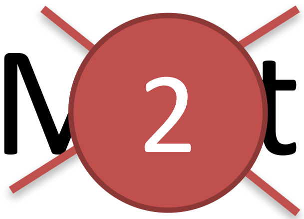


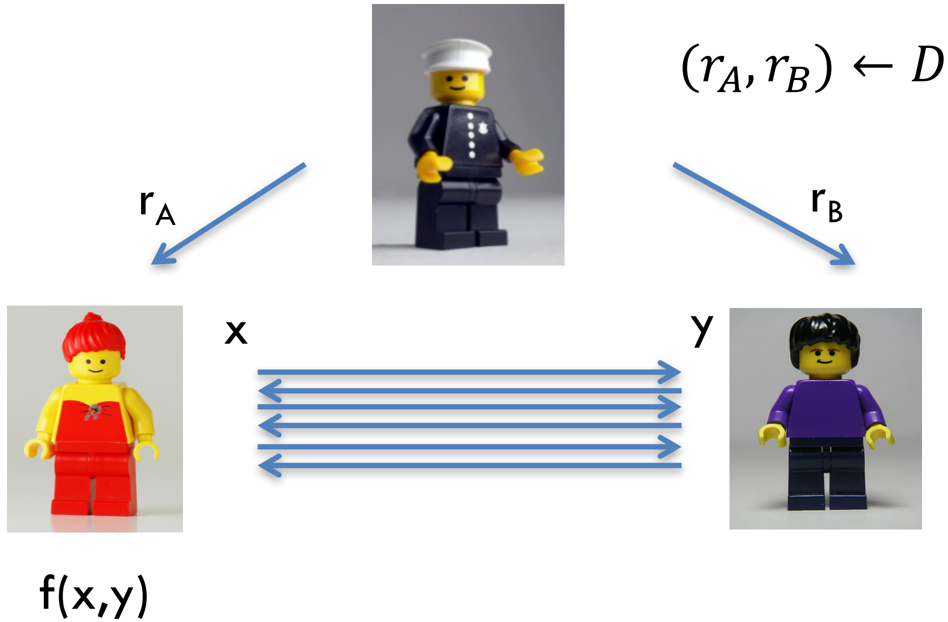
Multi-Party Computation
Part 2

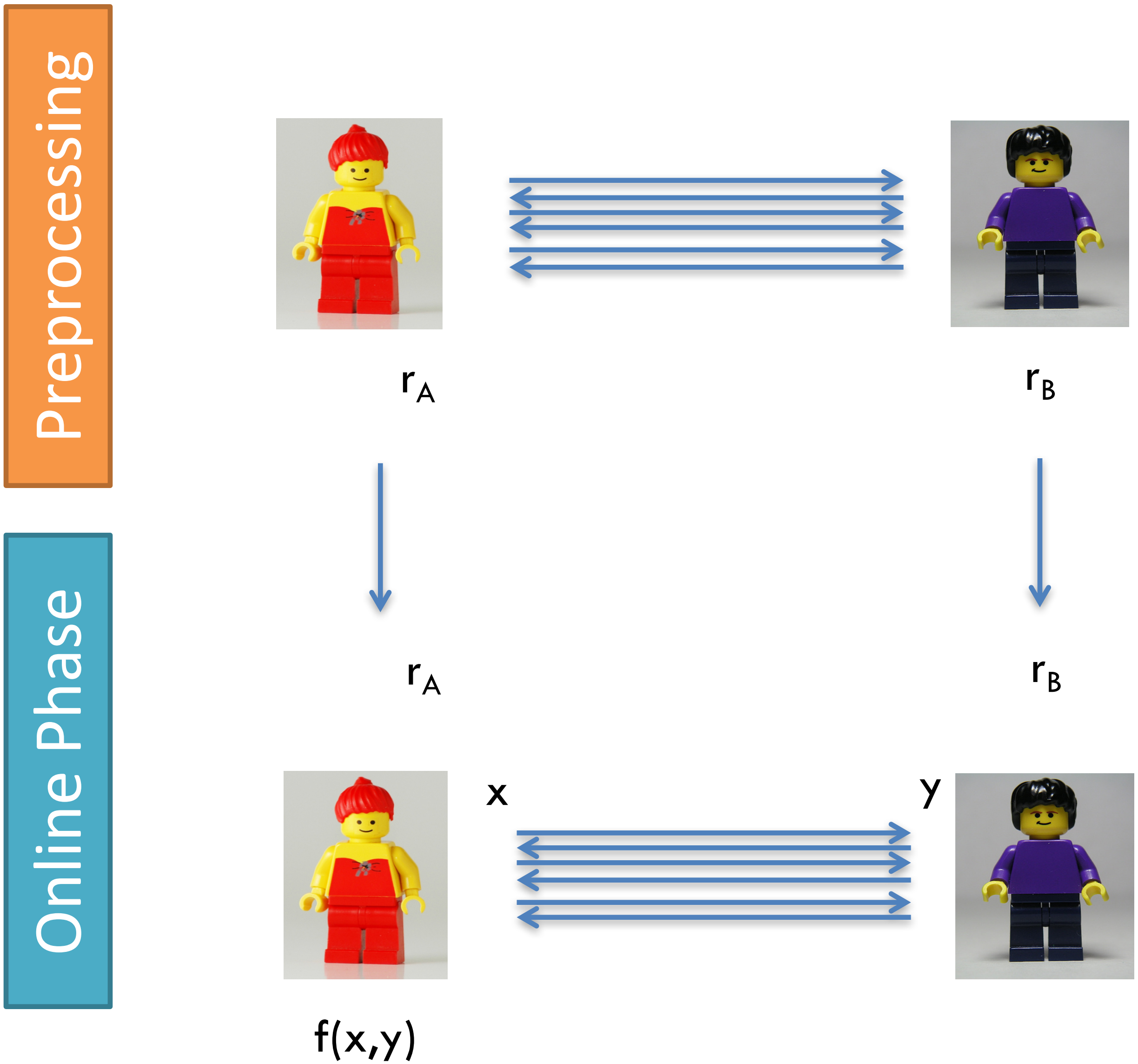
Claudio Orlandi, Aarhus University

Plan for the next 3 hours…

- **Part 1: Secure Computation with a Trusted Dealer**
  - Warmup: One-Time Truth Tables
  - Evaluating Circuits with Beaver's trick
  - MAC-then-Compute for Active Security
- **Part 2: Oblivious Transfer**
  - OT: Definitions and Applications
  - Passive Secure OT Extension
  - OT Protocols from DDH (Naor-Pinkas/PVW)
- **Part 3: Garbled Circuits**
  - GC: Definitions and Applications
  - Garbling gate-by-gate: Basic and optimizations
  - Active security 101: simple-cut-and choose, dual-execution

Trusted Dealer




Preprocessing
Online Phase
r_A
r_B
r_A
r_B
x
y
f(x,y)

Circuit Evaluation (Online phase)

3) Multiplication?

How to compute [z]=[xy] ?

Alice, Bob should compute

$$z_A + z_B = (x_A+x_B)(y_A+y_B)$$

$$= x_Ay_A + x_By_A + x_Ay_B + x_By_B$$

How do we compute this?

Alice can compute this

Bob can compute this

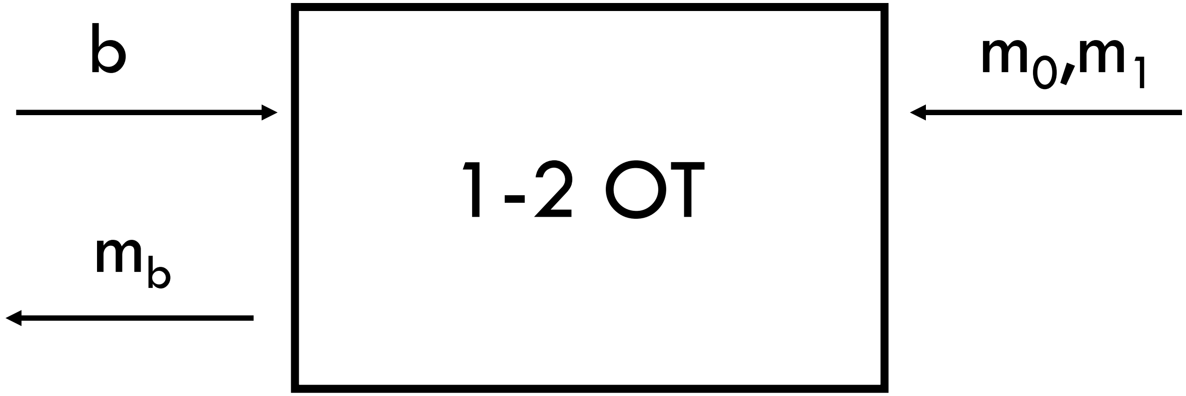
Part 2: Oblivious Transfer

- **OT: Definition, Applications (Gilboa's protocol)**
- Passive Secure OT Extension
- OT Protocols from DDH (Naor-Pinkas/PVW)

Receiver

1-2 OT

Sender

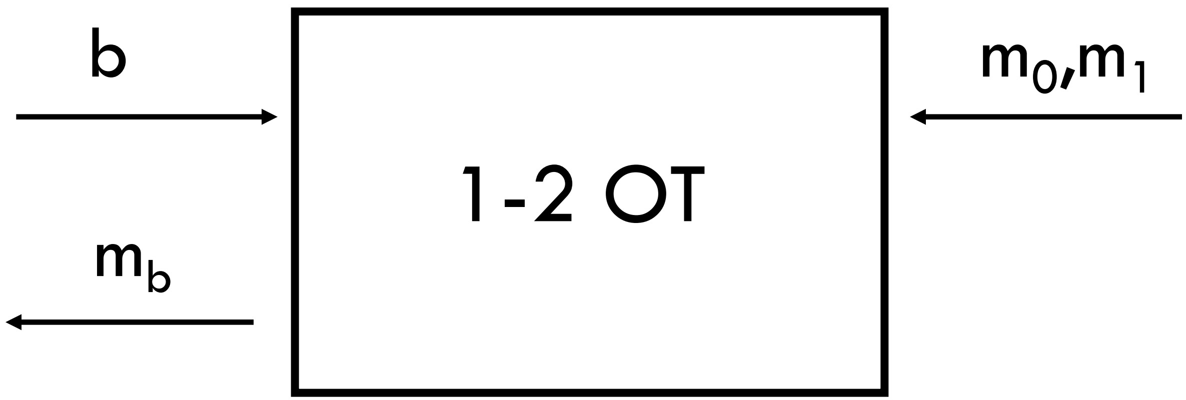


- Receiver does not learn m_{1-b}
- Sender does not learn b

1-2 OT

Receiver

Sender

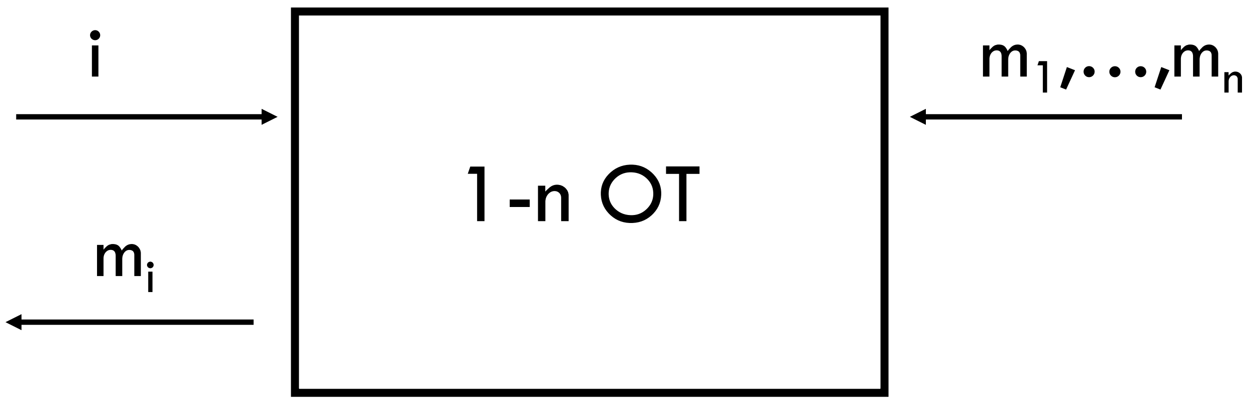


- $m_b = (1-b)\, m_0 + b\, m_1$
- $m_b = m_0 + b\,(m_1 - m_0)$

1-n OT

Receiver

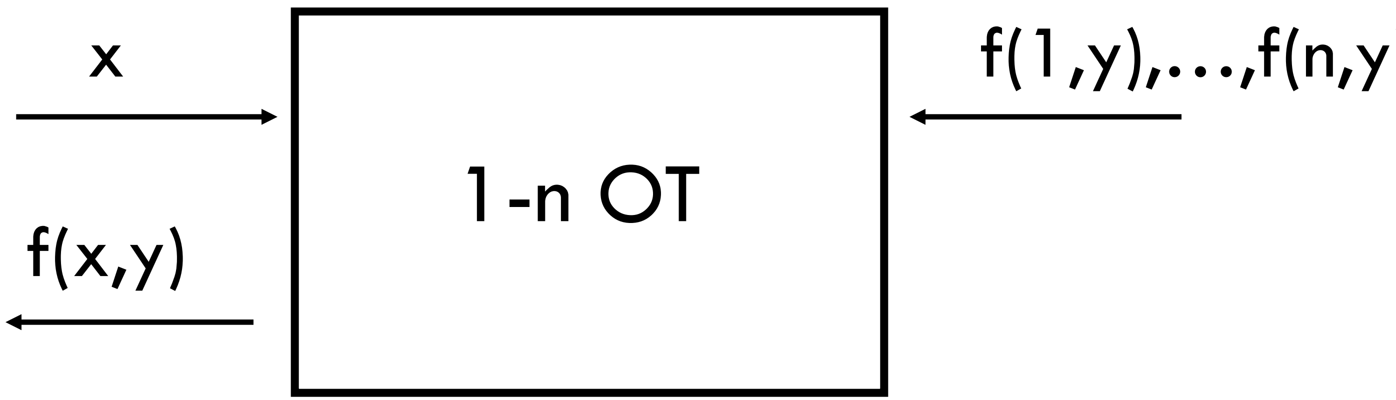
Sender



2PC via 1-n OT

Receiver

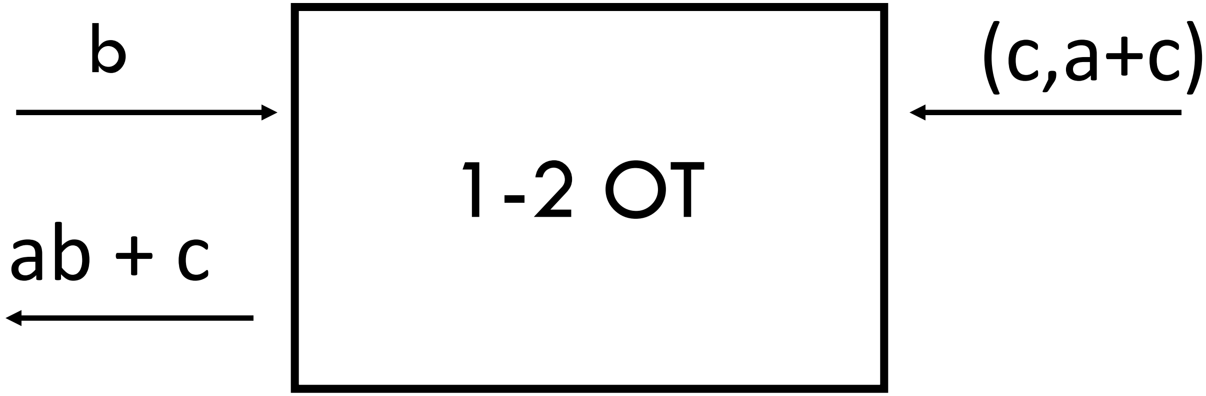
Sender



Oblivious Transfer
=
bit multiplication

Receiver

Sender



GILBOA’S PROTOCOL

n OTs = Arith. Multiplication

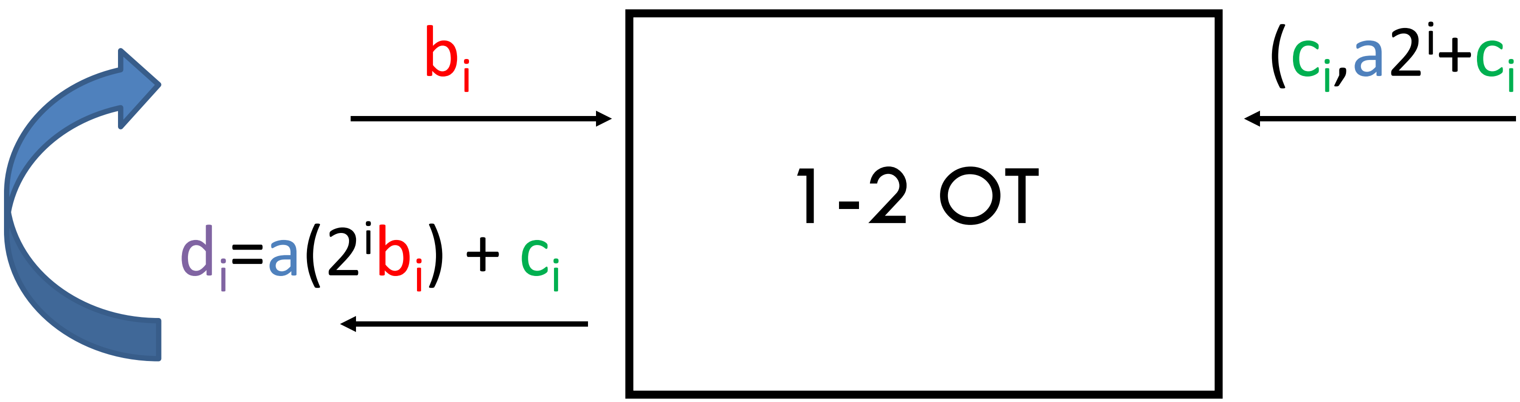
Receiver

$b=(b_0,b_1,\ldots,b_{n-1})$

Sender

a (n bit number)

$c_0+\ldots+c_{n-1}=c$



$$d_0+\ldots+d_{n-1}=a(b_0+2b_1+\ldots+2^{n-1}b_{n-1})+(c_0+\ldots+c_{n-1})=ab+c$$

Part 2: Oblivious Transfer

- OT definition, applications (Gilboa’s protocol)
- **Passive Secure OT Extension (IKNP03)**
- OT Protocols from DDH (Naor-Pinkas/PVW)

Efficiency

- ***Problem:*** OT requires public key primitives, inherently efficient

The Crypto Toolbox

Weaker assumption ← Stronger assumption

OTP >> SKE >> PKE >> FHE >> Obfuscation

More efficient ← Less efficient

Efficiency

- ***Problem:*** OT requires public key primitives, inherently efficient

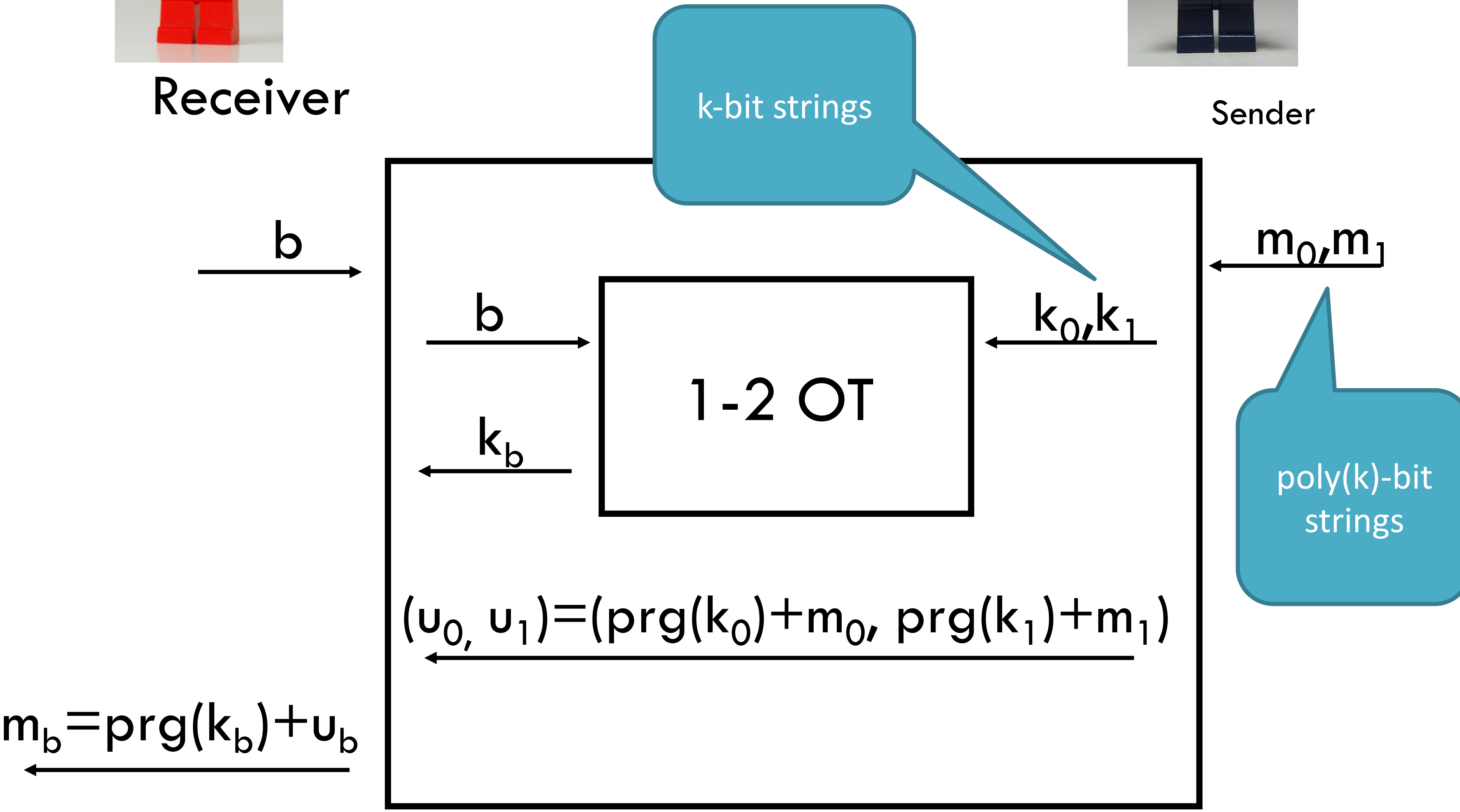
- ***Solution:*** OT extension
  - Like hybrid encryption!
  - Start with few (expensive) OT based on PKE
  - Get many (inexpensive) OT using only SKE

WARMUP: USEFUL OT PROPERTIES

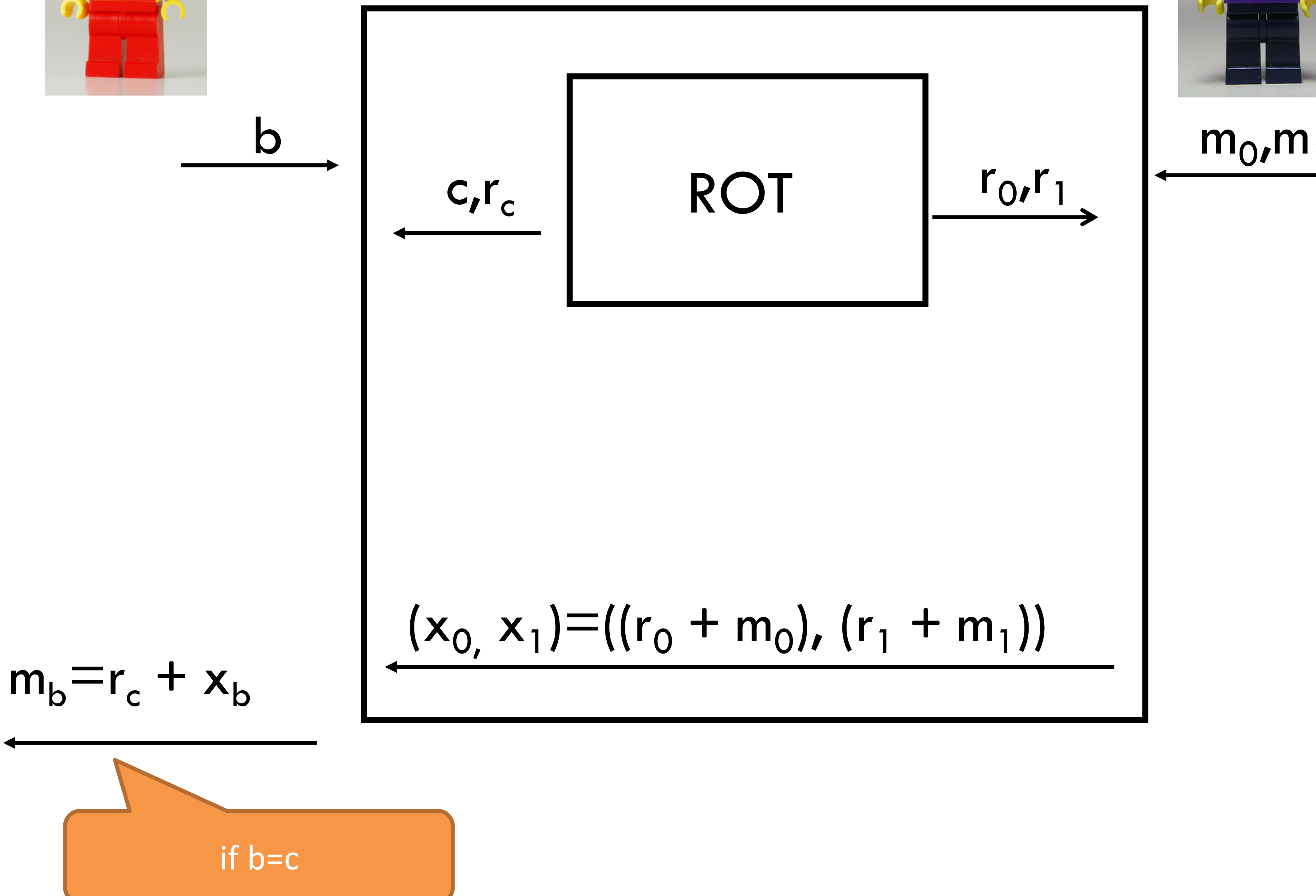
Short OT → Long OT

Receiver

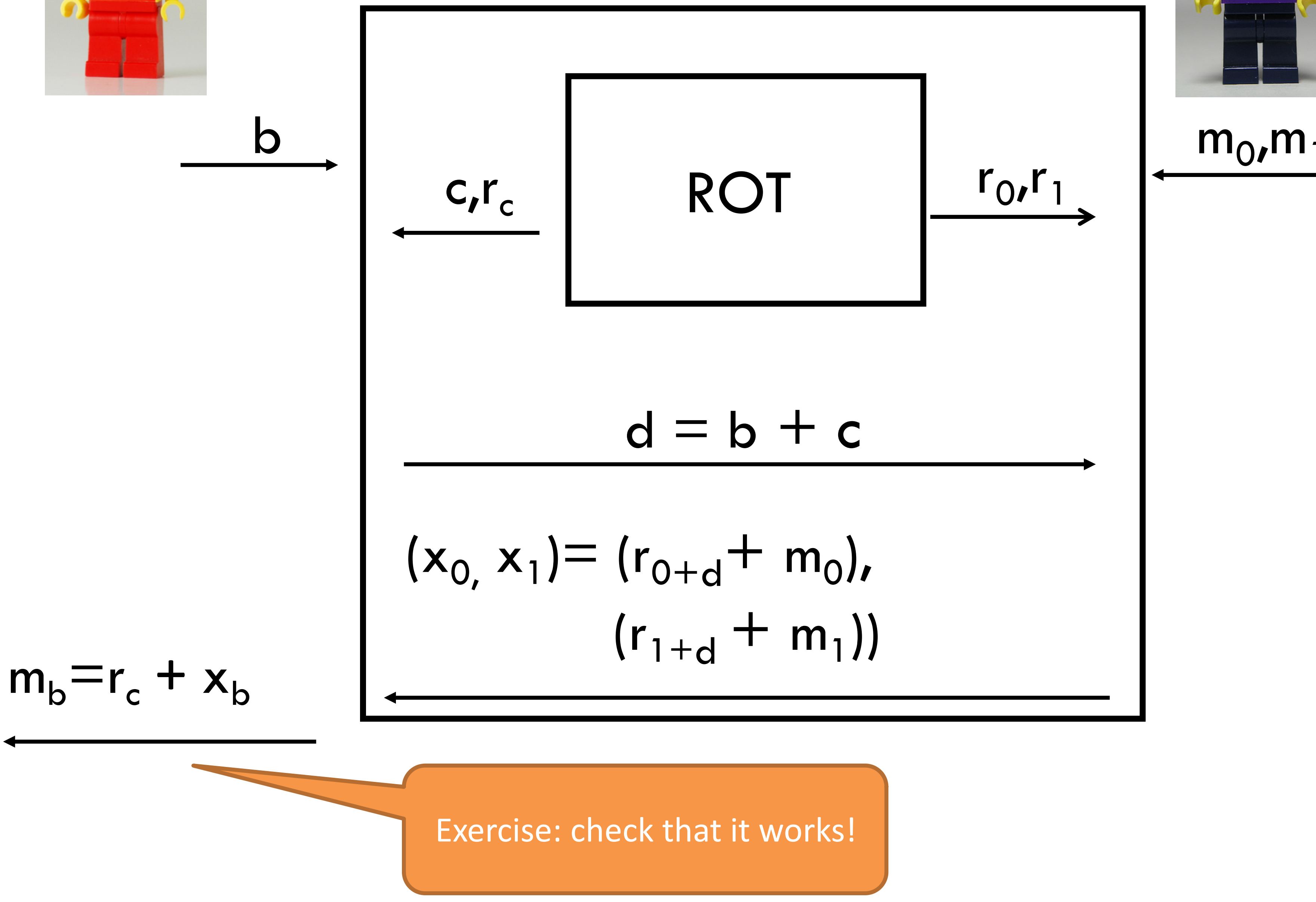
Sender



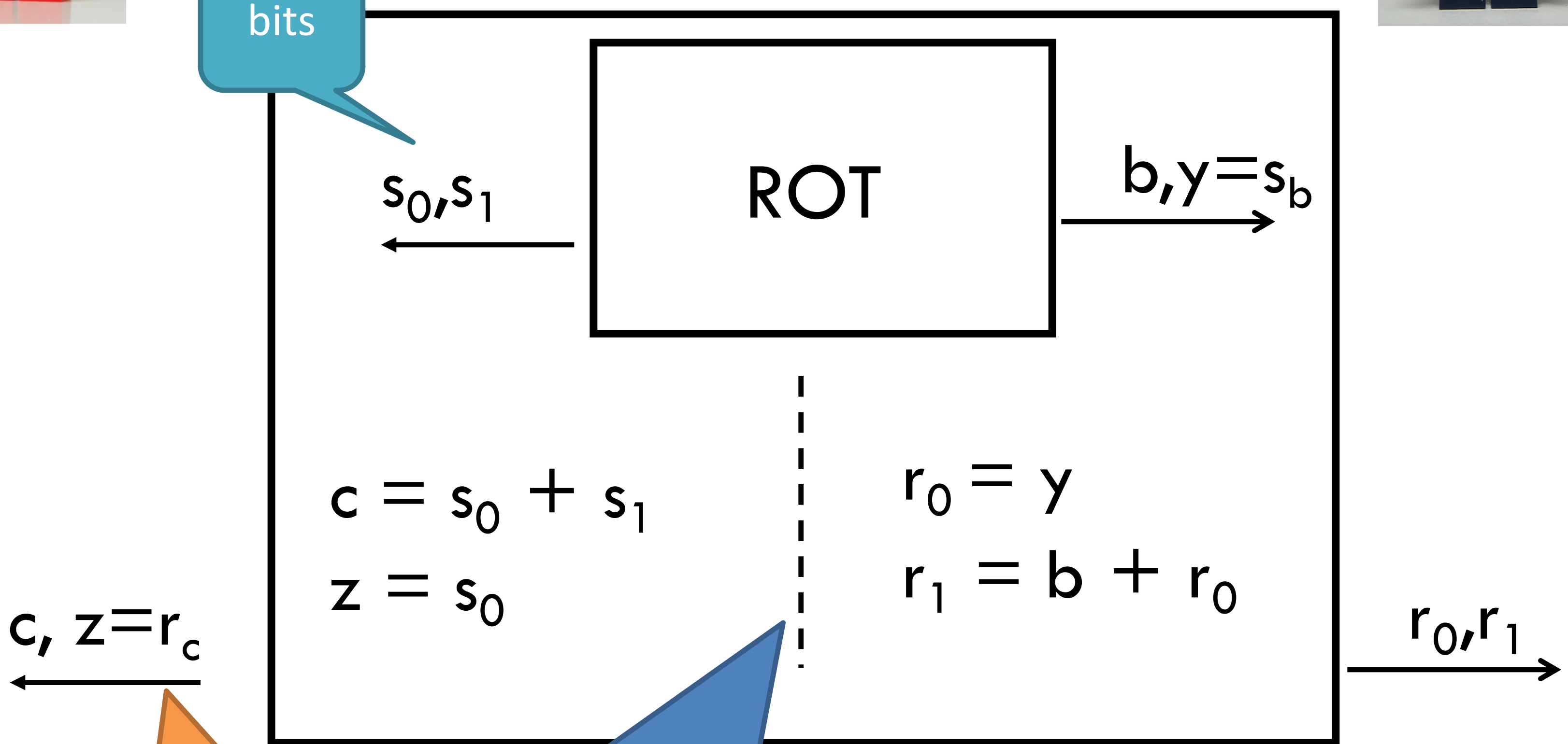
Random OT = OT



Random OT = OT


b
m_0, m_1
ROT
c, r_c
r_0, r_1
$d = b + c$
$(x_0, x_1) = (r_{0+d} + m_0),$
$(r_{1+d} + m_1))$
$m_b = r_c + x_b$
Exercise: check that it works!

(R)OT is symmetric



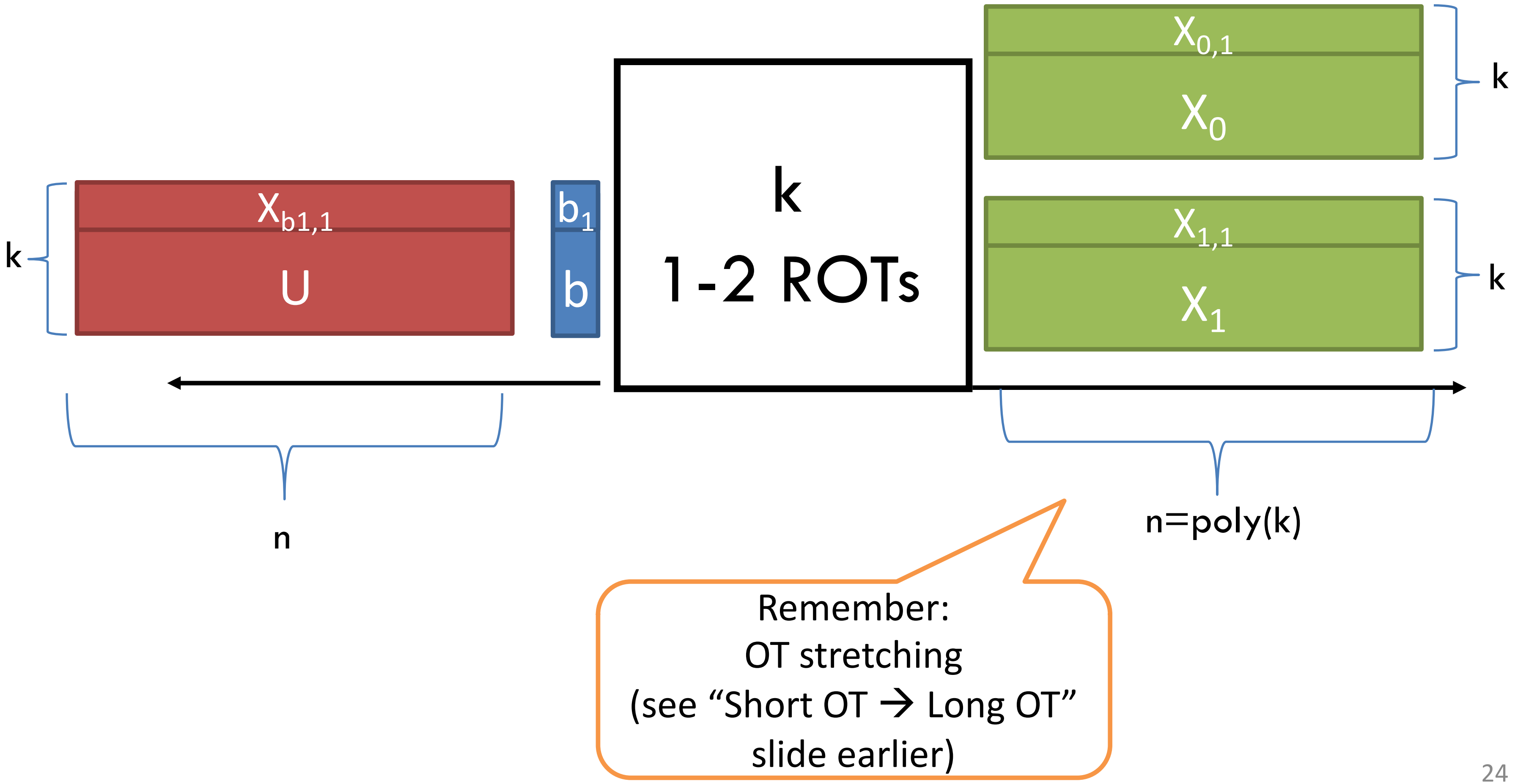
No communication!

Exercise: check that it works

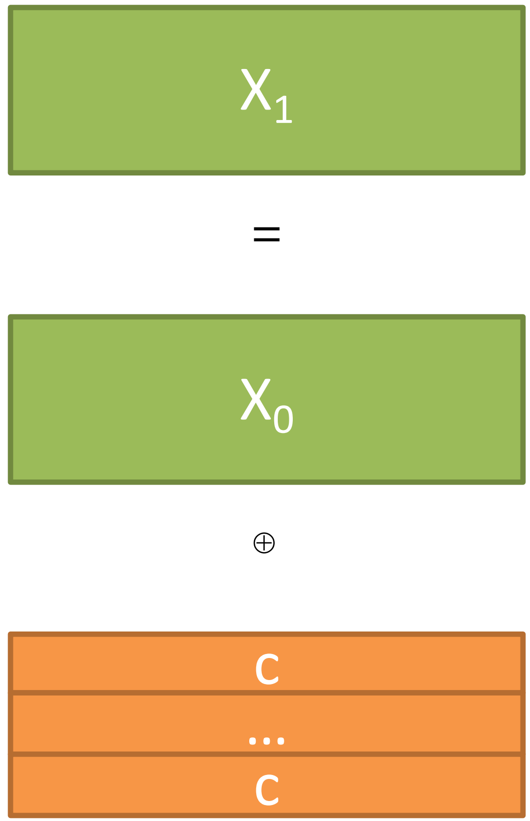
OT Extension

- OT pro(v/b)ably requires public-key primitivies
  - OT extension ≈ hybrid encryption
  - **Start from k “real” OTs**
  - **Turn them into poly(k) OTs using only few symmetric primitives per OT**

OT Extension, Pictorially



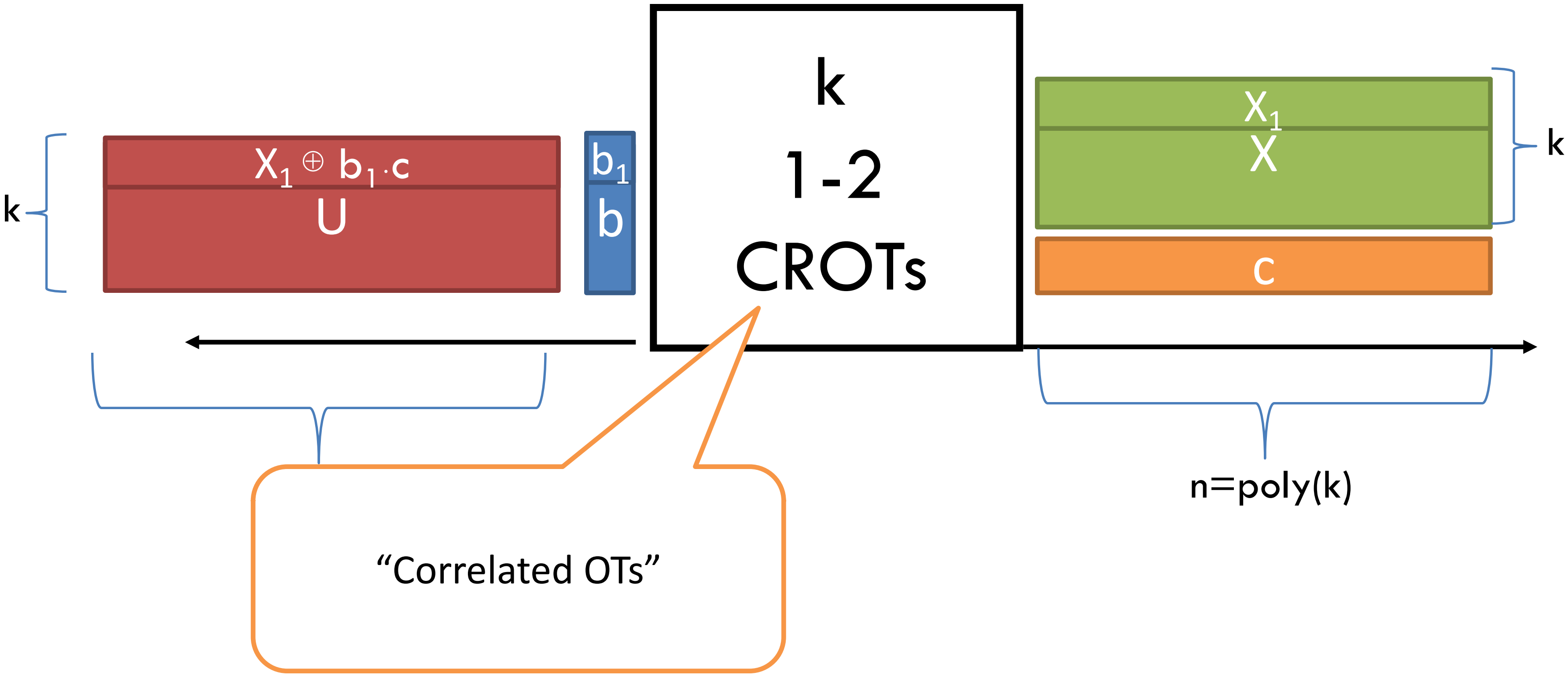
Condition for OT extension



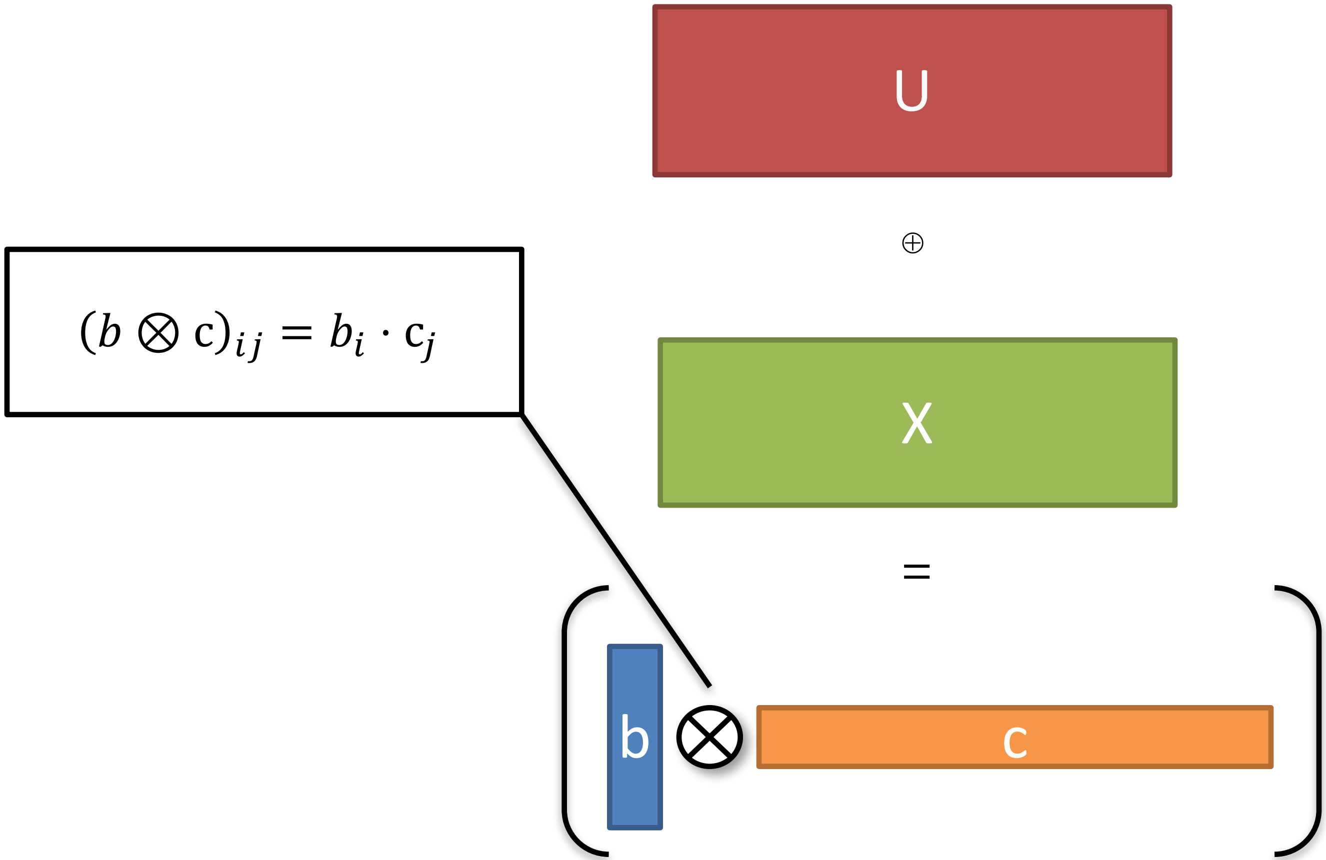
Remember:
"Random OT → OT"

Problem for active security!

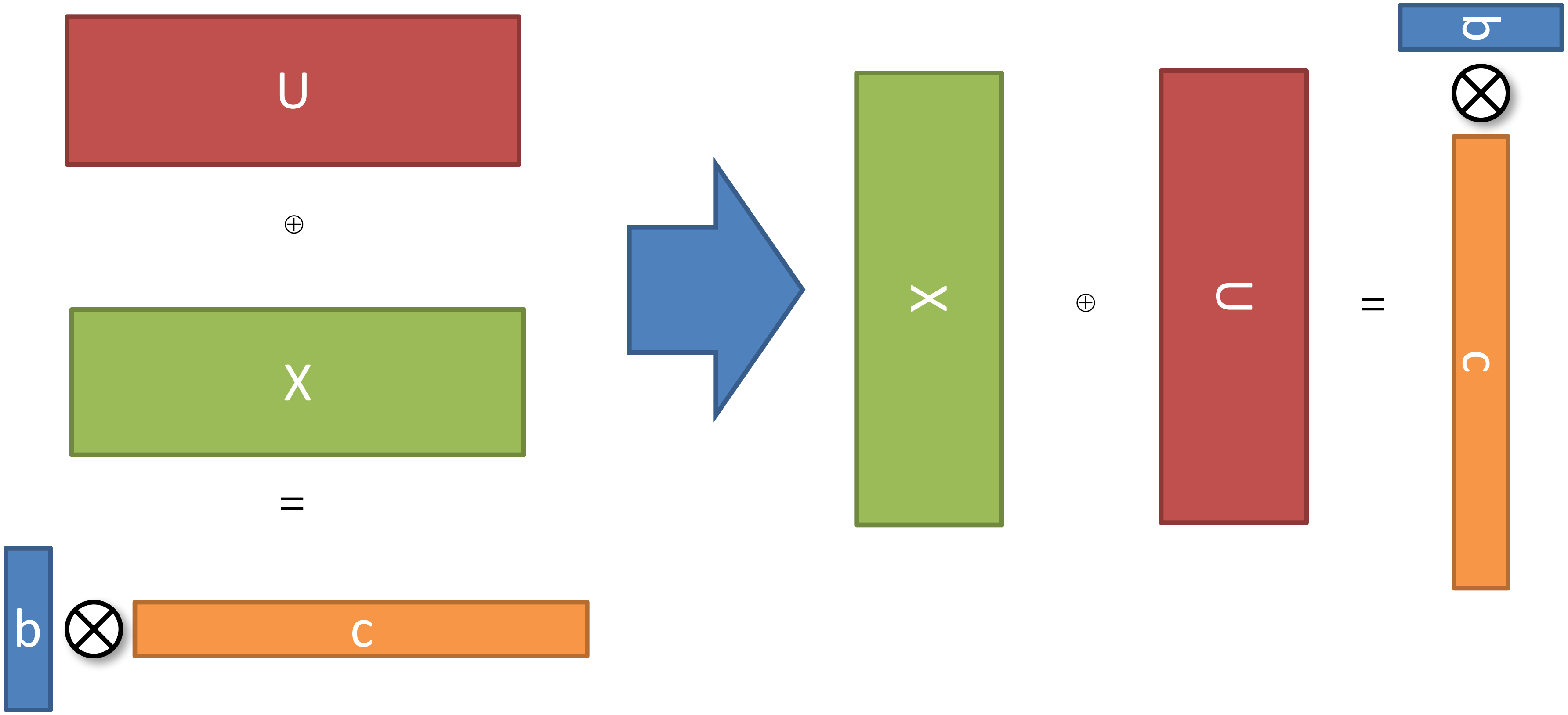
OT Extension, Pictorially



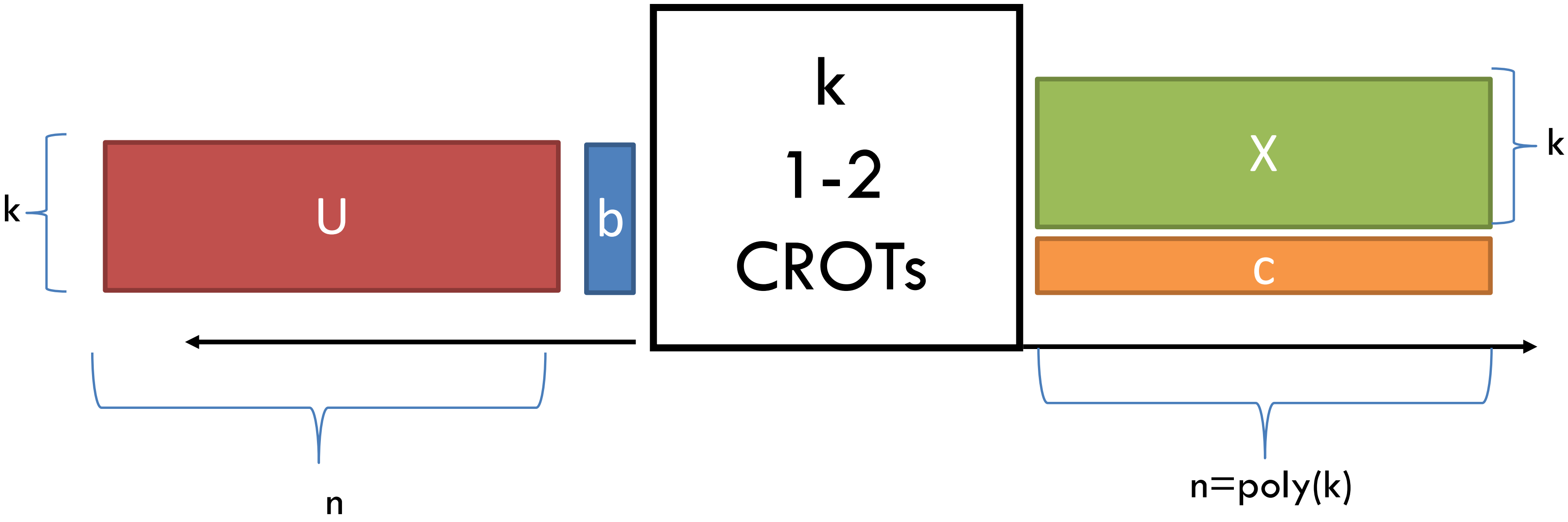
OT Extension, Pictorially



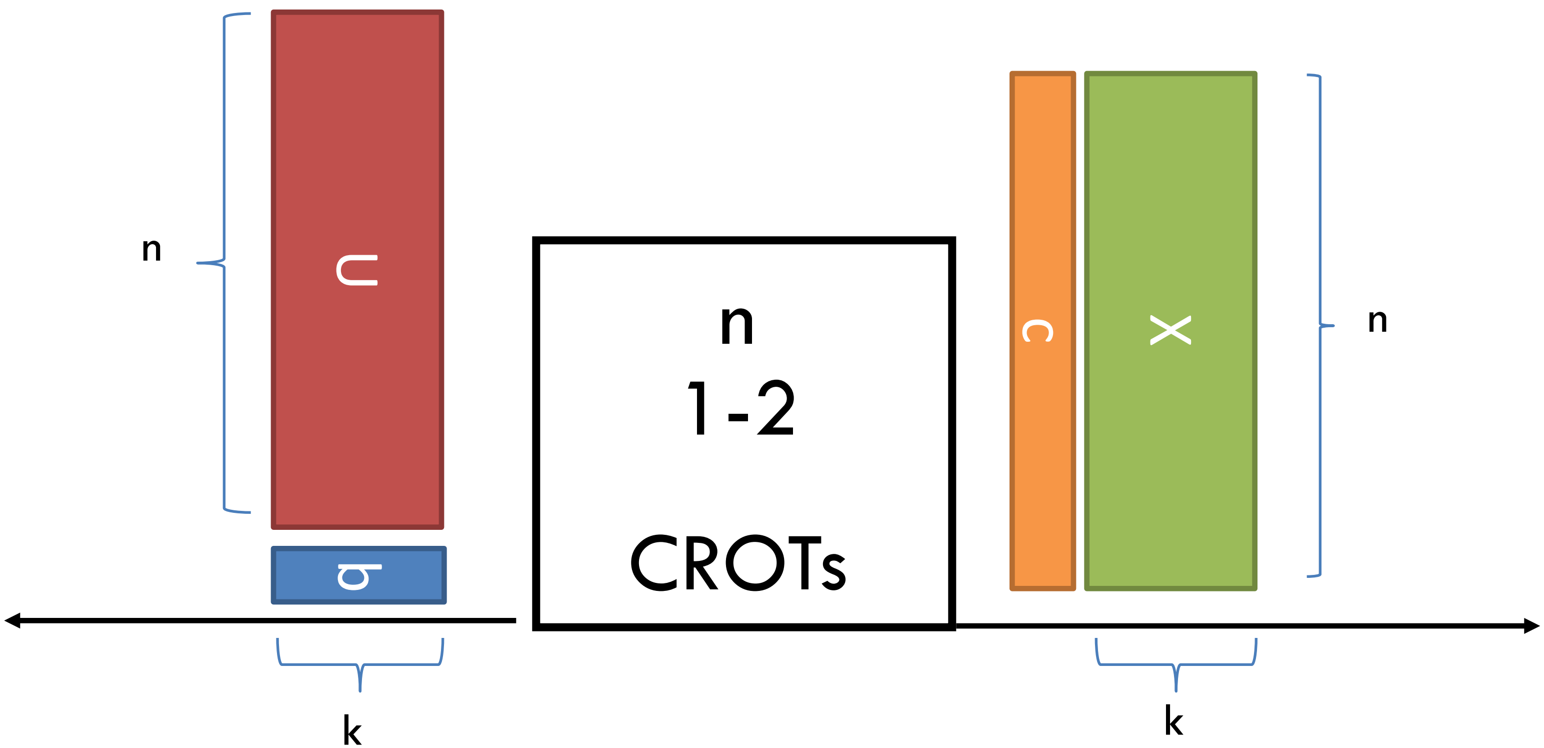
OT Extension, Turn your head!



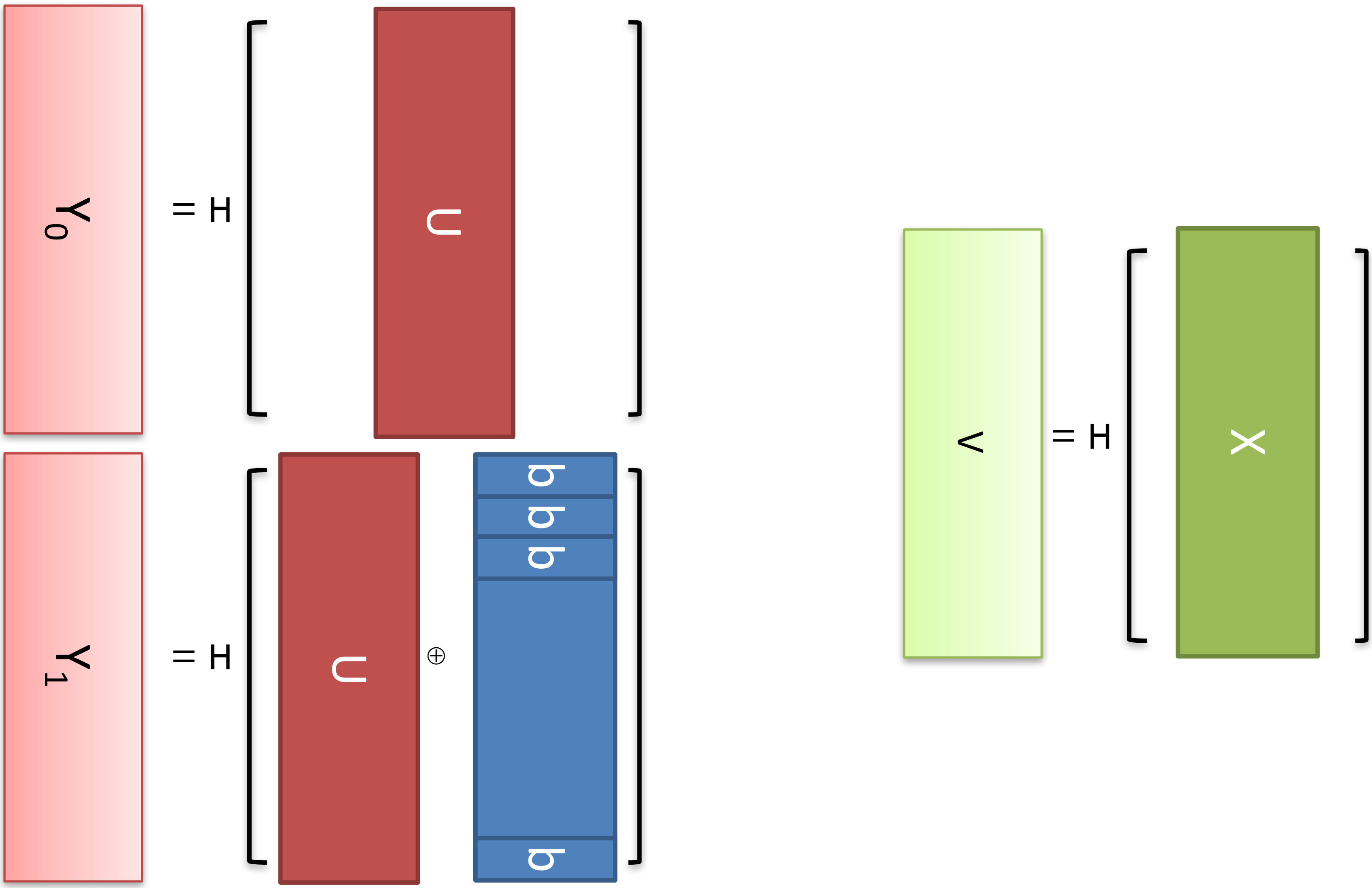
OT Extension, Pictorially



OT Extension, Pictorially



Break the correlation!

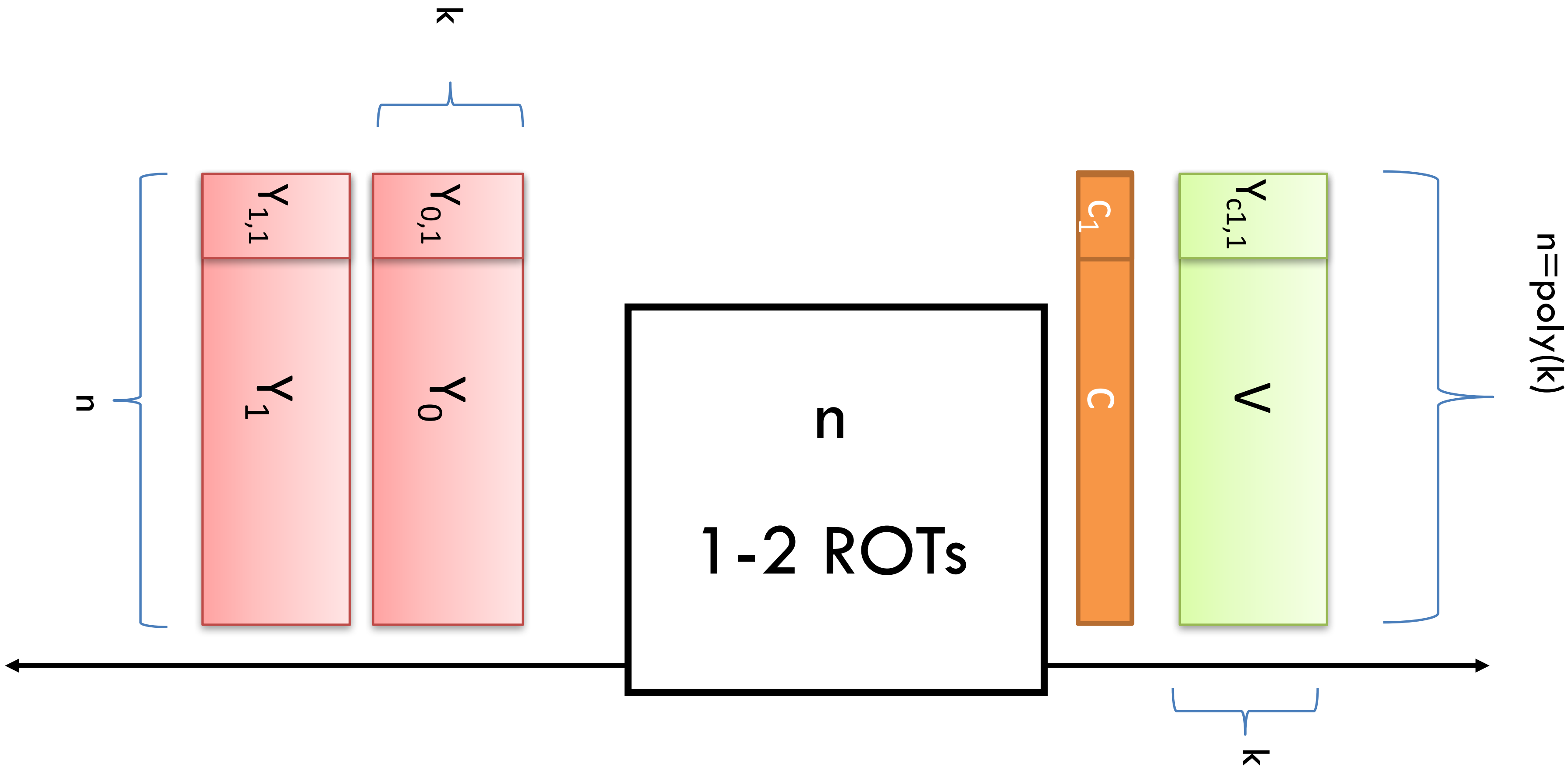


Breaking the correlation

- Using a **correlation robust hash function** H s.t.
  1. $\{a_0, \ldots, a_n, H(a_0 + r), \ldots, H(a_n + r)\}$ // (a_i's, r random)
  2. $\{a_0, \ldots, a_n, b_0, \ldots, b_n\}$ // (a_i's, b_i's random)

  are ***computationally indistinguishable***

OT Extension, Pictorially



Recap

0. Strech **k OTs** from *k- to poly(k)=n-bitlong strings*
1. Send correction for each pair of messages x^i_0, x^i_1
   s.t., $\mathbf{x^i_0 \oplus x^i_1 = c}$
2. **Turn your head** (S/R swap roles)
3. The bits of **c** are the new **choice bits**
4. Break the correlation: $\mathbf{y^j_0 = H(u^j),\ y^j_1 = H(u^j \oplus b)}$

- **Not secure against active adversaries**

Part 2: Oblivious Transfer

- OT definition, applications (Gilboa's protocol)
- Passive Secure OT Extension
- **OT Protocols from DDH (Naor-Pinkas/PVW)**

Passive Secure OT

Receiver(b)

Sender(m_0,m_1)

Receiver privacy:
Real pk ≈ "random" pk

$pk_b \leftarrow G(sk)$

$pk_{1-b} \leftarrow Rand()$

(pk_0,pk_1)

→

$c_0=E(pk_0,m_0)$, $c_1=E(pk_1,m_1)$

←

$m_b = D(sk,c_b)$

Sender privacy:
encryption is secure
(Alice does not have sk)

Malicious

Receiver(b)

Passive Secure OT

Sender(m_0,m_1)

$pk_0 \leftarrow G(sk_0)$

$pk_1 \leftarrow G(sk_1)$

(pk_0,pk_1) →

← c_0=E(pk_0,m_0), c_1=E(pk_1,m_1)

$m_0 \leftarrow D(sk_0, c_0)$

$m_1 \leftarrow D(sk_1, c_1)$

Active Secure OT

Receiver(b)

Sender(m_0,m_1)

crs

mpk ← f(crs,sk,b)

mpk →

(pk_0,pk_1)=G(mpk,crs)

c_0=E(pk_0,m_0), c_1=E(pk_1,m_1) ←

m_b = D(sk,c_b)

Keys are correlated, Receiver cannot learn the sk for both

Naor-Pinkas OT

(a la Chou-Orlandi)

Receiver(b)

Sender(m_0,m_1)

crs (single group element)

$mpk = crs^b g^{sk}$

mpk →

pk_0=mpk

pk_1=mpk/crs

← c_0=E(pk_0,m_0), c_1=E(pk_1,m_1)

Encryption is ElGamal

m_b = D(sk,c_b)

PVW OT

Receiver(b)

Sender(m_0,m_1)

crs=(g_0,h_0,g_1,h_1)

(u,v)=(g_b^{sk},h_b^{sk})

(u,v) →

pk_0=(g_0,h_0,u,v)

pk_1=(g_1,h_1,u,v)

← c_0=E(pk_0,m_0), c_1=E(pk_1,m_1)

m_b = D(sk,c_b)

Encryption is
“Double ElGamal”

Security for Receiver

- Random crs → (g_0,h_0,g_1,h_1) is ***not*** DDH tuple
- Then:
  - pk_b *is DDH tuple*

    $(g_b,h_b,u,v)=(g_b,h_b,g_b^{sk},h_b^{sk})$

  - pk_{1-b} *is ¬DDH tuple* (check)

    $(g_{1-b},h_{1-b},u,v)=(g_{1-b},h_{1-b},g_b^{sk},h_b^{sk})$

- ***DDH assumption says Bob cannot learn b***
- *(knowing the DLs in the crs the simulator can extract b)*

Security for Sender

ElGamal Encryption

- *Public key $u=g^x$ and secret key x*

$$(c,d)=(g^r,u^r m) \rightarrow m=dc^{-x}$$

Security for Sender

"Double ElGamal Encryption"

- *Public key $(u,v)=(g^x, h^x)$ and secret key x*

  $(c,d)=(g^r h^s, u^r v^s m)$

DDH : $(g,h,u,v)=(g,h,g^x,h^x)$

→ $dc^{-x}=m$

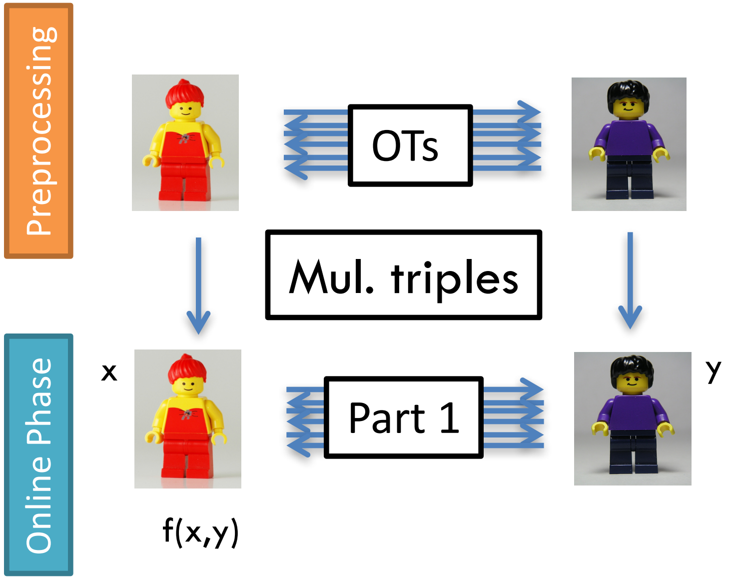
¬ DDH : $(g,h,u,v)=(g,h,g^x,g^y)$

→ *(c,d) unif. random pair*

- Random crs → (g_0,h_0,g_1,h_1) is ¬ DDH
  - → For all (u,v) : (g_0,h_0,u,v) OR (g_1,h_1,u,v) is ¬ DDH
  - → m_{1-b} is statistically hidden

In the proof simulator can set (g_0,h_0,g_1,h_1) = DDH (ind. from real world)
→ Both pk_0 and pk_1 are DDH and simulator can extract both messages

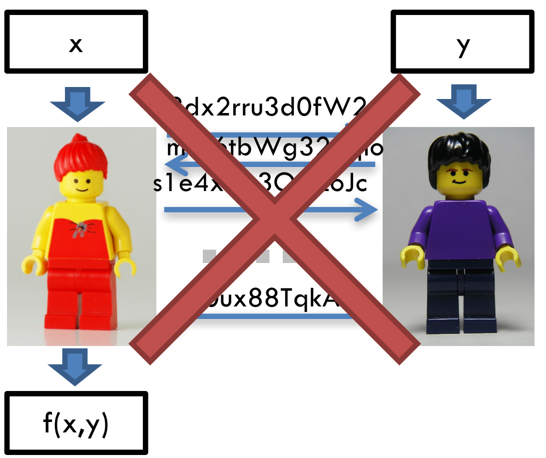
Recap of Part 2

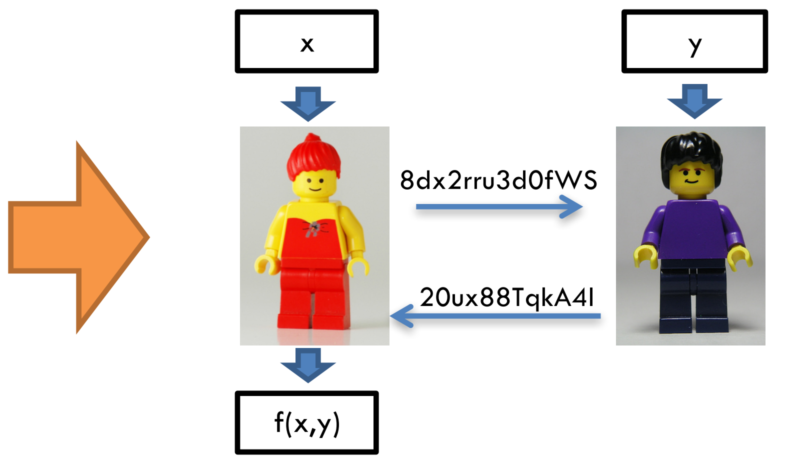


- OT: building block for 2PC
  - **Requires PKE** ☹
  - **OT Extension (using only SKE)** ☺
  - Can be combined with protocols from part 1 for 2PC without a trusted dealer (using computational assumptions) ☺
  - **#rounds** = depth of the circuit 😐

Coming up next…

- OT + Garbled Circuits → **Constant round 2PC!**





…aka layman fully-homomorphic encryption