CIS 2018

Efficient MPC

Optimizations for Garbled Circuits

SODA
Scalable Oblivious Data Analytics

erc
European Research Council
Established by the European Commission

DANMARKS FRIE FORSKNINGSFOND
INDEPENDENT RESEARCH FUND DENMARK

Claudio Orlandi, Aarhus University

Part 3: Garbled Circuits

- **GC: Definitions and Applications**
- Garbling gate-by-gate: Basic and optimizations
- Active security 101: simple-cut-and choose, dual-execution

Garbled Circuit

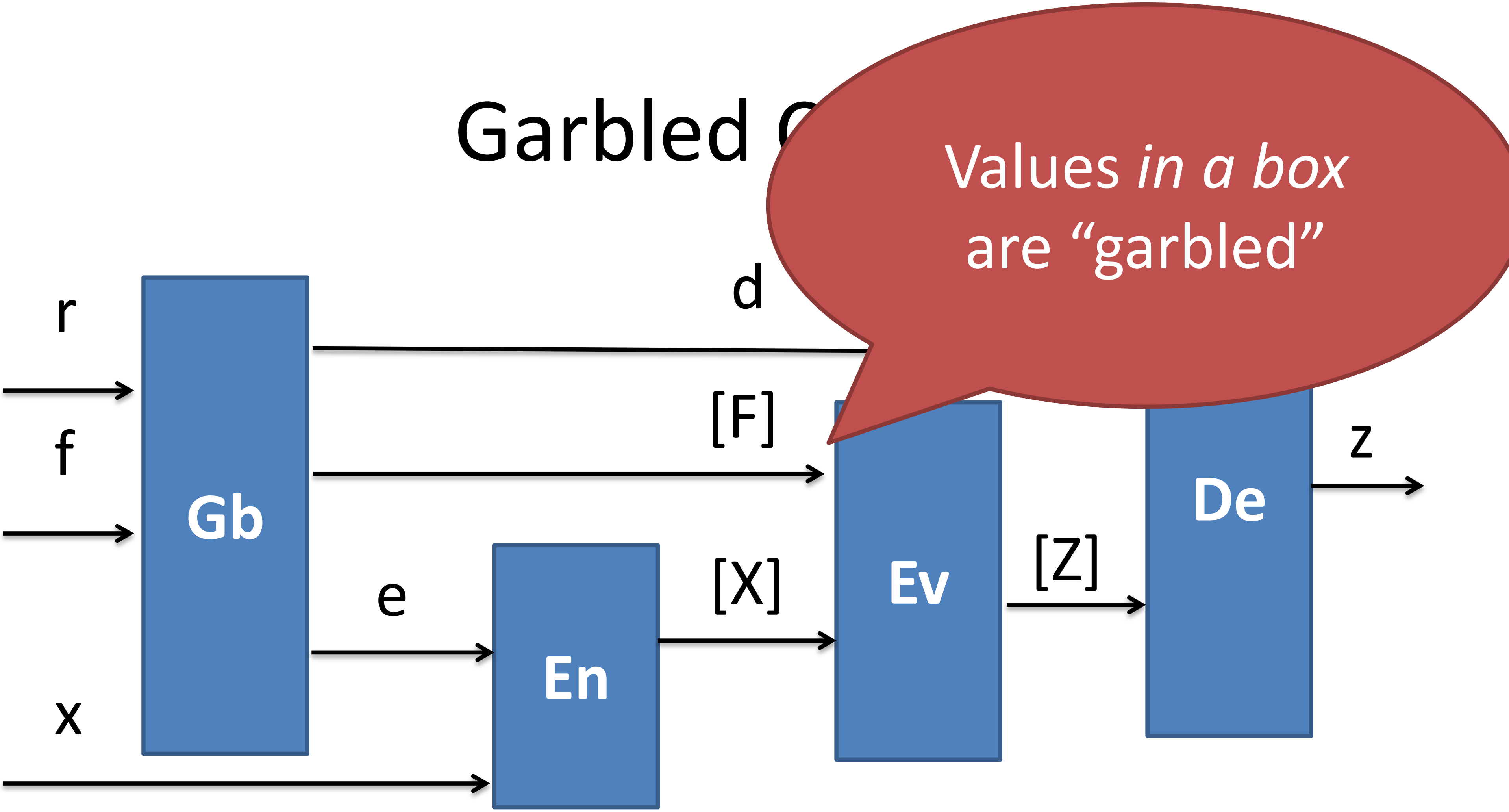
Cryptographic primitive that allows to evaluate

encrypted functions

on

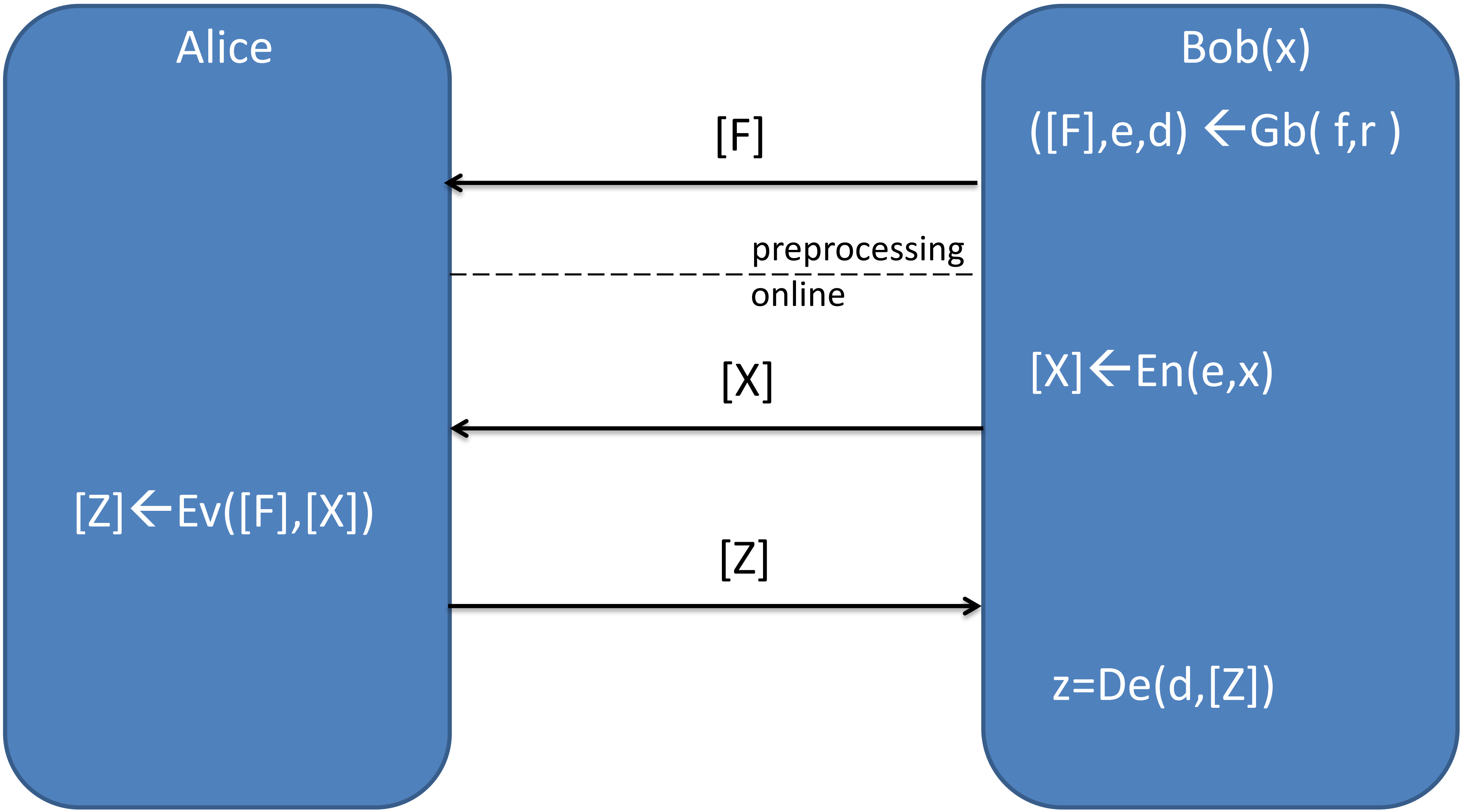
encrypted inputs

Garbled C

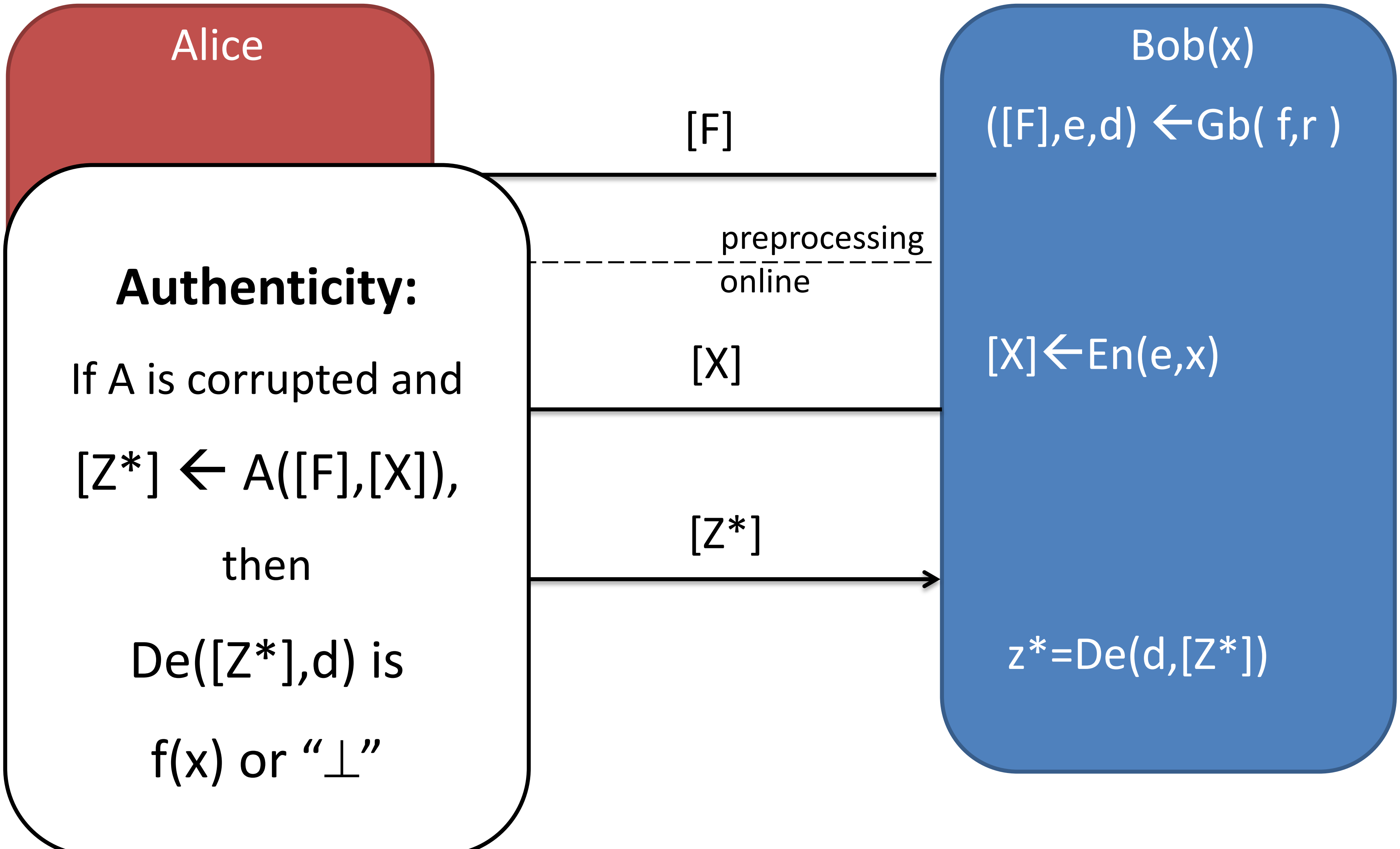


Correct if z=f(x)

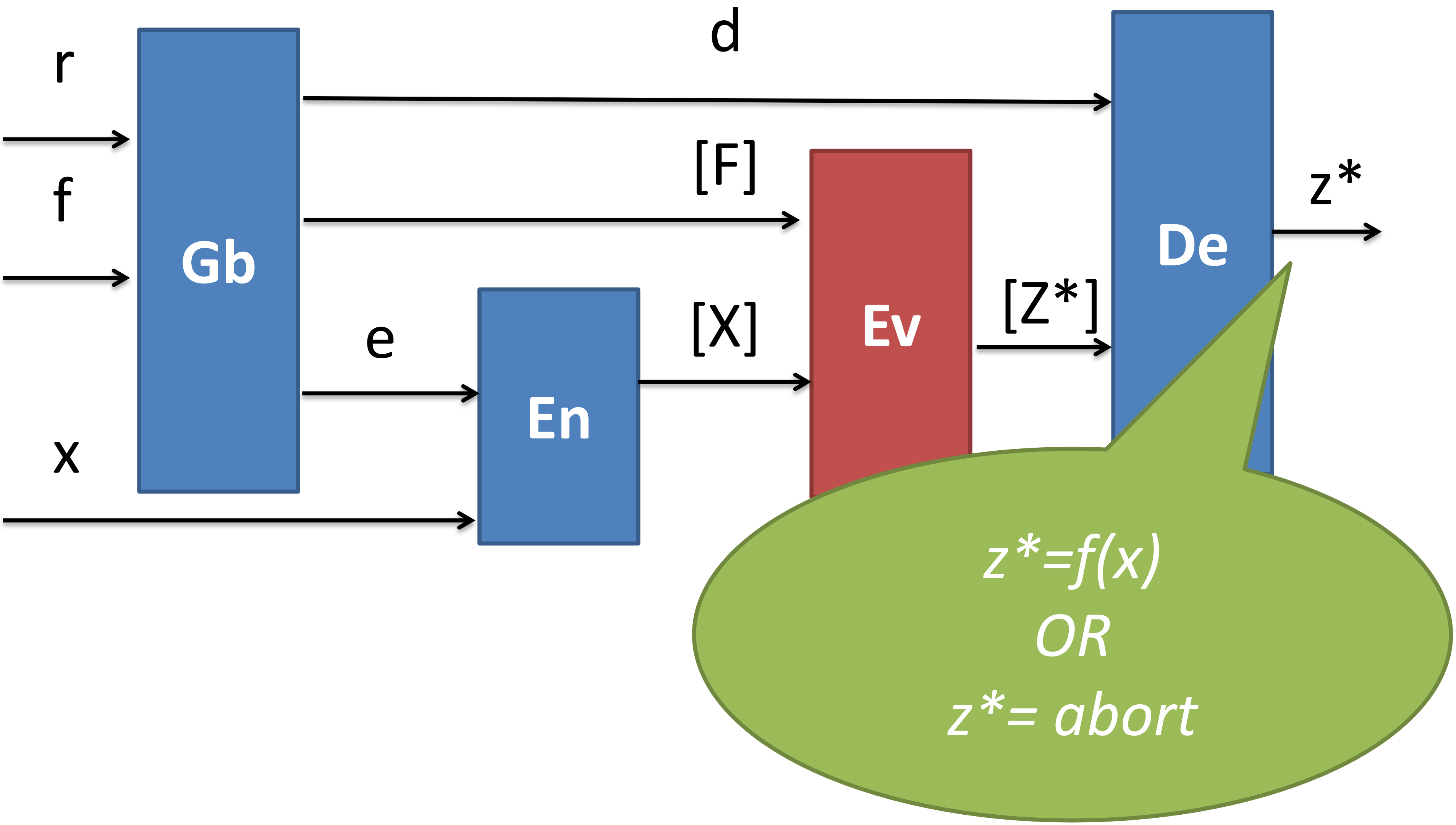
Application 1: One-time Delegation via GC



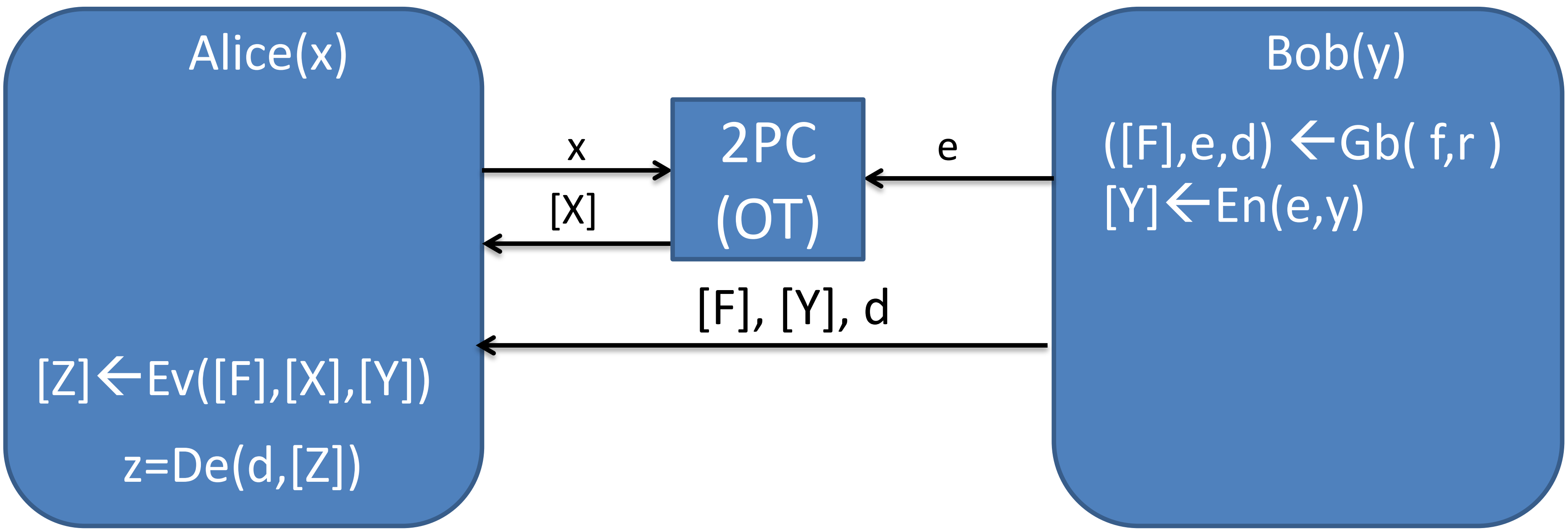
Application 1: One-time Delegation via GC



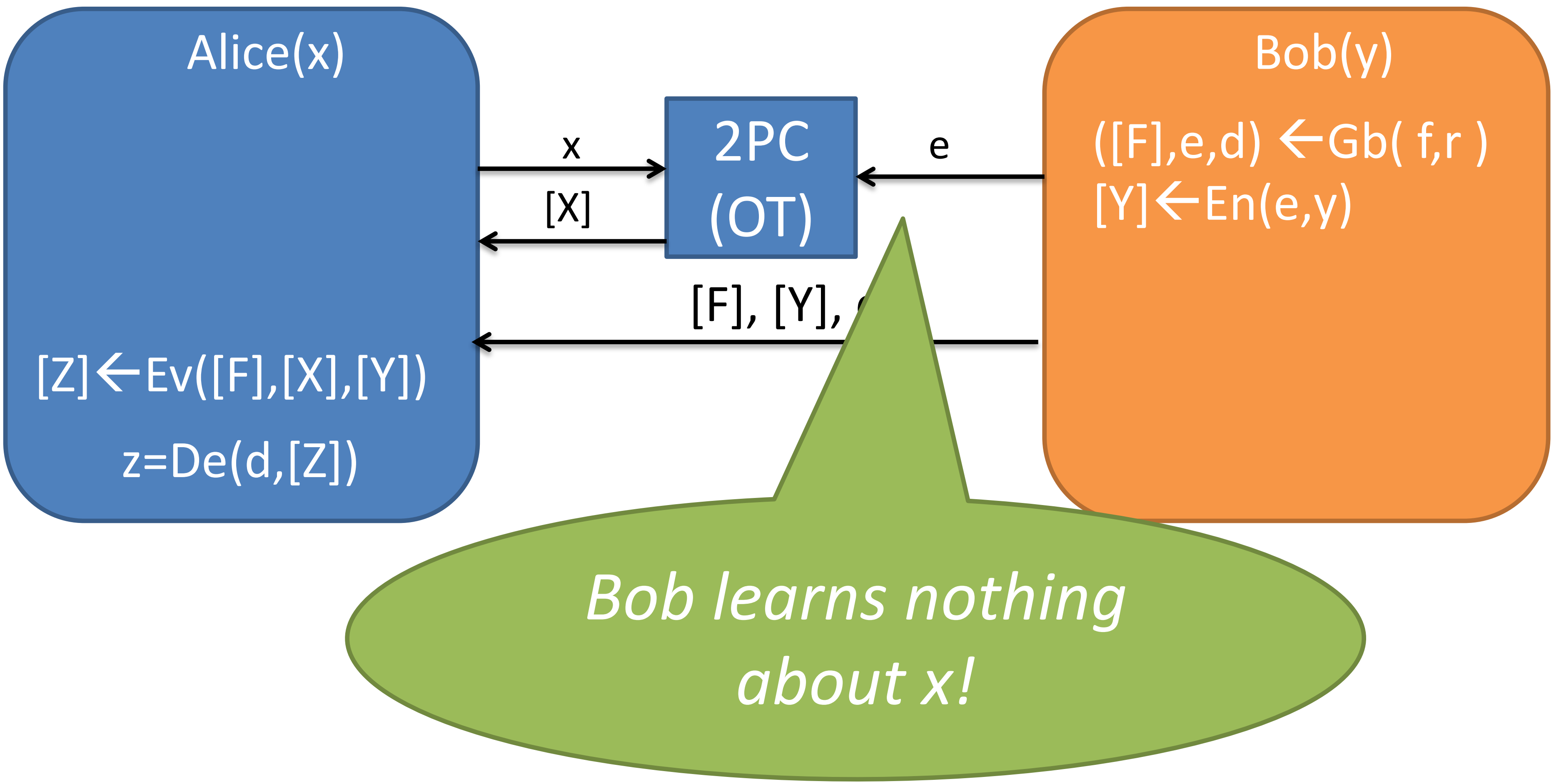
Garbled Circuits: Authenticity



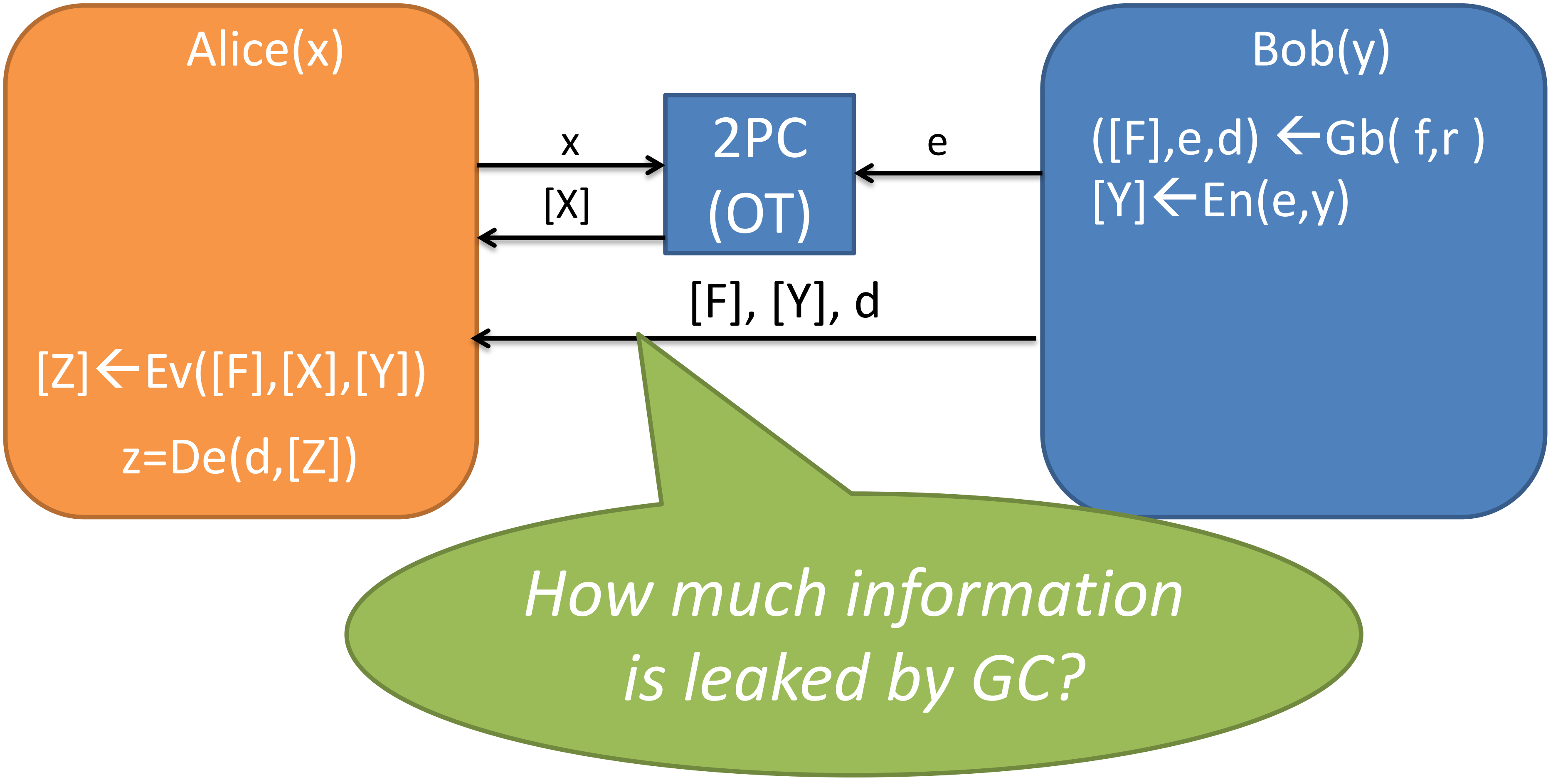
Application 2:
Passive Constant Round 2PC (Yao)



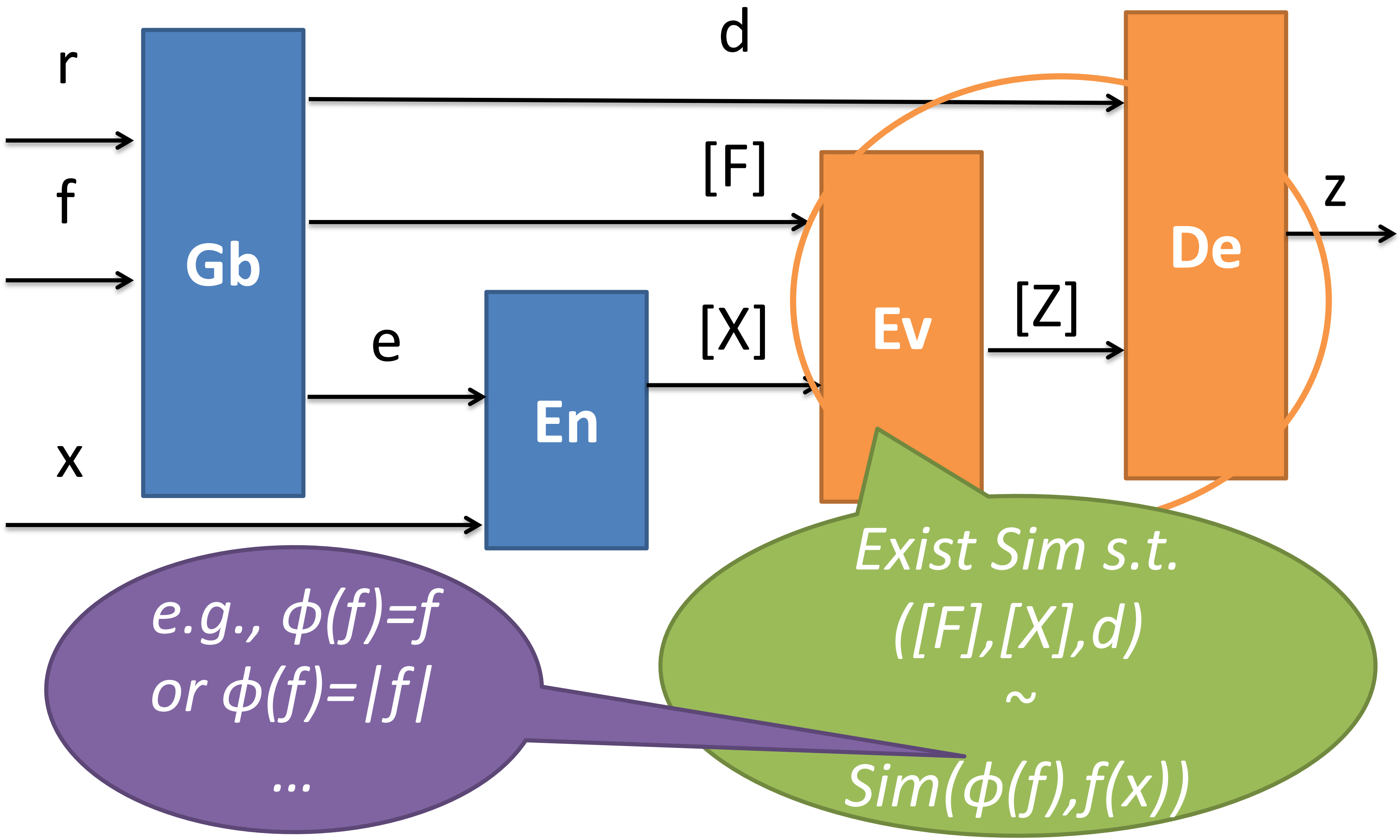
Application 2:
Passive Constant Round 2PC (Yao)



Application 2:
Passive Constant Round 2PC (Yao)



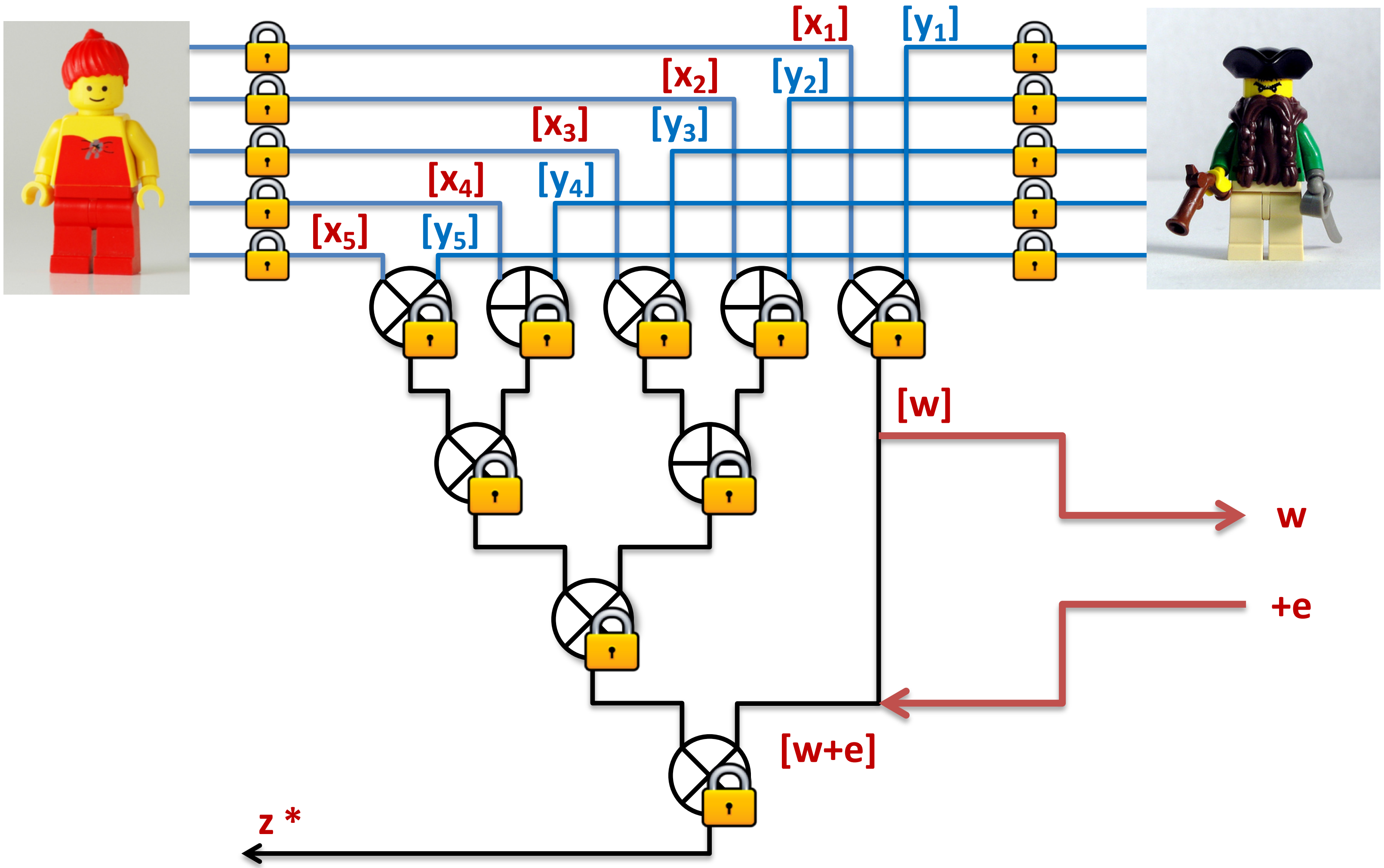
Garbled Circuits: Privacy



Part 3: Garbled Circuits

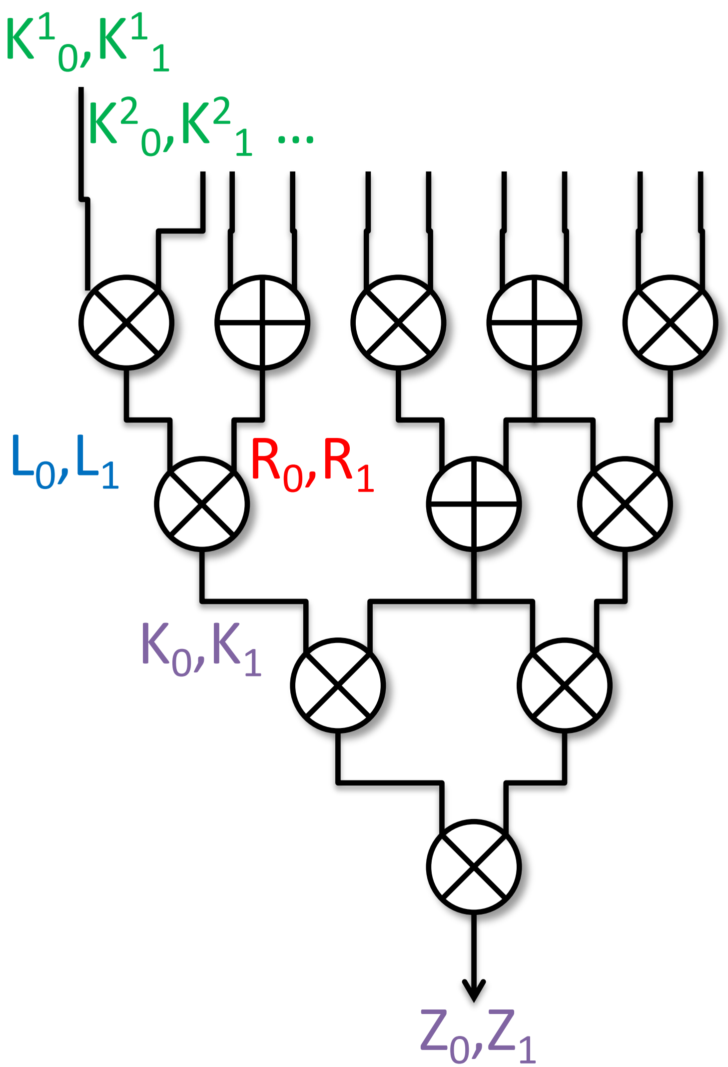
- Definitions and Applications
- **Garbling gate-by-gate: Basic and optimizations**
- Active security 101: simple-cut-and choose, dual-execution

Garbling: Gate-by-gate



PROJECTIVE SCHEMES:
CIRCUIT BASED GARBLING/EVALUATIONS

Garbling a Circuit : ([F],e,d)← Gb(f)



- Choose 2 random keys K^i_0,K^i_1 for each wire in the circuit
  - *Input, internal and, output wires*
- For each gate g compute
  - gg ← Gb(g,L_0,L_1,R_0,R_1,K_0,K_1)
- Output
  - e=(K^i_0,K^i_1) for all input wires
  - d=(Z_0,Z_1)
  - [F]=(gg^i) for all gates i

Encoding and Decoding

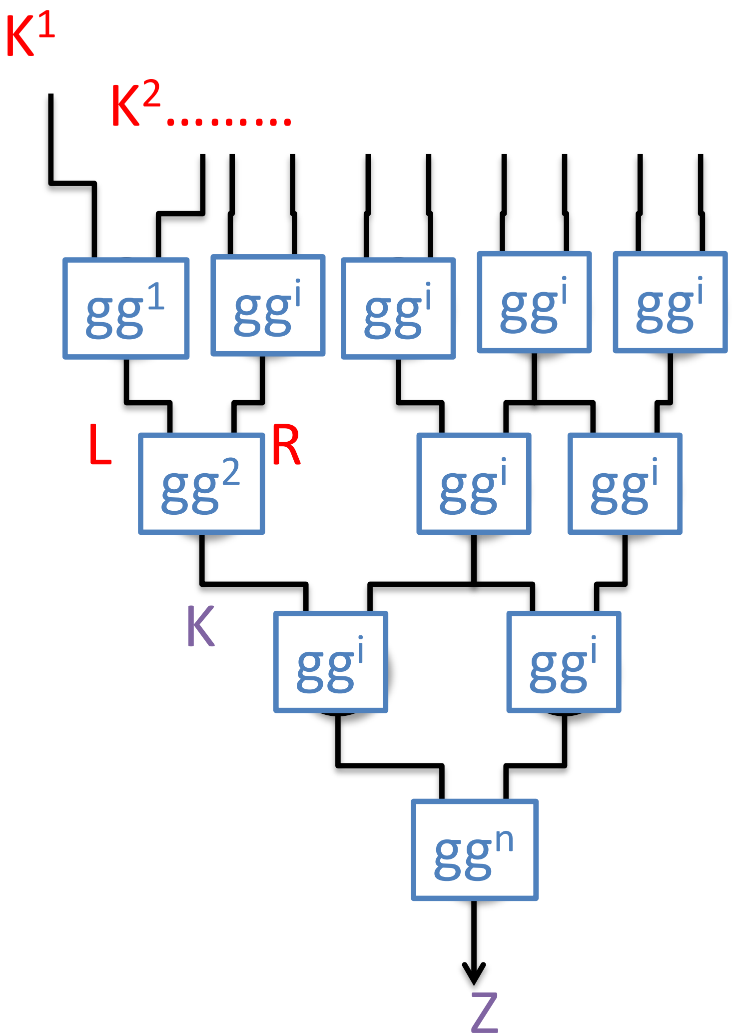
$[X] = En(e, x)$

- $e = \{ K^i_0, K^i_1 \}$
- $x = \{ x_1, \ldots, x_n \}$
- $[X] = \{ K^1_{x1}, \ldots, K^n_{xn} \}$

$z = De(d, [Z])$

- $d = \{ Z_0, Z_1 \}$
- $[Z] = \{ K \}$
- $z =$
  - 0 if $K = Z_0$,
  - 1 if $K = Z_1$,
  - "abort" else

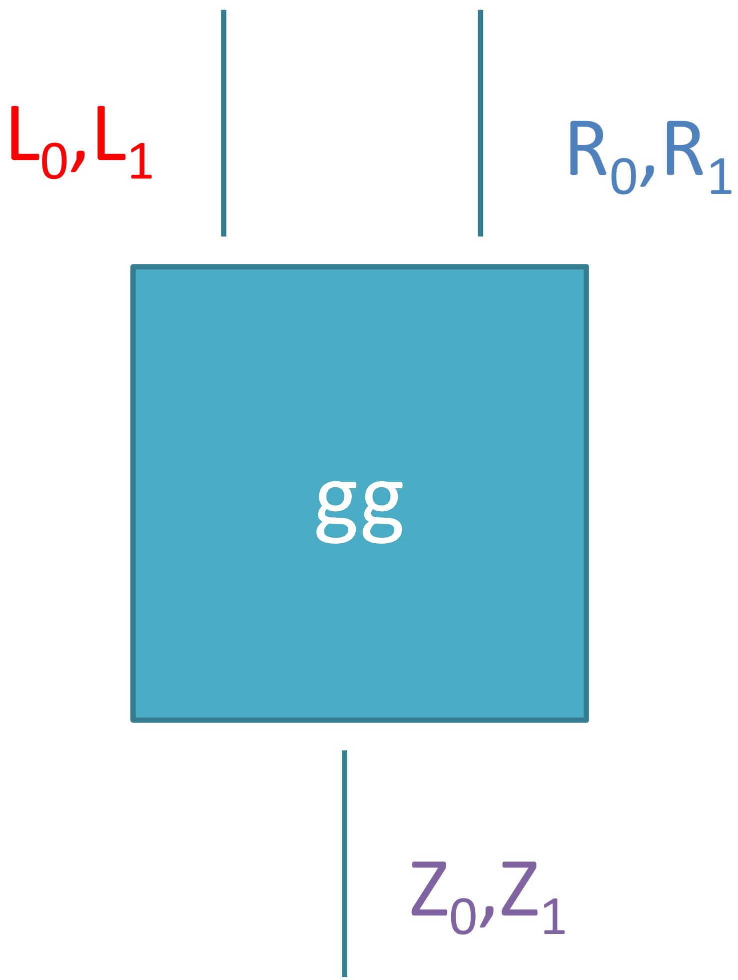
Evaluating a GC : [Z] ← Ev([F],[X])



- Parse [X]={K^1,…,K^n}
- Parse [F]={gg^i}
- For each gate i compute
  - K ← Ev(gg^i,L,R)
- Output
  - Z

INDIVIDUAL GATES GARBLING/EVALUATION

Notation



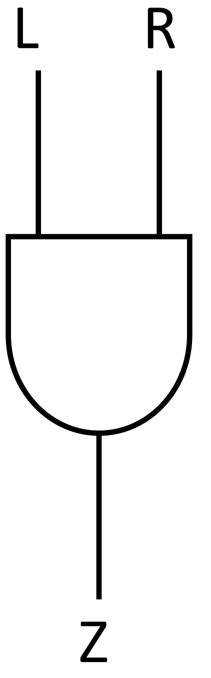
- A garbled gate is a gadget that given two inputs keys gives you the right output key *(and nothing else)*

- gg ← Gb($g, L_0, L_1, R_0, R_1, Z_0, Z_1$)
- $Z_{g(a,b)}$ ← Ev(gg, L_a, R_b)

- //and not $Z_{1-g(a,b)}$

Yao Gate Garbling (1)

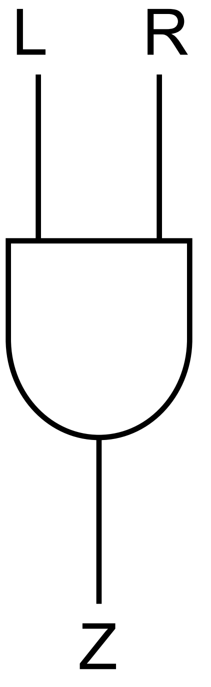
L	R	Z
0	0	0
0	1	0
1	0	0
1	1	1



- AND gate

Yao Gate Garbling (2)

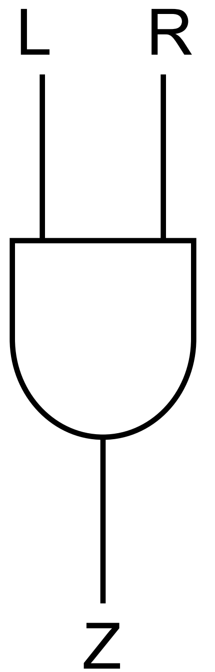
L	R	Z
L_0	R_0	Z_0
L_0	R_1	Z_0
L_1	R_0	Z_0
L_1	R_1	Z_1



- Choose labels (e.g., 128 bits strings) for every value on every wire

Yao Gate Garbling (3)

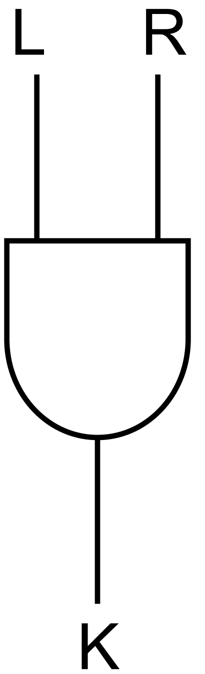
C
$C_1 = G(L_0,R_0) \oplus Z_0$
$C_2 = G(L_0,R_1) \oplus Z_0$
$C_3 = G(L_1,R_0) \oplus Z_0$
$C_4 = G(L_1,R_1) \oplus Z_1$



- Encrypt the output key with the input keys
  - G is some “key derivation function” so that the encryption is secure

Yao Gate Garbling (4)

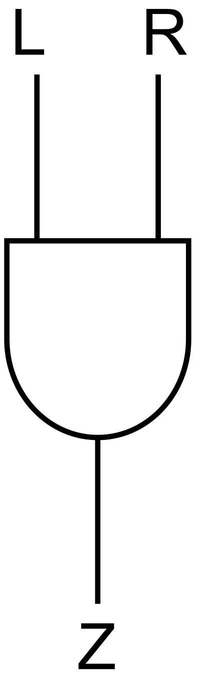
C
$C_1 = G(L_0,R_0) \oplus (Z_0,0^k)$
$C_2 = G(L_0,R_1) \oplus (Z_0,0^k)$
$C_3 = G(L_1,R_0) \oplus (Z_0,0^k)$
$C_4 = G(L_1,R_1) \oplus (Z_1,0^k)$



- Add redundancy (later used to check if decryption is successful)

Yao Gate Garbling (5)

C
$C_1 = G(L_0,R_0) \oplus (Z_0,0^k)$
$C_2 = G(L_0,R_1) \oplus (Z_0,0^k)$
$C_3 = G(L_1,R_0) \oplus (Z_0,0^k)$
$C_4 = G(L_1,R_1) \oplus (Z_1,0^k)$



$C'_1,C'_2,C'_3,C'_4 = \text{perm}(C_1,C_2,C_3,C_4)$

- Permute the order of the ciphertexts (to hide information about inputs/outputs)

Yao Gate Evaluation (1)

Eval(gg, L_a, R_b) //not a,b

- For i=1..4
  - $(K,t)=C'_i \oplus G(L_a,R_b)$
  - If $t=0^k$ output K

gg (permuted)
$C_1 = G(L_0,R_0) \oplus (K_1,0^k)$
$C_2 = G(L_0,R_1) \oplus (K_1,0^k)$
$C_3 = G(L_1,R_0) \oplus (K_1,0^k)$
$C_4 = G(L_1,R_1) \oplus (K_0,0^k)$

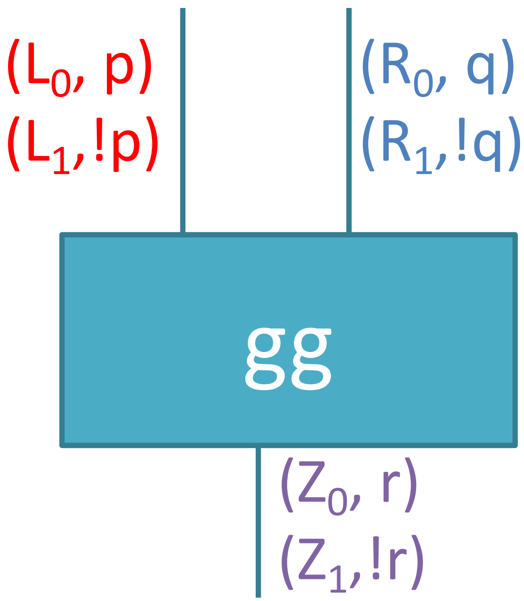
- Output is correct:
  - $t=0^k$ only for right row
- Evaluator learns nothing else:
  - Encryption + permutation

Efficiency

	|gg|	G/Gb	G/Eval	Assumption on G
Classic	8k	4	4	Standard

Point-and-permute

- Problem: Evaluator needs to try to decrypt all 4 rows
- Solution: add permutation bits to keys

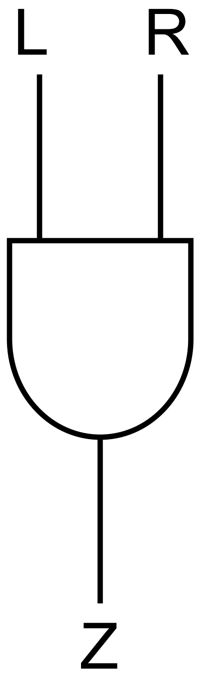


$gg \leftarrow Gb(g, L_0, L_1, p, R_0, R_1, q, Z_0, Z_1, r)$

$(Z_{g(a,b)}, r \oplus ab) \leftarrow Ev(gg, L_a, a \oplus p, R_b, b \oplus q)$

Point-and-permute Garbling (4)

C
$C_1 = G(L_0,R_0) \oplus (Z_0, r \oplus 0)$
$C_2 = G(L_0,R_1) \oplus (Z_0, r \oplus 0)$
$C_3 = G(L_1,R_0) \oplus (Z_0, r \oplus 0)$
$C_4 = G(L_1,R_1) \oplus (Z_1, r \oplus 1)$



- Remove redundancy
- Add random permutation bit

Point-and-permute Garbling (5)

C
$C'_0 = G(L_p, R_q) \oplus (Z_{p \cdot q},\ r \oplus p \cdot q)$
$C'_1 = G(L_p, R_{!q}) \oplus (Z_{p \cdot !q},\ r \oplus p \cdot !q)$
$C'_2 = G(L_{!p}, R_q) \oplus (Z_{!p \cdot q},\ r \oplus !p \cdot q)$
$C'_3 = G(L_{!p}, R_{!q}) \oplus (Z_{p \cdot !q},\ r \oplus !p \cdot !q)$

- Permute rows using p,q

Point-and-permute Evaluation

Eval(gg, L, u, R, v) //not a,b

- (Z,r)=$C'_{2u+v} \oplus G(L,R)$

C
$C'_0 = G(L_p,R_q) \oplus (Z_{p\cdot q},\ r \oplus p\cdot q)$
$C'_1 = G(L_p,R_{!q}) \oplus (Z_{p\cdot !q},\ r \oplus p\cdot !q)$
$C'_2 = G(L_{!p},R_q) \oplus (Z!_{p\cdot q},\ r \oplus !p\cdot q)$
$C'_3 = G(L_{!p},R_{!q}) \oplus (Z_{!p\cdot !q},\ r \oplus !p\cdot !q)$

- Output is correct:
  - (Check permutation)
- Privacy:
  - $u=p\oplus a$, $v=q\oplus b$
  - p,q are “one time pads” for a,b

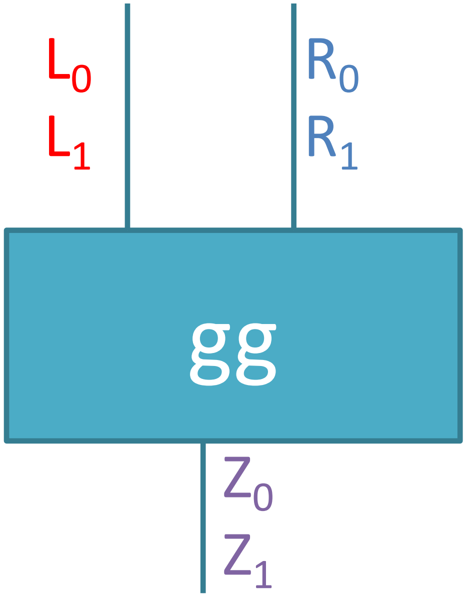
Efficiency

	|gg|	G/Gb	G/Eval	Assumption on G
Classic	8k	4	4	Standard
P&P	4k	4	1	Standard

GARBLING OPTIMIZATIONS:
SIMPLE GARBLED ROW REDUCTION

Changing the syntax

- Problem: each gg is 4 ciphertexts
- Solution: define output key pseudorandomly as functions of input keys, reduce comm. complexity

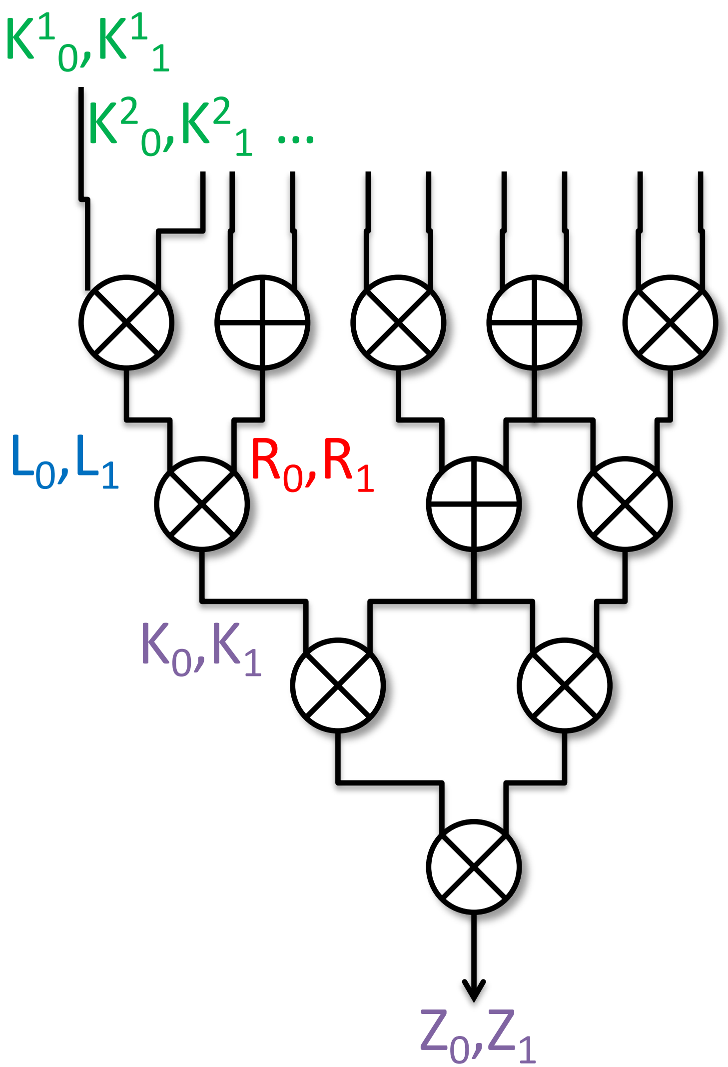


$(gg, Z_0, Z_1) \leftarrow Gb(g, L_0, L_1, R_0, R_1)$

$(Z_{g(a,b)}) \leftarrow Ev(gg, L_a, R_b)$

Note, now garbling cannot be done in parallel anymore!

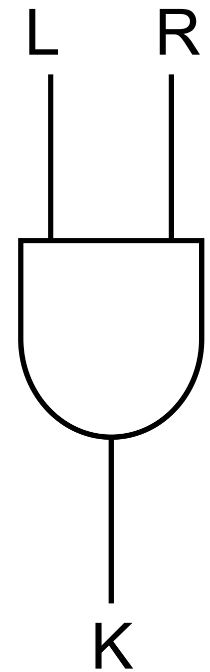
Garbling a Circuit : ([F],e,d)← Gb(f)



- Choose 2 random keys K^i_0,K^i_1 for each wire in the circuit
  - *Input wire only!*
- For each gate g compute
  - $(gg,K_0,K_1) \leftarrow Gb(g,L_0,L_1,R_0,R_1)$
- Output
  - $e=(K^i_0,K^i_1)$ for all input wires
  - $d=(Z_0,Z_1)$
  - $[F]=(gg^i)$ for all gates i

Yao Gate Garbling (3)

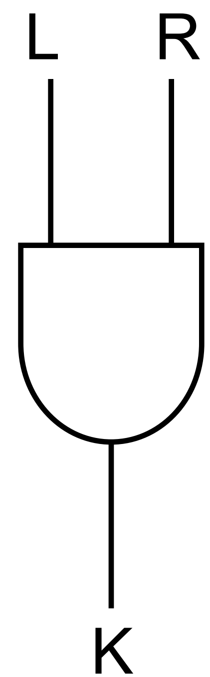
C
$C_1 = G(L_0,R_0) \oplus Z_0$
$C_2 = G(L_0,R_1) \oplus Z_0$
$C_3 = G(L_1,R_0) \oplus Z_0$
$C_4 = G(L_1,R_1) \oplus Z_1$



- Encrypt the output key with the input keys

Garbled Row Reduction Garbling

C
$Z_0 = G(L_0,R_0)$ $(C_1=0^k)$
$C_2 = G(L_0,R_1) \oplus Z_0$
$C_3 = G(L_1,R_0) \oplus Z_0$
$C_4 = G(L_1,R_1) \oplus Z_1$



- Define output keys as function of input keys
  - (compatible with p&p)
  - Can reduce 2 rows, but 1 is compatible with Free-XOR (coming up!)

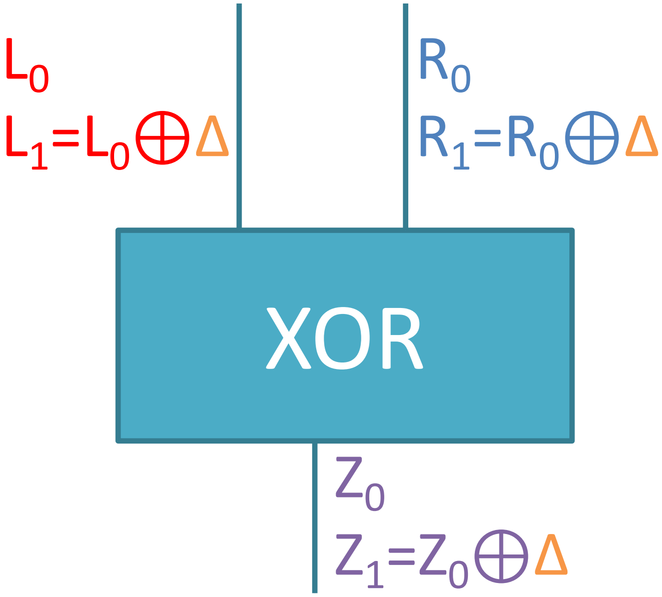
Efficiency

	|gg|	G/Gb	G/Eval	Assumption on G
Classic	8k	4	4	Standard
P&P	4k	4	1	Standard
+GRR	3k/2k	4	1	Standard

GARBLING OPTIMIZATIONS:
FREE XOR

Free-XOR

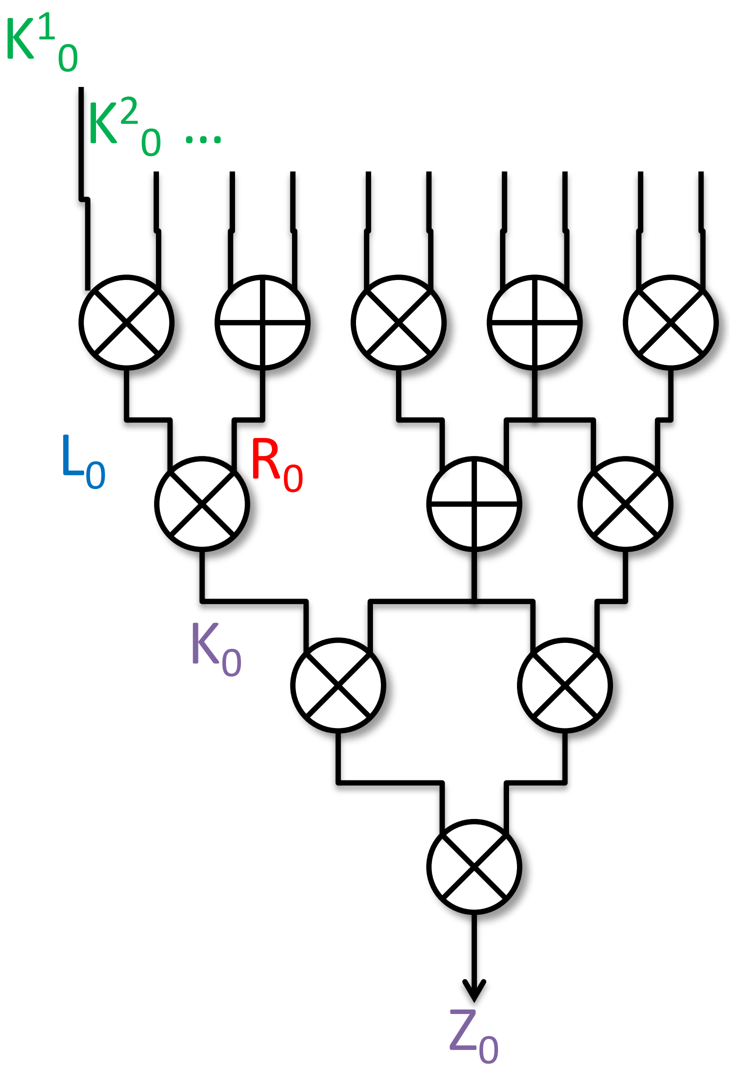
- Problem: with secret sharing linear gates are for free. What about GC?
- Solution: introduce correlation between keys, make XOR computation "free"



$(gg, Z_0) \leftarrow Gb(g, L_0, R_0, \Delta)$

$(Z_{g(a,b)}) \leftarrow Ev(gg, L_a, R_b)$

Changing syntax, again!



- Choose 1 random key K^i_0 for each input wire in the circuit
  - *And global difference* Δ
- For each gate g compute
  - $(gg, K_0) \leftarrow Gb(g, L_0, R_0, \Delta)$
- Output
  - $e=(K^i_0, K^i_1)$ for all input wires
  - $d=(Z_0, Z_1)$
  - $[F]=(gg^i)$ for all gates i

What about AND Gates?

- Like before, but requires “circular security assumption”

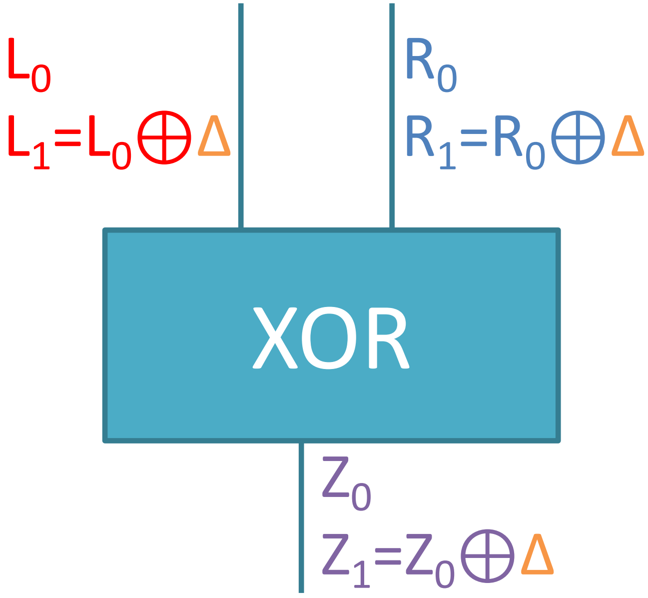
- Evaluator sees

$$L_0,\ R_0,\ Z_0, \text{ and}$$

$$G(L_0 \oplus \Delta,\ R_0 \oplus \Delta) \oplus Z_0 \oplus \Delta$$

- And should not be able to compute Δ !
- Effectively an encryption of Δ under Δ !

Garbling/Evaluating XOR Gates



$(gg, Z_0) \leftarrow Gb(g, L_0, R_0, \Delta)$

$Z_{a \oplus b} \leftarrow Ev(gg, L_a, R_b)$

$Gb(XOR, L_0, R_0, \Delta)$

- Output $Z_0 = L_0 \oplus R_0$
- (gg is empty)

$Ev(XOR, L_a, R_b, \Delta)$

- Output $Z_{a \oplus b} = L_a \oplus R_b$

$$L_a \oplus R_b = L_0 \oplus a\Delta \oplus R_0 \oplus b\Delta = Z_0 \oplus (a \oplus b)\Delta = Z_{a \oplus b}$$

Efficiency

	AND			XOR			
	|gg|	G/Gb	G/Eval	|gg|	G/Gb	G/Eval	Assumption on G
Classic	8k	4	4	8k	4	4	Standard
P&P	4k	4	1	4k	4	1	Standard
+GRR	3k/2k	4	1	3k/2k	4	1	Standard
+Free-XOR	3k	4	1	0	0	0	Circular

Privacy Free Garbing

- In some application (example, the delegation of computation from the first slide) we don't care about hiding the input/output of the circuit to the evaluator.

- Can we construct more efficient garbling if we don't care about *privacy*, but only *authenticity*?

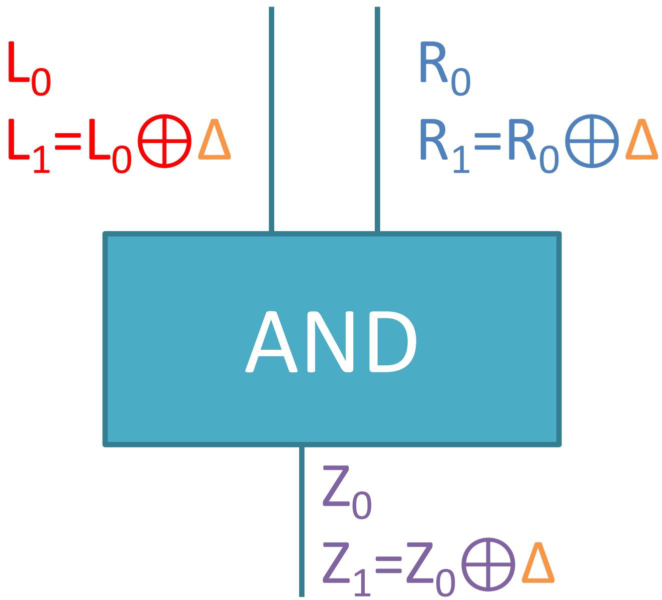
Privacy Free with Free XOR

- For XOR-gates it is hard to do better than *free*-XOR

- What about AND gates?

- Let c=AND(a,b) then
  - If a = 0 → c = 0
  - If a =1 → c = b

Privacy-Free AND gates



$(gg, Z_0) \leftarrow Gb(g, L_0, R_0, \Delta)$

$Z_{ab} \leftarrow Ev(gg, L_a, a, R_b, b)$

$Gb(AND, L_0, R_0, \Delta)$

- $Z_0 = G(L_0)$ // GRR
- $C = G(L_1) \oplus Z_0 \oplus R_0$

$Ev(gg, L_a, a, R_b, b, \Delta)$

- If $a=0$: output
  $Z_0 = G(L_0)$
- If $a=1$: output
  $Z_b = C \oplus G(L_1) \oplus R_b$

$$Z_b = C \oplus G(L_1) \oplus R_b = Z_0 \oplus R_0 \oplus (R_0 \oplus b\Delta) = Z_b$$

Efficiency

	AND			XOR			
	|gg|	G/Gb	G/Eval	|gg|	G/Gb	G/Eval	Assumption on G
Classic	8k	4	4	8k	4	4	Standard
P&P	4k	4	1	4k	4	1	Standard
+GRR	3k/2k	4	1	3k/2	4	1	Standard
+Free-XOR	3k	4	1	0	0	0	Circular
Privacy-Free*	k	4	1	0	0	0	Circular

Privacy-Free Garbling: Extension?

- Can the same trick help us also in garbling for 2PC?
- Can we let the evaluator learn the bit of some internal wires?
  - No!
- Can we let the evalutor learn a one-time pad encryption of some internal wire?
  - Sure, why not!

Half-Gate (Two Halves Make a Whole)

- Note that we can write:

$$a \cdot b = (a \cdot r) \oplus (a \cdot (r \oplus b))$$

r known to garbler → How to garble efficiently?

$r \oplus b$ known to evaluator → can use PF garbling

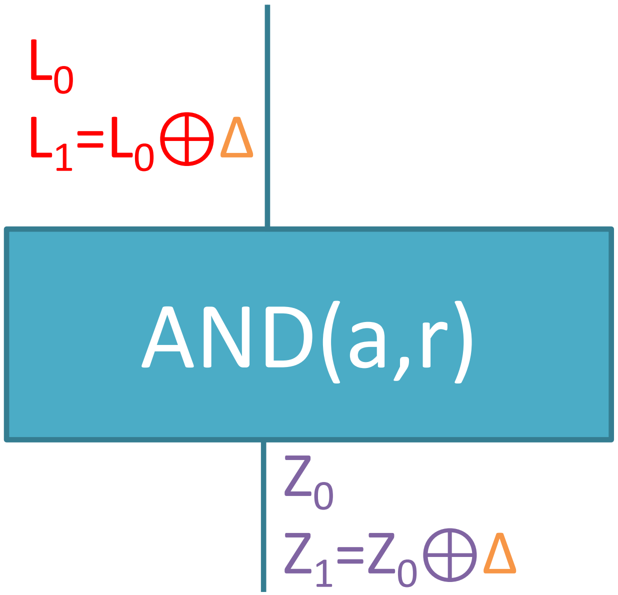
- 1 AND → 2 ANDs. How is this better?
  - The garbled can choose a random r at garbling time
  - Make sure that the evaluator learns $(r \oplus b)$
  - How? Use the permutation bits from point&permute!

How to garble with hidden constant

- The garbled knows r
  and wants to garble $c = a \cdot r$
  – If $r = 0 \rightarrow c=0$
  – If $r = 1 \rightarrow c=a$

- How to garble an unary gate, which is either the 0 gate or the identity gate depending on r?

Garbling AND with hidden-constant



$(gg, Z_0) \leftarrow Gb(g, L_0, r, \Delta)$

$Z_{ar} \leftarrow Ev(gg, L_a)$

$Gb(AND, L_0, p, \Delta)$

- $Z_p = G(L_p)$ //GRR $C_0 = 0^k$
- $C_1 = G(L_{!p}) \oplus Z_p \oplus r\Delta$

$Ev(gg, L_a, a \oplus p, \Delta)$

- Output
  $Z_{ar} = C_{a \oplus p} \oplus G(L_a)$

Let p=0 for simplicity

$$C_a \oplus G(L_a) = Z_0 \oplus a(r\Delta) = Z_{ar}$$

Efficiency

	AND			XOR			
	|gg|	G/Gb	G/Eval	|gg|	G/Gb	G/Eval	Assumption on G
Classic	8k	4	4	8k	4	4	Standard
P&P	4k	4	1	4k	4	1	Standard
+GRR	3k/2k	4	1	3k/2k	4	1	Standard
+Free-XOR	3k	4	1	0	0	0	Circular
Privacy-Free*	k	4	1	0	0	0	Circular
Half-Gate	2k	4	2	0	0	0	Circular
GLP	2k	4	1	1	4	1	Standard

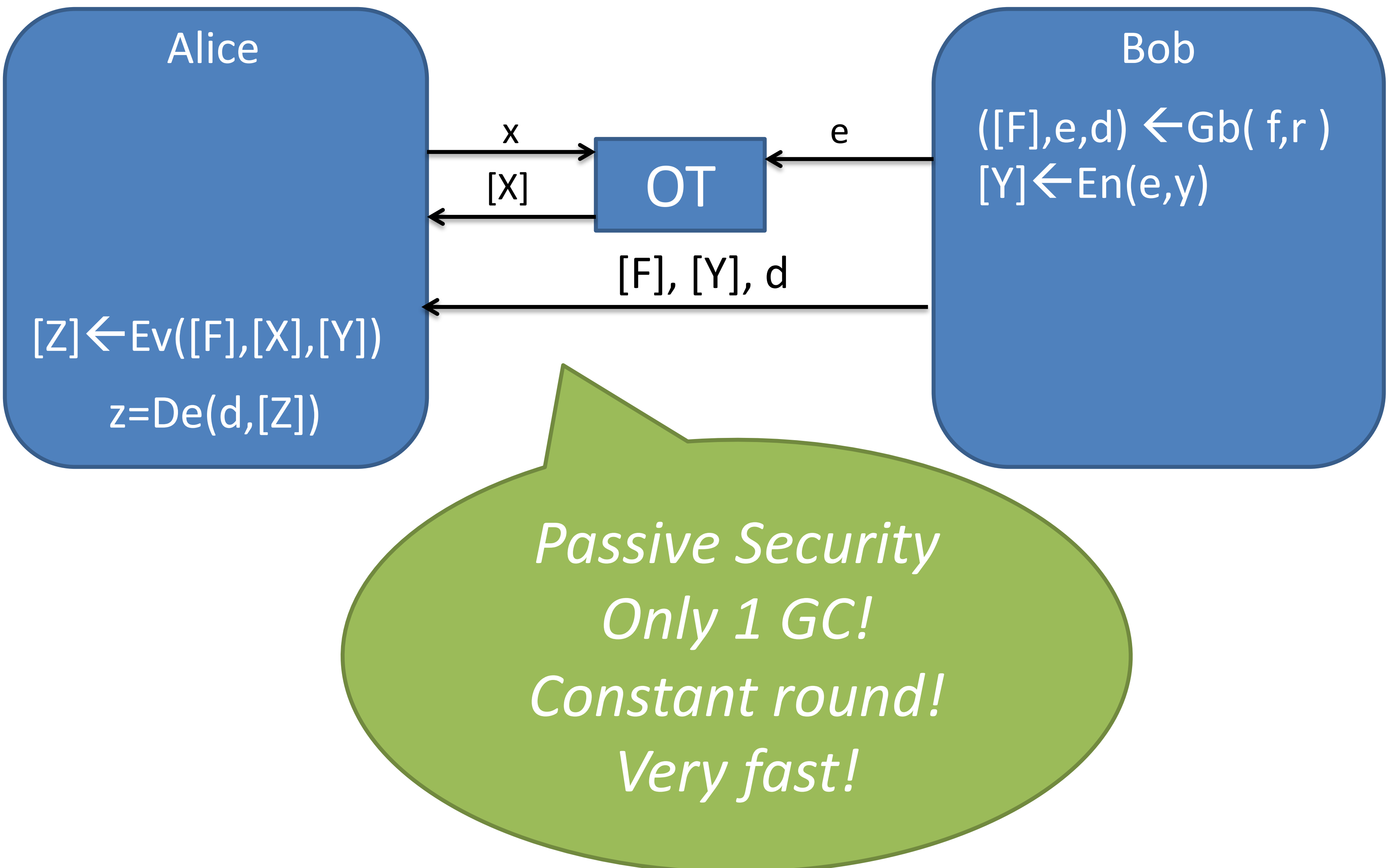
- GLP : Fast Garbling of Circuits Under Standard Assumptions (Gueron, Lindell, Pinkas)
- (The measure of G in this table is somehow arbitrary, in practice the size of the input to G makes a difference in runtime)

Part 3: Garbled Circuits

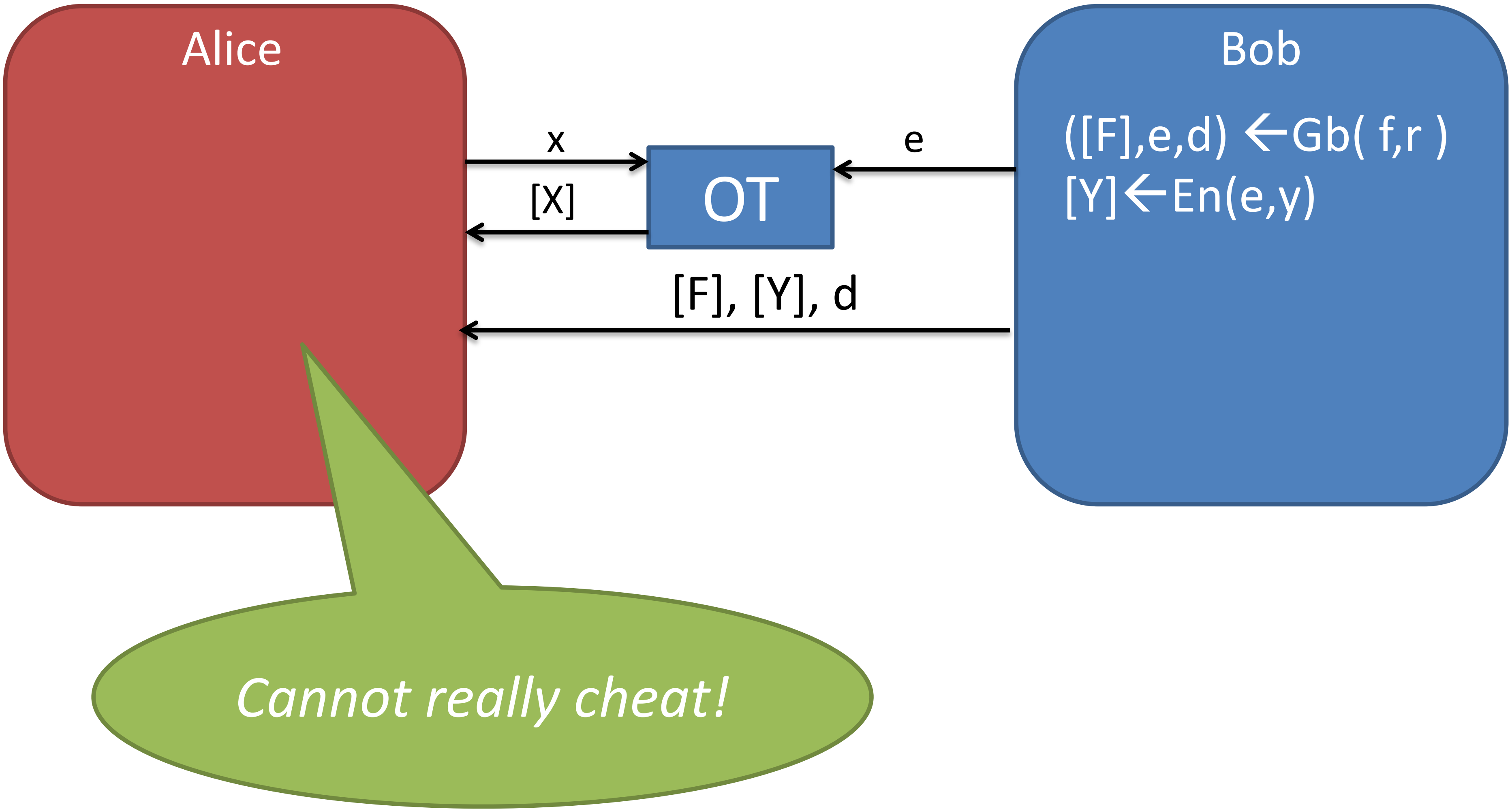
- Definitions and Applications
- Garbling gate-by-gate: Basic and optimizations
- **Active security 101: simple-cut-and choose, dual-execution**

ACTIVE ATTACKS VS YAO

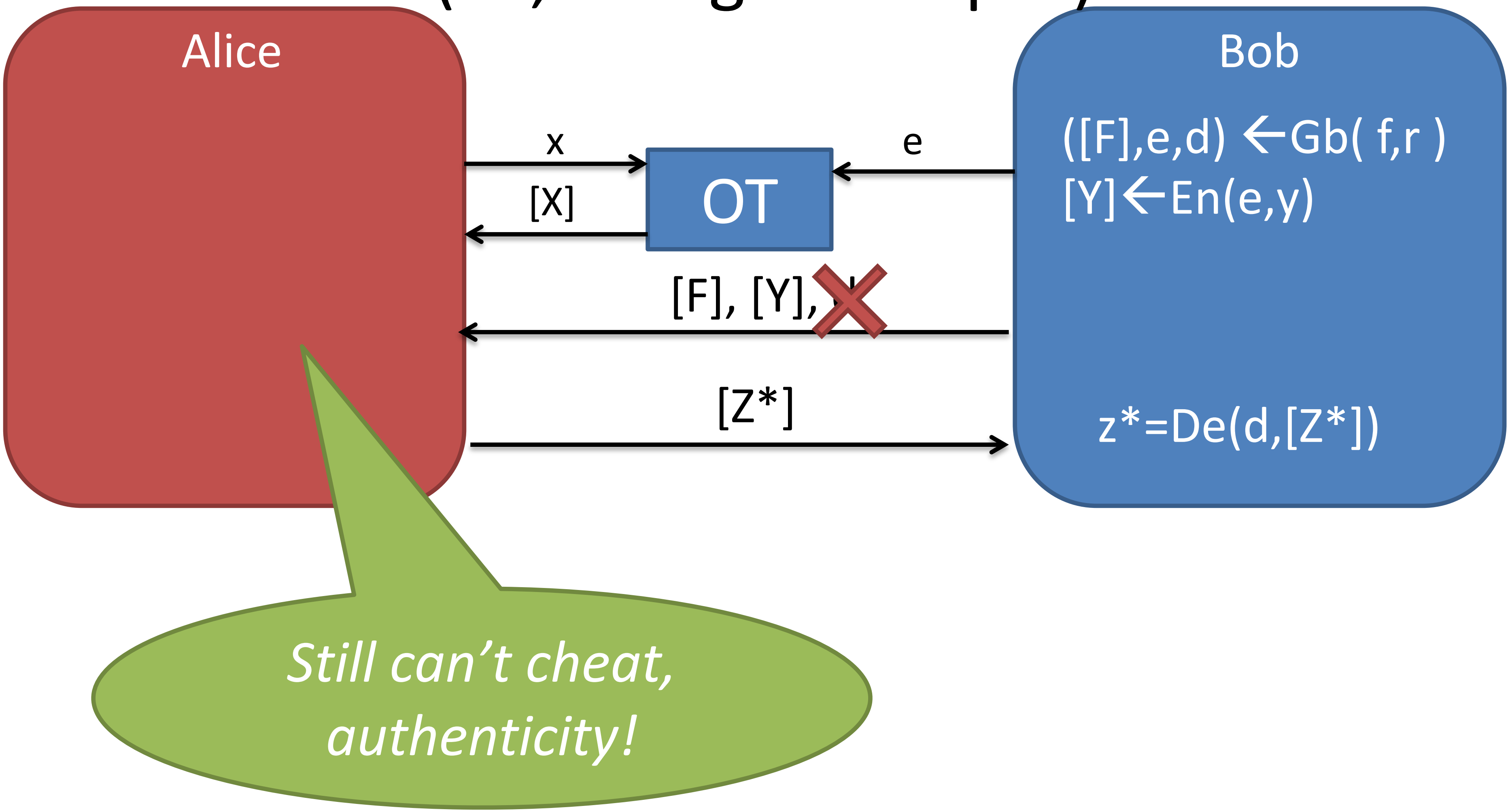
Yao´s protocol



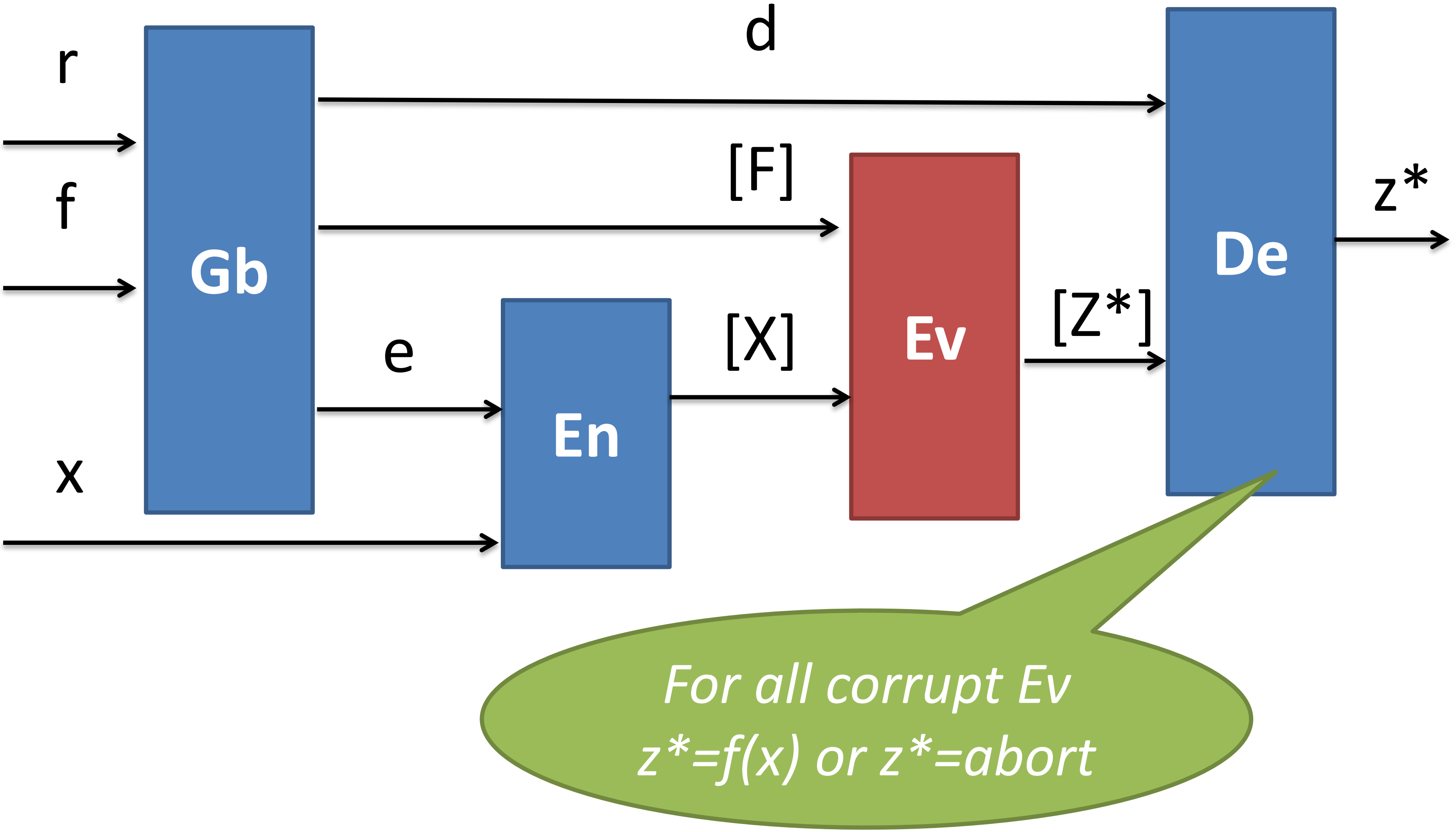
Active security of Yao



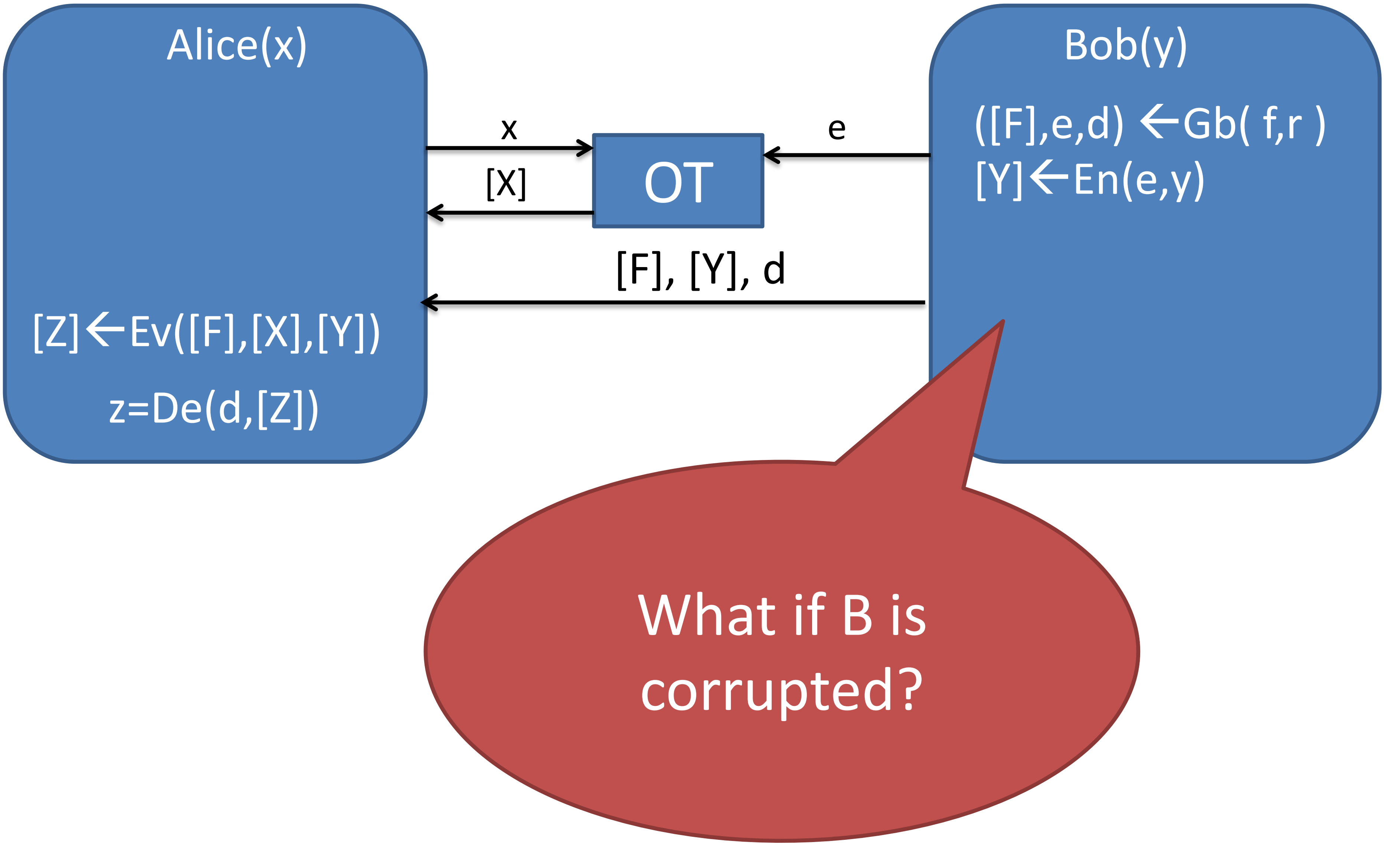
Active security of Yao (v2, Bob gets output)



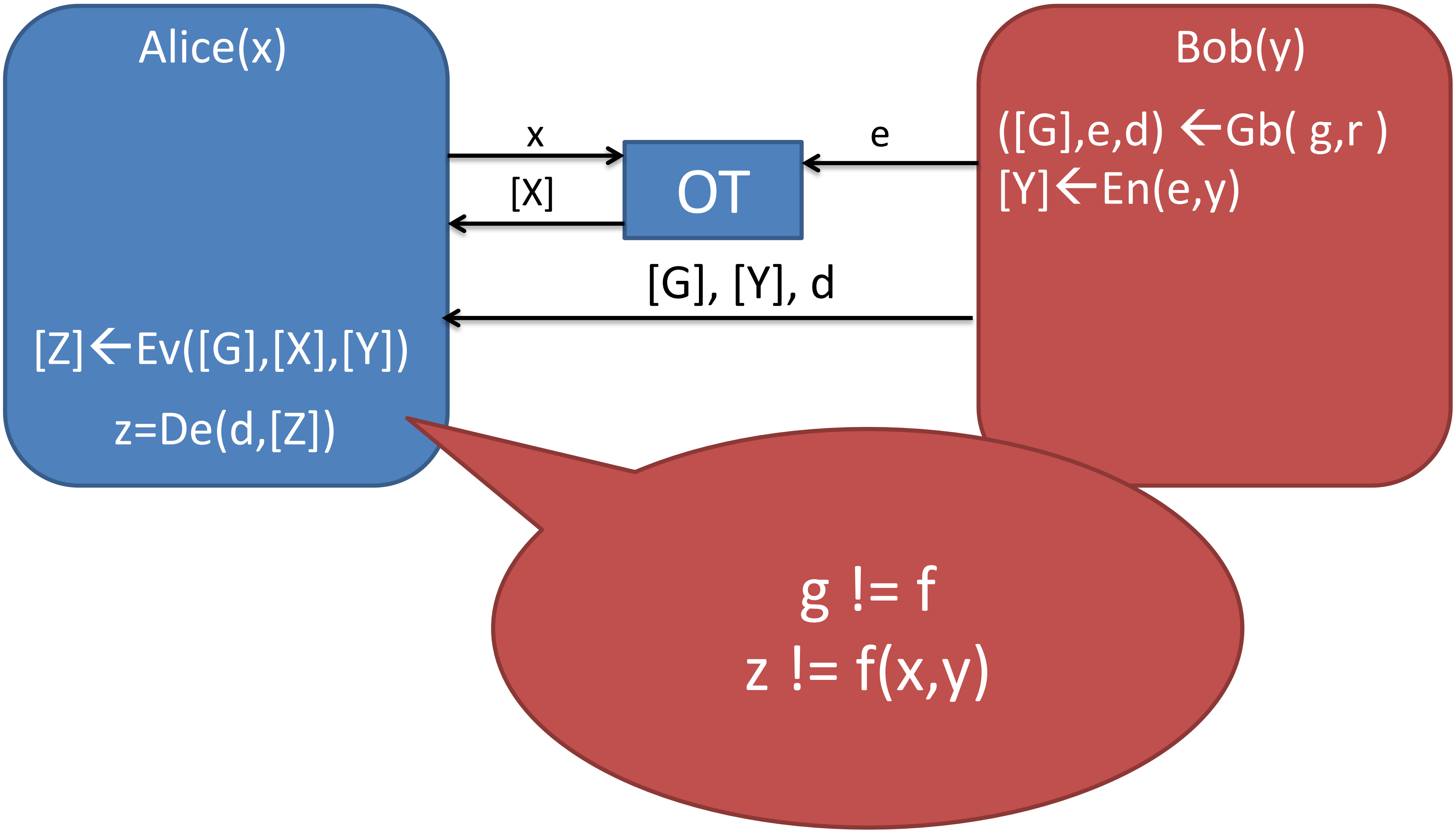
Garbled Circuits: Authenticy



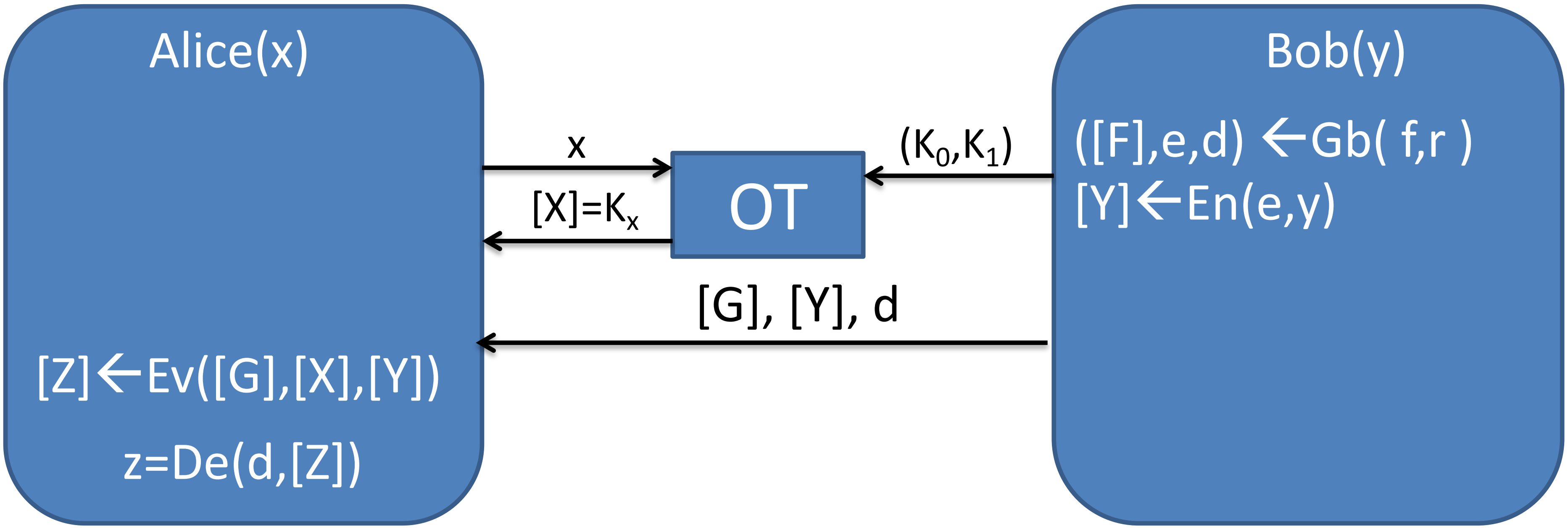
Active security of Yao



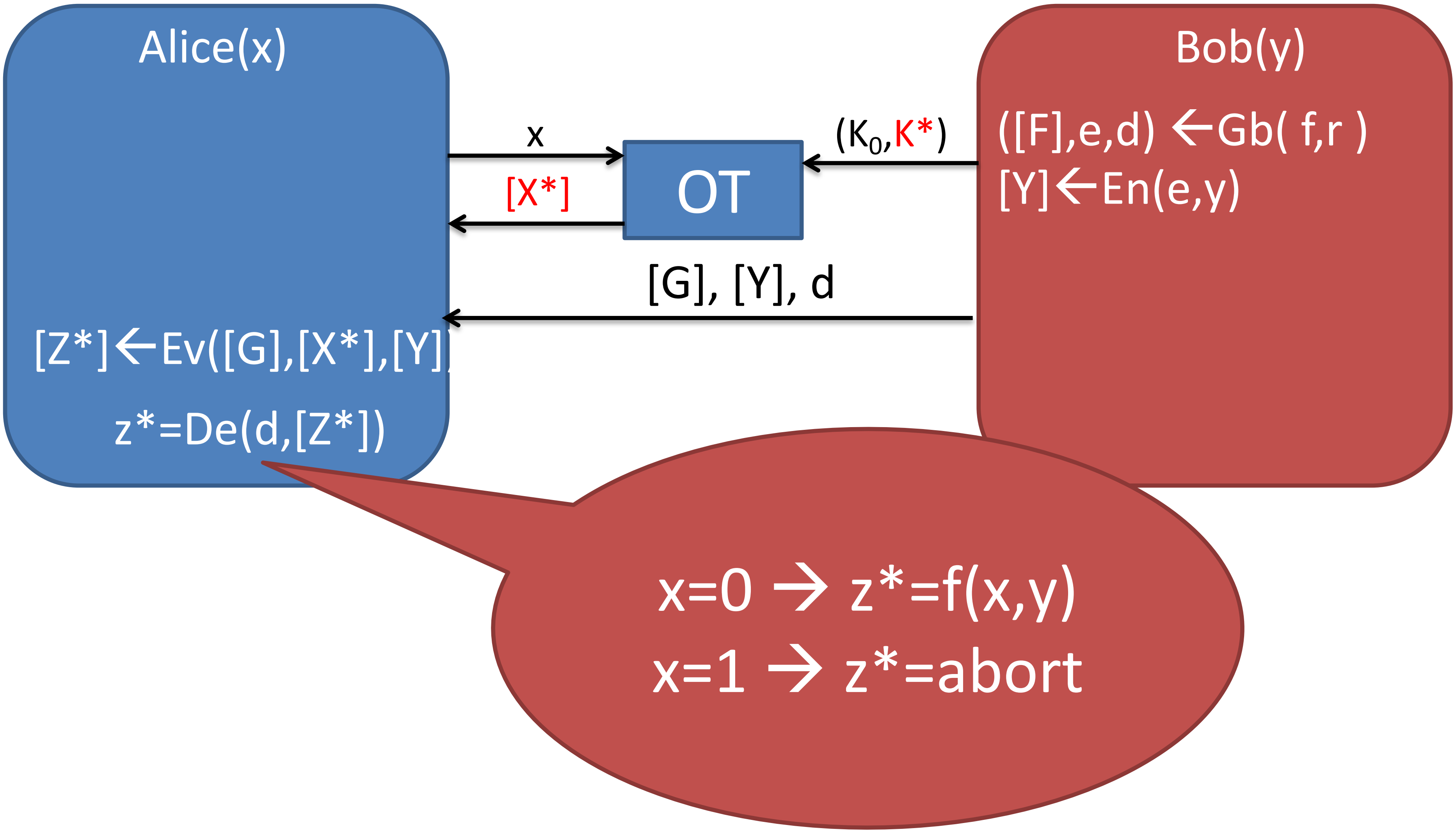
Insecurity 1 (wrong f)



Insecurity 2 (selective failure)

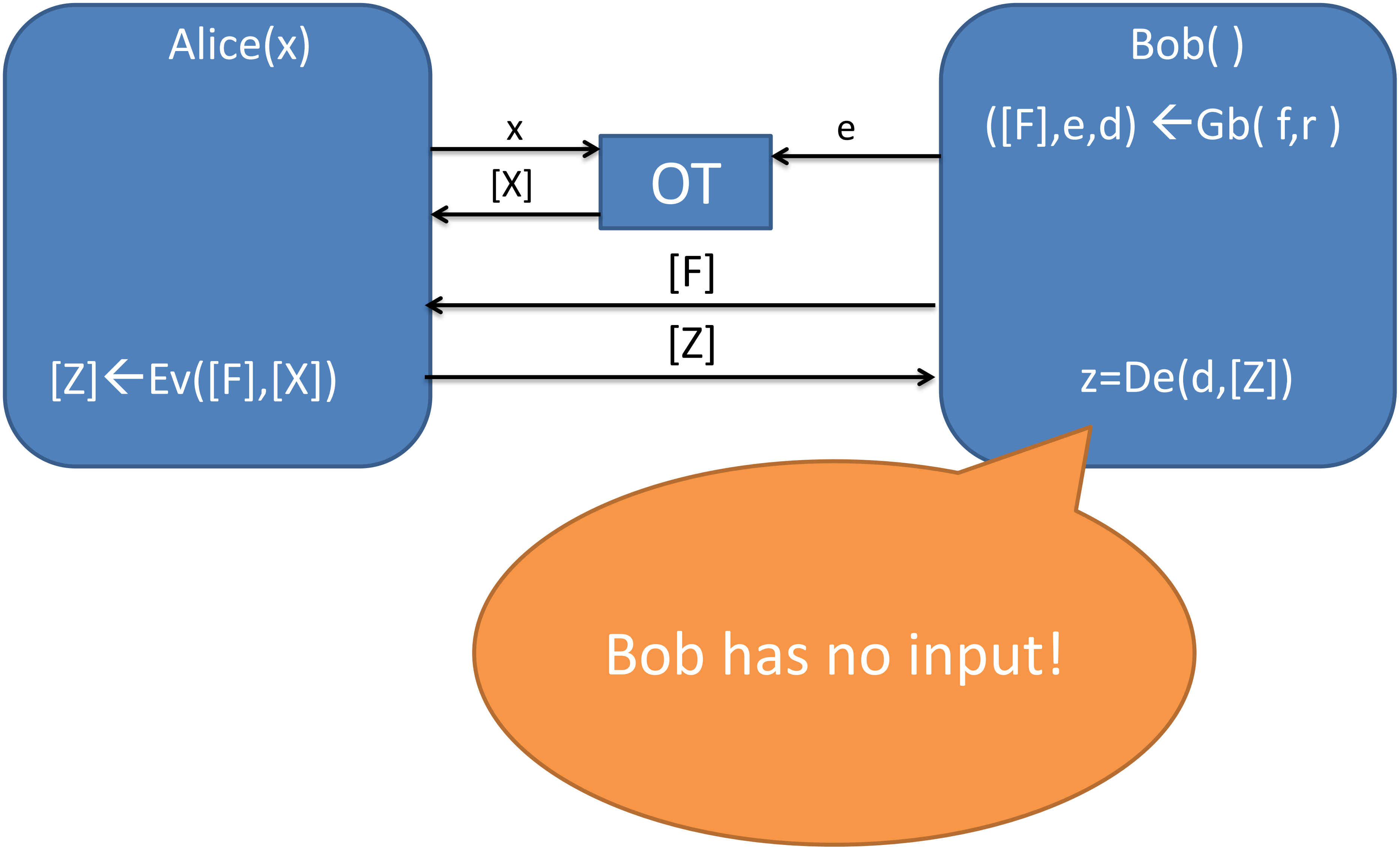


Insecurity 2 (selective failure)

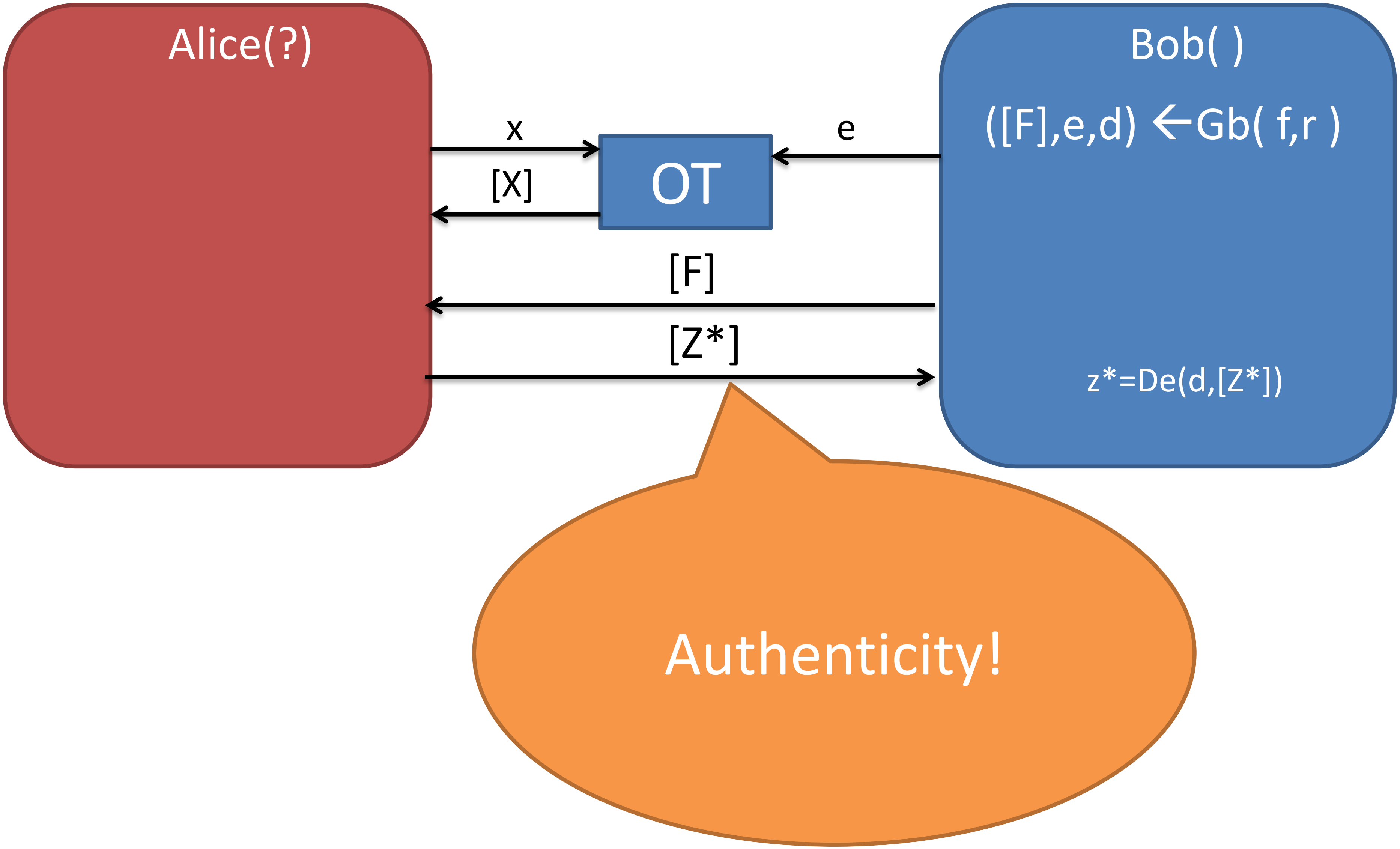


SIMPLE TRICKS FOR ACTIVE SECURITY

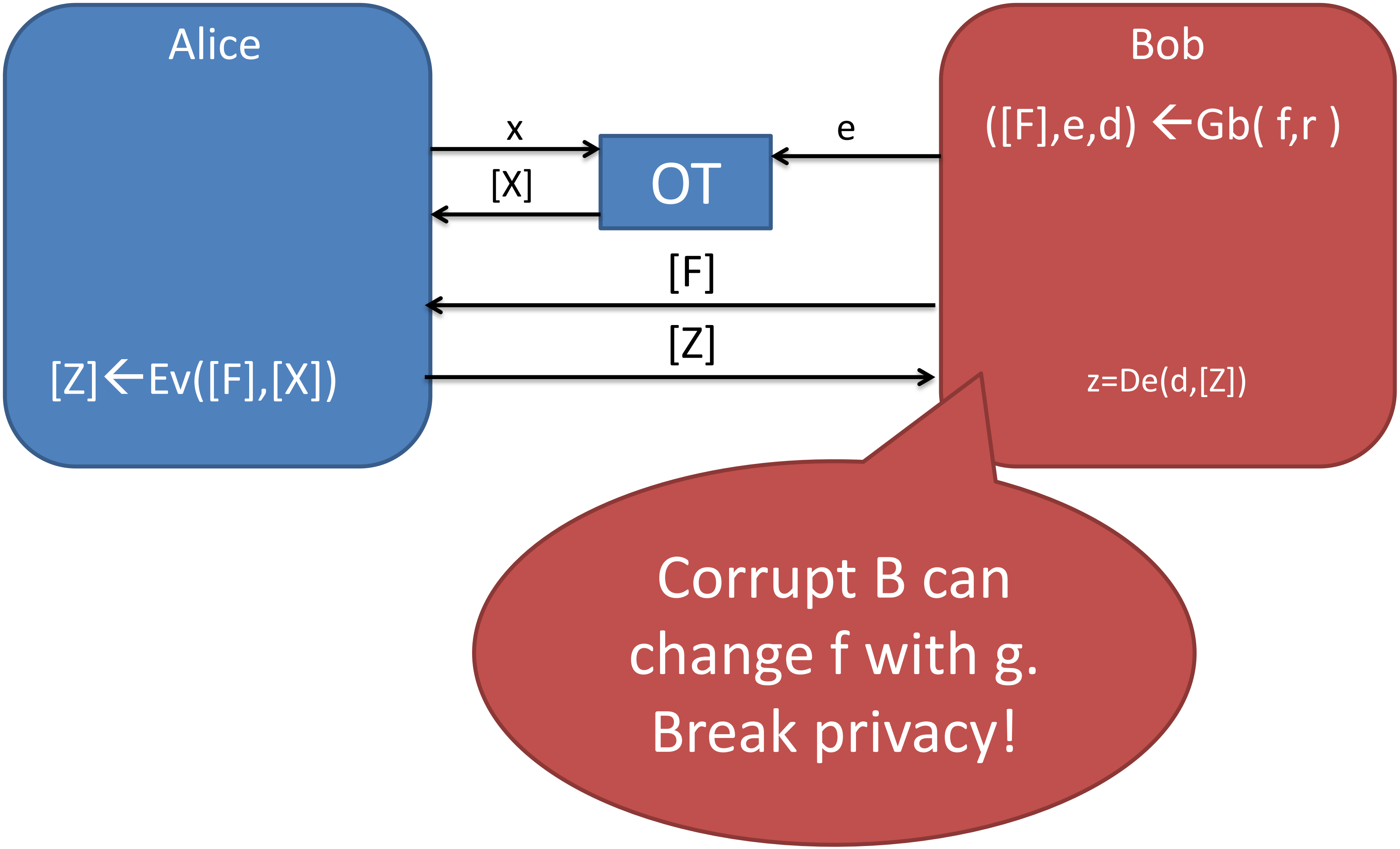
ZKGC (Alice proves f(x)=z)



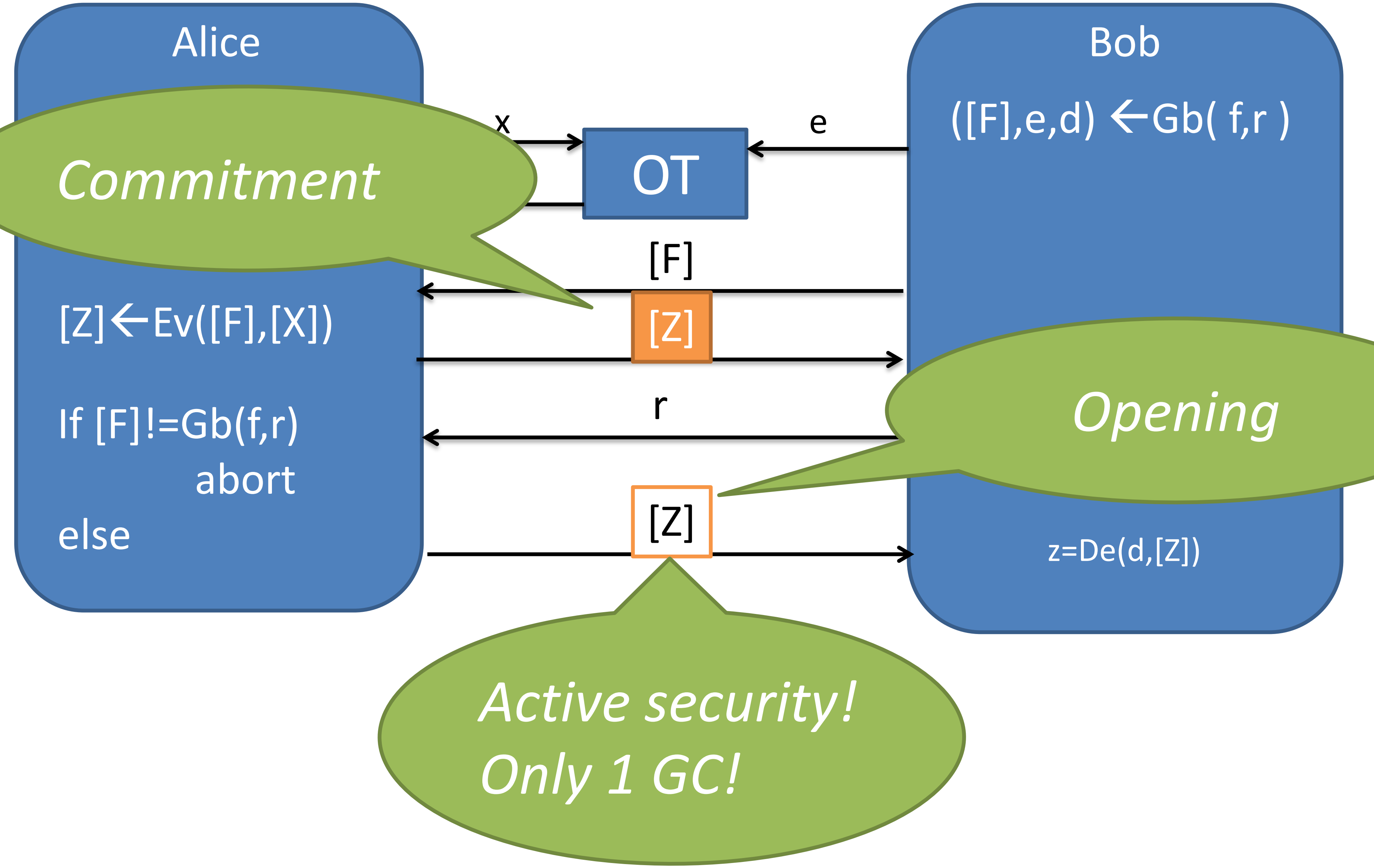
ZKGC (Alice proves f(x)=z)



ZKGC (Alice proves f(x)=z)

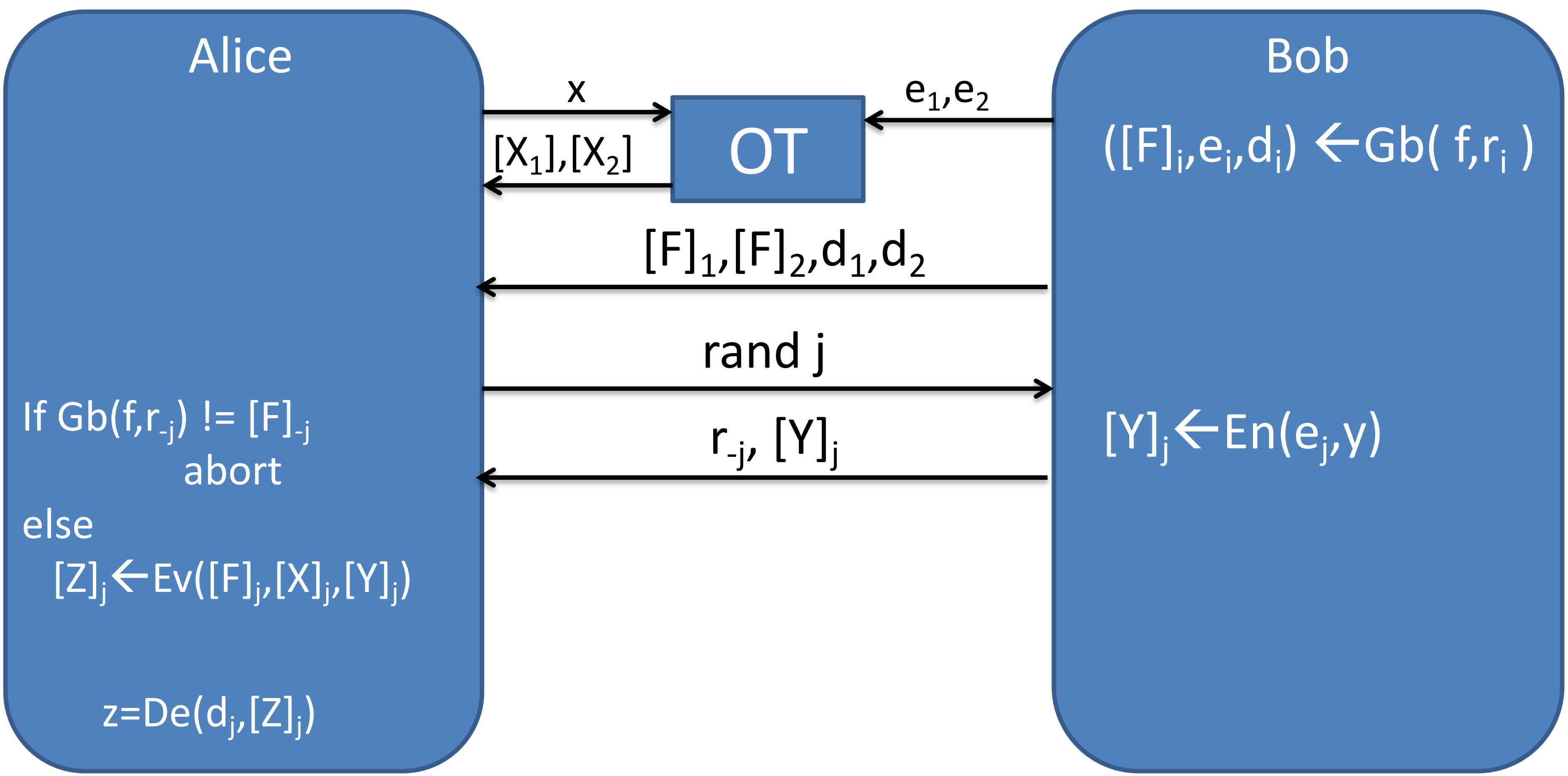


ZKGC (Alice proves f(x)=z)

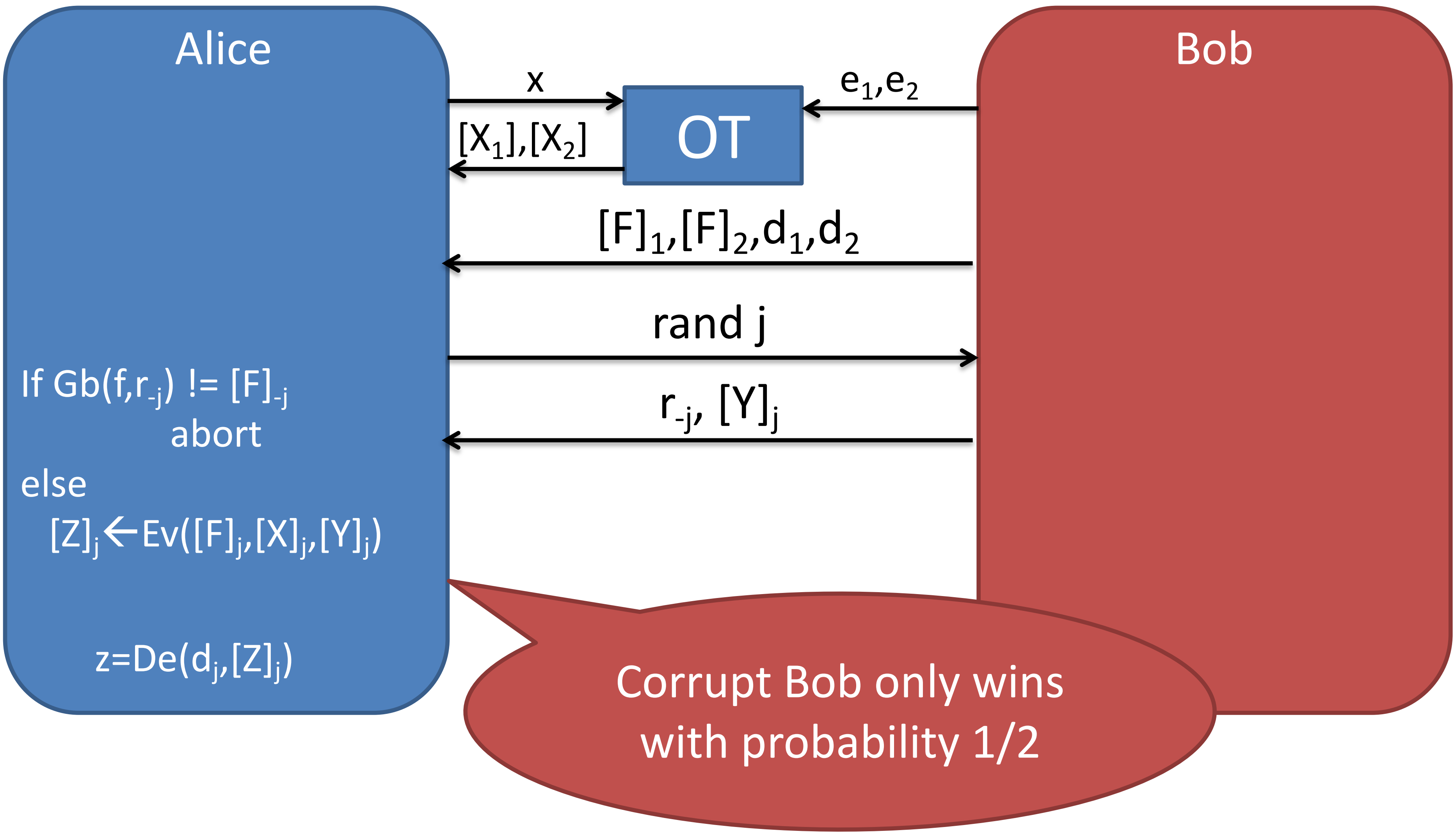


Cut-And-Choose

2PC, simple cut-and-choose



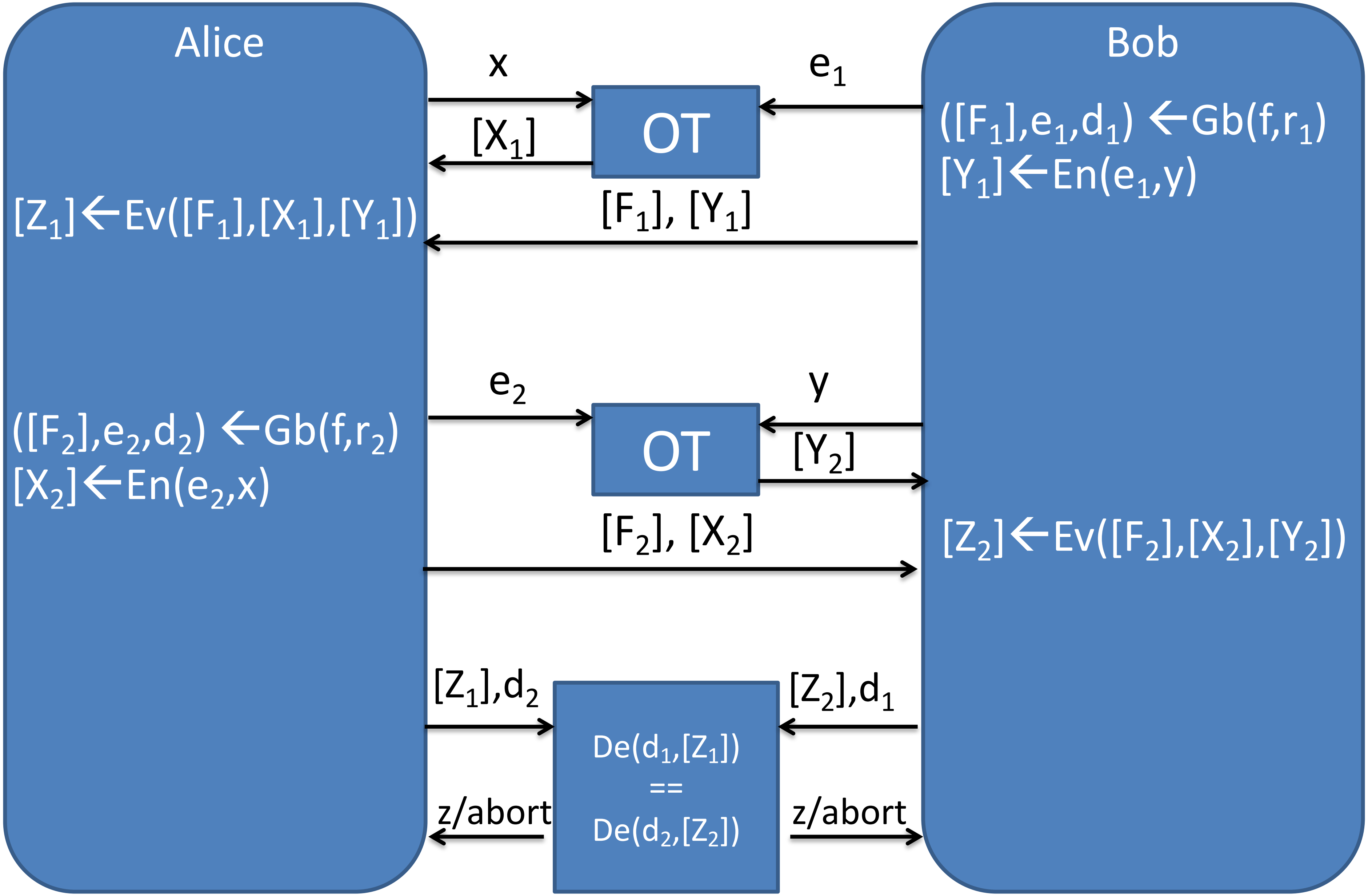
2PC, simple cut-and-choose

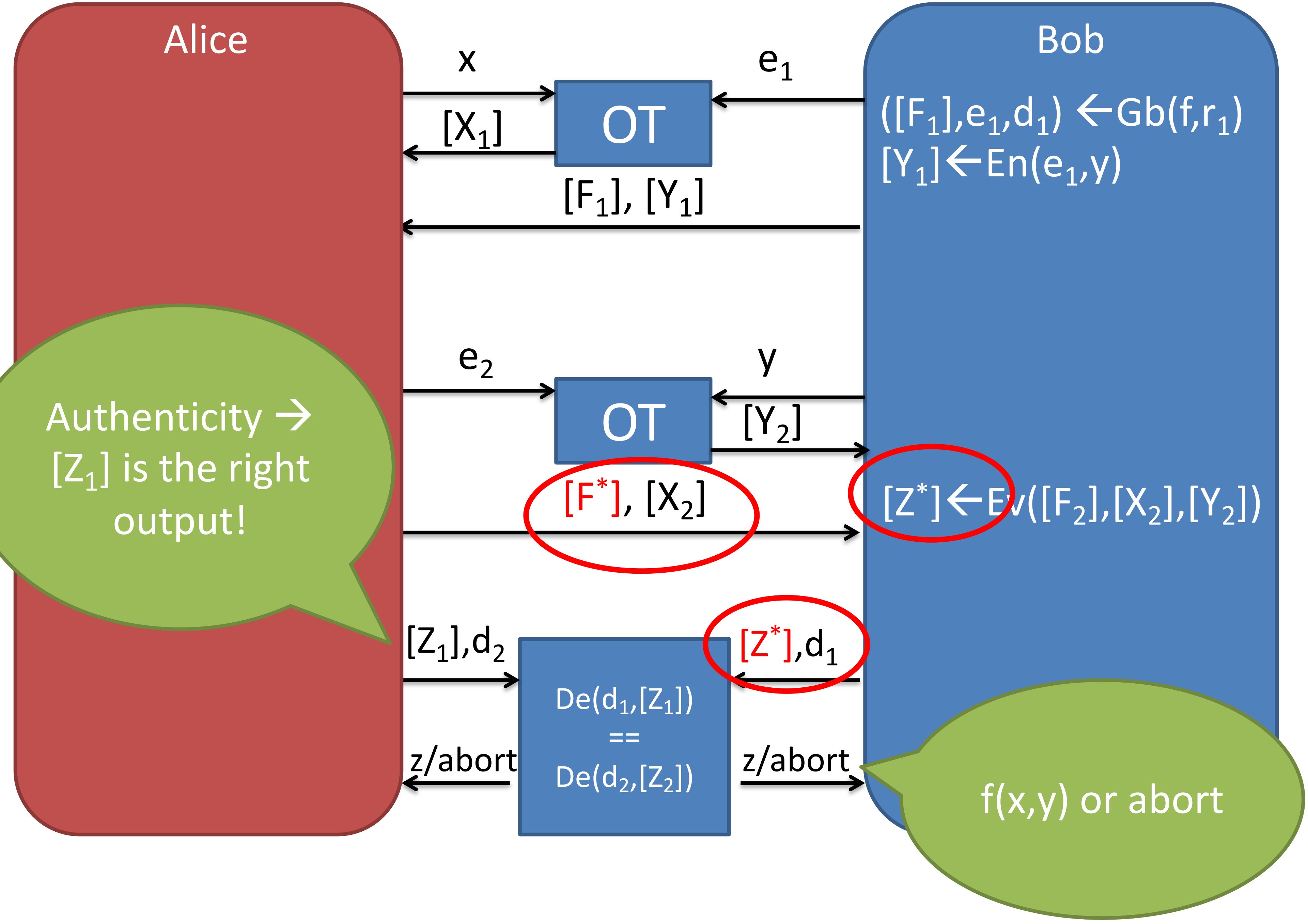


2PC, cut-and-choose

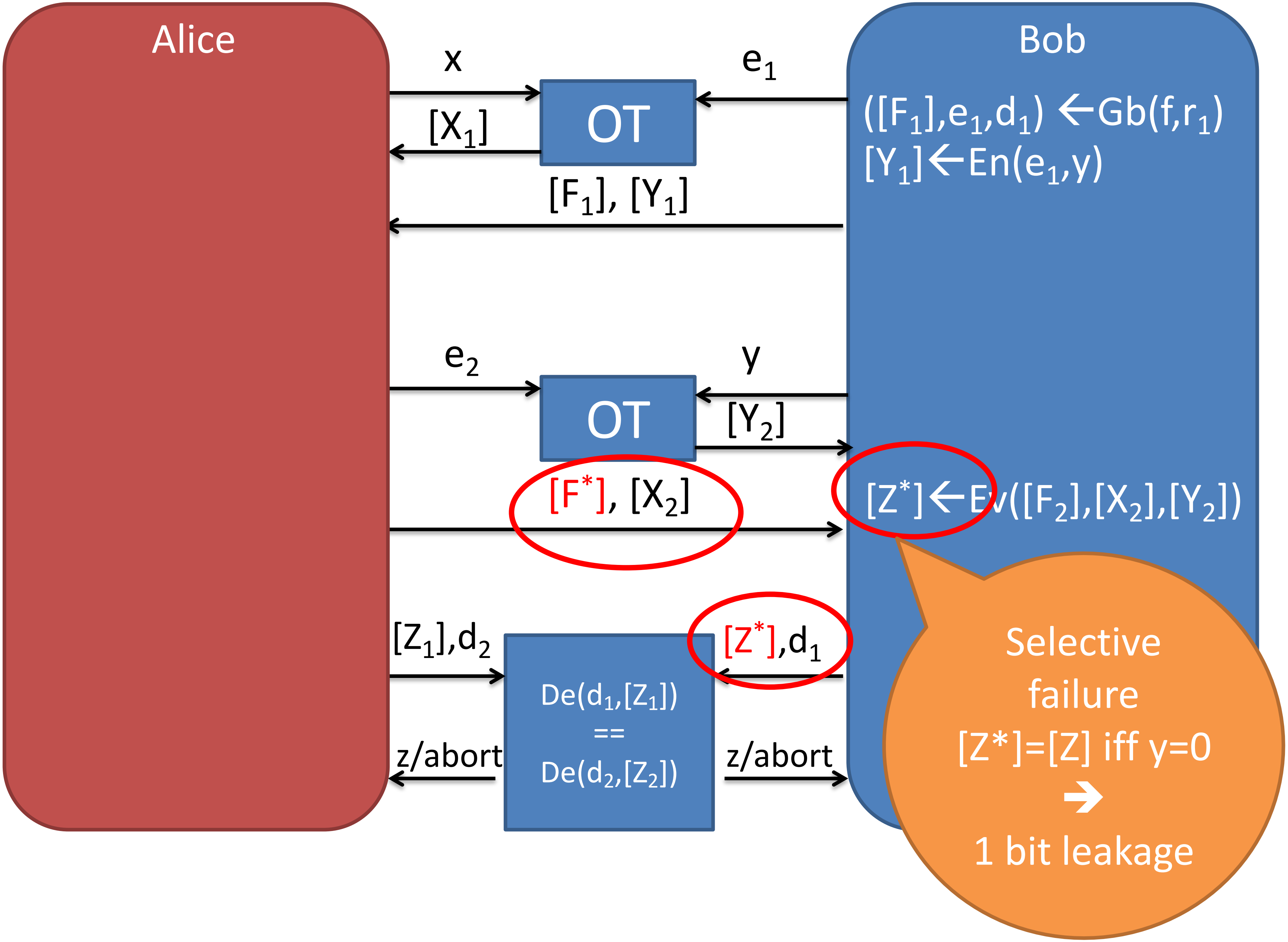
- Simple cut-and-choose
  - Garble k, check k-1, evaluate 1.
  - Security 1-1/k
- Advanced cut-and-choose (see references)
  - Garble 2k, check k, evaluate k
  - Output majority result
  - Security with $2^{-O(k)}$
  - (Need mechanisms to ensure the same input is used!)

Dual Execution


Alice
Bob
x
e1
OT
[X1]
([F1],e1,d1) ←Gb(f,r1)
[Y1]←En(e1,y)
[F1], [Y1]
[Z1]←Ev([F1],[X1],[Y1])
e2
y
OT
[Y2]
([F2],e2,d2) ←Gb(f,r2)
[X2]←En(e2,x)
[F2], [X2]
[Z2]←Ev([F2],[X2],[Y2])
[Z1],d2
[Z2],d1
De(d1,[Z1])
==
De(d2,[Z2])
z/abort
z/abort

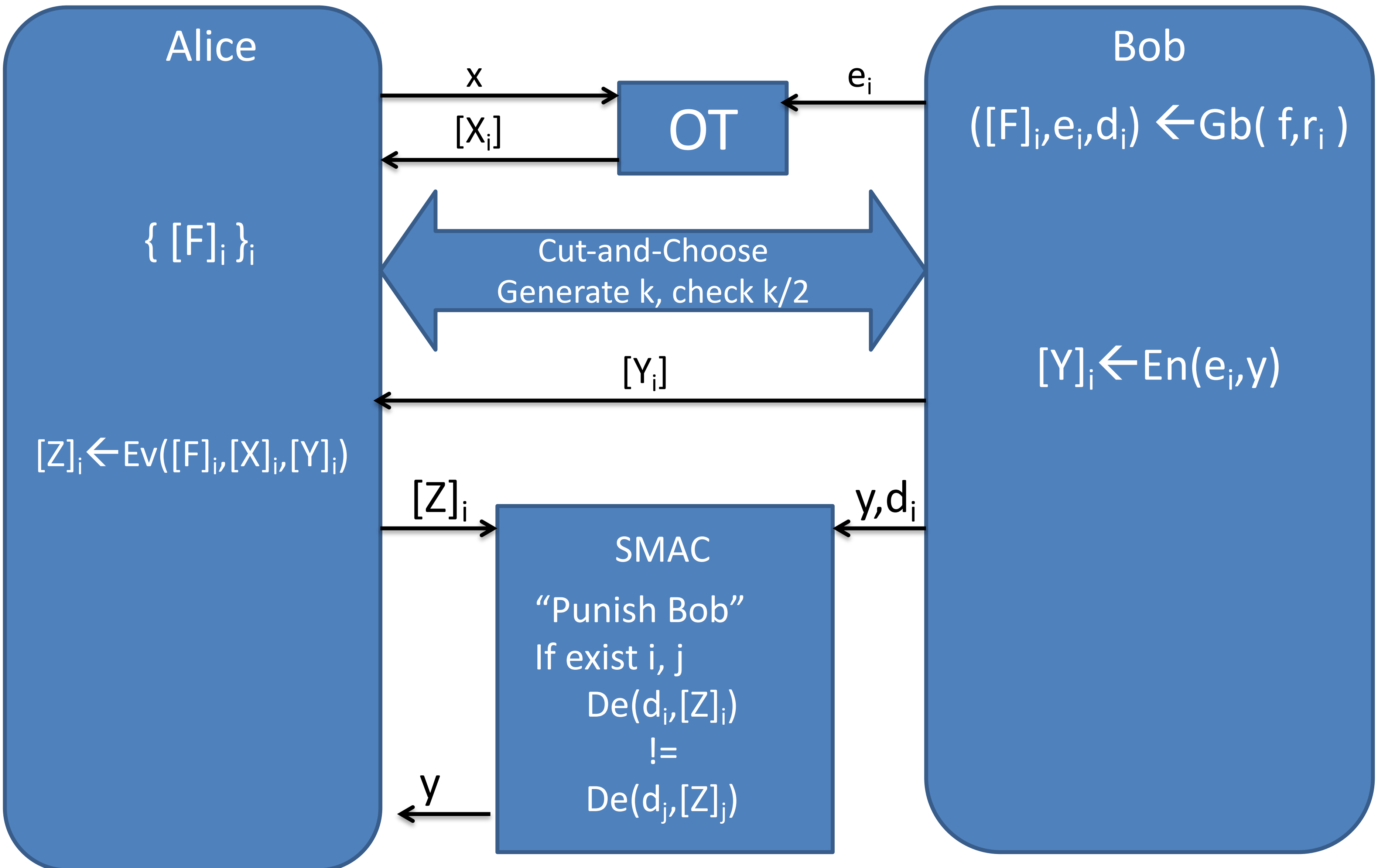


Alice
Bob
x
e_1
OT
$[X_1]$
$([F_1],e_1,d_1) \leftarrow Gb(f,r_1)$
$[Y_1] \leftarrow En(e_1,y)$
$[F_1], [Y_1]$
e_2
y
OT
$[Y_2]$
Authenticity → $[Z_1]$ is the right output!
$[F^*], [X_2]$
$[Z^*] \leftarrow Ev([F_2],[X_2],[Y_2])$
$[Z_1],d_2$
$[Z^*],d_1$
$De(d_1,[Z_1]) == De(d_2,[Z_2])$
z/abort
z/abort
f(x,y) or abort

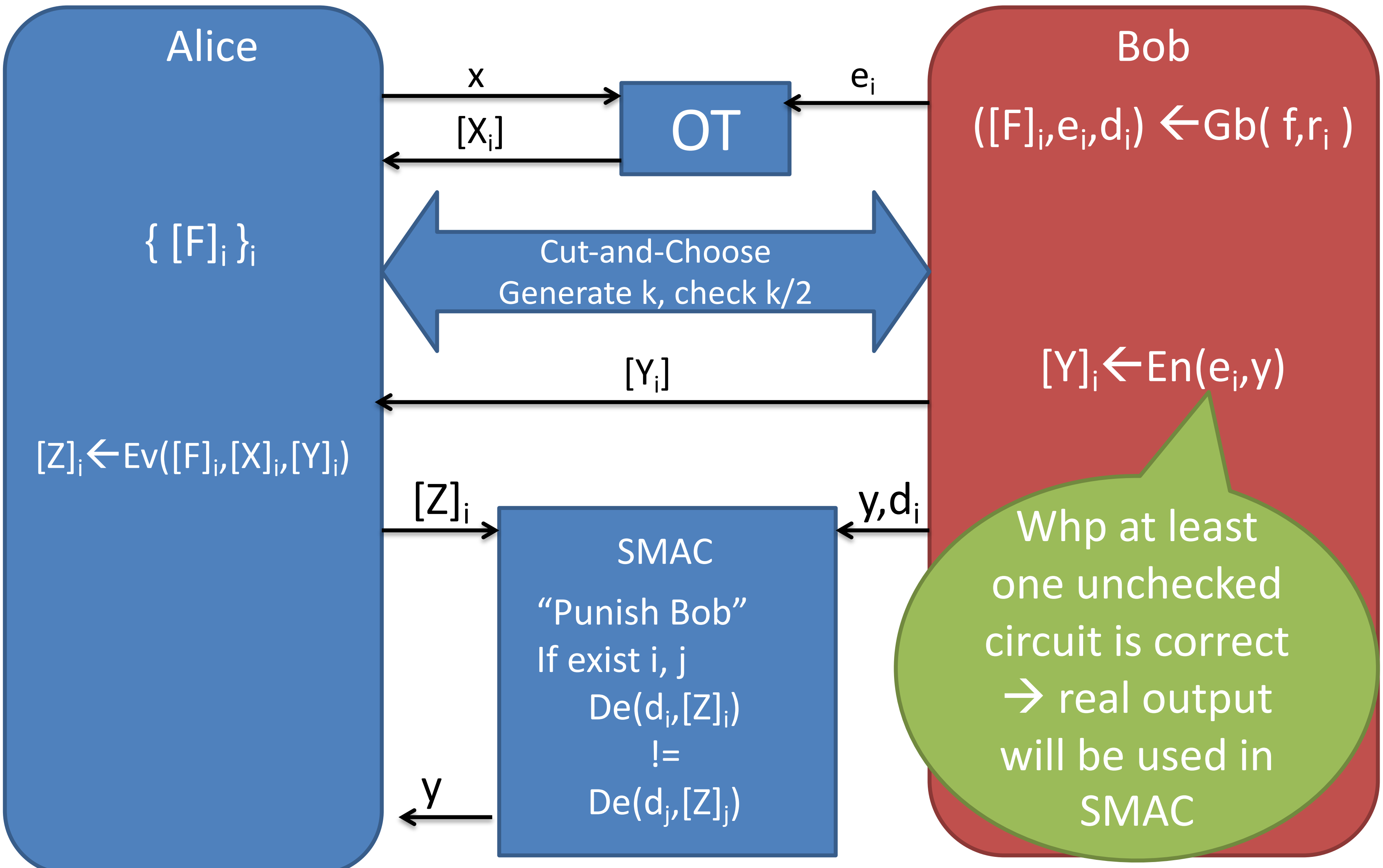

Alice
Bob
x
e_1
OT
$[X_1]$
$([F_1],e_1,d_1) \leftarrow Gb(f,r_1)$
$[Y_1] \leftarrow En(e_1,y)$
$[F_1], [Y_1]$
e_2
y
OT
$[Y_2]$
$[F^*], [X_2]$
$[Z^*] \leftarrow Ev([F_2],[X_2],[Y_2])$
$[Z_1],d_2$
$[Z^*],d_1$
$De(d_1,[Z_1])$
==
$De(d_2,[Z_2])$
z/abort
z/abort
Selective
failure
$[Z^*]=[Z]$ iff y=0
→
1 bit leakage

Forge And Lose

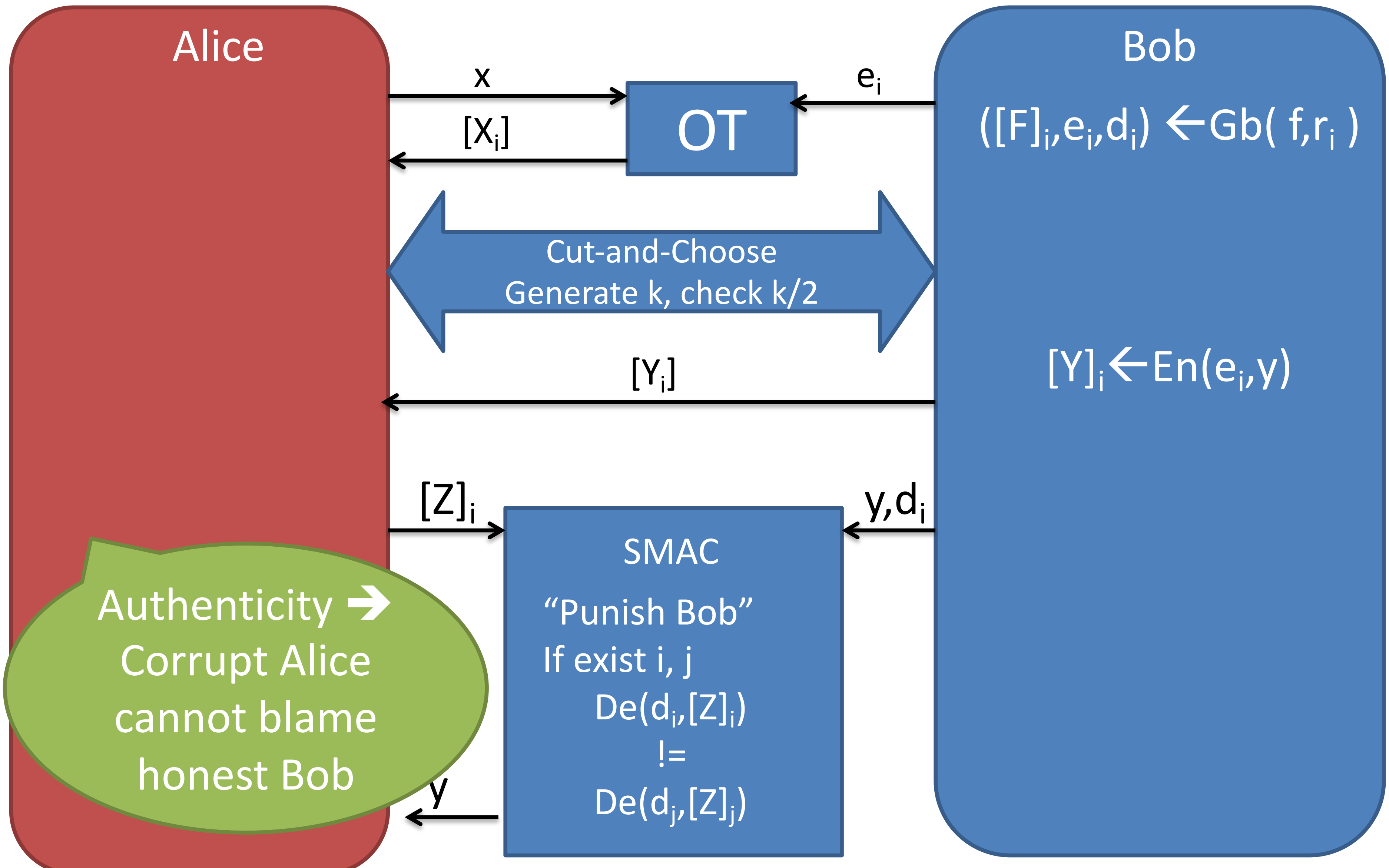
2PC, forge-and-lose (idea)



2PC, forge-and-lose (idea)



2PC, forge-and-lose (idea)



Summary

- Garbling Schemes
  - Definitions and Applications
- Efficient Garbling Techniques
  - Point&Permute, Garbled Row Reduction, Free-XOR, Half-Gate, Privacy-Free, …
- How to use Garbled Circuits with active corruptions

Primary References

- Cryptographic Computing, lecture notes, http://orlandi.dk/crycom (with theory and programming exercises)
- A Brief History of Practical Garbled Circuit Optimizations (Rosulek)
- Fast Cut-and-Choose-Based Protocols for Malicious and Covert Adversaries (Lindell)
- Two Halves Make a Whole - Reducing Data Transfer in Garbled Circuits Using Half Gates (Zahur et al.)
- Privacy-Free Garbled Circuits with Applications to Efficient Zero-Knowledge (Frederiksen et al.)
- Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently (Jawurek et al.)
- Improved Garbled Circuit: Free XOR Gates and Applications (Kolesnikov et al.)
- Foundations of Garbled Circuits (Bellare et al.)

Other References

- Fast Garbling of Circuits Under Standard Assumptions (Gueron et al.)
- Garbling Gadgets for Boolean and Arithmetic Circuits (Ball et al.)
- FleXOR: Flexible Garbling for XOR Gates That Beats Free-XOR (Kolesnikov et al.)
- MiniLEGO: Efficient Secure Two-Party Computation From General Assumptions (Frederiksen et al.)
- Secure Two-Party Computation with Reusable Bit-Commitments, via a Cut-and-Choose with Forge-and-Lose Technique (Brandao)
- Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer (Lindell, Pinkas)
- An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries (Lindell, Pinkas)
- Efficiency Tradeoffs for Malicious Two-Party Computation (Mohassel et al.)