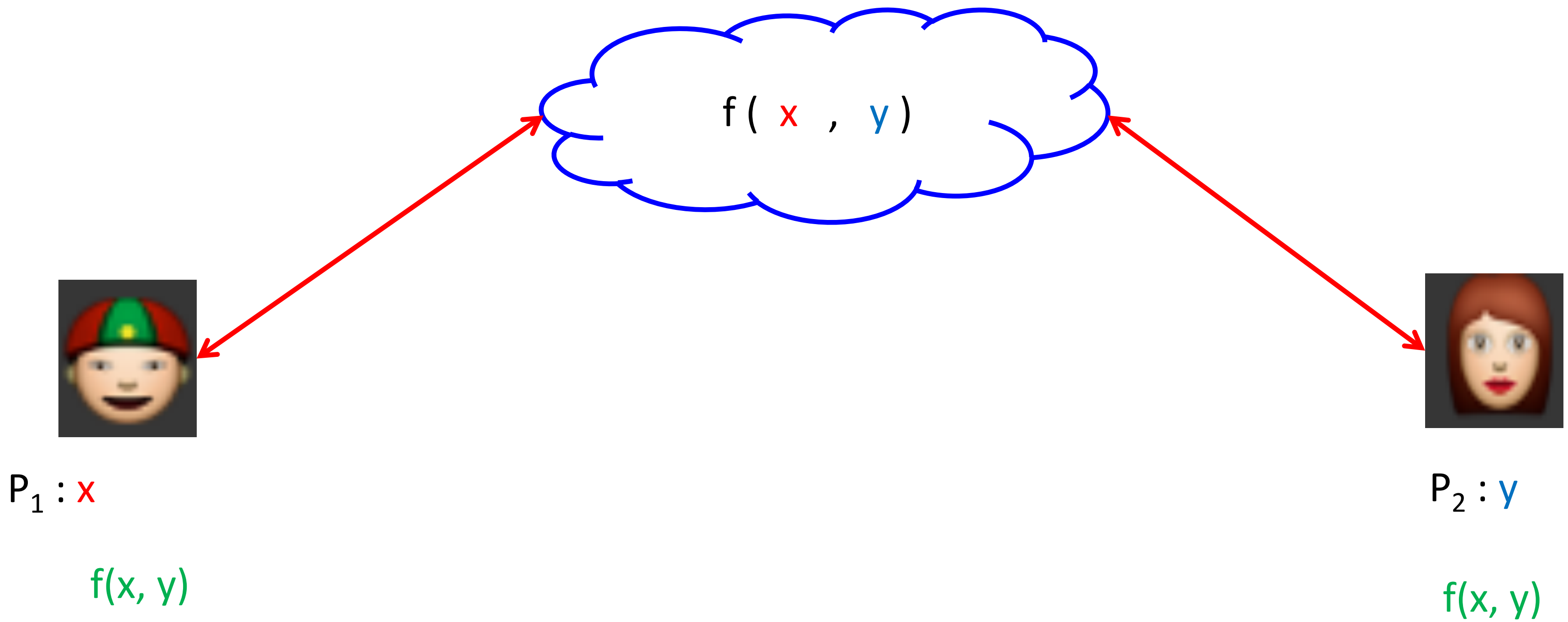
=

>

₹ Y

Find the richer without disclosing exact value of individual assets

Secure 2-PC

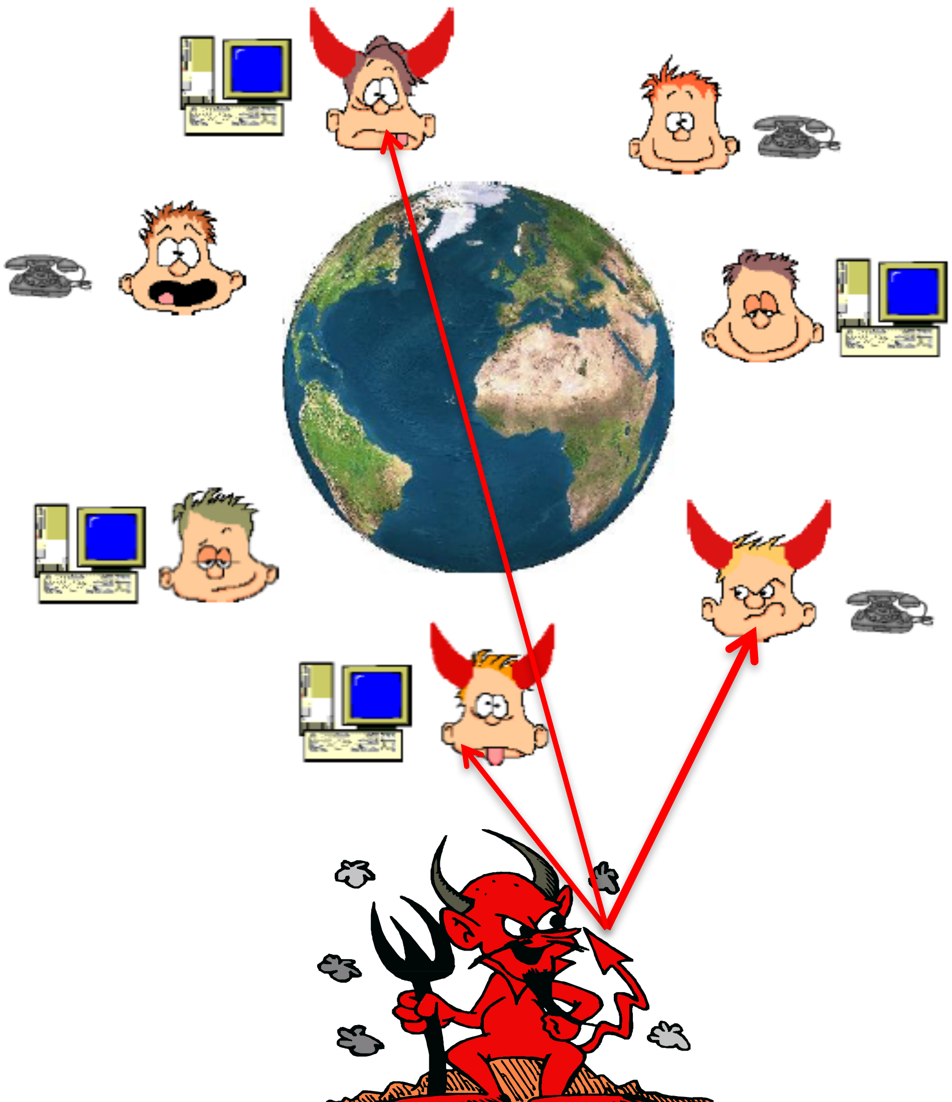


- Mutually distrusting entities with individual private data

- Want to compute a joint function of their inputs without revealing anything beyond

Secure Multiparty Computation (MPC)

MPC – holy grail



Setup:

- n parties $P_1,....,P_n$; 'some' are corrupted
- P_i has private input x_i
- A common n-input function f

Goals:

- **Correctness:** Compute $f(x_1,x_2,..x_n)$
- **Privacy:** Nothing beyond function output must be leaked

Applications: (Dual need of data privacy & data usability)

Preventing Satellite Collision

E-auction

Data Analytics

Privacy-preserving ML

Outsourcing

E-voting

Application of 2PC- Privacy-preserving Data mining

- How many patients suffering from AIDS in total ?
- Are there any common patient registered for disease X in all the hospitals ?
- Varieties of other statistics …

AIIMS
(All India Institute of Medical Sciences)

Apollo
HOSPITALS

How to solve 2PC?

- Trusted third party (TTP) $\rightarrow$ solution for secure 2PC
  - Send input to TTP, obtain function output : Ideal solution

x

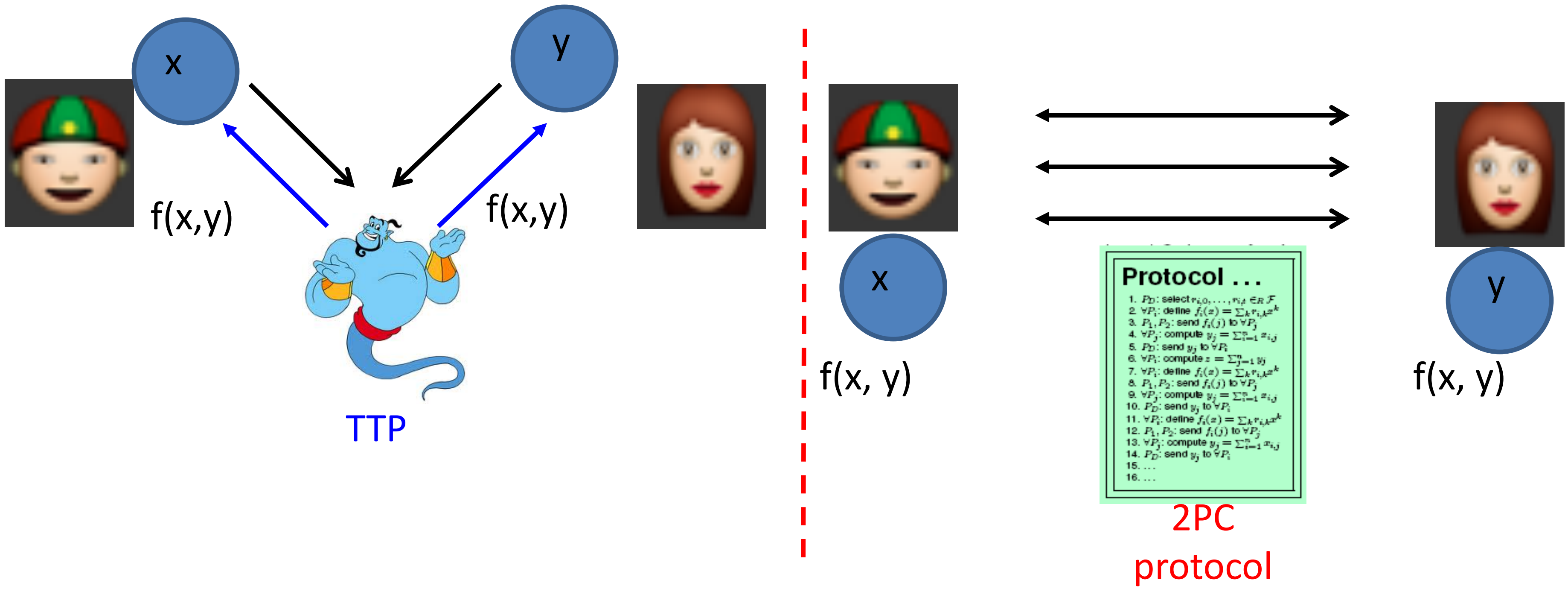
y

IDEAL world secure 2PC protocol

TTPs exist only in fairy tales!!

Security goal of 2PC

- Goal of a secure 2PC protocol : emulate the role of a TTP
  - ➢ De-centralizing the trust



IDEAL world $\approx$ REAL world

i : Alice wealth $\quad$ j : Bob wealth $\quad$ $1 < i, j < 10 \rightarrow N$ bit integer

M: set of all Non-Neg integer

Q_n : set of 1-1 of function $\rightarrow$ M to M

E_a: Alice public key, randomly chosen from Q_n

D_a: Alice private key

x : random N bit integer selected by Bob

$k = E_a(x)$ created by Bob

Bob sends $k - j + 1$ to Alice

Alice computes: $Y_u = D_a(k - j + u)$ for $u = 1, 2, \ldots, 10$

Then: $Z_u = Y_u \bmod P$, where P is prime number and N/2 bits

Alice send the set Z_u and P to Bob

Bob selects the j^{th} number in the set

If it is equal to $x \bmod p \rightarrow i >= j$ else $i < j$

Garbled circuit

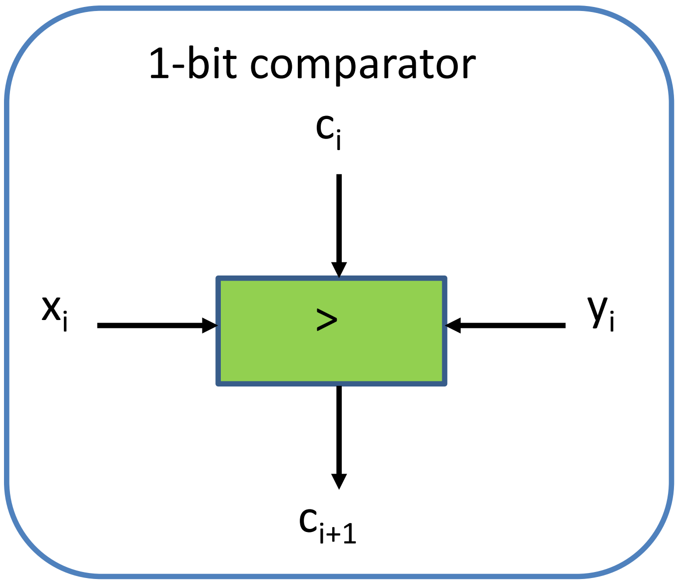
- Creating the circuit
- Encrypting and garbling the circuit
- Evaluation of circuit and Oblivious Transfer

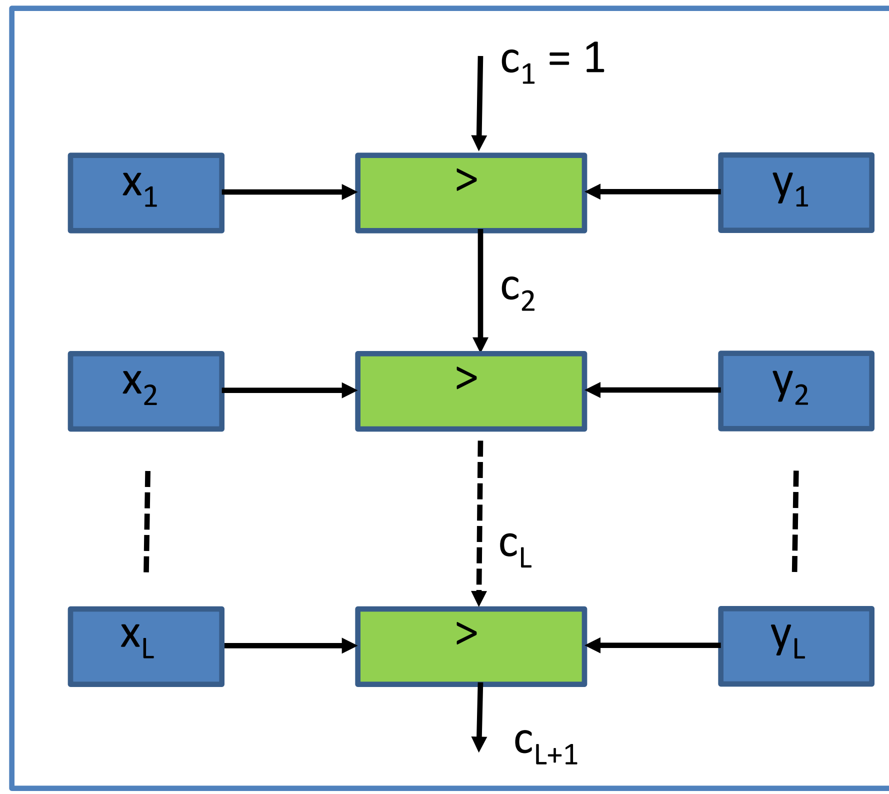
Circuit Representation of function

- Circuit abstraction
  - f : represented as a Boolean circuit C
  - Any efficiently computable f can be represented as a C
  - C: DAG with input gates, output gates and internal Boolean gates ((AND, OR, NOT), (NAND), (NOR): universal gates)

Circuit Abstraction Example: $\geq$

- X, Y: L-bit non-negative integers





- $c_{i+1} = 1 \leftrightarrow (x_i > y_i)$ OR
  $([x_i = y_i]$ AND $[c_i = 1])$

- $c_{i+1} = x_i \oplus [(x_i \oplus c_i) \wedge (y_i \oplus c_i)]$

- $X \geq Y \leftrightarrow c_{L+1} = 1$

Circuit Garbling

What we do?

- Encode/Garble the circuit
- Encode input
- Evaluate encoded circuit on encoded input and get encoded output
- Decode output using decoding information

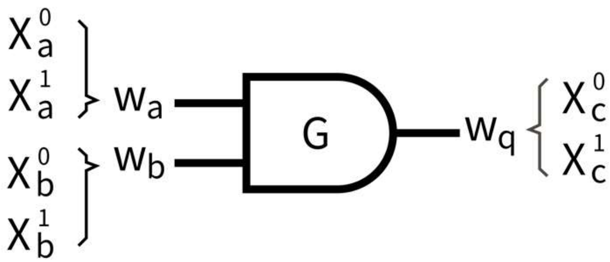
What is the goal?

- Nothing beyond function output is leaked
  - ✓ Preserves input privacy
  - ✓ No leaking of intermediate gate outputs
  - ✓ No leaking of output if decoding info is withheld

Yao: secure circuit evaluation

- Parties jointly evaluate the circuit securely
- Only final outcome revealed during evaluation
- Intermediate values remain private

Alice assigns label to the wires and replace them in truth table



b \ a	**0**	**1**
0	0	0
1	0	1

→

b \ a	X_a^0	X_a^1
X_b^0	X_c^0	X_c^0
X_b^1	X_c^0	X_c^1

→

b \ a	X_a^0	X_a^1
X_b^0	$E_{X_a^0, X_b^0}(X_c^0)$	$E_{X_a^1, X_b^0}(X_c^0)$
X_b^1	$E_{X_a^0, X_b^1}(X_c^0)$	$E_{X_a^1, X_b^1}(X_c^1)$

Alice Encrypts the output with corresponding input label

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

a	b	c
X_0^a	X_0^b	X_0^c
X_0^a	X_1^b	X_0^c
X_1^a	X_0^b	X_0^c
X_1^a	X_1^b	X_1^c

Garbled Table
$Enc_{X_0^a, X_0^b}(X_0^c)$
$Enc_{X_0^a, X_1^b}(X_0^c)$
$Enc_{X_1^a, X_0^b}(X_0^c)$
$Enc_{X_1^a, X_1^b}(X_1^c)$

Alice randomly permutes the table such that the output value cannot be determined from the row

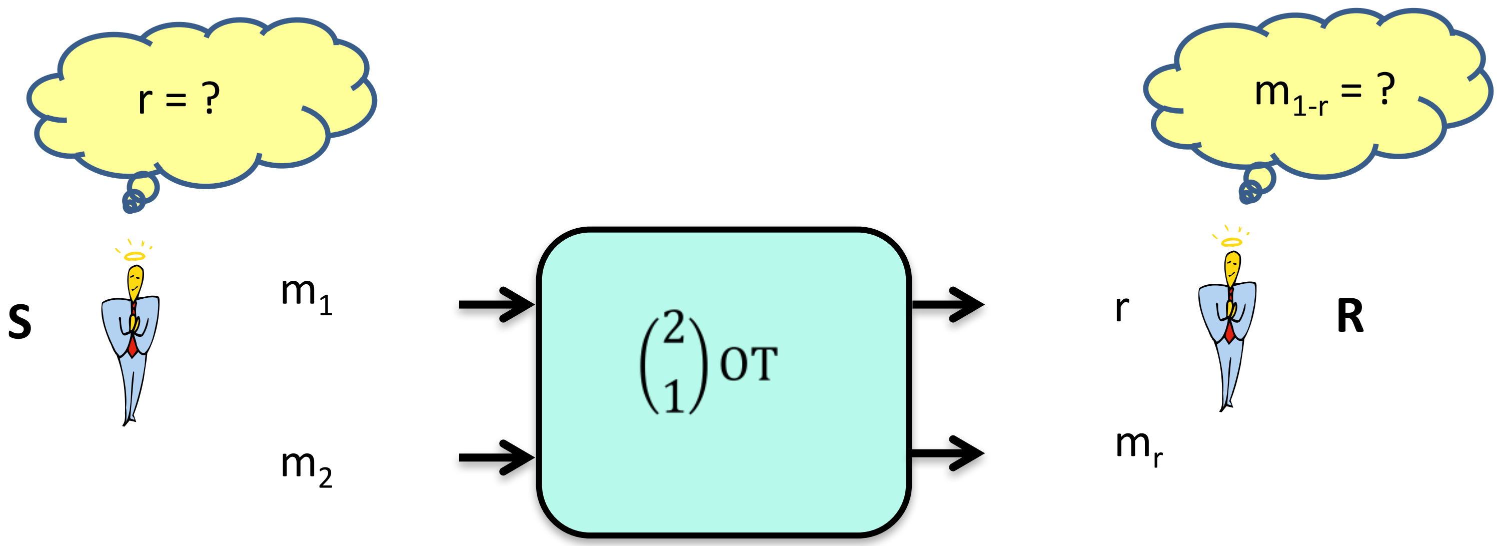
Sharing

- Alice sends computed garbled table for all gates
- Alice chooses label for her input
  - If $a = a_2a_1a_0 = 011$ Then she sends X_0^{a2}, X_1^{a1}, X_1^{a0}
- Label are random strings → Bob gets no information
- Bob uses **Oblivious Transfer** to receive label for his input bits
- Alice gets no information about Bob choice in OT

Evaluation

- Bob has the garbled table
- Bob has the input lables
- Goes through all the gates
  - Decrypt the rows in garbled table
  - Able to open one row in each table
  - Bob gets the output label X^c
- Alice knows the mapping of X^c to Boolean value
- One of them shares the information to the other one
- One or both of them get the result

Oblivious Transfer



Oblivious Transfer (1 out of 2)

- Sender transfer one piece of info to the receiver
- Sender does not know what piece is transferred

- Alice has m0 and m1
- Bob chooses b{0,1}
- Bob gets : m0*(1-b) + m1*b
  - b = 0 gets m0
  - b = 1 gets m1

OT(1 out of 2) by Goldreich

Alice

- M0 and m1
- RSA keys(n,e,d)
- X1 and x0 : two random msgs

- $K_0 = (v - x_0)^d \bmod N$
- $K_1 = (v - x_1)^d \bmod N$
- Sends: $m'_0 = m_0 + k_0$ and $m'_1 = m_1 + k_1$

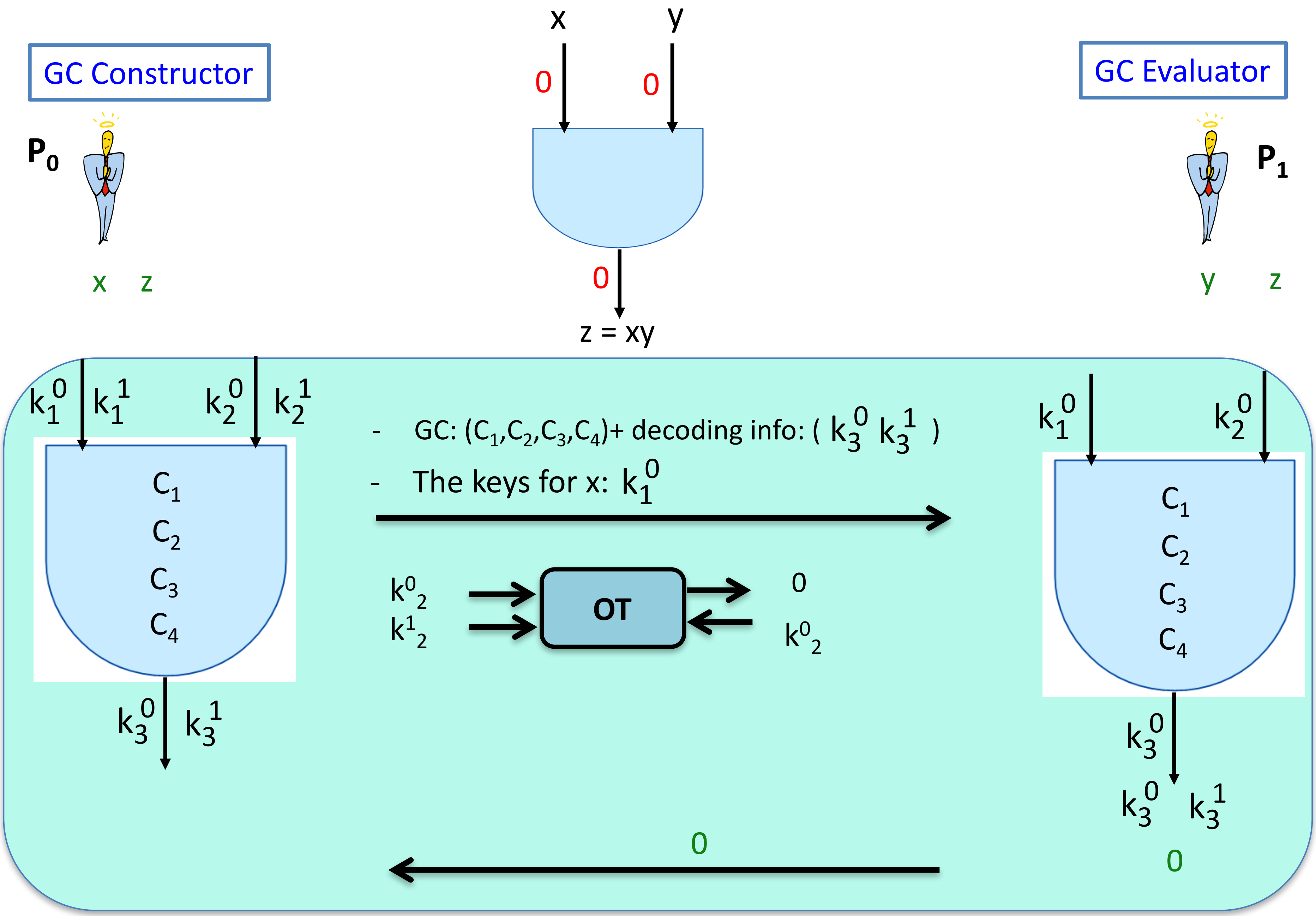
Bob

- Receives public key and random msgs
- Chose $b \in \{0,1\}$ and generate random k
- $V = (x_b + k^e) \bmod N$, send it

- Bob computes:

$m_b = m'_b + k_b$

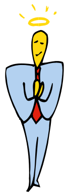
Yao's 2-Party Protocol



Yao's 2-Party Protocol

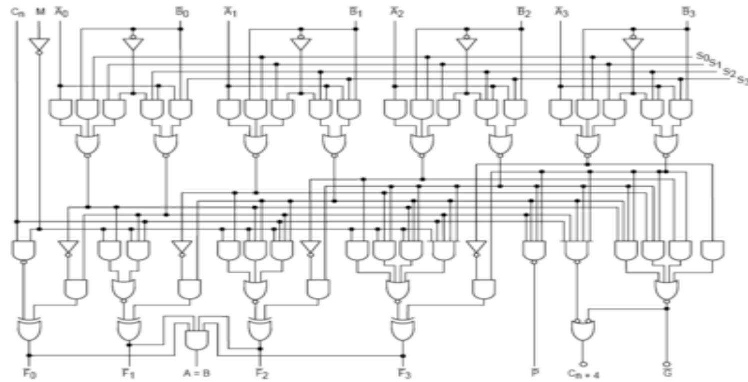
GC Constructor

P_0

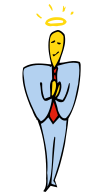


$X = (x_1, x_2, \dots x_k)$

Z



GC Evaluator



P_1

$Y = (y_1, y_2, \dots y_k)$

Z

- Garbled Circuit + decoding information
- The keys for X

k^0_1, k^1_1 → **OT_1** → y_1, k^{y1}_1

k^0_k, k^1_k → **OT_k** → y_k, k^{yk}_k

Z

Optimization

- Optimizing Boolean circuits
  - TinyGarble paper
  - Reduce number of non-XOR gates
  - Unroll the loops in runtime
  - Compactness → less memory footprints
- Combine with secret sharing sharing
  - ABY3 paper
  - Use arithmetic, yao and binary secret sharing
  - Use the most efficient one in each section
  - Convert the presentations when required

Properties

- Constant round MPC
- Level of privacy(the threshold)
- HbC and strong adversary(ZKNP)

- In theory, every function can be presented by Boolean circuit and we can apply GC, but in practice, it’s a long way …

Secret sharing

- Trivial secret sharing → XOR or Additive
- Shamir secret sharing → with threshold T
- Homomorphic secret sharing
- Many other sharing schemes

Name	#Party	scheme	threat	Operations
ABY	2PC	SS & GC	Semi honest	Millionaire, AES, Arith Inner product
BatchDualEx	2PC	GC	Strong Adversary	-
ABY3	3PC	SS	Semi honest	Machine learning
SCALE MAMBA	General MPC	SS	Strong Adversary	General
CrypTen	MPC	SS	Semi honest	Focused on PyTorch applications
Tf-Encrypted	3PC	SS	Semi honest	Focused on Tensorflow applications