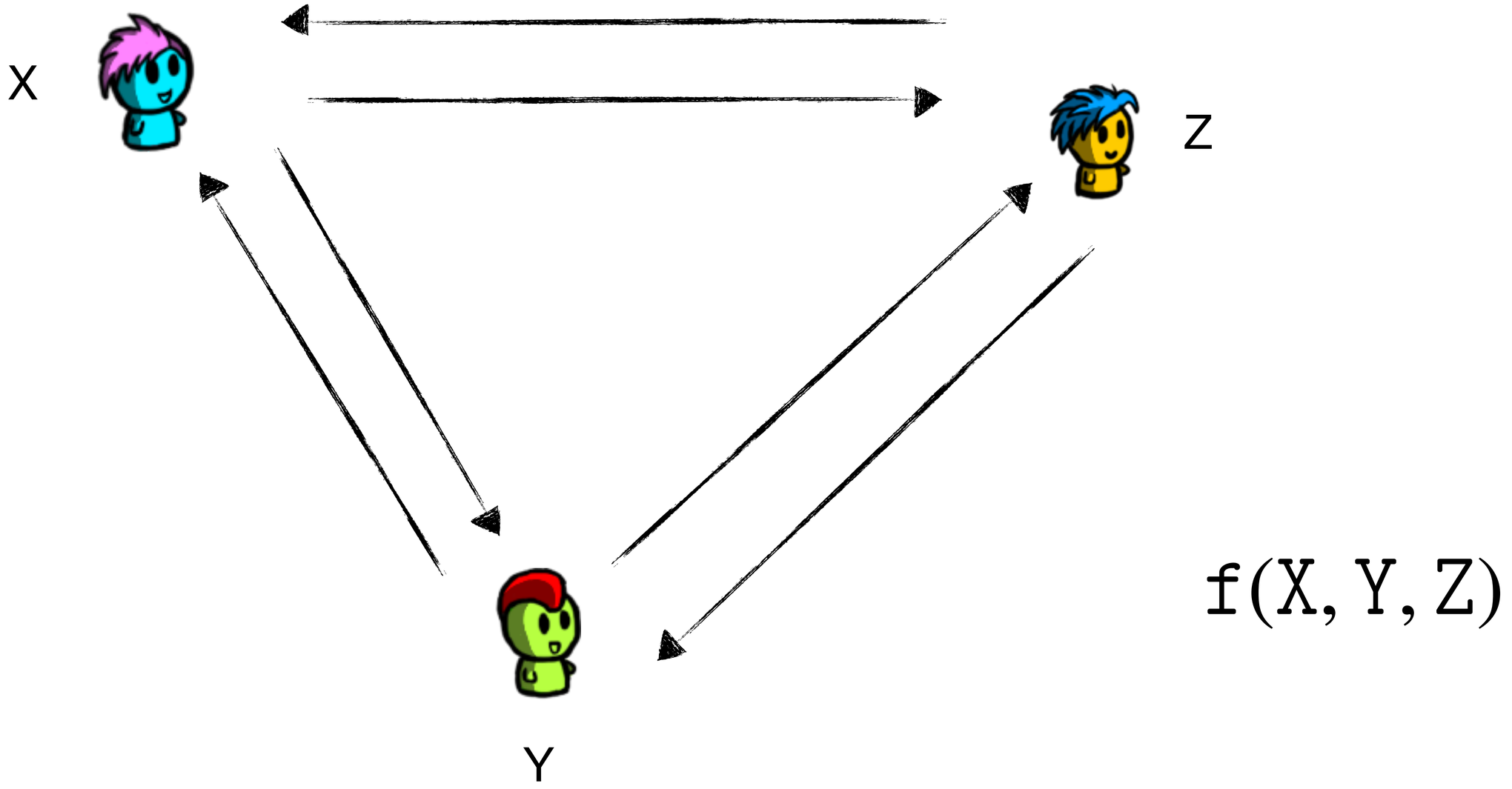
CS670: Cryptographic Techniques for Privacy Preservation

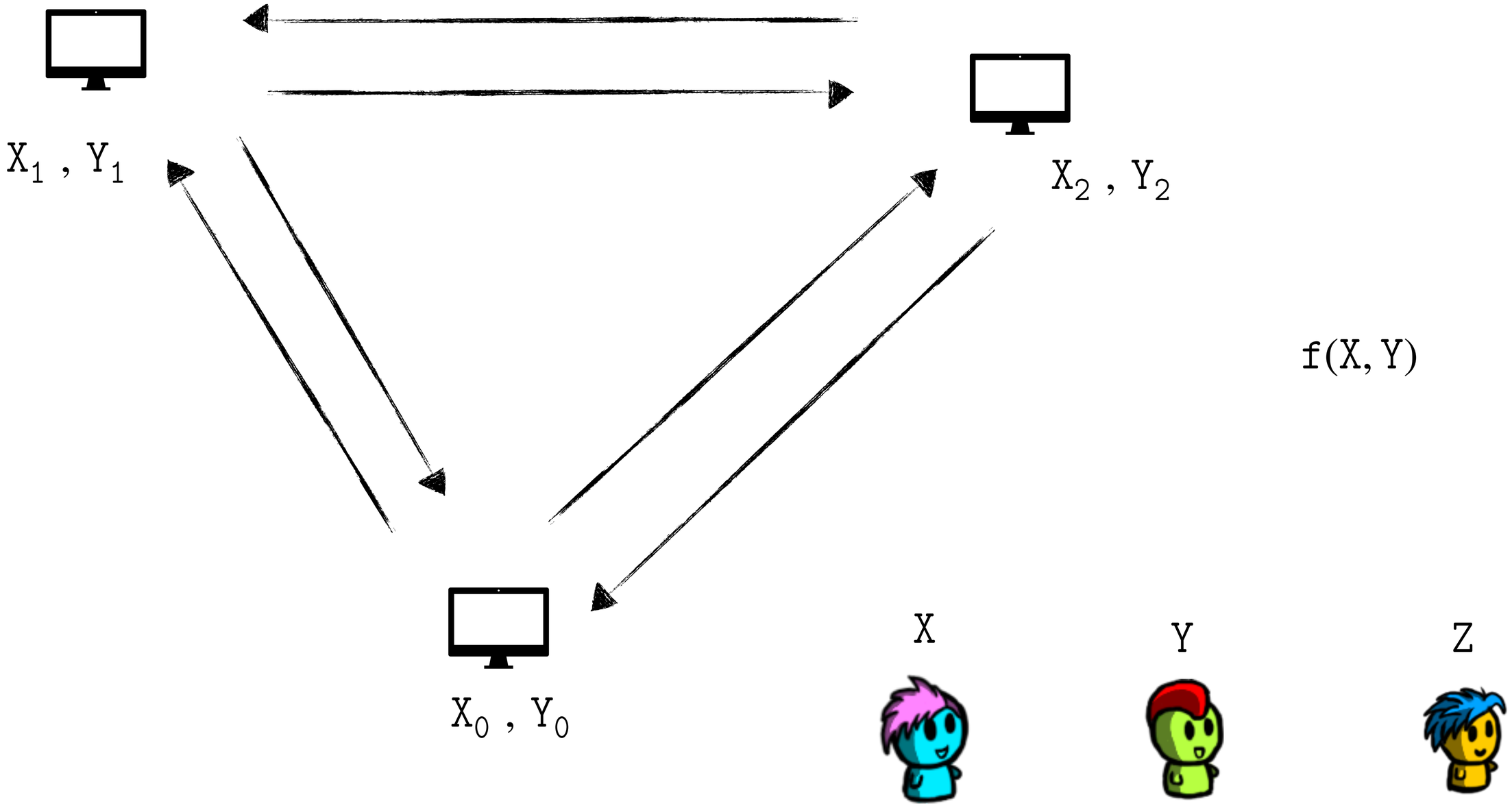
Module 1: Secure Multiparty Computation

Adithya Vadapalli

Drawings of Alice, Bob, and other crypto characters by Simon Oya: https://simonoya.com/drawings/



f(X, Y, Z)


X_1 , Y_1
X_2 , Y_2
X_0 , Y_0
f(X, Y)
X
Y
Z

Properties of MPC

Expressibility

Minimum # Parties

Threat Model

Maximum # Adversarial Parties

Performance

Expressibility

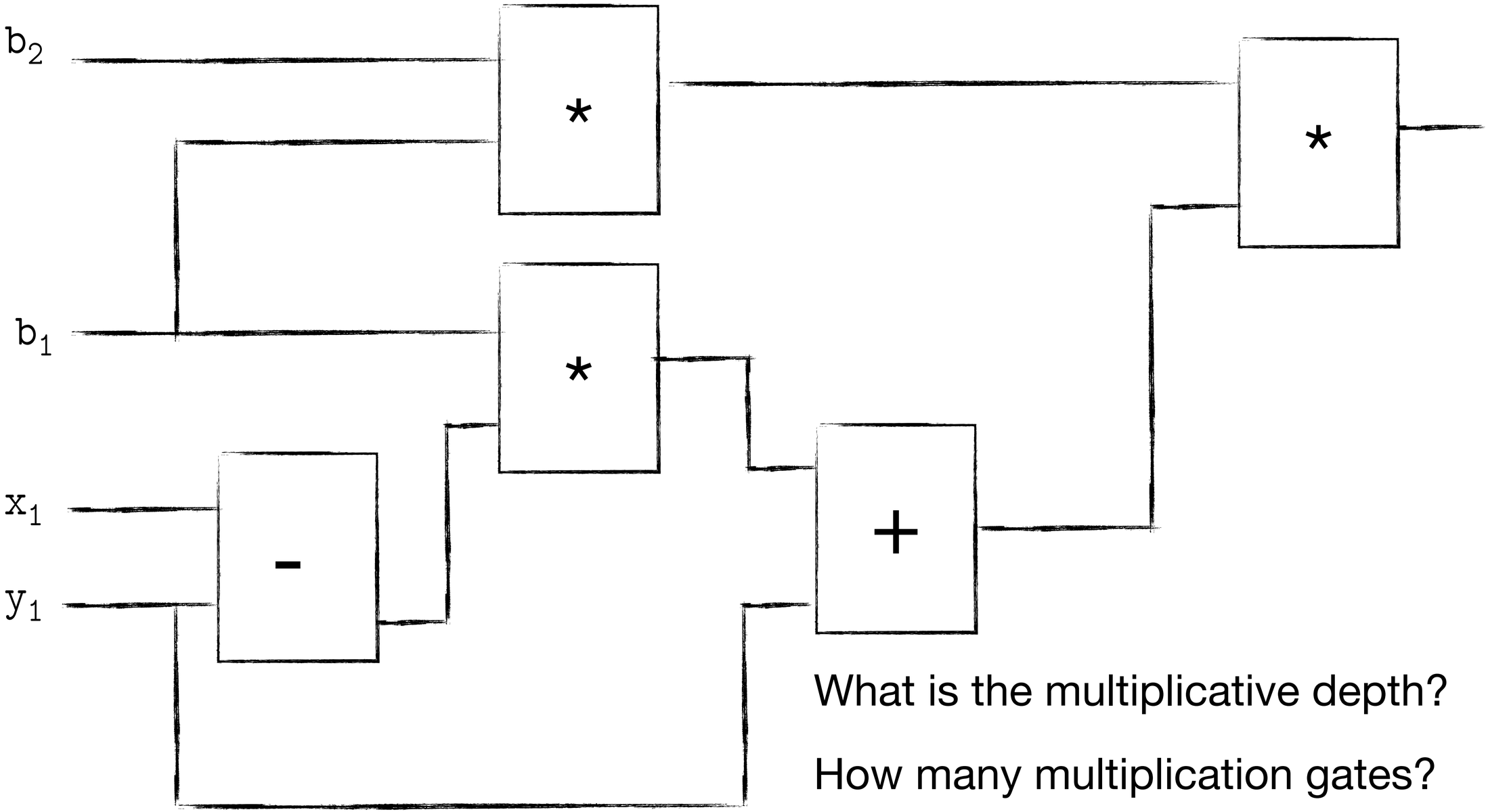
What kind of functions can the MPC protocol evaluate?

Generic

Some MPC protocols can compute any function

Specific

Some MPC protocols are only for specific function



What is the multiplicative depth?

How many multiplication gates?

Types of Gates

Linear Gates ⟶ Typically very easy to compute

$x_0 \quad y_0$

$x_1 \quad y_1$

hold shares of x and y

How can they get shares of: $\alpha \cdot x + \beta \cdot y$

$z_0 \leftarrow \alpha \cdot x_0 + \beta \cdot y_0$

$z_1 \leftarrow \alpha \cdot x_1 + \beta \cdot y_1$

Non-Linear Gates ⟶ More complicated

Min # Parties

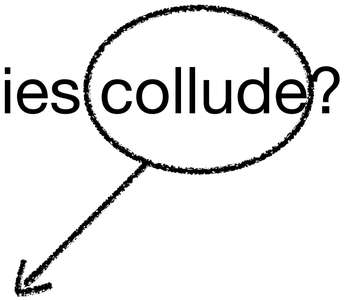
How many computational parties are needed to run the protocol?

The larger the value, the more challenging it is to deploy the protocol.

A protocol that requires, for example, 3 Parties is called a 3PC

Threat Model

Does the protocol remain secure if some of the parties collude?



Two parties collude if they share with each other the secret values they are not supposed to.

Max # Adversarial Parties

How many parties in the protocol can be adversarial?

Honest Majority

Honest Minority

Performance

Local Computation • Round Complexity • Bandwidth Consumption

Which is the most important parameter?

Depends on the deployment scenario.

Secure Multiparty Computation

(On Secret Shares)

x_0 , y_0

x_1 , y_1

$x_0 + x_1 = x$

$y_0 + y_1 = y$

Alice and Bob hold additive shares of: x *and* y

Their goal is to get additive shares of: $x + y$

This is basically free *(in terms of communication)*

$z_0 \leftarrow x_0 + y_0$

$z_1 \leftarrow x_1 + y_1$

Secure Multiparty Computation

(On Additive Secret Shares)

x_0 , y_0

x_1 , y_1

$x_0 + x_1 = x$

$y_0 + y_1 = y$

Alice and Bob hold additive shares of: x *and* y

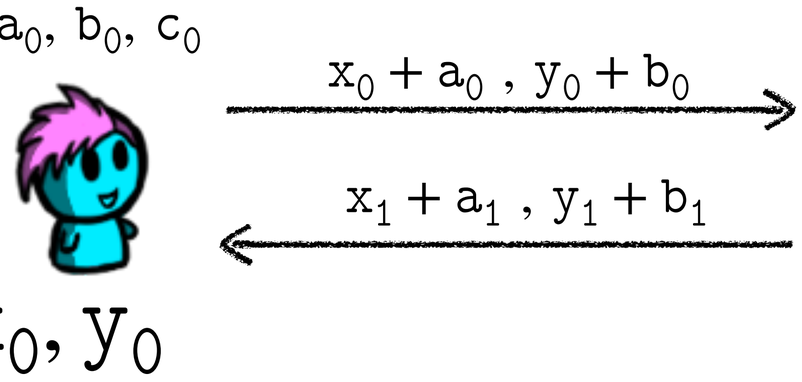
Their goal is to get additive shares of: $x \cdot y$

Secure Multiparty Computation

(On Additive Secret Shares)

Beaver's Triples:

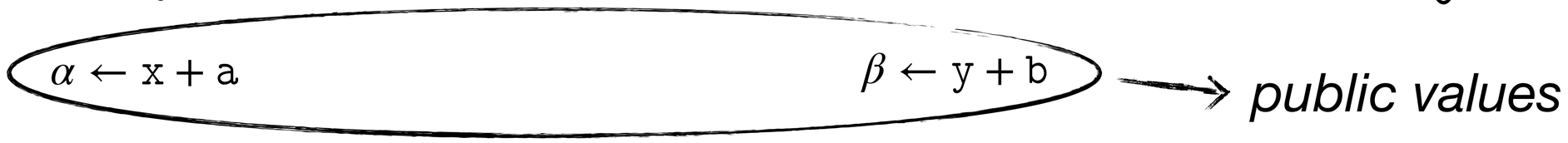
Distribute shares of random inputs (a and b) and their product c



$$x_0 + x_1 = x$$

$$y_0 + y_1 = y$$

Goal: z_0 z_1 *such that* $z_0 + z_1 = z$

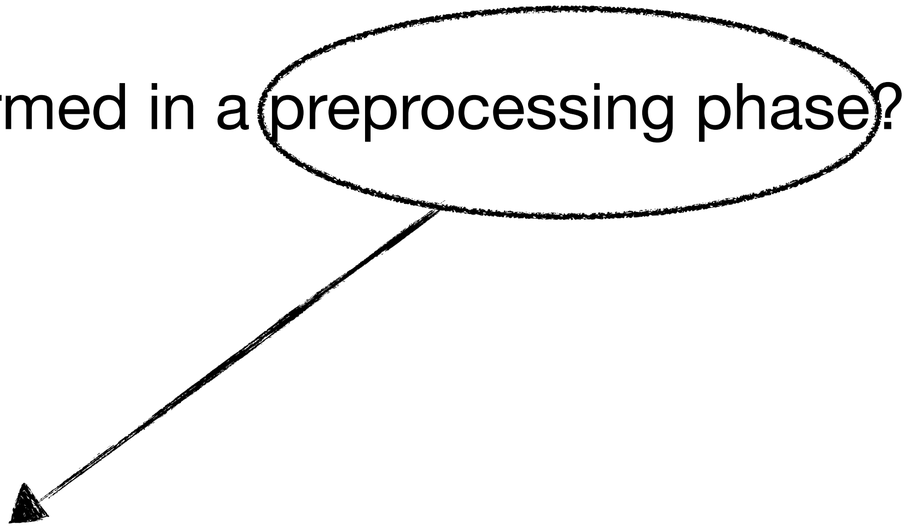


$z_0 \leftarrow \alpha \cdot y_0 - \beta \cdot a_0 + c_0$

$z_1 \leftarrow \alpha \cdot y_1 - \beta \cdot a_1 + c_1$

Secure Multiparty Computation

Can any of the operations be performed in a preprocessing phase?



Can be done ahead of time before we know the actual inputs.

Secure Multiparty Computation

Preprocessing Phase

vs

Online Phase

It can happen before the actual inputs are known

It is okay if this process is somewhat expensive

In Multiplication: Beaver Triples

Happens when the actual inputs are known

Ideally you'd like it to be as efficient as possible

The exchange of blinds and computation of the answer

Secure Multiparty Computation

(On Additive Secret Shares)

$x_0 \cdot y_0$

$+$

$x_0 \cdot (y_1 + Y_1)$

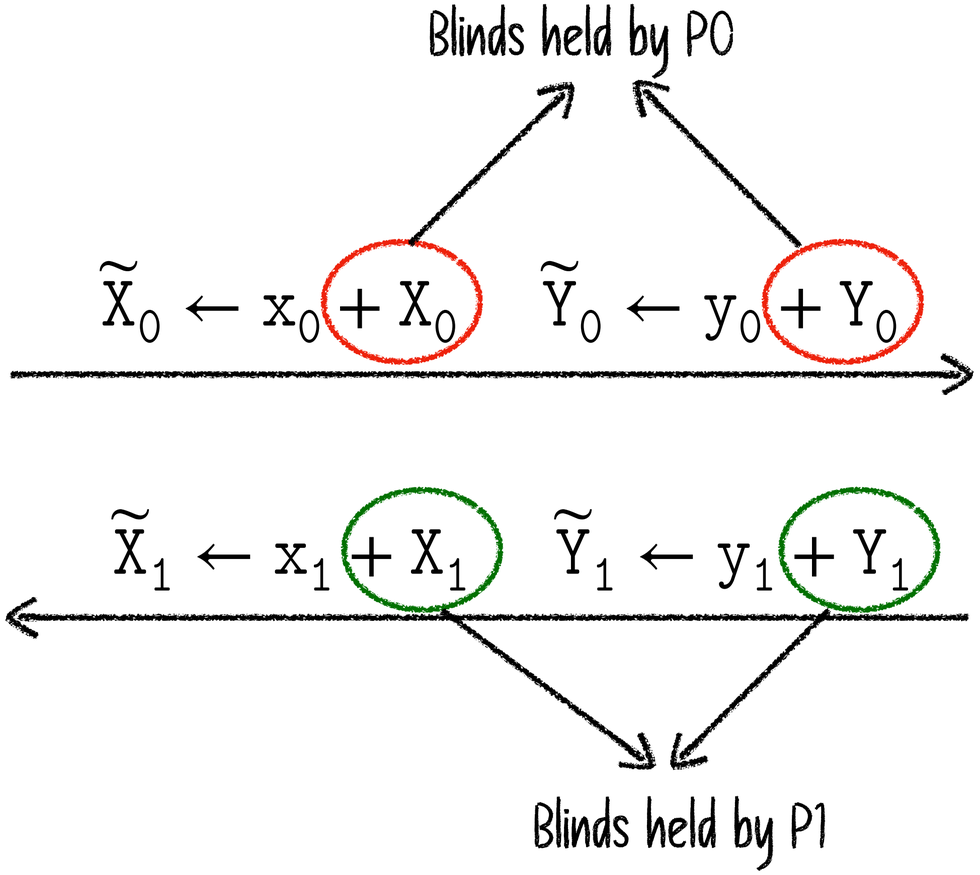
$x_1 \cdot y_1$

$+$

$x_1 \cdot (y_0 + Y_0)$

$x_0 + x_1 = x$

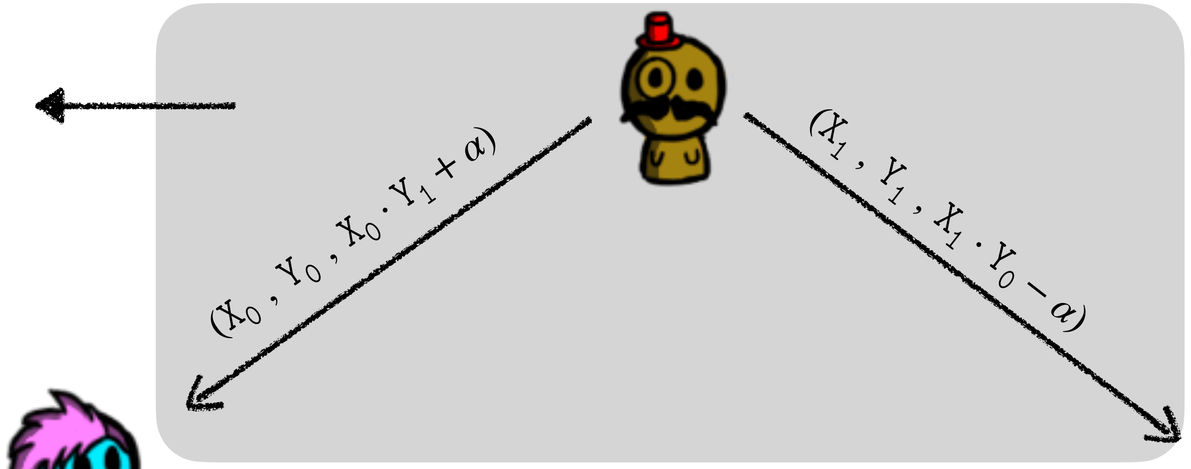
$y_0 + y_1 = y$



x_0 , y_0

x_1 , y_1

Secure Multiparty Computation $\wedge$ (On Additive Secret Shares)



$x_0 + x_1 = x$

$y_0 + y_1 = y$

x_0 y_0

x_1, y_1

$x_0 \cdot (y_0 + \tilde{Y}_1) - Y_0 \cdot \tilde{X}_1 + X_0 \cdot Y_1 + \alpha$

$\tilde{X}_0 \leftarrow x_0 + X_0 \quad \tilde{Y}_0 \leftarrow y_0 + Y_0$

$x_1 \cdot (y_1 + \tilde{Y}_0) - Y_1 \cdot \tilde{X}_0 + X_1 \cdot Y_0 + \alpha$

$\tilde{X}_1 \leftarrow x_1 + X_1 \quad \tilde{Y}_1 \leftarrow y_1 + Y_1$

Properties of the Protocol

Expressibility: generic

Minimum number of parties: 2

Threat Model: Semi-honest

Maximum number of adversarial parties: 1

Performance (g total gates, m multiplication gates, multiplication depth d):

- Local Computation: $\mathcal{O}(g)$
- Total Communication: 6m (preprocessing) + 2m per party
- Round Complexity: 1 preprocessing + d

Secure Multiparty Computation

(On Replicated Secret Shares)

x_0, x_1

y_0, y_1

x_1, x_2

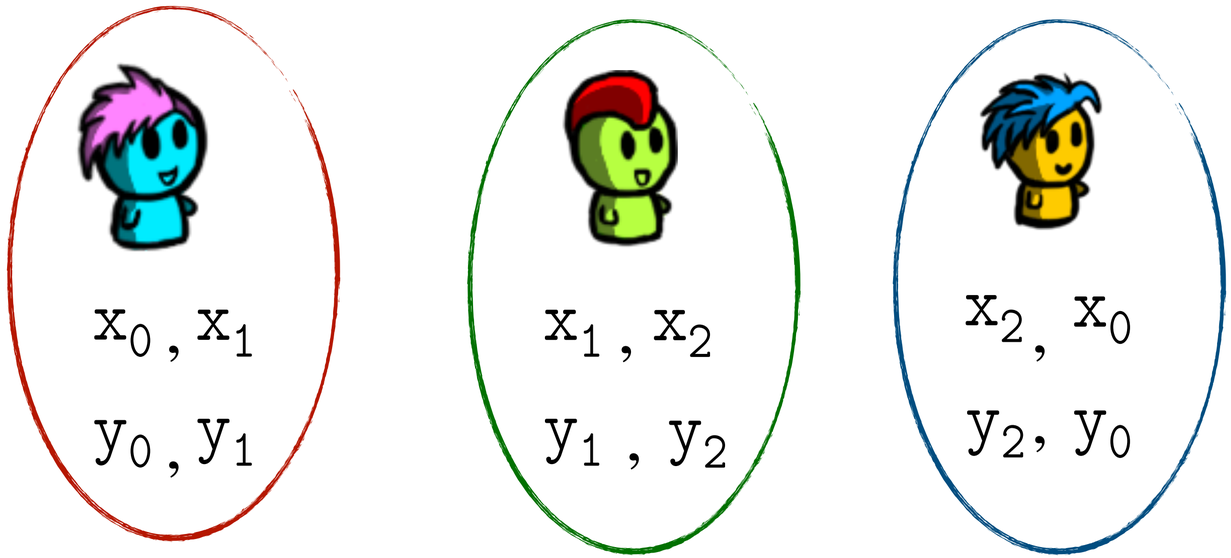
y_1, y_2

x_3, x_0

y_3, y_0

$$z_0 + z_1 + z_2 = (x_0 + x_1 + x_2) \cdot (y_0 + y_1 + y_2)$$

Secure Multiparty Computation



(On Replicated Secret Shares)

$$z_0 + z_1 + z_2 = (x_0 + x_1 + x_2) \cdot (y_0 + y_1 + y_2)$$

$$x_0 \cdot y_0 + x_0 \cdot y_1 + x_0 \cdot y_2 +$$
$$x_1 \cdot y_0 + x_1 \cdot y_1 + x_1 \cdot y_2 +$$
$$x_2 \cdot y_0 + x_2 \cdot y_1 + x_2 \cdot y_2 +$$

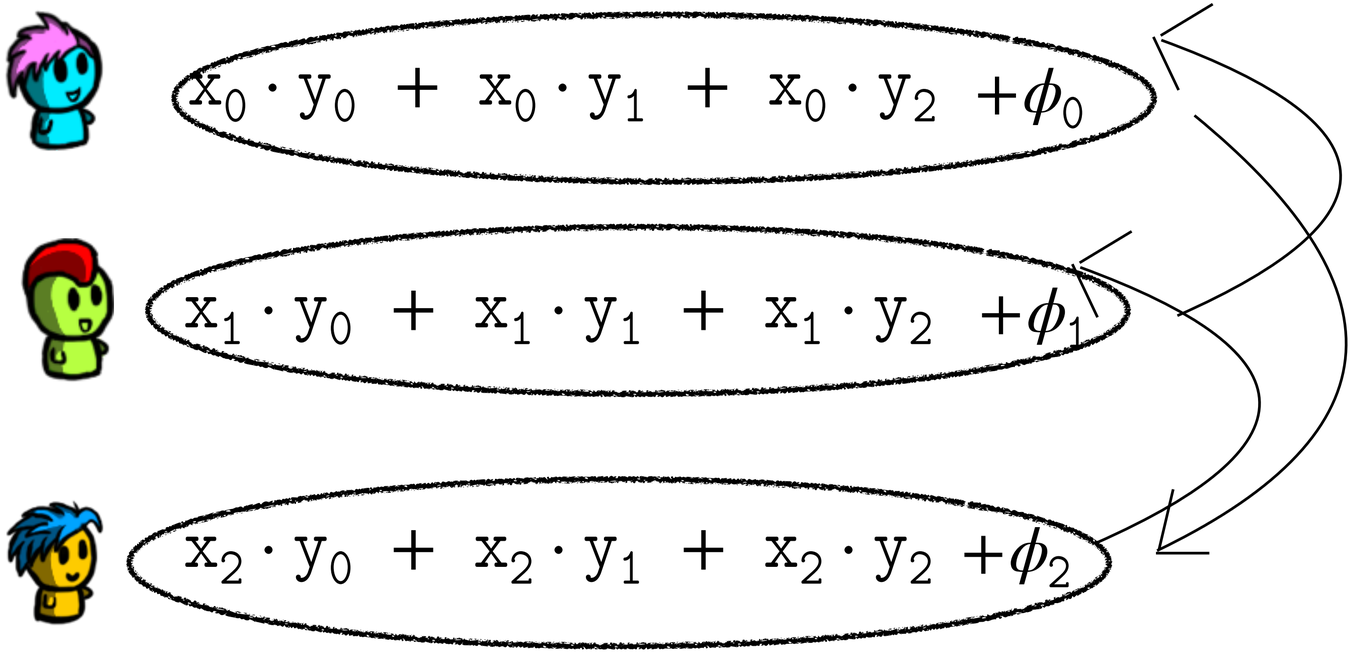
What is the problem with this?

Acadly Attendance Check

Zero Sharing

(On Replicated Secret Shares)

Parties can compute shares of zero: $\phi_0 + \phi_1 + \phi_2 = 0$



Zero Sharing

(On Replicated Secret Shares)

How do you get *zero* shares?

PRGs

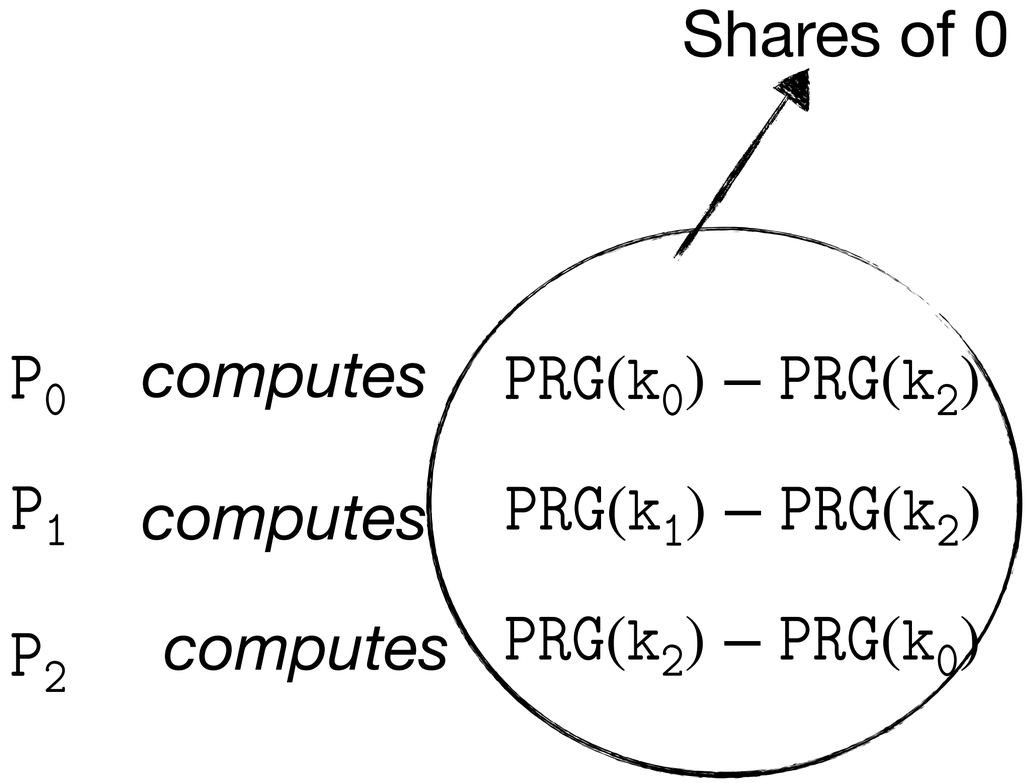
Each party i picks a random key k_i

In the preprocessing phase:

$P_0 \xrightarrow{k_0} P_1$

$P_1 \xrightarrow{k_1} P_2$

$P_2 \xrightarrow{k_2} P_0$



Properties of the Protocol

Expressibility: generic

Minimum number of parties: 3

Threat Model: Semi-honest

Maximum number of adversarial parties: 1

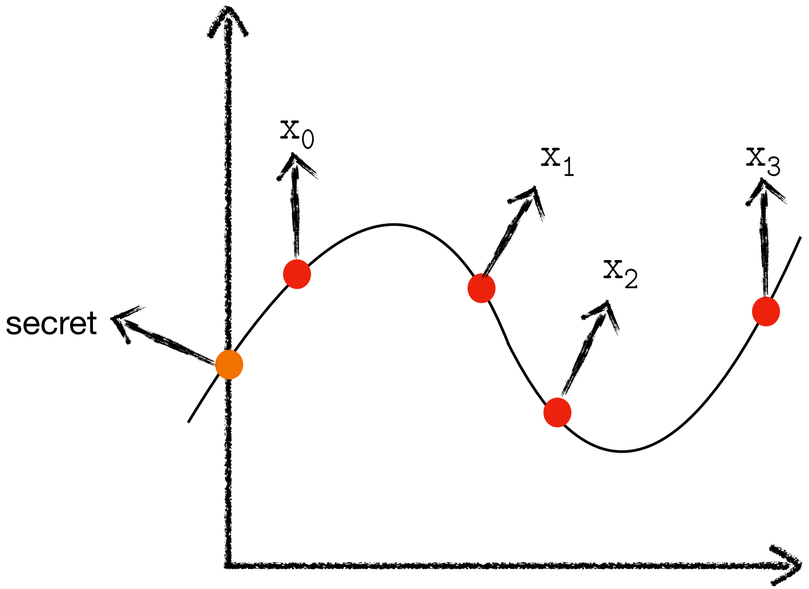
Performance (*g* total gates, *m* multiplication gates, multiplication depth *d*):

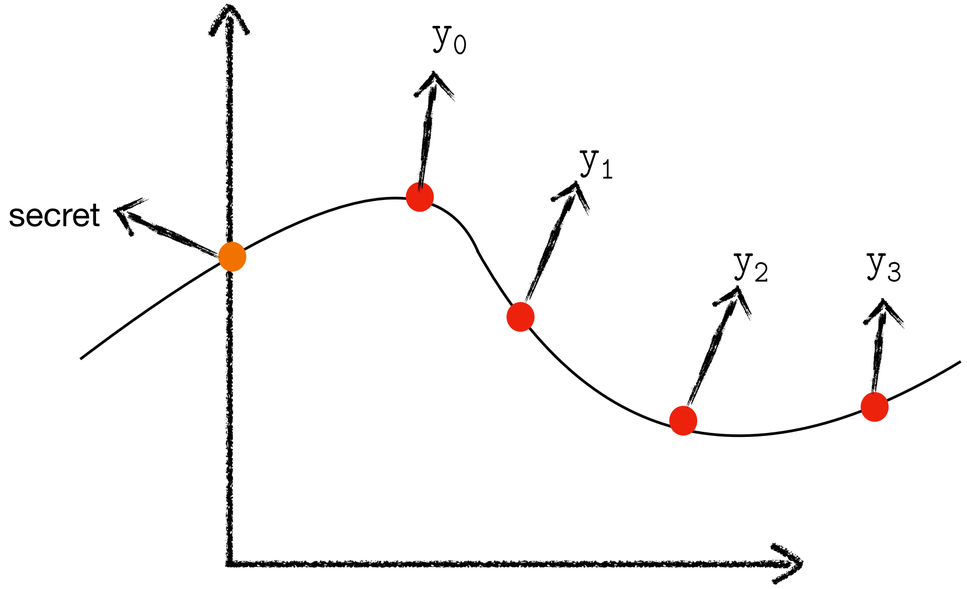
Local Computation: $\mathcal{O}(g)$

Total Communication: 3 (preprocessing) + m per party

Round Complexity: 1 preprocessing + d

Multiplication in Shamir Secret Shares



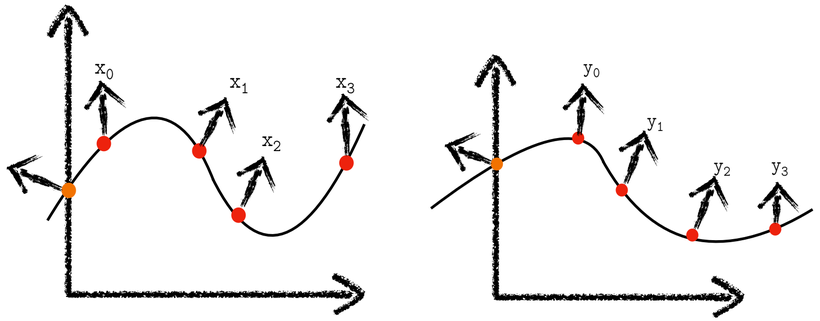


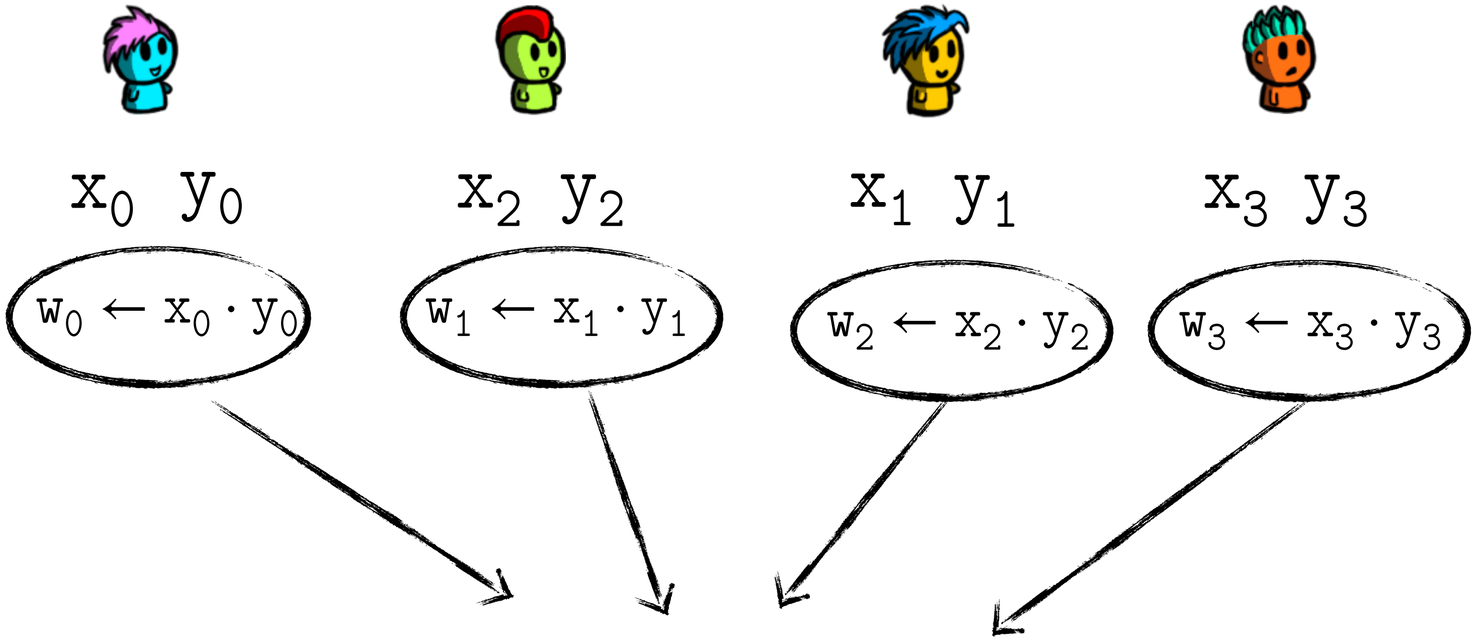



x_0 y_0 x_2 y_2 x_1 y_1 x_3 y_3

$w_0 \leftarrow x_0 \cdot y_0$ $w_1 \leftarrow x_1 \cdot y_1$ $w_2 \leftarrow x_2 \cdot y_2$ $w_3 \leftarrow x_3 \cdot y_3$

Multiplication in Shamir Secret Shares





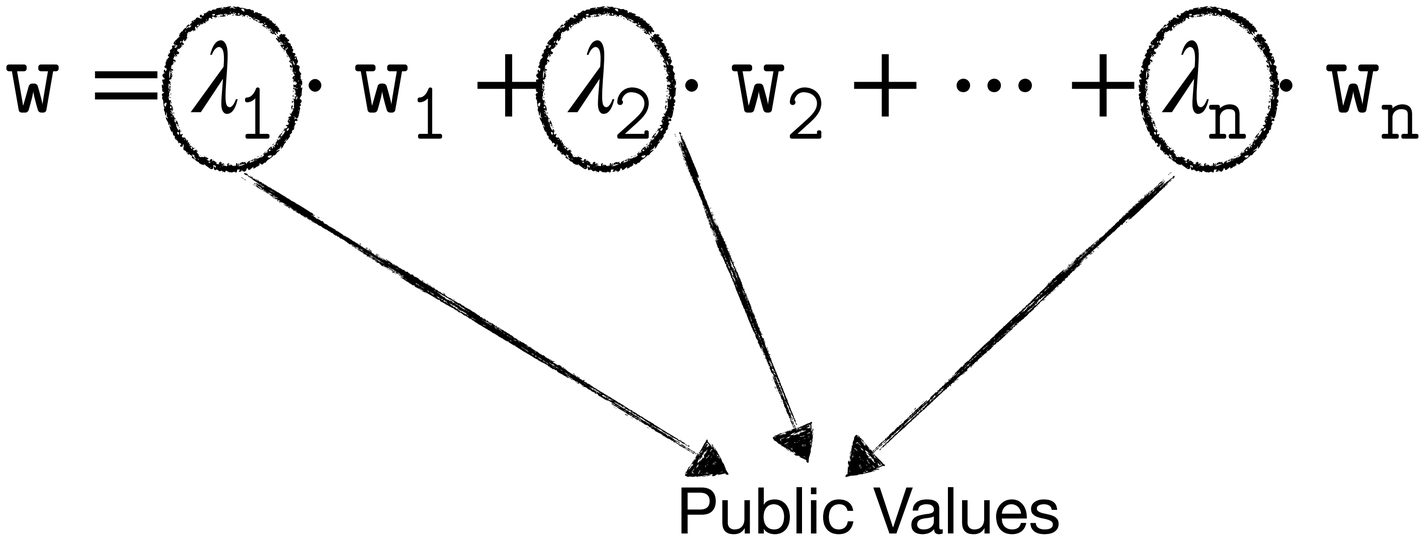
Is that good enough?

The degree of that polynomial is $2t$ instead of t.

Degree Reduction

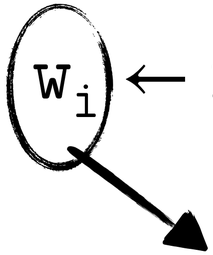
Lagrange Interpolation:

We want to privately compute



Degree Reduction

Each party i locally computes $w_i \leftarrow x_i \cdot y_i$



Shares of $x \cdot y$ in a polynomial of degree 2t

Each party shares w_i among the n parties with the correct t

$w_{i,j} \rightarrow$ share of w_i held by party j

$$z_j \leftarrow \lambda_1 \cdot w_{1,j} + \lambda_2 \cdot w_{2,j} + \cdots + \lambda_n \cdot w_{n,j}$$

Degree Reduction

What is the minimum number of parties needed to execute this protocol?

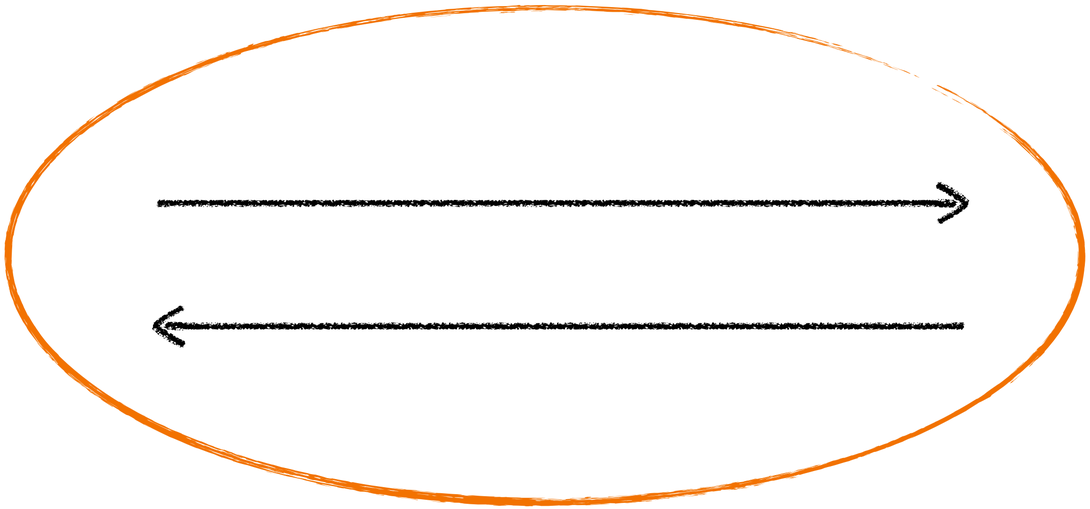
We must have enough parties to reconstruct the intercept of the polynomial with degree $2t$

Oblivious Transfer

Receiver

b

x_b

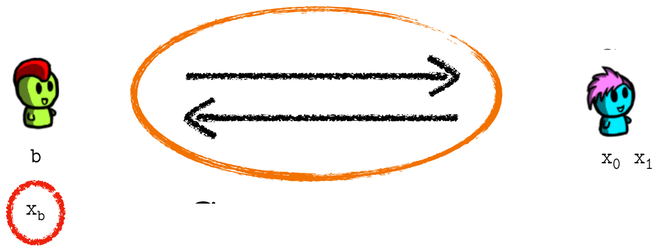


Secure Multiparty Computation

Sender

x_0 x_1

Oblivious Transfer



What is a simple semi-honest protocol to execute Oblivious Transfer?

Bob samples a *public key* and *private key* pair pk, sk

Bob samples a *random key* rk

If $b = 1$ (rk, pk)

If $b = 0$ (pk, rk)

Alice receives (k_0, k_1)

Alice sends back $(Enc_{k_0}(x_0), Enc_{k_1}(x_1))$

Why does this work?

Why does this only work in a *semi-honest* setting?

Rather than sampling a random key, Bob can sample a public private key pair

Oblivious Transfer

Receiver

b

$k_b \leftarrow (pk, sk)$

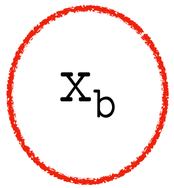
$k_{1-b} \leftarrow rk$

(k_0 , k_1)

$(Enc_{k_0}(x_0) , Enc_{k_1}(x_1))$

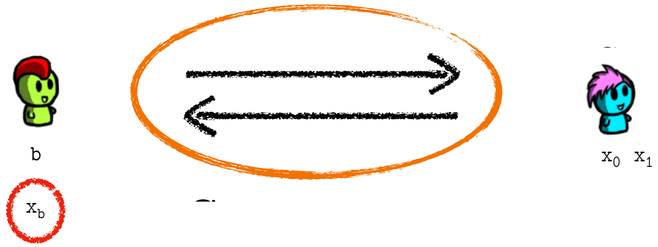
Sender

$x_0 \quad x_1$



x_b

Oblivious Transfer



How can we make the above semi-honest protocol work under a malicious setting?

The problem with the protocol is that Bob is free to choose both keys as per his choice.

Malicious Bob would pick both keys to be public privat key pairs

Can we somehow tie Bob’s hands so that he *does not* do what he pleases to do?

Oblivious Transfer

Receiver

Sender

b

c

$x_0 \quad x_1$

$(k_0 \ , \ k_1)$

$k_b \leftarrow (pk, sk)$

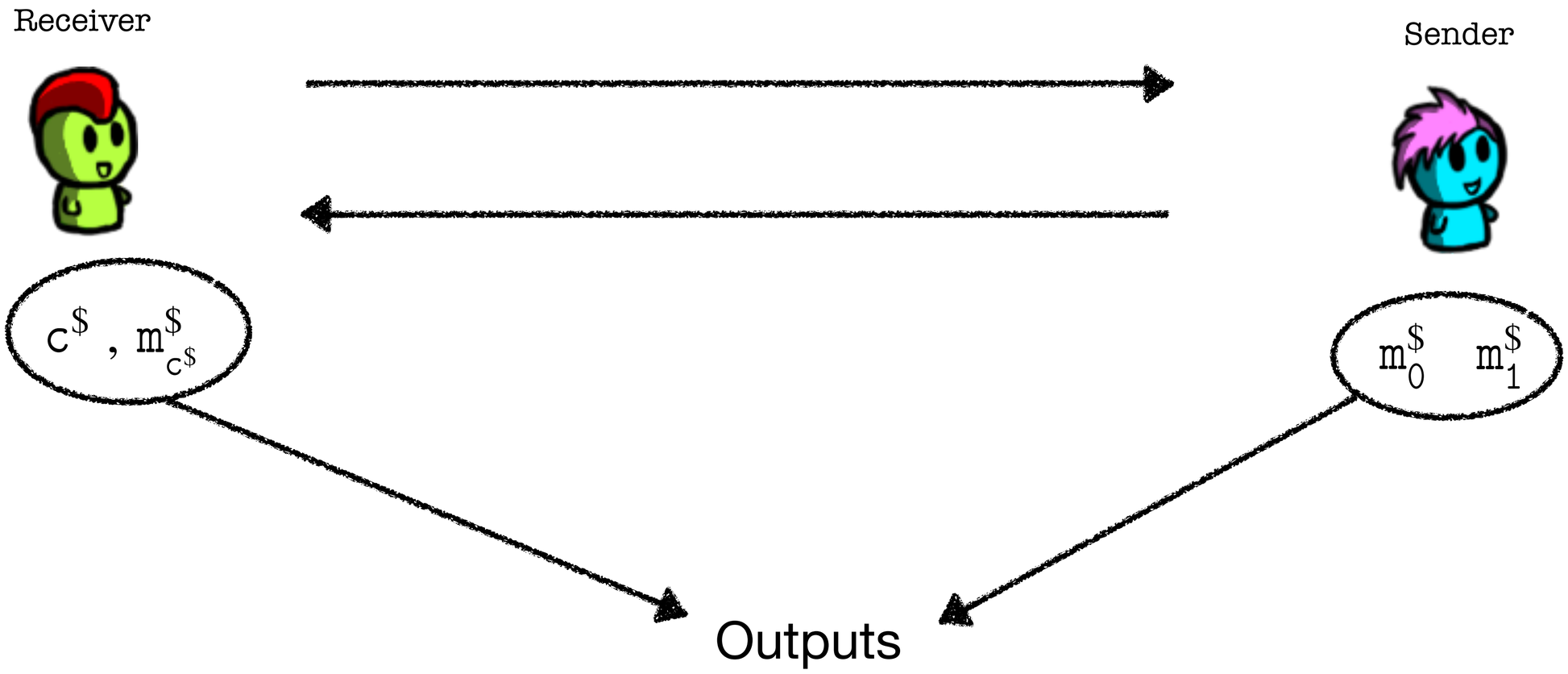
$k_{1-b} \leftarrow c/k_b$

$\text{if}(k_0 \cdot k_1 == c) \quad (\text{Enc}_{k_0}(x_0) \ , \ \text{Enc}_{k_1}(x_1))$

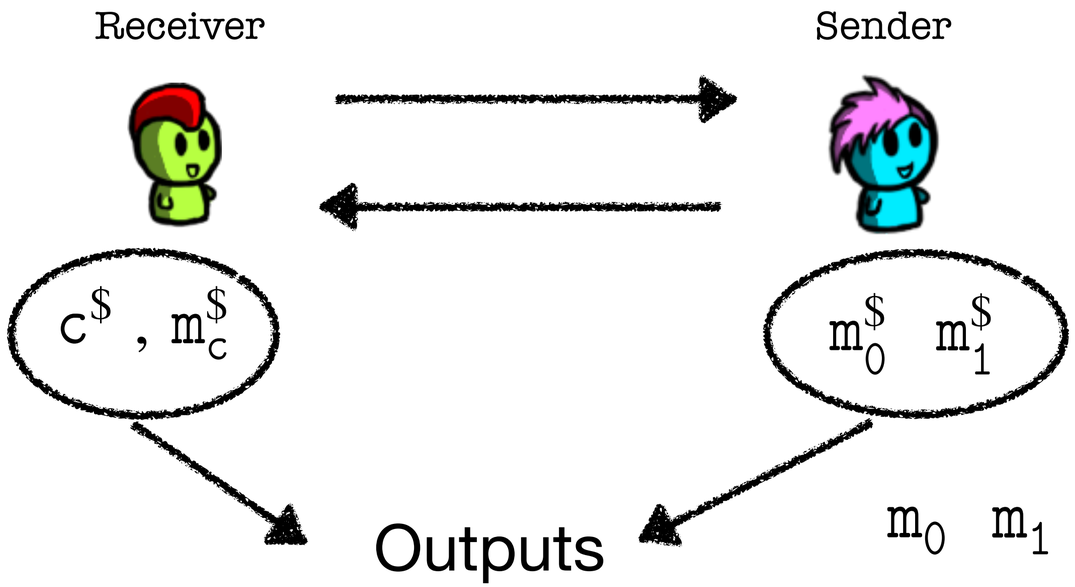
$\text{if}(k_0 \cdot k_1 != c) \quad \text{Abort!}$

x_b

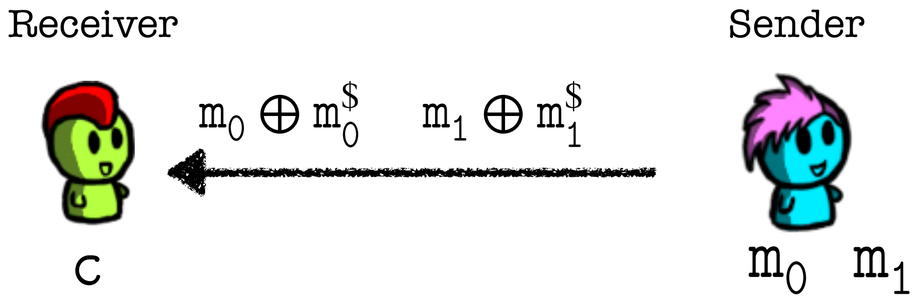
Random Oblivious Transfer (OT)



Random Oblivious Transfer (ROT)

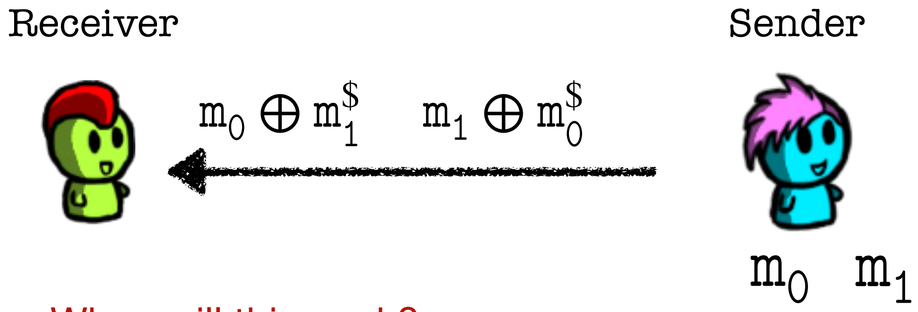


What is the point of this?



When will this work?

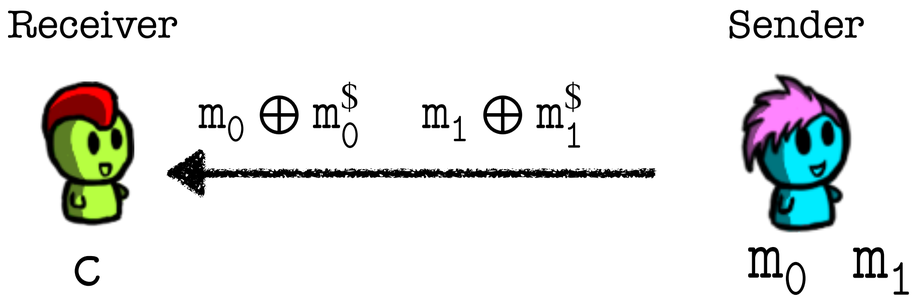
If $c = c^{\$}$



When will this work?

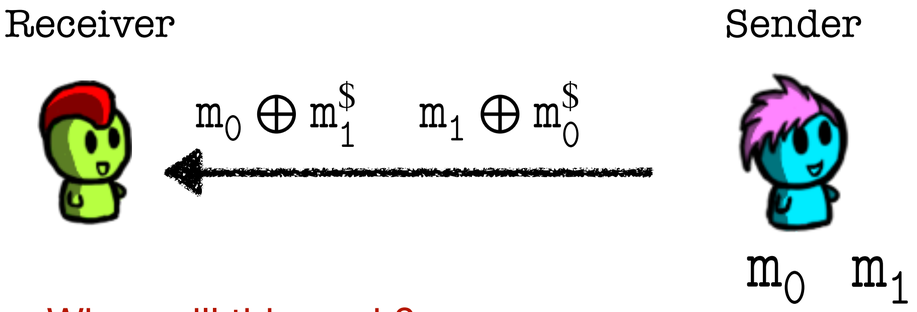
If $c = 1 - c^{\$}$

Random Oblivious Transfer (ROT)



When will this work?

If $c = c^{\$}$



When will this work?

If $c = 1 - c^{\$}$

How can Alice give this information to Bob?

Alice just tells it!

OT Extension

Can we get multiple OTs for the cost of a few OTs?

IKNP

$s = 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0$

r

1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

1	1	0	1	0	0	1	1
0	0	1	1	0	0	0	1
1	0	1	0	1	1	0	1
0	1	0	0	0	1	1	0
0	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1
1	1	1	1	0	1	0	1
0	0	1	0	1	1	0	0

T

1	0	0	1	0	1	0	1
0	0	1	1	0	0	0	1
1	1	1	0	1	0	1	1
0	1	0	0	0	1	1	0
0	0	1	0	1	1	1	1
1	0	0	1	1	0	0	1
1	0	1	1	0	0	1	1
0	0	1	0	1	1	0	0

T’

0	1	1	0	1	0	1	0
0	0	1	1	0	0	0	1
0	0	0	1	0	1	0	0
0	1	0	0	0	1	1	0
1	1	0	1	0	0	0	0
1	0	0	1	1	0	0	1
0	1	0	0	1	1	0	0
0	0	1	0	1	1	0	0

IKNP

Bob has input $\mathtt{r}$ Extend to matrix and secret share as $\mathtt{T}$ and $\mathtt{T}'$

Alice chooses a random string $\mathtt{s}$

OT for each column: Alice obtains matrix $\mathtt{Q}$

Whenever $\mathtt{r}_i = 0$ Alice row = Bob row

$\mathtt{r}_i = 1$ Alice row = Bob row $\oplus\, \mathtt{s}$

$$\mathtt{q}_i = \begin{cases} \mathtt{t}_i & \mathtt{r}_i = 0 \\ \mathtt{t}_i \oplus \mathtt{s} & \mathtt{r}_i = 1 \end{cases}$$

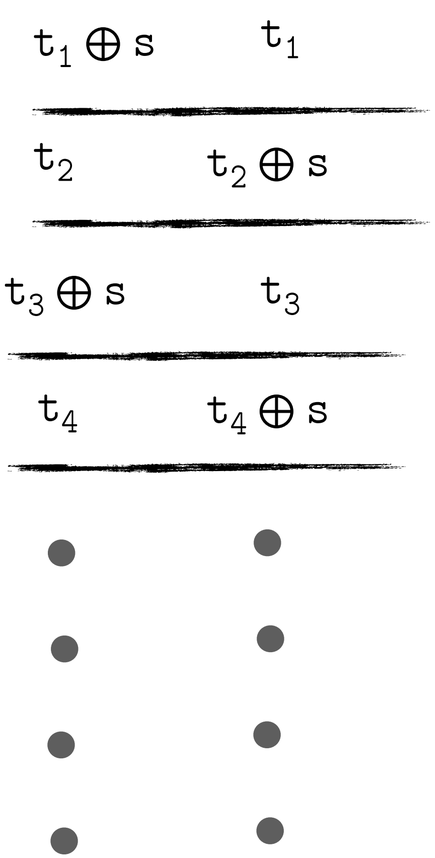
IKNP

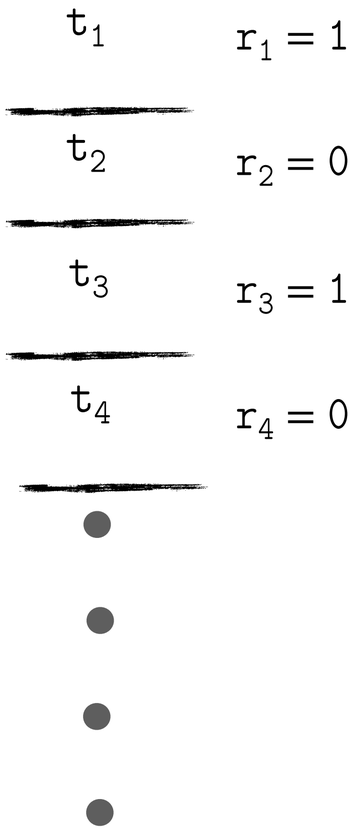
q_1	$q_1 \oplus s$
q_2	$q_2 \oplus s$
q_3	$q_3 \oplus s$
q_4	$q_4 \oplus s$
⋮	⋮

t_1
t_2
t_3
t_4
⋮

$$q_i = \begin{cases} t_i & r_i = 0 \\ t_i \oplus s & r_i = 1 \end{cases}$$

IKNP

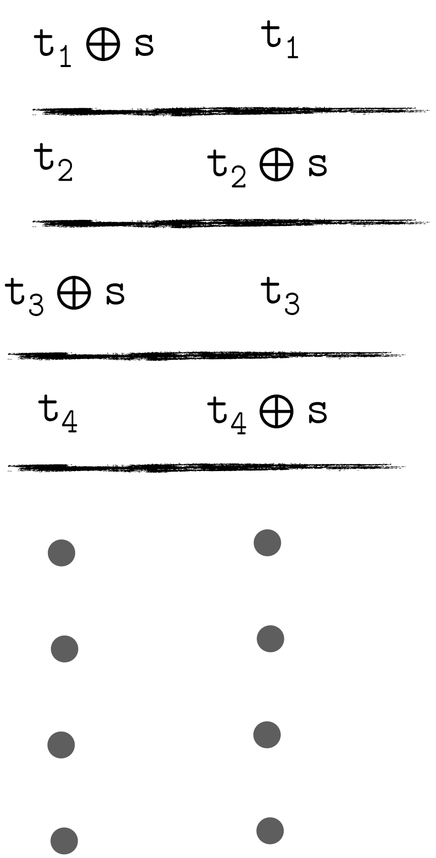


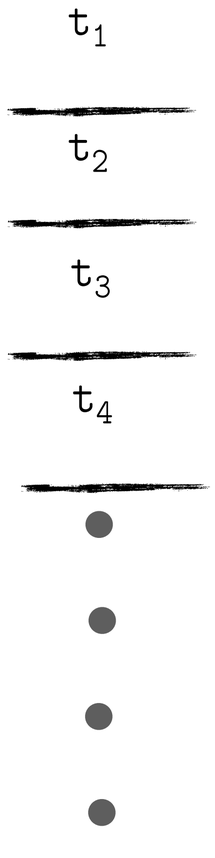


$$q_i = \begin{cases} t_i & r_i = 0 \\ t_i \oplus s & r_i = 1 \end{cases}$$

IKNP

$$q_i = \begin{cases} t_i & r_i = 0 \\ t_i \oplus s & r_i = 1 \end{cases}$$





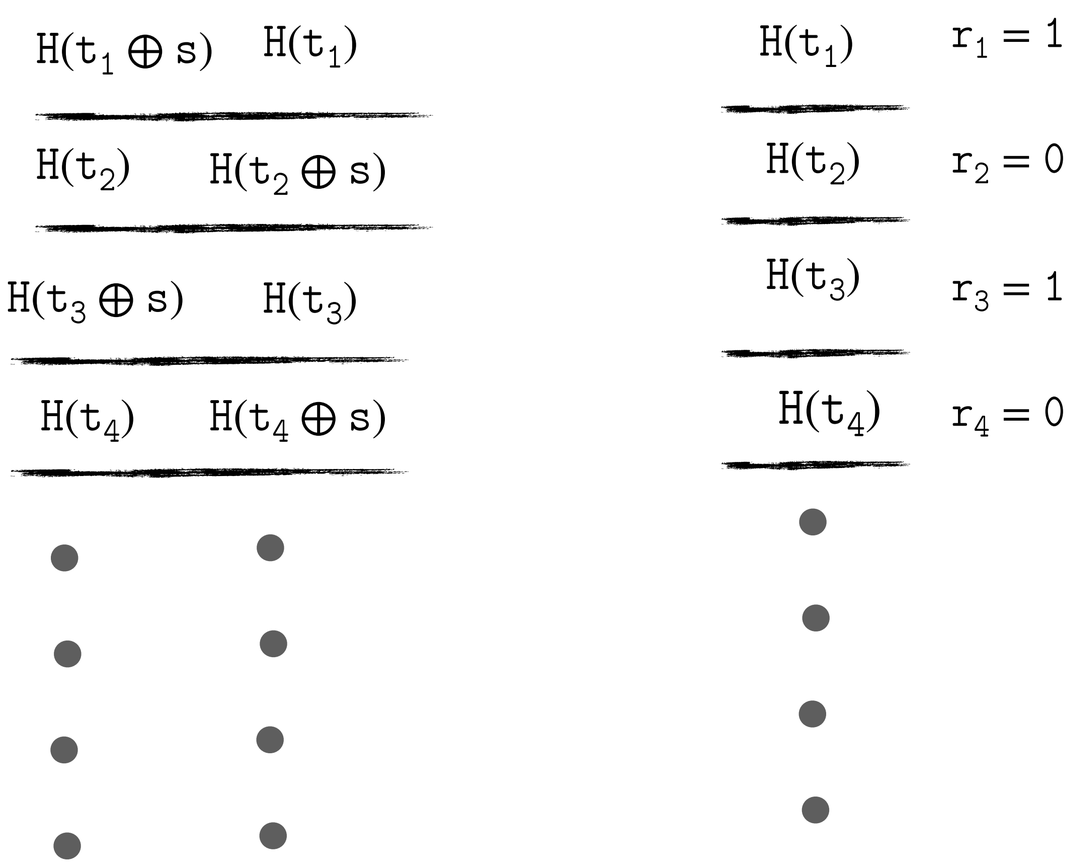
For every i Bob gets t_i

Alice can compute q_i $q_i \oplus s$

Bob knows exactly one of these two values

Almost an OT
Reusing s leads to corellations

IKNP



$$q_i = \begin{cases} t_i & r_i = 0 \\ t_i \oplus s & r_i = 1 \end{cases}$$

For every i Bob gets t_i

Alice can compute q_i $q_i \oplus s$

Bob knows exactly one of these two values

Almost an OT

Reusing s leads to corellations

IKNP

Does this work in a malicious setting?

Tossing Coin Over Phone (Blum's Coin Toss)

Bits

Nonce

$H_{bob} \leftarrow H\,(a \| r_A)$

$H_{Alice} \leftarrow (b \| r_B)$

$a\ ,\ r_A$

$b\ ,\ r_B$

$H_{Alice} =^{?} (b \| r_B)$

$H_{bob} =^{?} H\,(a \| r_A)$

$a \oplus b$

$a \oplus b$

Private Set Intersection

Sender

$\{s_1, s_2, \ldots, s_m\}$

Strings held by the *Sender*

Each party has a set of elements

Goal of the receiver is to learn which elements the two parties have in common

Both parties can learn the size of each other's sets

Sender learns nothing

Receiver

$\{r_1, r_2, \ldots, r_m\}$

Strings held by the *Receiver*

Private Set Intersection

$S = \{s_1, s_2, \ldots, s_m\}$

$R = \{r_1, r_2, \ldots, r_m\}$

$\{H(s_1), H(s_2), \ldots, H(s_m)\}$

Does this work?

Private Set Intersection

$S = \{s_1, s_2, \ldots, s_m\}$

$R = \{r_1, r_2, \ldots, r_n\}$

Hashes into a group

$\{H_p(s_1), H_p(s_2), \ldots, H_p(s_m)\}$

$\{a \cdot H_p(r_1), a \cdot H_p(r_2), \ldots, a \cdot H_p(r_n)\}$

$\{H_p(r_1), H_p(r_2), \ldots, H_p(r_n)\}$

$\{H(b \cdot a \cdot H_p(r_1)), H(b \cdot a \cdot H_p(r_2)), \ldots, H(b \cdot a \cdot H_p(r_n))\}$

$\{b \cdot H_p(s_1), b \cdot H_p(s_2), \ldots, b \cdot H_p(s_n)\}$

Why do we have them hash into a group? We can learn the ratio of the hashes if not!

What protocol does it resemble? DH Protocol

Garbled Circuits

One of the first methods to do MPC

Garbler

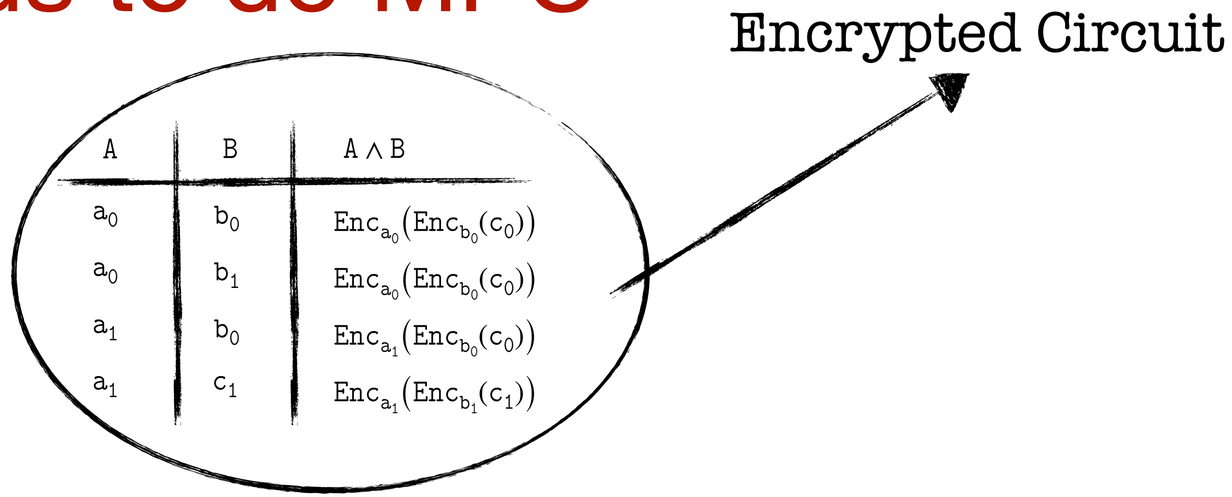
Evaluator

Garbled Circuits

One of the first methods to do MPC

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A ∧ B
a_0	b_0	c_0
a_0	b_1	c_0
a_1	b_0	c_0
a_1	c_1	c_1



Garbler

Let’s say we can somehow manage to give Alice the correct keys

What is still wrong with this?

The ciphertext that was decrypted will reveal the input.

Evaluator

Garbled Circuits

One of the first methods to do MPC

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A ∧ B
a_0	b_0	c_0
a_0	b_1	c_0
a_1	b_0	c_0
a_1	c_1	c_1

A	B	A ∧ B
a_1	b_0	$Enc_{a_1}(Enc_{b_0}(c_0))$
a_0	b_0	$Enc_{a_0}(Enc_{b_0}(c_0))$
a_1	c_1	$Enc_{a_1}(Enc_{b_1}(c_1))$
a_0	b_1	$Enc_{a_0}(Enc_{b_0}(c_0))$

Permuted
Encrypted Circuit

Garbler

How can we send the *correct keys* to Alice!?

Start with Bob: How can we get Alice Bob's key?

Simply tell the key

Evaluator

Garbled Circuits

One of the first methods to do MPC

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A ∧ B
a_0	b_0	c_0
a_0	b_1	c_0
a_1	b_0	c_0
a_1	c_1	c_1

A	B	A ∧ B
a_1	b_0	$Enc_{a_1}(Enc_{b_0}(c_0))$
a_0	b_0	$Enc_{a_0}(Enc_{b_0}(c_0))$
a_1	c_1	$Enc_{a_1}(Enc_{b_1}(c_1))$
a_0	b_1	$Enc_{a_0}(Enc_{b_0}(c_0))$

Encrypted Circuit

How can we send the *correct keys* to Alice!?

How about Alice's Key? Oblivious Transfer

Garbler

Evaluator

Garbled Circuits

One of the first methods to do MPC

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A ∧ B
a_0	b_0	c_0
a_0	b_1	c_0
a_1	b_0	c_0
a_1	c_1	c_1

A	B	A ∧ B
a_1	b_0	$Enc_{a_1}(Enc_{b_0}(c_0))$
a_0	b_0	$Enc_{a_0}(Enc_{b_0}(c_0))$
a_1	c_1	$Enc_{a_1}(Enc_{b_1}(c_1))$
a_0	b_1	$Enc_{a_0}(Enc_{b_0}(c_0))$

Encrypted Circuit

Garbler

What kind of encryption algorithm do we need? *(what properties?)*

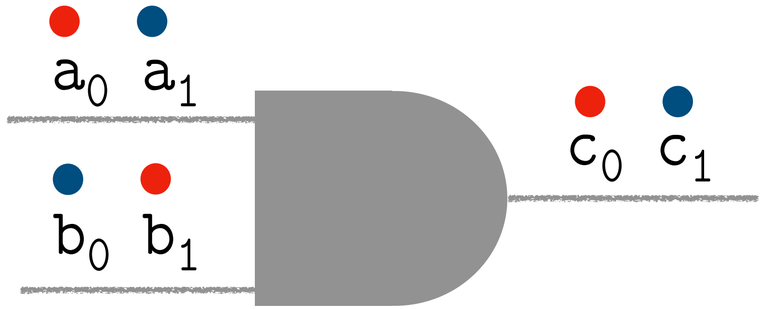
We need to know if the decryption was successful or not?.

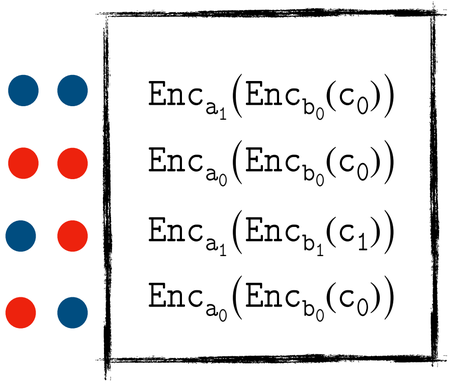
Evaluator

Garbled Circuits: Optimization

Point and Permute Technique

A	B	A ∧ B
a_0	b_0	c_0
a_0	b_1	c_0
a_1	b_0	c_0
a_1	c_1	c_1





Garbler

How do these colors help us?

Bob can simply tell Alice exactly which row to decrypt

Bob can use one-time-pads to do the encryption

Why?

Evaluator

Garbled Circuits: Optimization

GRR3 (Garbled Row Reduction)

A	B	A ∧ B
a_0	b_0	c_0
a_0	b_1	c_0
a_1	b_0	c_0
a_1	c_1	c_1

How can we send one less cipher text to the evaluator?

The Garbler can choose the wire labels such that one of the ciphers is: 0^n

Garbler

Evaluator

Admistrivia

Quiz 1 on Thursday at RM101

Will be around 40 minutes, followed by a discussion on the answers

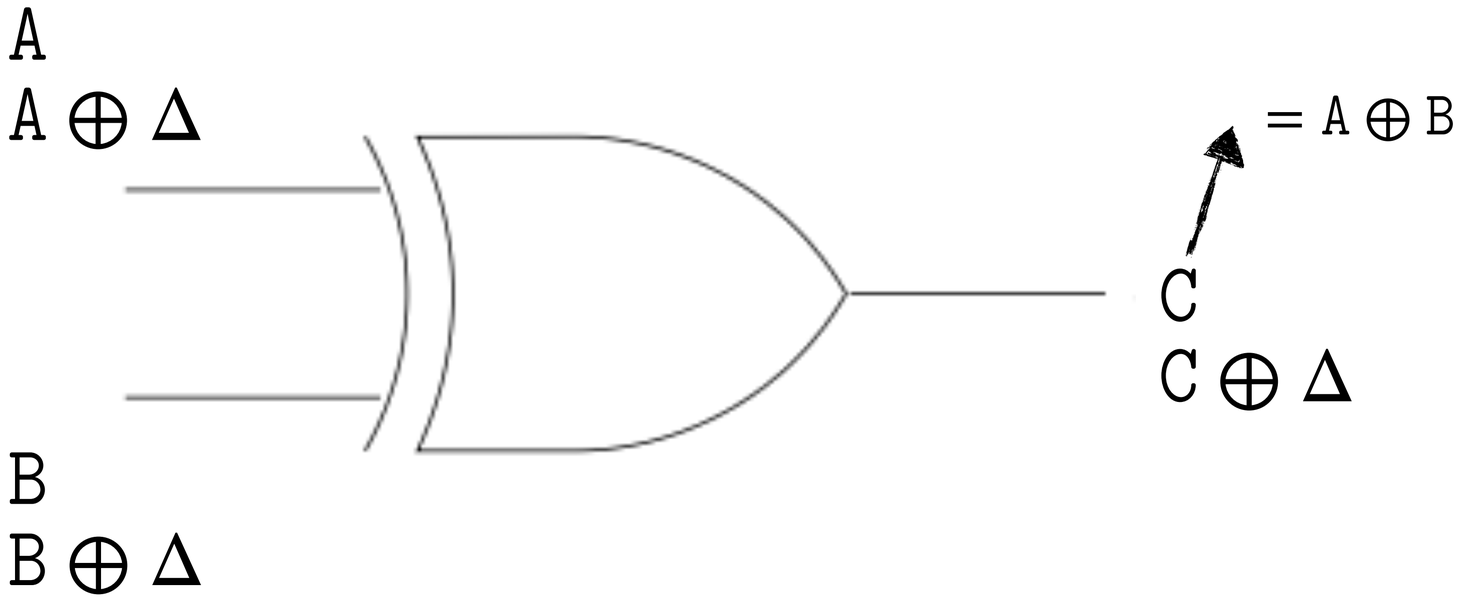
Assignment 1 will be out on Thursday, 28th August

Deadline will be: 10th September, 2025 EOD

Garbled Circuits: Optimization

Free XOR

A	B	A ∧ B
a_0	b_0	c_0
a_0	b_1	c_0
a_1	b_0	c_0
a_1	c_1	c_1



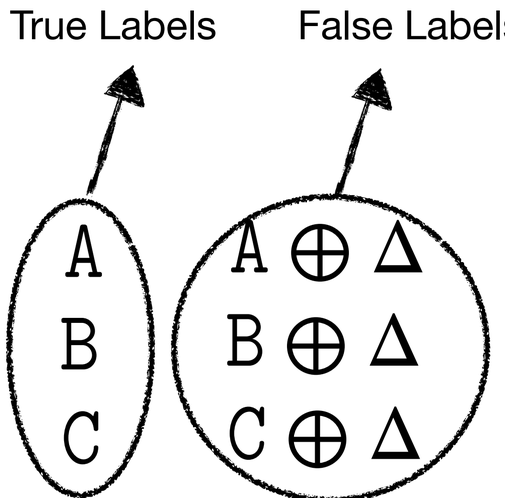
Garbler

Choose an offset Δ

Set one of the input wire label as

Set the other input wire label as

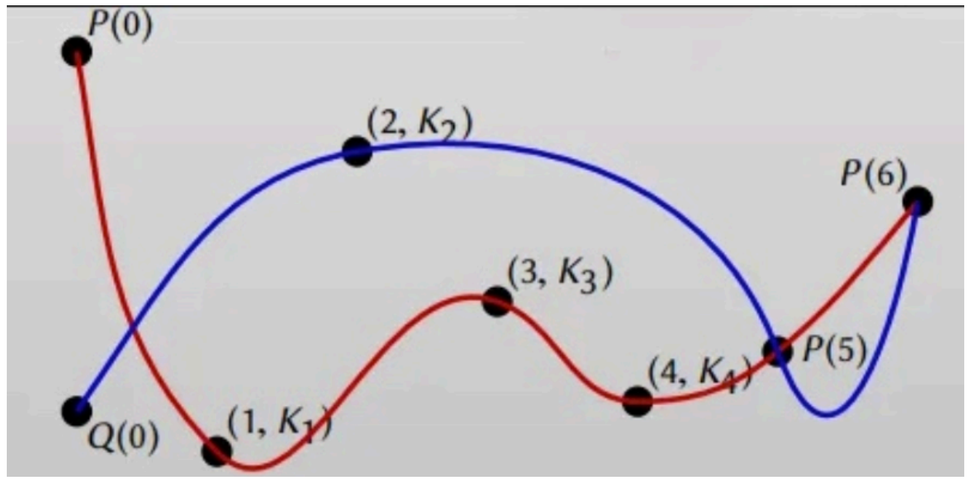
Set the output wire labels as



Evaluator

Garbled Circuits: Optimization

GRR2



Garbler computes

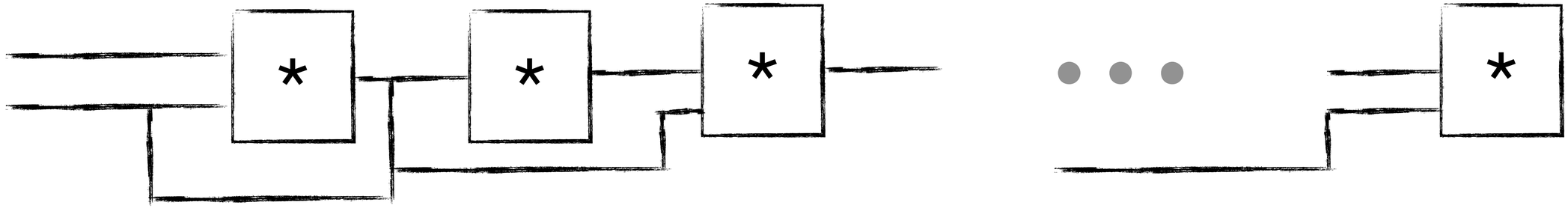
$$K_1 = \mathrm{Dec}_{a_0,b_0}(0^n)$$
$$K_2 = \mathrm{Dec}_{a_0,b_1}(0^n)$$
$$K_3 = \mathrm{Dec}_{a_1,b_0}(0^n)$$
$$K_4 = \mathrm{Dec}_{a_1,b_1}(0^n)$$

Garbler

P(0)	Is the first label
Q(0)	Is the second label

Evaluator

Garbled Circuits vs Secret Sharing



Properties of GC

How many Oblivious Transfers are needed? # input wires of the evaluater

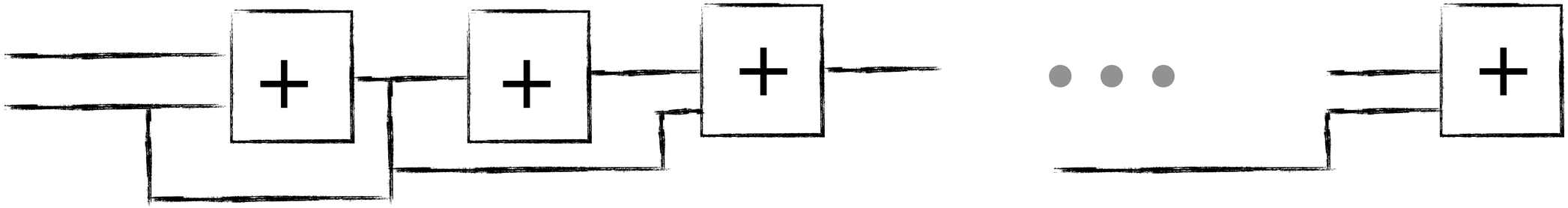
How many ciphertexts are needed? # gates

What is the round complexity? $\mathcal{O}(1)$

Properties of Secret Sharing Mechanisms?

What is the round complexity? $\mathcal{O}$ (# gates)

Garbled Circuits vs Secret Sharing



How about this circuit?

Properties of GC

How many Oblivious Transfers are needed?

How many ciphertexts are needed?

What is the round complexity?

Properties of Secret Sharing Mechanisms?

What is the round complexity?

Garbled Circuits vs Secret Sharing

It seems like in certain cases Garbled Circuits is useful

It seems like in certain cases Secret Sharing is useful

What do we do then?

Types of Secret Sharing

Broadly Three Types

Additive

Boolean

Yao

(Garbled Circuits)

Interpreting Garbled Circuits as Secret Shares

Yao's Garbled Circuits with Free XOR and Point and Permute Optimization

For each wire z we have keys K_1^z and K_0^z

Pick R such that: $K_1^z = K_0^z \oplus R$

K_0^z and $K_0^z \oplus z \cdot R$ Are the Yao shares of the bit z

→ Share Algorithm

set: $\mathrm{LSB}(R) = 1$

Why? $\mathrm{LSB}(K_0^z) \neq \mathrm{LSB}(K_1^z)$

Therefore, we can use point and permute!

What would be the color bits then?

Y2B: Yao to Boolean Shares

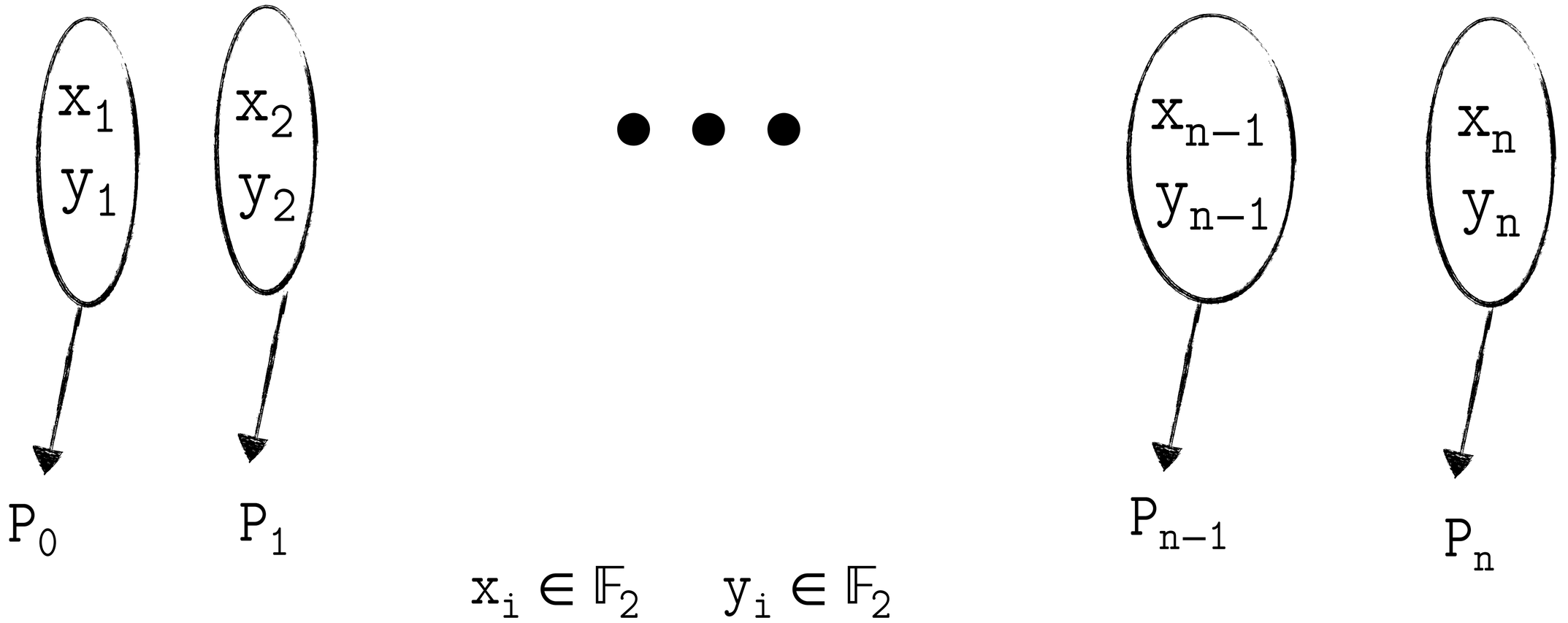
Y2B is free! Why?

Observe that the color bit, i.e., the LSB of the key is already the boolean share

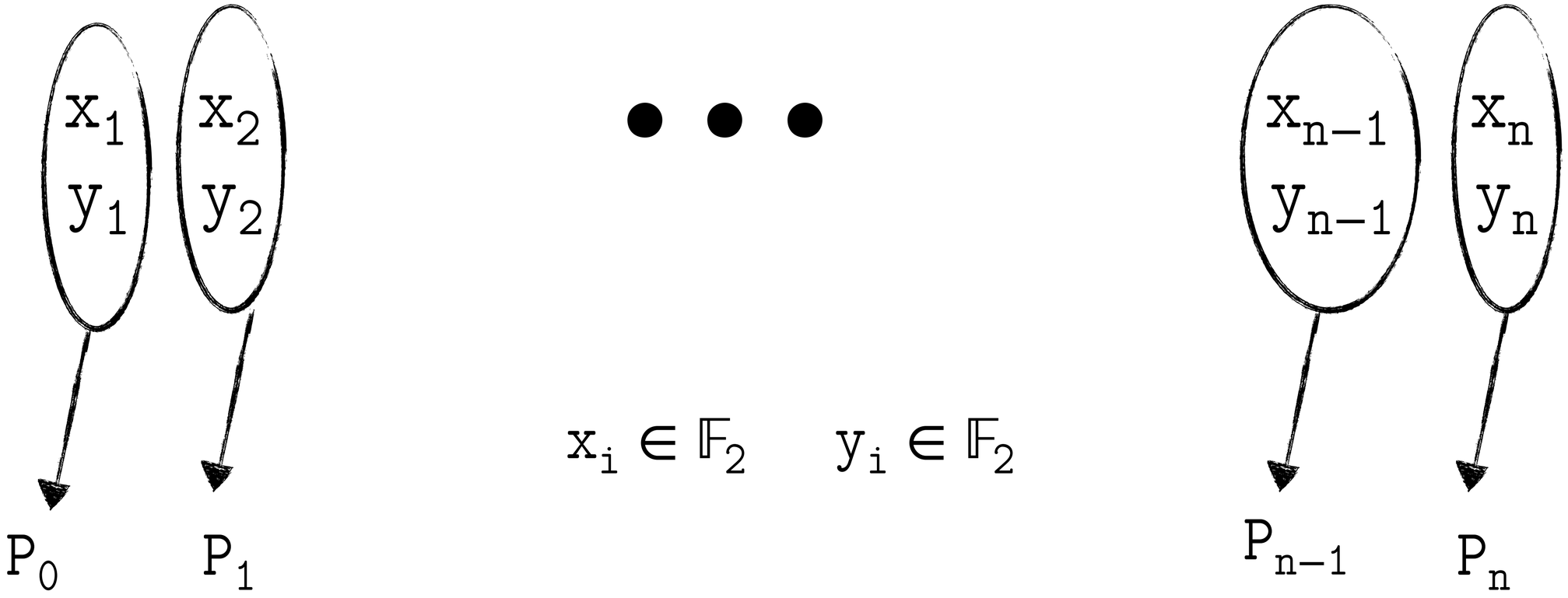
Similarly, there are ways to convert one “type” of share to another!

(We are not going into those details in this course)

GMW n-party protocol



GMW n-party protocol

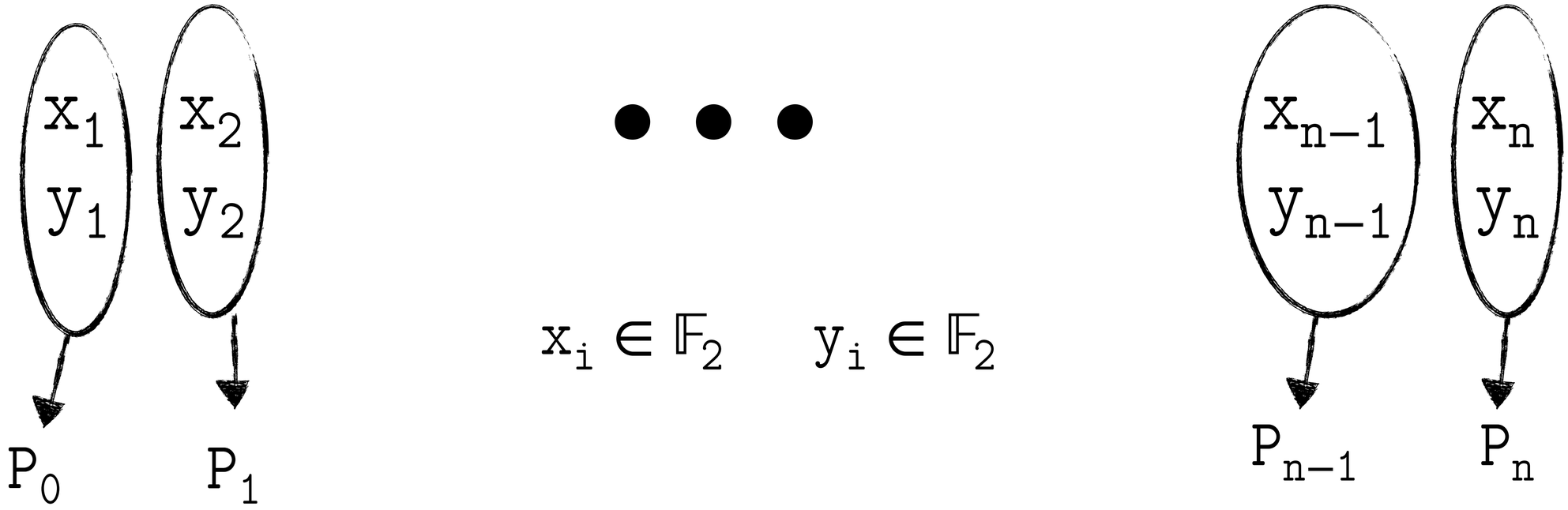


$x_i \in \mathbb{F}_2 \quad y_i \in \mathbb{F}_2$

Addition: $z_i = x_i + y_i$

There is no communication

GMW n-party protocol



$$x_i \in \mathbb{F}_2 \qquad y_i \in \mathbb{F}_2$$

$$(x_i + y_i) \cdot (x_j + y_j) = x_i \cdot y_i + x_i \cdot y_j + x_j \cdot y_i + x_j \cdot y_j$$

P_i P_j

GMW *n-party* protocol

$$(x_i + y_i) \cdot (x_j + y_j) = x_i \cdot y_i + x_i \cdot y_j + x_j \cdot y_i + x_j \cdot y_j$$

$x_i \cdot y_i$ → P_i

$x_j \cdot y_j$ → P_j

$x_i \cdot y_j + x_j \cdot y_j$ → there are four possible values

P_i Picks a random: s

$x_i \cdot y_j + x_j \cdot y_j + s$ For all possible *four* values

P_j Knows which one it wants!

GMW *n-party* protocol

$x_i \cdot y_j + x_j \cdot y_j$ → there are four possible values

P_i Picks a random: s

$x_i \cdot y_j + x_j \cdot y_j + s$ For all possible *four* values

P_j Knows which one it wants!

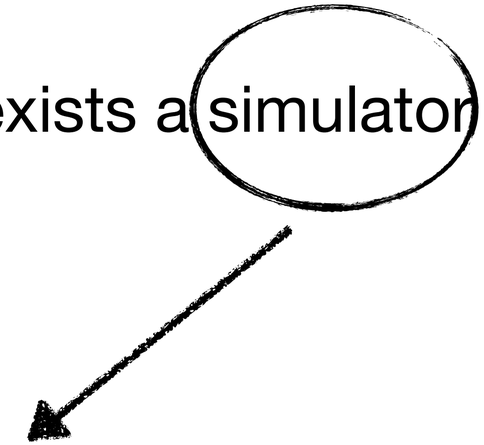
How do we do this now?

4-out-of-1 OT

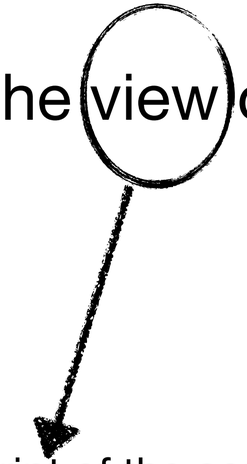
Proving Security

In the semi-honest setting:

Show that there exists a simulator that produces the view of all the parties



A probabilistic polynomial time (PPT) algorithm



The transcript of the conversation

Proving Security

How do we prove the security of Beaver triple multiplication?

Summary of Module 1

there are four possible values