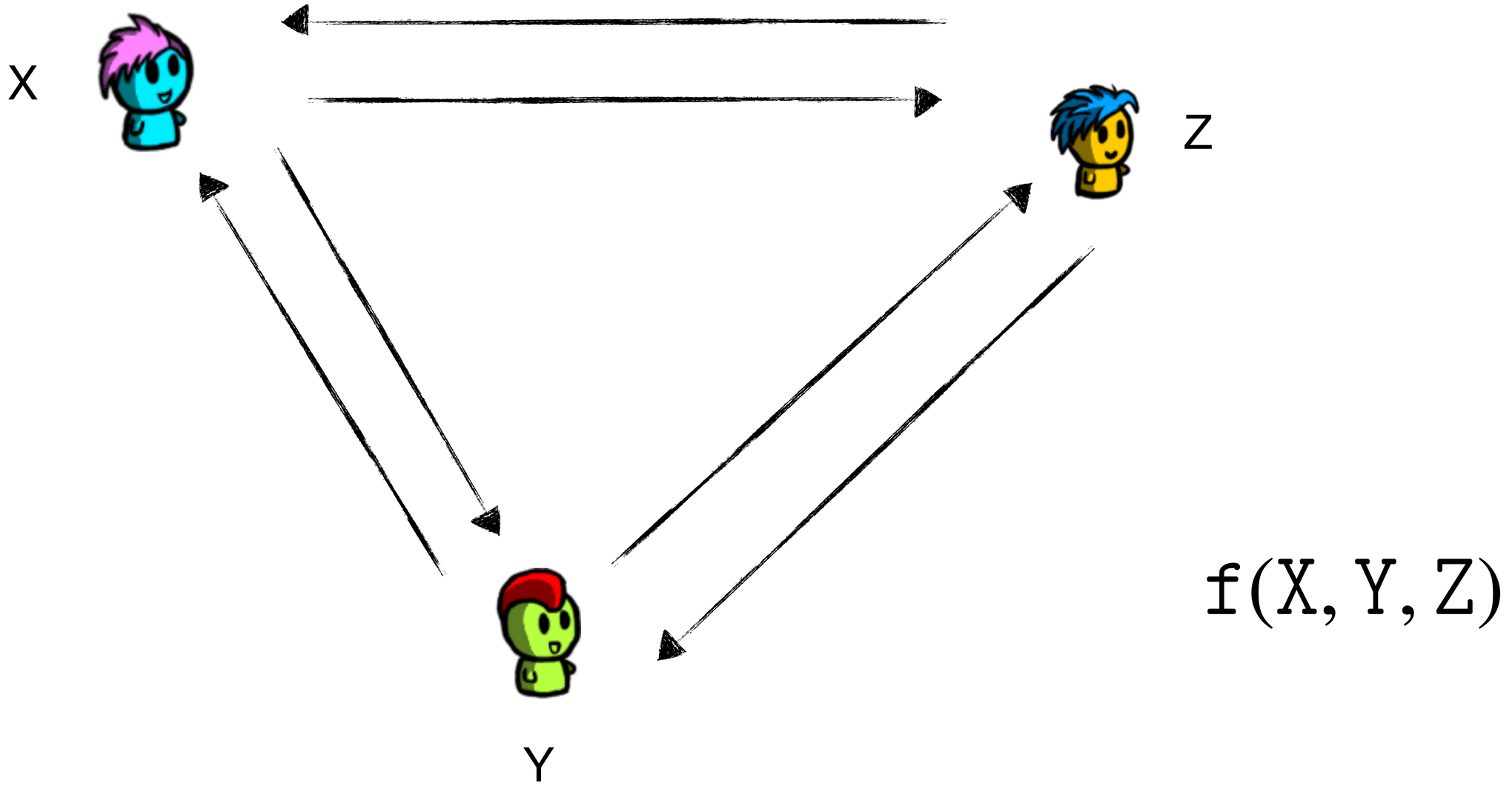
CS670: Cryptographic Techniques for Privacy Preservation

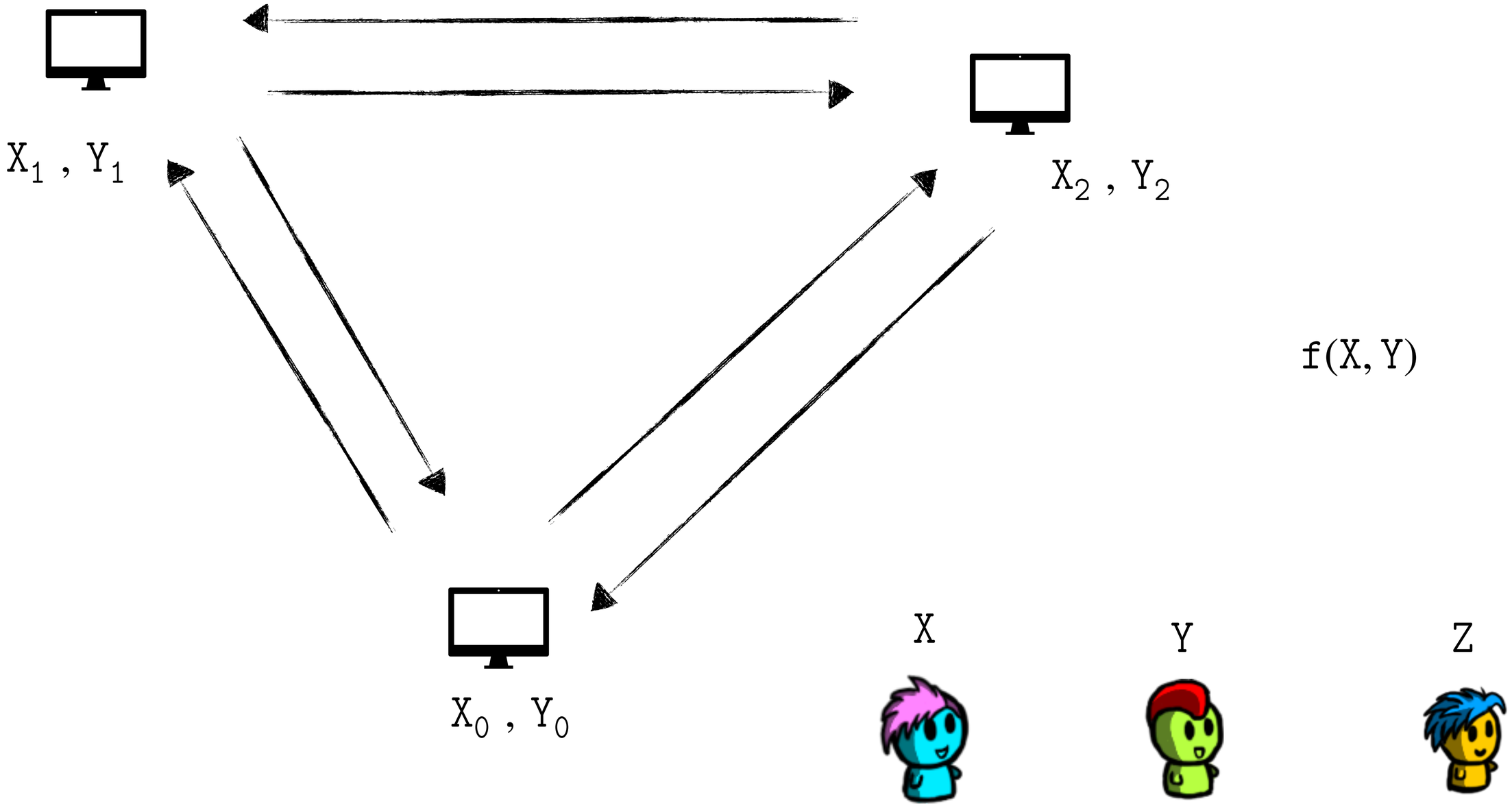
Module 1: Secure Multiparty Computation

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

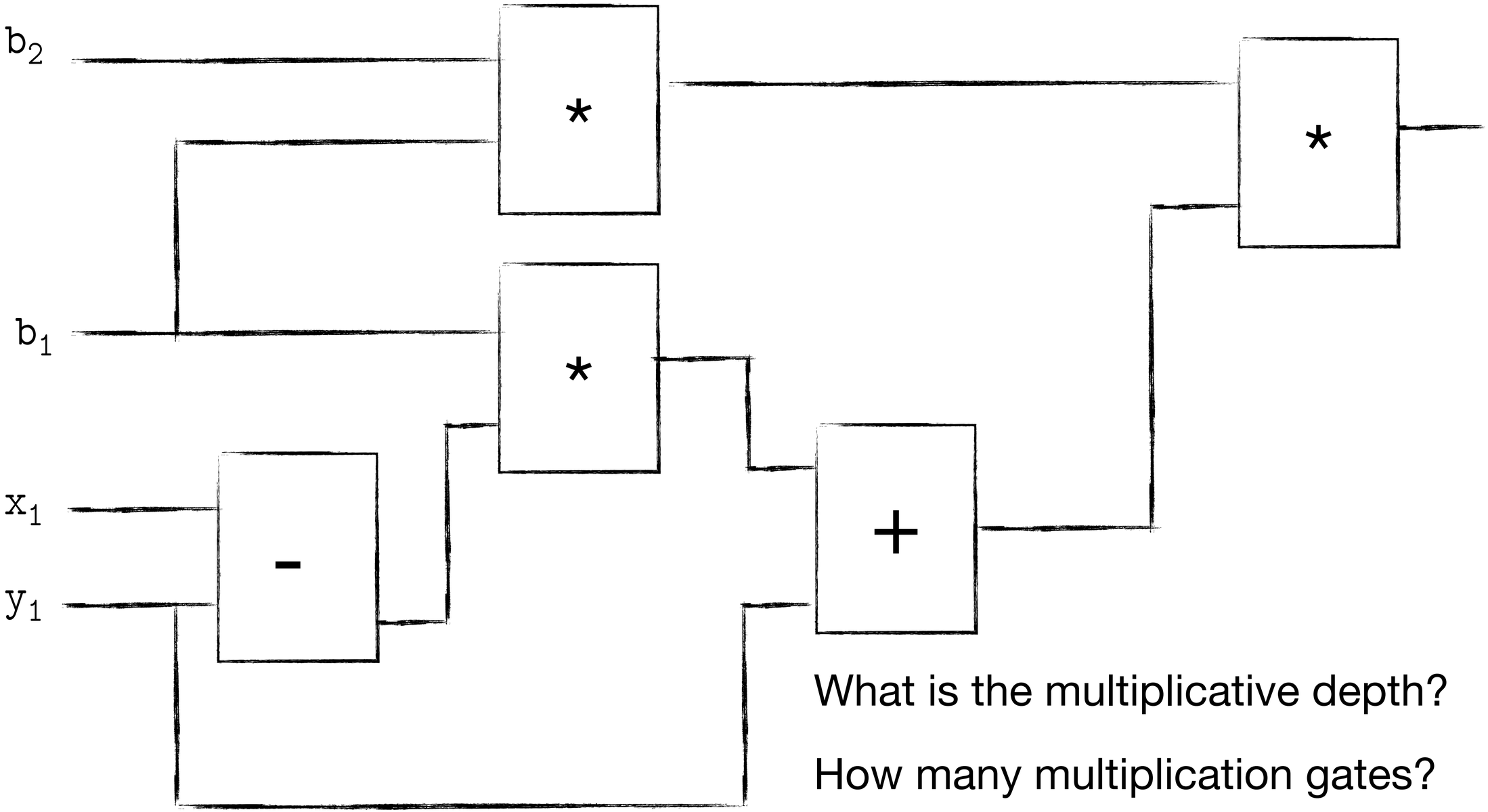
What kind of functions can the MPC protocol evaluate?

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

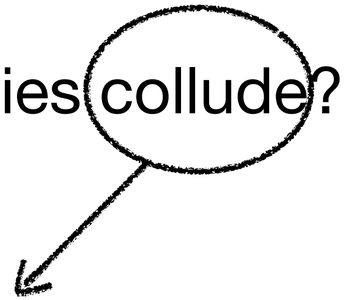
How many computational parties are needed to run the protocol?

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: $x \ \& \ y$

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

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

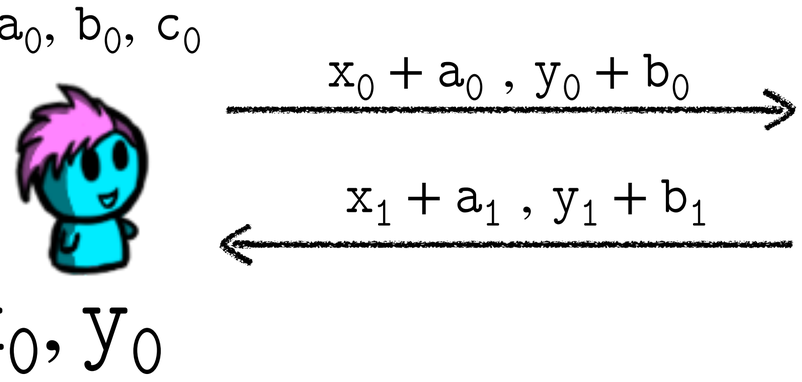
Their goal is to get additive shares of: $x \cdot y$

Secure Multiparty Computation

(On Additive Secret Shares)

Beaver's Triples:

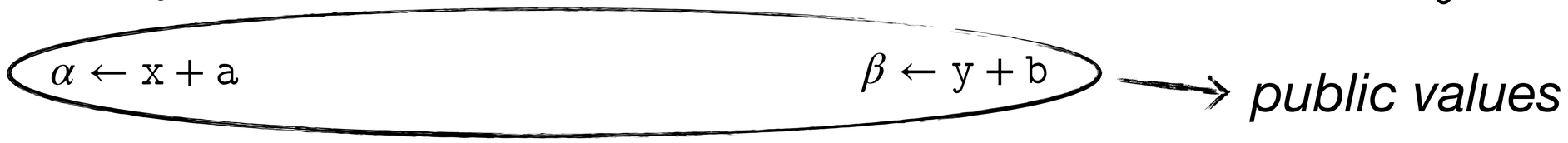
Distribute shares of random inputs (a and b) and their product c



$$x_0 + x_1 = x$$

$$y_0 + y_1 = y$$

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

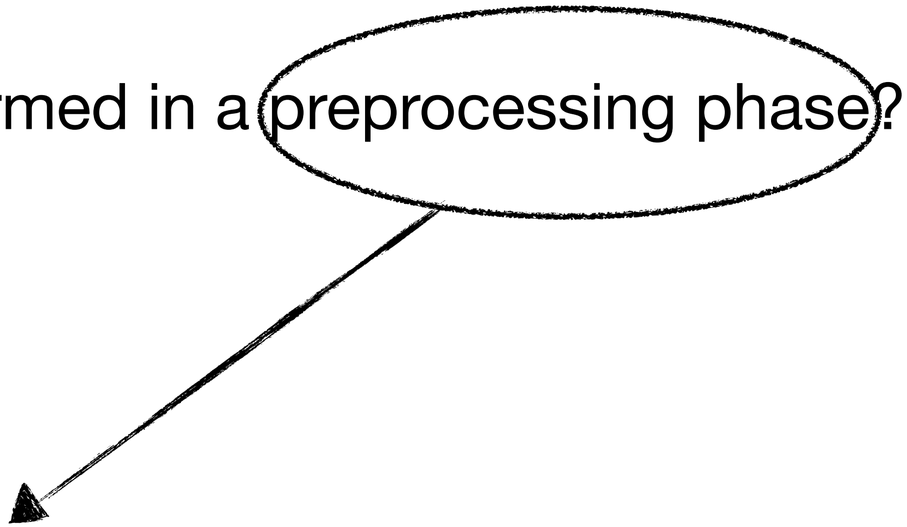


$z_0 \leftarrow \alpha \cdot y_0 - \beta \cdot a_0 + c_0$

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

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

Secure Multiparty Computation

(On Additive Secret Shares)

$x_0 \cdot y_0$

$+$

$x_0 \cdot (y_1 + Y_1)$

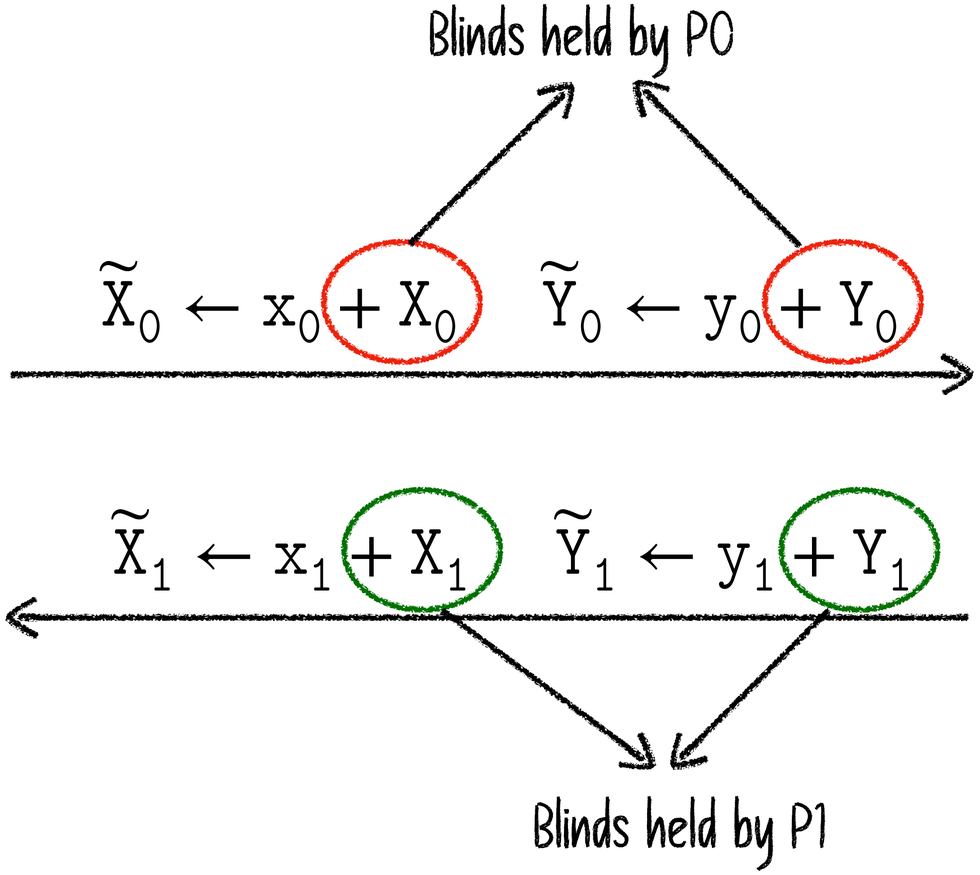
$x_1 \cdot y_1$

$+$

$x_1 \cdot (y_0 + Y_0)$

$x_0 + x_1 = x$

$y_0 + y_1 = y$

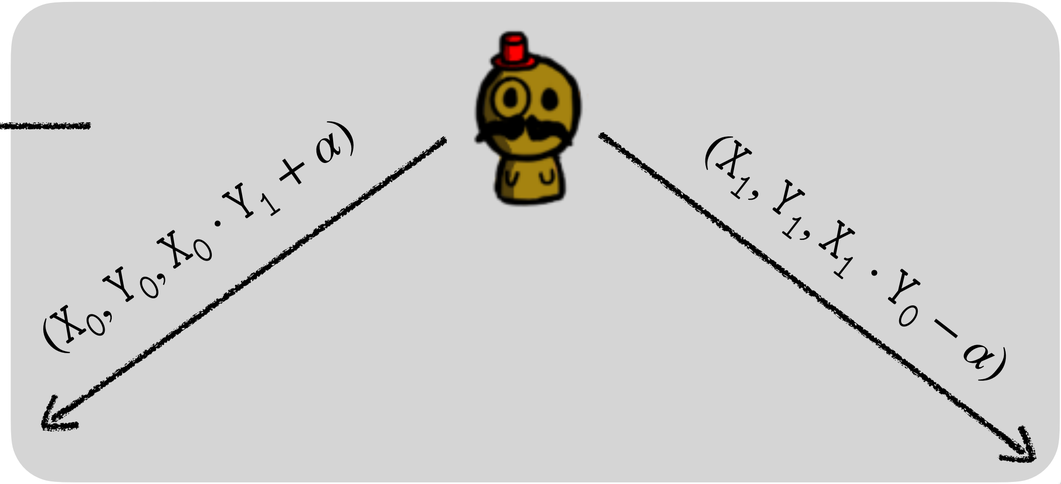


x_0 , y_0

x_1 , y_1

Secure Multiparty Computation (On Additive Secret Shares)

Preprocessing



x_0 y_0

x_1, y_1

$x_0 + x_1 = x$

$y_0 + y_1 = y$

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

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

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

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

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

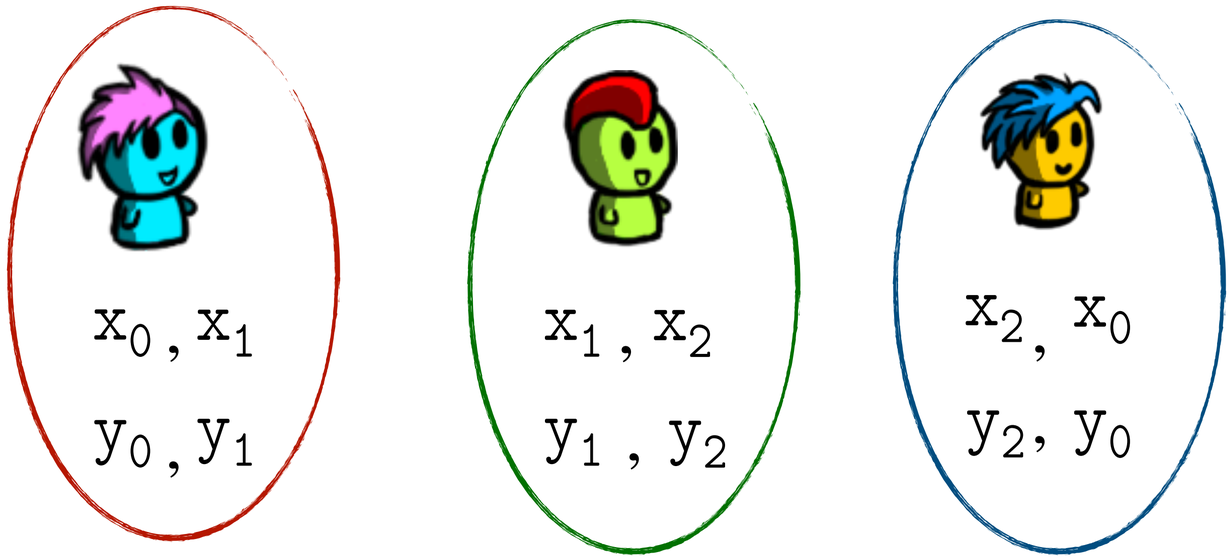
y_1, y_2

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)

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

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

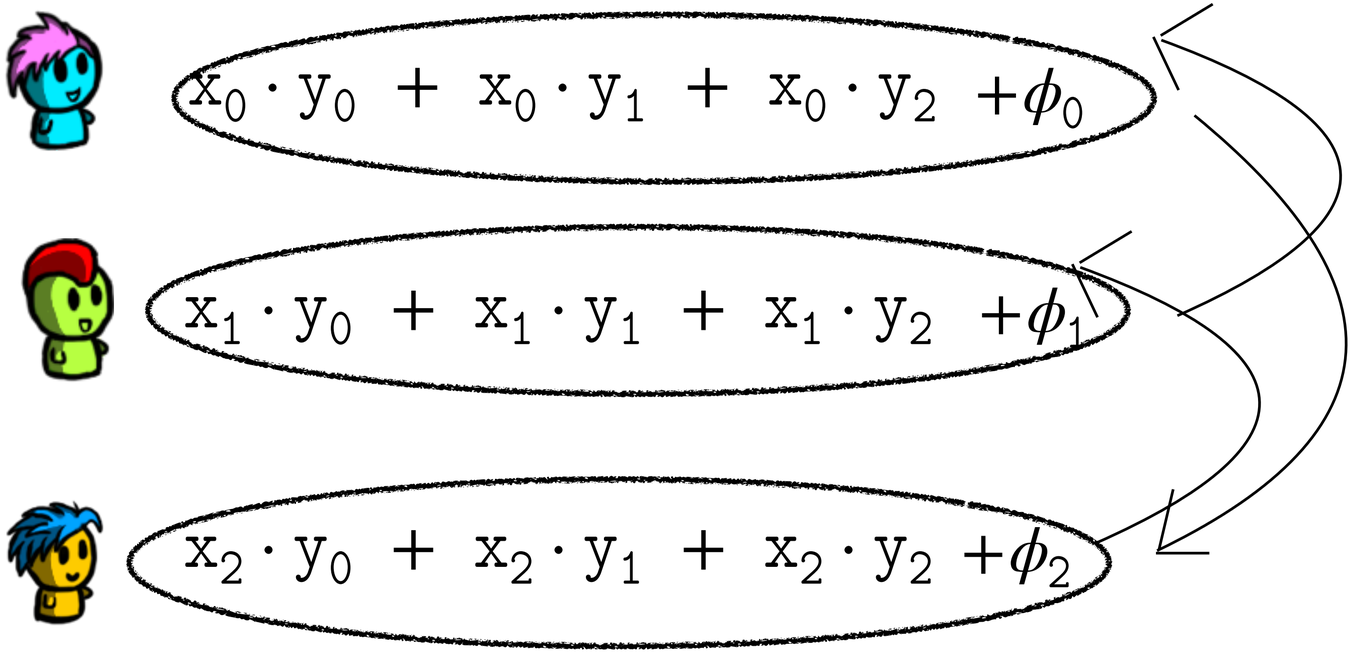
What is the problem with this?

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

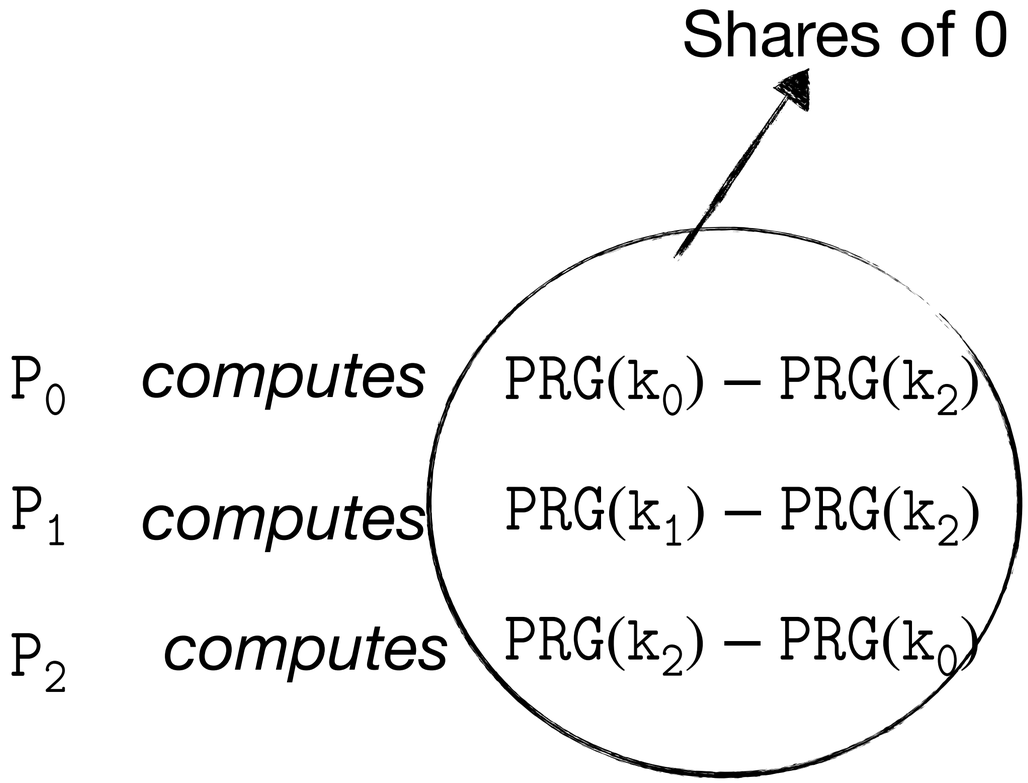
Each party i picks a random key k_i

In the preprocessing phase:

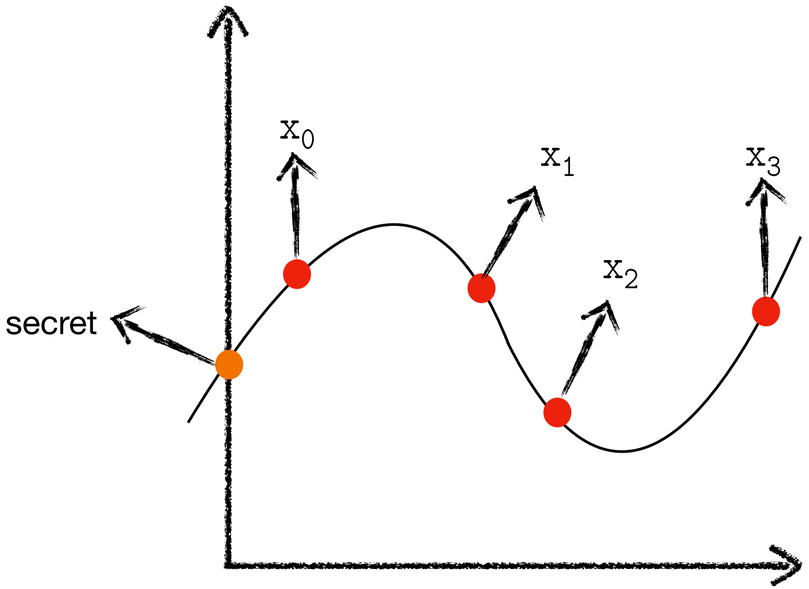
$P_0 \xrightarrow{k_0} P_1$

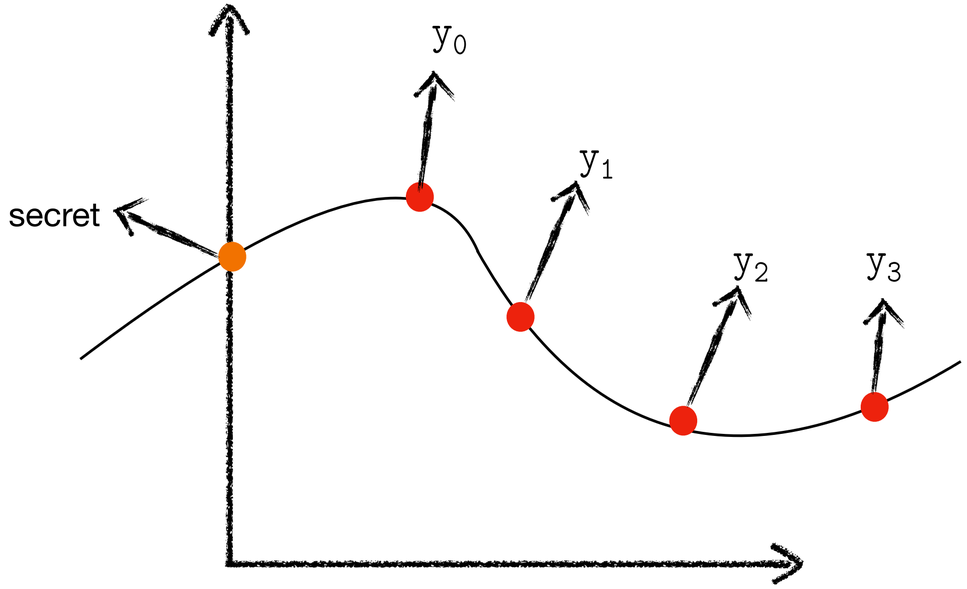
$P_1 \xrightarrow{k_1} P_2$

$P_2 \xrightarrow{k_2} P_0$



Multiplication in Shamir Secret Shares



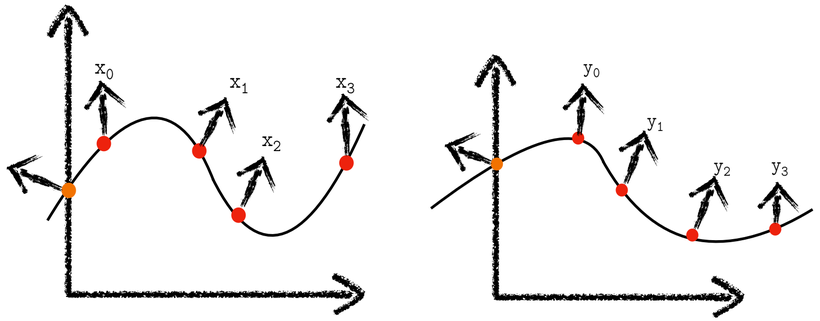


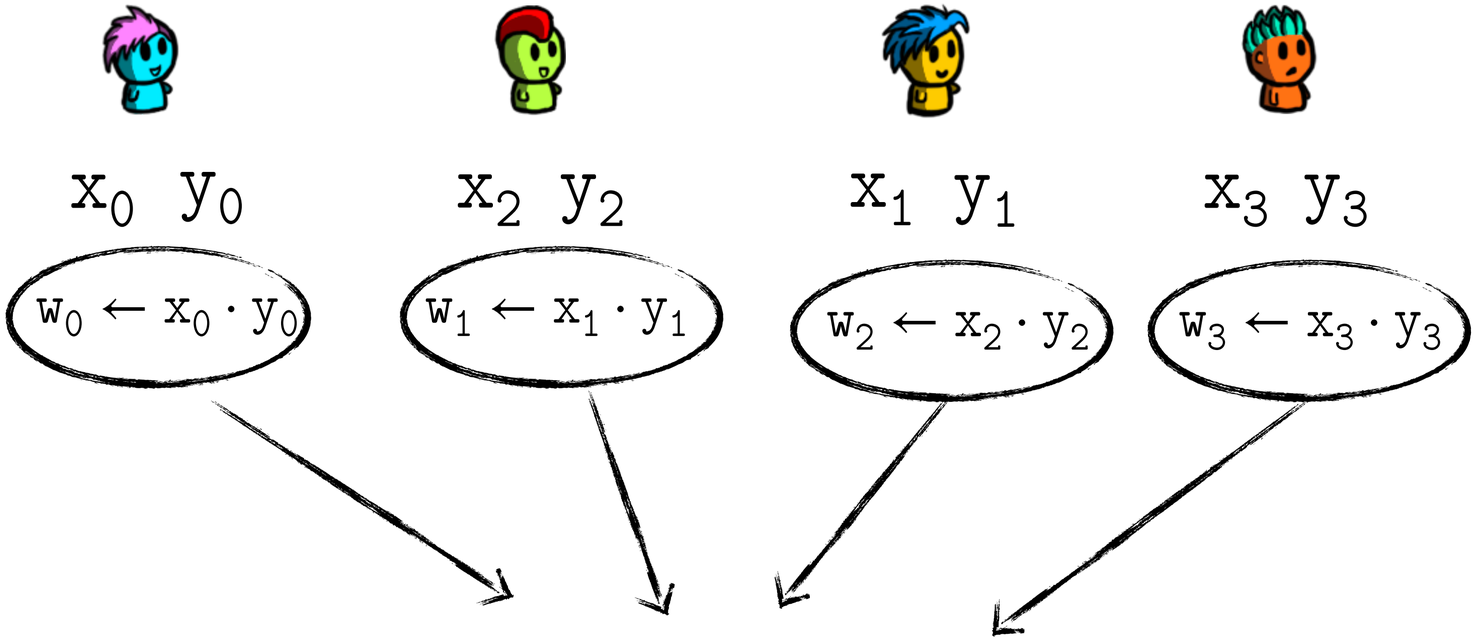



x_0 y_0 x_2 y_2 x_1 y_1 x_3 y_3

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

Multiplication in Shamir Secret Shares





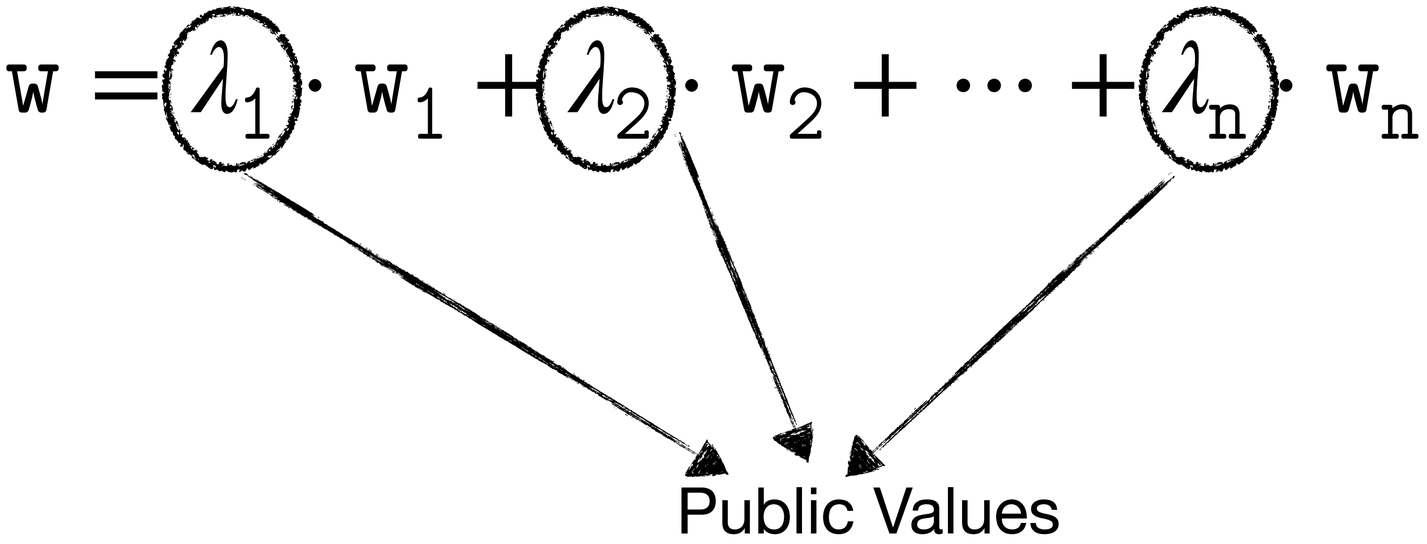
Is that good enough?

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

Degree Reduction

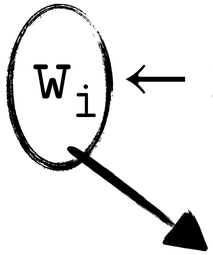
Lagrange Interpolation:

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

$$w_{i,j}$$

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

Degree Reduction

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

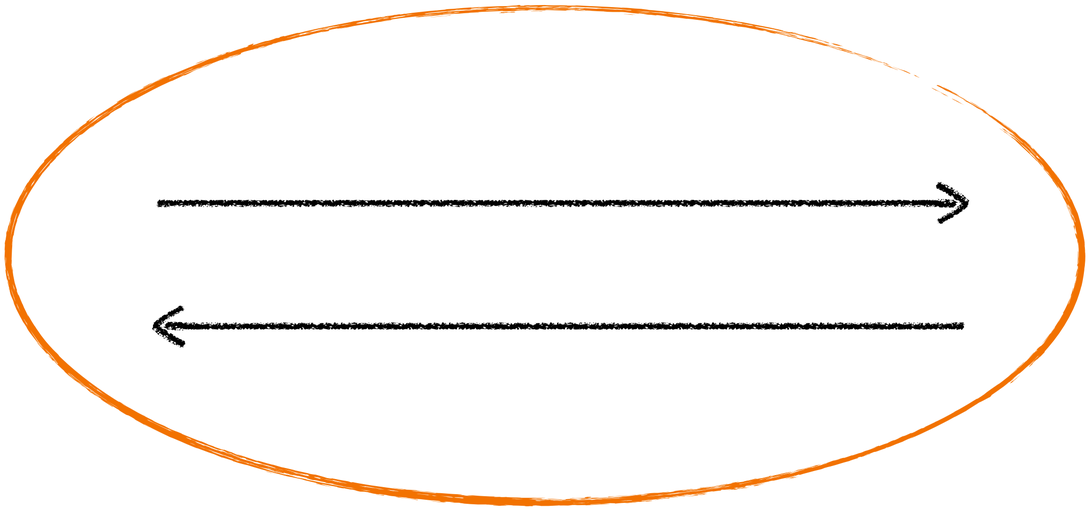
We must have enough parties to reconstruct the intercept of the polynomial with degree $2t$

Oblivious Transfer

Receiver

b

x_b

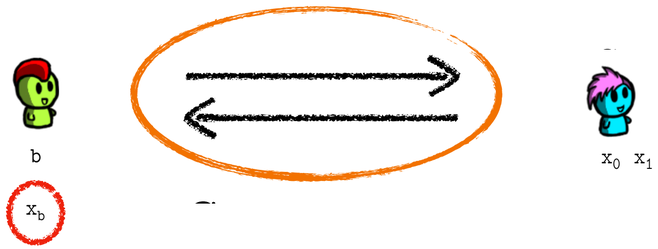


Secure Multiparty Computation

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)

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

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)

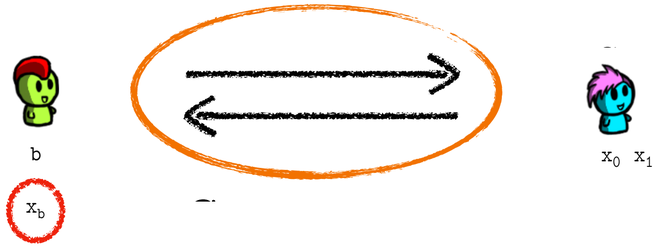
$(Enc_{k_0}(x_0) , Enc_{k_1}(x_1))$

Sender

$x_0 \quad x_1$

x_b

Oblivious Transfer



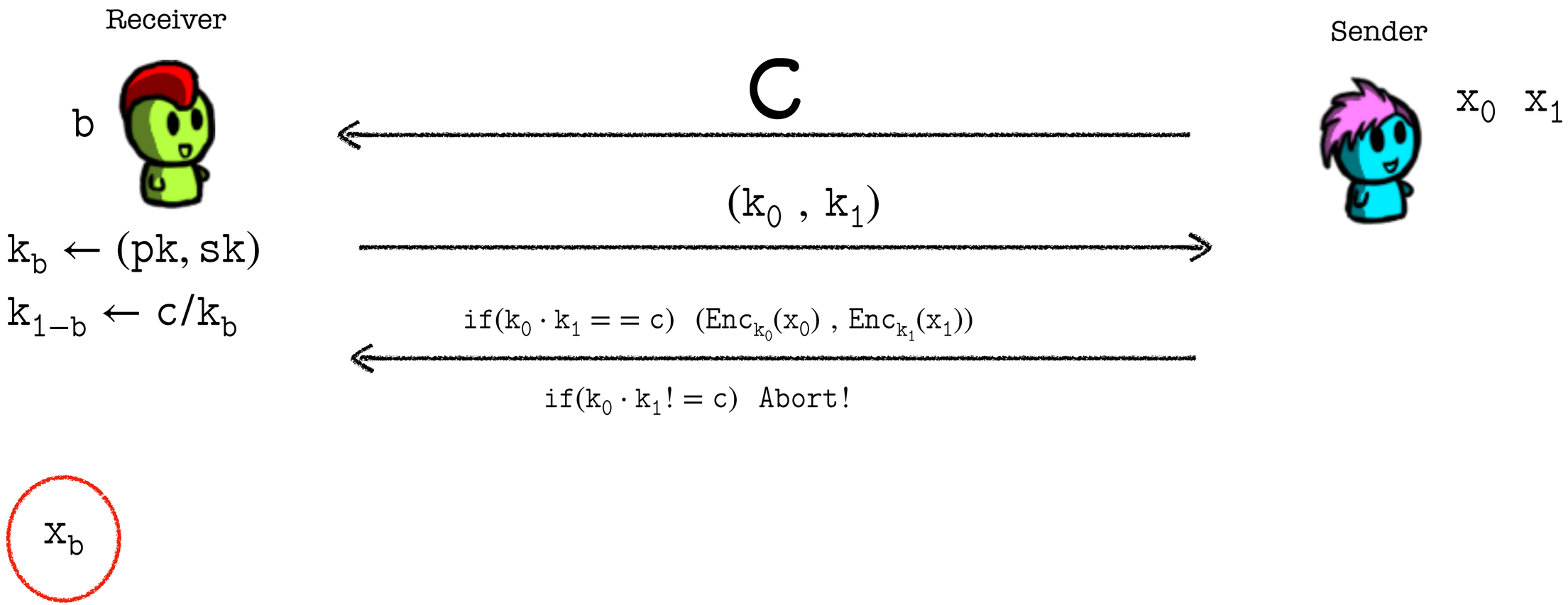
How can we make the above semi-honest protocol work under a malicious setting?

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



Oblivious Transfer

Receiver

b

$k_b \leftarrow (pk, sk)$

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

Sender

$x_0 \quad x_1$

c

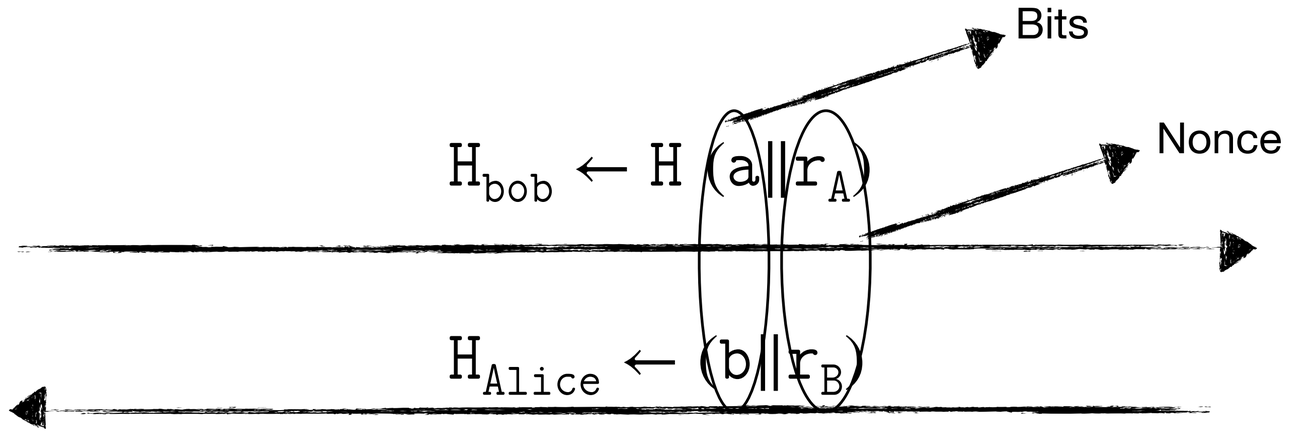
$(k_0 \ , \ k_1)$

$\texttt{if}(k_0 \cdot k_1 == c) \quad (Enc_{k_0}(x_0) \ , \ Enc_{k_1}(x_1))$

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

x_b

Tossing Coin Over Phone (Blum's Coin Toss)



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

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

$a \oplus b$

$a \oplus b$

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

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

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

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

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

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

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

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