Secure 2PC

Oleg Fomenko

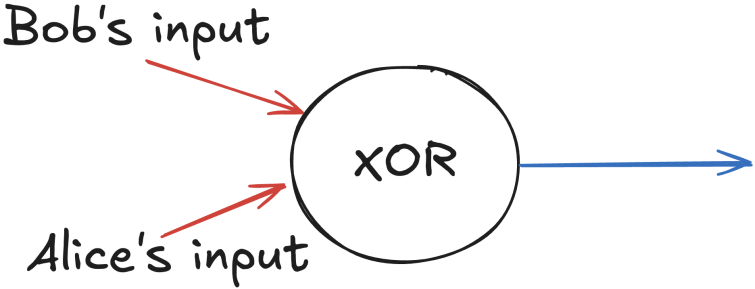
2025

What is 2PC?

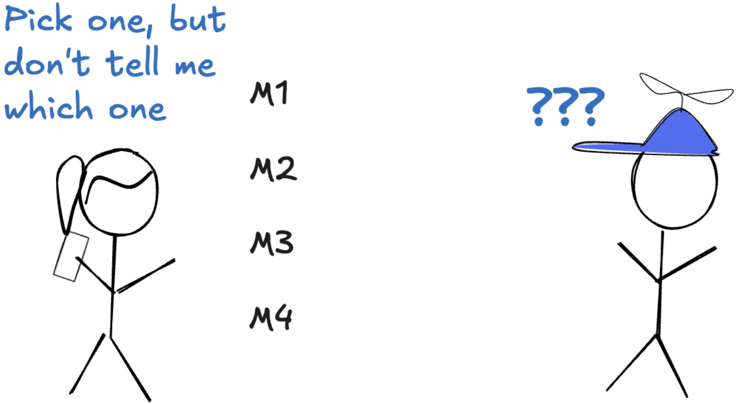
Secure 2PC – Secure Two Party Computation:

- $\mathcal{X}$ – Alice's private input
- $\mathcal{Y}$ – Bob's private input
- $\mathcal{O} = f(\mathcal{X}, \mathcal{Y})$ – public output for arbitrary circuit f

Simple example



Obvious transfer



Obvious transfer

- Alice selects $m_1, \cdots, m_n$
- Alice **DOES NOT** share them
- Bob selects $j \in \{1, \ldots, n\}$
- Alice and Bob run the OT protocol
- Bob receives m_j
- Alice **knows nothing** about j

Commutative encryption scheme

Commutative encryption – is a type of encryption scheme where the order of encryption with different keys doesn't matter:

$$\forall k_1, k_2 :$$
$$E_{k_1}(E_{k_2}(m)) = E_{k_1}(E_{k_2}(m))$$

$$\forall k_1, k_2\colon c = E_{k_1}(E_{k_2}(m))\colon$$
$$D_{k_1}(D_{k_2}(c)) = D_{k_2}(D_{k_1}(c))$$

Example

One-Time Pad:

- k – key (random bit string of size n)
- m – message (bit string of size n)
- $E(m, k) = k_i \oplus m_i \Big|_{i=0}^{n}$
- $D(m, k) = E(m, k)$

$$m \oplus k_1 \oplus k_2 = m \oplus k_2 \oplus k_1$$

CTR mode

- k – private key
- E_k – encryption function for private key k
- ctr_i – counter for i-th message
- m_i – i-th message
- $C_i = m_i \oplus E_k(ctr_i)$
- $m_i = C_i \oplus E_k(ctr_i)$

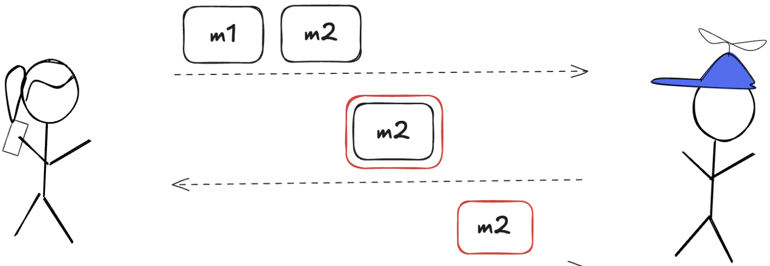
  Ref: Block cipher mode of operation

  Ref: Bruce Schneier: Practical Cryptography

Back to Obvious transfer

- Alice has $m_1, \ldots, m_n$
- Alice encrypts m_i and sends $E_A(m_1), \ldots, E_A(m_n)$ to Bob
- Bob selects j and submits $E_B(E_A(m_j))$ to Alice
- Alice decrypts ad submits $D_A(E_B(E_A(m_j))) = E_B(m_j)$ to Bob
- Bob decrypts and receives m_j

Obvious transfer



Important: Bob knows the reasons for picking a second value.

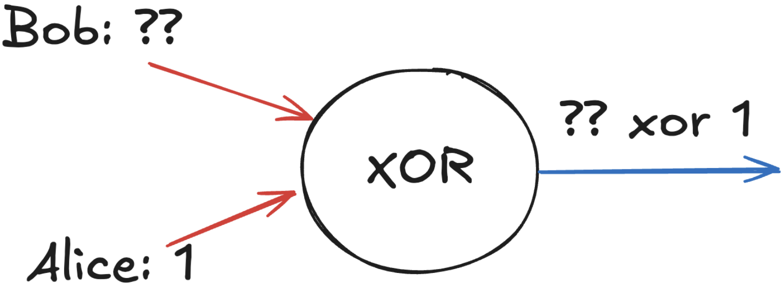
Back to 2PC: Garbled Circuits

Every possible XOR gate evaluation:

A	B	$\oplus$
0	0	0
1	0	1
0	1	1
1	1	0

Imagine Alice's bit equals 1, Alice reduces this table:

A	B	$\oplus$
1	0	1
1	1	0



Bob: ??
Alice: 1
XOR
?? xor 1

For each possible Bob's input, Alice puts:

A	B	$\oplus$
1	K_1 – corresponds to Bob's 0 input	$E_{K_1}(1)$
1	K_2 – corresponds to Bob's 1 input	$E_{K_1}(0)$

- Alice shares $E_{K_1}(1)$ and $E_{K_2}(0)$
- Alice and Bob run OT for K_1, K_2

From Bob's perspective

- Bob receives $E_{K_1}(1)$ and $E_{K_2}(0)$
- Imagine Bob's input equals 0
- Bob using K_1 decrypts $E_{K_1}(1)$ receiving result 1
- Bob shares the result with Alice if needed

Summary

- Alice, using her input, evaluates a circuit for each possible Bob's input
- Alice picks keys for each possible Bob's input and encrypts each possible result
- Bob selects a key for his input using the OT protocol
- Bob opens the result

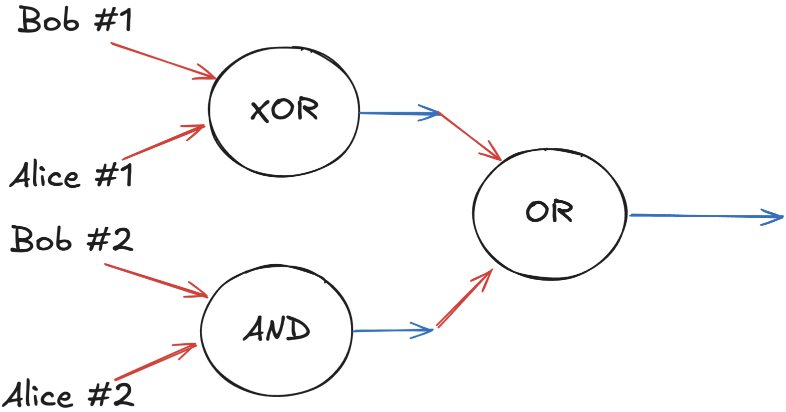
Third side

- Anton evaluates circuit results for each possible pair of inputs from Alice and Bob.
- Anton doubly encrypts each with different keys:

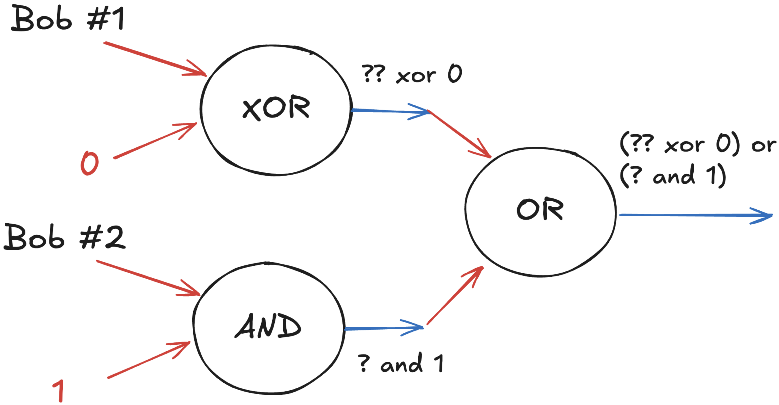
A	B	$\bigoplus$
0	0	$E_{K_A^1}(E_{K_B^1}(0))$
1	0	$E_{K_A^1}(E_{K_B^0}(1))$
0	1	$E_{K_A^0}(E_{K_B^1}(1))$
1	1	$E_{K_A^0}(E_{K_B^0}(0))$

- Alice selects a key for her bit, decrypts the corresponding results, and shares with Bob
- Bob selects a key, decrypts the result

Multiple gates



Multiple gates



$$f(??, ?) = (?? \oplus 0) \vee (? \wedge 1)$$

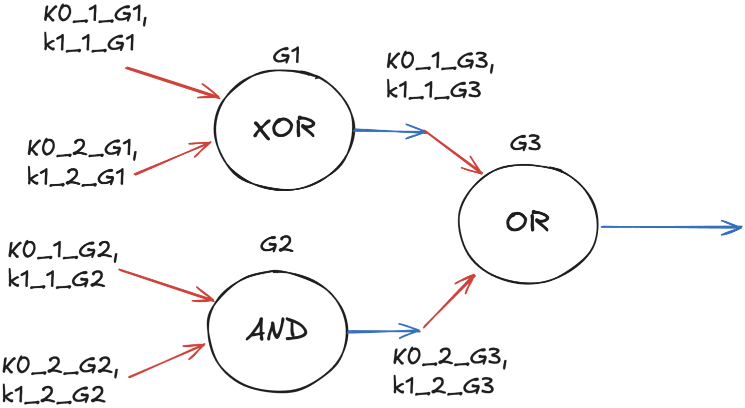
Bob 1 (??)	Bob 2 (?)	f
0	0	0
1	0	1
0	1	1
1	1	1

WRONG! requires more work from the circuit generator side: $\mathcal{O}(2^n)$. We want $\mathcal{O}(g)$

n – number of inputs, g – number of gates

Solution

- For each gate's input, select a pair of keys for 0 and 1 bits.
- All intermediate gates return corresponding keys to the next gate input
- Output gate returns 0/1


K0_1_G1,
k1_1_G1
K0_2_G1,
k1_2_G1
G1
XOR
K0_1_G3,
k1_1_G3
G3
OR
K0_1_G2,
k1_1_G2
K0_2_G2,
k1_2_G2
G2
AND
K0_2_G3,
k1_2_G3

To read about:

- Garbled Circuits
- Obvious transfer
- CTR cipher mode of operation
- Commutative encryption