Batched Non-interactive 2PC

Payman Mohassel
Visa Research

Mike Rosulek
OSU

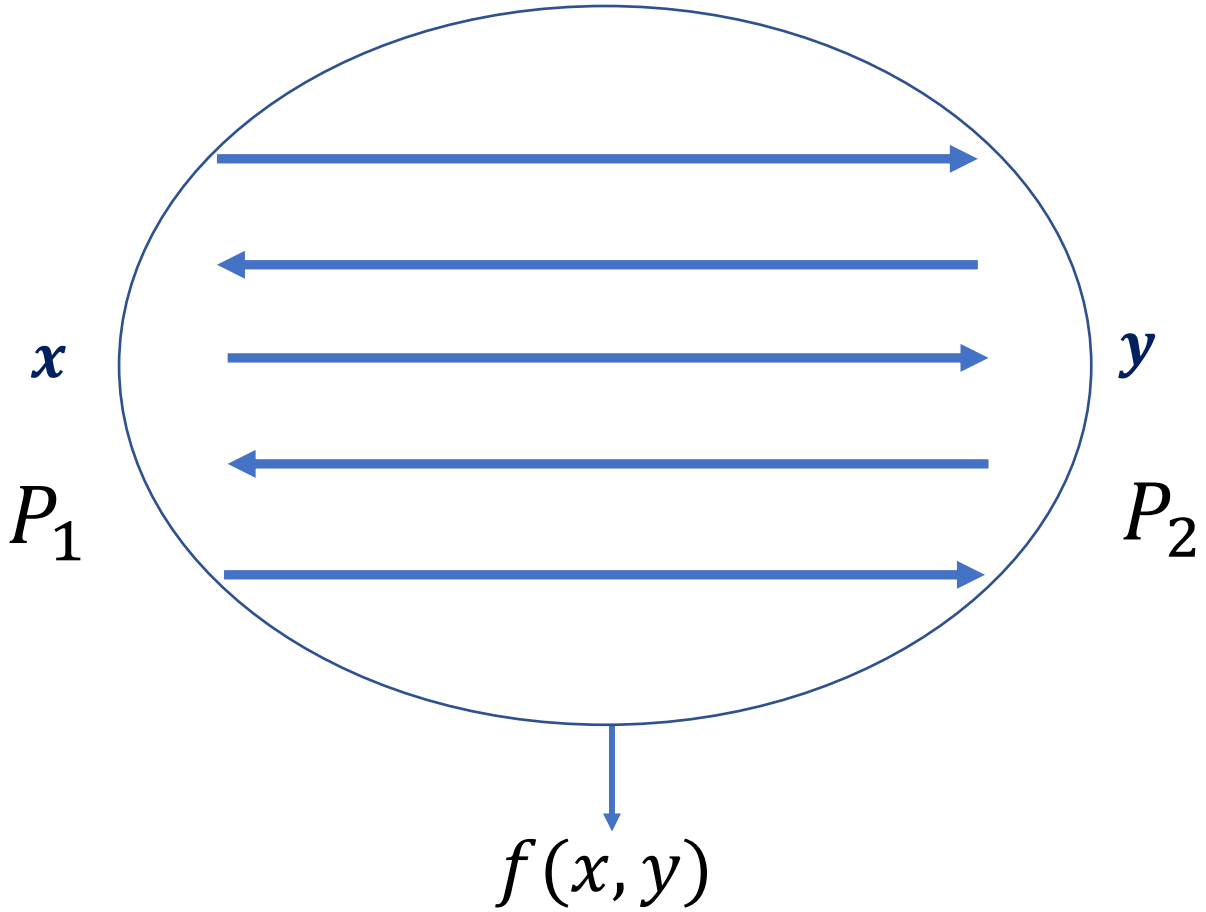
Secure Two-Party Computation

$\boldsymbol{x}$ $\boldsymbol{y}$

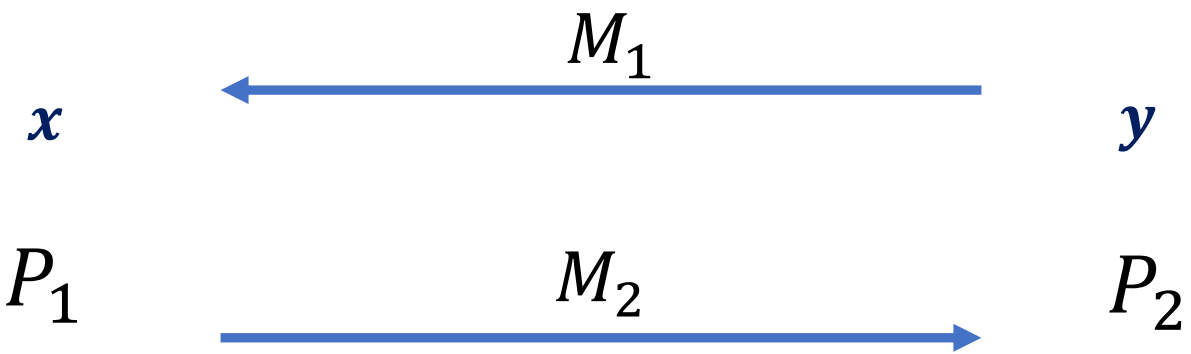
P_1 P_2

$f(x, y)$

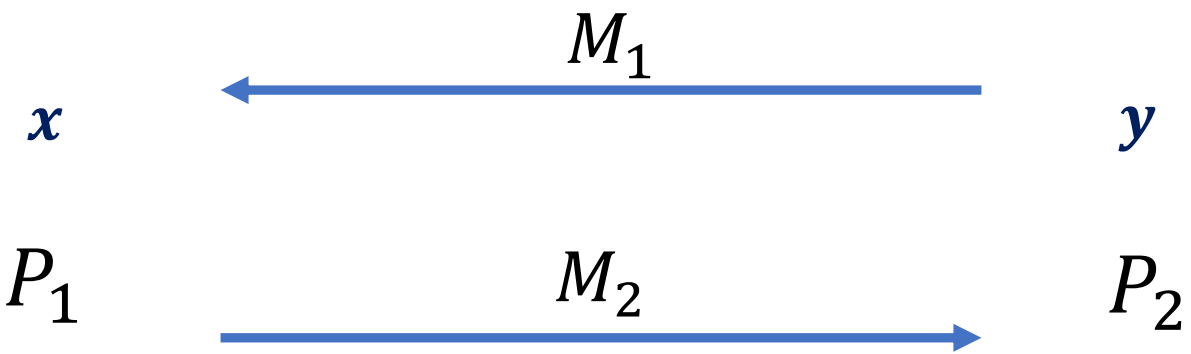
Secure Two-Party Computation



Non-Interactive Secure Computation (NISC)

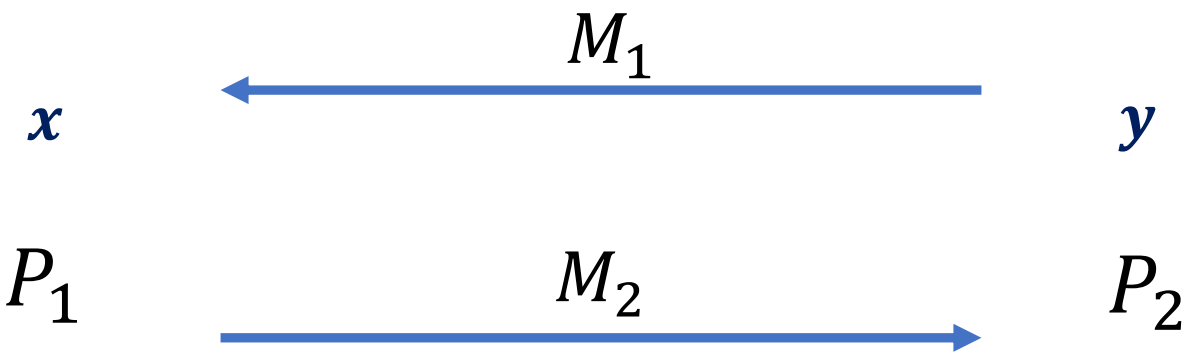


Non-Interactive Secure Computation (NISC)



- Over the internet
- Without coordination
  - Email
  - Bulletin boards

Non-Interactive Secure Computation (NISC)

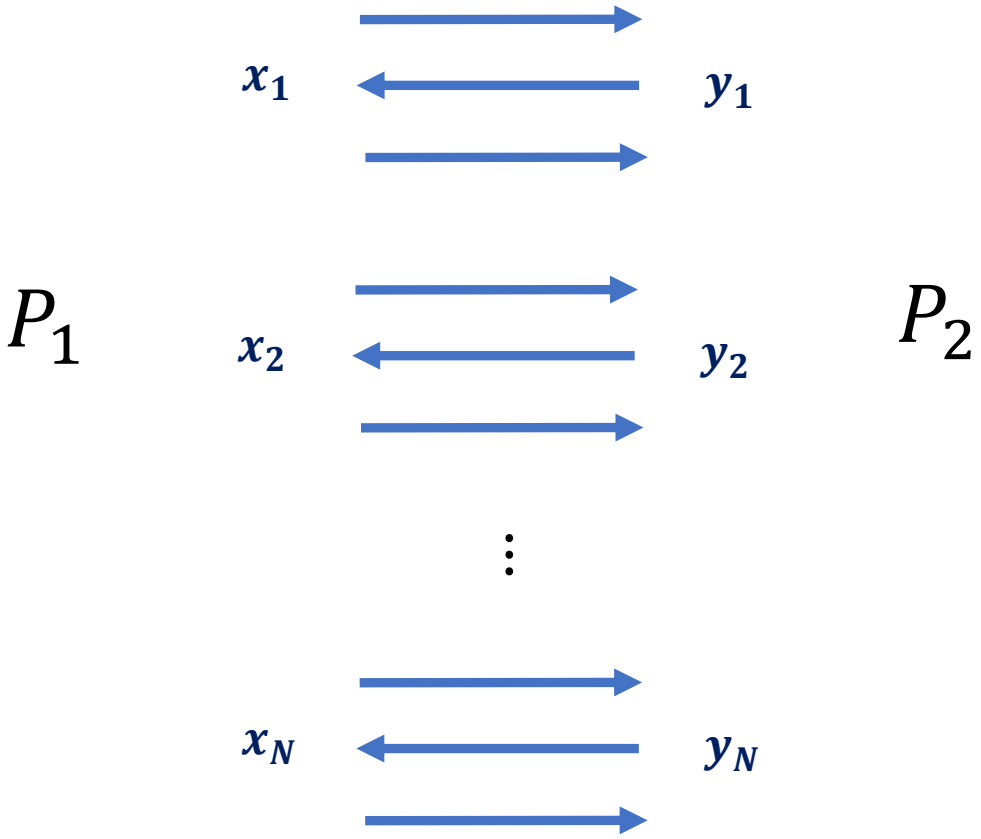


- Over the internet
- Without coordination
  - Email
  - Bulletin boards

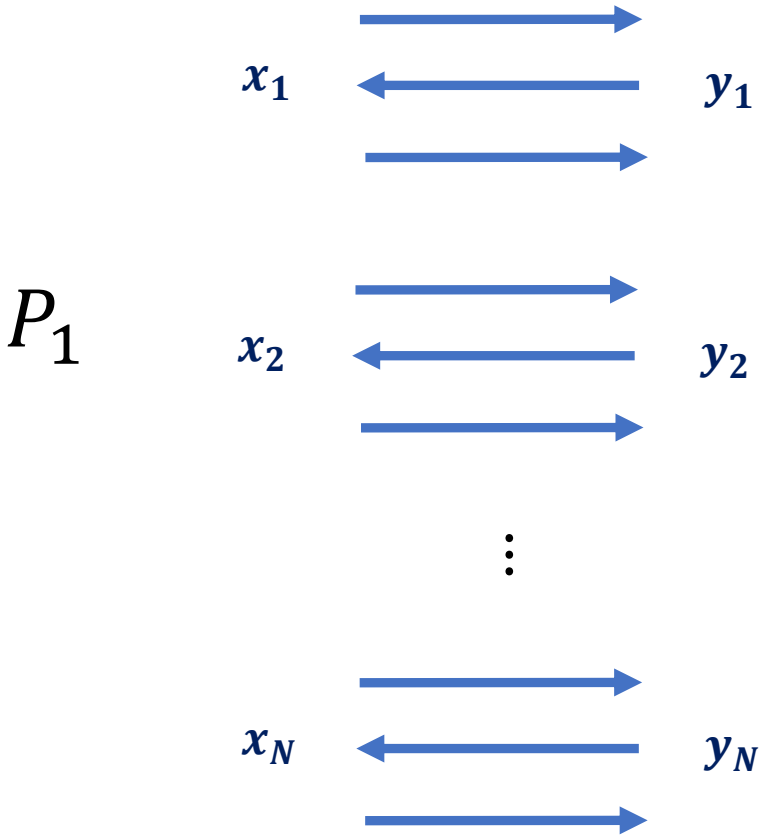
Comparable to best 2PC

[AMPR14]

Batched 2PC

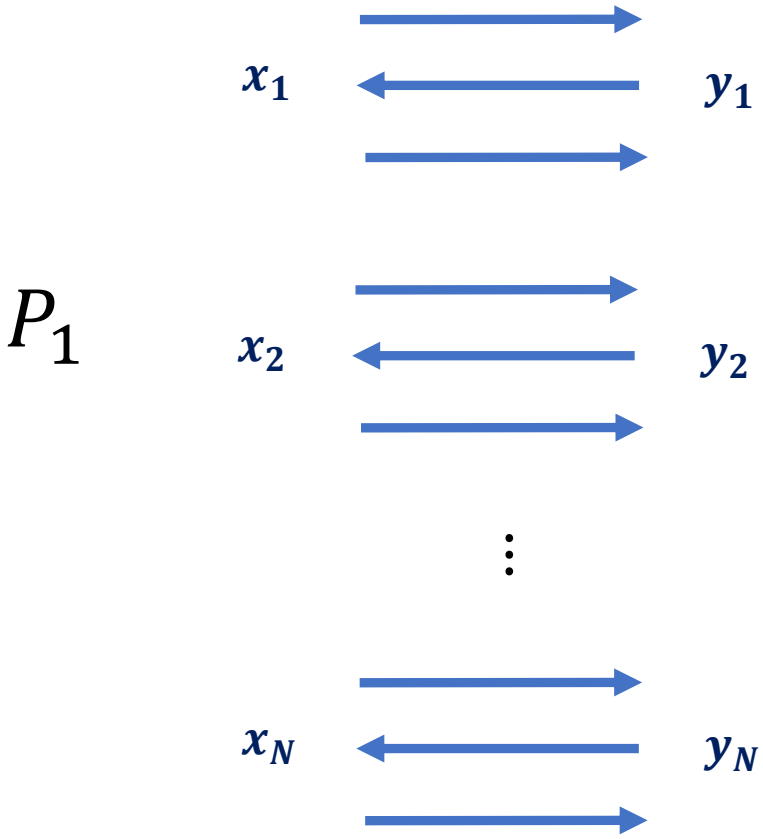


Batched 2PC



- Better amortized efficiency
  - $logN$ improvement
- [NO09,FJNNO13, LR14,HKKKM14, …]

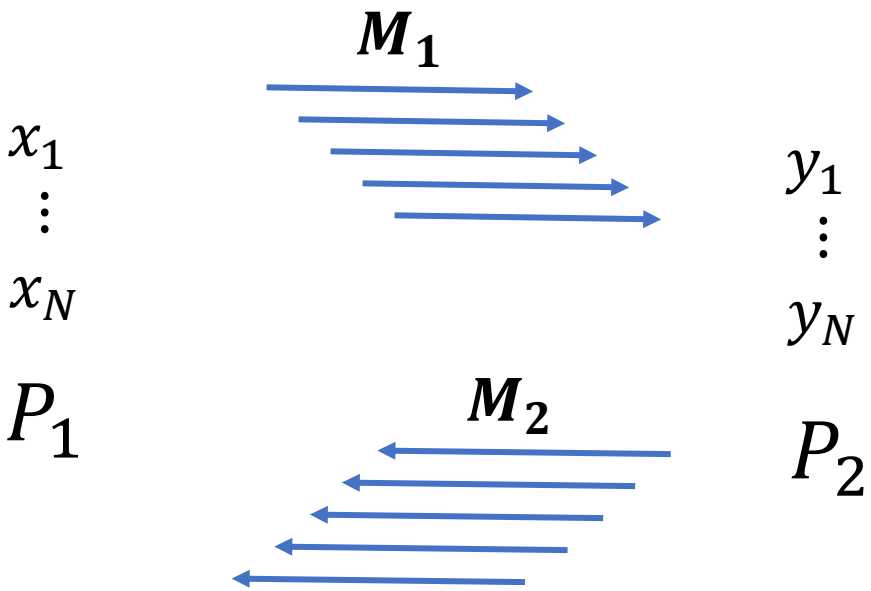
Batched 2PC



- Better amortized efficiency
  - $logN$ improvement
- [NO09,FJNNO13, LR14,HKKKM14, …]

4 rounds

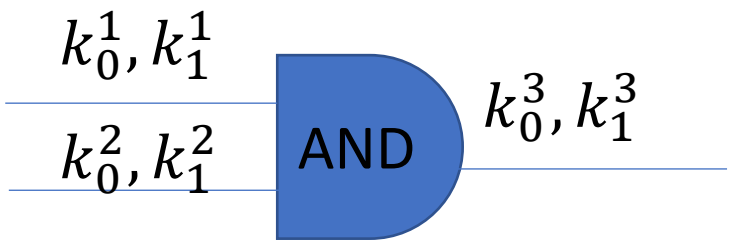
Best of Both Worlds



- $Two\ rounds$
- $logN\ improvement$

Yao's Garbled Circuits

$$C(x, y) = f(x, y)$$



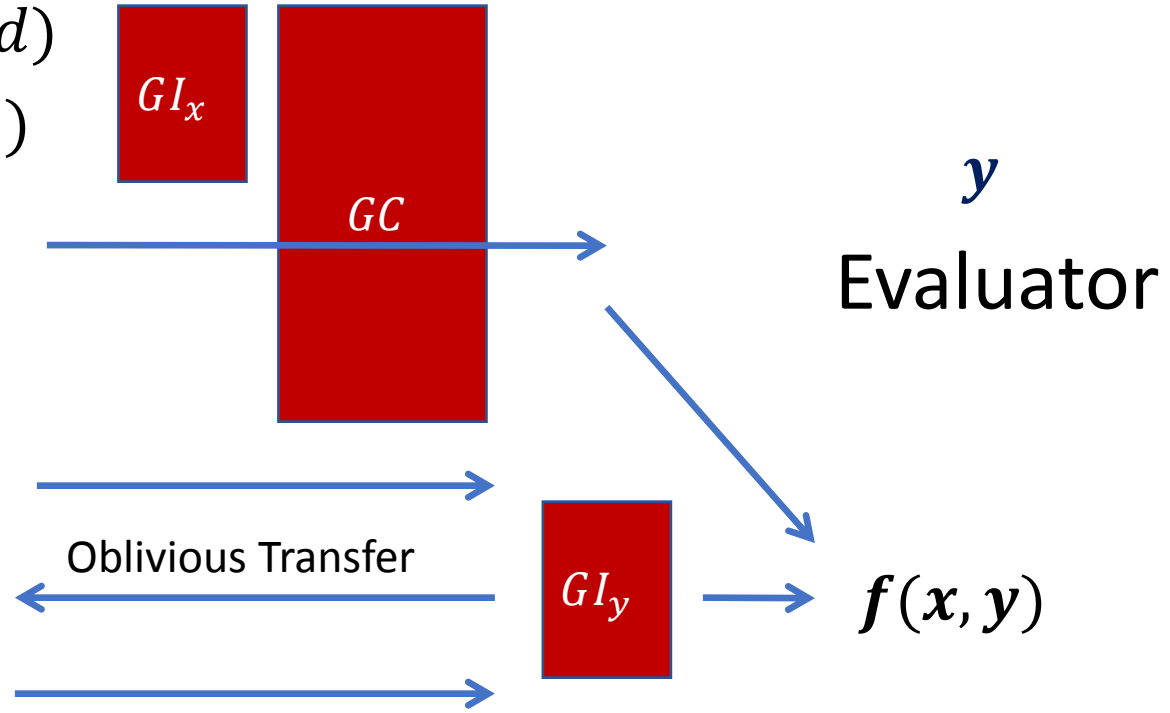
$$c_{0,0} = E_{\{k_0^1, k_0^2\}}(k_0^3)$$
$$c_{0,1} = E_{\{k_0^1, k_1^2\}}(k_0^3)$$
$$c_{1,0} = E_{\{k_1^1, k_0^2\}}(k_0^3)$$
$$c_{1,1} = E_{\{k_1^1, k_1^2\}}(k_1^3)$$

$GC \leftarrow Garb(C, sd)$

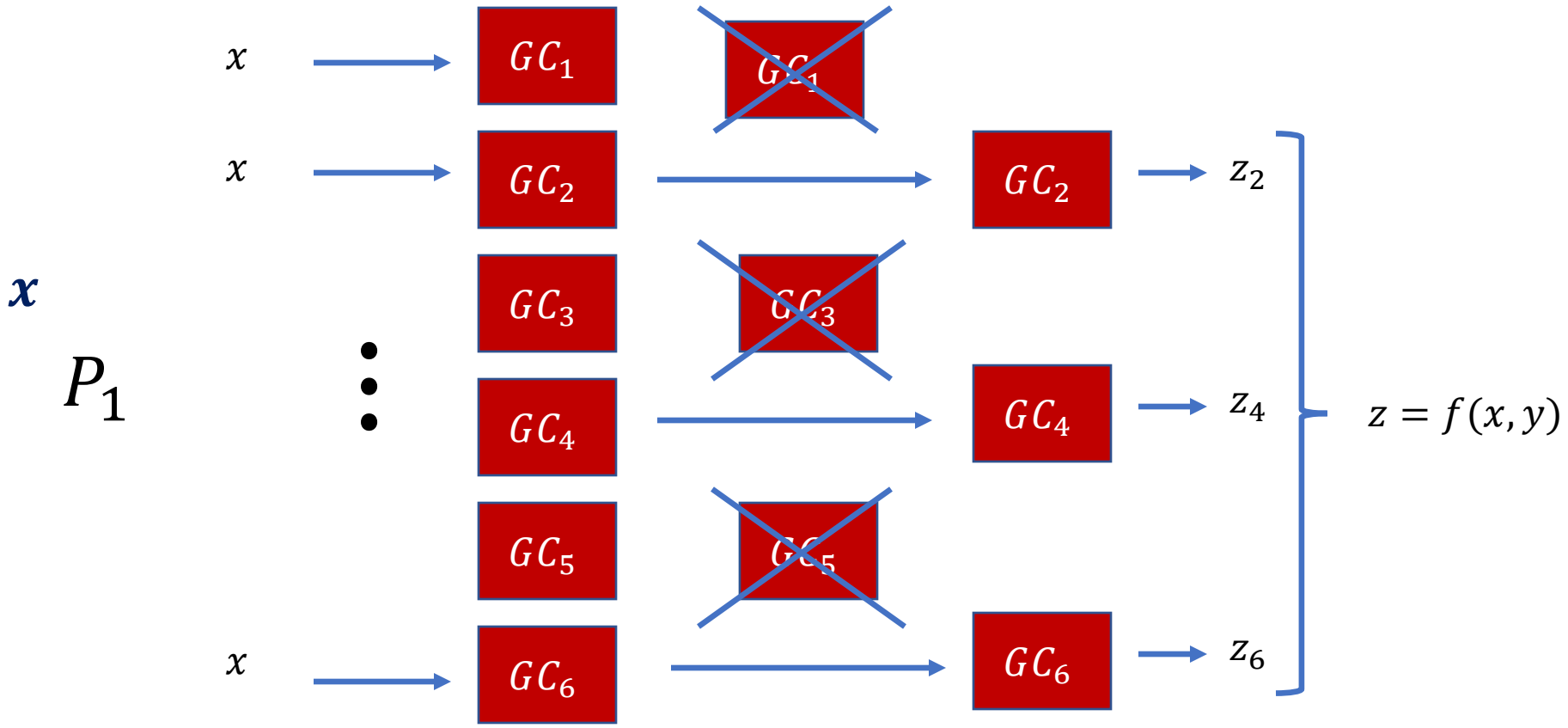
$GI_x \leftarrow GIn(x, sd)$

$\boldsymbol{x}$

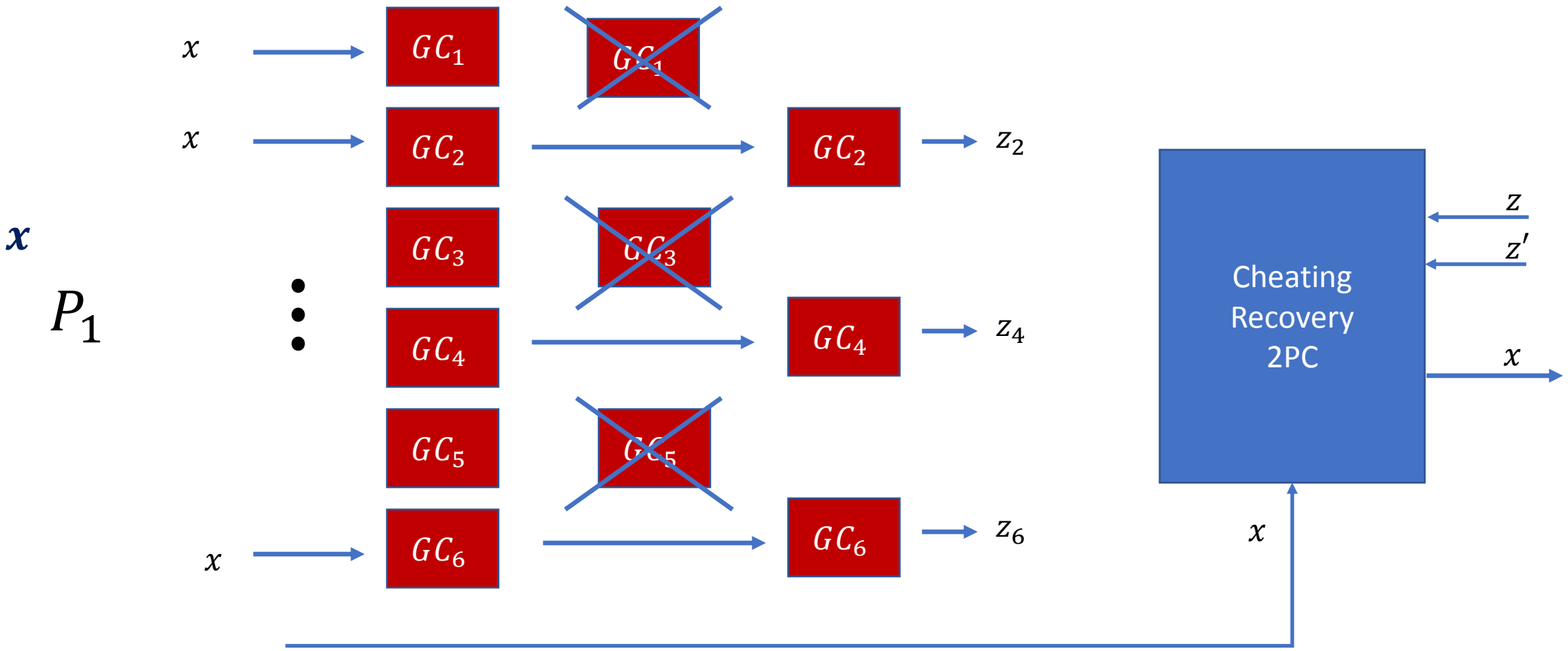
Garbler



Cut-and-Choose 2PC (majority)



Cut-and-Choose 2PC (Forge and Lose)

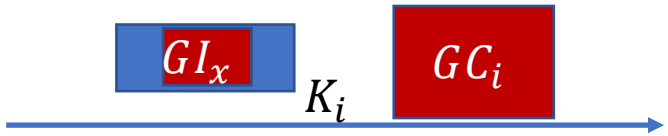


Homomorphic Commitments

- *Hiding and Binding*
- $HCOM(a, d_a), HCOM(b, d_b)$
  - Open to $a \oplus b$, using opening $d_a \oplus d_b$
- Pedersen commitments
- OT-based Commitments [LR15]
  - Non-interactive, rate $1/\lambda$
- (OT+ code)-based commitments [FJNT16]
  - Constant rate, interactive setup
  - Fiat-Shamir

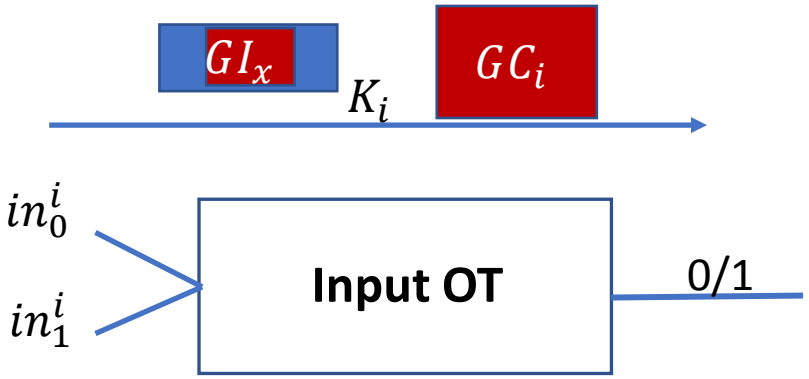
Single NISC

$GC_i \leftarrow Garb(C, sd_i)$



Single NISC

$GC_i \leftarrow Garb(C, sd_i)$

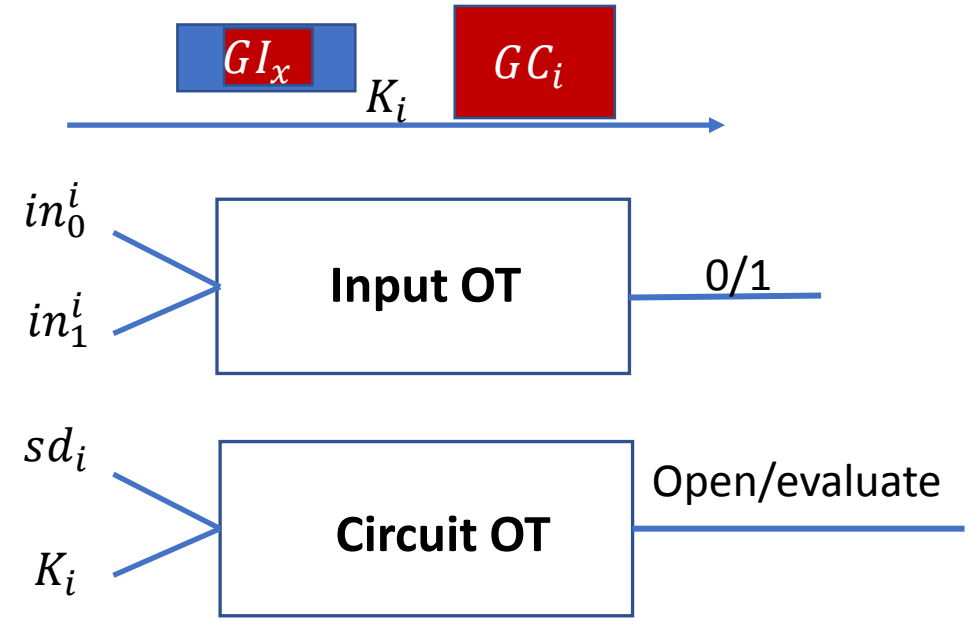


Evaluator input

Probe-resistant encoding

Single NISC

$GC_i \leftarrow Garb(C, sd_i)$



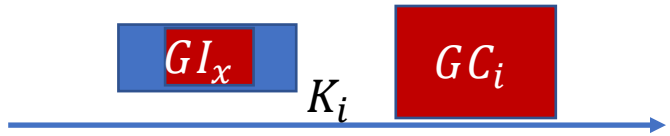
Evaluator input

Cut and choose

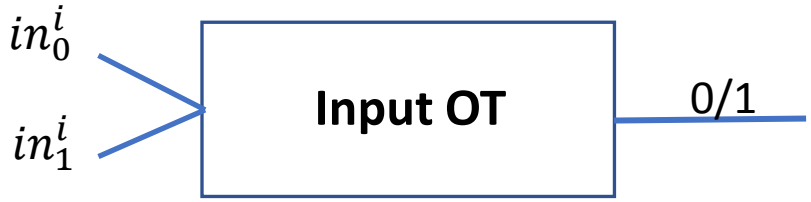
Probe-resistant encoding

Single NISC

$GC_i \leftarrow Garb(C, sd_i)$



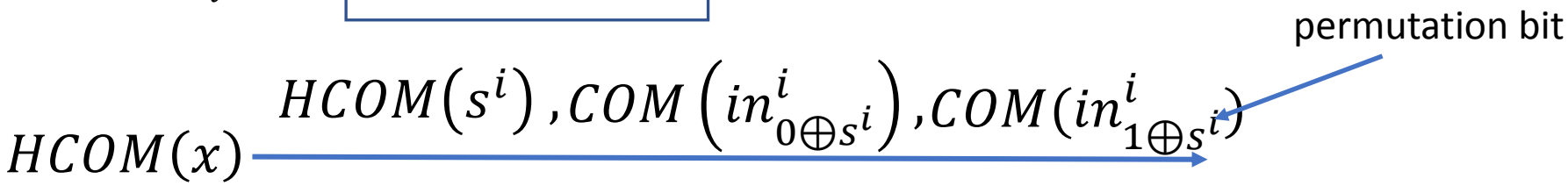
Evaluator input



Probe-resistant encoding

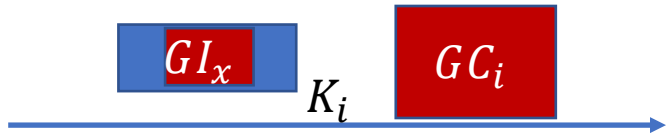
Cut and choose

Garbler input



Single NISC

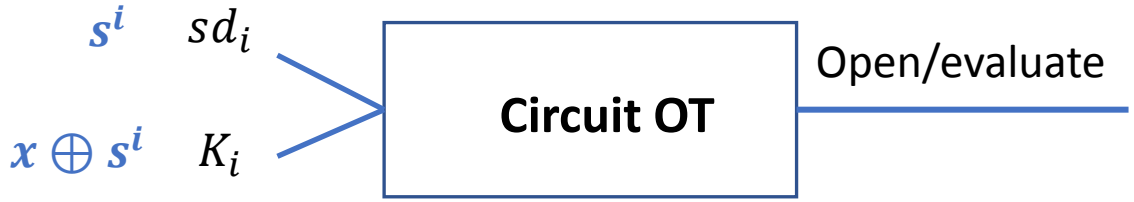
$GC_i \leftarrow Garb(C, sd_i)$



Evaluator input

Probe-resistant encoding

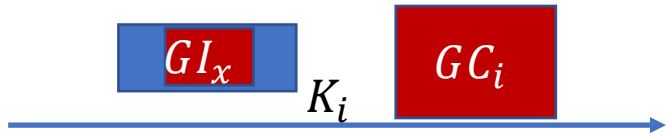
Cut and choose



Garbler input

Single NISC

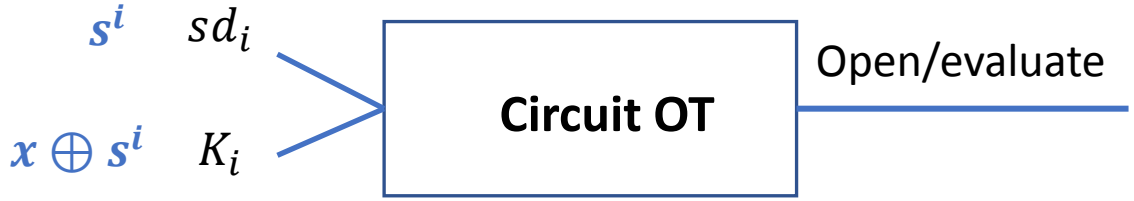
$GC_i \leftarrow Garb(C, sd_i)$



Evaluator input

Probe-resistant encoding

Cut and choose



permutation bit

Garbler input

$HCOM(x)$ $HCOM(s^i), COM\left(in^i_{0\oplus s^i}\right), COM(in^i_{1\oplus s^i})$

Cheating recovery

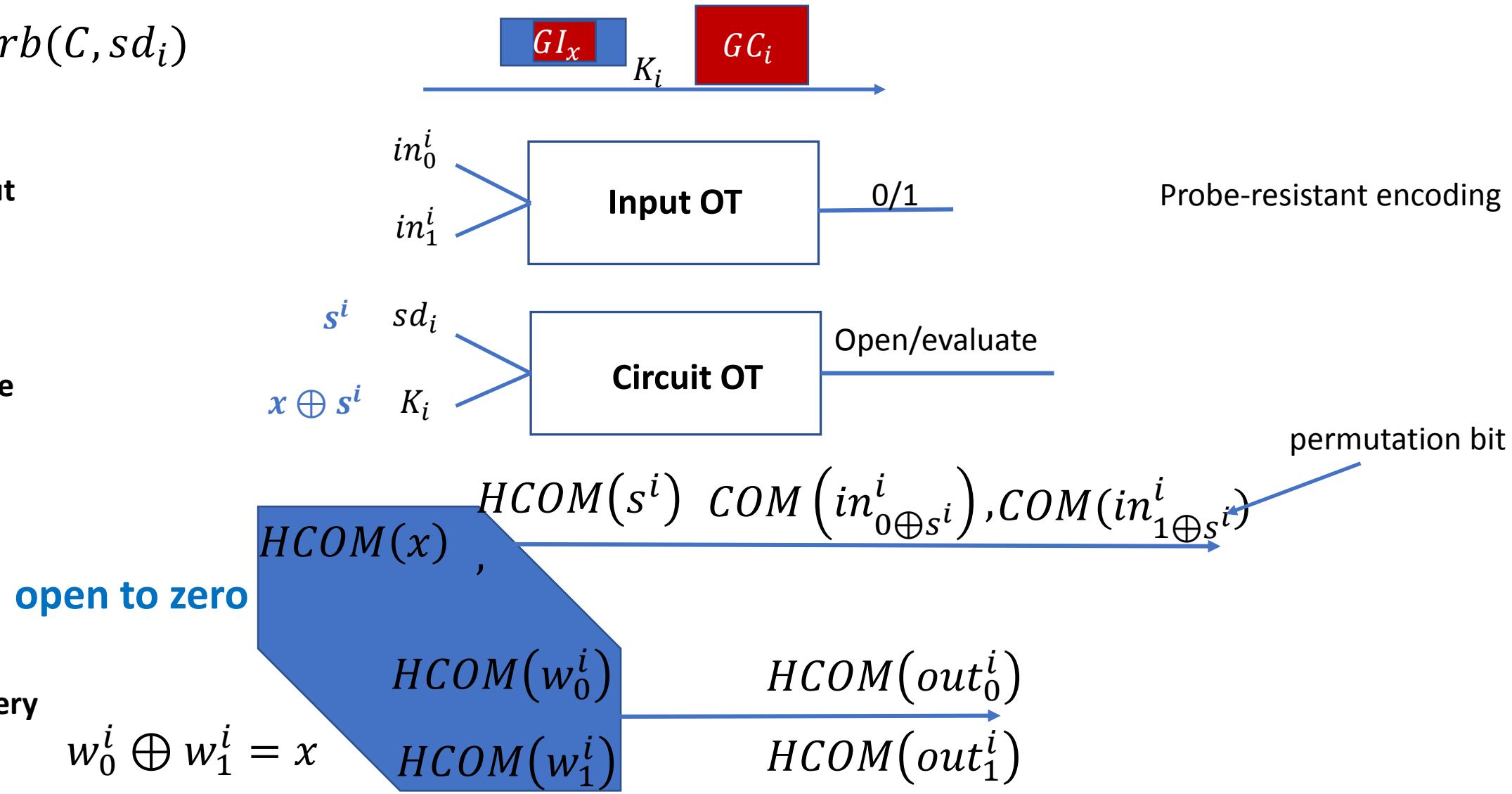
$w_0^i \oplus w_1^i = x$

$HCOM(w_0^i)$ $HCOM(out_0^i)$

$HCOM(w_1^i)$ $HCOM(out_1^i)$

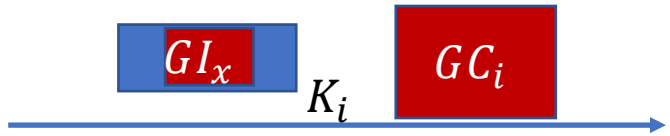
Single NISC

$GC_i \leftarrow Garb(C, sd_i)$



Single NISC

$GC_i \leftarrow Garb(C, sd_i)$



Evaluator input

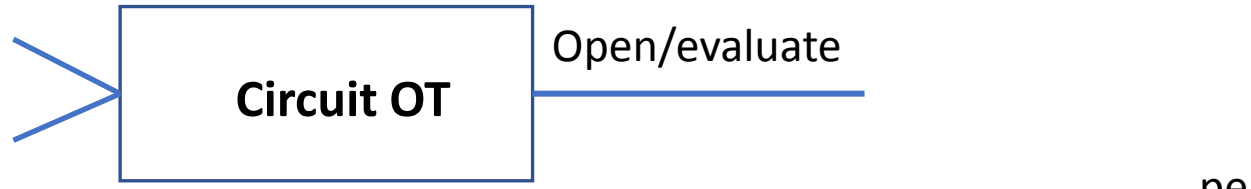
Probe-resistant encoding

Cut and choose

out_0^i, out_1^i s^i sd_i

$out_0^i \oplus w_0^i$ $x \oplus s^i$ K_i

$out_1^i \oplus w_1^i$



permutation bit

Garbler input

$HCOM(s^i)$ $COM\left(in^i_{0\oplus s^i}\right), COM(in^i_{1\oplus s^i})$

$HCOM(x)$,

open to zero

$HCOM(w_0^i)$

$HCOM(w_1^i)$

Cheating recovery

$w_0^i \oplus w_1^i = x$

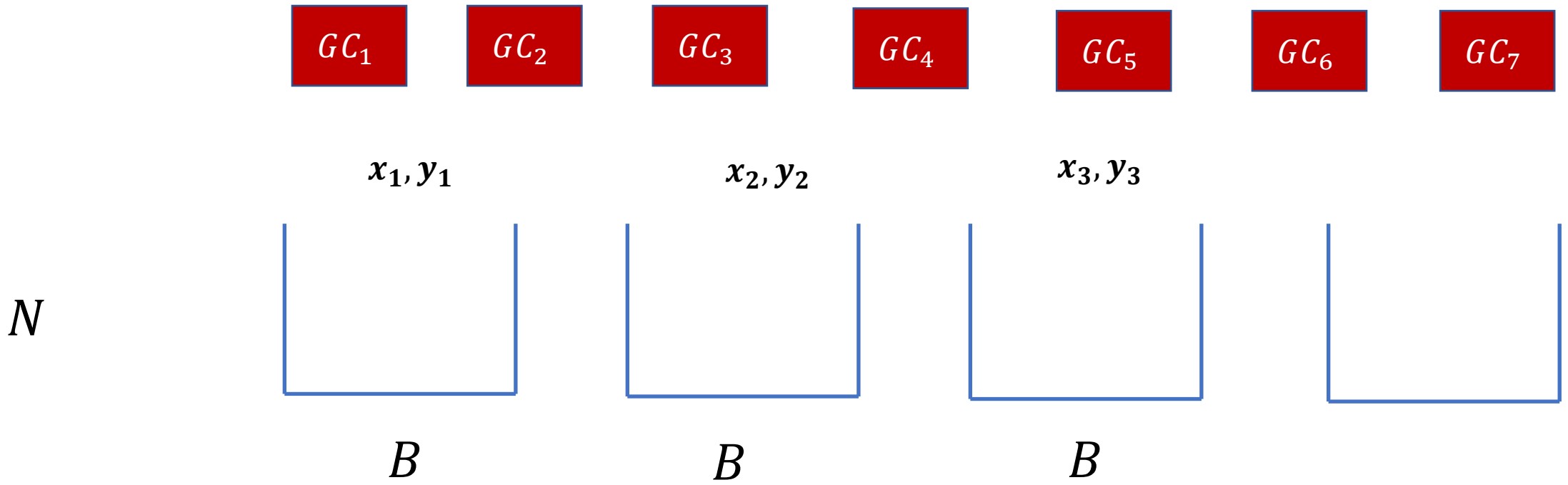
$HCOM(out_0^i)$

$HCOM(out_1^i)$

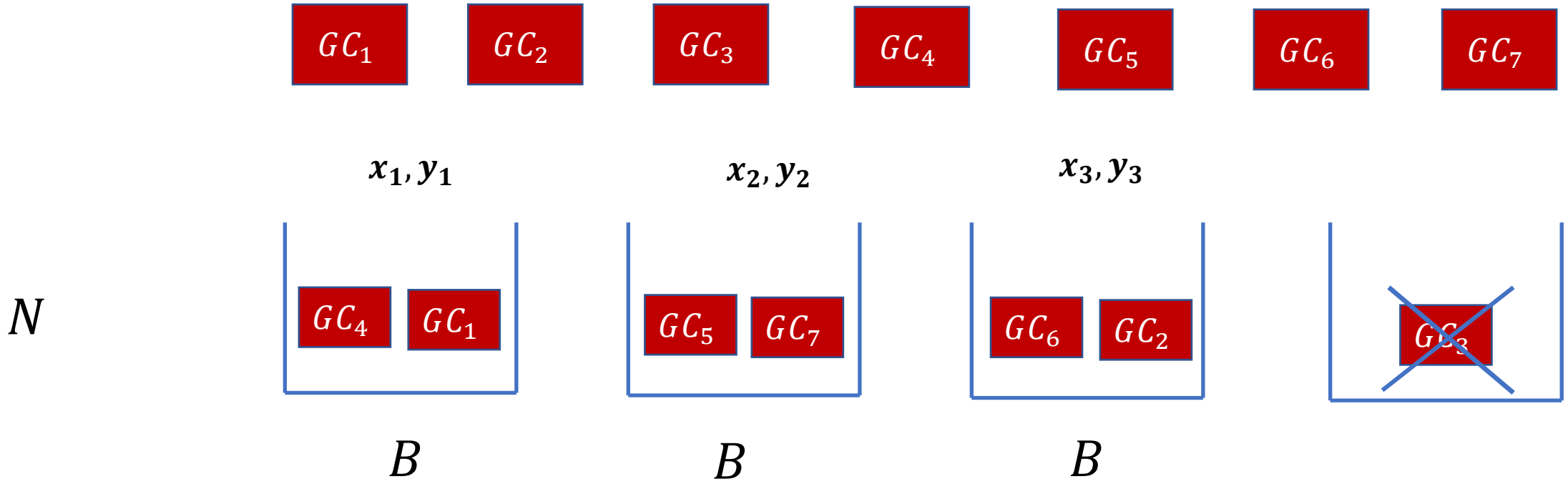
Batch 2PC

Batch 2PC

Batch 2PC

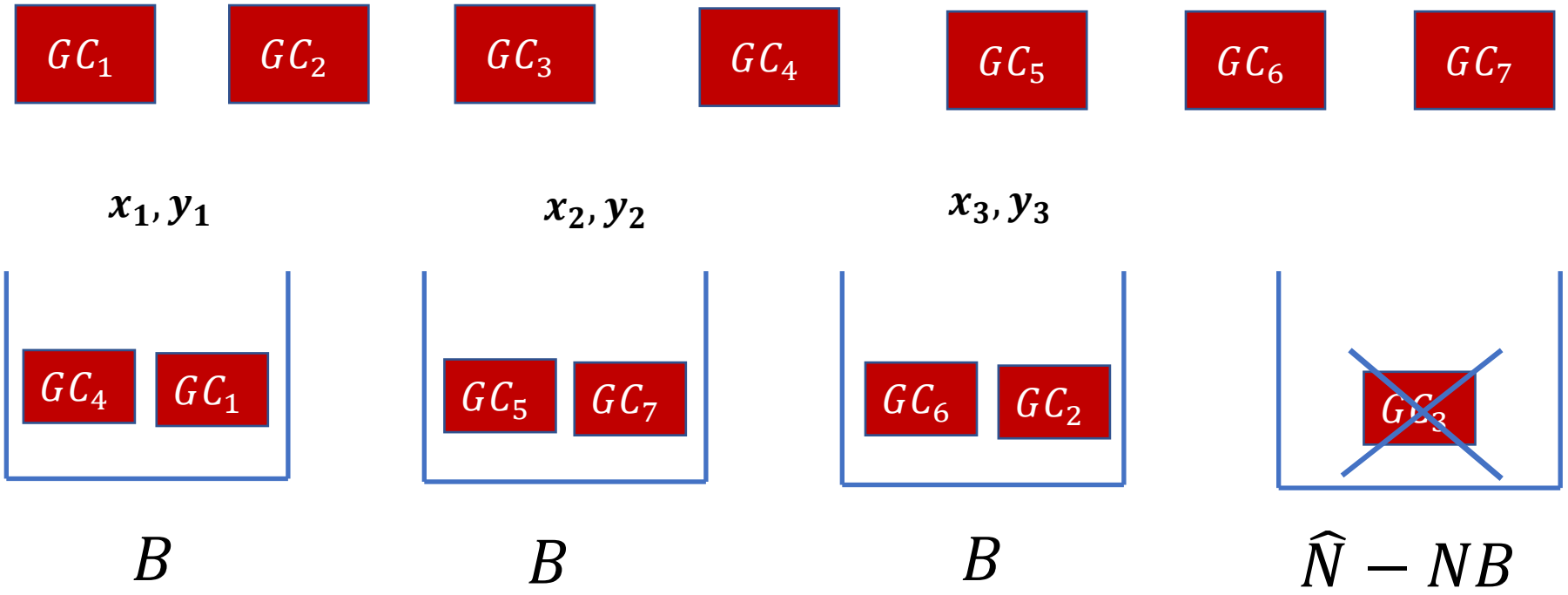


Batch 2PC


GC_1
GC_2
GC_3
GC_4
GC_5
GC_6
GC_7
x_1, y_1
x_2, y_2
x_3, y_3
N
GC_4
GC_1
GC_5
GC_7
GC_6
GC_2
GC_3
B
B
B

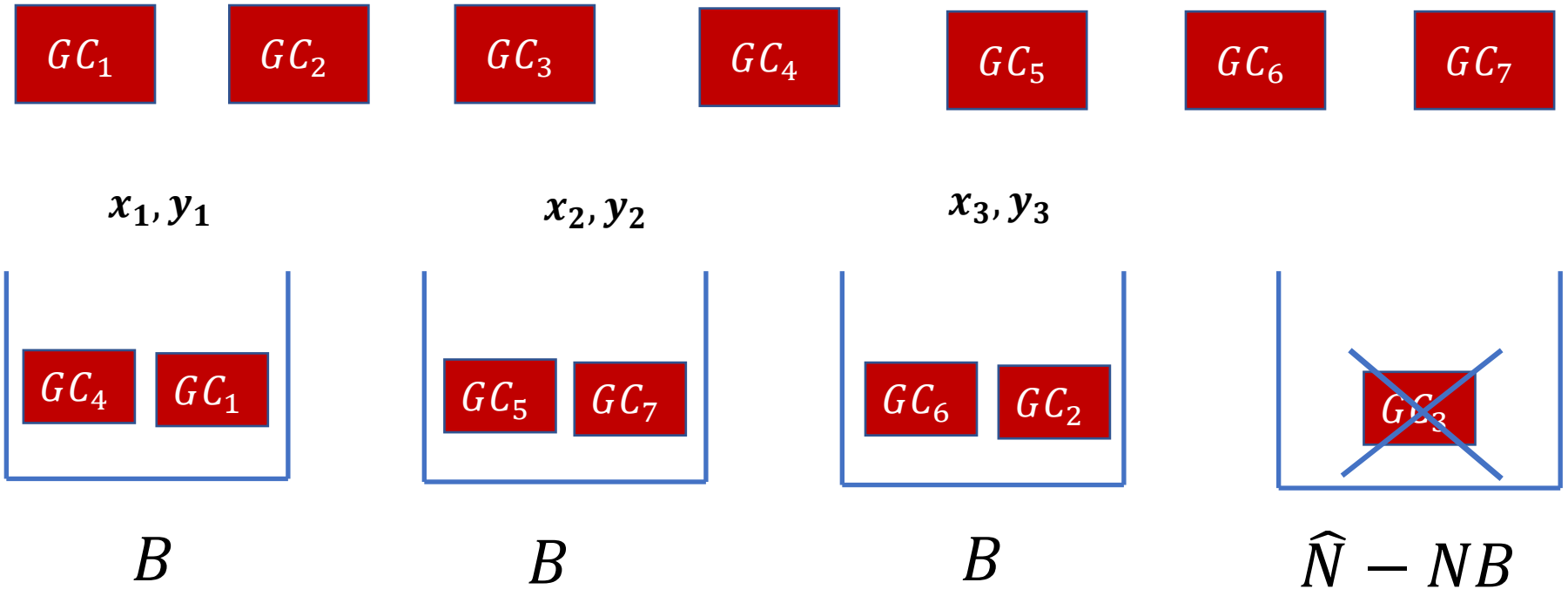
Batch 2PC

$$\widehat{N} = \frac{N\lambda}{logN}$$



Batch 2PC

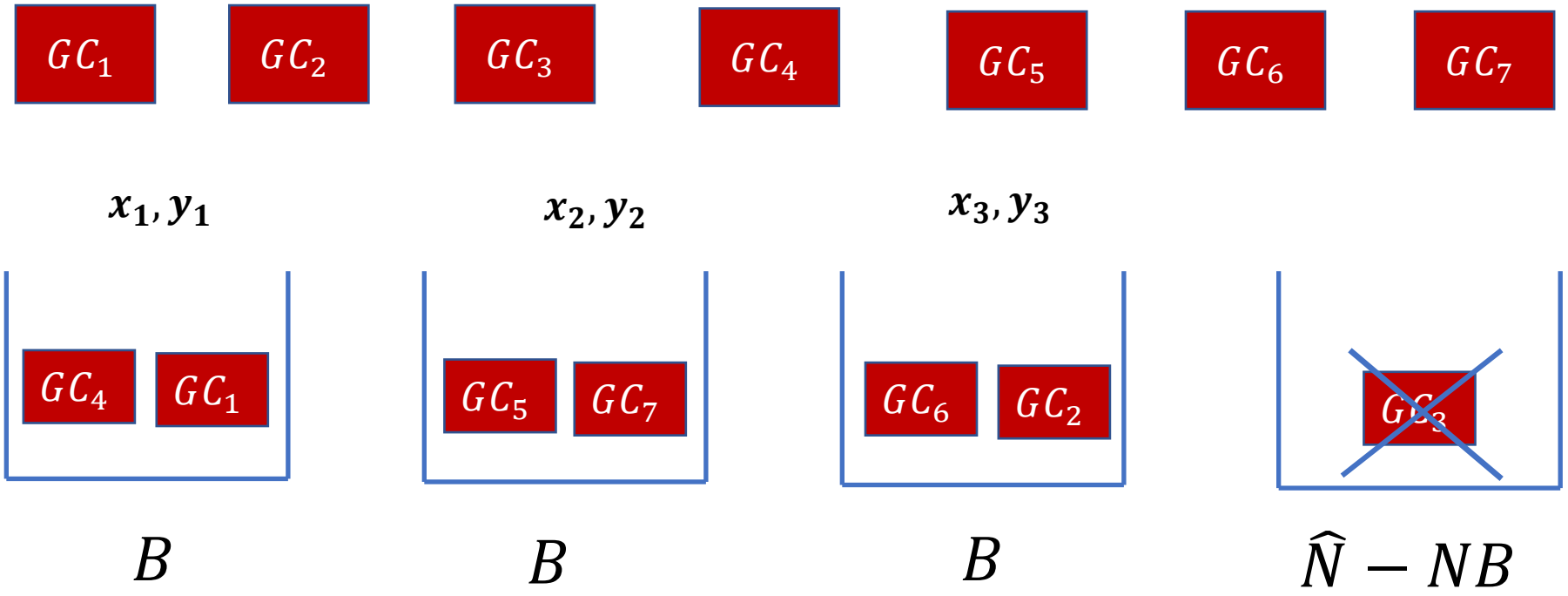
$$\widehat{N} = \frac{N\lambda}{logN}$$



1. Obliviously assign circuits to open/evaluate buckets

Batch 2PC

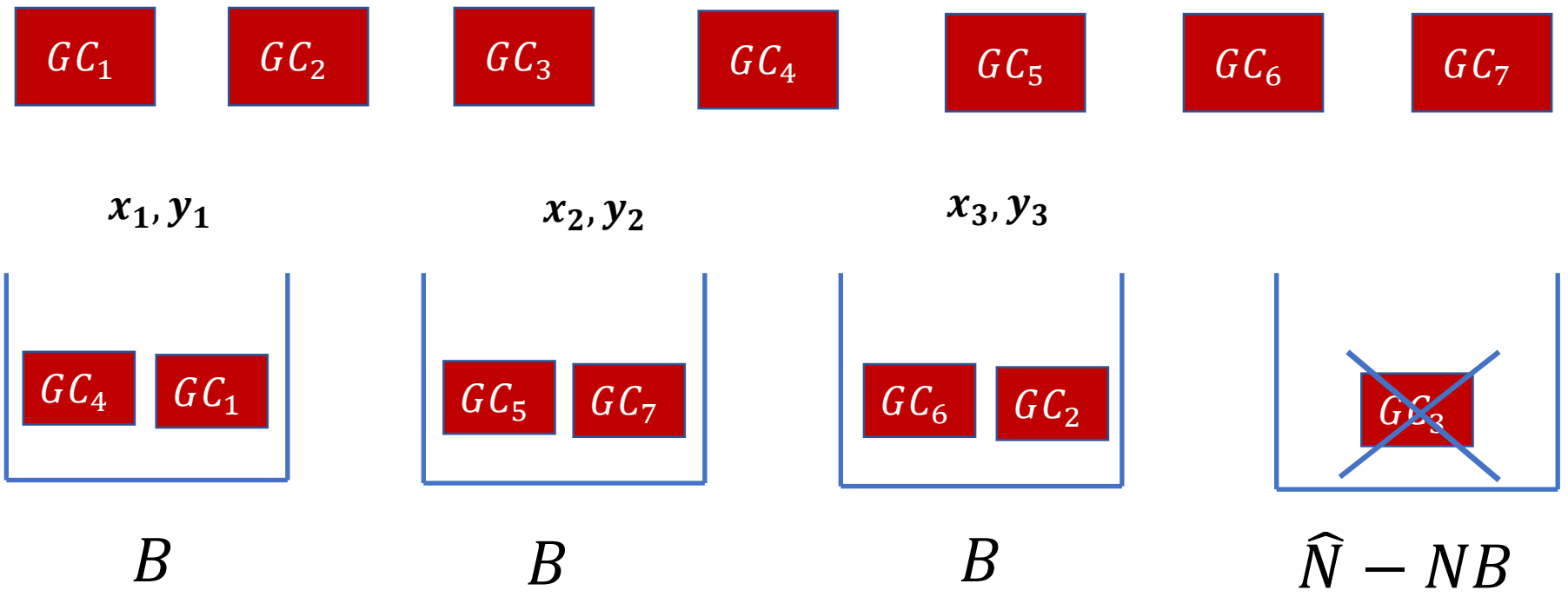
$$\widehat{N} = \frac{N\lambda}{logN}$$



1. Obliviously assign circuits to open/evaluate buckets
2. Garble inputs before knowing assignment

Batch 2PC

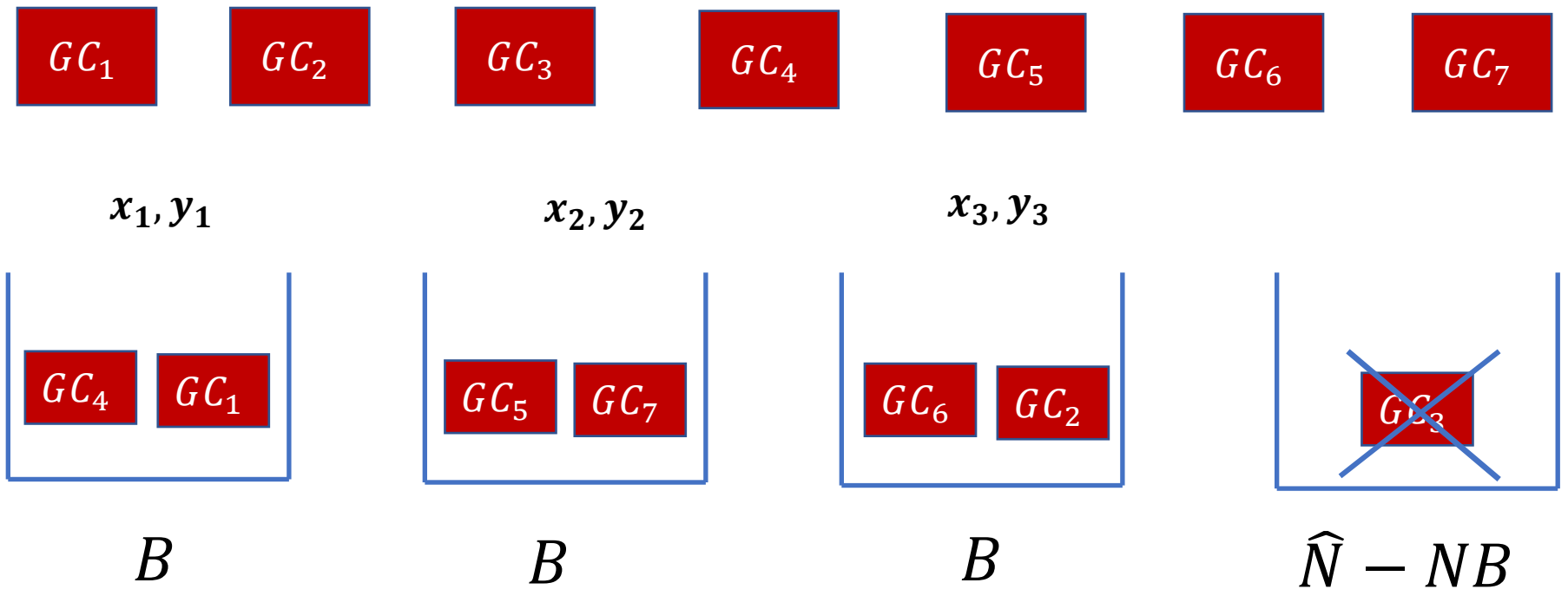
$$\widehat{N} = \frac{N\lambda}{logN}$$



1. Obliviously assign circuits to open/evaluate buckets
2. Garble inputs before knowing assignment
3. Input consistency before knowing assignment

Batch 2PC

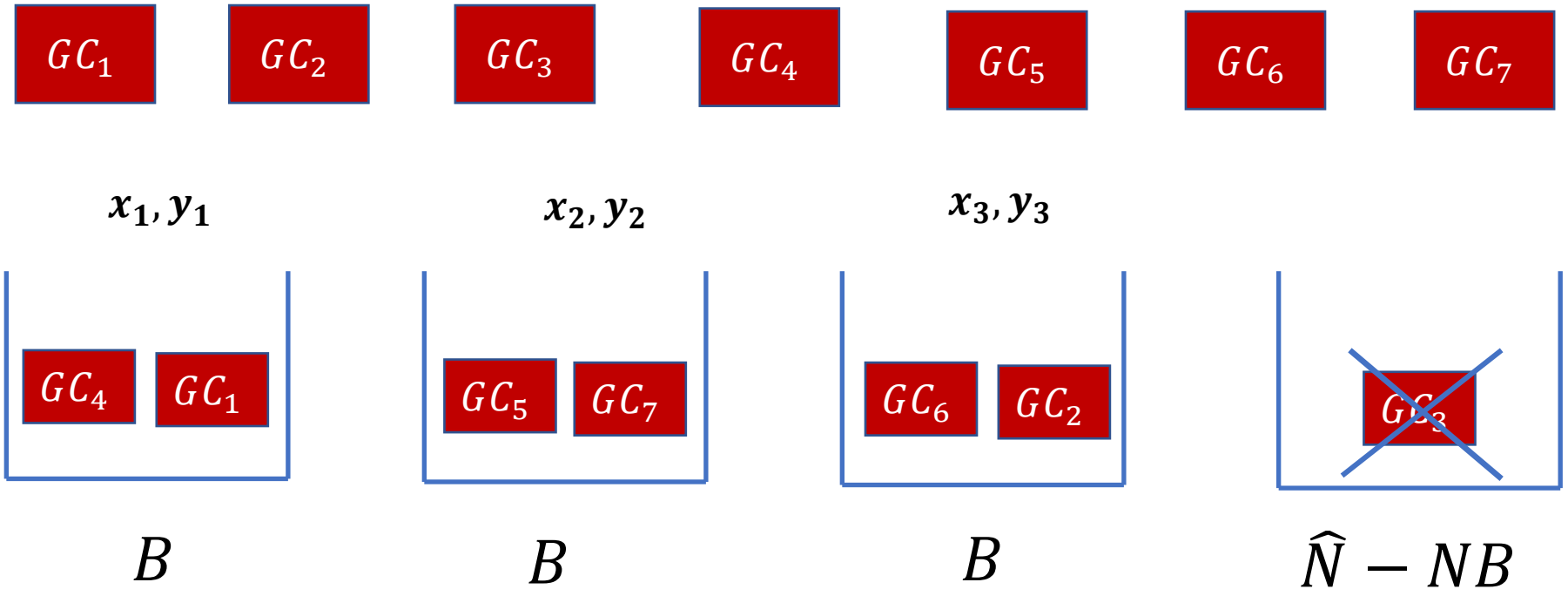
$$\widehat{N} = \frac{N\lambda}{logN}$$



1. Obliviously assign circuits to open/evaluate buckets
2. Garble inputs before knowing assignment
3. Input consistency before knowing assignment
4. Output recovery before knowing assignment

Batch 2PC

$$\widehat{N} = \frac{N\lambda}{logN}$$



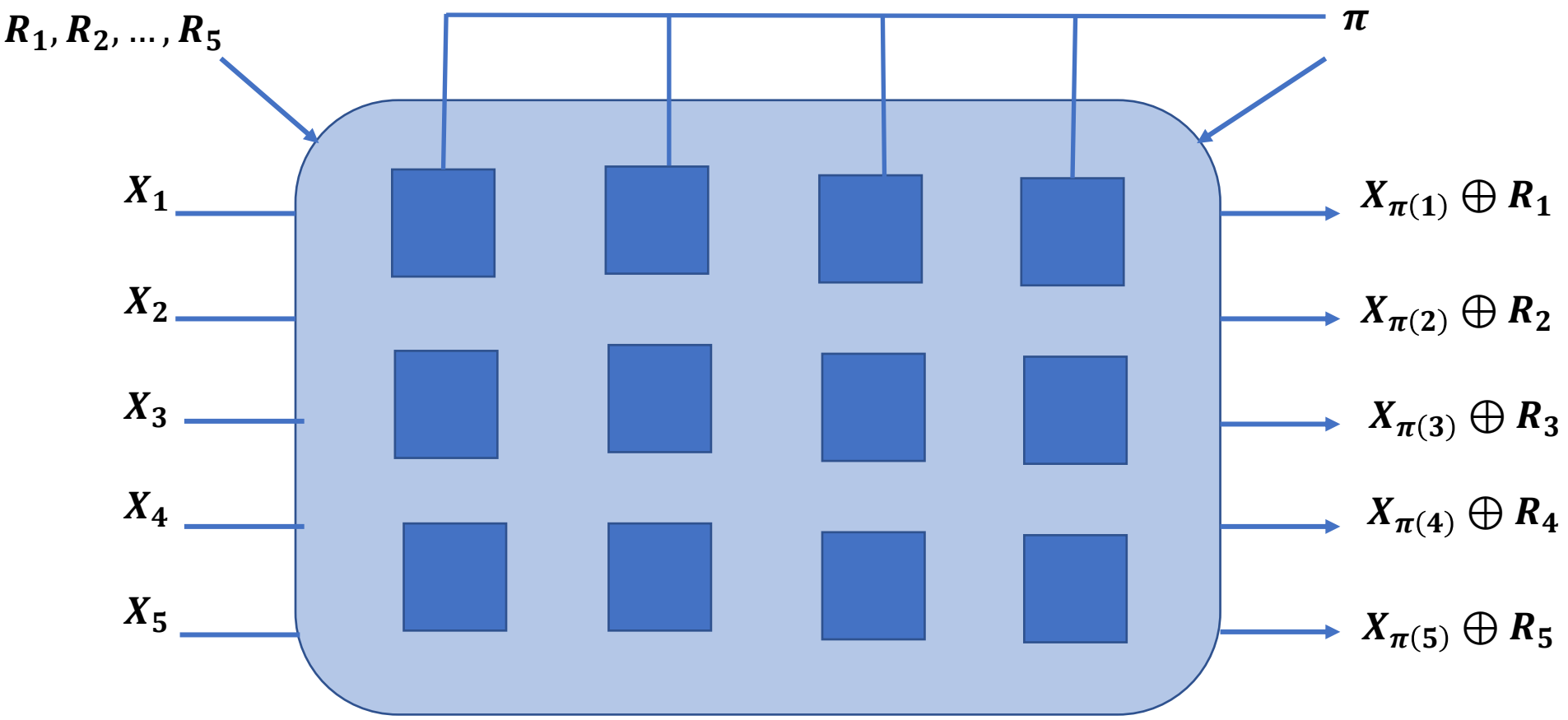
1. Obliviously assign circuits to open/evaluate buckets

Naive Solution: Prepare garbled inputs and gadgets for all N possibilities
Perform 1-out-of-N OT for each circuit

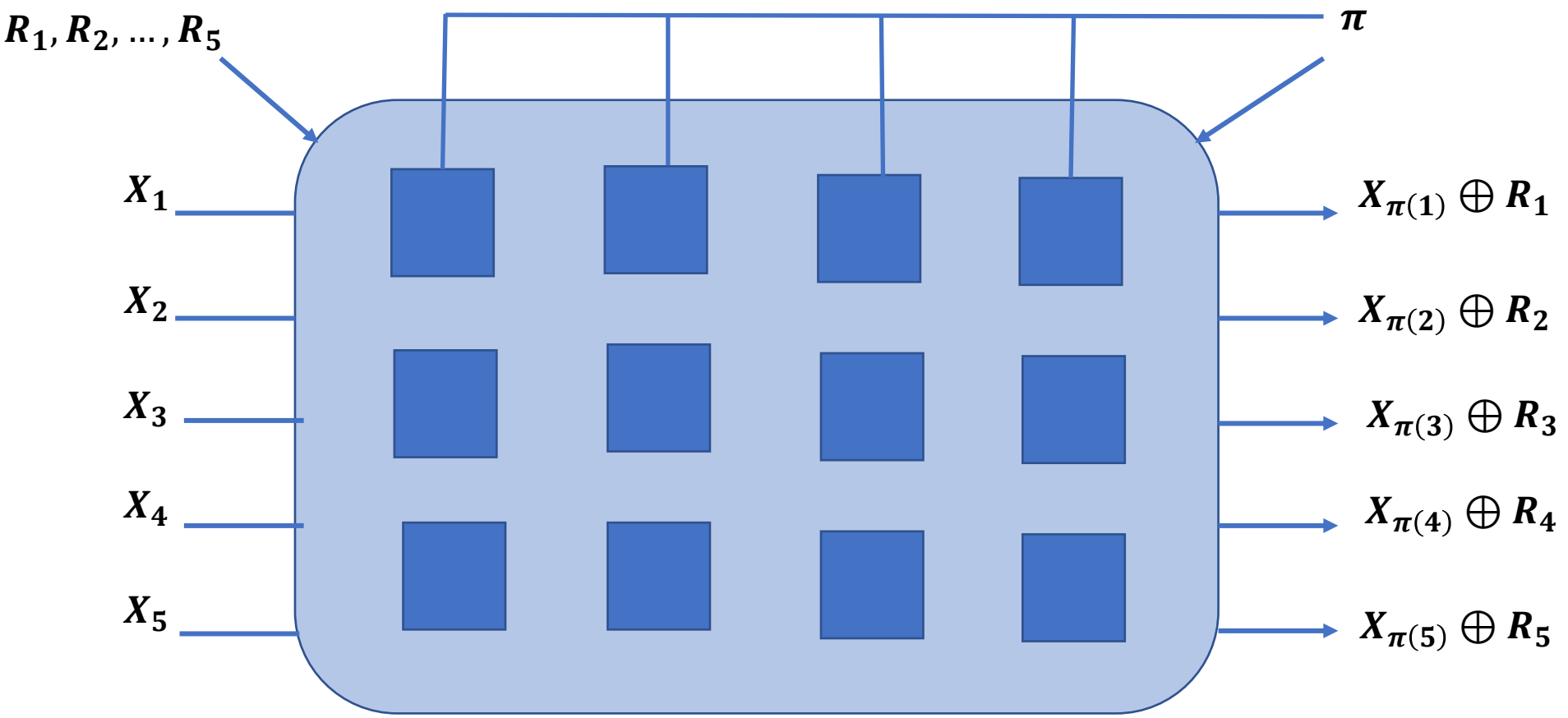
3. Input consistency before knowing assignment

4. Output recovery before knowing assignment

Oblivious Switching Networks (OSN)

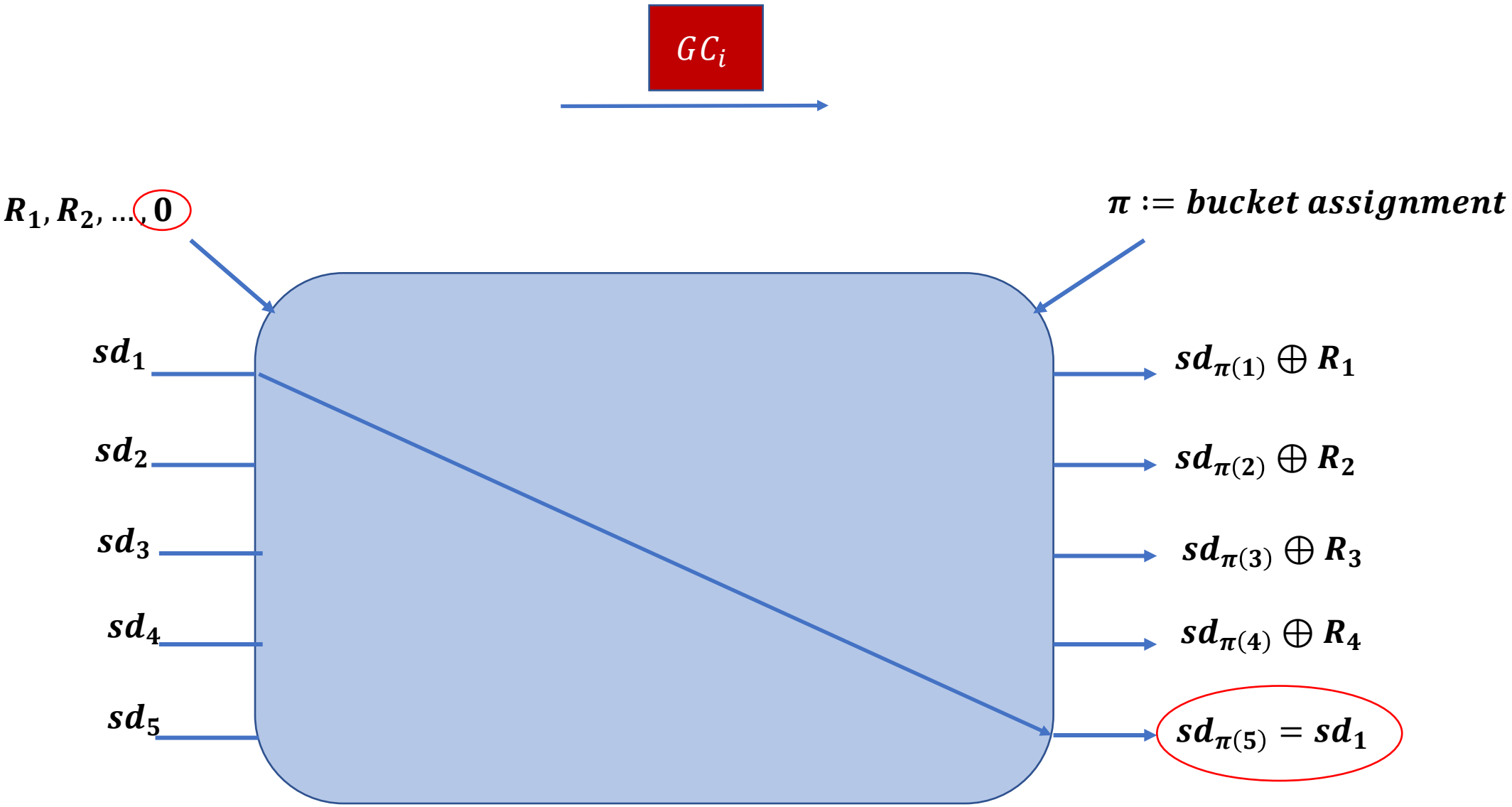


Oblivious Switching Networks (OSN)



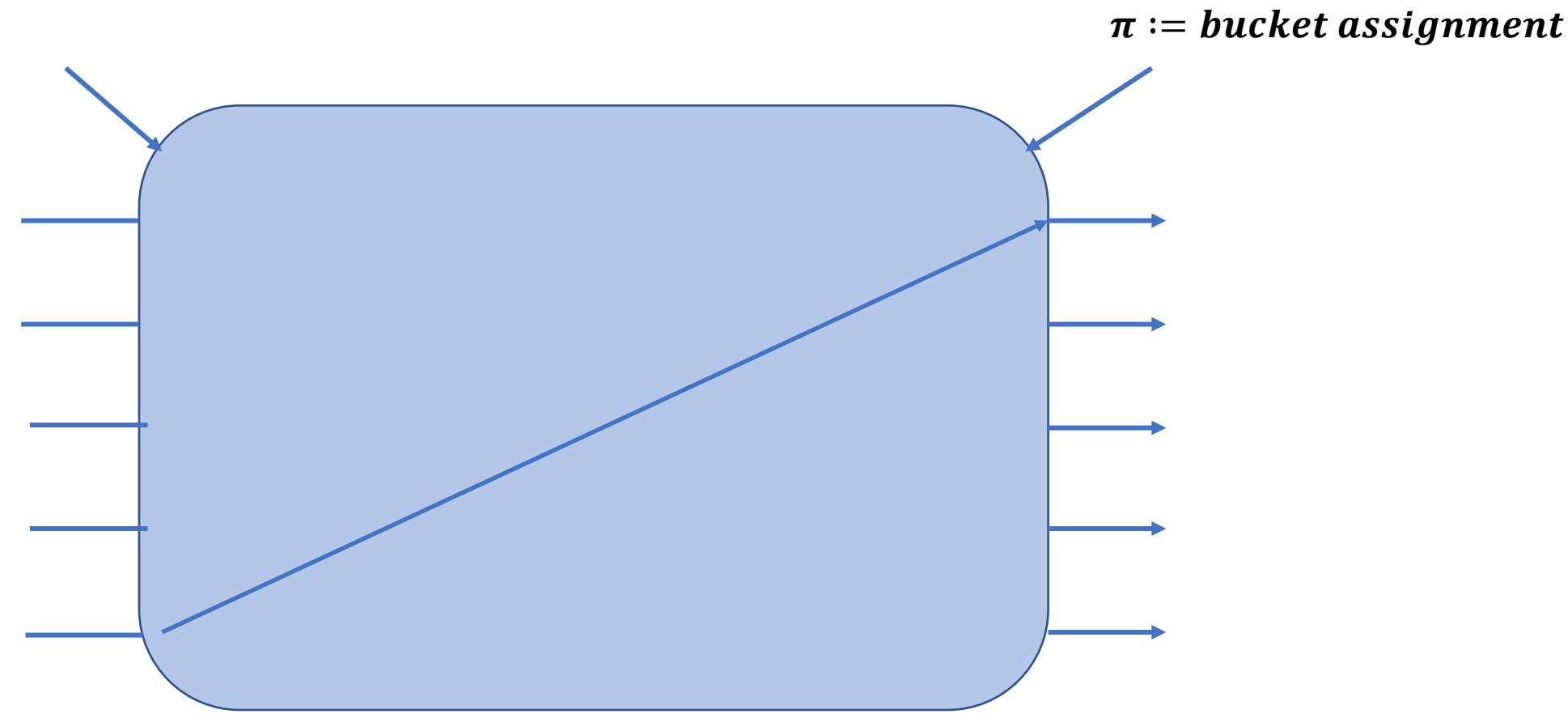
[MS13]: $O(nlogn)$ *parallel OTs*

Cut and Choose



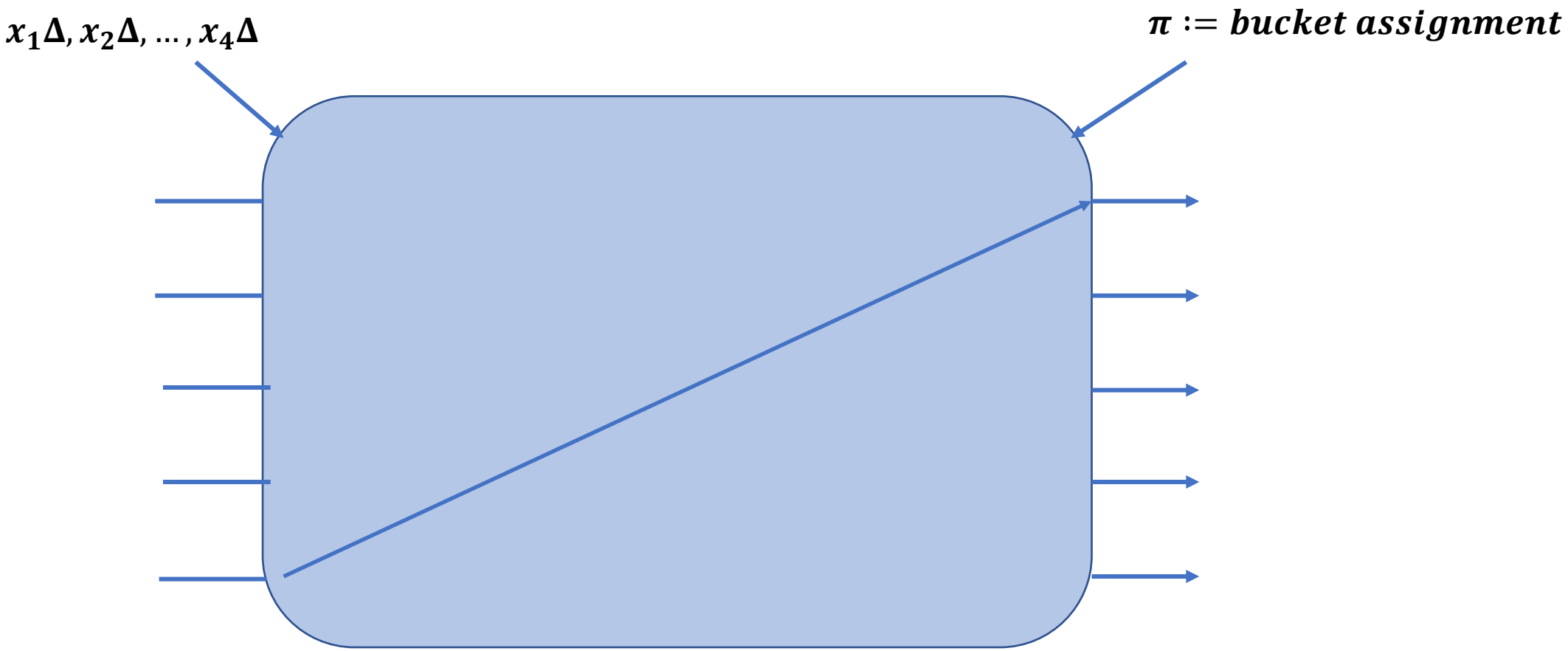
Garbler Input

$$COM\left(k^i_{b\oplus s^i}, d^i_{b\oplus s^i}\right) for\ b \in \{0,1\}$$



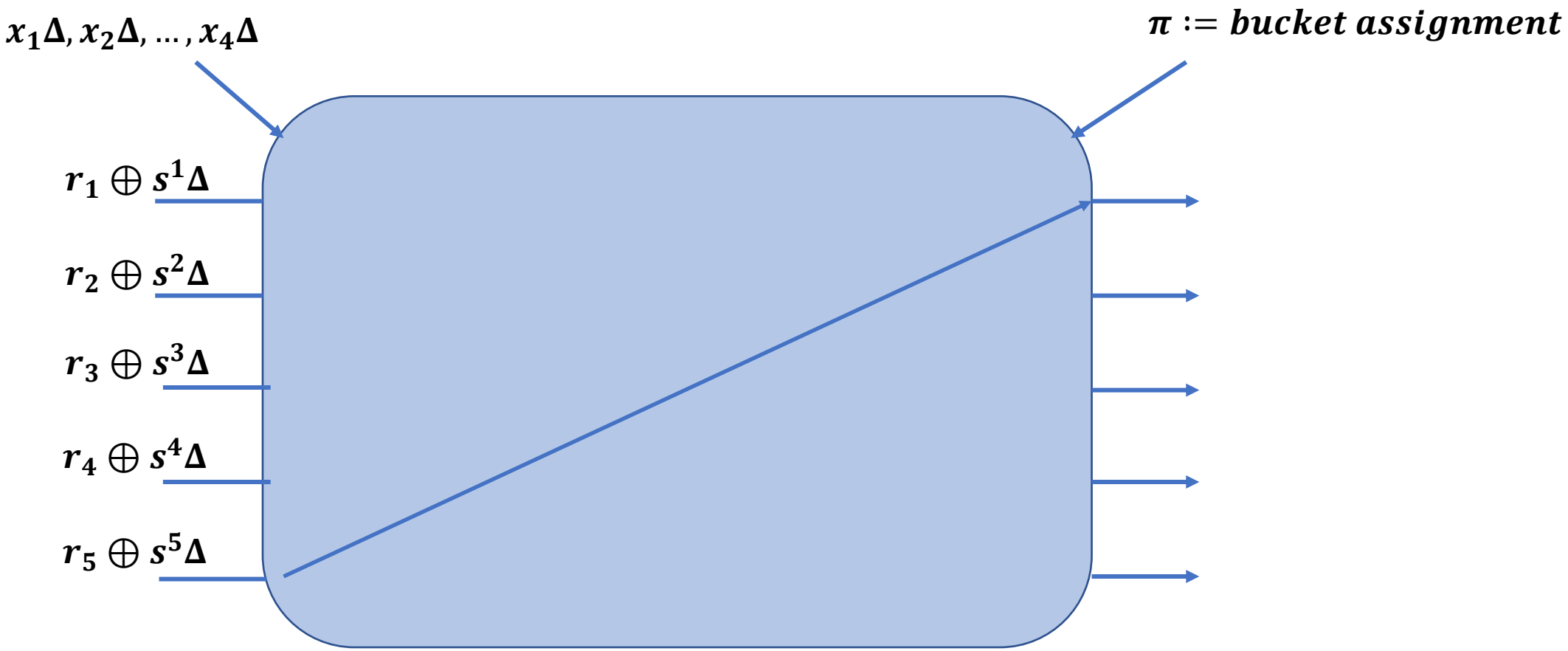
Garbler Input

$$COM\left(k^i_{b\oplus s^i}, d^i_{b\oplus s^i}\right) for\ b \in \{0,1\}$$



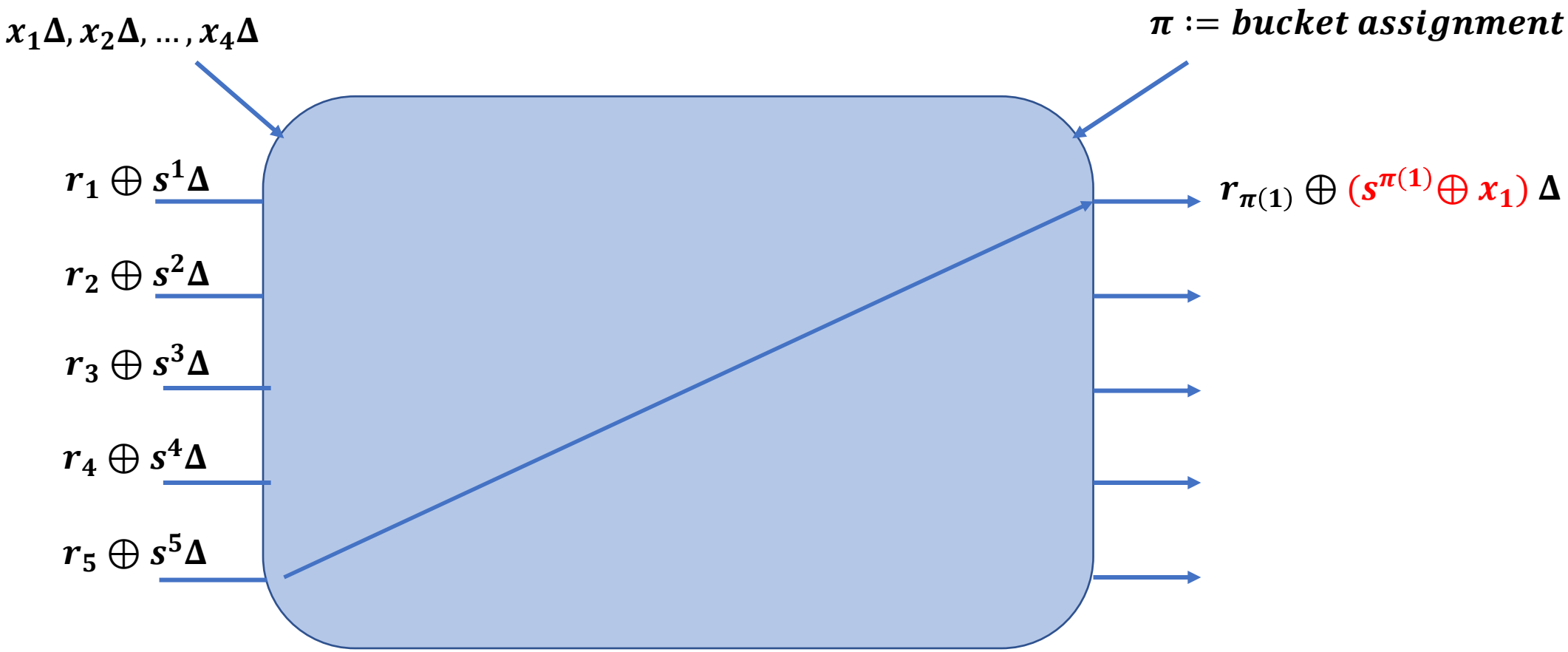
Garbler Input

$$COM\left(k^i_{b\oplus s^i}, d^i_{b\oplus s^i}\right) for\ b \in \{0,1\}$$

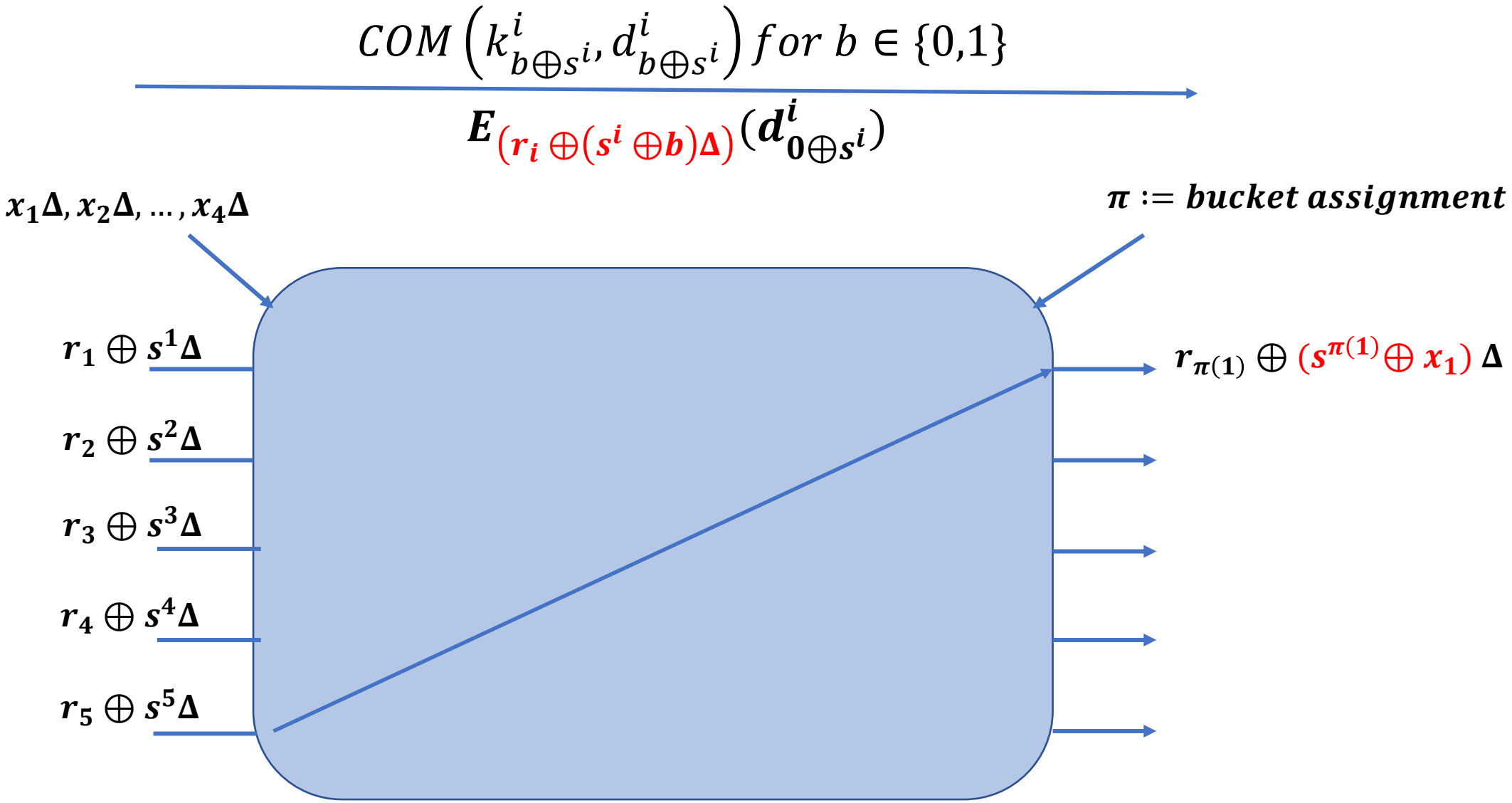


Garbler Input

$$COM\left(k^i_{b\oplus s^i}, d^i_{b\oplus s^i}\right) for\ b \in \{0,1\}$$

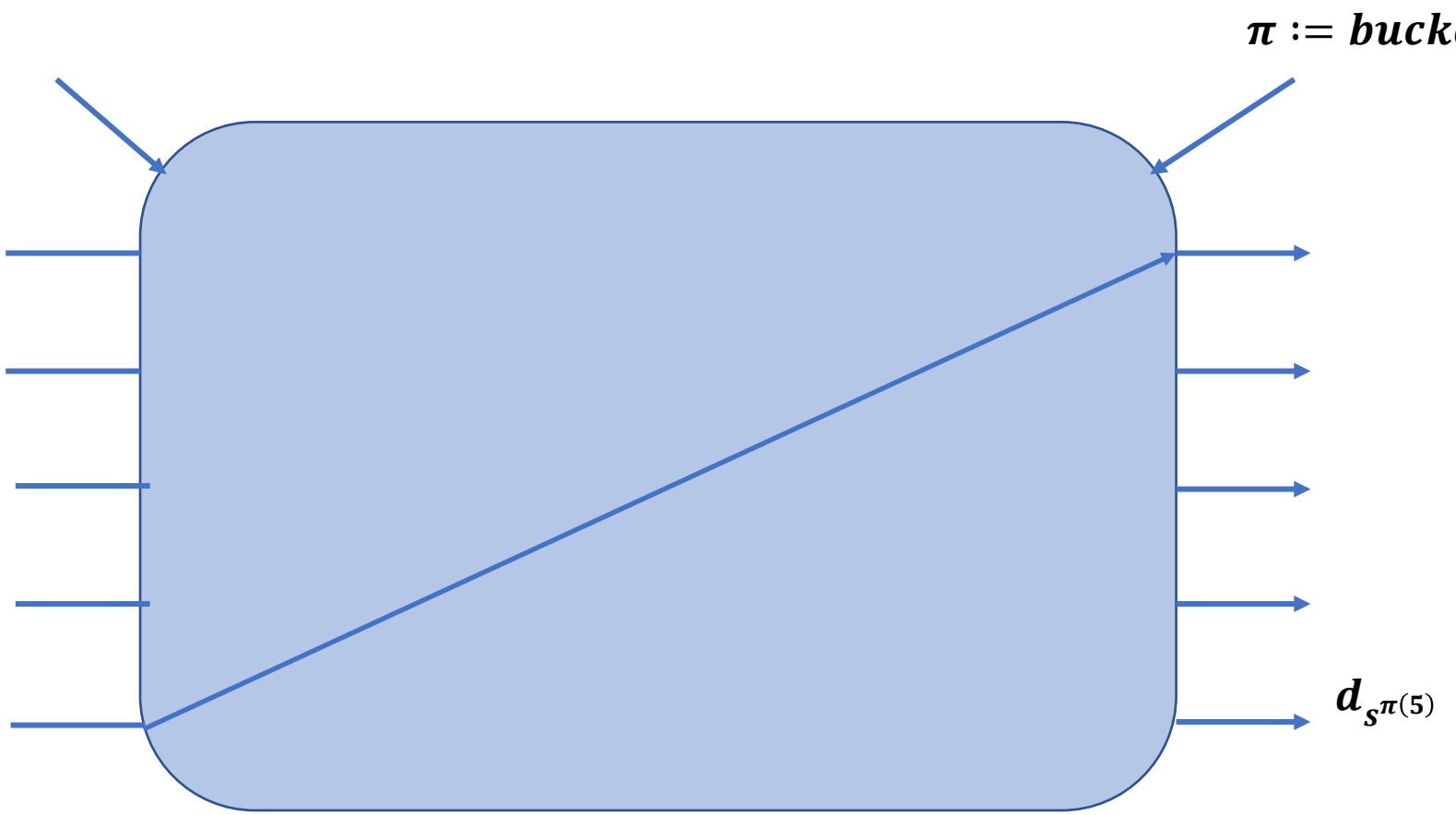


Garbler Input



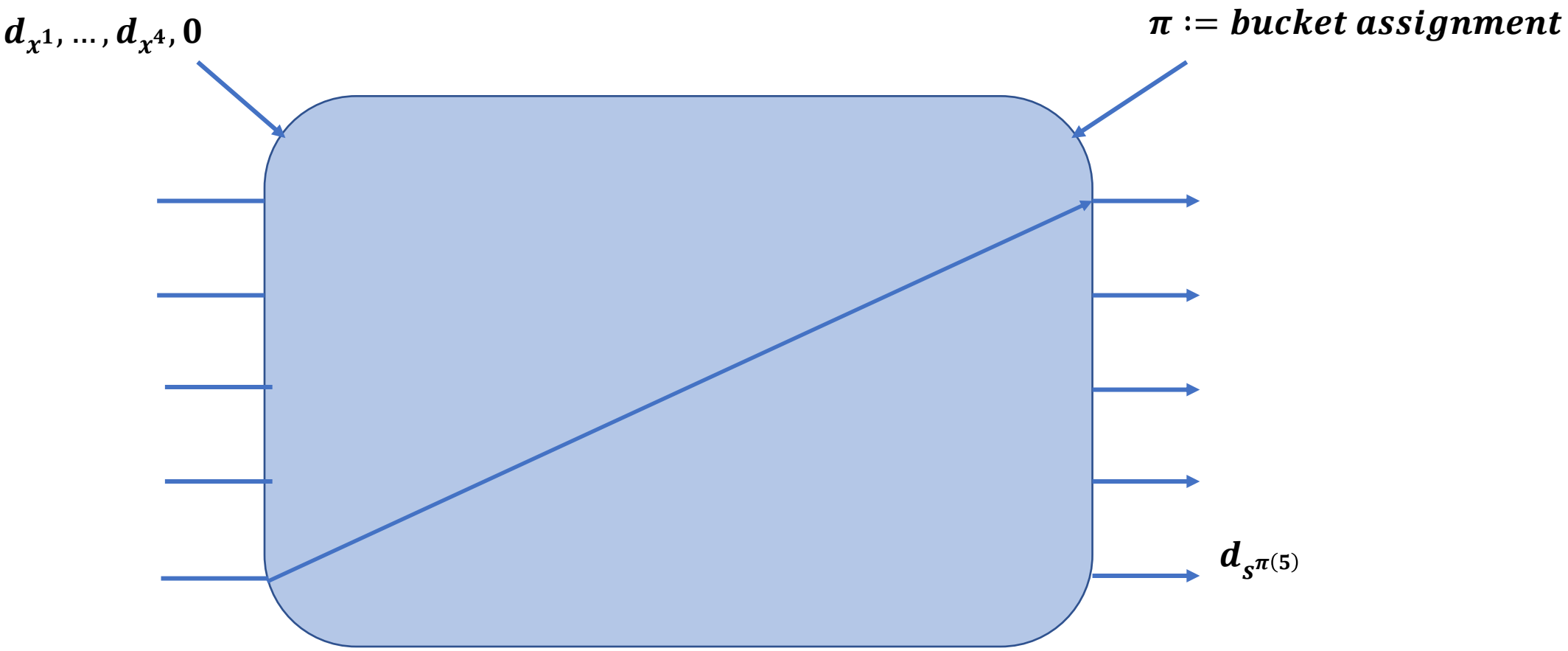
Garbler Input Consistency

$$HCOM(x^j, d_{x^j}), HCOM(s^i, d_{s^i}), HCOM(k^i_{b \oplus s^i}, d^i_{b \oplus s^i})$$



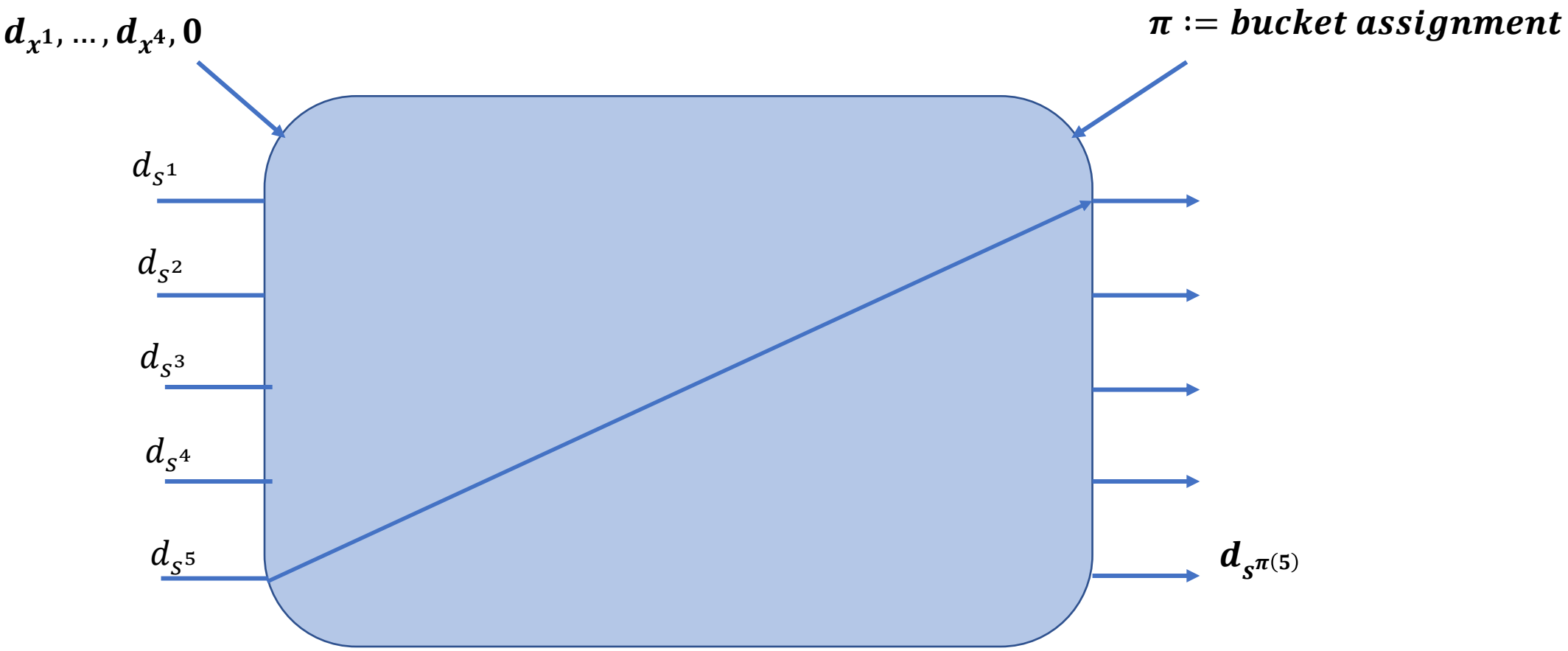
Garbler Input Consistency

$$HCOM(x^j, d_{x^j}), HCOM(s^i, d_{s^i}), HCOM(k^i_{b \oplus s^i}, d^i_{b \oplus s^i})$$



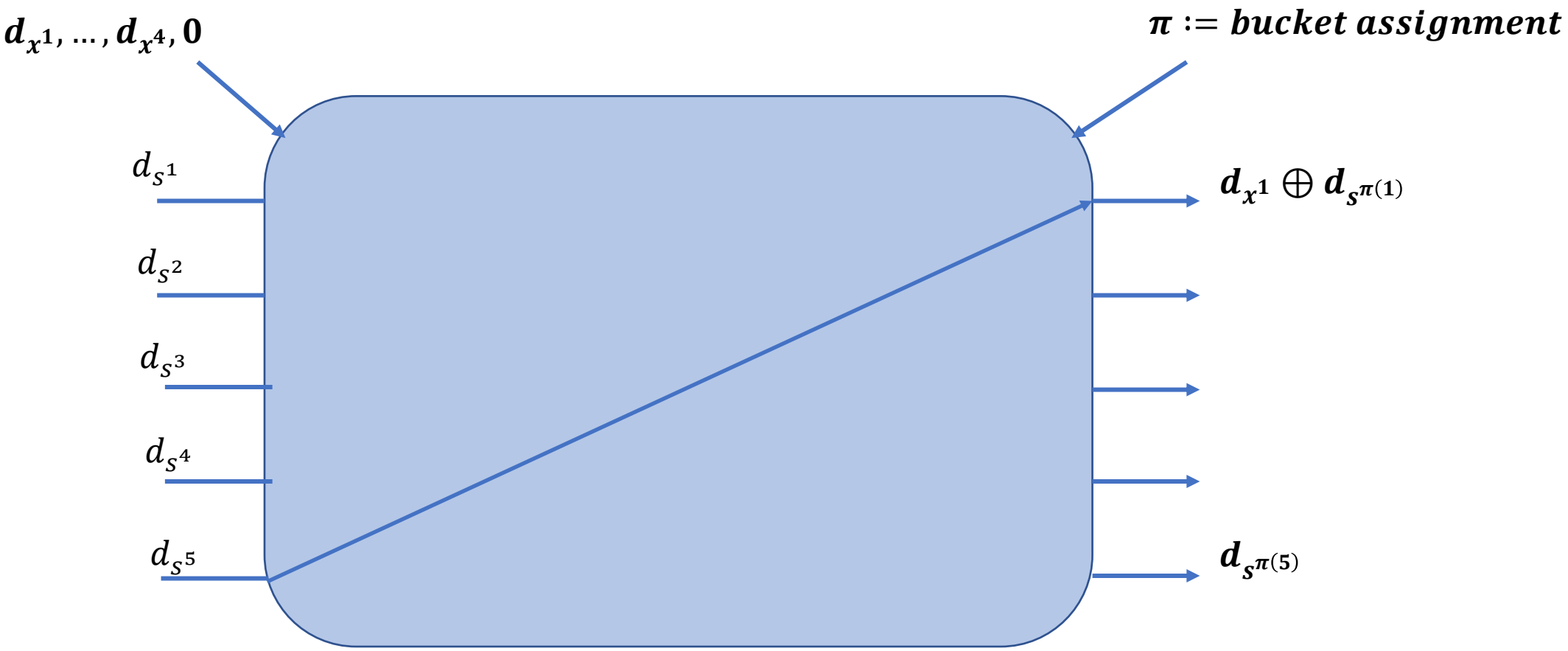
Garbler Input Consistency

$$HCOM(x^j, d_{x^j}), HCOM(s^i, d_{s^i}), HCOM(k^i_{b \oplus s^i}, d^i_{b \oplus s^i})$$



Garbler Input Consistency

$$HCOM(x^j, d_{x^j}), HCOM(s^i, d_{s^i}), HCOM(k^i_{b \oplus s^i}, d^i_{b \oplus s^i})$$



Evaluator input

Cheating recovery

Sending decommitments through the OSN

Summary

Summary

λ: *statistical sec.*
κ: *computation sec.*
N: *of exectutions*
n_{out}: *of outputs*
n_{in}: *# of inputs*

	NISC	RO-NISC	online-offline
rounds	$0+2$	$0+2$	$2+2$
# GC	$O(N\lambda/\log N)$	$O(N\kappa/\log N)$	$O(N\lambda/\log N)$
# plain commit	$O(n_{\mathsf{in}}N\lambda/\log N)$	$O(n_{\mathsf{in}}N\kappa/\log N)$	$O(n_{\mathsf{in}}N\lambda/\log N)$
# hom commit	$O(n_{\mathsf{out}}N\lambda/\log N)$	$O(n_{\mathsf{out}}N\kappa/\log N)$	$O(n_{\mathsf{out}}N\lambda/\log N)$
OSN OTs	$O(n_{\mathsf{both}}N\lambda)$	-	-
other OTs	$O(n_{\mathsf{in}}N)$	$O(n_{\mathsf{in}}N)$	$O(n_{\mathsf{in}}N)$

Open Questions

- Two-round OT extension
- Non-interactive RAM 2PC
- ??? Some other nice problems …