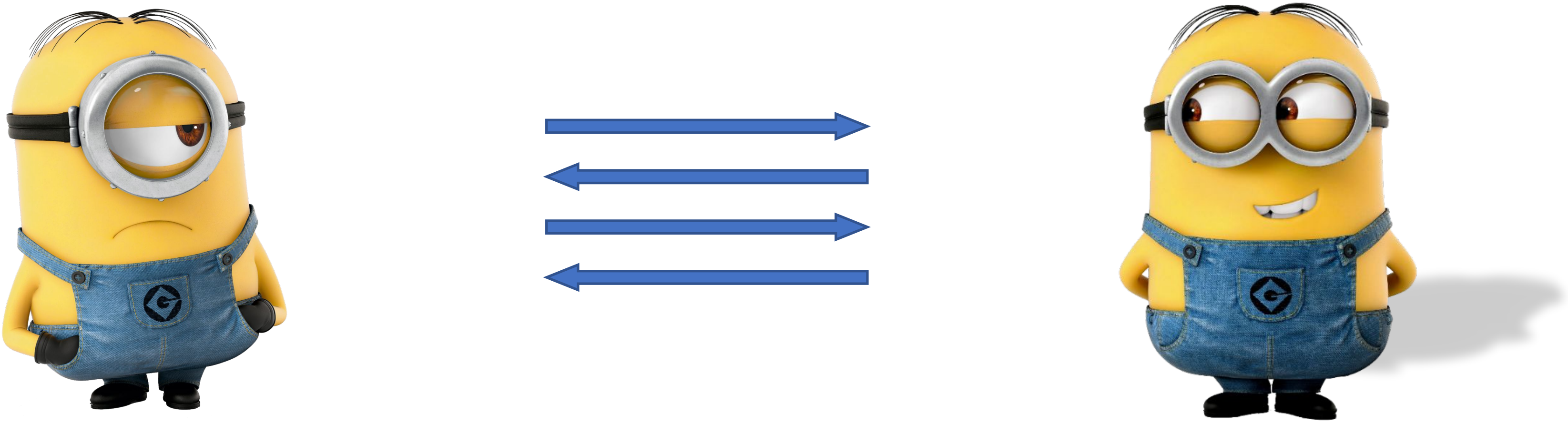
Secure Multi-Party Computation

Secure two-party computation

f

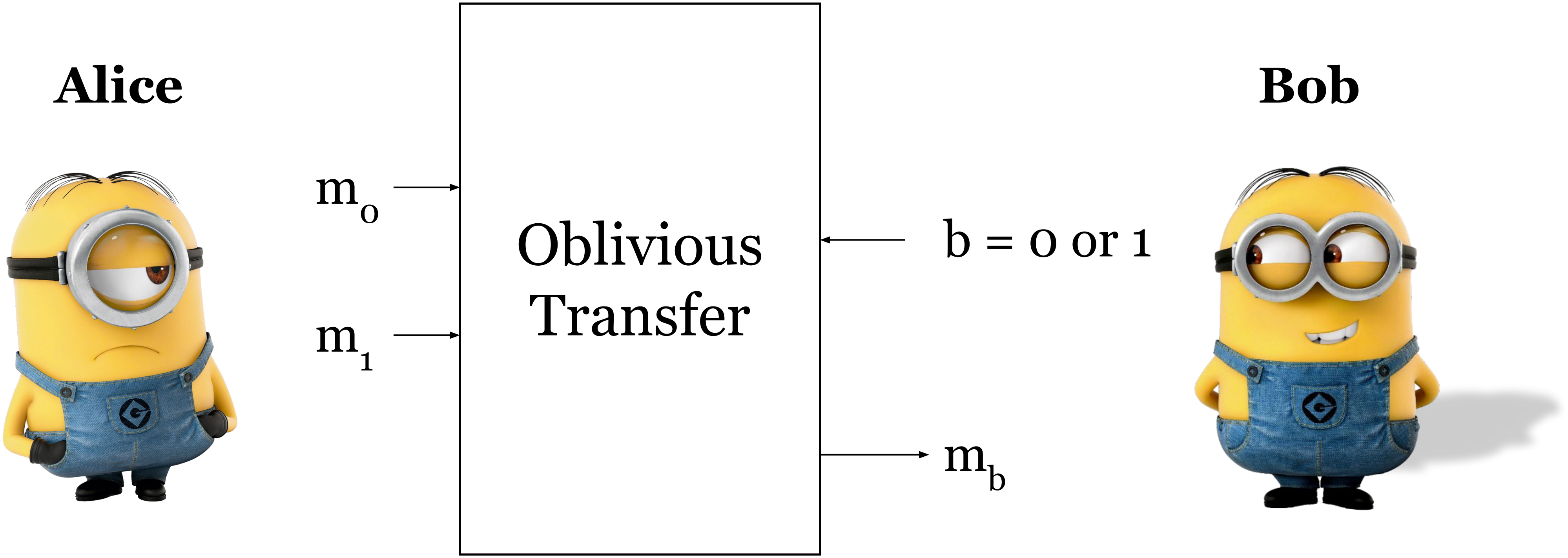
x y



$f(x, y)$ $f(x, y)$

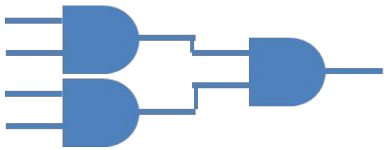
x and y remain secret

Building block: oblivious transfer (OT)



Putting everything together

f



x

Alice

Garbled Table
Enc A_0, B_0 (C_0)
Enc A_0, B_1 (C_0)
Enc A_1, B_0 (C_0)
Enc A_1, B_1 (C_1)

Garbled Table
Enc D_0, E_0 (F_0)
Enc D_0, E_1 (F_0)
Enc D_1, E_0 (F_0)
Enc D_1, E_1 (F_1)

Garbled Table
Enc D_0, E_0 (F_0)
Enc D_0, E_1 (F_0)
Enc D_1, E_0 (F_0)
Enc D_1, E_1 (F_1)

Keys for x: e.g., A_0 , B_1

Keys for y: oblivious transfer

$D_0 \rightarrow$ OT $\leftarrow y_0 = 1$

$D_1 \rightarrow$ OT $\rightarrow D_1, E_0$

G_0

$f(x,y) = 0$

y

Bob

Properties of Yao’s garbled circuit

- Constant round
- Boolean circuits

Goldreich-Micali-Wigderson(GMW) protocol

Secret sharing

Alice

$$a = a_0 \oplus a_1$$

a_0 a_1

Bob

Why XOR?

Truth table of XOR

Inputs		Outputs
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Generalization to finite field

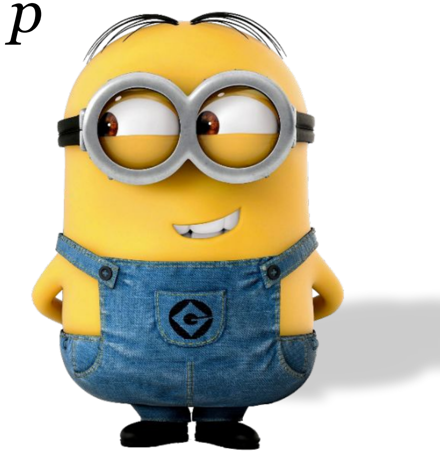
Alice

a

$a_0 = a - r \bmod p$

$a_1 = r \bmod p$

Bob



Why finite field?

Generalization to multiple parties

$$a = a_0 \oplus a_1 \oplus a_2 \oplus \ldots \oplus a_n$$

$$a = a_1 + a_1 + a_2 + \ldots + a_n \bmod p$$

Generalization to threshold t out of n

How to Share a Secret, Adi Shamir 1979

Idea of GMW protocol

Boolean circuits with XOR + AND gates

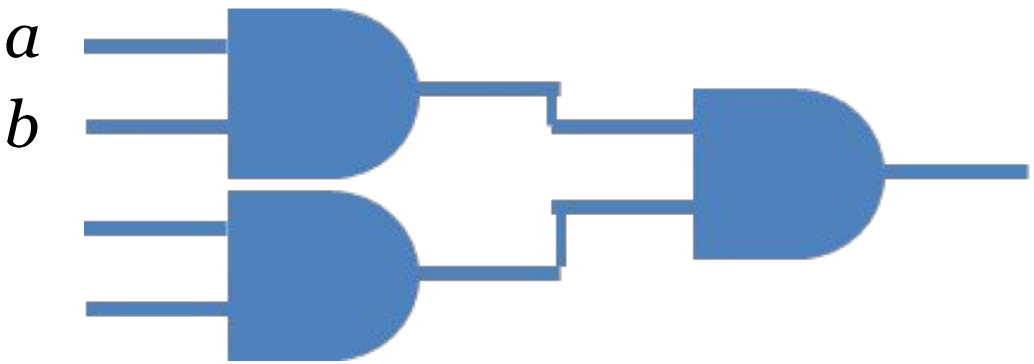
Why?

XOR + AND is universal: construct NAND!!!

- Inputs are secret-shared between two parties
- After every gate, the output is secret-shared between two parties

GMW protocol

Input:

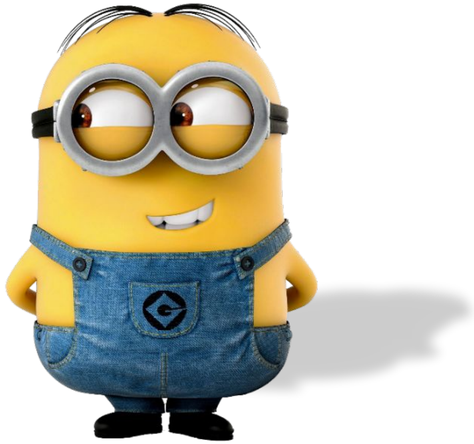


Alice

Bob

$a \quad a_0 = a \oplus a_1$

$\xrightarrow{a_1}$



$\xleftarrow{b_0}$

$b_1 = b \oplus b_0 \quad b$

GMW protocol

XOR gates

Alice

a_0

b_0

$c_0 = (a_0 \oplus b_0)$

Bob

a_1

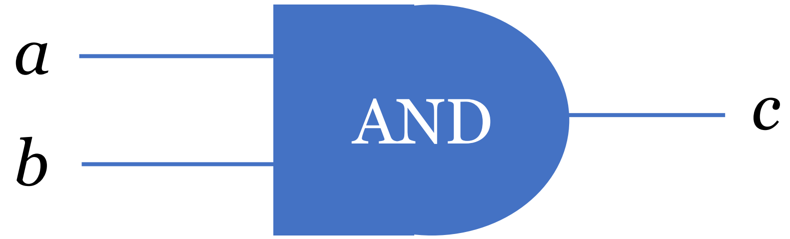
b_1

$c_1 = (a_1 \oplus b_1)$

$$
\begin{aligned}
c &= a \oplus b \\
&= (a_0 \oplus a_1) \oplus (b_0 \oplus b_1) \\
&= (a_0 \oplus b_0) \oplus (a_1 \oplus b_1)
\end{aligned}
$$

GMW protocol

AND gates



Alice

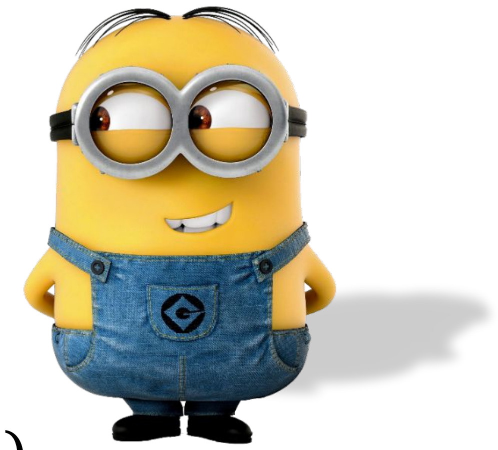
a_0

b_0

a_1

b_1

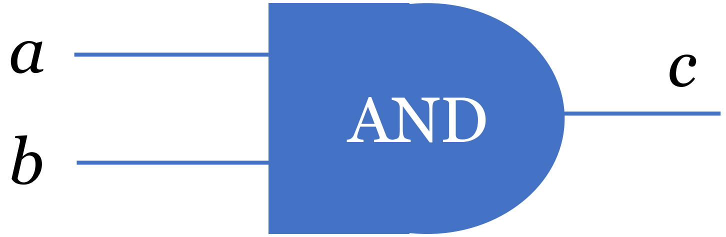
Bob



$$
\begin{aligned}
c &= a \wedge b \\
&= (a_0 \oplus a_1) \wedge (b_0 \oplus b_1) \\
&= (a_0 \wedge b_0) \oplus (a_1 \wedge b_1) \oplus (a_0 \wedge b_1) \oplus (a_1 \wedge b_0)
\end{aligned}
$$

GMW protocol

AND gates

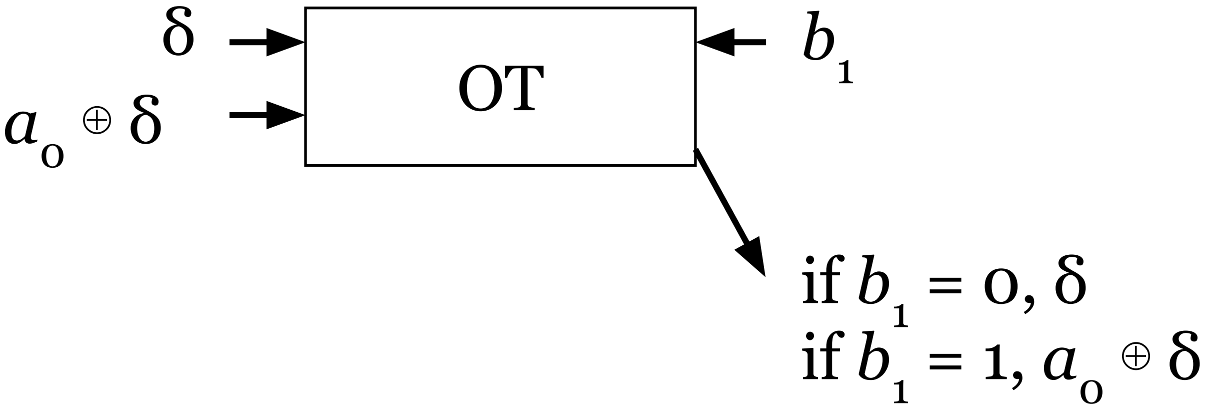


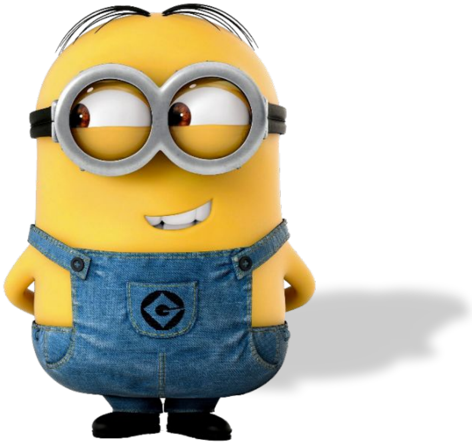
Alice

a_0

b_1

Bob

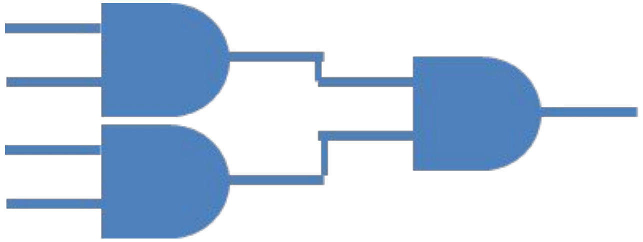




δ

Putting everything together

f

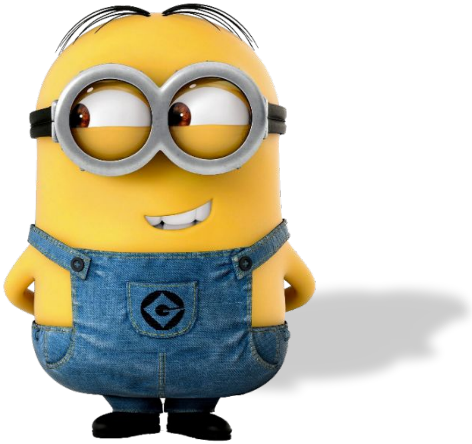


x

Alice

y

Bob



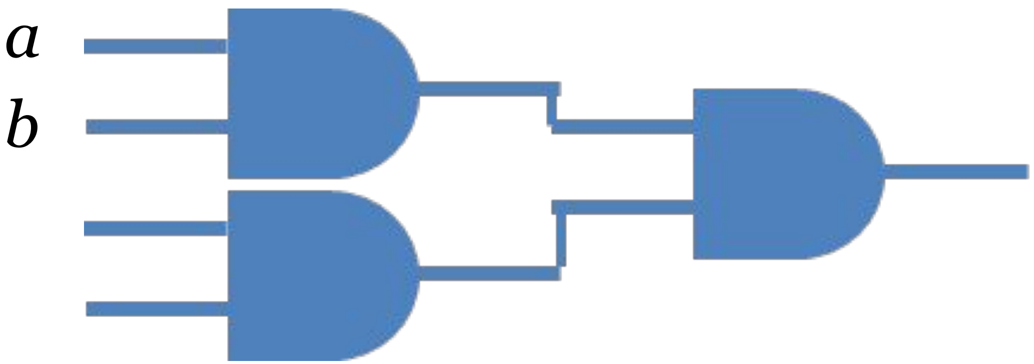
1. Secret sharing inputs
2. XOR locally for every XOR gate
3. Run OTs for every AND gate
4. Reconstruct the output

Properties of GMW protocol

- Interactive, O(depth) rounds
- Can be generalized to arithmetic circuits

Generalization to arithmetic circuits

Input:

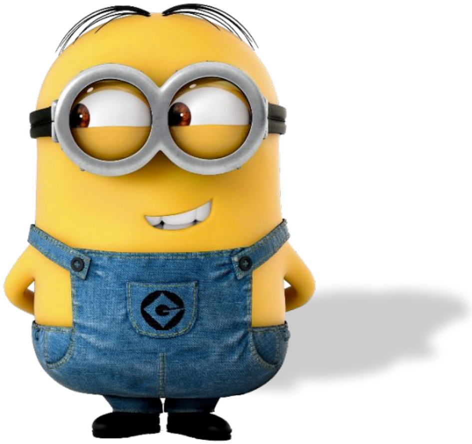


Alice

Bob

a $\quad a_0 = a + a_1 \bmod p$

a_1 $\longrightarrow$



b_0 $\longleftarrow$

b

$b_1 = b + b_0 \bmod p$

Generalization to arithmetic circuits

Addition gates

Alice

a_0

b_0

$c_0 = (a_0 + b_0)$

Bob

a_1

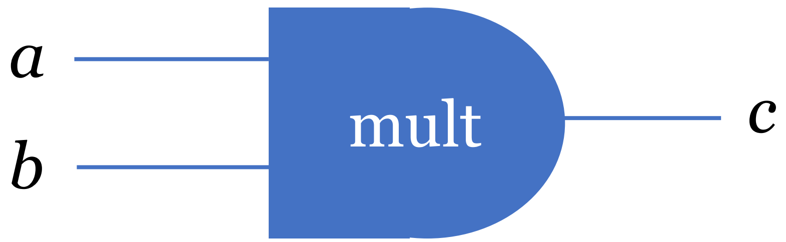
b_1

$c_1 = (a_1 + b_1)$

$$
\begin{aligned}
c &= a + b \\
&= (a_0 + a_1) + (b_0 + b_1) \\
&= (a_0 + b_0) + (a_1 + b_1) \bmod p
\end{aligned}
$$

Generalization to arithmetic circuits

Multiplication gates



Alice

a_0

b_0

a_1

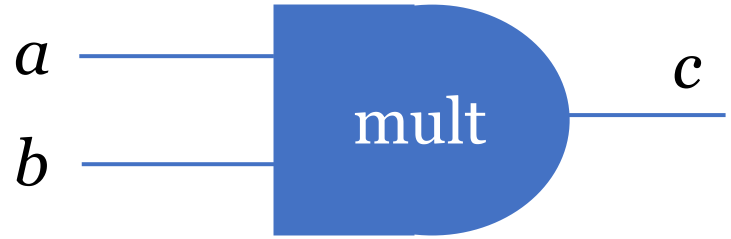
b_1

Bob

$$
\begin{aligned}
c &= a \times b \\
&= (a_0 + a_1) \times (b_0 + b_1) \\
&= (a_0 \times b_0) + (a_1 \times b_1) + (a_0 \times b_1) + (a_1 \times b_0) \bmod p
\end{aligned}
$$

Generalization to arithmetic circuits

Multiplication gates

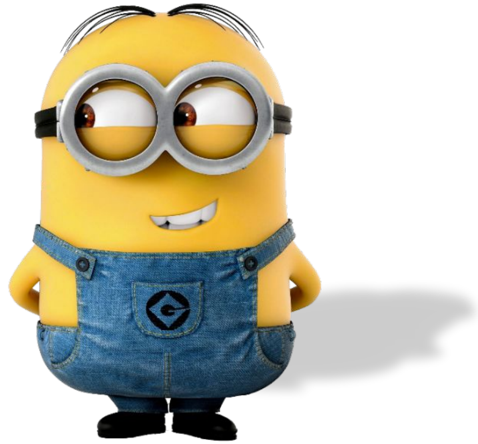


Alice

a_0

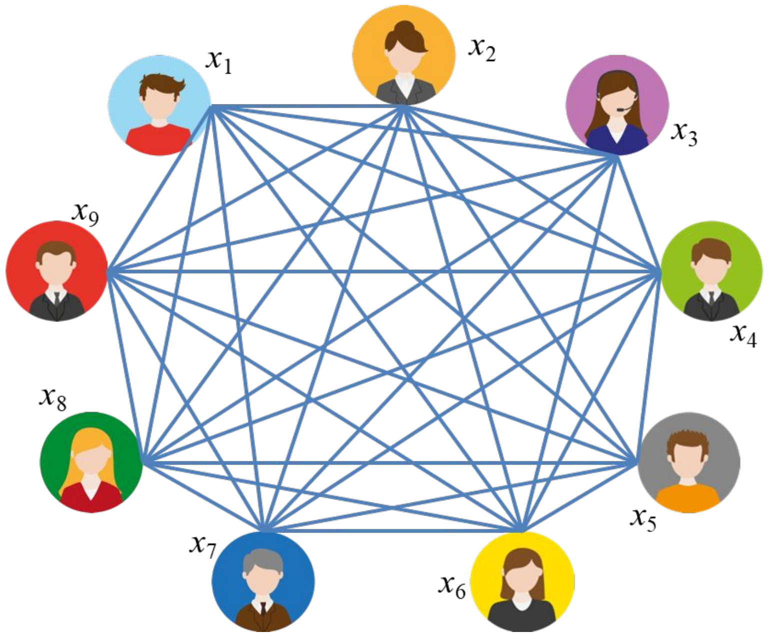
b_1

Bob



OT using every bit of b_1

Generalization to multiparty computation



$a = a_0 \oplus a_1 \oplus a_2 \oplus \ldots \oplus a_n$ $\qquad$ $b = b_0 \oplus b_1 \oplus b_2 \oplus \ldots \oplus b_n$

$a \oplus b = (a_0 \oplus b_0) \oplus (a_1 \oplus b_1) \oplus \ldots \oplus (a_n \oplus b_n)$

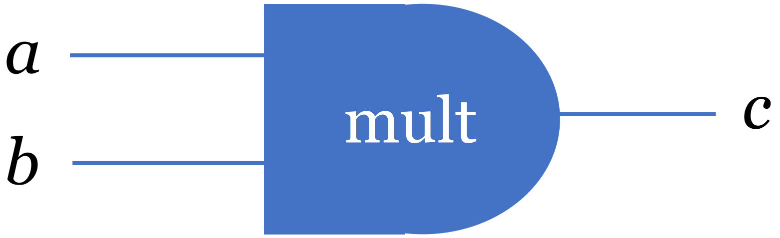
$a \wedge b = \sum a_i \wedge b_i \oplus \sum a_i \wedge b_j$

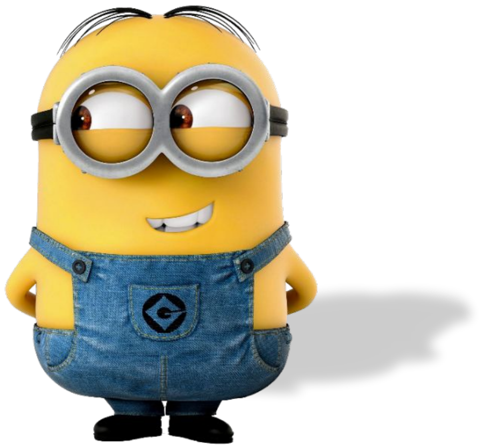
Properties of GMW protocol

- Interactive, O(depth) rounds
- Can be generalized to arithmetic circuits
- Easily extended to multi-party

Multiplication triplets

Multiplication gates:





$$a_0 - u_0 \,,\, b_0 - v_0 \longrightarrow$$

$$\longleftarrow a_1 - u_1 \,,\, b_1 - v_1$$

a_0 $\quad$ b_0

$e = a - u$

$f = b - v$

a_1 $\quad$ b_1

$e = a - u$

$f = b - v$

$c_0 = -ef + a_0 f + eb_0 + z_0$

$c_1 = a_1 f + eb_1 + z_1$

$c_0 + c_1 = a \times b$

u_0, v_0, z_0

u_1, v_1, z_1

$$(u_0 + u_1) \times (v_0 + v_1) = (z_0 + z_1)$$

Online-offline cost

- Multiplication triplets are data independent
- Cannot be reused

All types of “triplets”

- Square, truncation, etc, …