5th Bar-Ilan Winter School on Cryptography
Advances in Practical Multiparty Computation

"Tiny OT" – Part 1

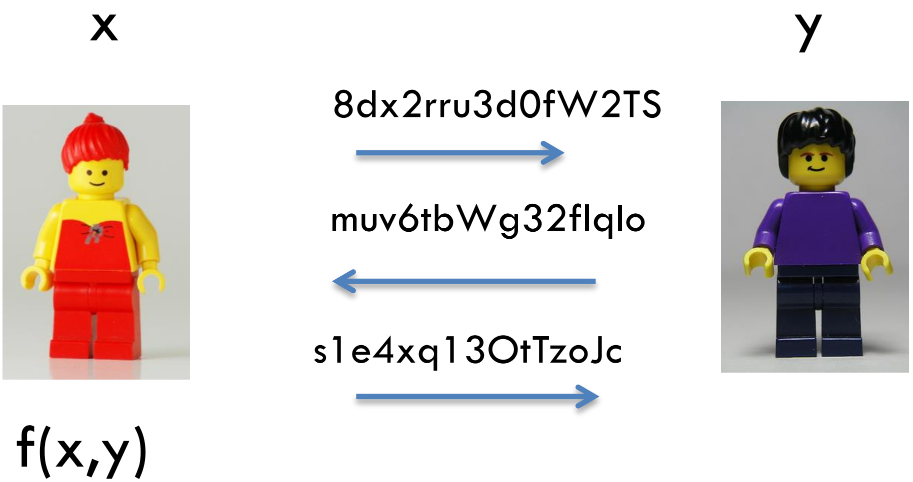
A ~~New~~ (4 years old) Approach to Practical Active-Secure Two-Party Computation

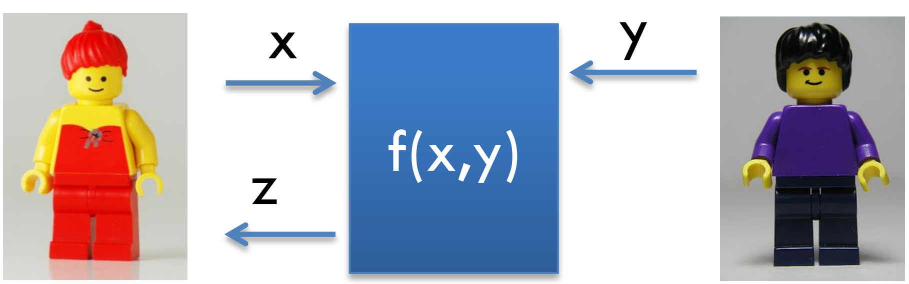
Claudio Orlandi, Aarhus University

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: Active Secure OT Extension**
  - Warmup: OT properties
  - Recap: Passive Secure OT Extension
  - Active Secure OT Extension
- **Part 3: From “Auth. Bits” to “Auth. Triples”**
  - Authenticated local-products (*aAND*)
  - Authenticated cross-products (*aOT*)
  - “LEGO” bucketing

Secure Computation





- *Privacy*
- *Correctness*
- …

What kind of *Secure* Computation?

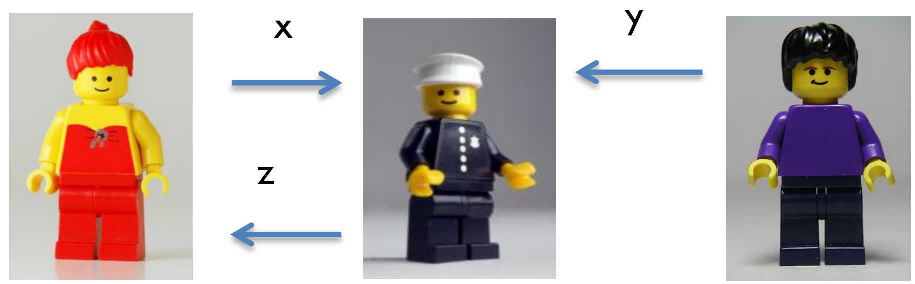
- ***Dishonest majority***
  - The adversary can corrupt up to n-1 participants (n=2).

- ***Static Corruptions***
  - The adversary chooses which party is corrupted before the protocol starts.

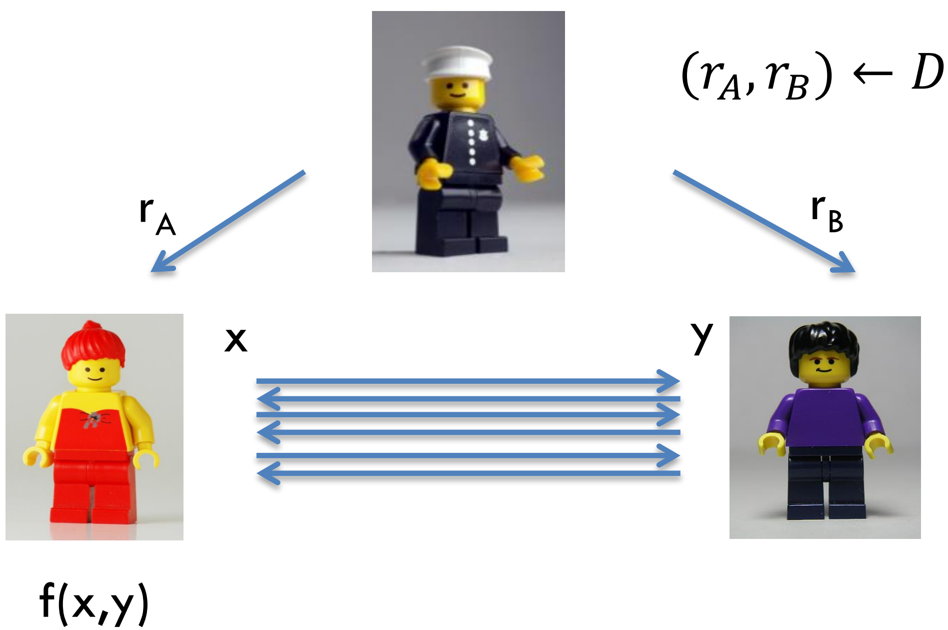
- ***Active Corruptions***
  - Adversary can behave arbitrarily (aka *malicious*)

- ***No guarantees of fairness, termination***
  - Security with abort

Trusted Party



Trusted Dealer

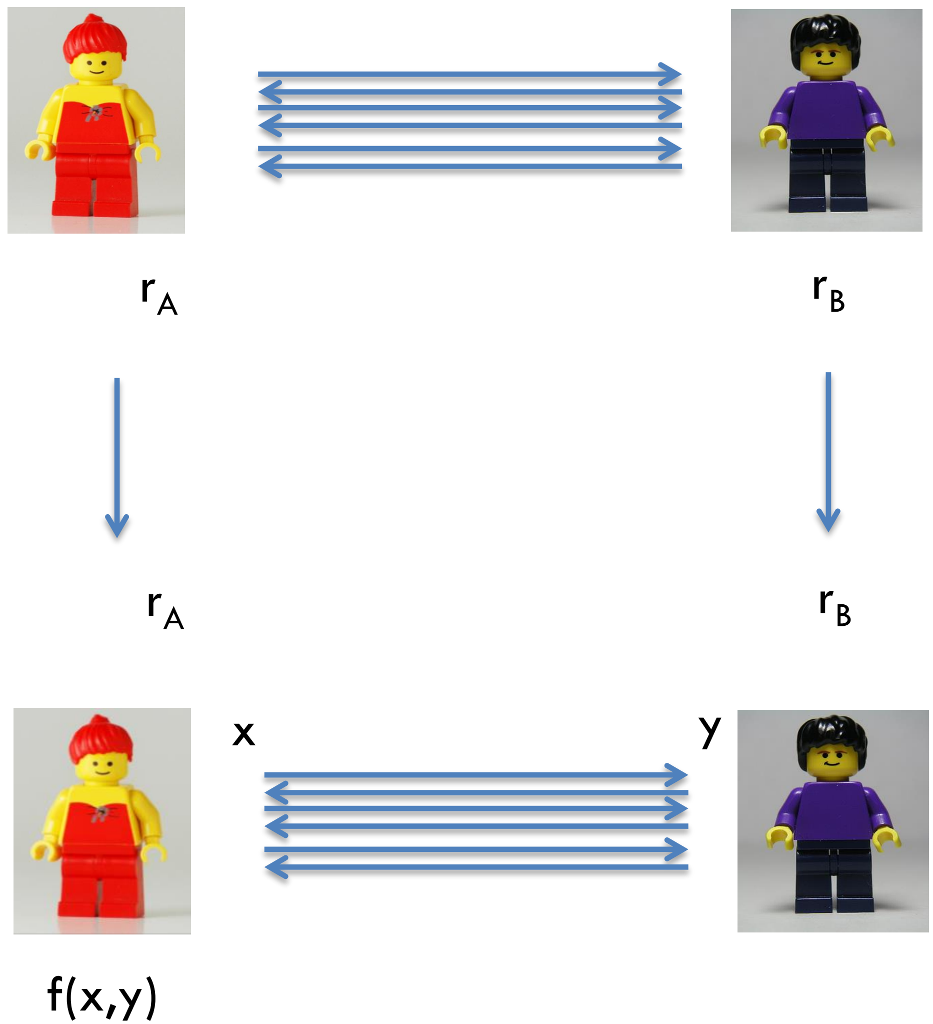


Preprocessing

- Independent of ***x,y***
- Tipically only depends on ***size of f***
- Uses public key crypto technology ***(slower)***

Online Phase

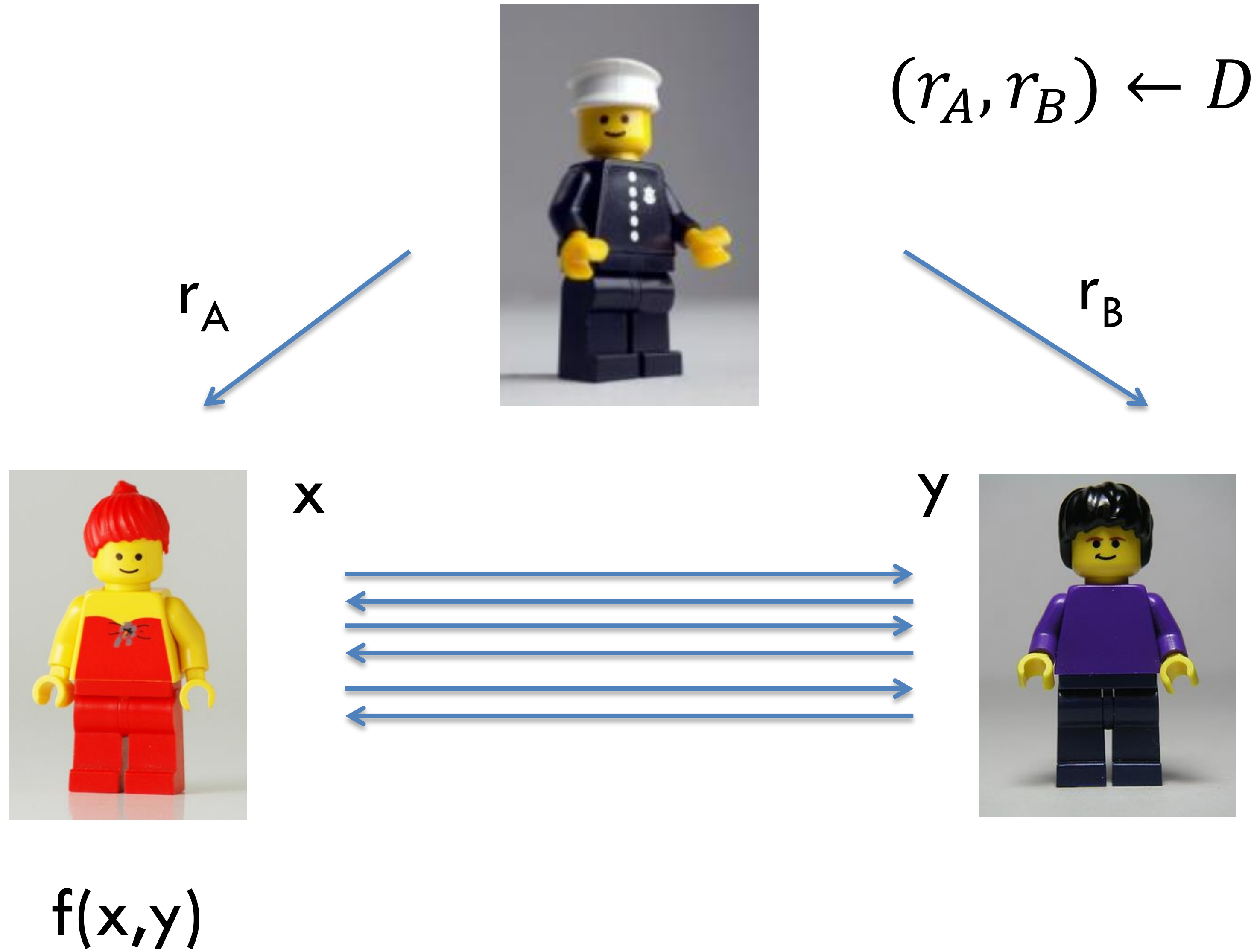
- Uses only information theoretic tools ***(order of magn. faster)***



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

“The simplest 2PC protocol ever”



"The simplest 2PC protocol ever" OTTT (Preprocessing phase)

1) Write the truth table of the function F you want to compute

y

x		0	1	2	3
	0	3	2	2	2
	1	3	0	0	4
	2	1	0	0	4
	3	1	1	4	4

"The simplest 2PC protocol ever" OTTT (Preprocessing phase)

2) Pick random (r, s), rotate rows and columns

s=3

r=1

	0	1	2	3
0	1	4	4	1
1	2	2	2	3
2	0	0	4	3
3	0	0	4	1

“The simplest 2PC protocol ever” OTTT (Preprocessing phase)

3) Secret share the truth table i.e.,

Pick T1 at random, and let

T2 =

1	4	4	1
2	2	2	3
0	0	4	3
0	0	4	1

\- T1

“The simples

(c

“Privacy”:
inputs masked w/uniform random values

T1 , r

T2 , s

$u = x + r$ →

$v = y + s$ ←

T2[u,v] ←

Correctness:
by construction

output $f(x,y) = T1[u,v] + T2[u,v]$

What about active security?

T1 , r

T2 , s

$u = x + r$ →

← $v = y + s + e1$

← $T2[u,v] + e2$

Is this cheating?

- $v = y + s + e1 = (y+e1) + s = y' + s$
  - Input substitution, not cheating according to the definition!
- $M2[u,v] + e2$
  - Changes output to $z' = f(x,y) + e2$
  - Example: $f(x,y)=0$ for all inputs
  - With $e2=1$ Alice outputs 1
    - *Clearly breach of correctness!*

How to force Bob to send the right value?

- **Problem:** Bob can send the wrong shares
- **Solution:** use MACs
  - e.g. *M=aT+b with (a,b)← F*

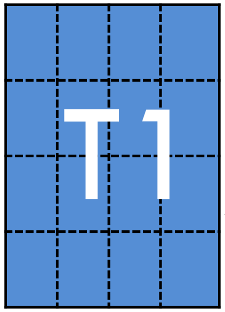
(a,b)

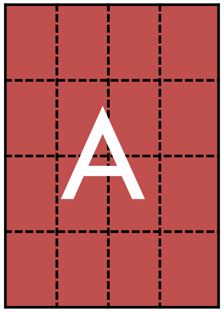
(M',T')

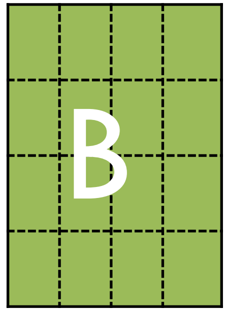
(M,T)

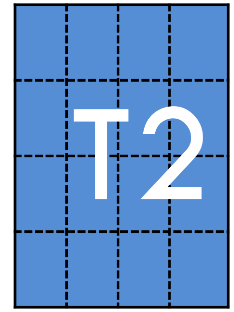
Abort if M'≠aT'+b

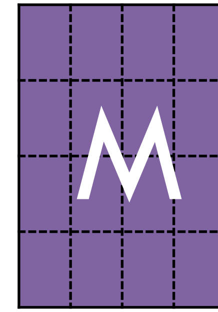
OTTT+MAC



, r






, s


u = x + r →

v = y + s ←

T2[u,v], M[u,v] ←

If (M[u,v]=A[u,v]*T2[u,v]+B[u,v])

output f(x,y) = T1[u,v] + T2[u,v]

else

abort

Statistical security vs. malicious Bob w.p. 1-1/|F|

Curiosity

- Can we get ***perfect*** security?
  - Yes!
  - **On the Power of Correlated Randomness in Secure Computation**
  - Ishai, Kushilevitz, Meldgaard, O, Paskin
  - TCC 2013

“The simplest 2PC protocol ever” OTTT

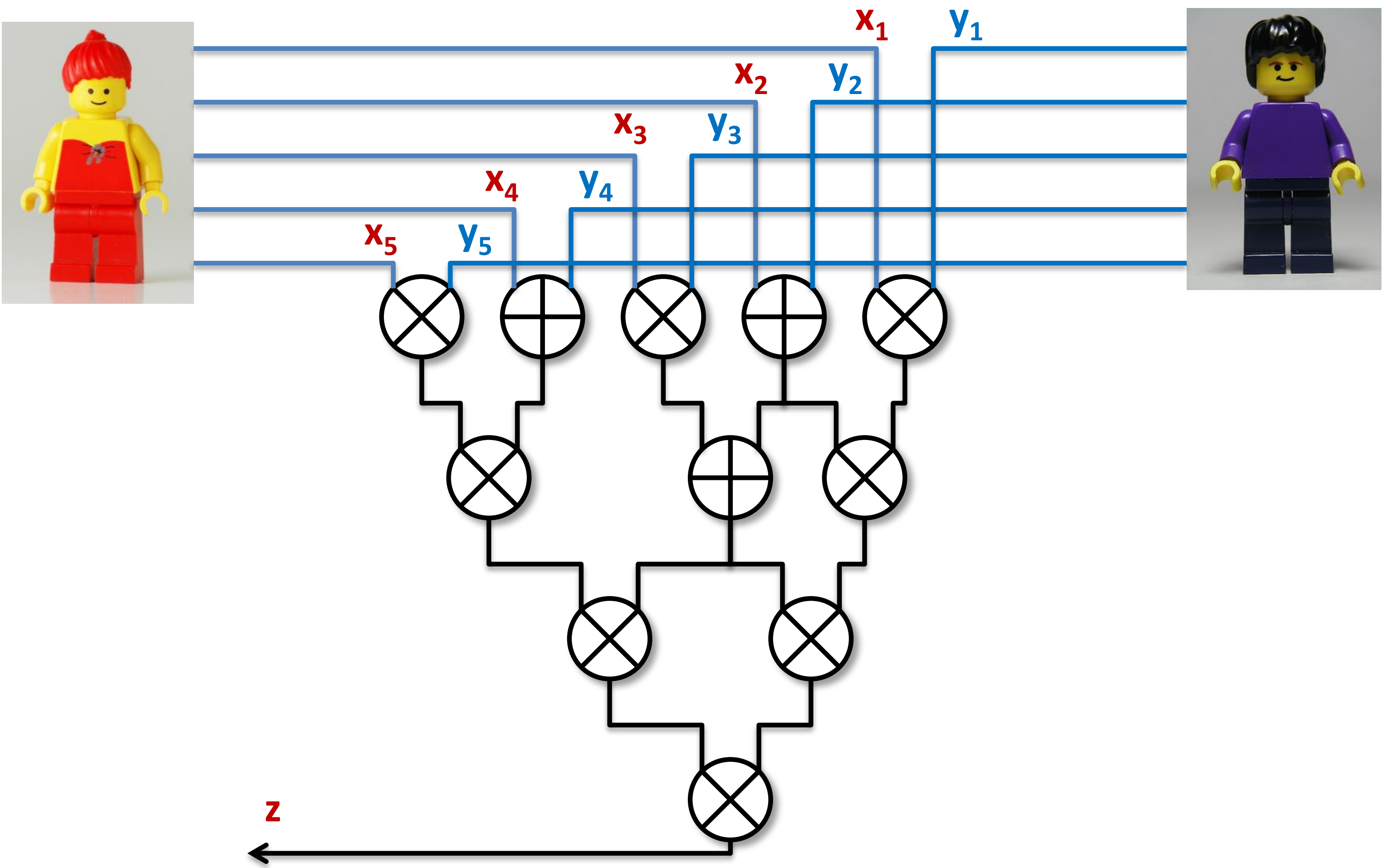
- **Optimal communication complexity ☺**
- **Storage exponential in input size ☹**

➔Represent function using circuit instead of truth table!

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

Circuit based computation



Invariant

- For each ***wire x*** in the circuit we have
  - $[x] := (x_A, x_B)$ *// read "x in a box"*
  - Where Alice holds x_A
  - Bob holds x_B
  - Such that $x_A+x_B=x$

- Notation overload:
  - x is both the r-value and the l-value of x
  - use n(x) for name of x and v(x) for value of x when in doubt.
  - Then $[n(x)] = (x_A,x_B)$ such that $x_A+x_B=v(x)$

Circuit Evaluation (Online phase)

1) [x] ← Input(A,x) :

- chooses random x_B and send it to Bob
- set x_A=x+x_B // symmetric for Bob

Alice only sends a random bit! ”Clearly” secure

2) z ← Open(A,[z]): // z← Open([z]) if both get output

- Bob sends z_B
- Alice outputs z=z_A+z_B // symmetric for Bob

Alice should learn z anyway! ”Clearly” secure

Circuit Evaluation (Online phase)

2) [z]← Add([x],[y]) // at the end z=x+y

- Alice computes $z_A = x_A + y_A$
- Bob computes $z_B = x_B + y_B$

- We write [z] = [x] + [y]

No interaction! ”Clearly” secure
As expensive as a local addition!

Circuit Evaluation (Online phase)

2a) [z]← Mul(a,[x]) // at the end z=a*x

- Alice computes $z_A = a*x_A$
- Bob computes $z_B = a*x_B$

2c) [z]← Add(a,[x]) // at the end z=a+x

- Alice computes $z_A = a+x_A$
- Bob computes $z_B = x_B$

Circuit Evaluation (Online phase)

3) Multiplication?

How to compute [z]=[xy] ?

How do we compute this?

Alice, Bob should compute

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

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

Alice can compute this

Bob can compute this

Circuit Evaluation (Online phase)

3) [z]←Mul([x],[y]):

1. Get *[a],[b],[c]* with *c=ab* from trusted dealer

2. *e=Open([a]+[x])*
3. *d=Open([b]+[y])*

Is this secure?
e,d are "one-time-pad" encryptions of x and y using a and b

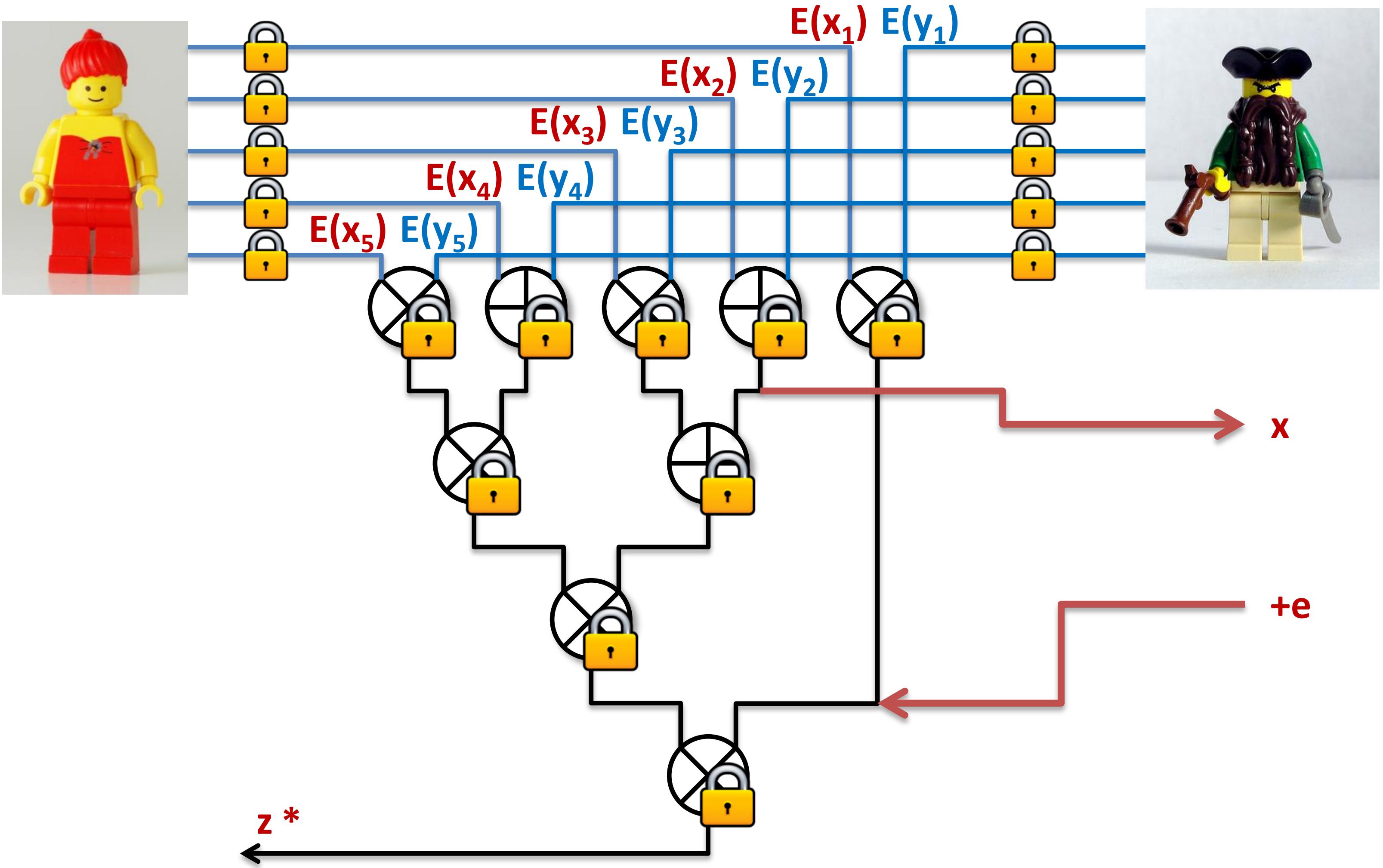
4. Compute [z] = [c] + e[y] + d[x] - ed

$$ab + (ay+xy) + (bx+xy) - (ab+ay+bx+xy)$$

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**

Secure Computation



Active Security?

- **“Privacy”**
  – even a malicious Bob does not learn anything.

- **“Correctness”**
  – a corrupted Bob can change his share during any “Open” (both final result or during multiplication) leading the final output to be incorrect.

Problem

2) z ← Open(A,[z]):

- Bob sends $z_B + e$
- Alice outputs $z = z_A + z_B + e$ // symmetric for Bob

Problem

2) $z \leftarrow$ Open(A,[z]):

- Bob sends $z_{B,}$ m_B
- Alice outputs
  - $z = z_A + z_B$ if $m_B = k_A + z_B \Delta_A$
  - "abort" otherwise

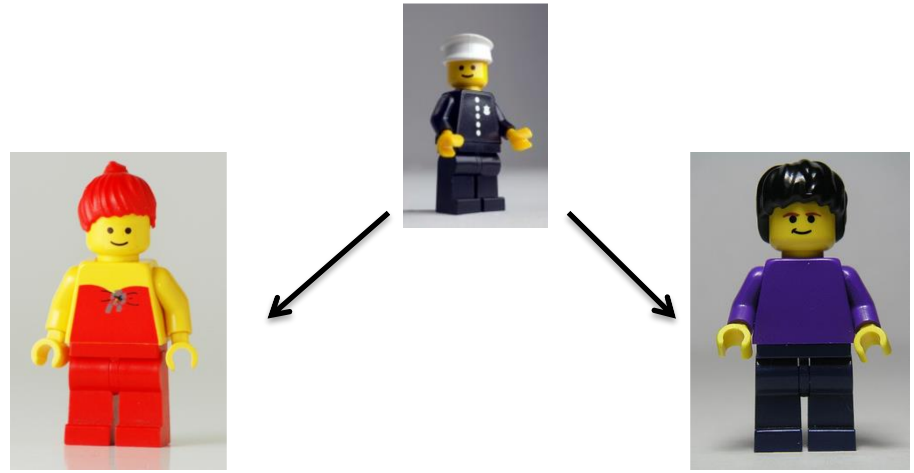
- **Solution:** Enhance representation [x]
  - $[x] = ((x_A, k_A, m_A) , (x_B, k_B, m_B))$ s.t.
  - $m_B = k_A + x_B \Delta_A$ (symmetric for m_A)
  - $\Delta_{A,} \Delta_B$ is the same for all wires.

Linear representation

- Given
  - $[x] = ((x_A, k_{Ax}, m_{Ax}) , (y_B, k_{Bx}, m_{Bx}))$
  - $[y] = ((y_A, k_{Ay}, m_{Ay}) , (y_B, k_{By}, m_{By}))$
  - Compute $[z] = ($
    $(z_A = x_A + y_A, \quad k_{Az} = k_{Ax} + k_{Ay}, \quad m_{Az} = m_{Ax} + m_{Ay})$,
    $(z_B = x_B + y_B, \quad k_{Bz} = k_{Bx} + k_{By}, \quad m_{Bz} = m_{Bx} + m_{By})$,)
- And [z] is in the right format since…

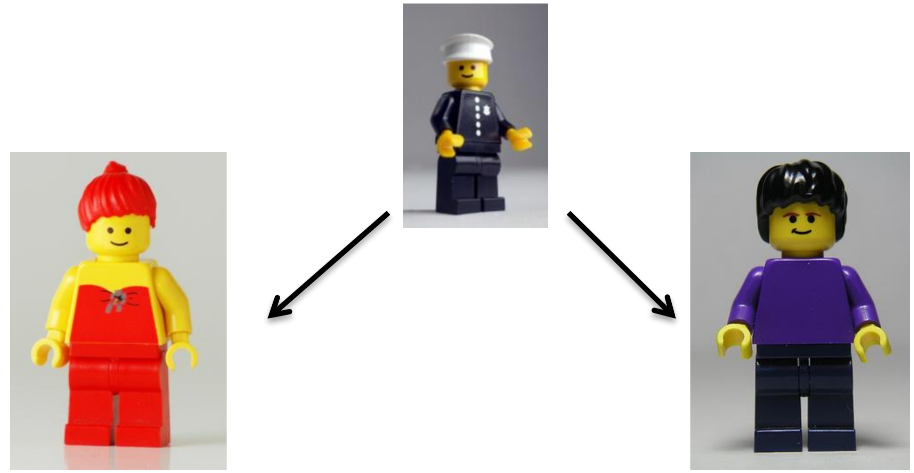
$$m_{Bz} = (m_{Bz} + m_{By}) = (k_{Ax} + x_B\Delta_A) + (k_{Ay} + y_B\Delta_A)$$
$$= (k_{Ax} + k_{Ay}) + (x_B + y_B)\Delta_A = k_{Az} + z_B\Delta_A$$

Recap



1. **Output Gates:**
   - Exchange shares and MACs
   - Abort if MAC does not verify
2. **Input Gates**:
   - Get a random [r] from ***trusted dealer***
   - r ← Open(A,[r])
   - Alice sends Bob *d=x-r,*
   - Compute [x]=[r]+d

Recap



1. **Addition Gates:**
   - Use linearity of representation to compute

$$[z]=[x]+[y]$$

2. **Multiplication gates:**
   - Get a random triple [a][b][c] with c=ab from TD.
   - e ←Open([a]+[x]), d ← Open([b]+[y])
   - Compute [z] = [c] + a[y] + b[x] - ed

Final remarks

- Size of MACs

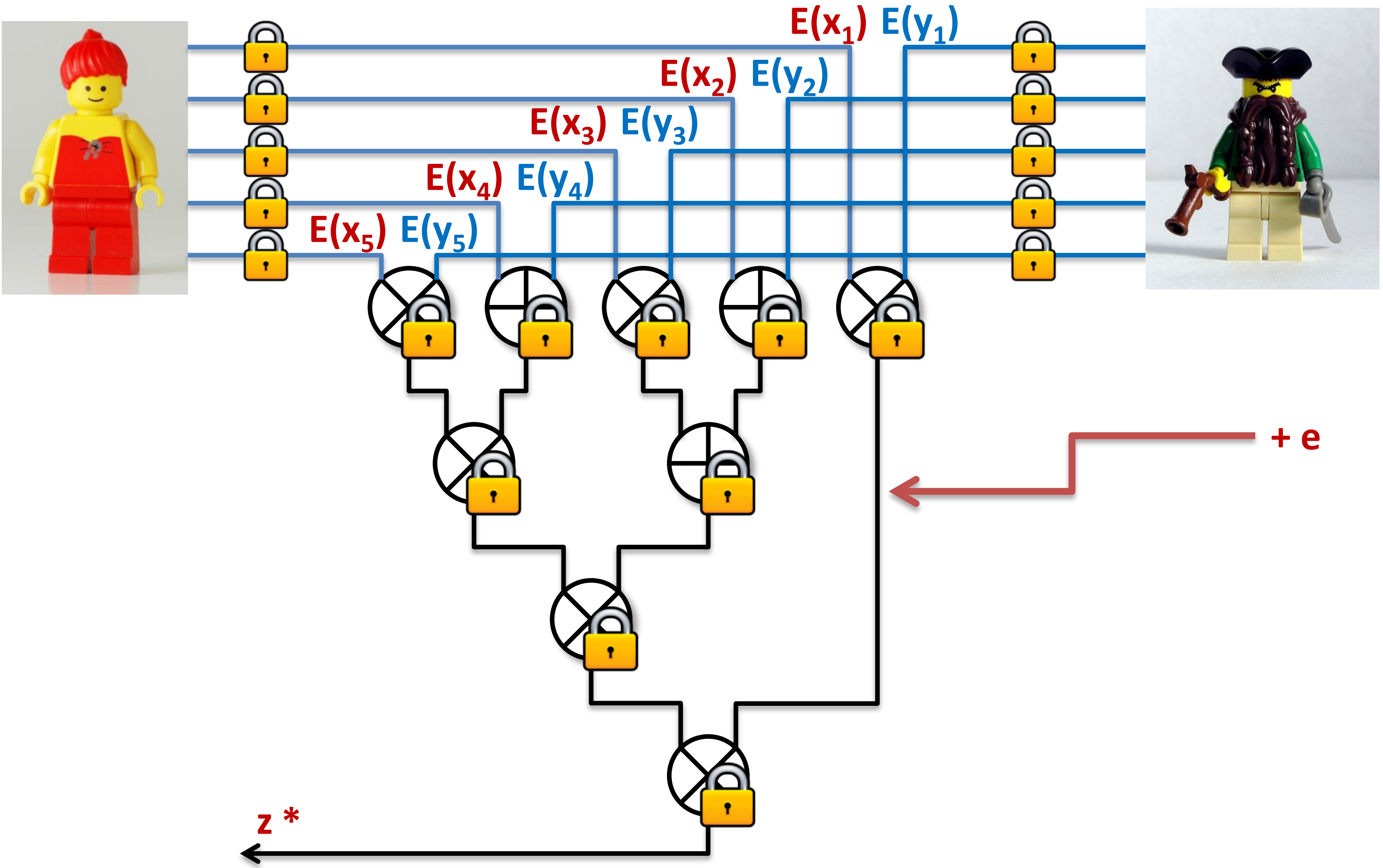
- Lazy MAC checks

Size of MACs

1. Each party must store a mac/key pair ***for each other party***
   – quadratic complexity! ☹
   – SPDZ (tomorrow) for linear complexity.

2. MAC is only as hard as guessing key!
   k MACs in parallel give security $\boldsymbol{1/|F|^k}$
   – In *TinyOT* $\mathbf{F=Z_2}$, then MACs/Keys are *k-bit strings*
   – *MiniMACs* for constant overhead

Lazy MAC Check



Lazy MAC Check

1) $z \leftarrow$ PartialOpen(A,[z]):

1. Bob sends z_B
2. Bob runs OutMAC.append(m_B)
3. Alice runs InMAC.append($k_A + z_B\ \Delta_A$)
4. Alice outputs $z=z_A+z_B$

2) $z \leftarrow$ FinalOpen(A,[z]):

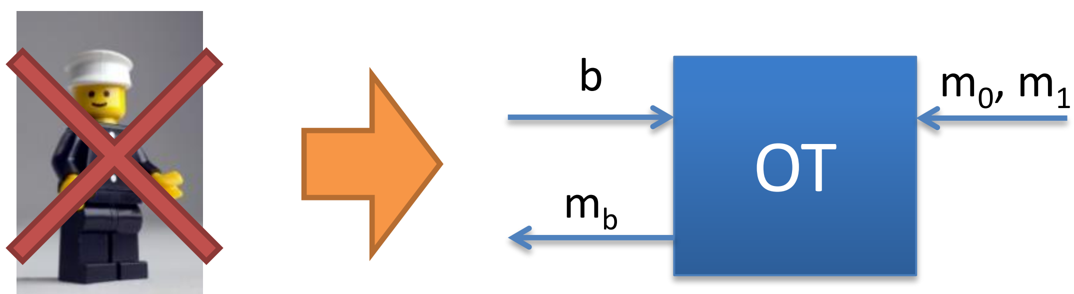
1. Steps 1-3 as before
2. Bob sends u=H(OutMAC) to Alice
3. Alice outputs $z=z_A+z_B$ if u=H(InMAC)
4. "abort" otherwise

Recap of Part 1

- Two protocols *“**in the trusted dealer model**”*
  - **One Time-Truth Table**
    - **Storage** exp(input size) ☹
    - **Communication** O(input size) ☺
    - 1 **round** ☺
  - **(BeDOZa)/TinyOT online phase**
    - **Storage** linear #number of AND gates
    - **Communication** linear #number of AND gates
    - #**rounds** = depth of the circuit
  - …and add enough **MACs** to get **active security**

Recap of Part 1

- To do secure computation is enough to precompute enough **random multiplications**!



- If no ***semi-trusted party is available***, we can use **cryptographic assumption** (next)