OBLIVIOUS TRANSFER WITH CONSTANT COMPUTATIONAL OVERHEAD

Elette Boyle
IDC Herzliya,
NTT Research

Geoffroy Couteau
IRIF

Niv Gilboa
Ben-Gurion University

Yuval Ishai
Technion

Lisa Kohl
CWI

Nicolas Resch
UvA

Peter Scholl
Aarhus University

COMPUTATIONAL OVERHEAD

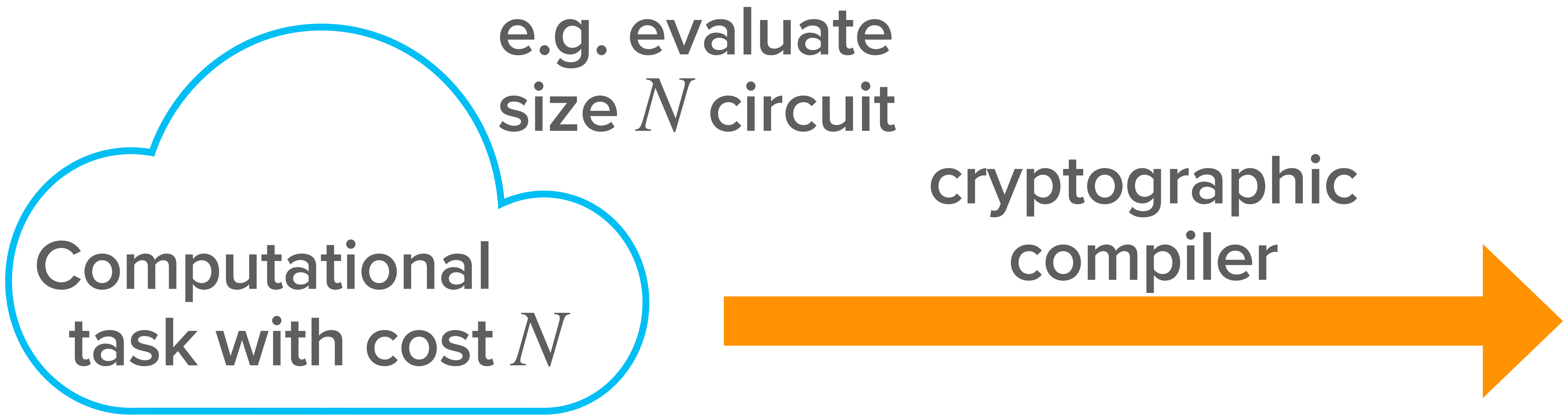
Computational
task with cost N

COMPUTATIONAL OVERHEAD

e.g. evaluate
size N circuit

Computational
task with cost N

COMPUTATIONAL OVERHEAD



COMPUTATIONAL OVERHEAD

COMPUTATIONAL OVERHEAD

New cost: typically $\geq C_\lambda N$, where C_λ grows with security parameter λ

COMPUTATIONAL OVERHEAD

New cost: typically $\geq C_\lambda N$, where C_λ grows with security parameter λ

Dream: cost independent of security level?

COMPUTATIONAL OVERHEAD

New cost: typically $\geq C_\lambda N$, where C_λ grows with security parameter λ

Dream: cost independent of security level?

Ishai, Kushilevitz, Ostrovsky and Sahai '08: constant comp. overhead for

COMPUTATIONAL OVERHEAD

New cost: typically $\geq C_\lambda N$, where C_λ grows with security parameter λ

Dream: cost independent of security level?

Ishai, Kushilevitz, Ostrovsky and Sahai '08: constant comp. overhead for

encryption

COMPUTATIONAL OVERHEAD

New cost: typically $\geq C_\lambda N$, where C_λ **grows** with security parameter λ

Dream: cost independent of security level?

Ishai, Kushilevitz, Ostrovsky and Sahai '08: constant comp. overhead for

encryption

signatures

COMPUTATIONAL OVERHEAD

New cost: typically $\geq C_\lambda N$, where C_λ grows with security parameter λ

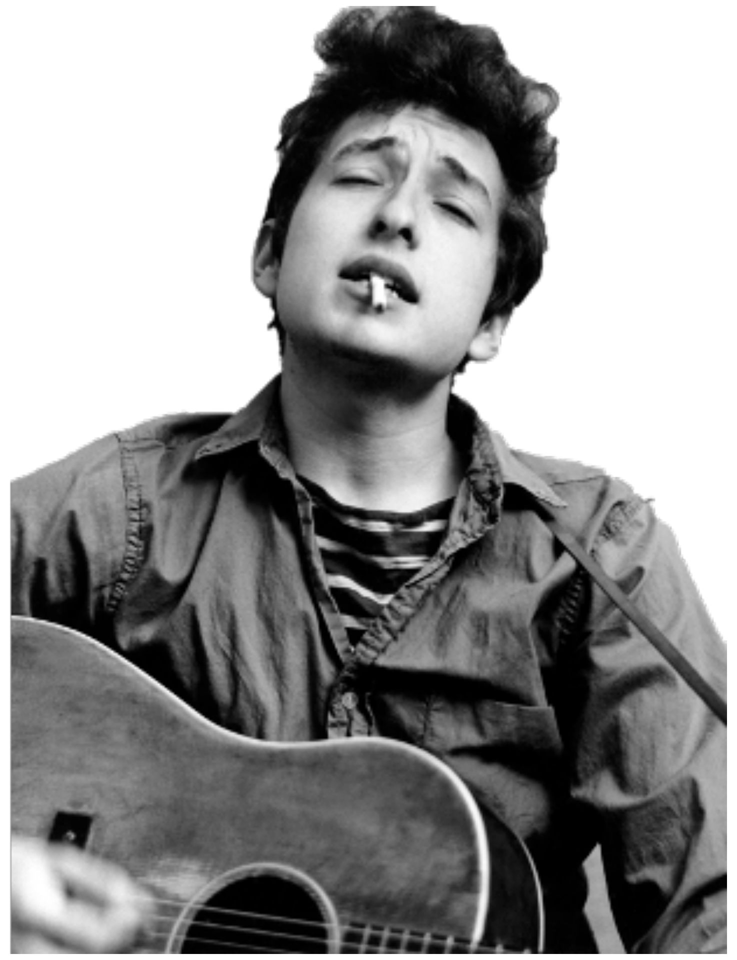
Dream: cost independent of security level?

Ishai, Kushilevitz, Ostrovsky and Sahai '08: constant comp. overhead for

- encryption
- signatures
- semi-honest 2PC

SECURE (2-PARTY) COMPUTATION (2PC)

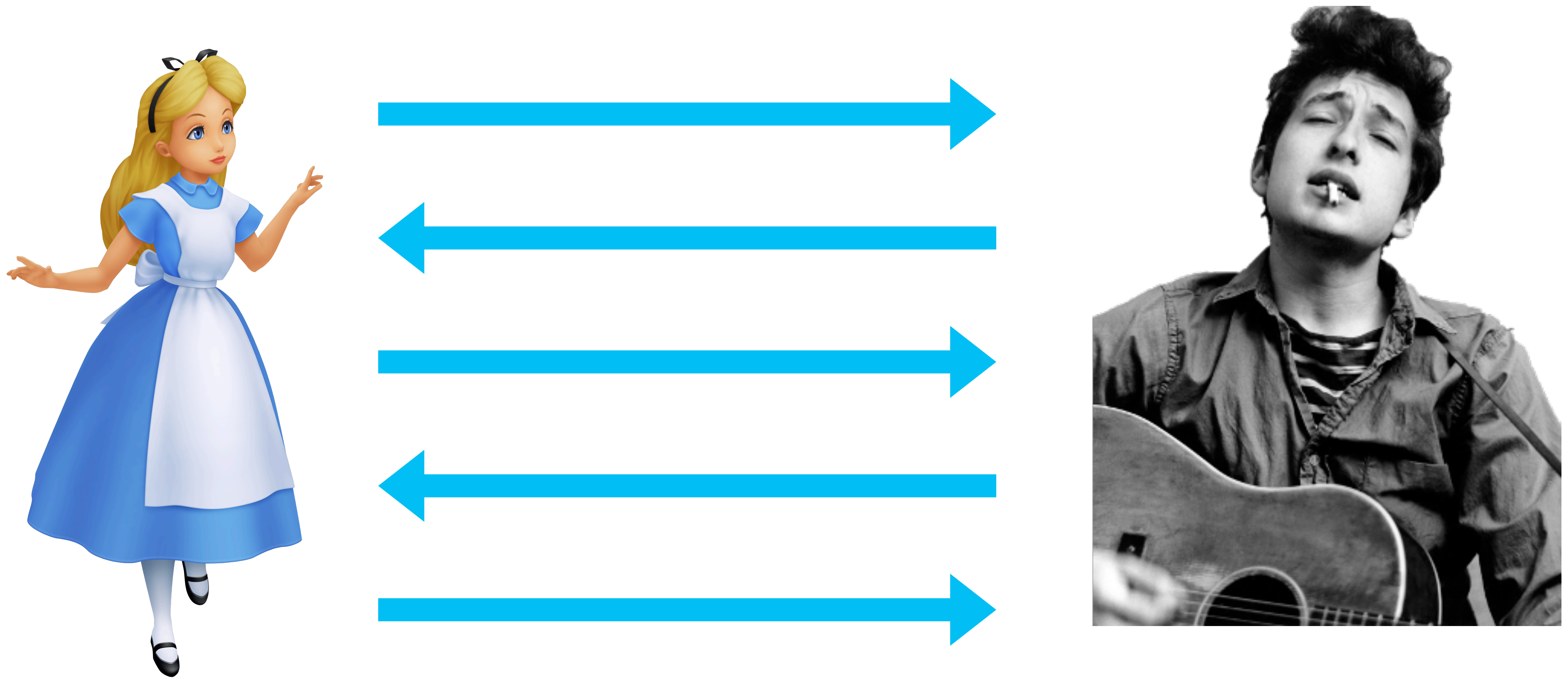
x



y

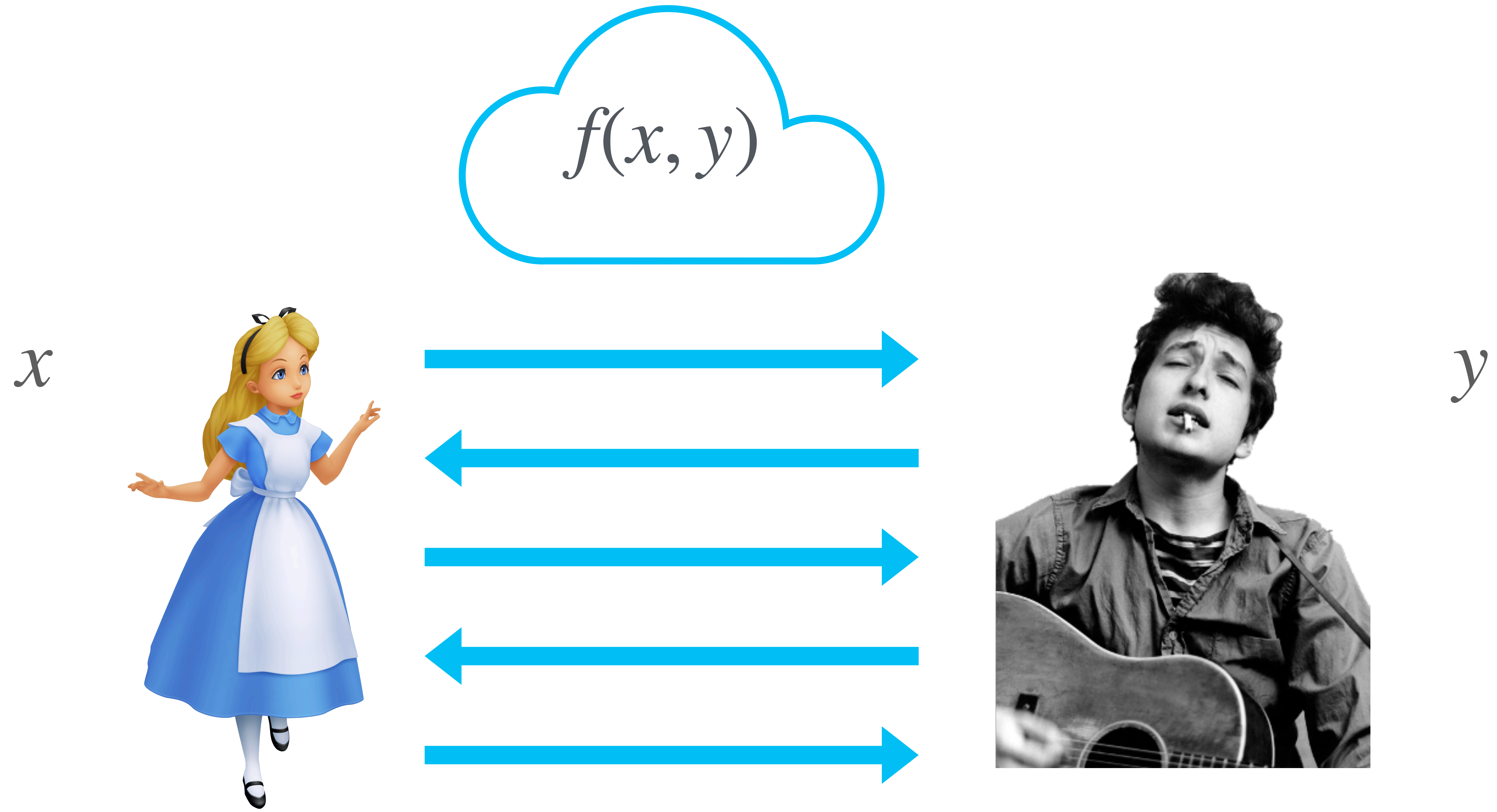
SECURE (2-PARTY) COMPUTATION (2PC)

x



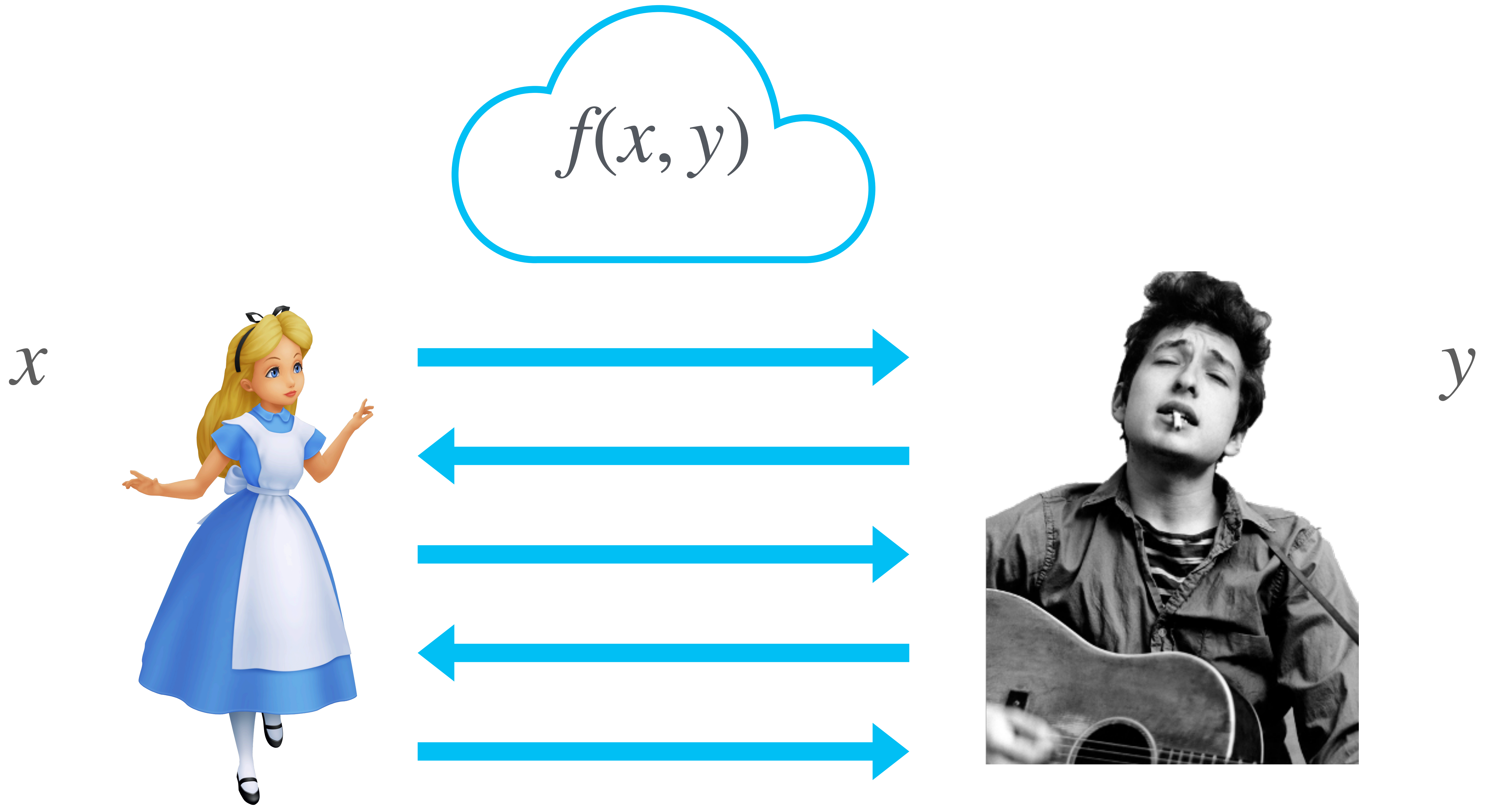
y

SECURE (2-PARTY) COMPUTATION (2PC)



Goal:
jointly compute $f(x, y)$, without revealing anything more about private inputs x and y

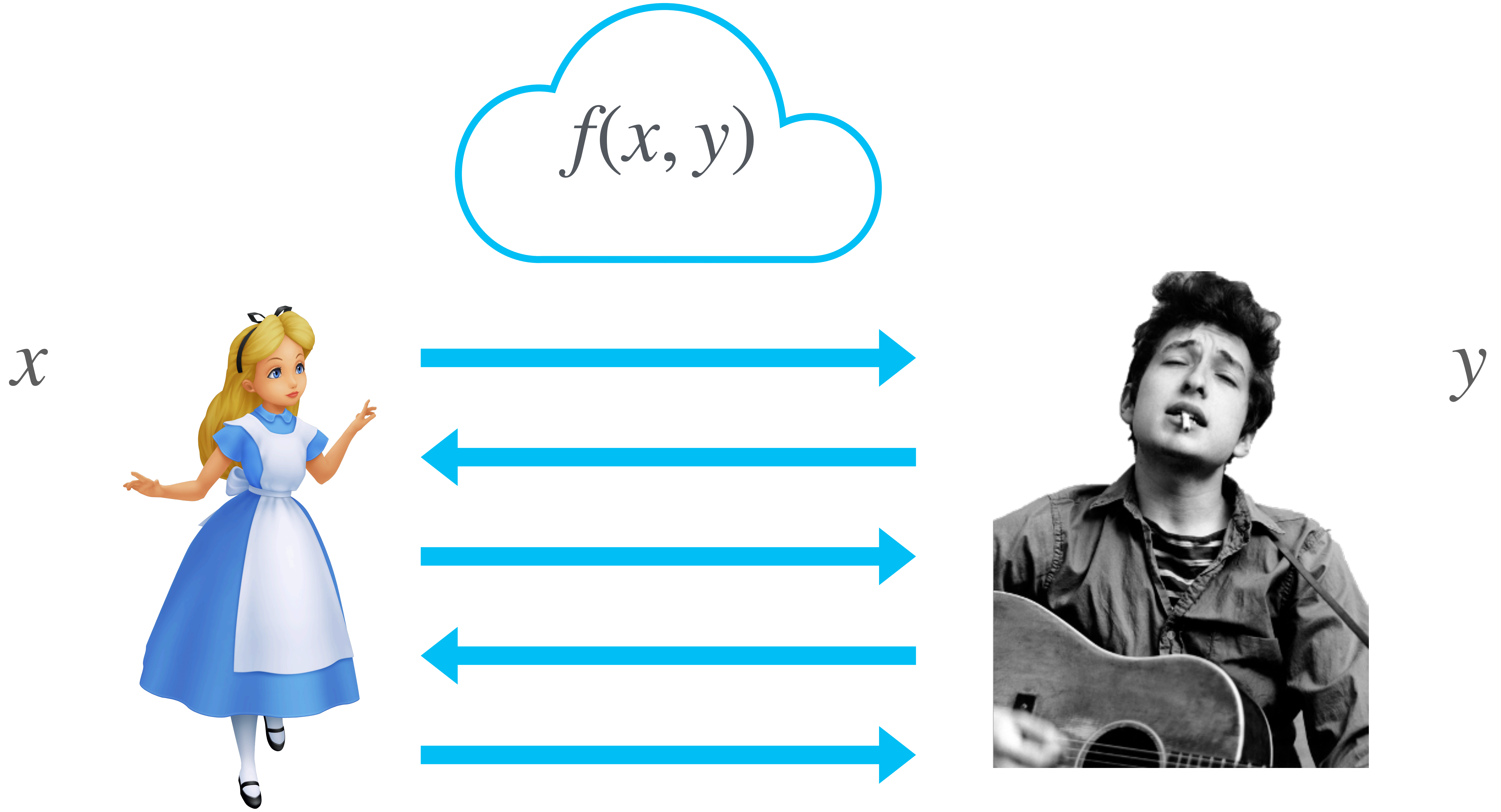
SECURE (2-PARTY) COMPUTATION (2PC)



Goal:
jointly compute $f(x, y)$, without revealing anything more about private inputs x and y

Semi-honest security:
assume parties follow protocol

SECURE (2-PARTY) COMPUTATION (2PC)



Goal:
jointly compute $f(x, y)$, without revealing anything more about private inputs x and y

Semi-honest security:
assume parties follow protocol

Malicious security:
parties may deviate from protocol

HISTORY FOR CONSTANT-OVERHEAD 2PC

HISTORY FOR CONSTANT-OVERHEAD 2PC

	Semi-honest vs. malicious?	Boolean vs. large field?	Computation	Communication

HISTORY FOR CONSTANT-OVERHEAD 2PC

	Semi-honest vs. malicious?	Boolean vs. large field?	Computation	Communication
[IKOS'08]	S	B	$O(N)$	$O(N)$

HISTORY FOR CONSTANT-OVERHEAD 2PC

	Semi-honest vs. malicious?	Boolean vs. large field?	Computation	Communication
[IKOS'08]	S	B	$O(N)$	$O(N)$
[ADINZ'17, BCCGHJ'17]	M	L	$O(N)$	$O(N)$

HISTORY FOR CONSTANT-OVERHEAD 2PC

	Semi-honest vs. malicious?	Boolean vs. large field?	Computation	Communication
[IKOS'08]	S	B	$O(N)$	$O(N)$
[ADINZ'17, BCCGHJ'17]	M	L	$O(N)$	$O(N)$
[DIK'10, dCHIVV'21]	M	B	$O(N \operatorname{polylog} N)$	$O(N)$

HISTORY FOR CONSTANT-OVERHEAD 2PC

	Semi-honest vs. malicious?	Boolean vs. large field?	Computation	Communication
[IKOS'08]	S	B	$O(N)$	$O(N)$
[ADINZ'17, BCCGHJ'17]	M	L	$O(N)$	$O(N)$
[DIK'10, dCHIVV'21]	M	B	$O(N \text{ polylog} N)$	$O(N)$
[BCGIKS'19A, BCGIKS'19B, YWLZW'20, CRR'21, CGIKRS'22]	M	B	$N^{1+\Omega(1)}$	$o(N)$

HISTORY FOR CONSTANT-OVERHEAD 2PC

	Semi-honest vs. malicious?	Boolean vs. large field?	Computation	Communication
[IKOS'08]	S	B	$O(N)$	$O(N)$
[ADINZ'17, BCCGHJ'17]	M	L	$O(N)$	$O(N)$
[DIK'10, dCHIVV'21]		B	$O(N \text{ polylog} N)$	$O(N)$
[BCGIKS'19A, BCGIKS'19B, YWLZW'20, CRR'21, CGIKRS'22]	M	B	$N^{1+\Omega(1)}$	$o(N)$

pseudorandom correlation generators

TODAY: GENERATING N BIT-OBLIVIOUS TRANSFERS

TODAY: GENERATING N BIT-OBLIVIOUS TRANSFERS

Complete for
semi-honest 2PC

TODAY: GENERATING N BIT-OBLIVIOUS TRANSFERS

Complete for semi-honest 2PC

Partially extends to malicious setting

TODAY: GENERATING N BIT-OBLIVIOUS TRANSFERS

Complete for semi-honest 2PC

Partially extends to malicious setting

- relaxed security guarantees

TODAY: GENERATING N BIT-OBLIVIOUS TRANSFERS

Complete for semi-honest 2PC

Partially extends to malicious setting

- relaxed security guarantees
- results for "finite" functionalities

TODAY: GENERATING N BIT-OBLIVIOUS TRANSFERS

Complete for semi-honest 2PC

Partially extends to malicious setting

- relaxed security guarantees
- results for "finite" functionalities
- reductions for open questions

TODAY: GENERATING N BIT-OBLIVIOUS TRANSFERS

Complete for semi-honest 2PC

Good benchmark for techniques

Partially extends to malicious setting

- relaxed security guarantees
- results for "finite" functionalities
- reductions for open questions

TODAY: GENERATING N BIT-OBLIVIOUS TRANSFERS

Complete for semi-honest 2PC

Good benchmark for techniques

Partially extends to malicious setting

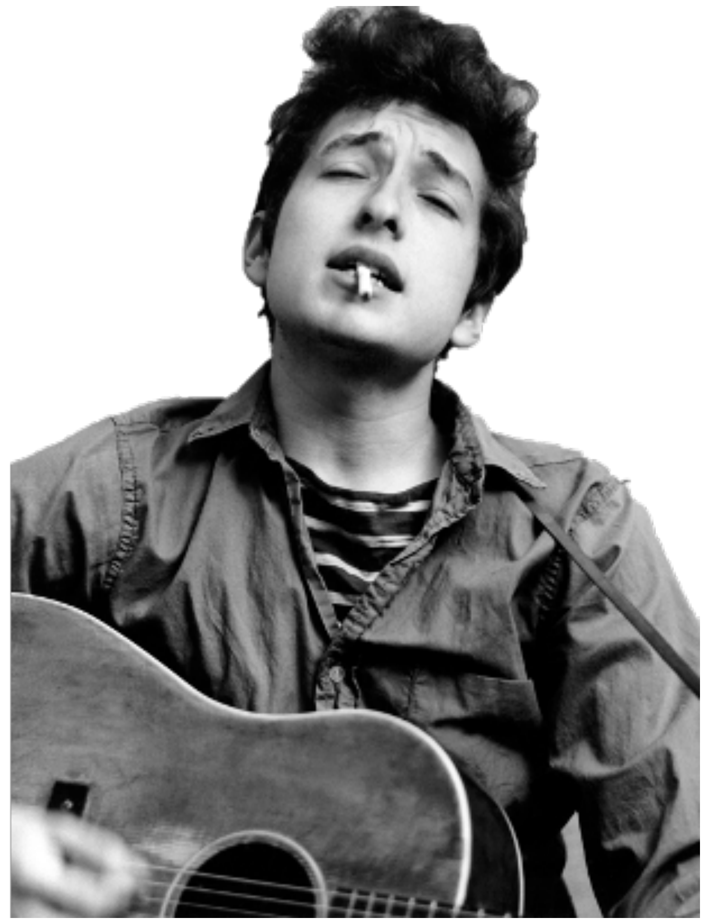
- relaxed security guarantees
- results for "finite" functionalities
- reductions for open questions

- Many past research efforts (often called "batch-OT/OT-extension") [ACPS'09, IKOPSW'11, BCGIKS'19, OSY'21, BBDP'22] minimizing computation/communication costs

BIT OBLIVIOUS TRANSFER

b

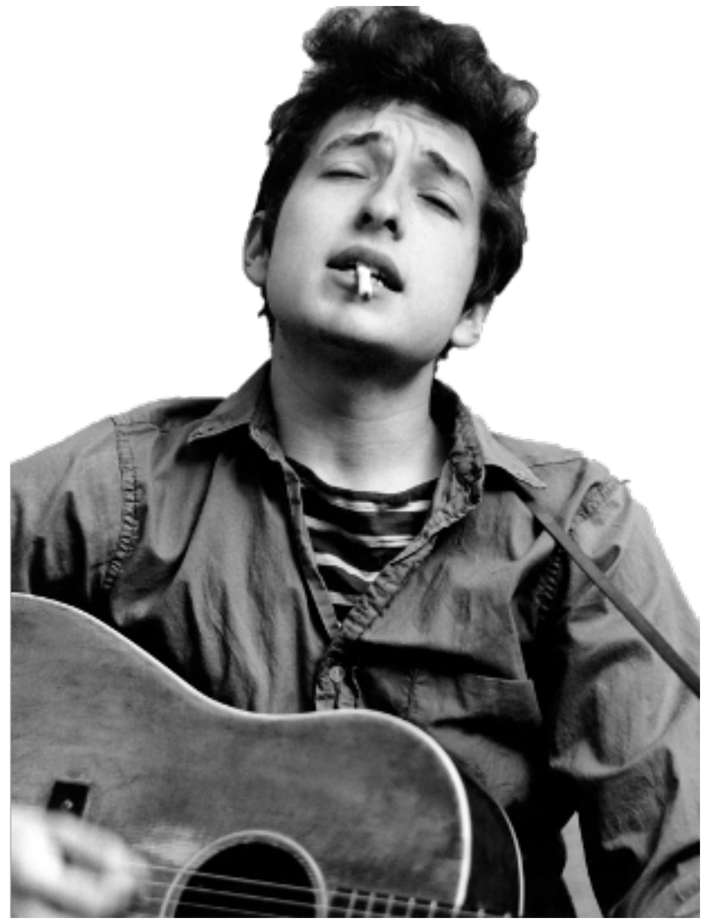
(m_0, m_1)



BIT OBLIVIOUS TRANSFER

b

(m_0, m_1)



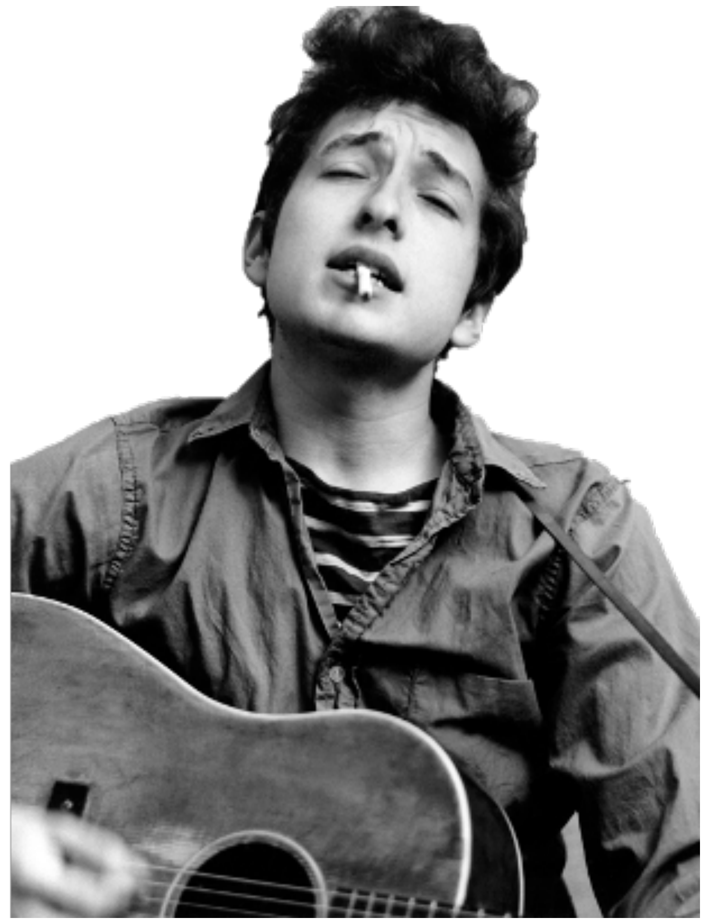
- b is choice bit

BIT OBLIVIOUS TRANSFER

b

OT

(m_0, m_1)



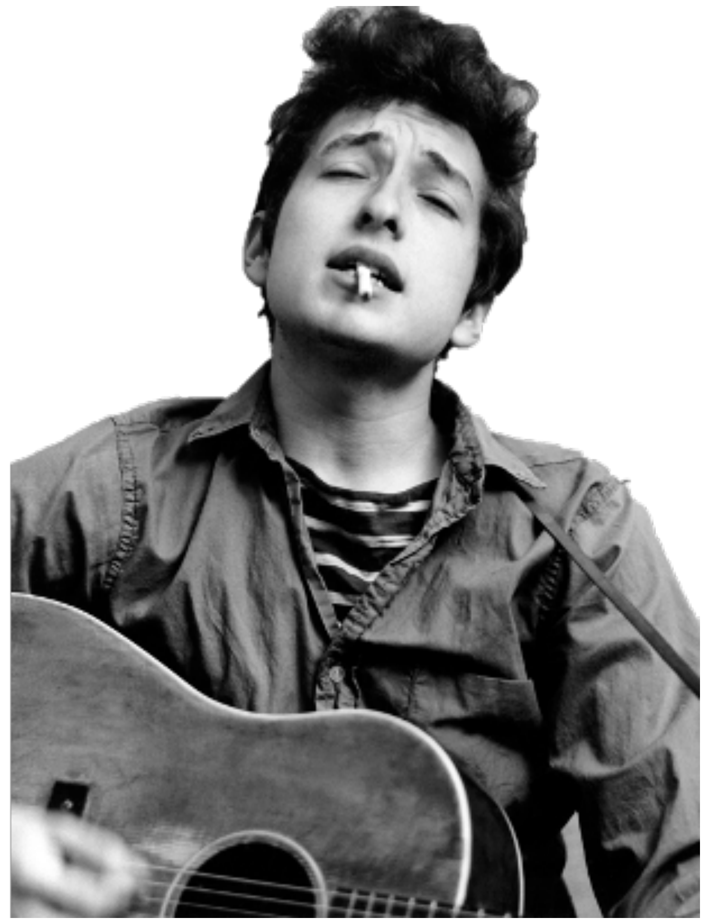
- b is choice bit

BIT OBLIVIOUS TRANSFER

b

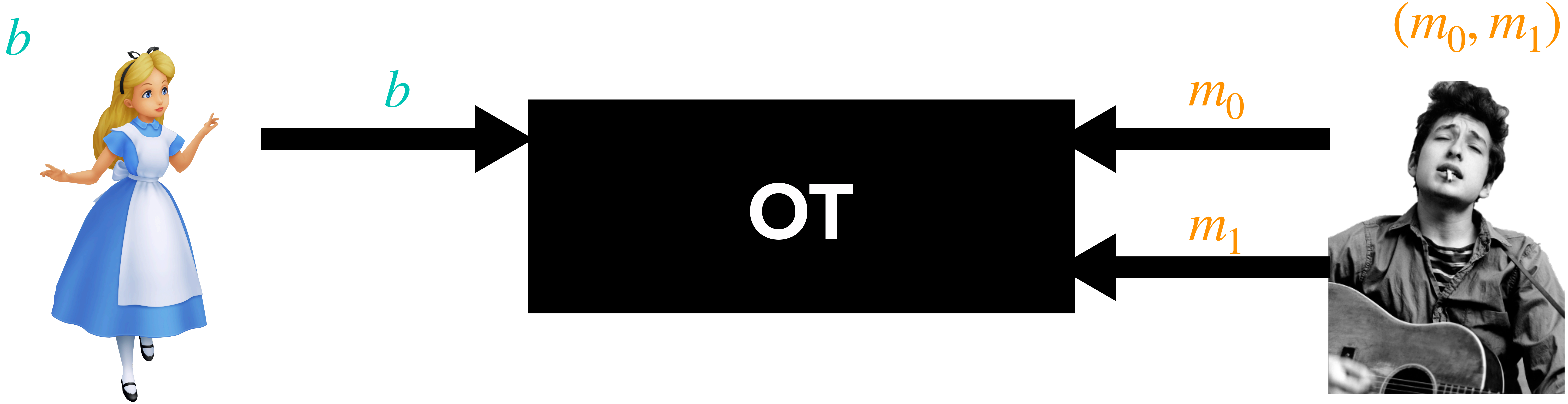
$\xrightarrow{b}$ **OT**

(m_0, m_1)



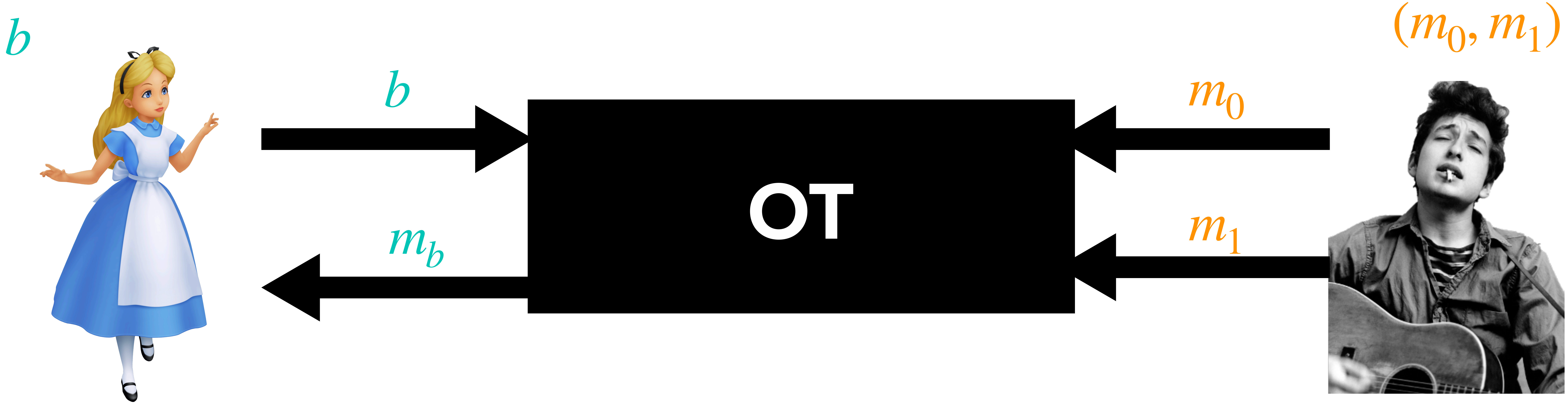
- b is choice bit

BIT OBLIVIOUS TRANSFER



- b is choice bit

BIT OBLIVIOUS TRANSFER



- b is choice bit

BIT OBLIVIOUS TRANSFER

- b is choice bit
- Alice learns one (and only one!) of Bob's messages

BIT OBLIVIOUS TRANSFER

- b is choice bit
- Alice learns one (and only one!) of Bob's messages
- Bob doesn't learn which message Alice received

(RANDOM) OBLIVIOUS TRANSFER

(RANDOM) OBLIVIOUS TRANSFER

(b, m_b)

(m_0, m_1)

b, m_0 and m_1 are independent uniformly random bits

OUR CONTRIBUTION

OUR CONTRIBUTION

Assume:

OUR CONTRIBUTION

Assume:

There exists a standard OT protocol (necessary)

OUR CONTRIBUTION

Assume:	
There exists a standard OT protocol (necessary)	Learning Parity with Noise (LPN) for a sparse matrix is hard

OUR CONTRIBUTION

Assume:		
There exists a standard OT protocol (necessary)	Learning Parity with Noise (LPN) for a sparse matrix is hard	There exists a correlation-robust local PRG

OUR CONTRIBUTION

Assume:		
There exists a standard OT protocol (necessary)	Learning Parity with Noise (LPN) for a sparse matrix is hard	There exists a correlation-robust local PRG

Then there exists:

OUR CONTRIBUTION

Assume:		
There exists a standard OT protocol (necessary)	Learning Parity with Noise (LPN) for a sparse matrix is hard	There exists a correlation-robust local PRG

Then there exists:
2-party protocol with malicious security realizing N instances of bit-OT with

OUR CONTRIBUTION

Assume:

There exists a standard OT protocol (necessary)	Learning Parity with Noise (LPN) for a sparse matrix is hard	There exists a correlation-robust local PRG

Then there exists:

2-party protocol with malicious security realizing N instances of bit-OT with

Computation costs:

$O(N) + o(N) \cdot \text{poly}(\lambda)$

OUR CONTRIBUTION

Assume:

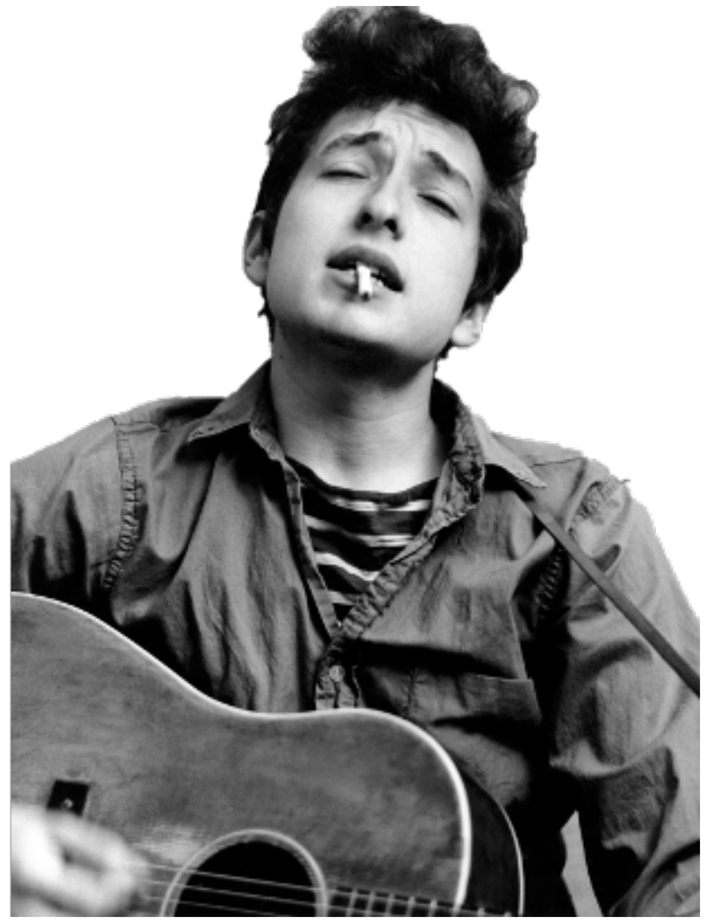
There exists a standard OT protocol (necessary)	Learning Parity with Noise (LPN) for a sparse matrix is hard	There exists a correlation-robust local PRG

Then there exists:

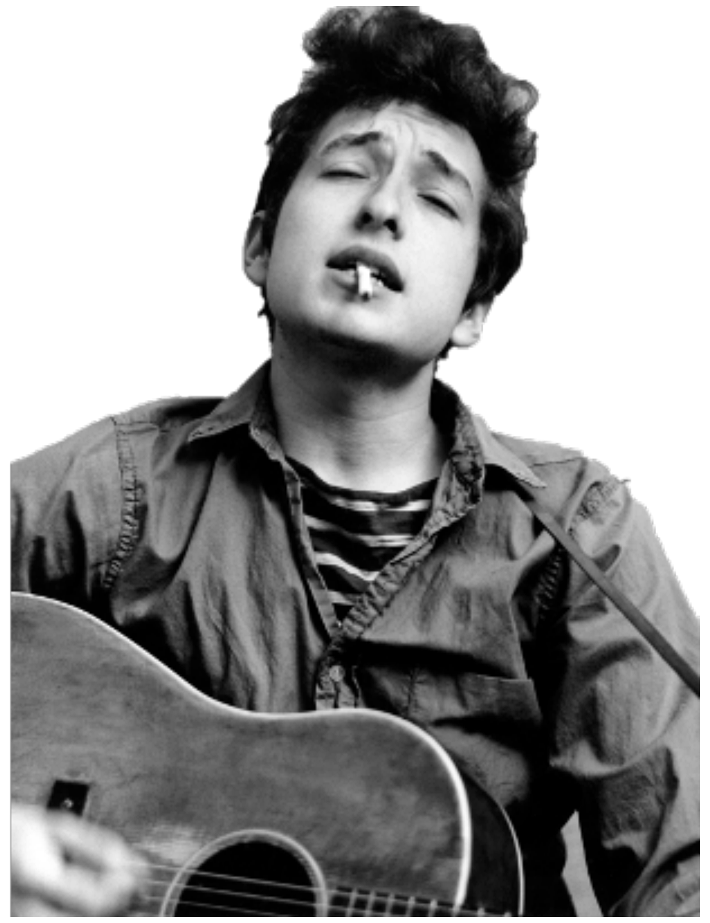
2-party protocol with malicious security realizing N instances of bit-OT with

Computation costs:	Communication costs:
$O(N) + o(N) \cdot \mathrm{poly}(\lambda)$	$o(N) \cdot \mathrm{poly}(\lambda)$

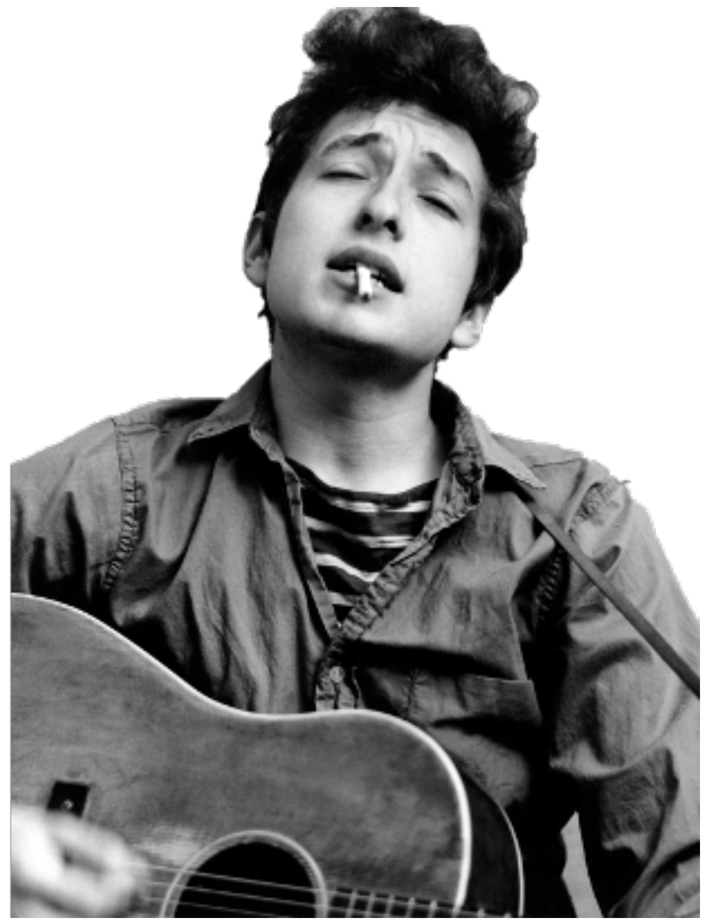
PSEUDORANDOM CORRELATION GENERATOR (PCG)



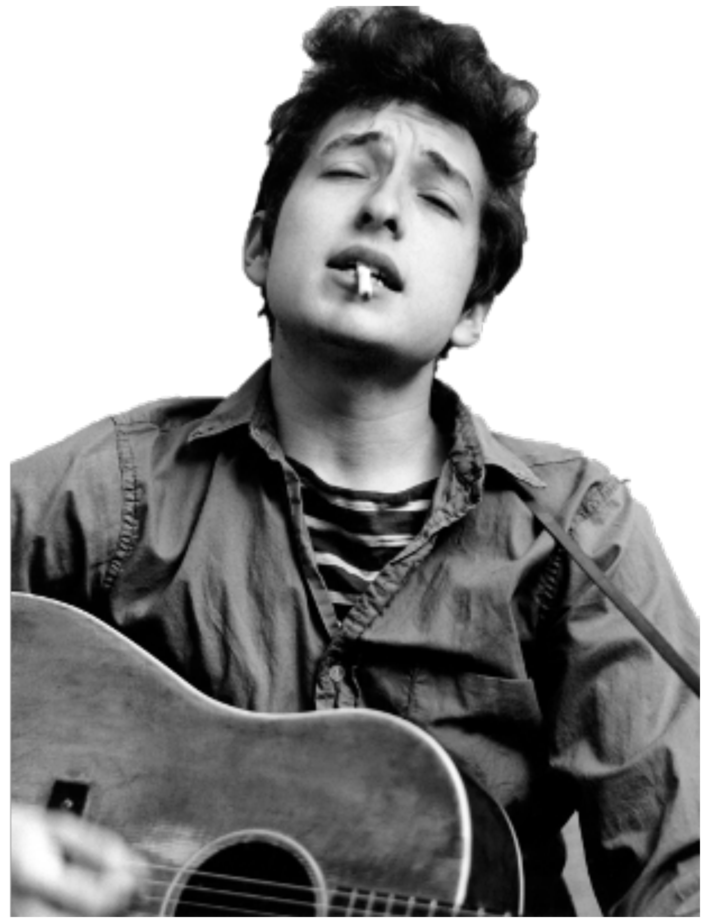
PSEUDORANDOM CORRELATION GENERATOR (PCG)

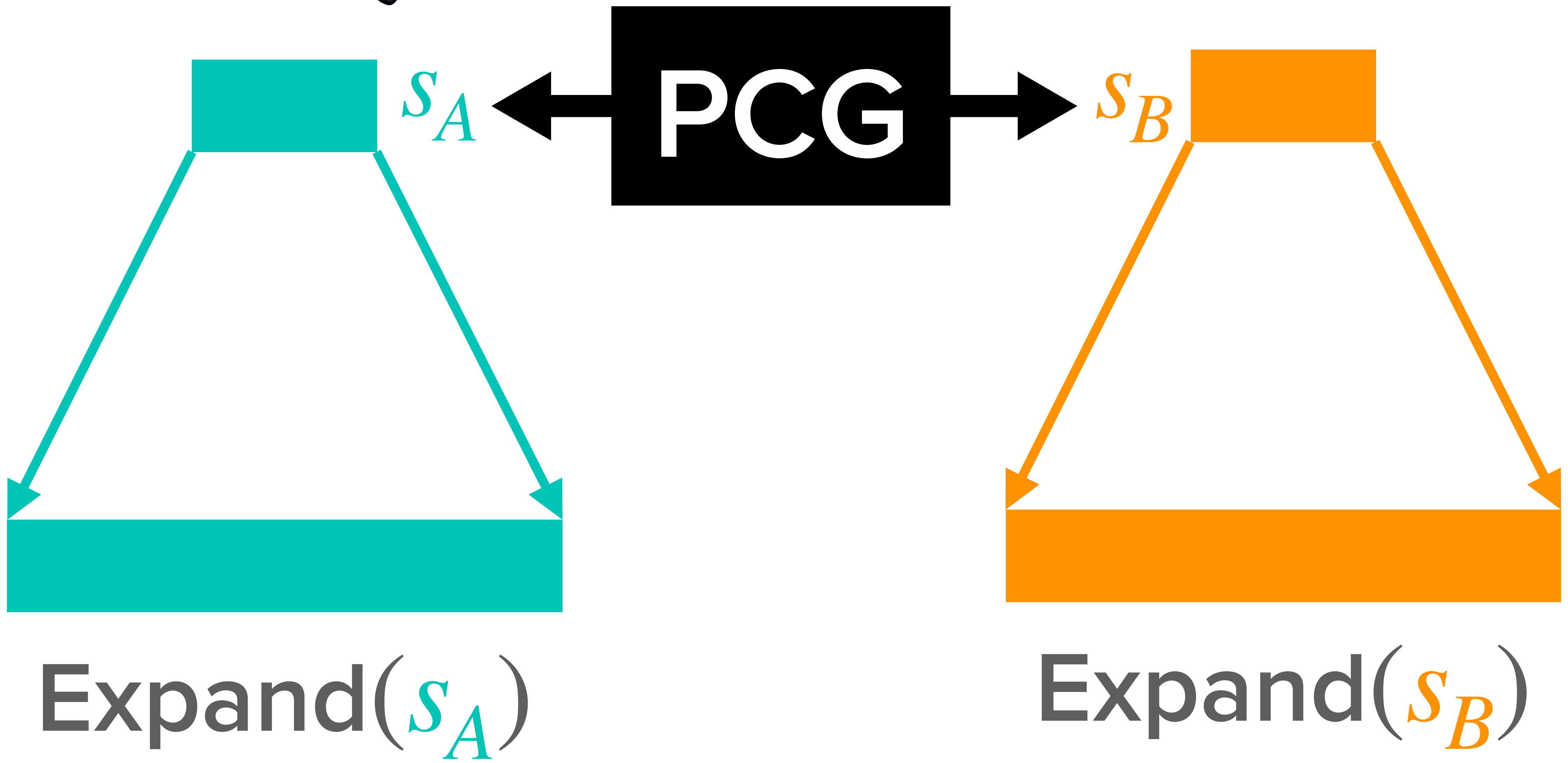


PSEUDORANDOM CORRELATION GENERATOR (PCG)

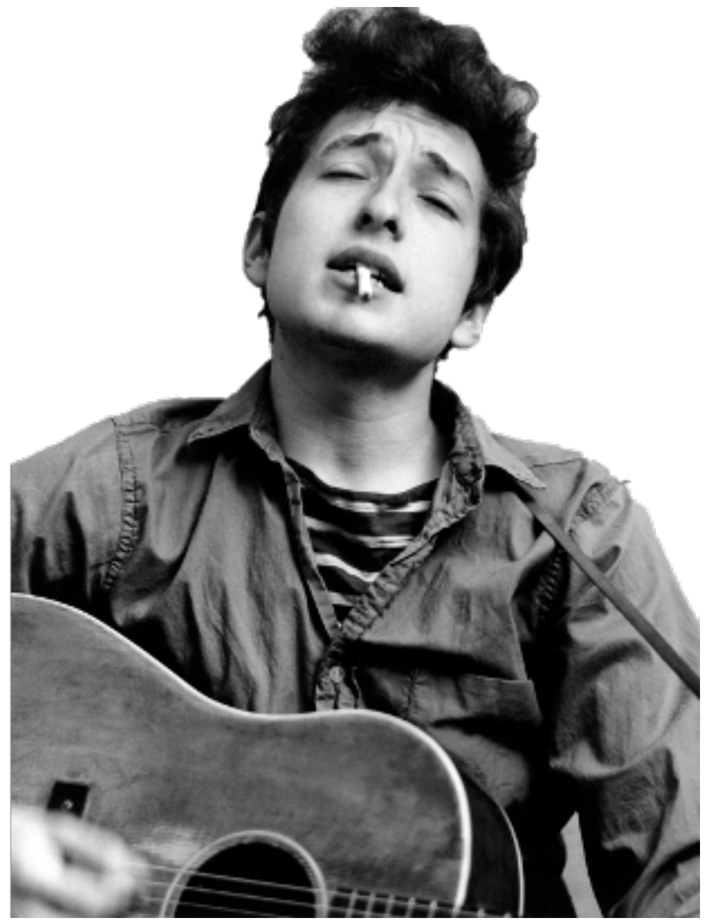


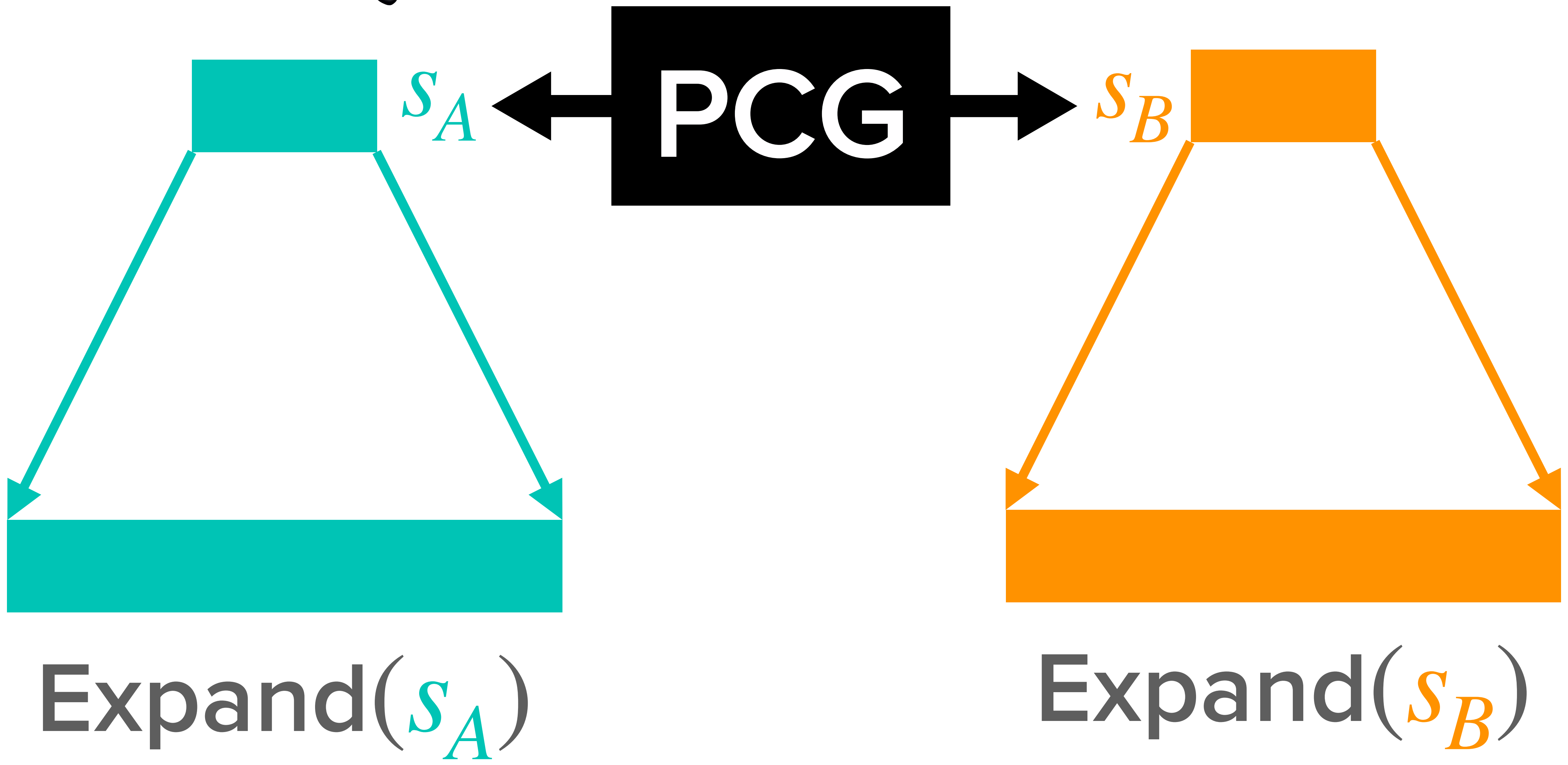
PSEUDORANDOM CORRELATION GENERATOR (PCG)





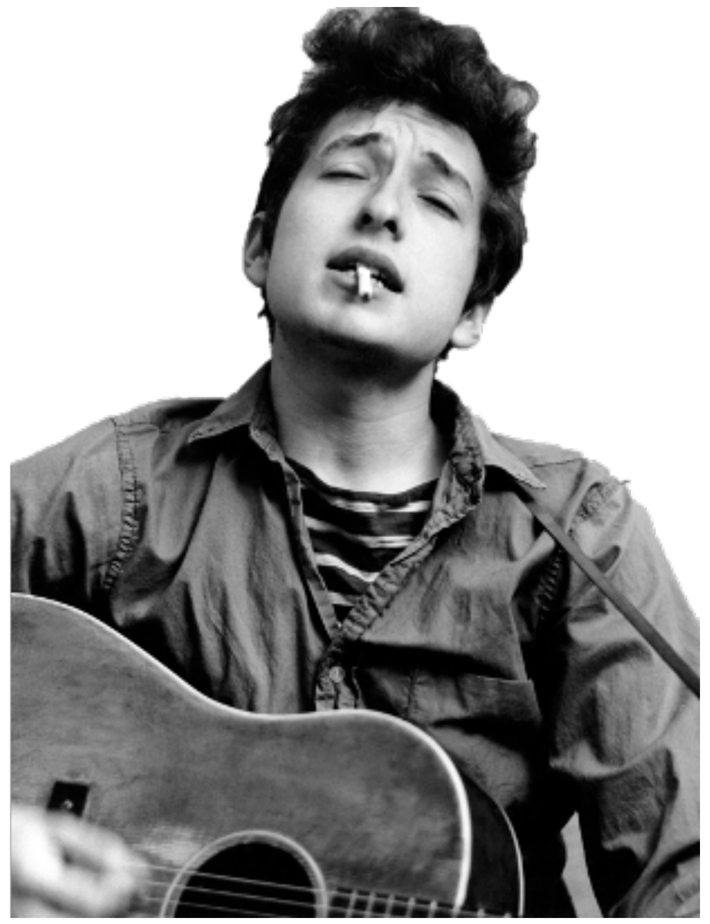
PSEUDORANDOM CORRELATION GENERATOR (PCG)



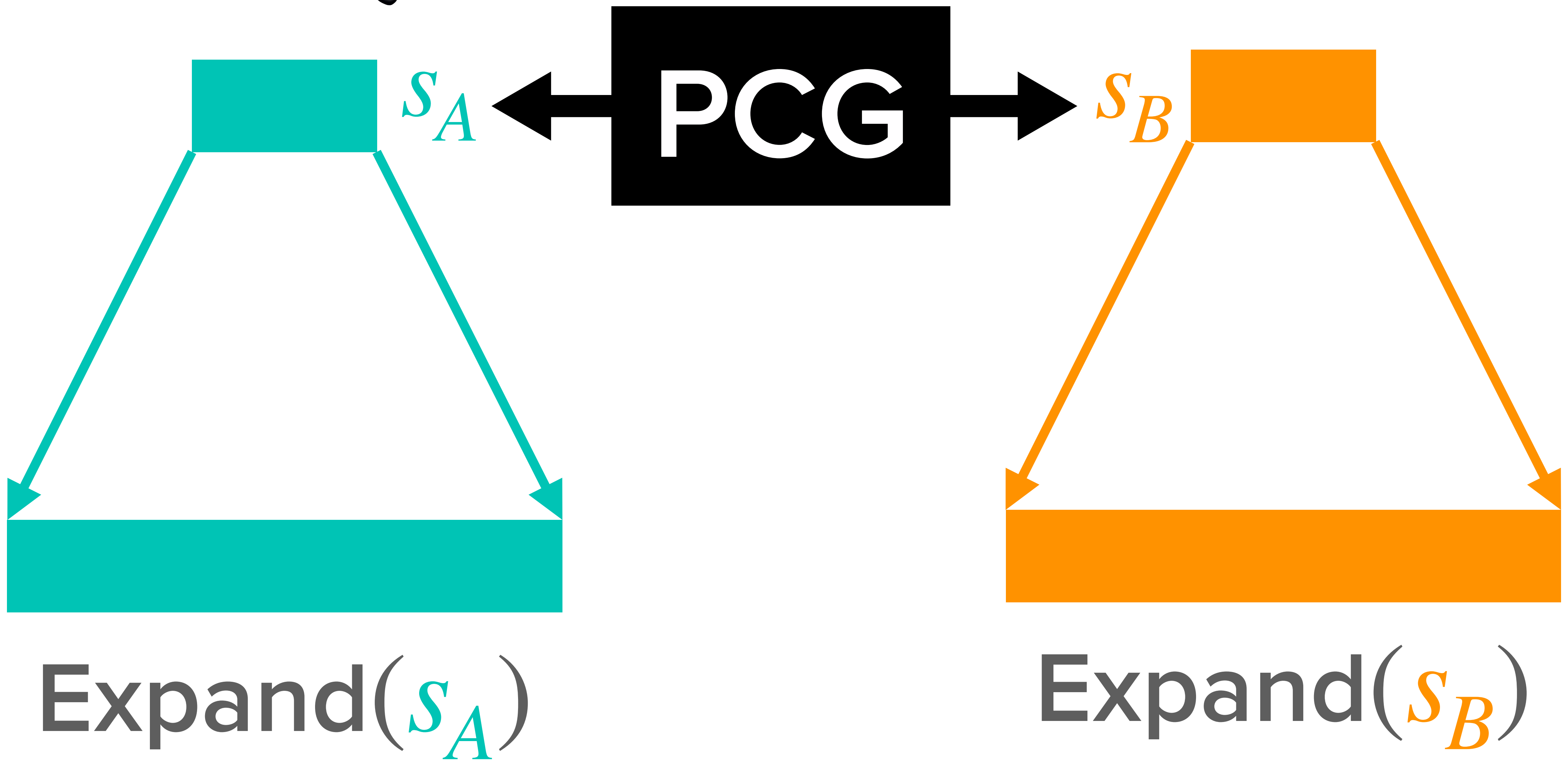


- Pseudorandomness:
  - Expand(s_A), Expand(s_B) pseudorand.

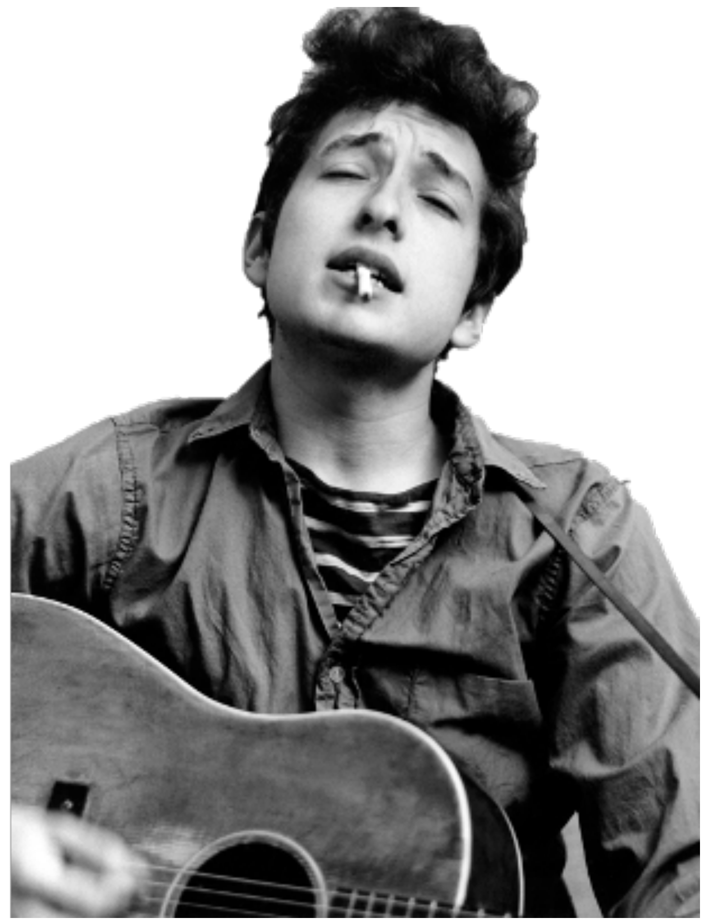
PSEUDORANDOM CORRELATION GENERATOR (PCG)

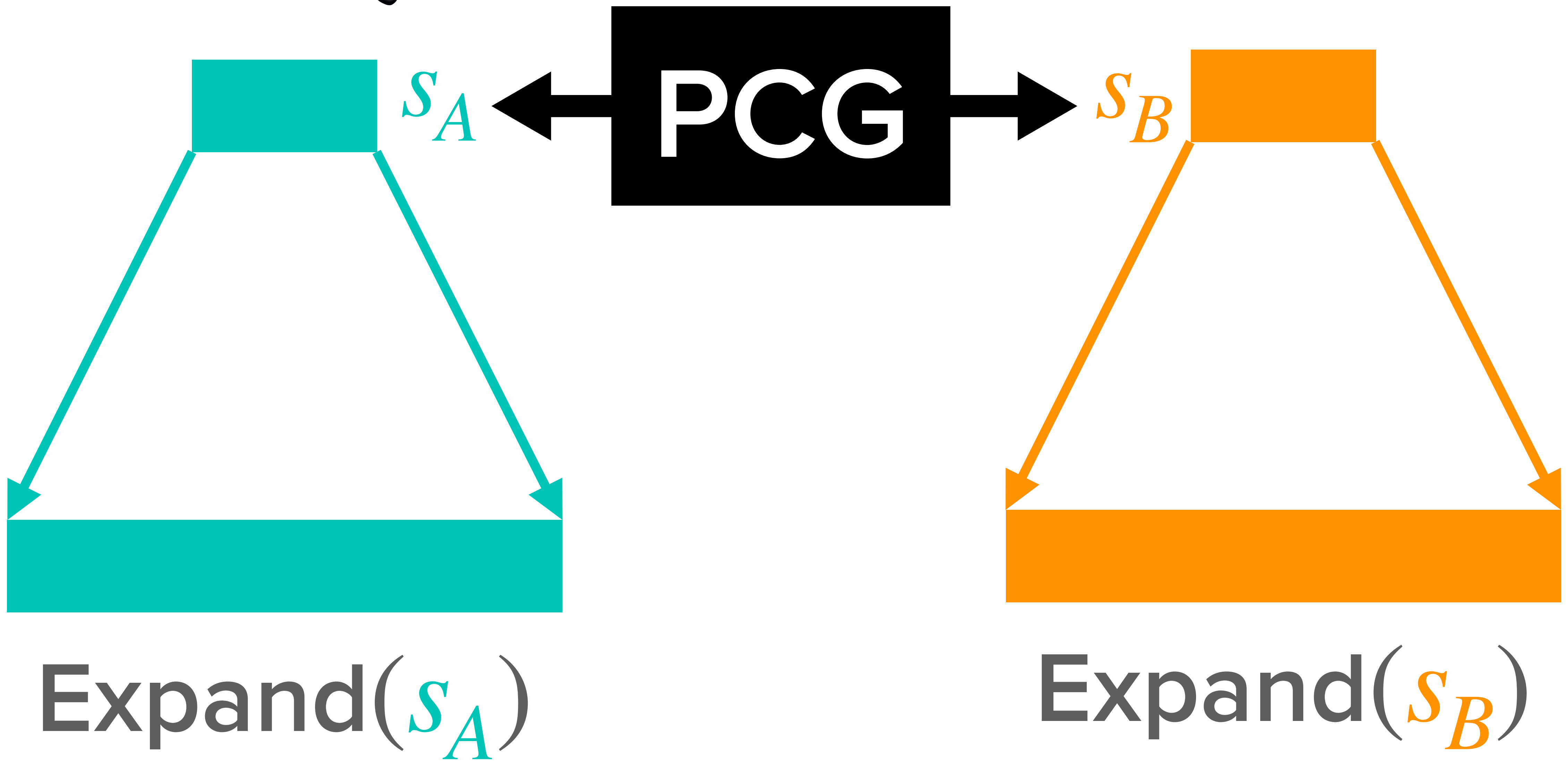


- Pseudorandomness:
  - Expand(s_A), Expand(s_B) pseudorand.
- Correctness:
  - (Expand(s_A), Expand(s_B)) $\in C^N$



PSEUDORANDOM CORRELATION GENERATOR (PCG)

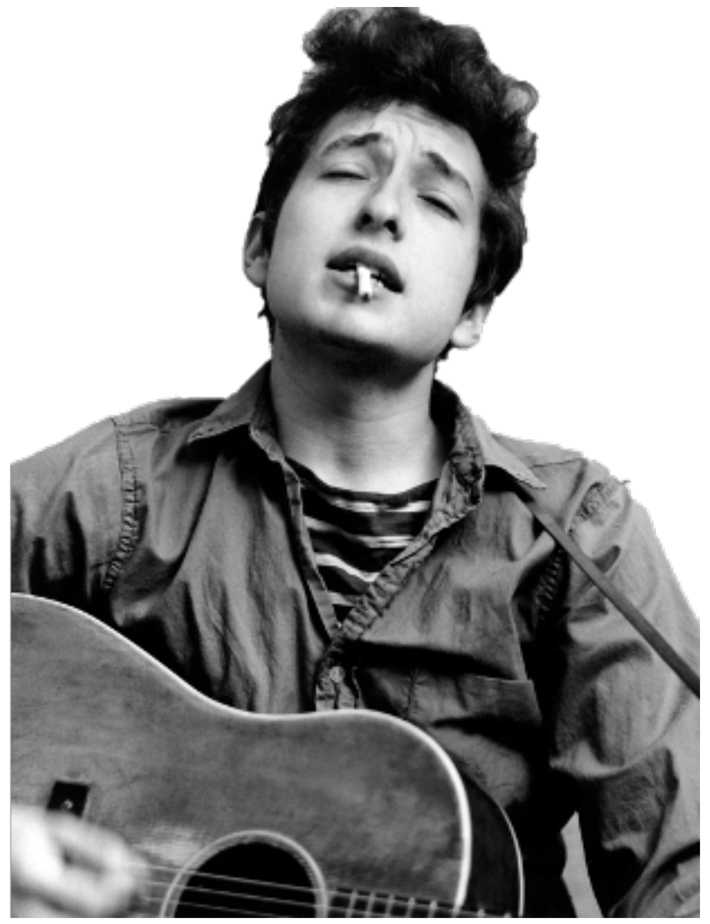


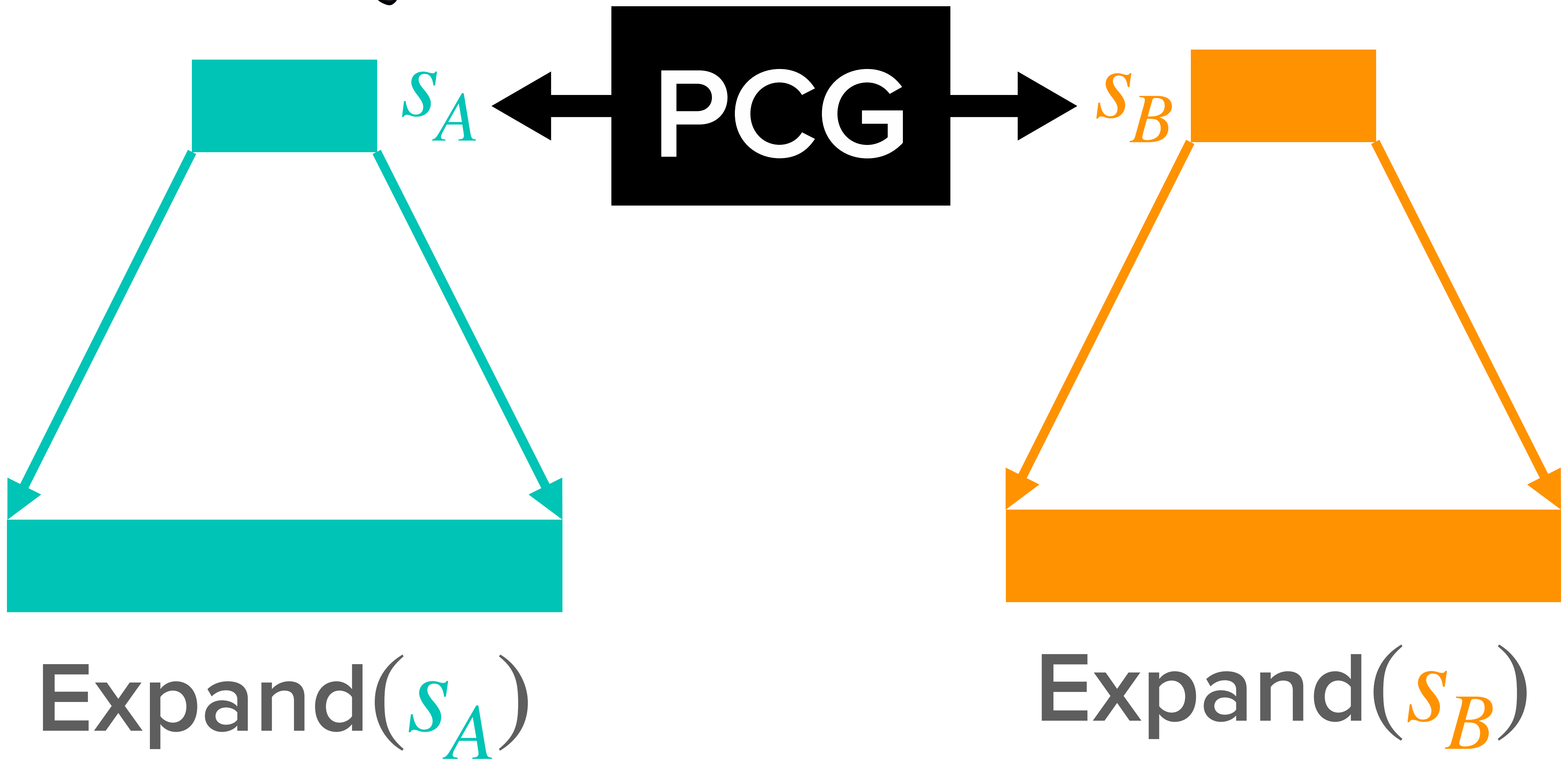


- Pseudorandomness:
  - Expand(s_A), Expand(s_B) pseudorand.
- Correctness:
  - (Expand(s_A), Expand(s_B)) $\in C^N$

N indep. OT's

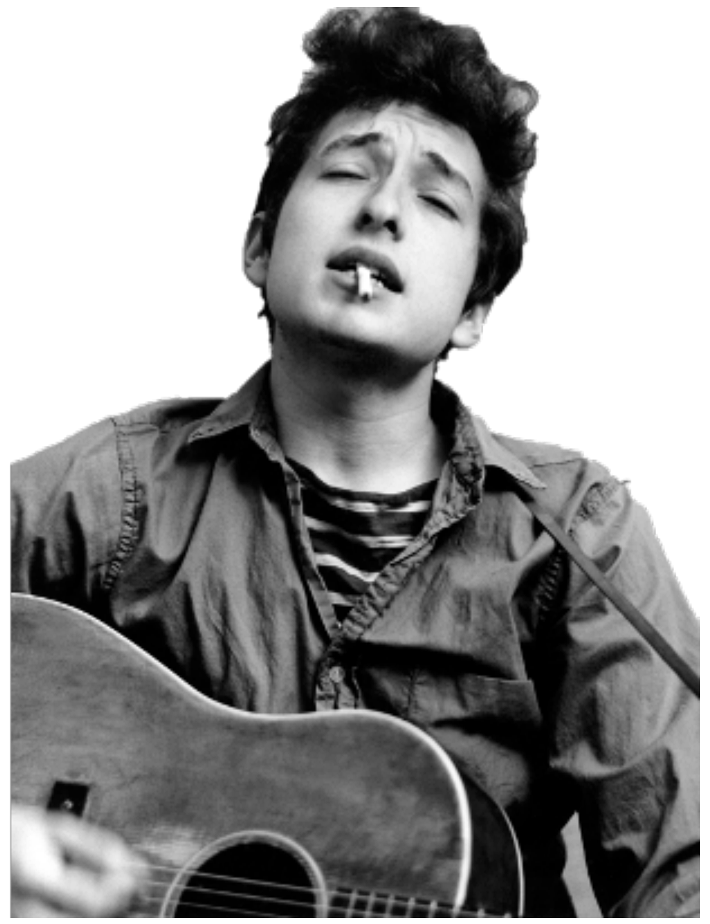
PSEUDORANDOM CORRELATION GENERATOR (PCG)

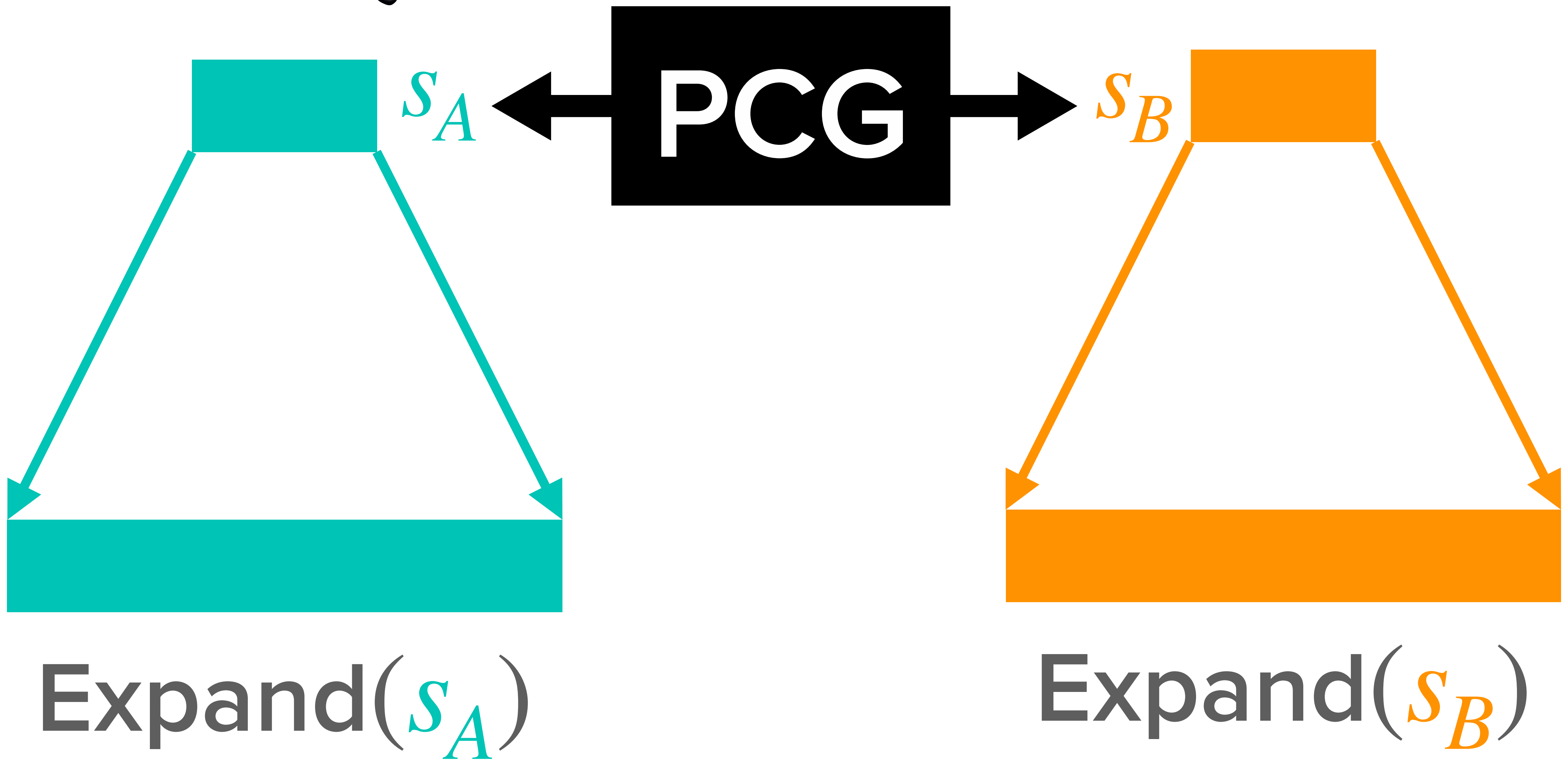




- Pseudorandomness:
  - Expand(s_A), Expand(s_B) pseudorand.
- Correctness: N indep. OT's
  - (Expand(s_A), Expand(s_B)) $\in C^N$
- Security:
  - Other party's output looks pseudorandom up to correlation

PSEUDORANDOM CORRELATION GENERATOR (PCG)





- **Pseudorandomness:**
  - Expand(s_A), Expand(s_B) pseudorand.
- **Correctness:** N indep. OT's
  - (Expand(s_A), Expand(s_B)) $\in C^N$
- **Security:**
  - Other party's output looks pseudorandom up to correlation

Can replace ⟵ PCG ⟶ with efficient maliciously-secure 2PC protocol [IPS'08]

MAIN RESULT

MAIN RESULT

Assume:

MAIN RESULT

Assume:

Learning Parity with Noise (LPN) for a sparse matrix is hard

MAIN RESULT

Assume:	
Learning Parity with Noise (LPN) for a sparse matrix is hard	There exists a correlation-robust local PRG

MAIN RESULT

Assume:	
Learning Parity with Noise (LPN) for a sparse matrix is hard	There exists a correlation-robust local PRG

Then there exists:

MAIN RESULT

Assume:

Learning Parity with Noise (LPN) for a sparse matrix is hard	There exists a correlation-robust local PRG

Then there exists:

PCG realizing N instances of bit-OT with

MAIN RESULT

Assume:

Learning Parity with Noise (LPN) for a sparse matrix is hard	There exists a correlation-robust local PRG

Then there exists:

PCG realizing N instances of bit-OT with

Expansion phase computation costs:
$O(N) + o(N) \cdot \text{poly}(\lambda)$

MAIN RESULT

Assume:	
Learning Parity with Noise (LPN) for a sparse matrix is hard	There exists a correlation-robust local PRG

Then there exists:	
PCG realizing N instances of bit-OT with	
Expansion phase computation costs: $O(N) + o(N) \cdot \text{poly}(\lambda)$	Seed size: $o(N) \cdot \text{poly}(\lambda)$

INGREDIENTS

PCG for "non-independent OT-like" correlation C

$+$

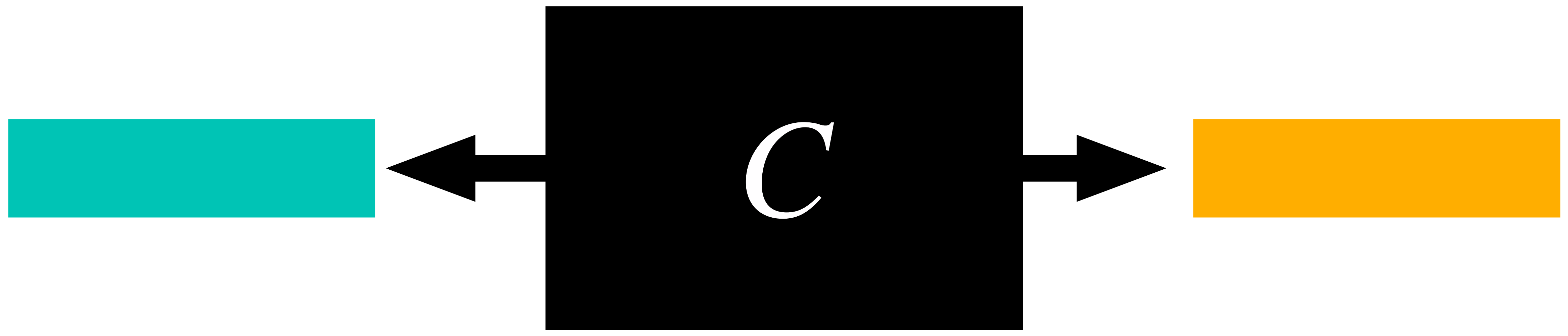
Break correlations with local PRG

INGREDIENTS

PCG for "non-independent OT-like" correlation C

+

Break correlations with local PRG

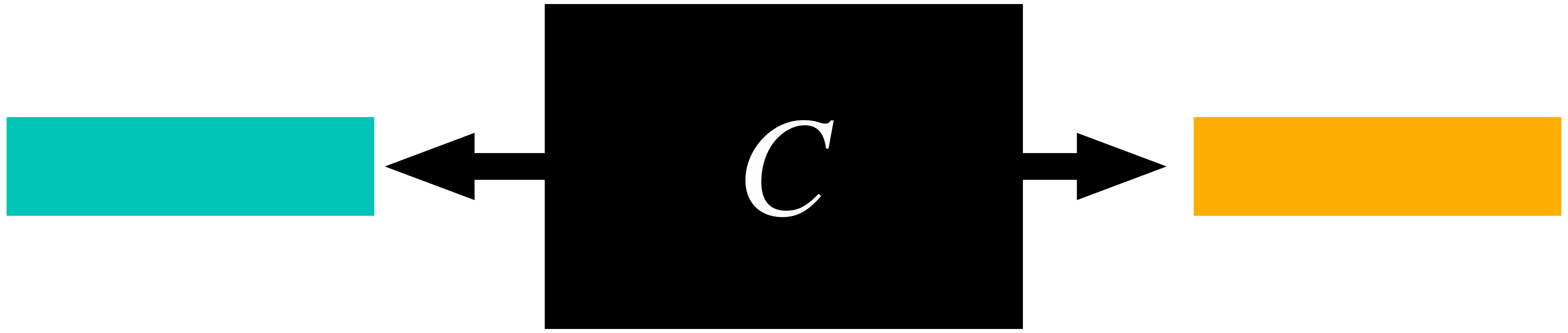


INGREDIENTS

PCG for "non-independent OT-like" correlation C

+

Break correlations with local PRG



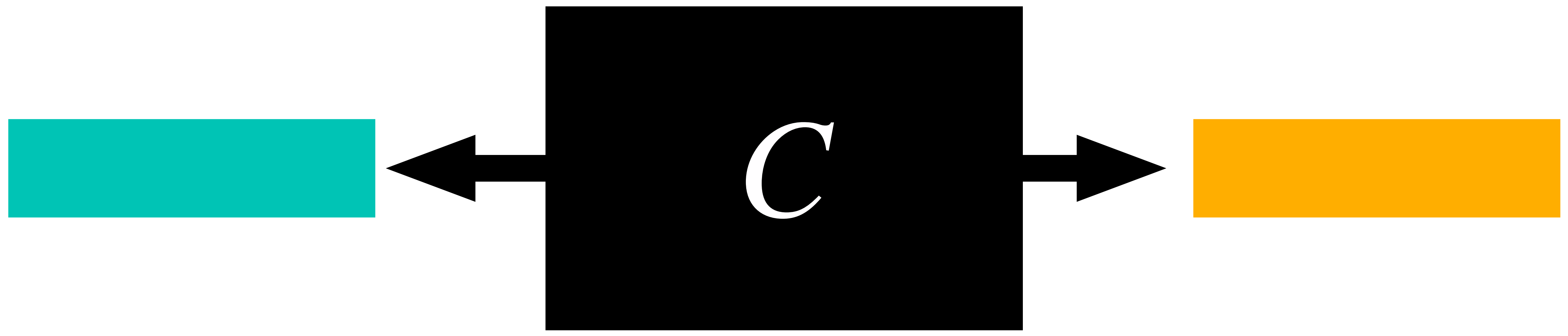
Pushes techniques of [BCGI'18]

INGREDIENTS

PCG for "non-independent OT-like" correlation C

$+$

Break correlations with local PRG



Pushes techniques of [BCGI'18]

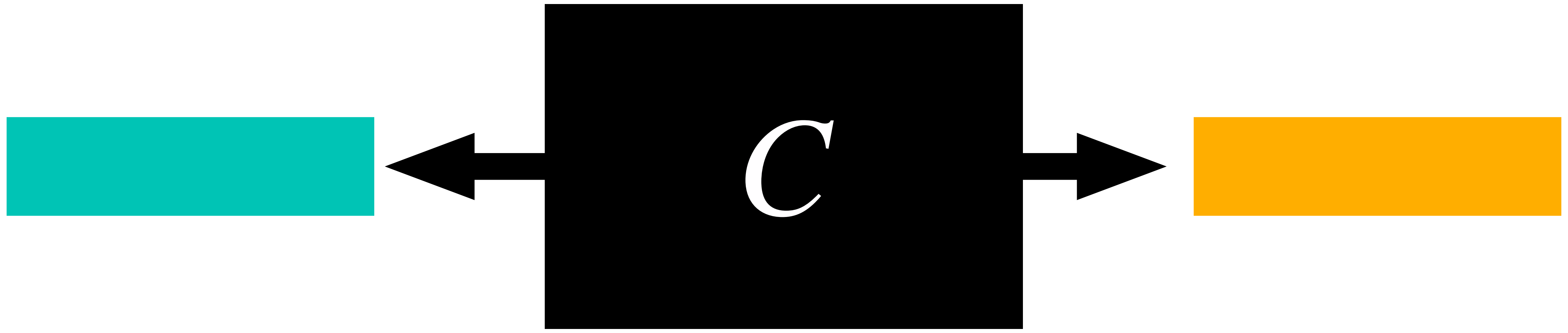
PRG from sparse-LPN

INGREDIENTS

PCG for "non-independent OT-like" correlation C

+

Break correlations with local PRG



Pushes techniques of [BCGI'18]

PRG from sparse-LPN

+

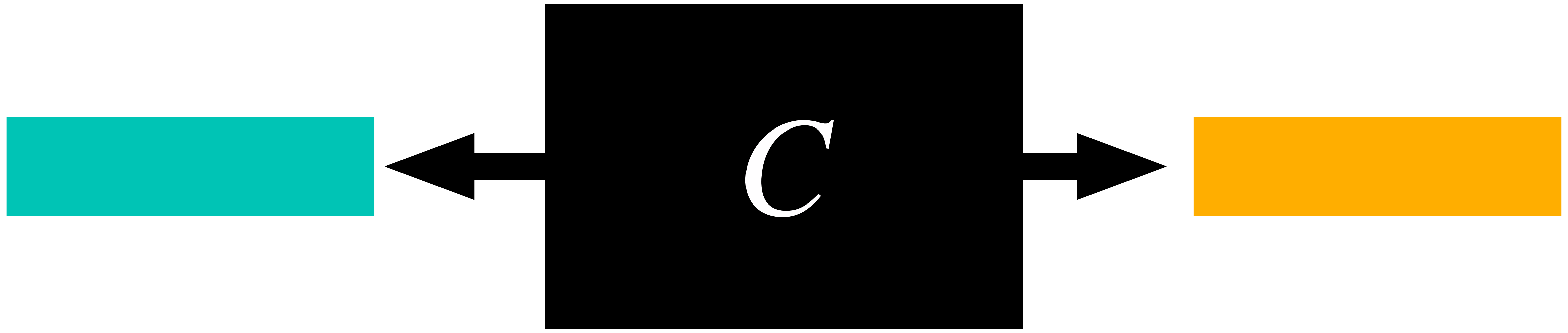
succinct additive sharings of "structured" vectors

INGREDIENTS

PCG for "non-independent OT-like" correlation C

+

Break correlations with local PRG



We'll focus on this step

Pushes techniques of [BCGI'18]

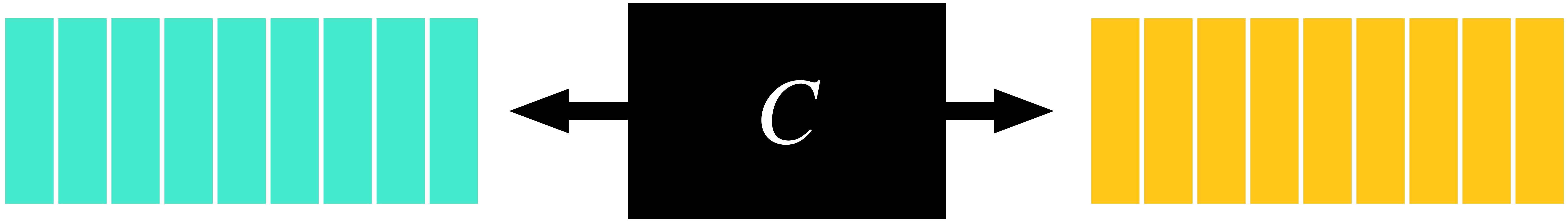
PRG from sparse-LPN

+

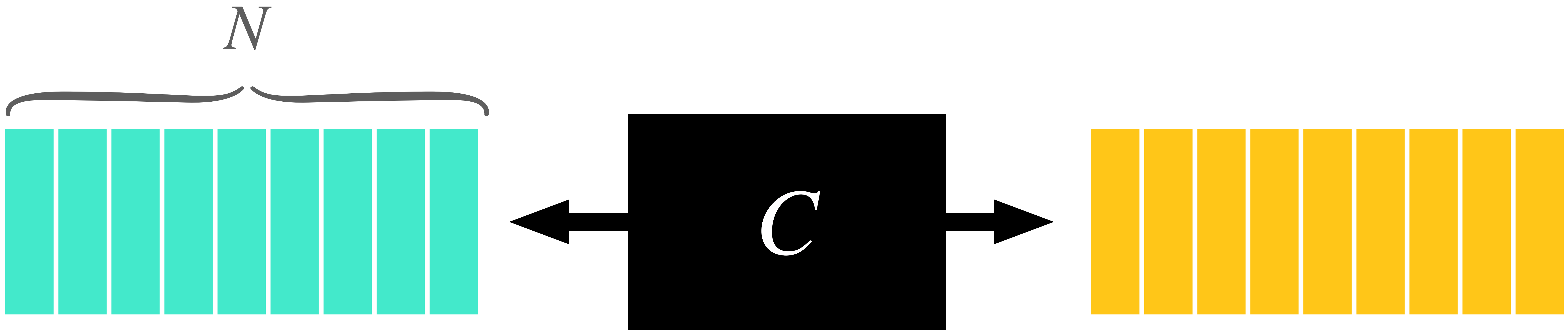
succinct additive sharings of "structured" vectors

Inspired by [IKOS'08]

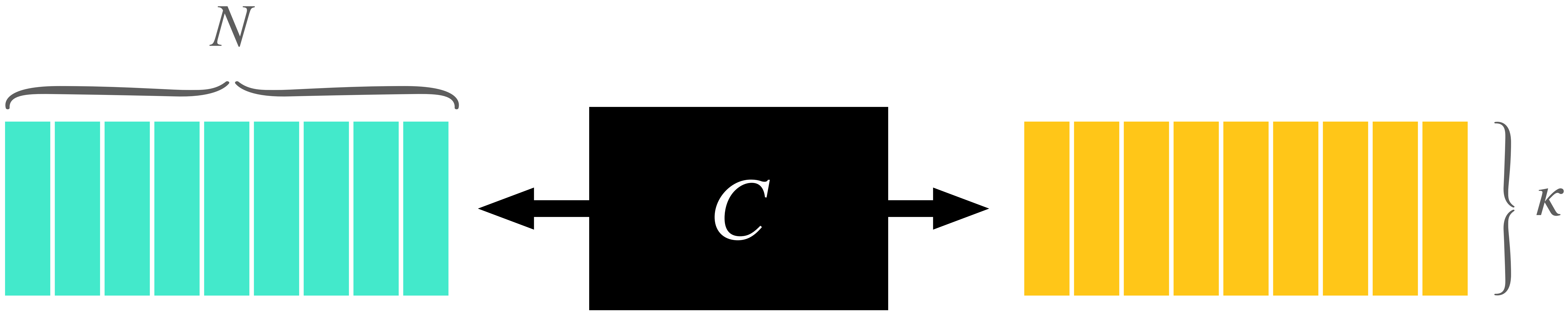
BREAKING CORRELATIONS



BREAKING CORRELATIONS

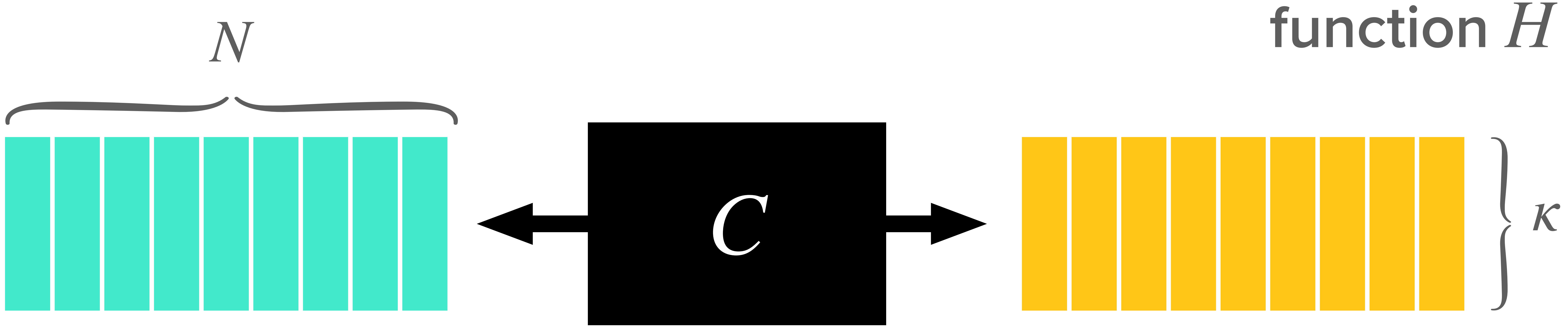


BREAKING CORRELATIONS



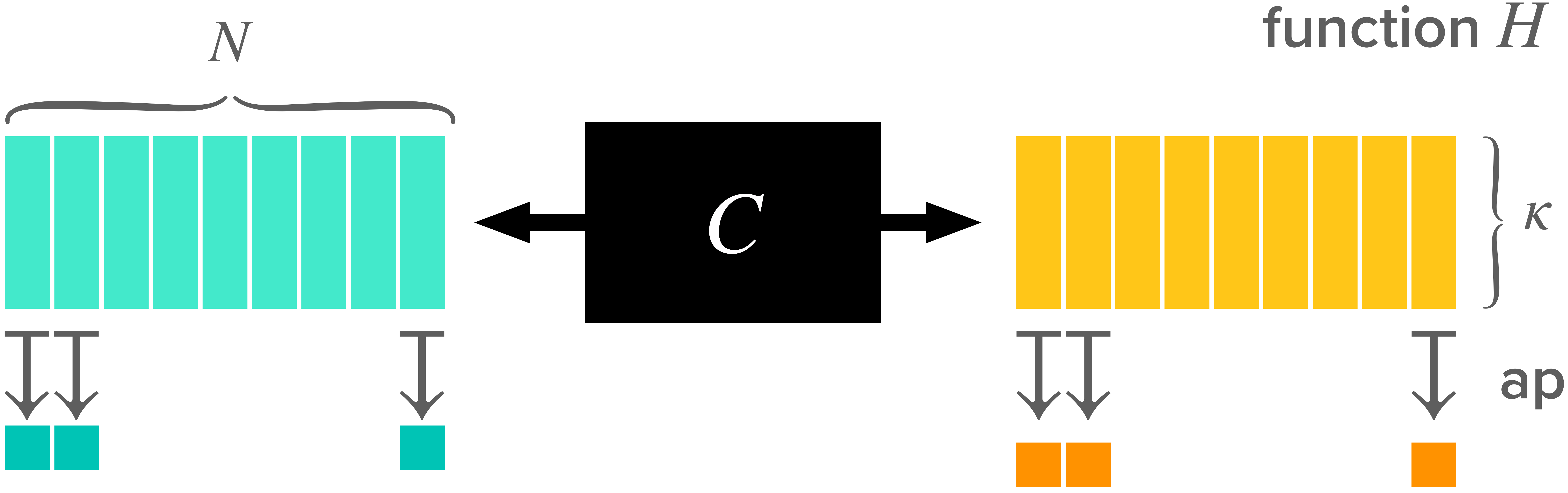
BREAKING CORRELATIONS

[IKNP'03]: Break correlations w/ correlation-robust hash function $H : \{0,1\}^\kappa \to \{0,1\}$



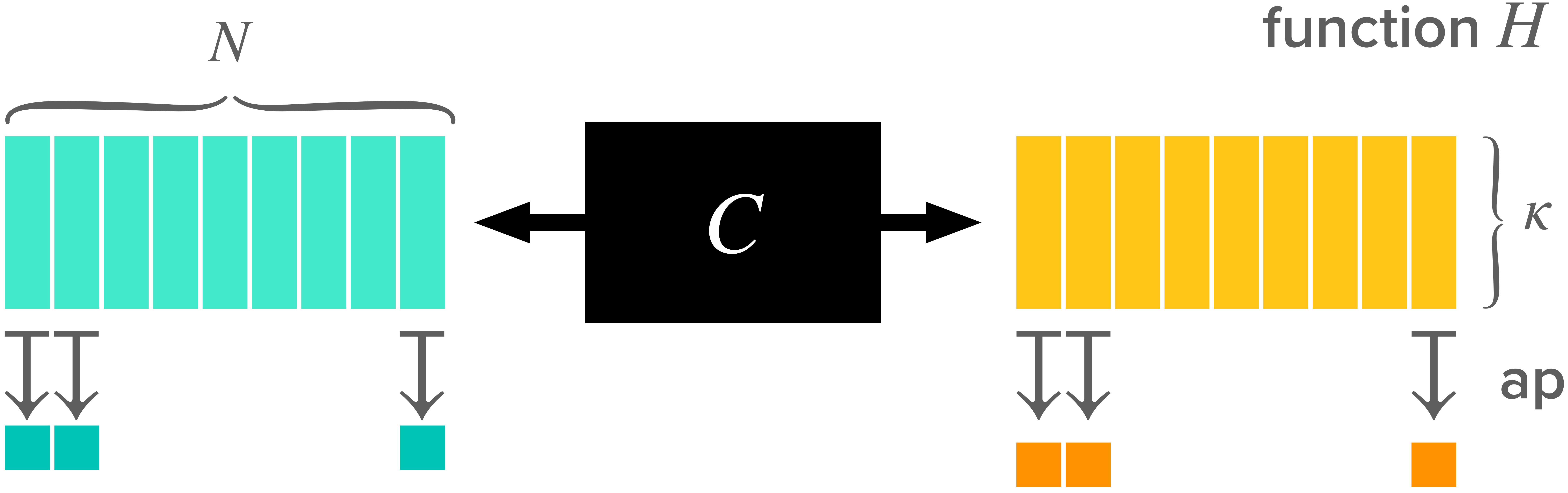
BREAKING CORRELATIONS

[IKNP'03]: Break correlations w/ correlation-robust hash function $H : \{0,1\}^\kappa \to \{0,1\}$



BREAKING CORRELATIONS

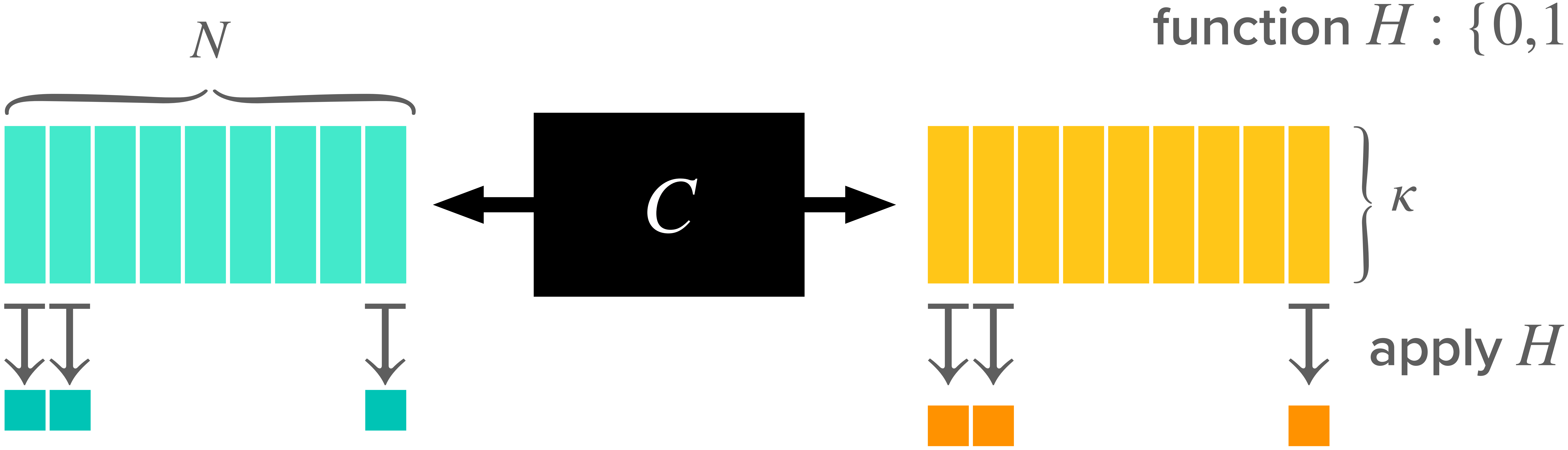
[IKNP'03]: Break correlations w/ correlation-robust hash function $H : \{0,1\}^\kappa \to \{0,1\}$



N independent bit-OTs!

BREAKING CORRELATIONS

[IKNP'03]: Break correlations w/ correlation-robust hash function $H : \{0,1\}^\kappa \to \{0,1\}$

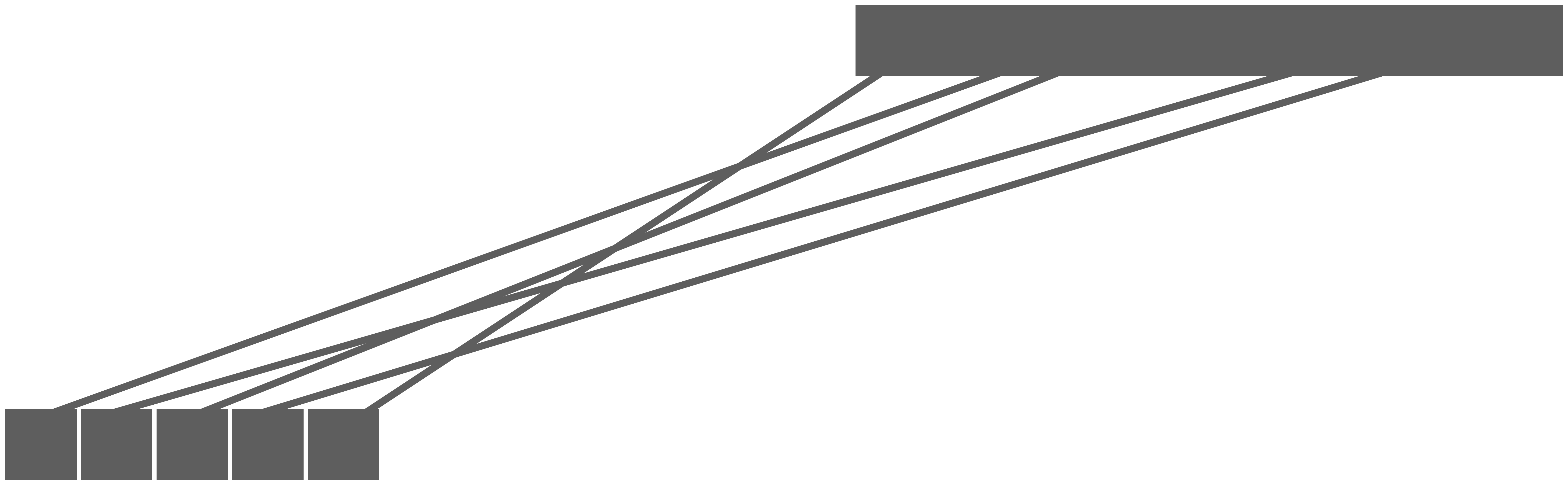


N independent bit-OTs!

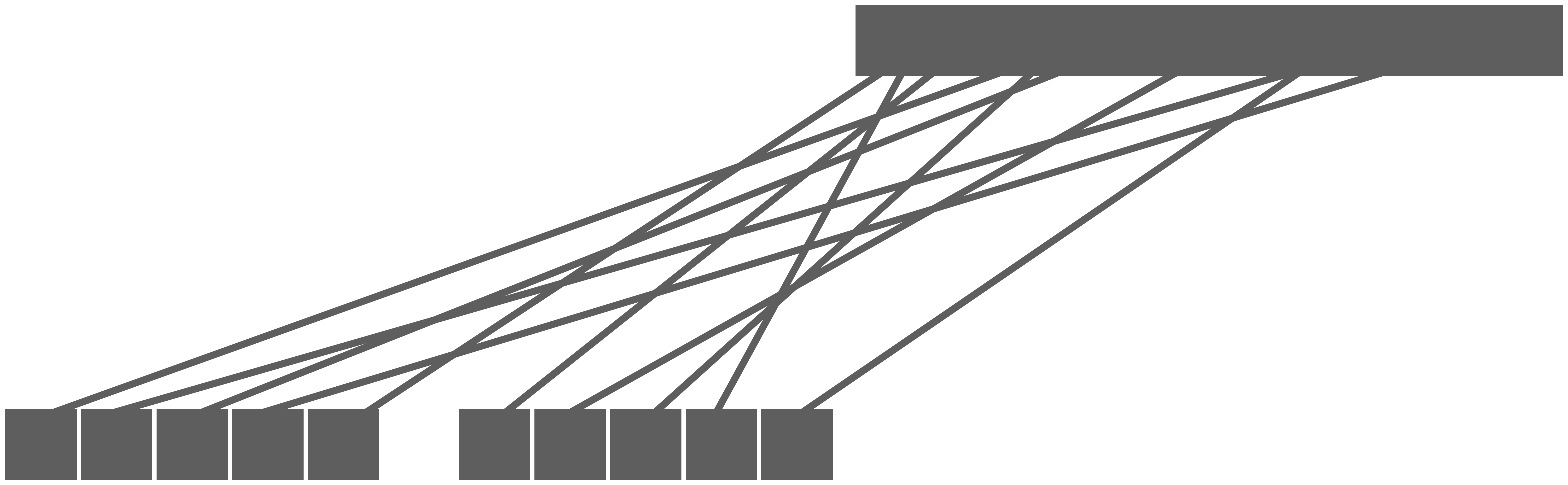
Problem: $\kappa \geq \lambda$ overhead per bit-OT

LOCAL PRG

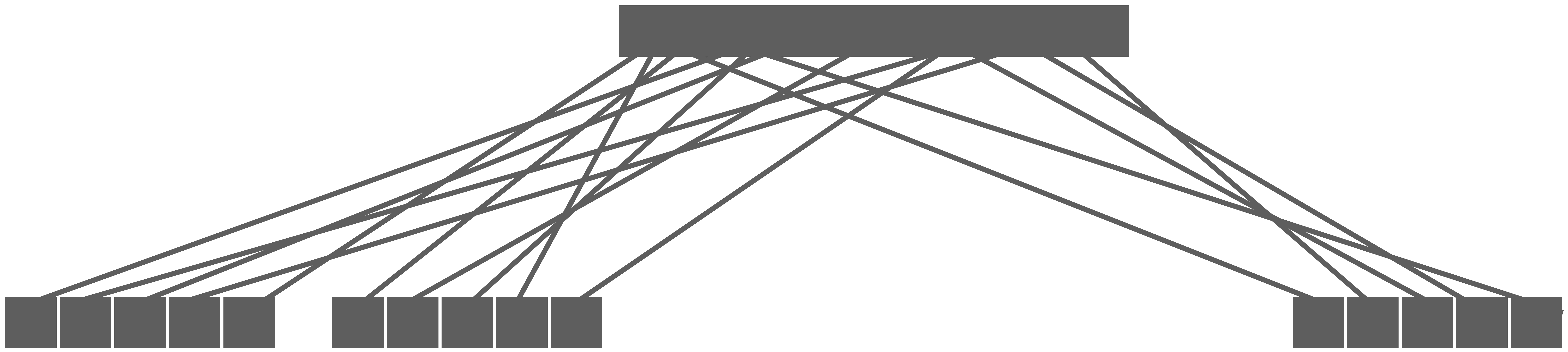
LOCAL PRG



LOCAL PRG



LOCAL PRG



LOCAL PRG

Predicate

$$P : \{0,1\}^{\ell} \rightarrow \{0,1\}$$

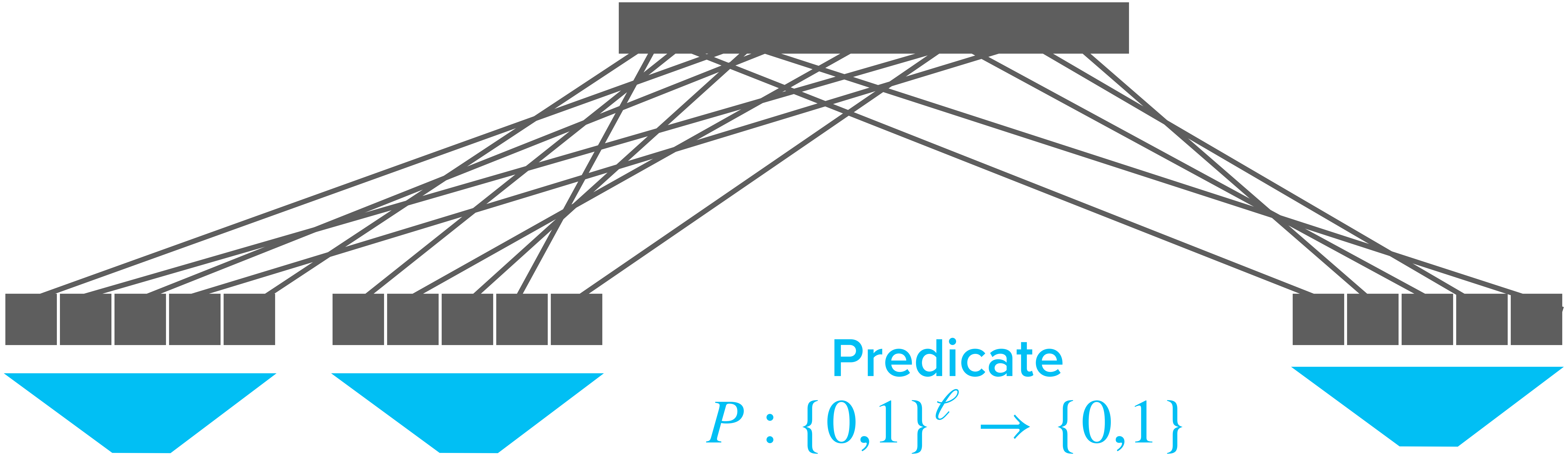
LOCAL PRG

Predicate

$P : \{0,1\}^{\ell} \rightarrow \{0,1\}$

$\ell = O(1)$ (small)

LOCAL PRG

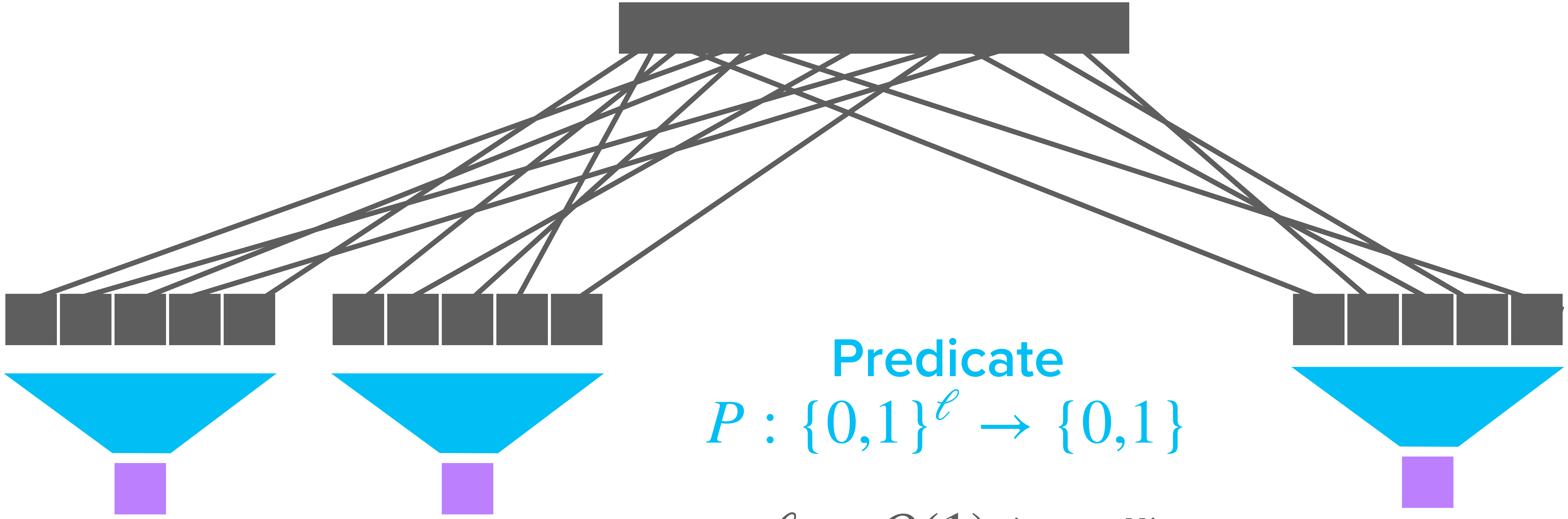


Predicate

$$P : \{0,1\}^{\ell} \to \{0,1\}$$

$\ell = O(1)$ (small)

LOCAL PRG

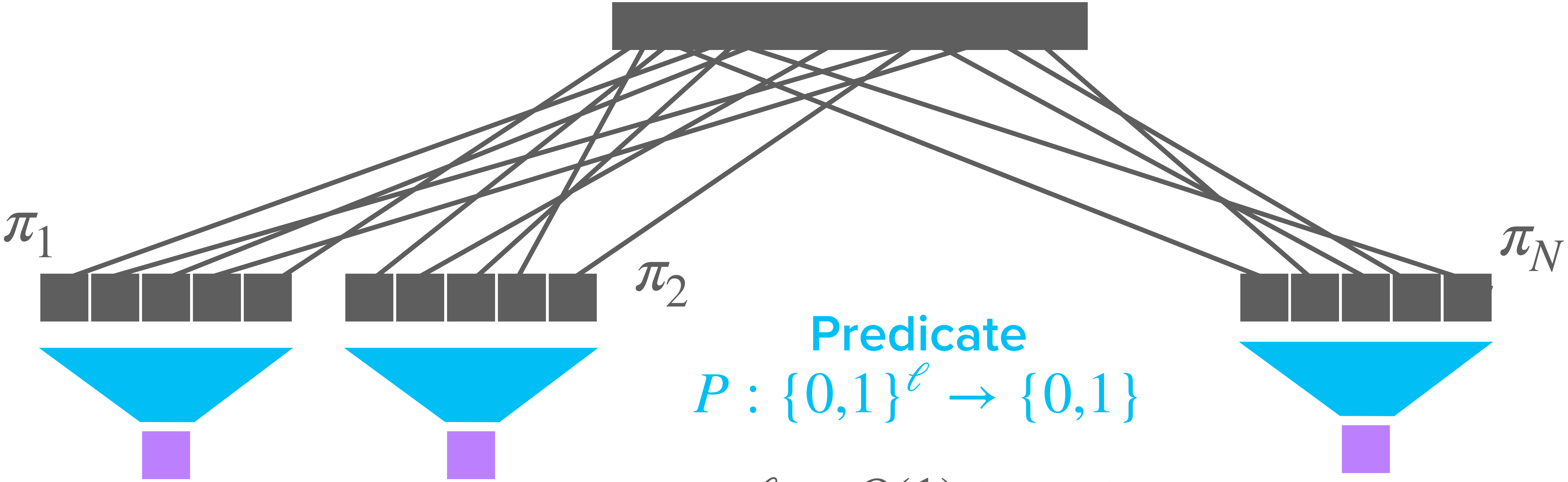


Predicate

$P : \{0,1\}^{\ell} \to \{0,1\}$

$\ell = O(1)$ (small)

LOCAL PRG

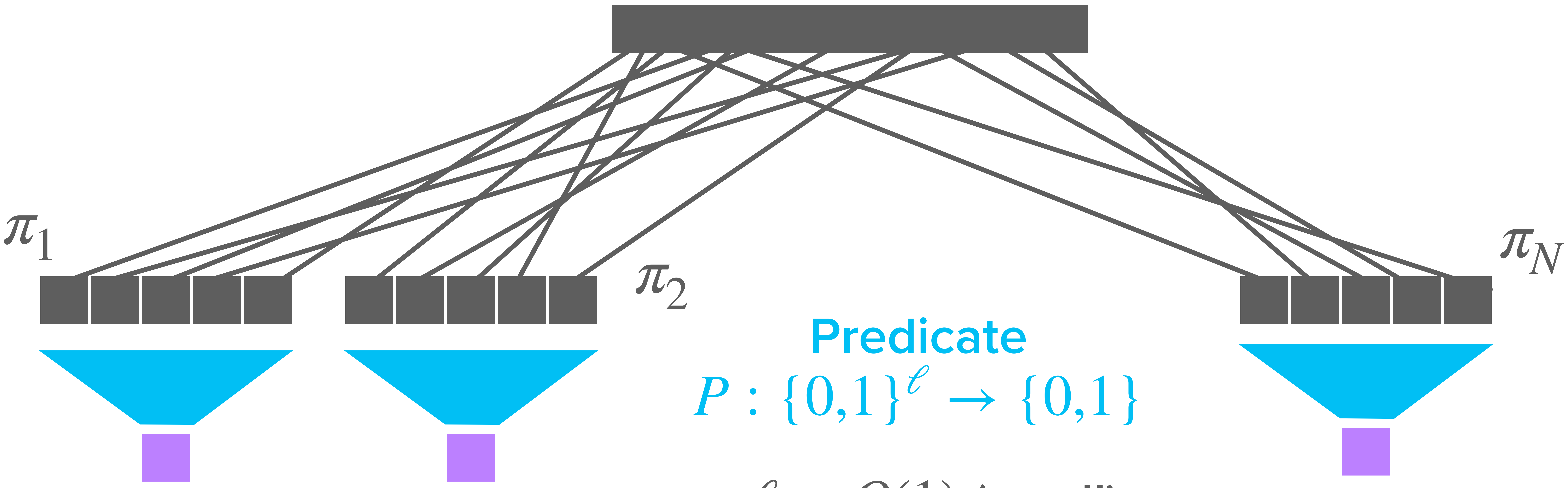


Predicate

$P : \{0,1\}^\ell \to \{0,1\}$

$\ell = O(1)$ (small)

LOCAL PRG



Predicate

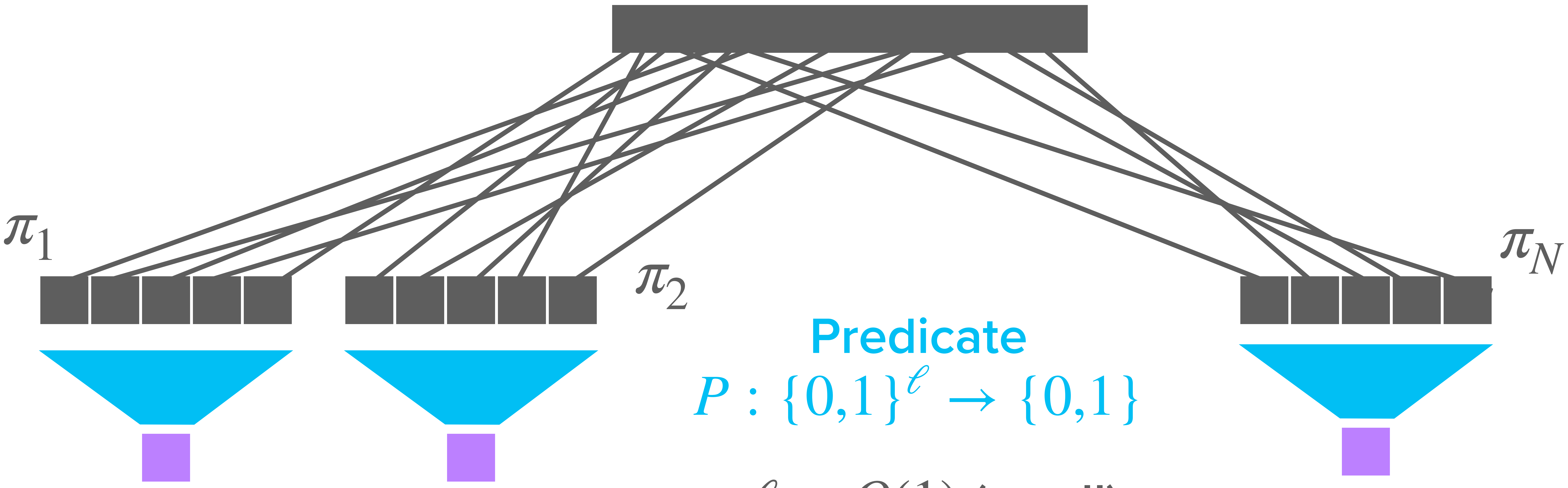
$P : \{0,1\}^{\ell} \to \{0,1\}$

$\ell = O(1)$ (small)

Replace i-th application of H with $P \circ \pi_i$!

LOCAL PRG

[Goldreich'00]

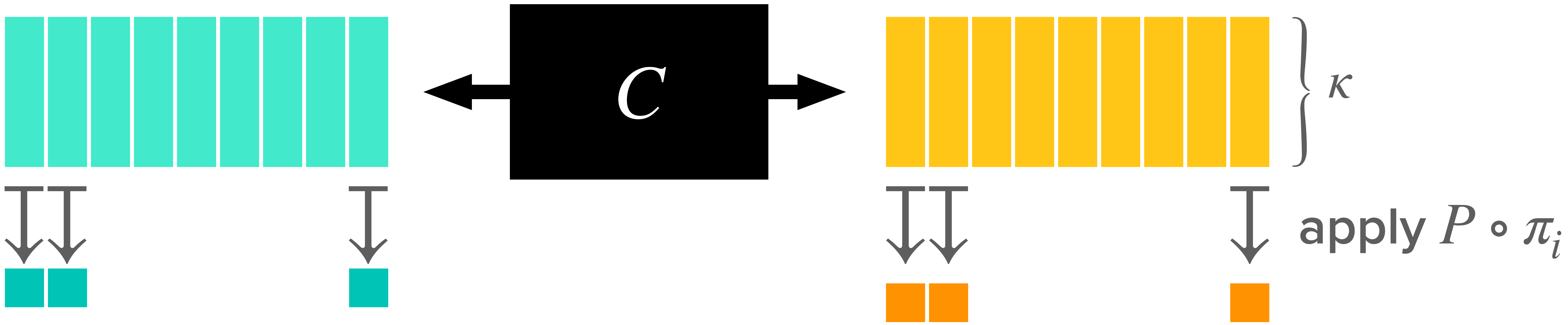


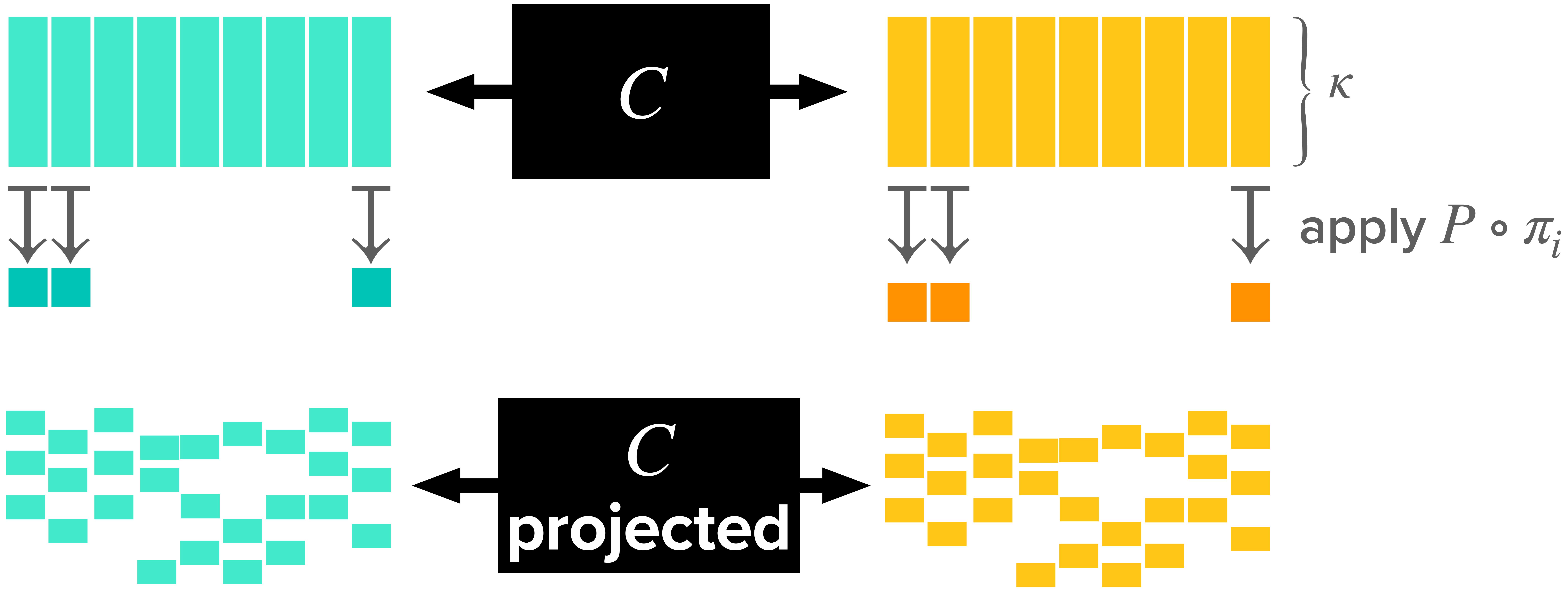
Predicate

$P : \{0,1\}^\ell \to \{0,1\}$

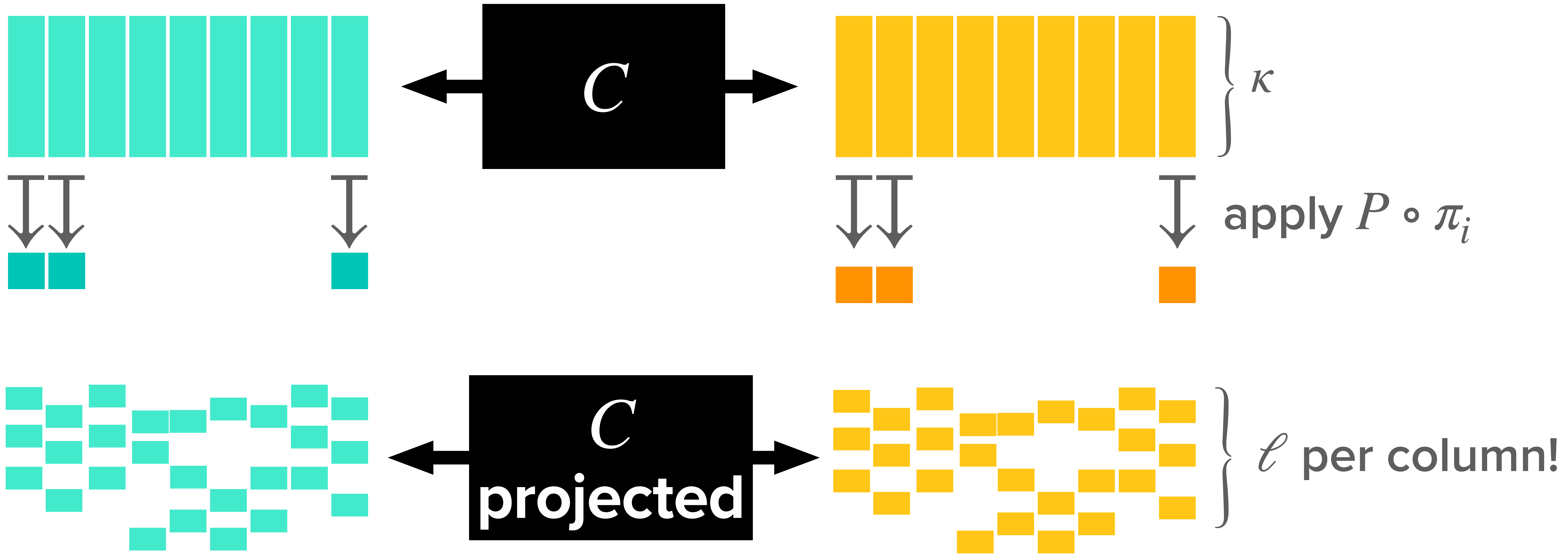
$\ell = O(1)$ (small)

Replace i-th application of H with $P \circ \pi_i$!

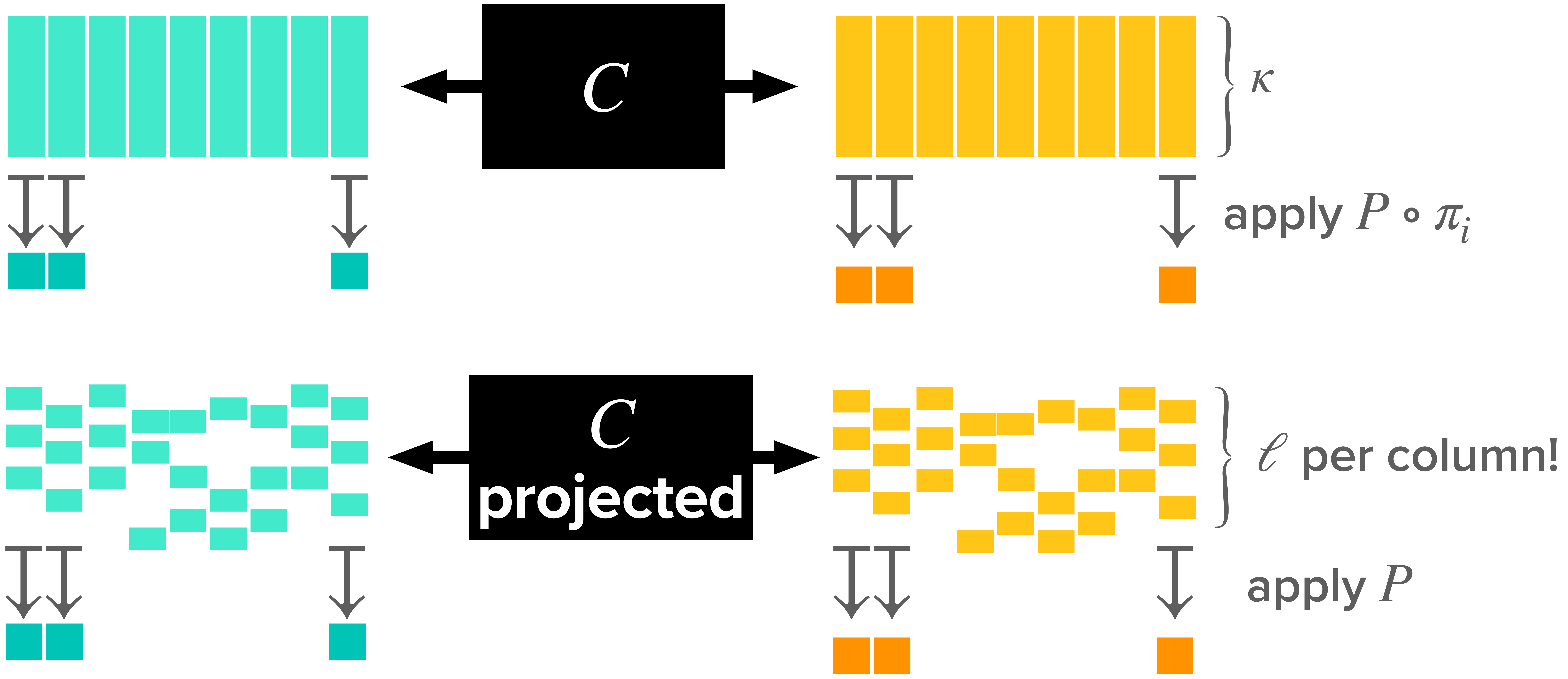

C
κ
apply $P \circ \pi_i$



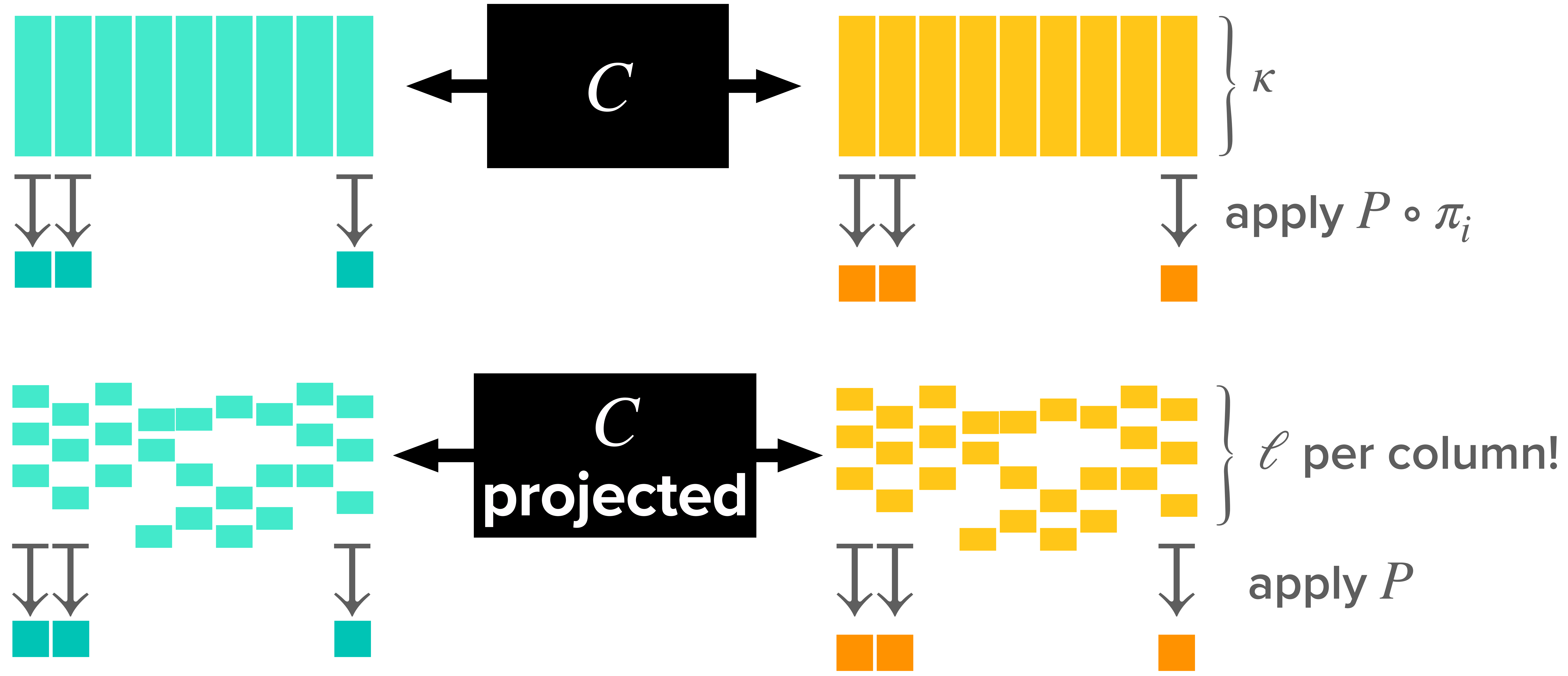
C
κ
apply $P \circ \pi_i$
C
projected



C
κ
apply $P \circ \pi_i$
C
projected
ℓ per column!



C
κ
apply $P \circ \pi_i$
C
projected
ℓ per column!
apply P



Need new sharing schemes for "projections" of structured vectors

CONCRETE EFFICIENCY ESTIMATES

CONCRETE EFFICIENCY ESTIMATES

Primal Construction

CONCRETE EFFICIENCY ESTIMATES

Primal Construction	< 300 ops. per OT

CONCRETE EFFICIENCY ESTIMATES

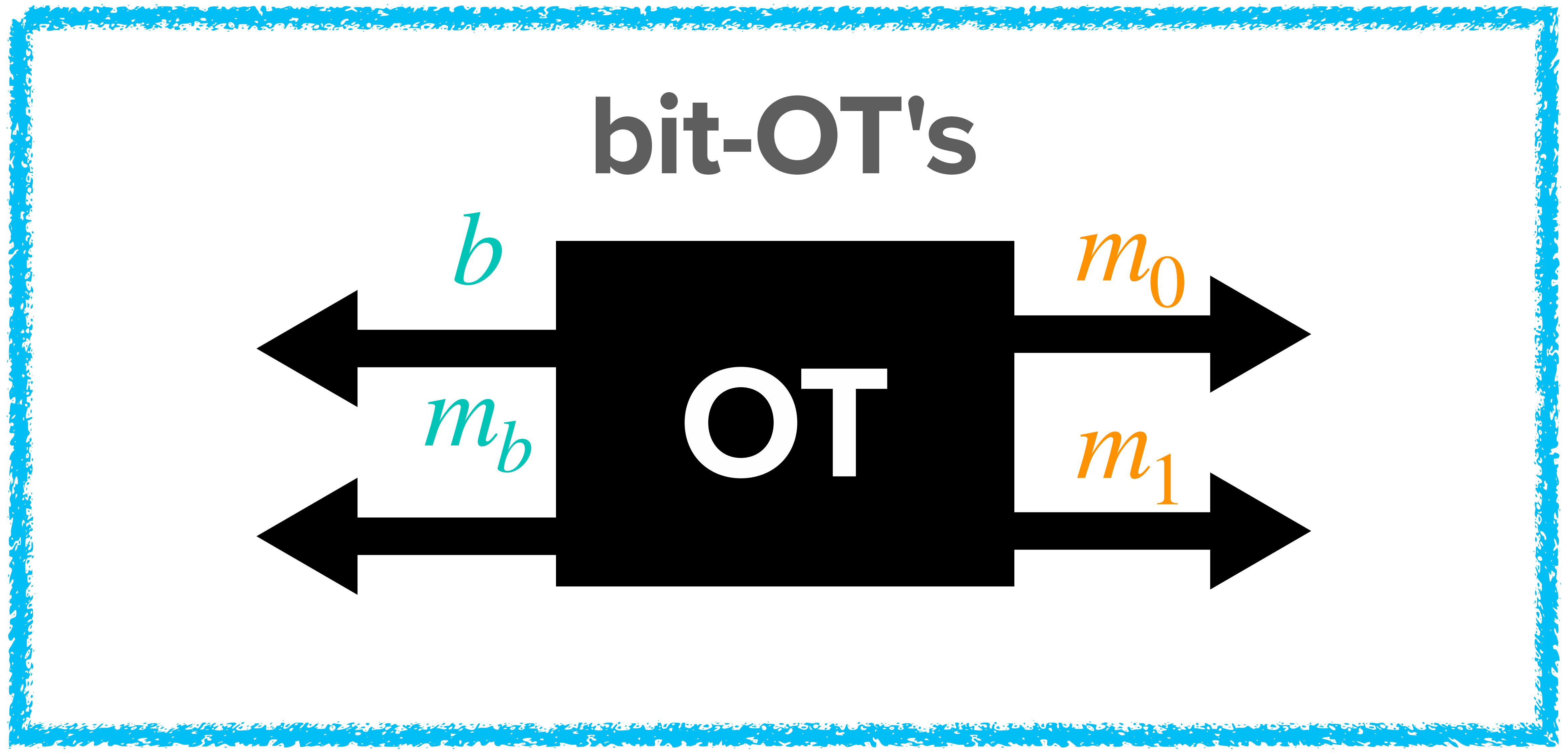
Primal Construction	< 300 ops. per OT
Dual Construction	

CONCRETE EFFICIENCY ESTIMATES

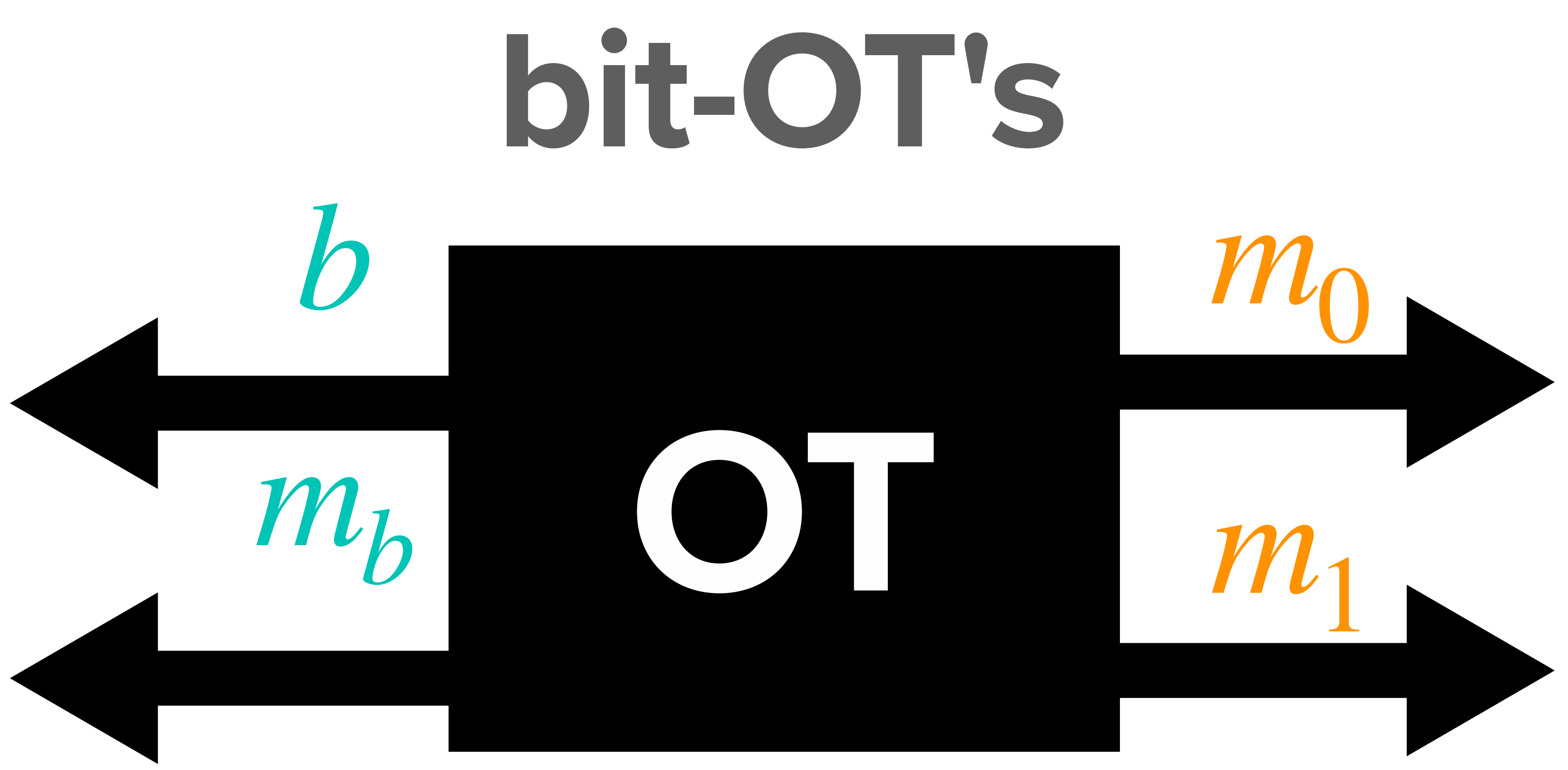
Primal Construction	< 300 ops. per OT
Dual Construction	< 100 ops. per OT

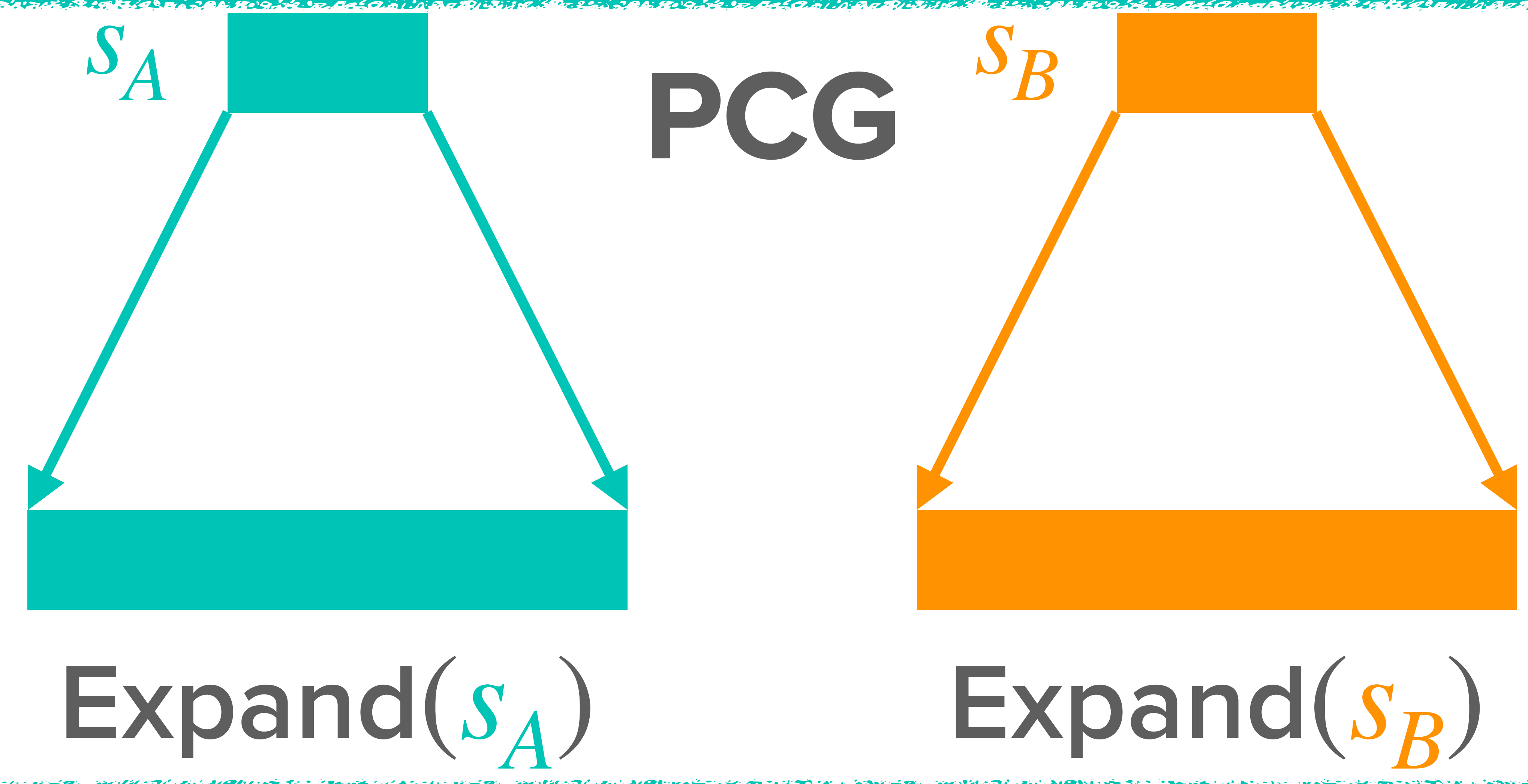
RECAP

RECAP

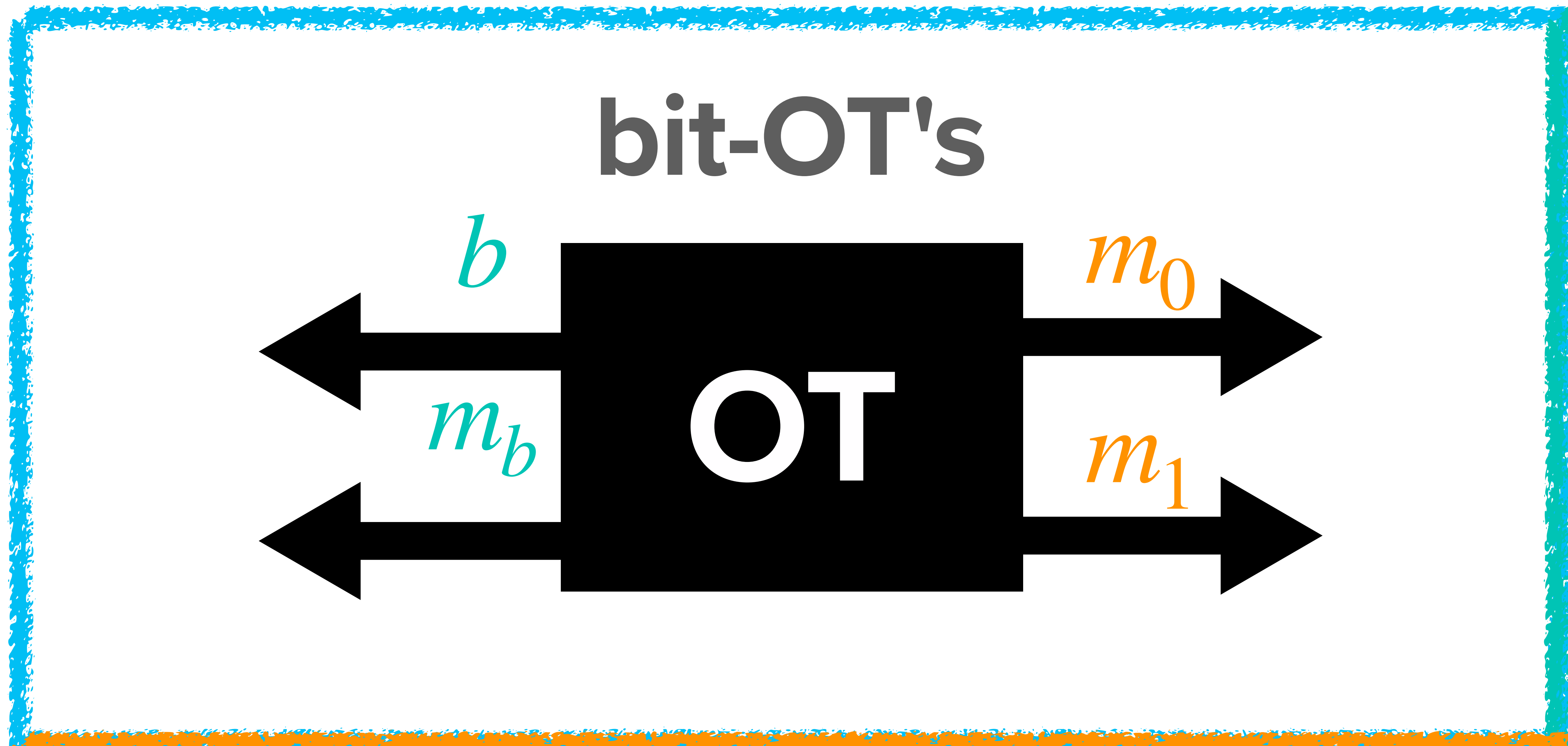


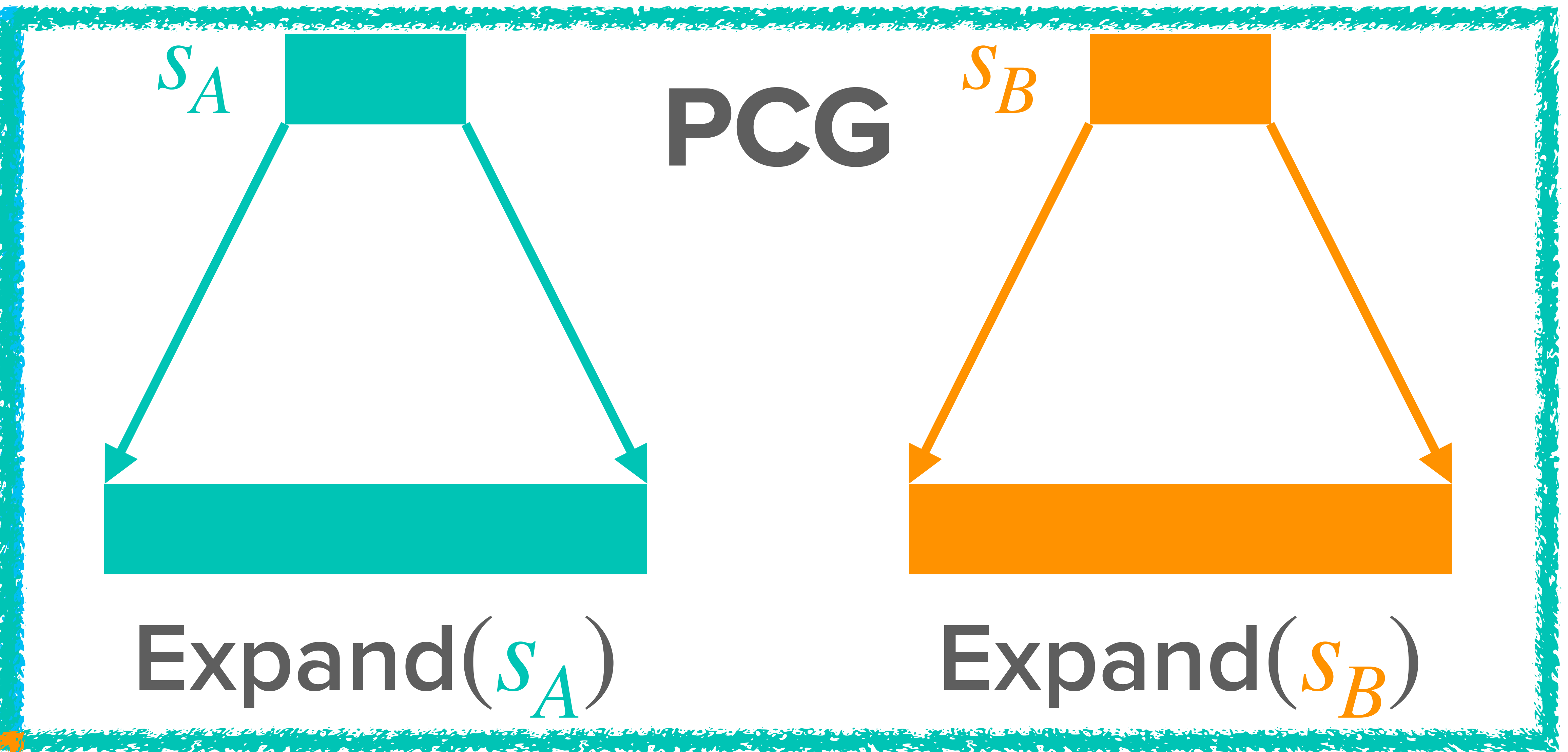
RECAP

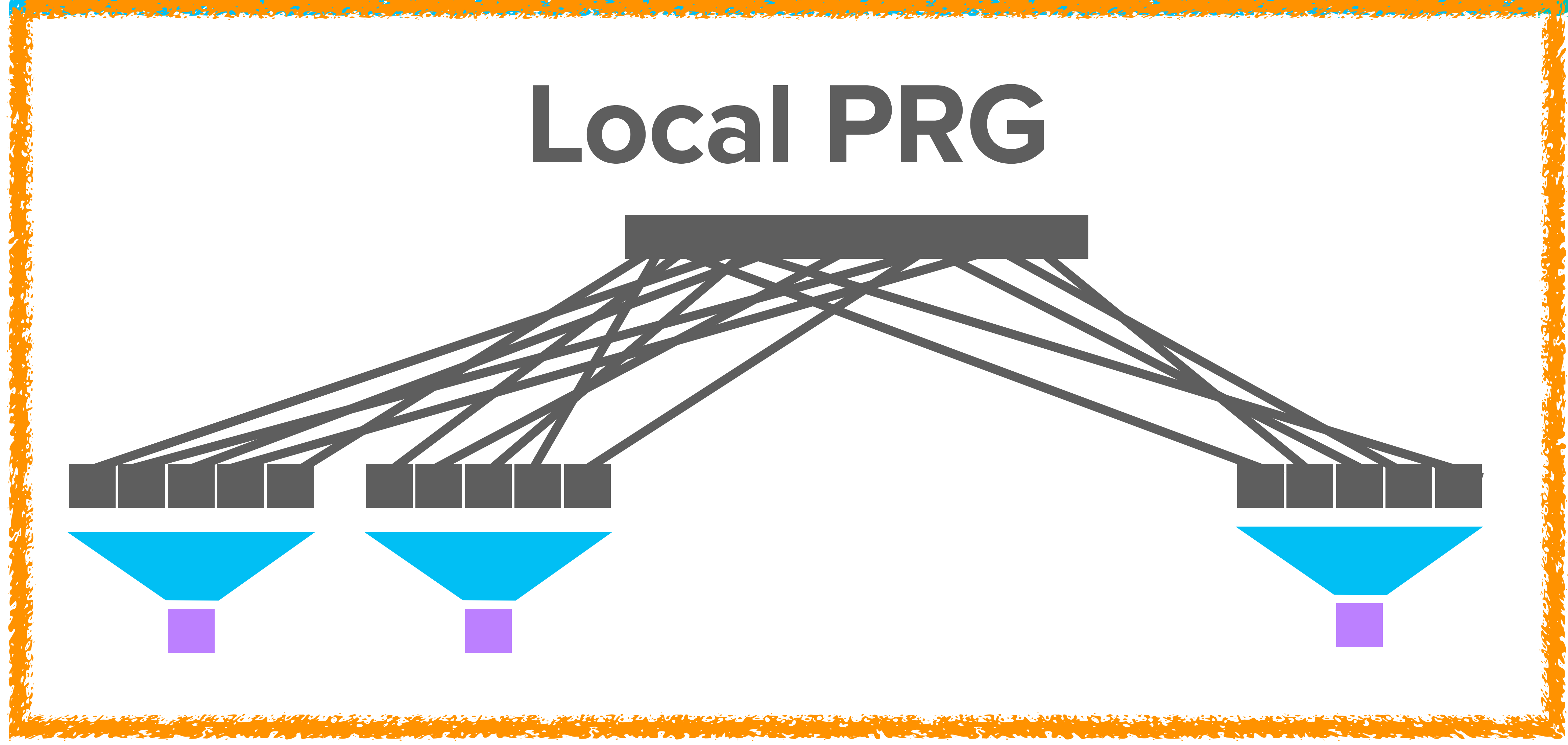




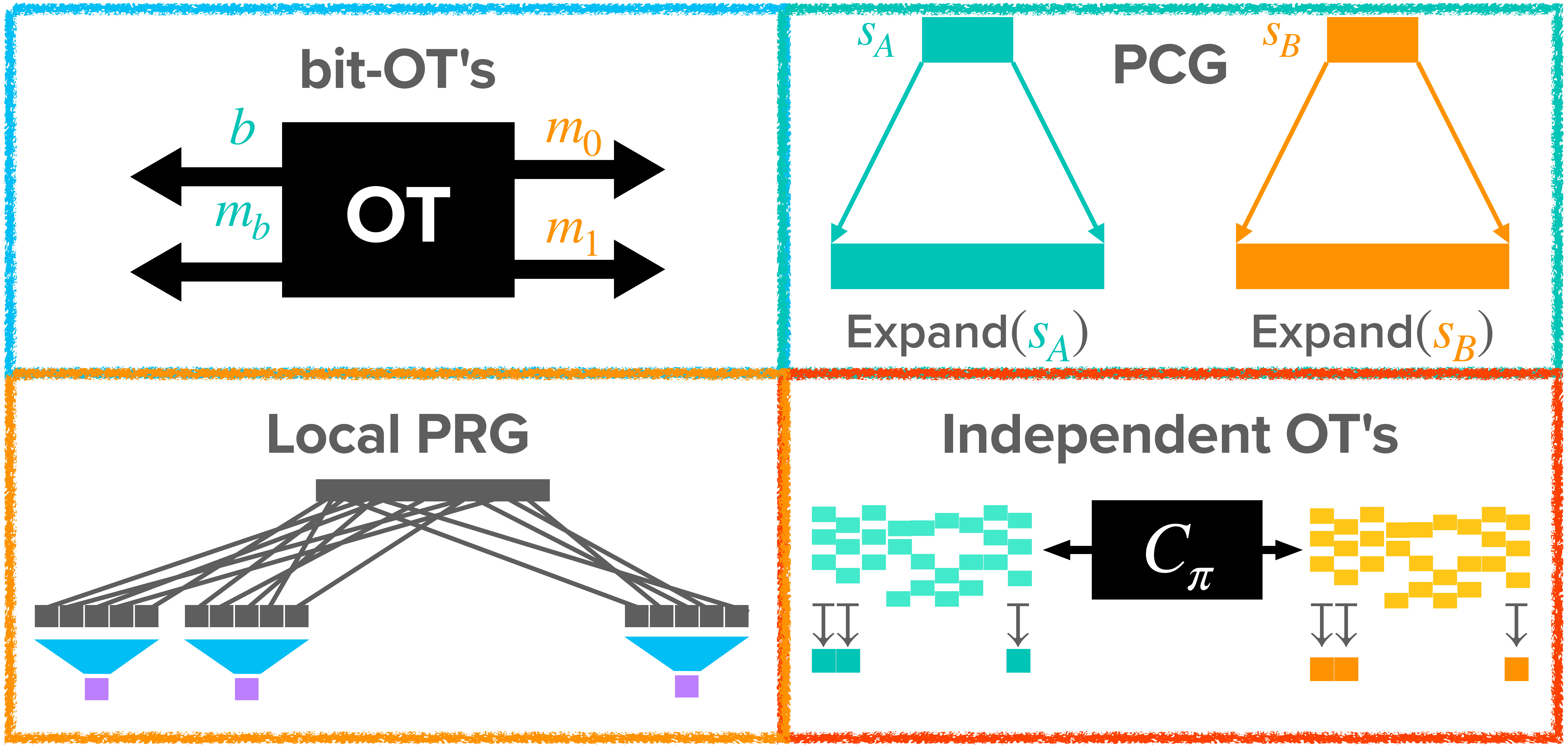
RECAP





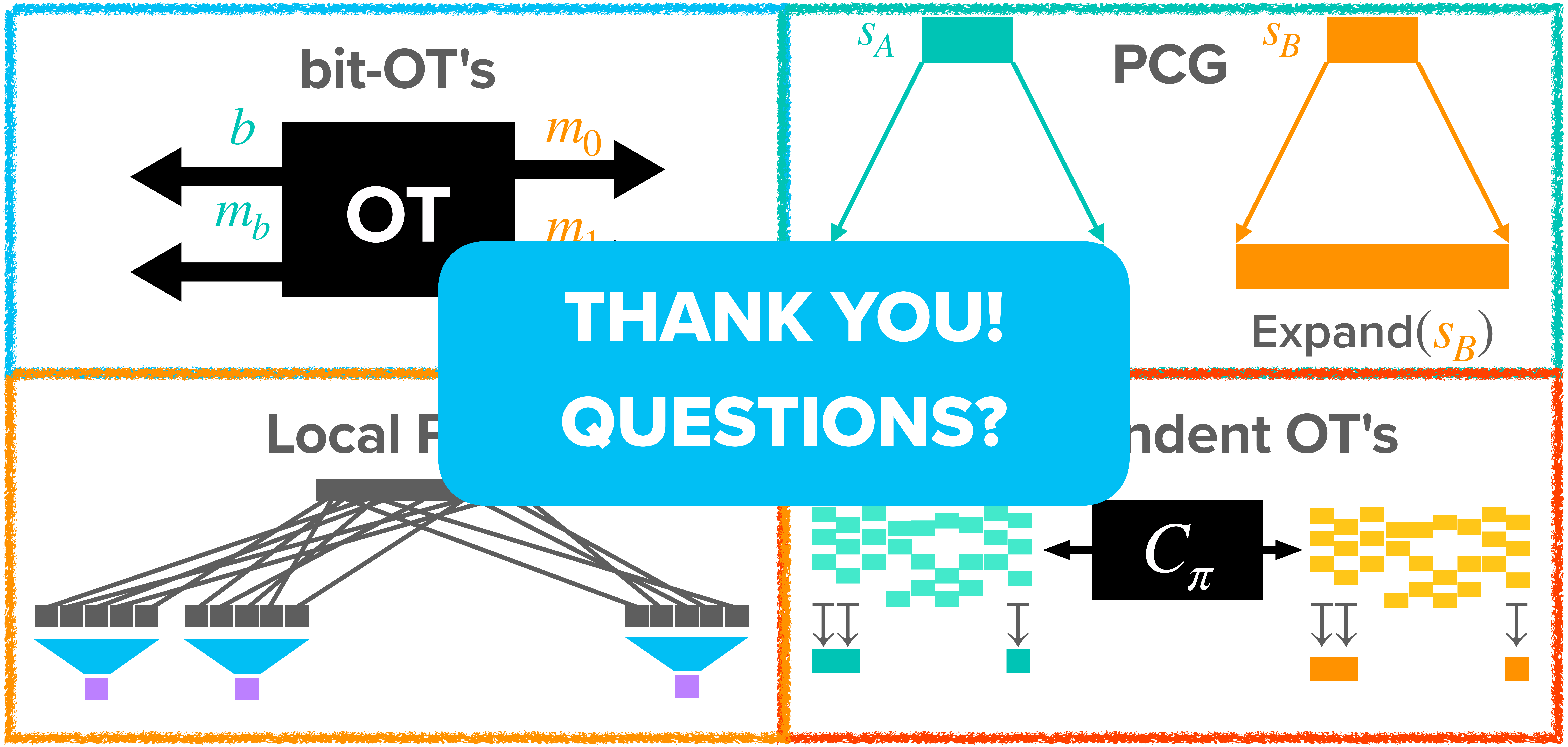


RECAP



bit-OT's
OT
b
m_b
m_0
m_1
PCG
s_A
s_B
Expand(s_A)
Expand(s_B)
Local PRG
Independent OT's
C_π

RECAP



THANK YOU!
QUESTIONS?