TreePIR: Sublinear Time Polylog Bandwidth Private Information Retrieval from DDH

Arthur Lazzaretti

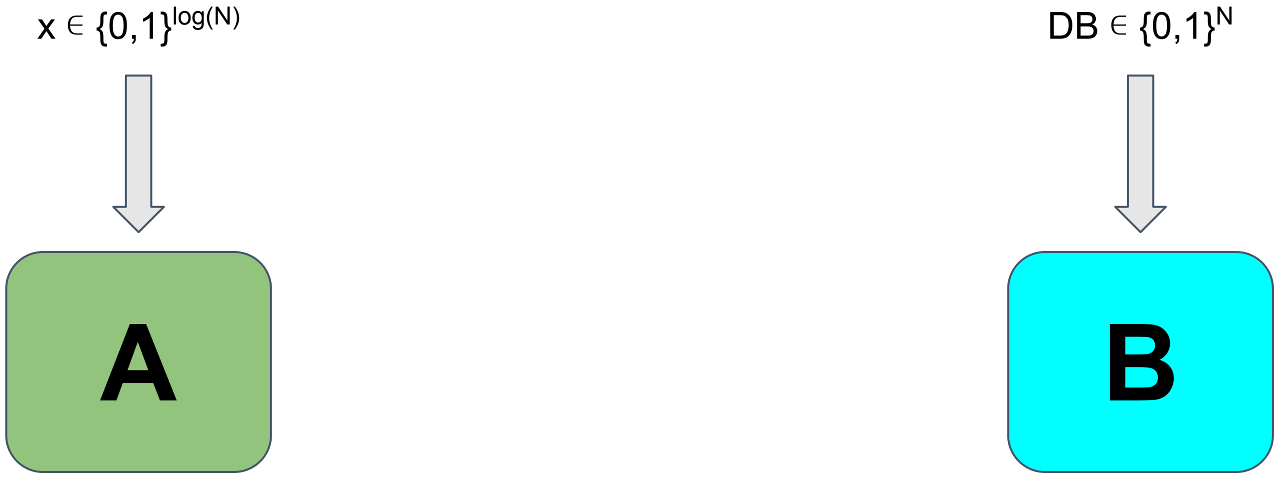
joint work with Charalampos Papamanthou

Private Information Retrieval [CGKM ‘95, KO ‘97,....]

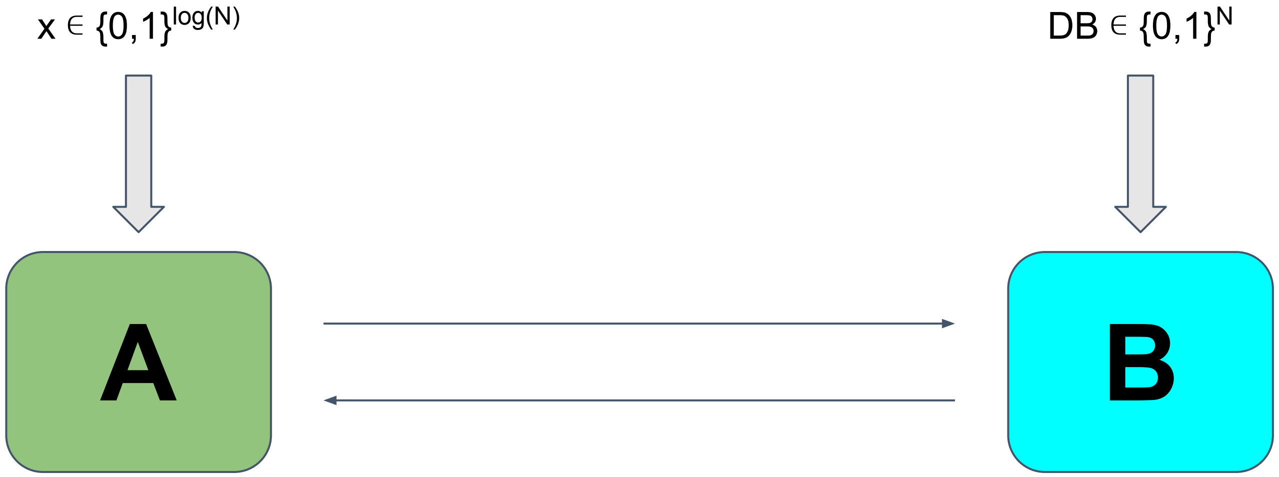
A

B

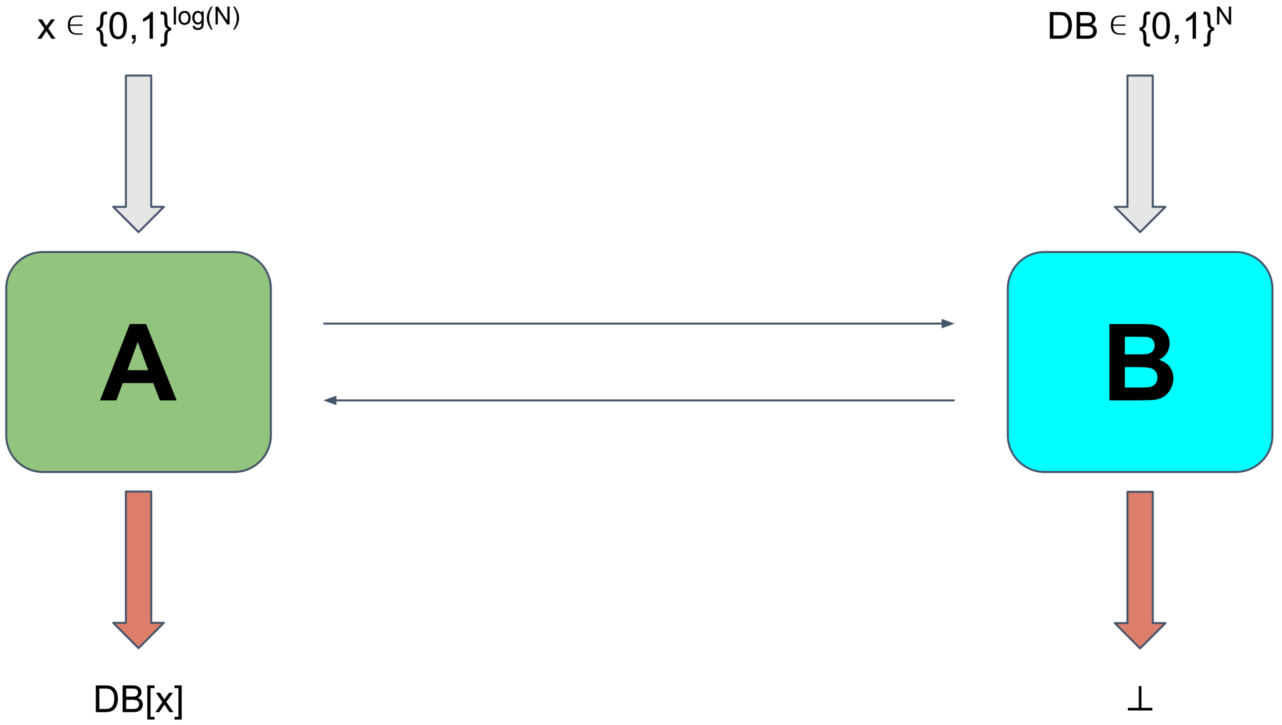
Private Information Retrieval [CGKM ‘95, KO ‘97,....]



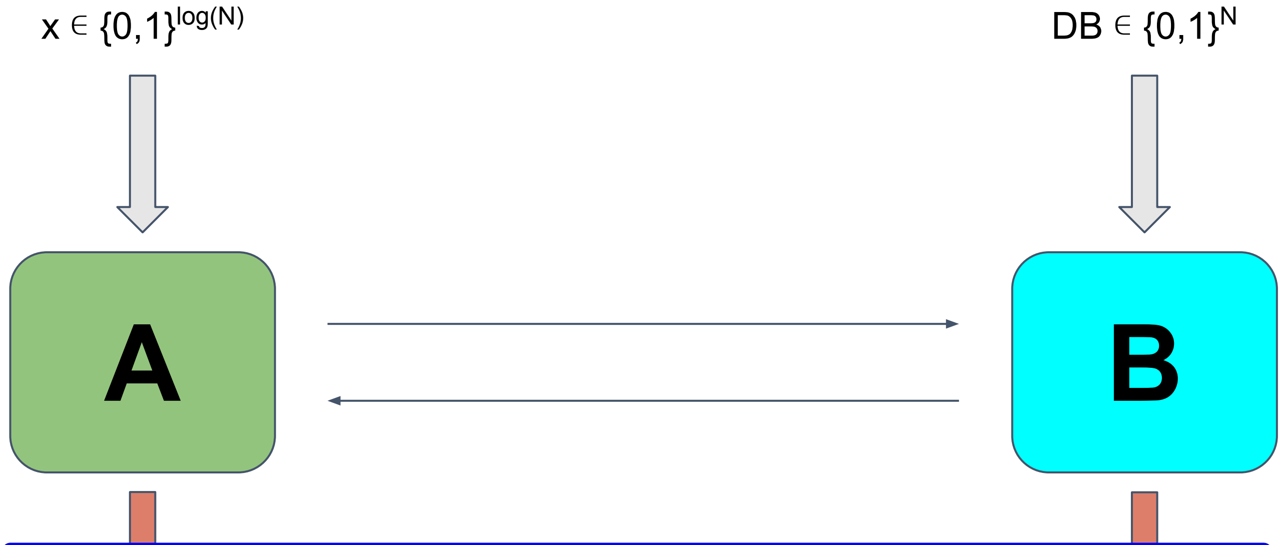
Private Information Retrieval [CGKM ‘95, KO ‘97,....]



Private Information Retrieval [CGKM ‘95, KO ‘97,....]

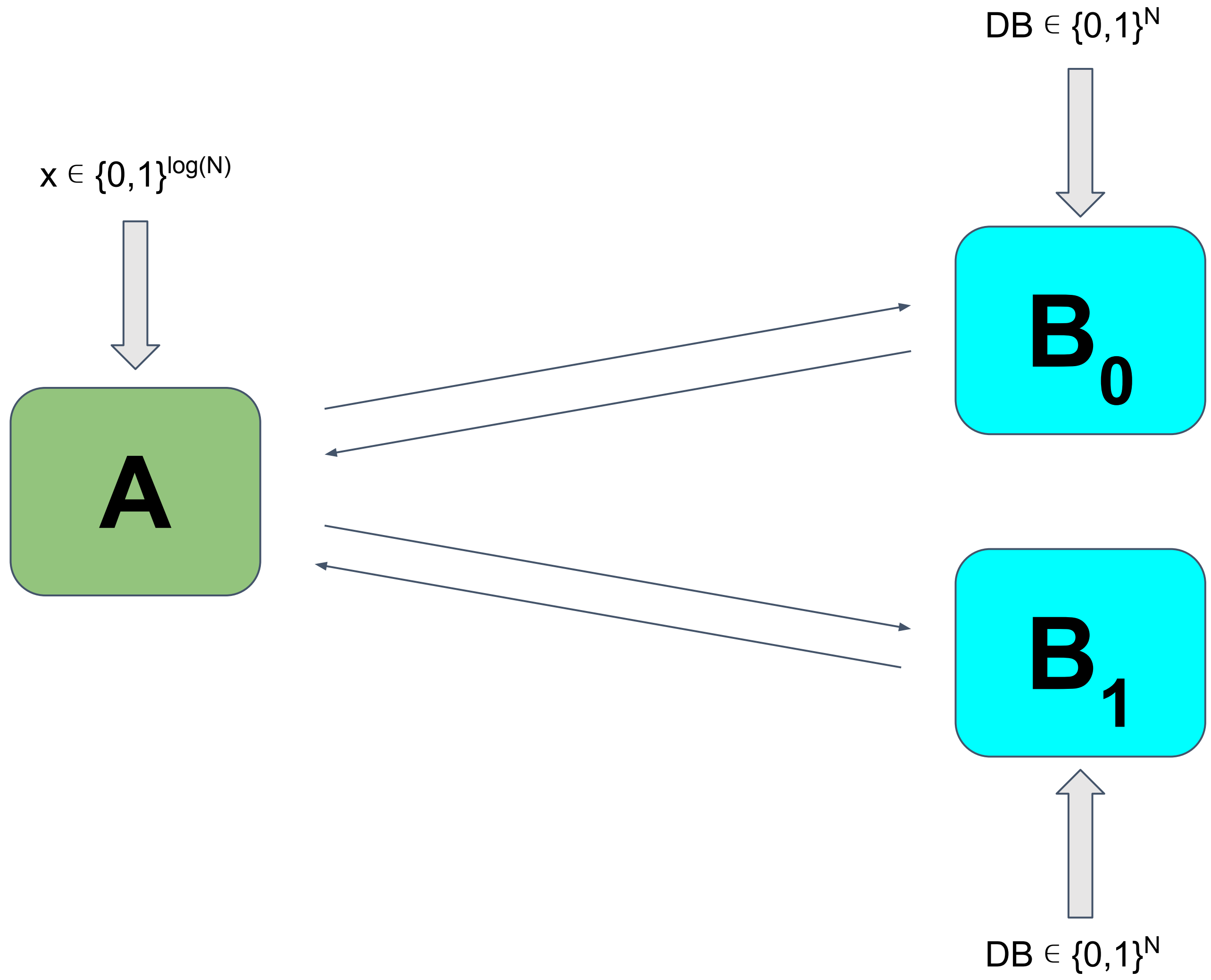


Private Information Retrieval [CGKM ‘95, KO ‘97,....]

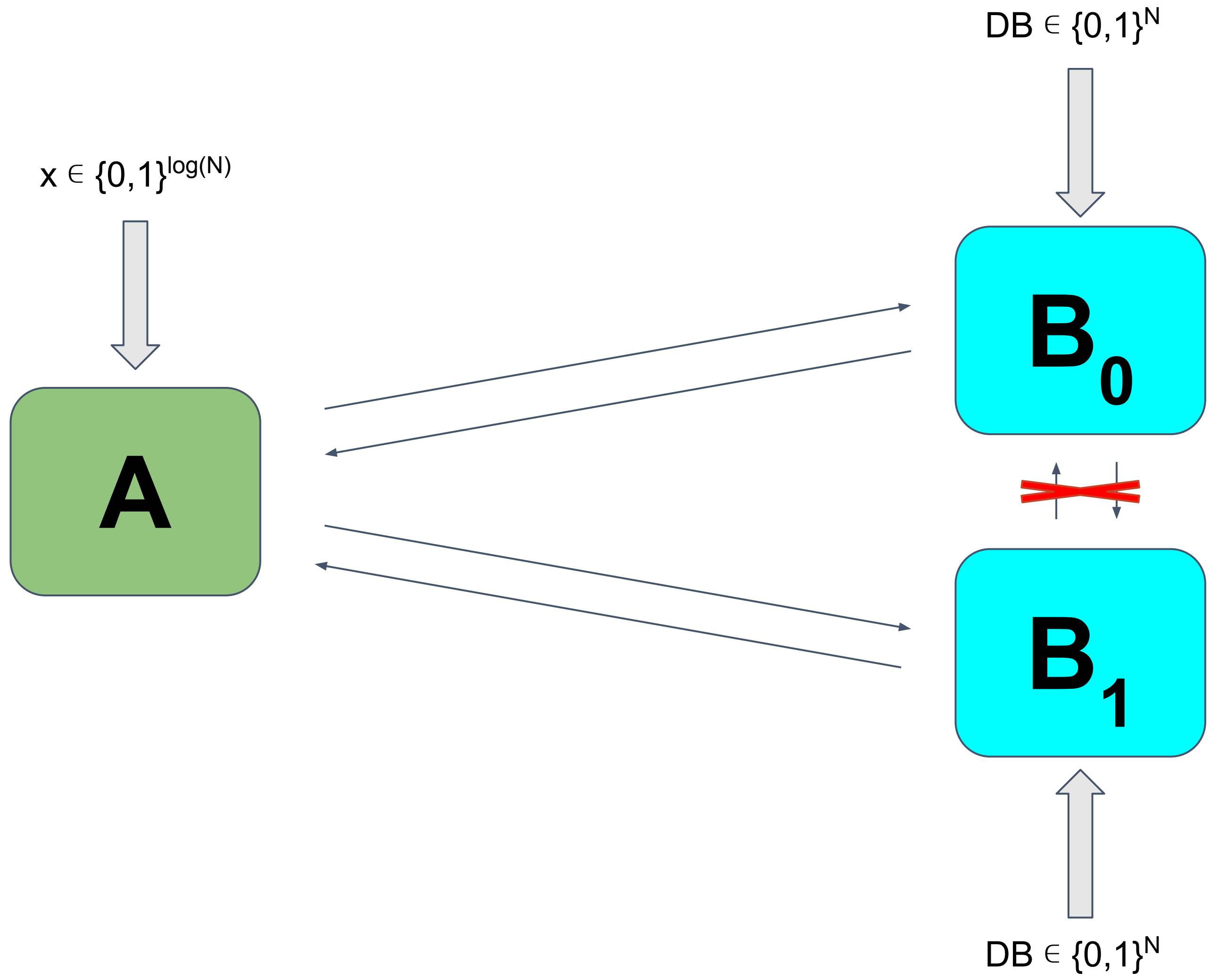


Privacy:
There exists a PPT simulator A* with no knowledge of x that can act as A and B cannot tell the difference

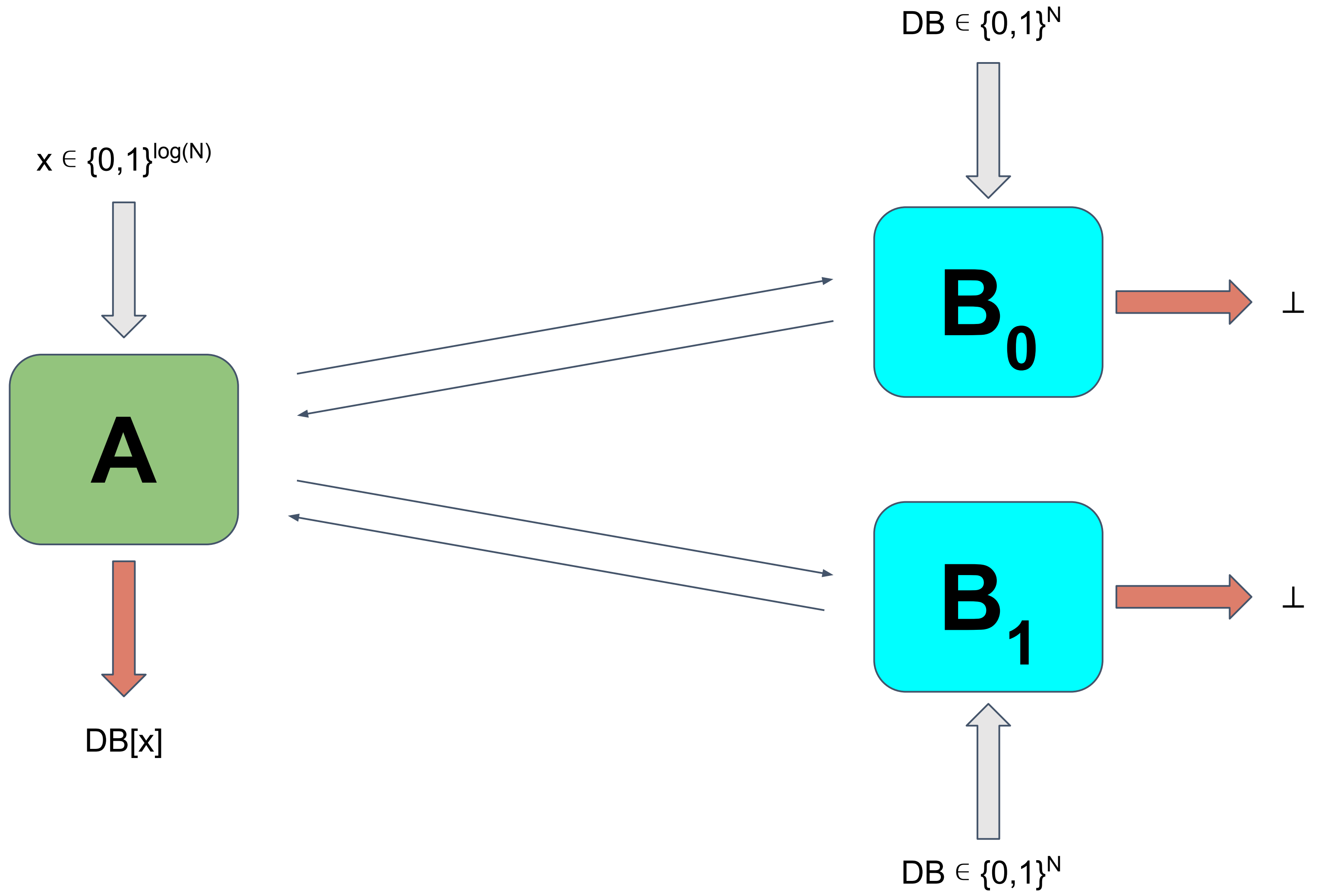
Two-Server Private Information Retrieval [CGKM ‘95, KO ‘97,....]



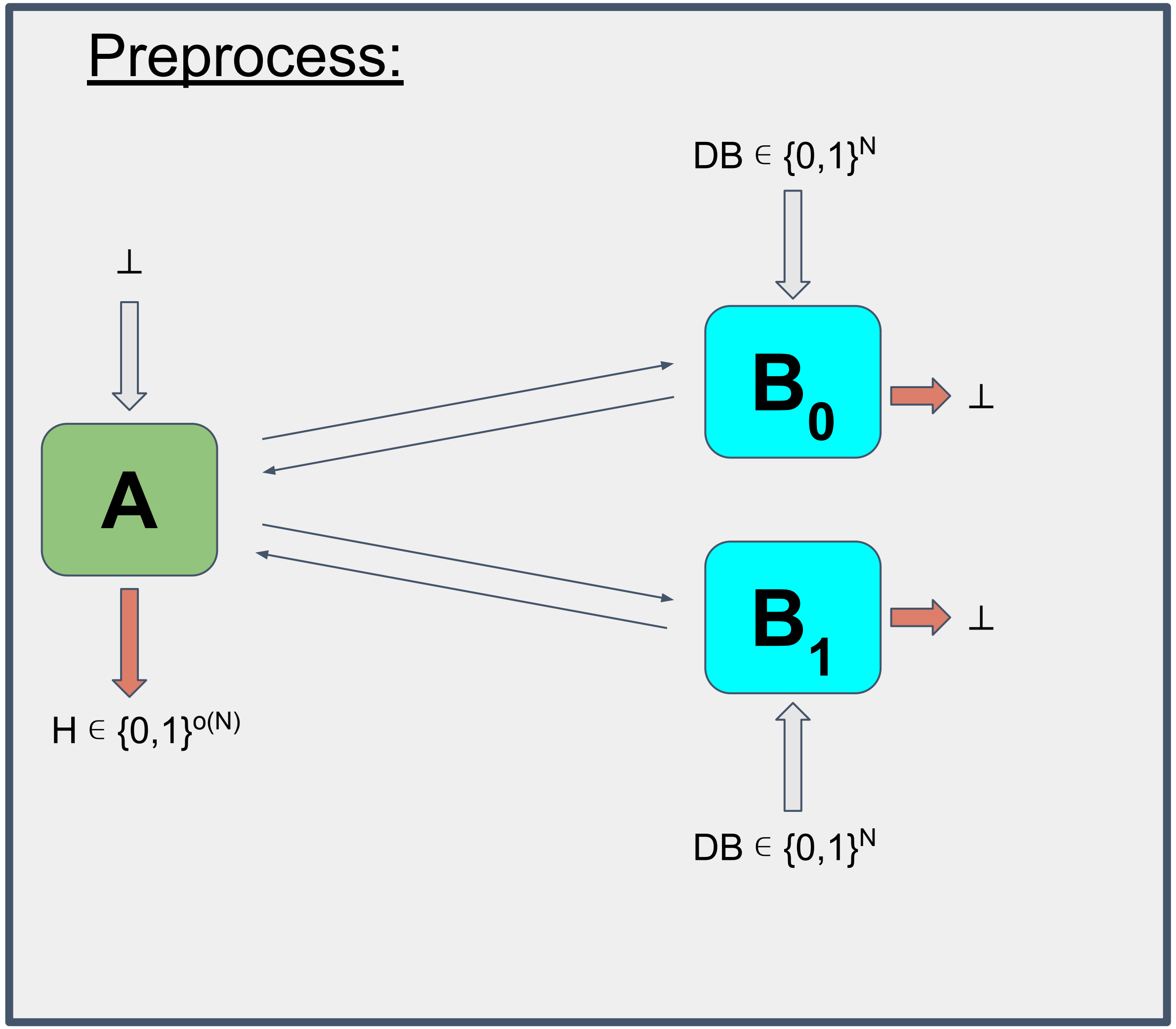
Two-Server Private Information Retrieval [CGKM ‘95, KO ‘97,....]



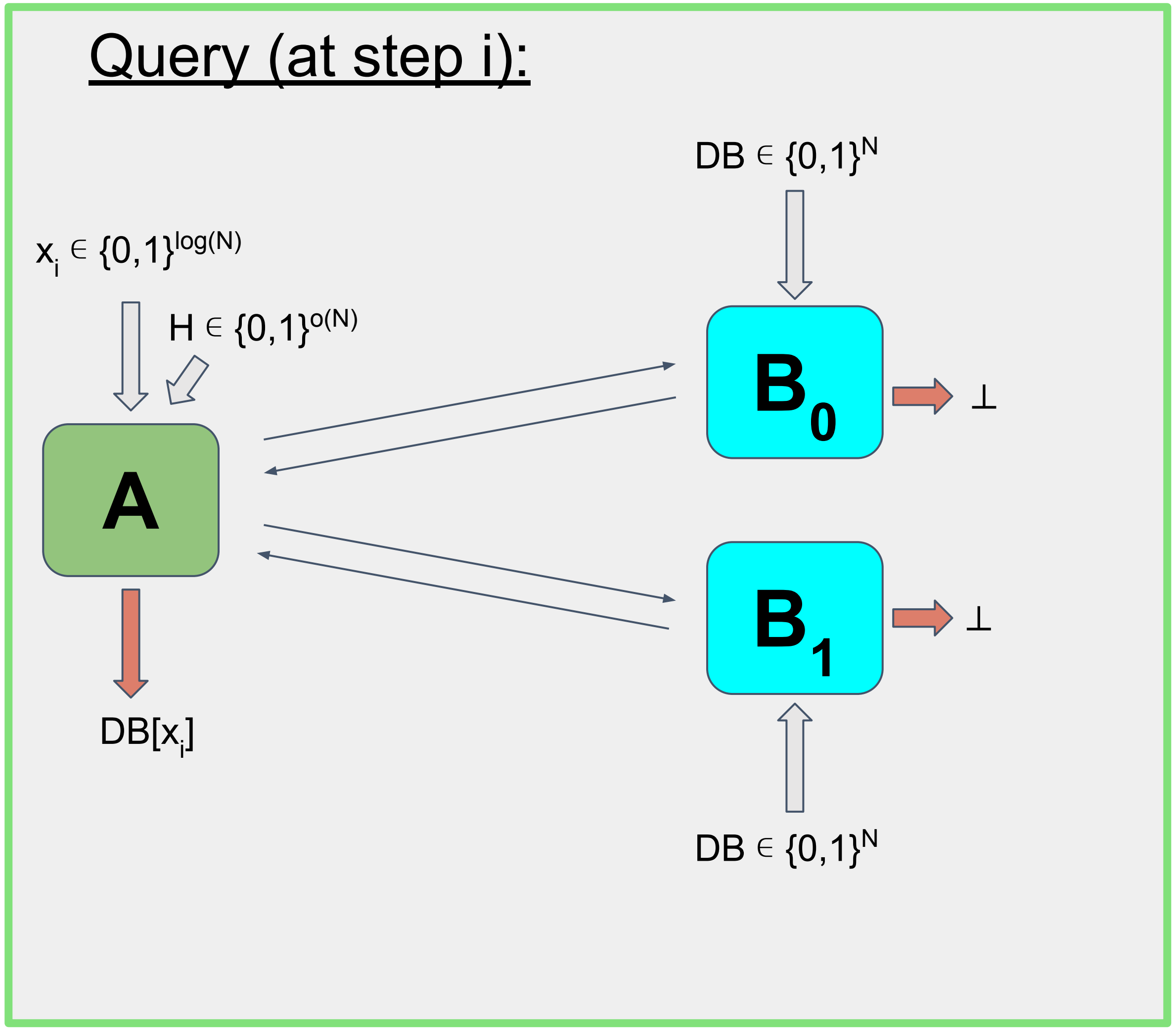
Two-Server Private Information Retrieval [CGKM ‘95, KO ‘97,....]



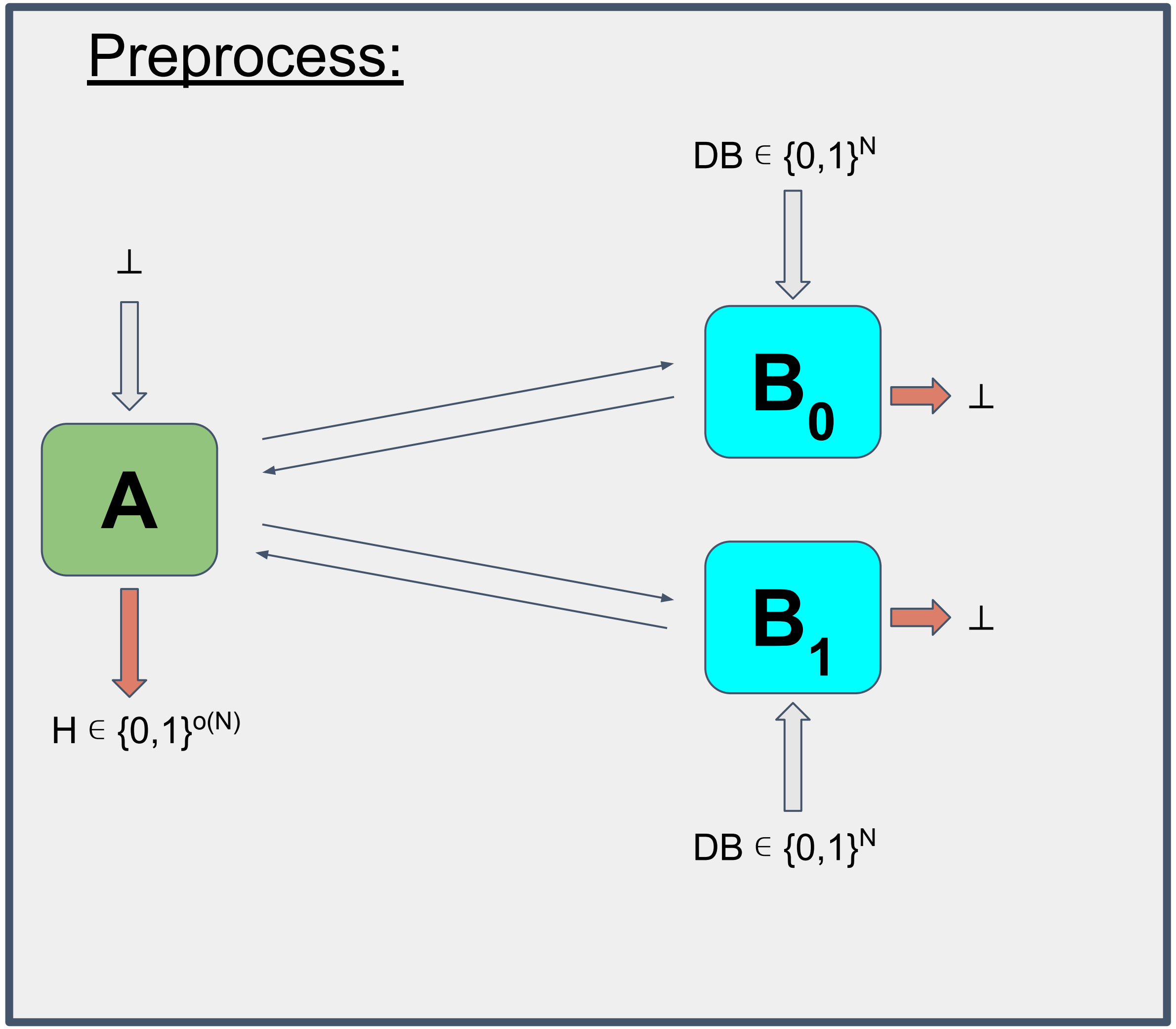
Two-Server PIR, Client Preprocessing [BIM ‘04, …, CK ‘20, ...]

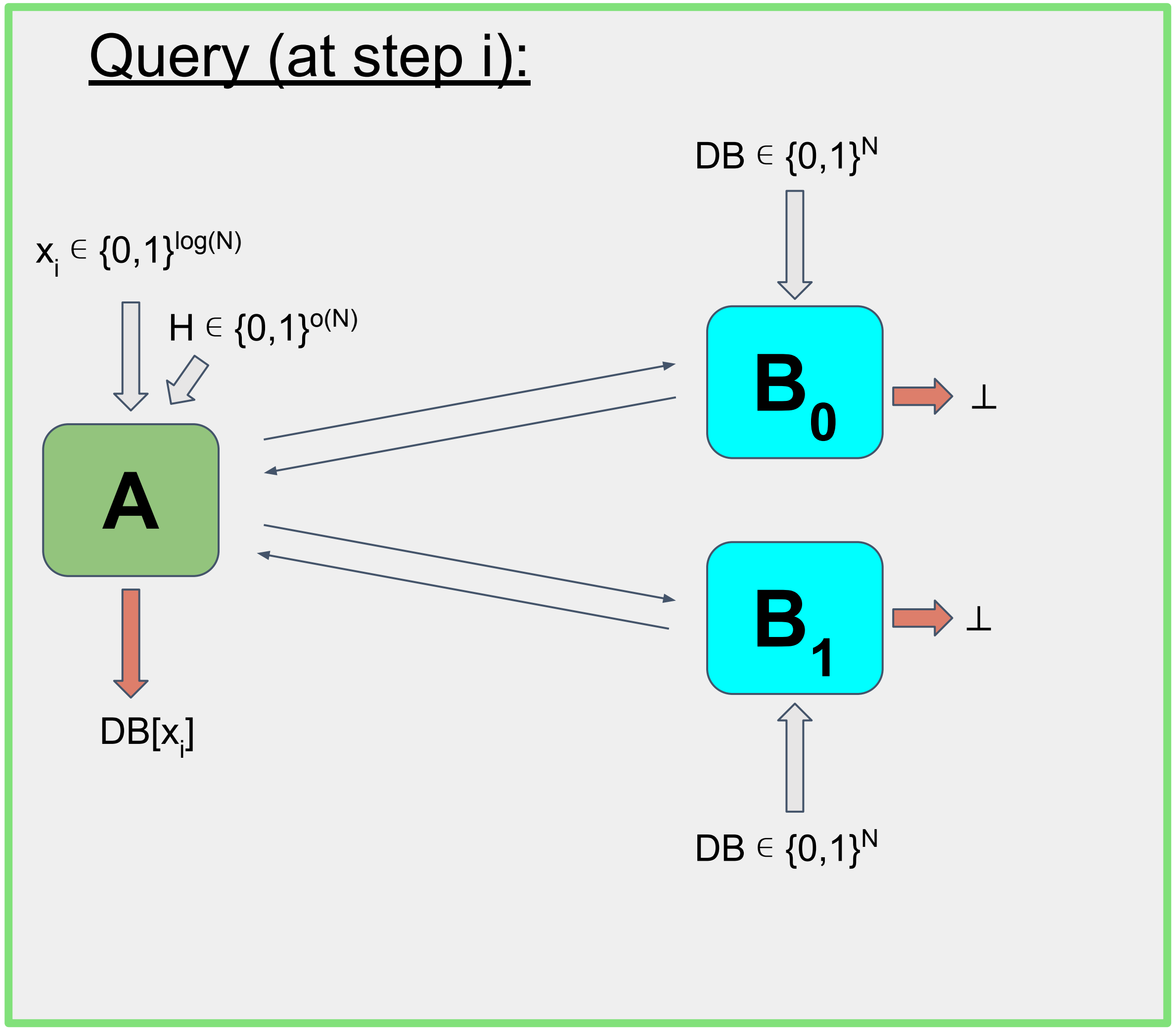


Two-Server PIR, Client Preprocessing [BIM ‘04, …, CK ‘20, ...]

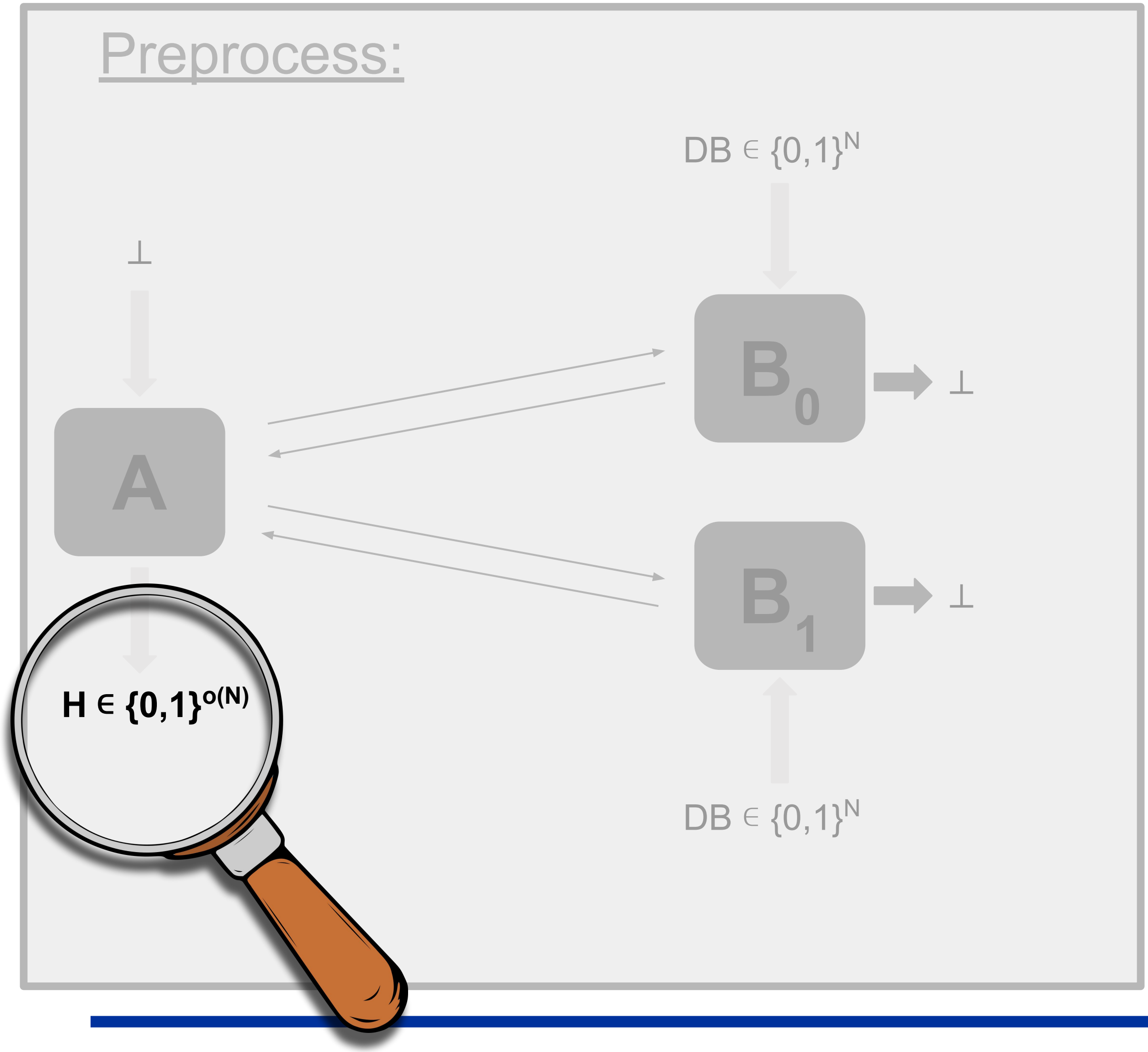


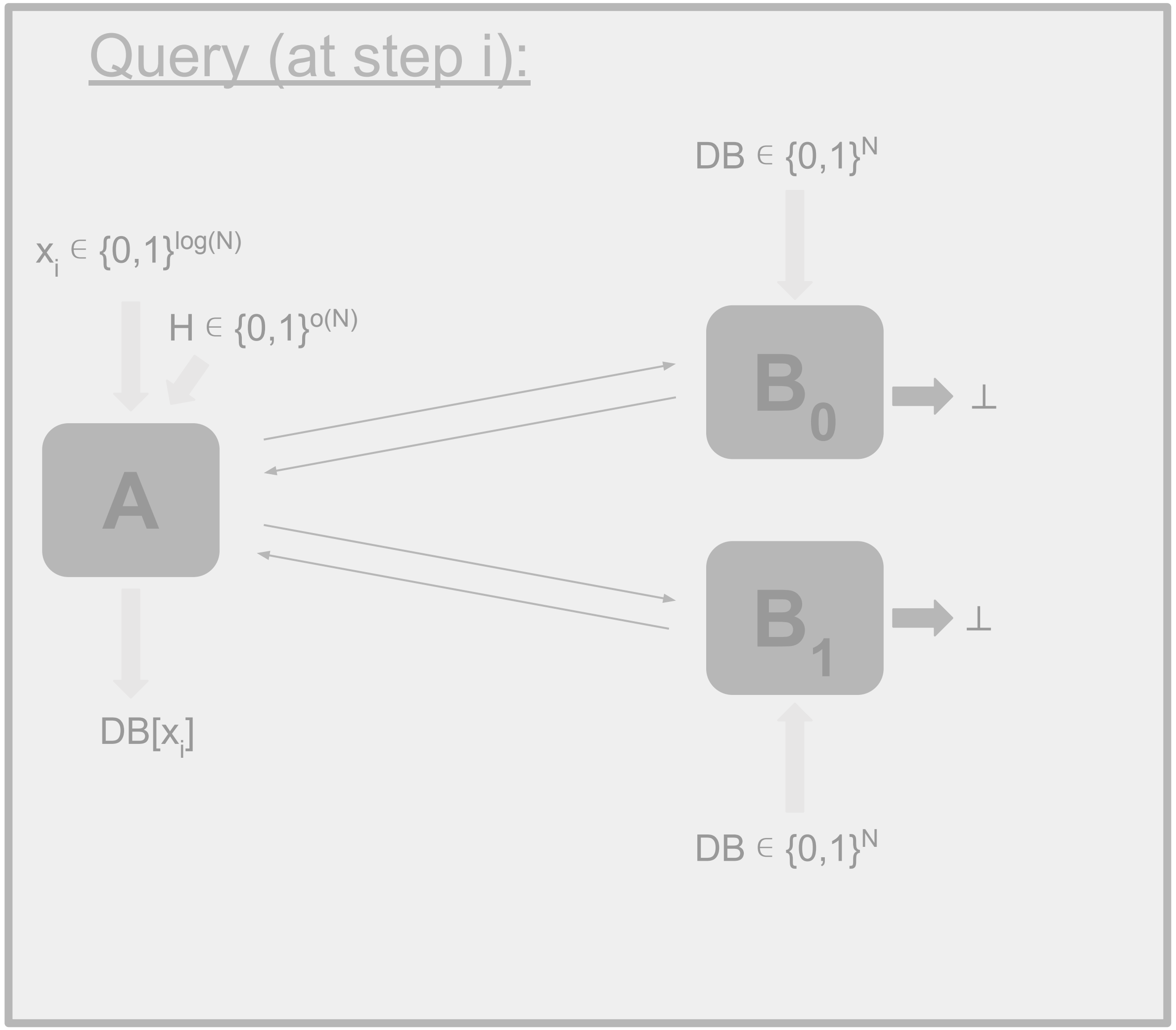
Two-Server PIR, Client Preprocessing [BIM ‘04, …, CK ‘20, ...]



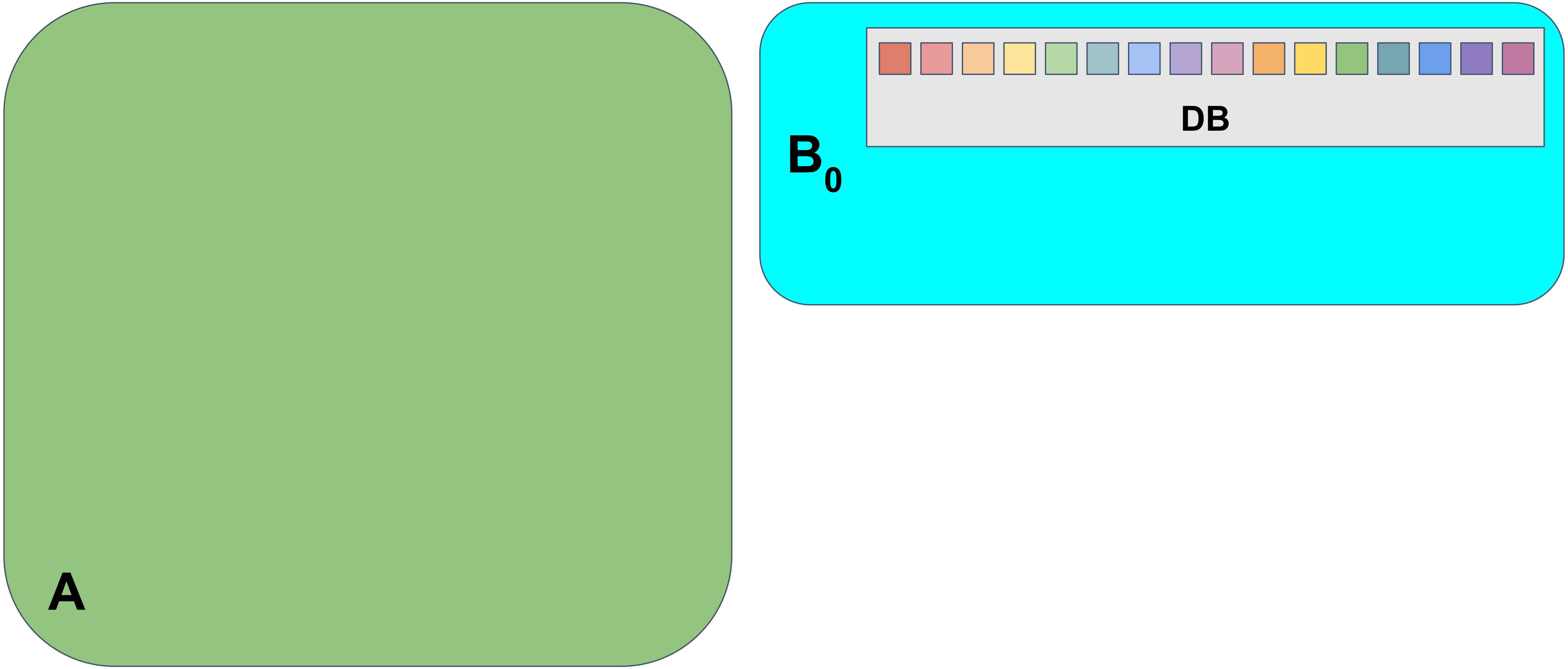


Two-Server PIR, Client Preprocessing [BIM ‘04, …, CK ‘20, ...]

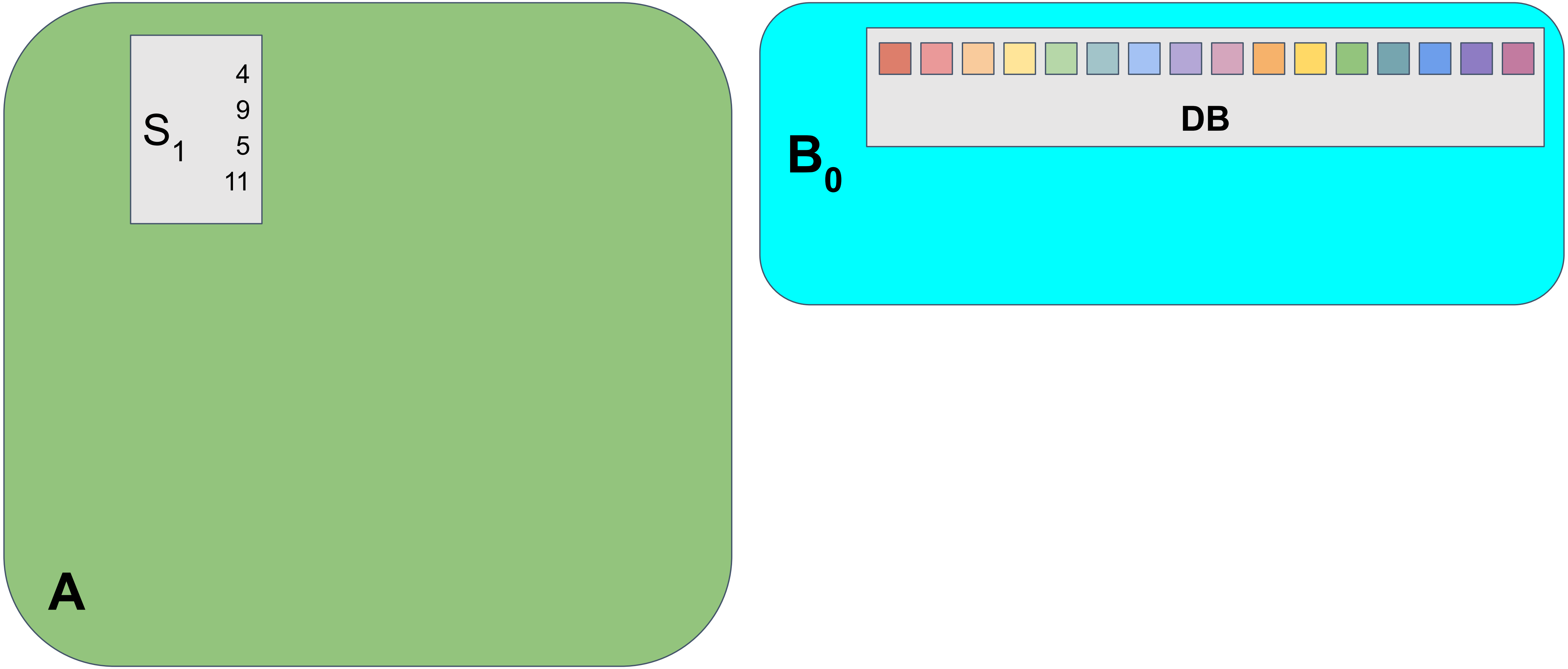




The Client’s Hint [CK ‘20, KC ‘21, …]



The Client’s Hint [CK ‘20, KC ‘21, …]



S_1

4

9

5

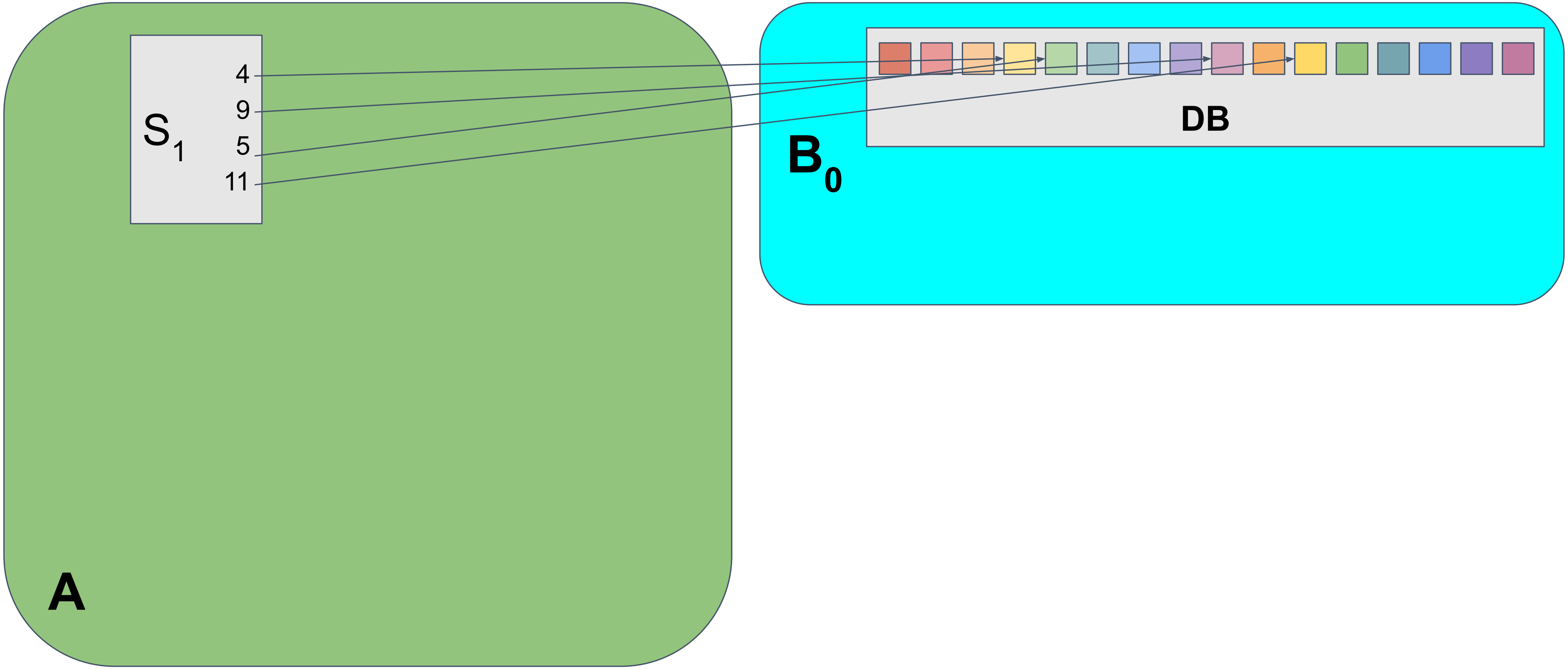
11

A

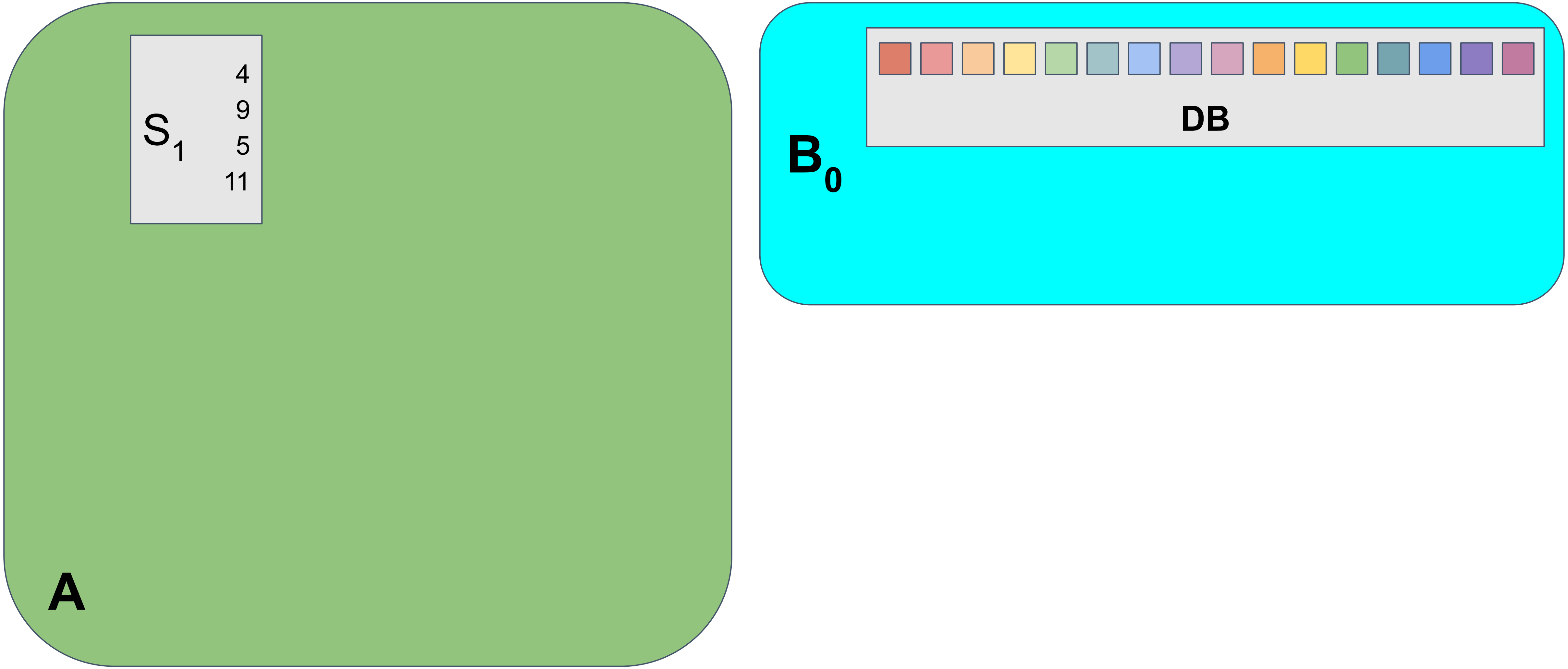
B_0

DB

The Client’s Hint [CK ‘20, KC ‘21, …]



The Client’s Hint [CK ‘20, KC ‘21, …]



S_1

4

9

5

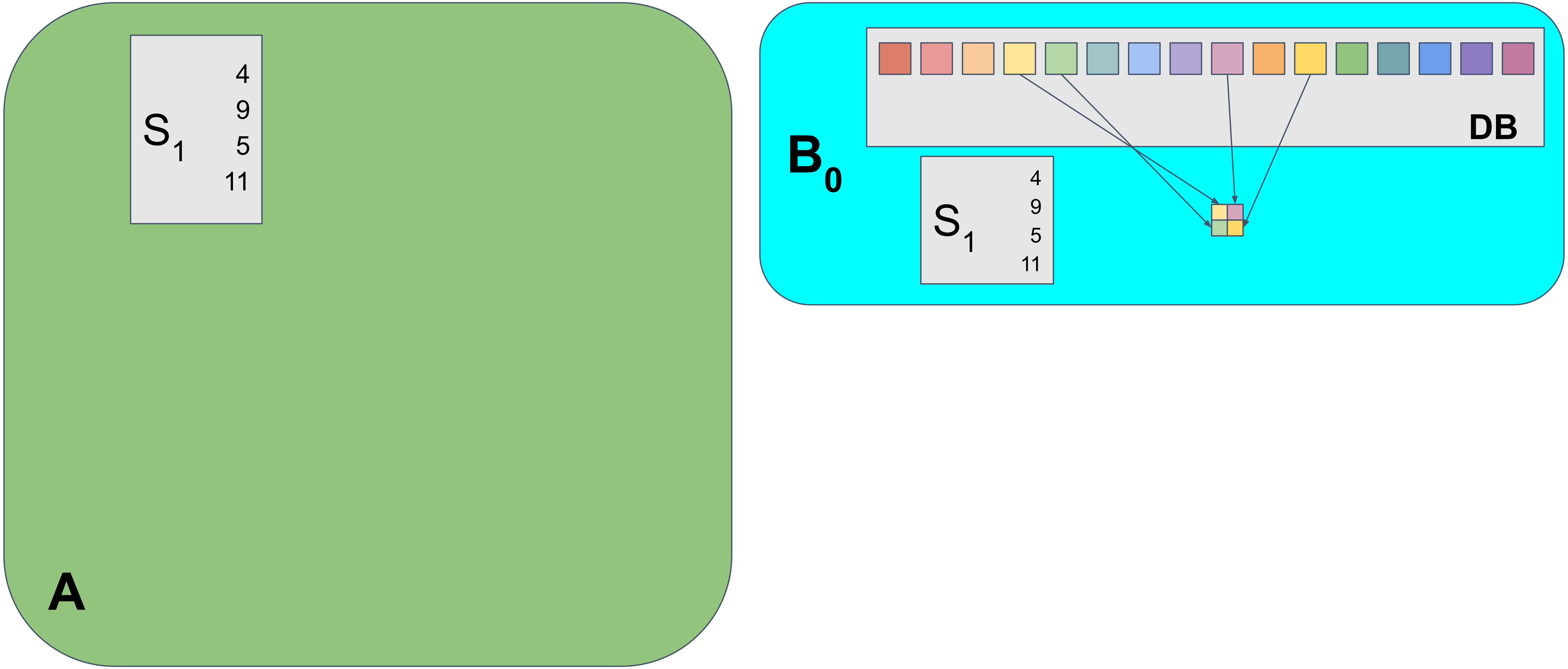
11

A

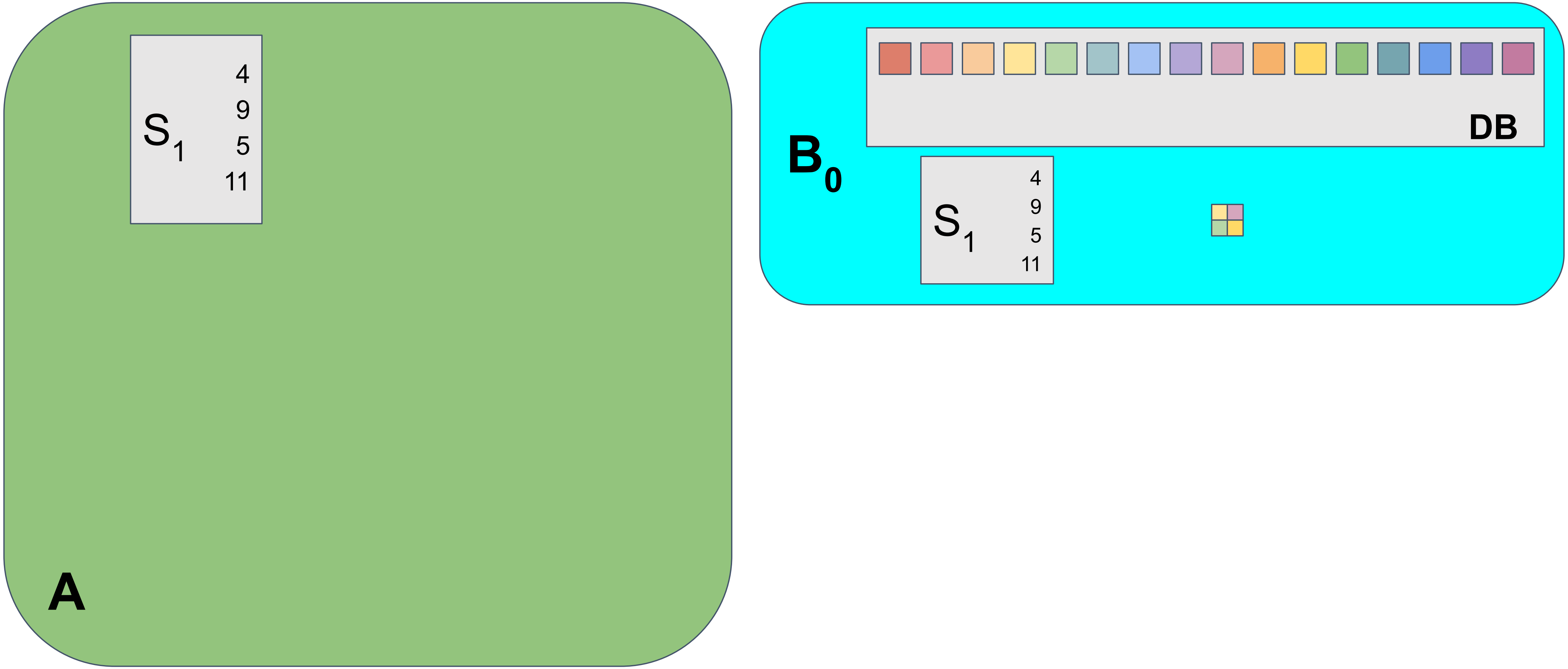
B_0

DB

The Client’s Hint [CK ‘20, KC ‘21, …]



The Client’s Hint [CK ‘20, KC ‘21, …]



S_1 4 9 5 11

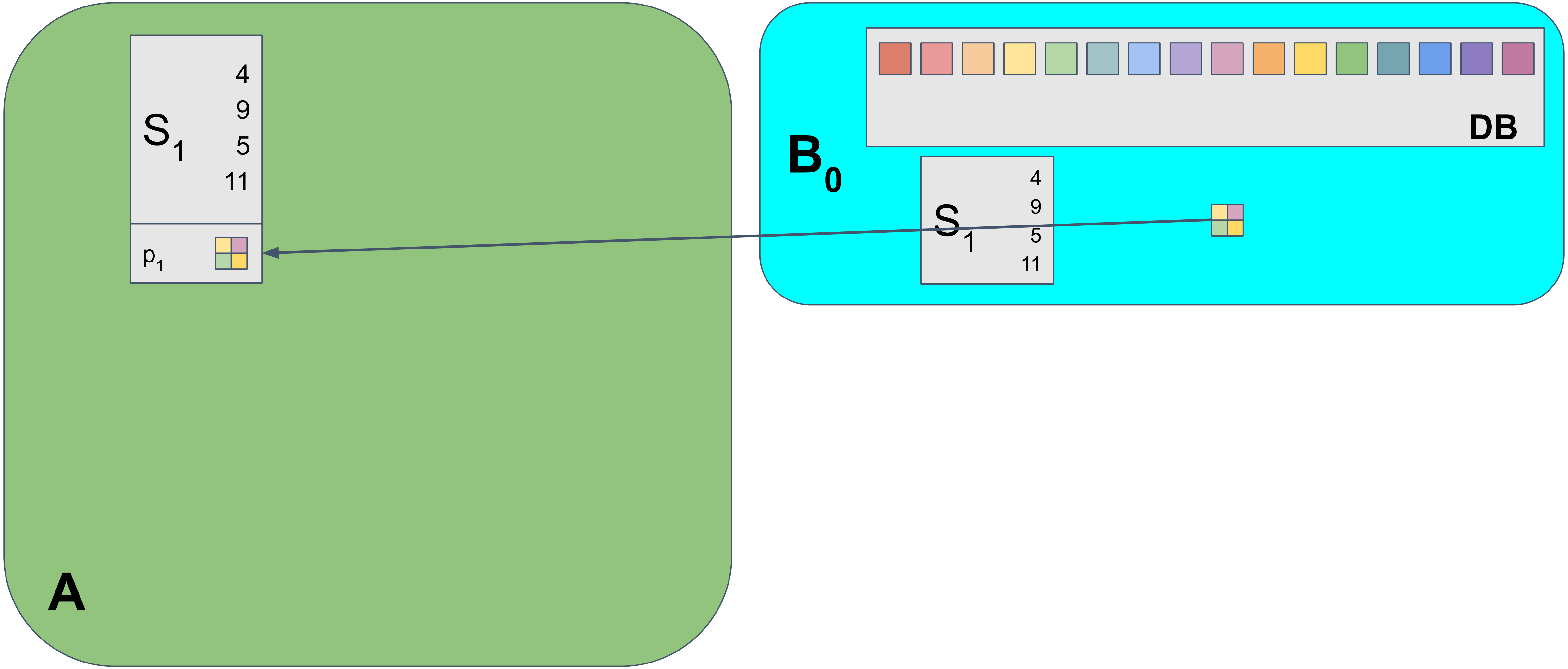
A

B_0

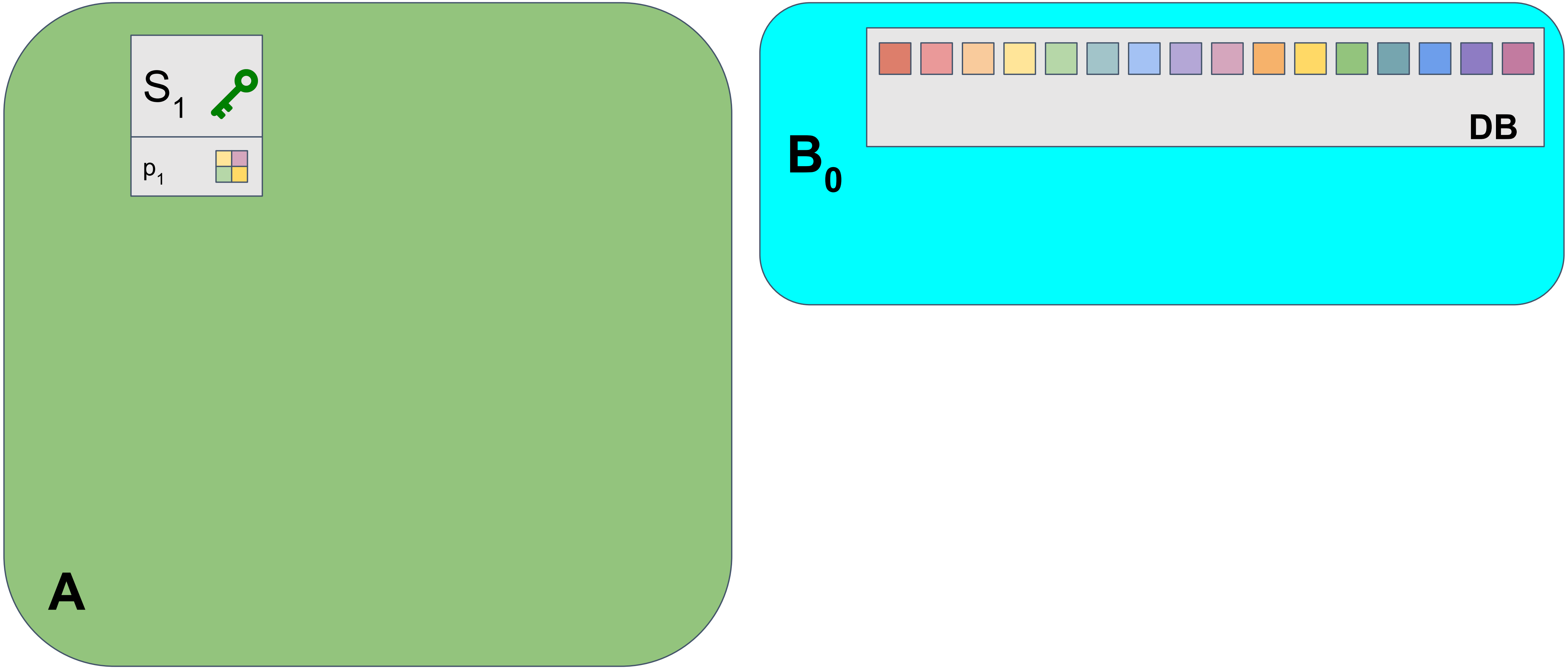
DB

S_1 4 9 5 11

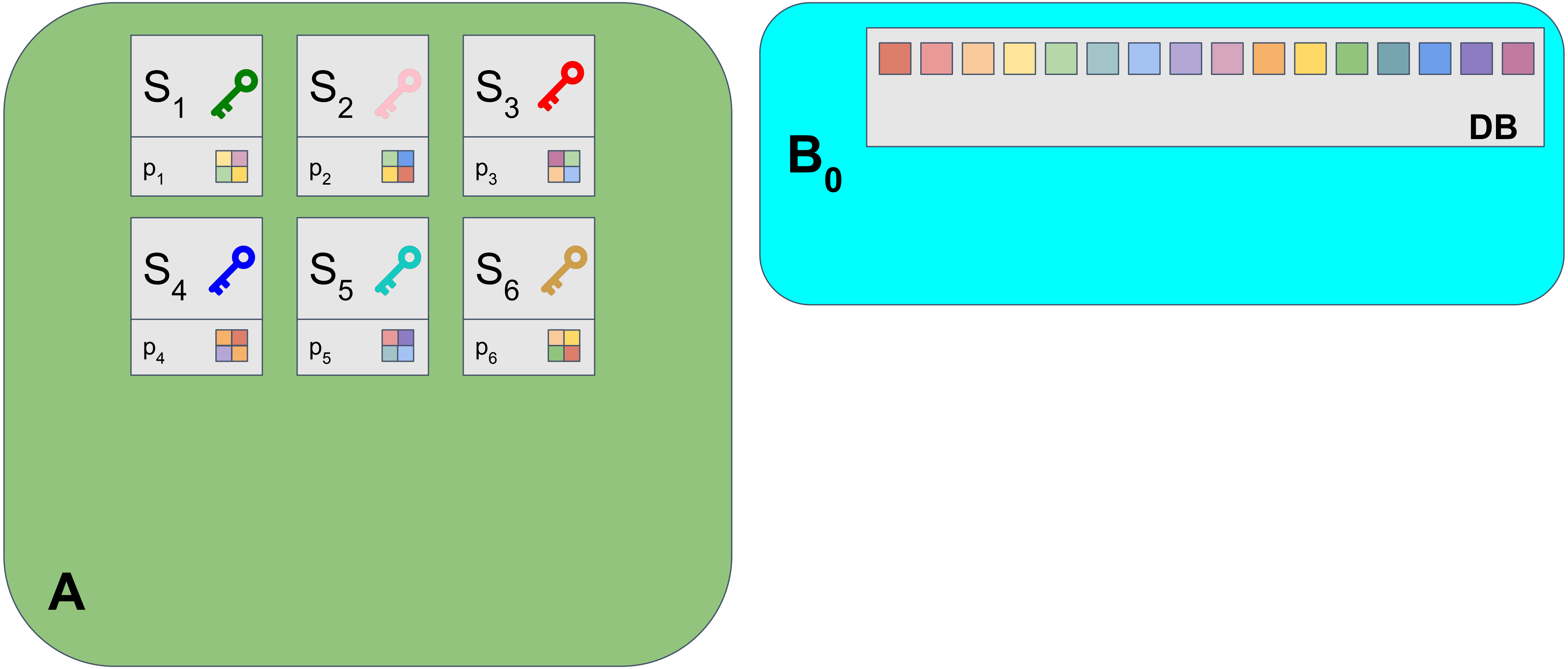
The Client’s Hint [CK ‘20, KC ‘21, …]



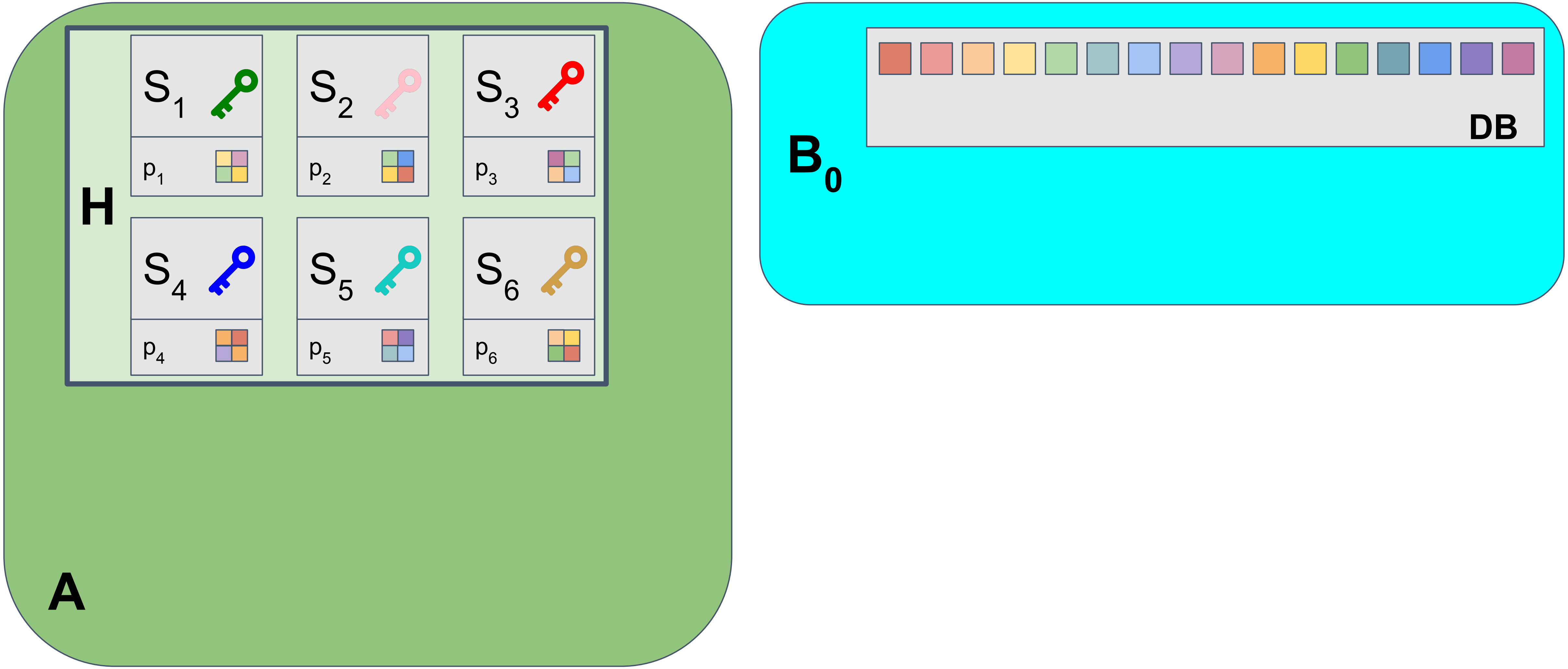
The Client's Hint [CK '20, KC '21, …]



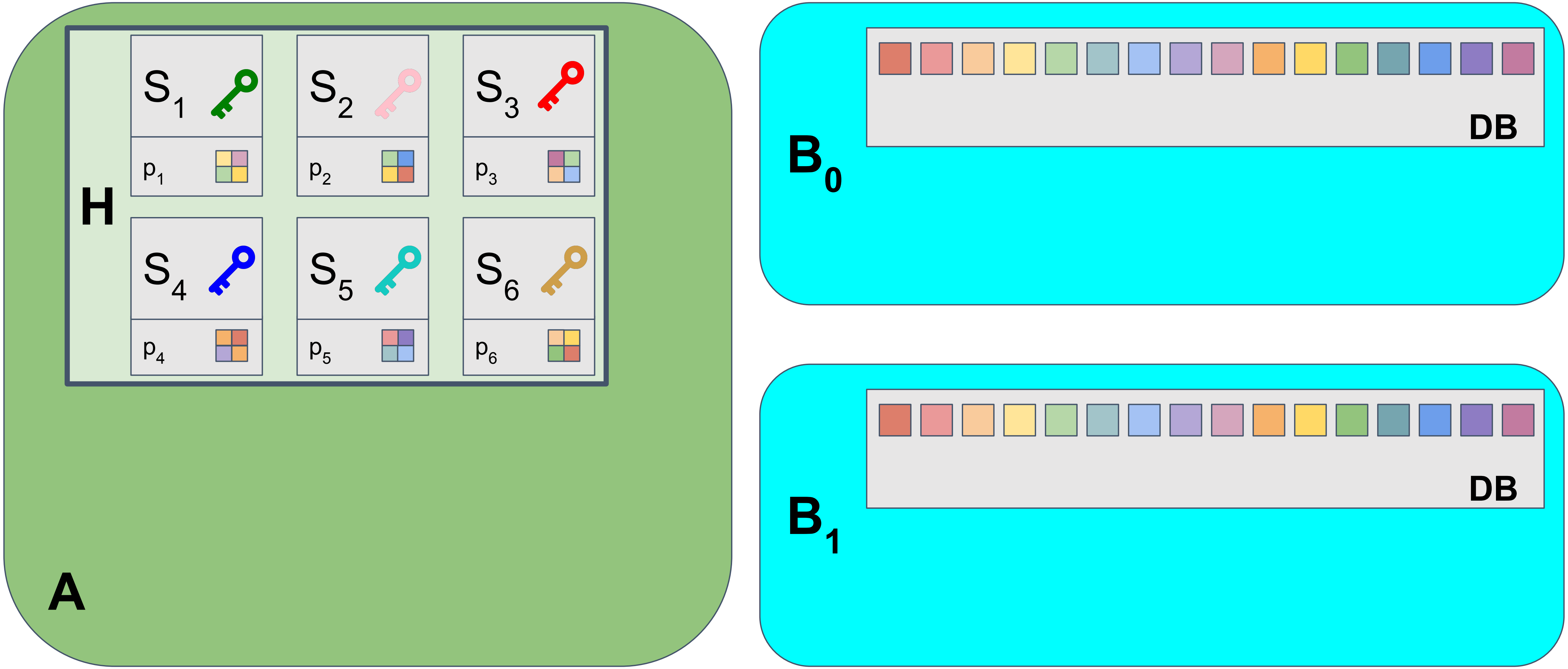
The Client’s Hint [CK ‘20, KC ‘21, …]



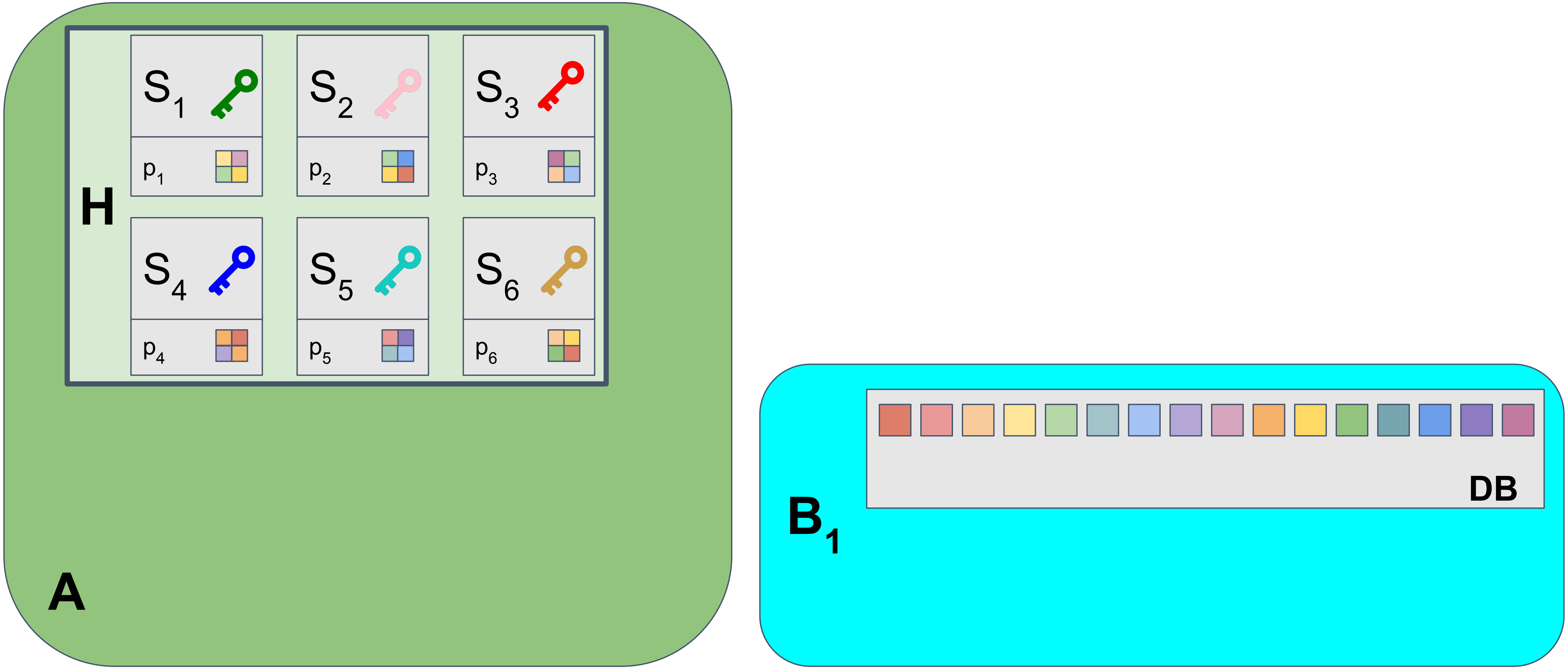
The Client’s Hint [CK ‘20, KC ‘21, …]



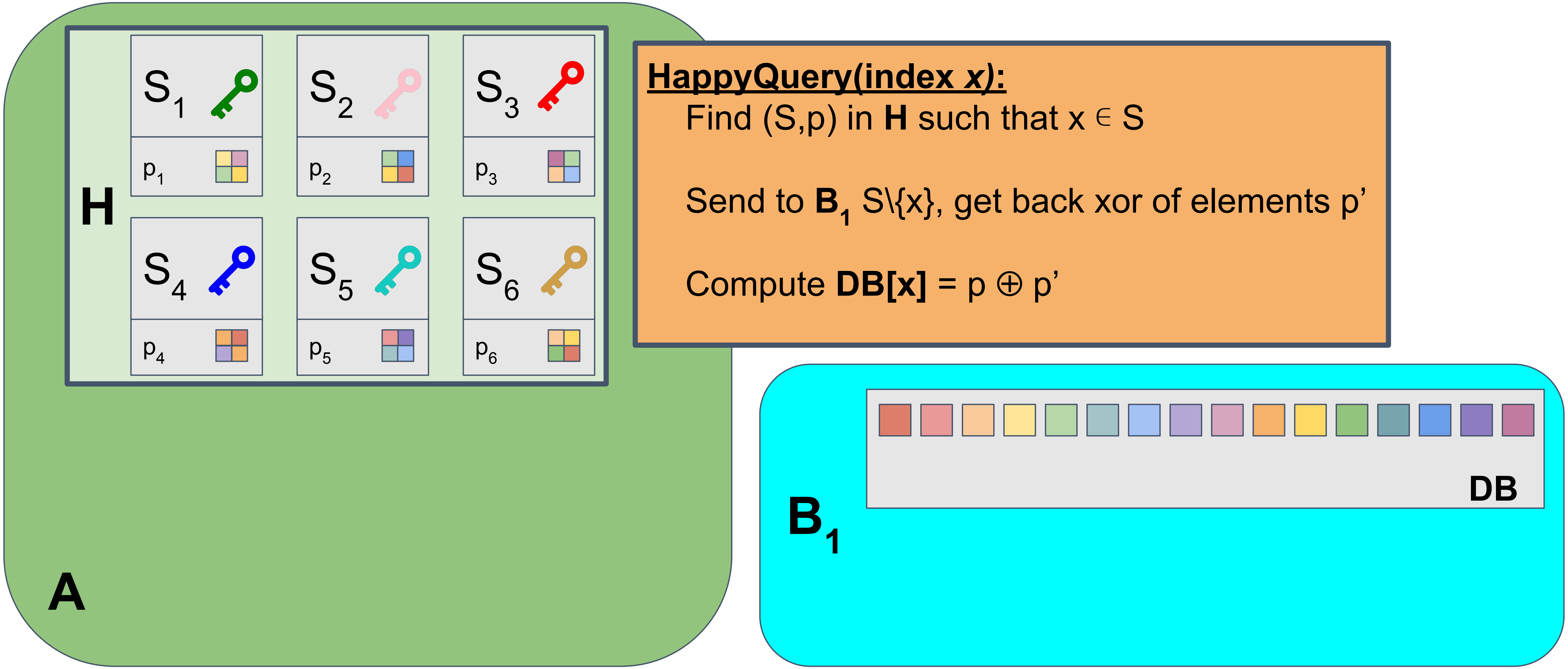
The Client’s Hint [CK ‘20, KC ‘21, …]



The Client's Hint [CK '20, KC '21, …]



Query Outline

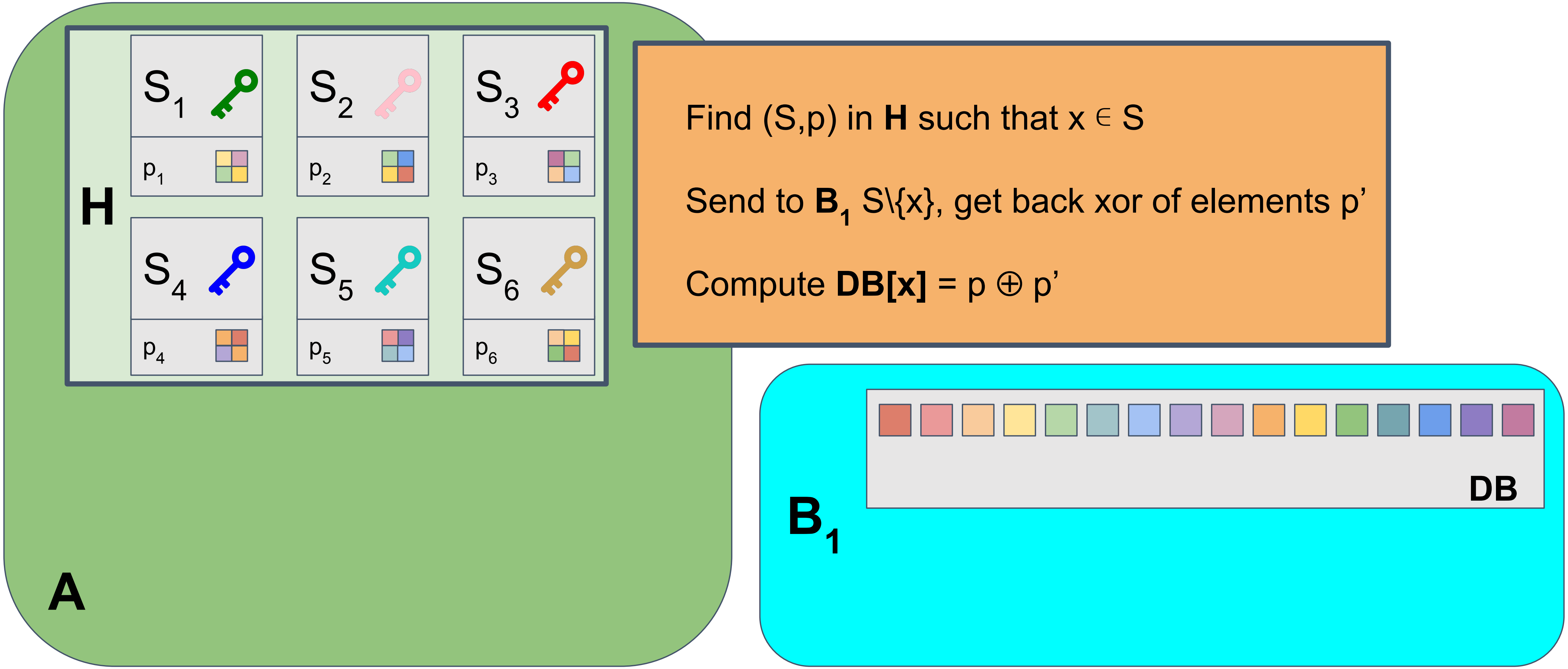


```
HappyQuery(index x):
    Find (S,p) in H such that $x \in S$

    Send to B$_1$ S\{x}, get back xor of elements p'

    Compute DB[x] = $p \oplus p'$
```

Query Outline

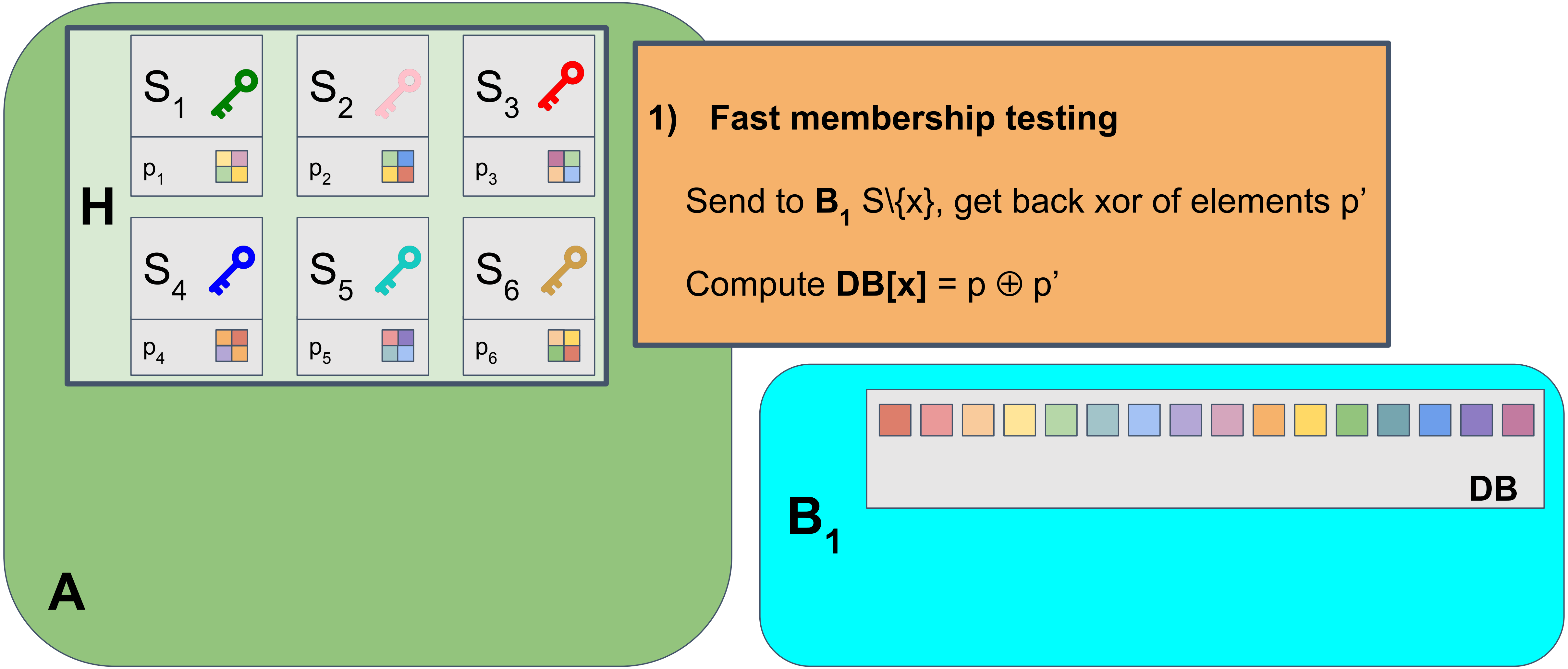


Find (S,p) in **H** such that x ∊ S

Send to **B_1** S\{x}, get back xor of elements p’

Compute **DB[x]** = p ⊕ p’

Query Requirements

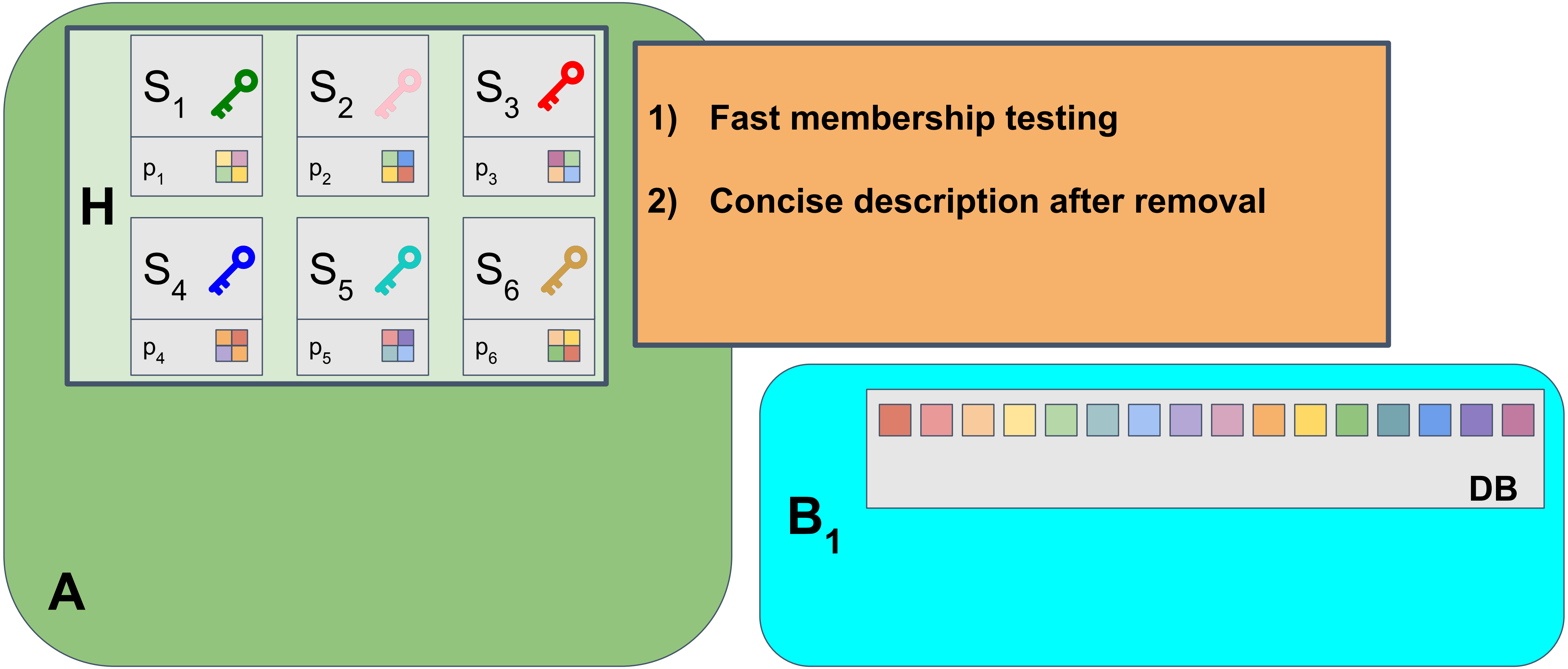


1) **Fast membership testing**

Send to **B_1** $S\setminus\{x\}$, get back xor of elements p'

Compute **DB[x]** = $p \oplus p'$

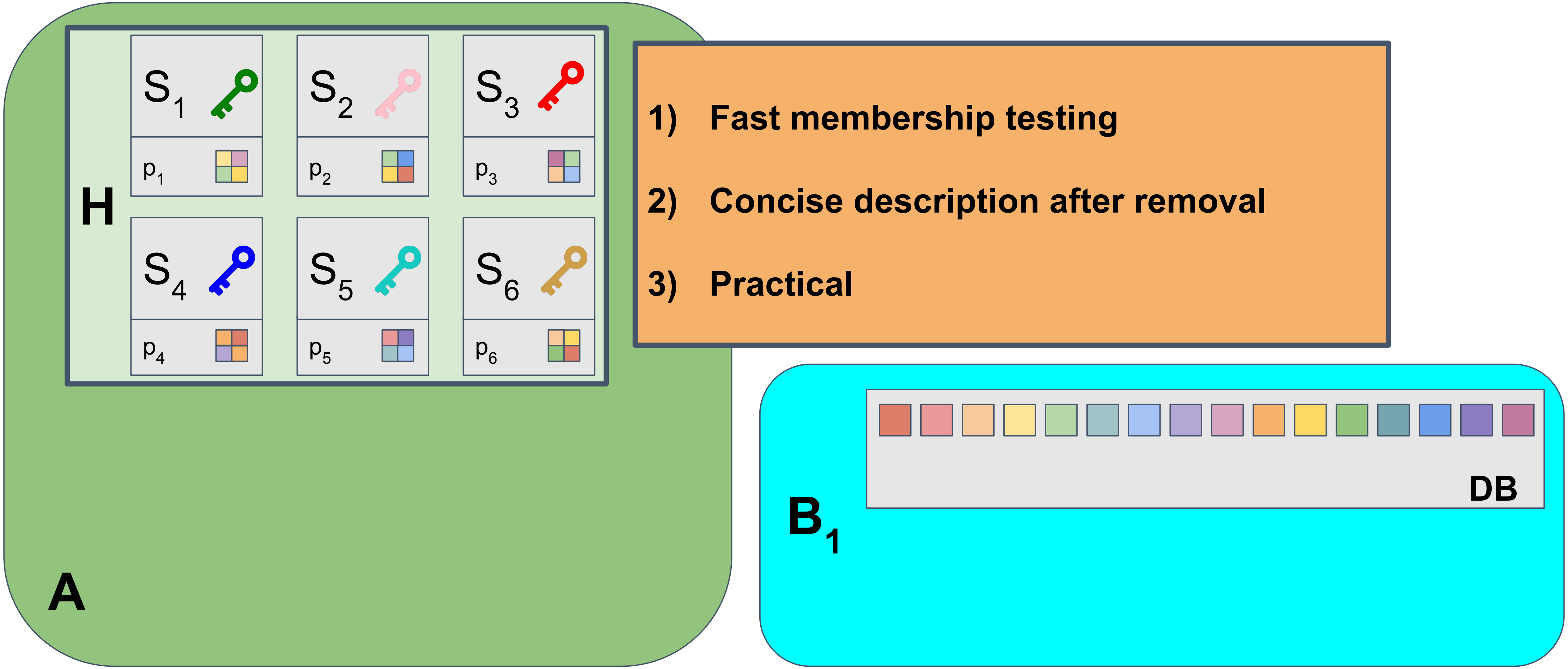
Query Requirements



1) **Fast membership testing**

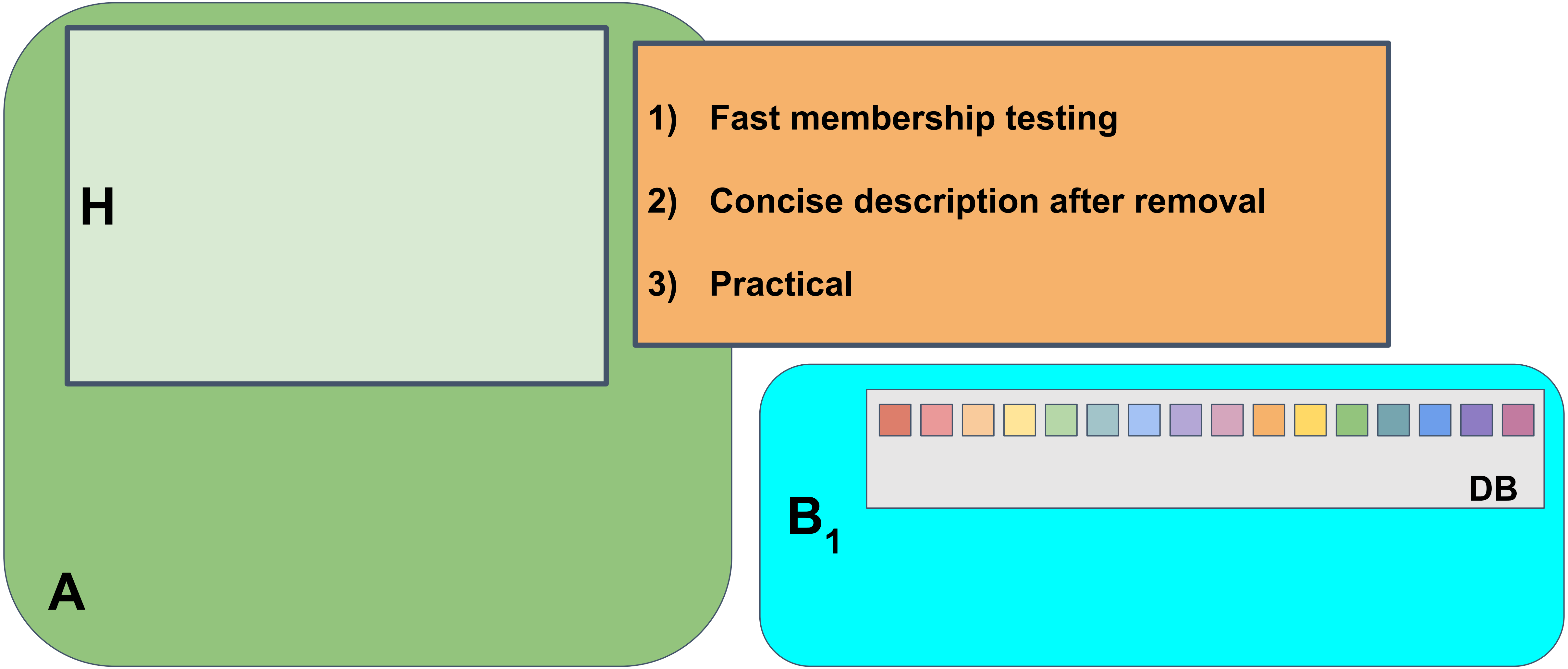
2) **Concise description after removal**

Query Requirements



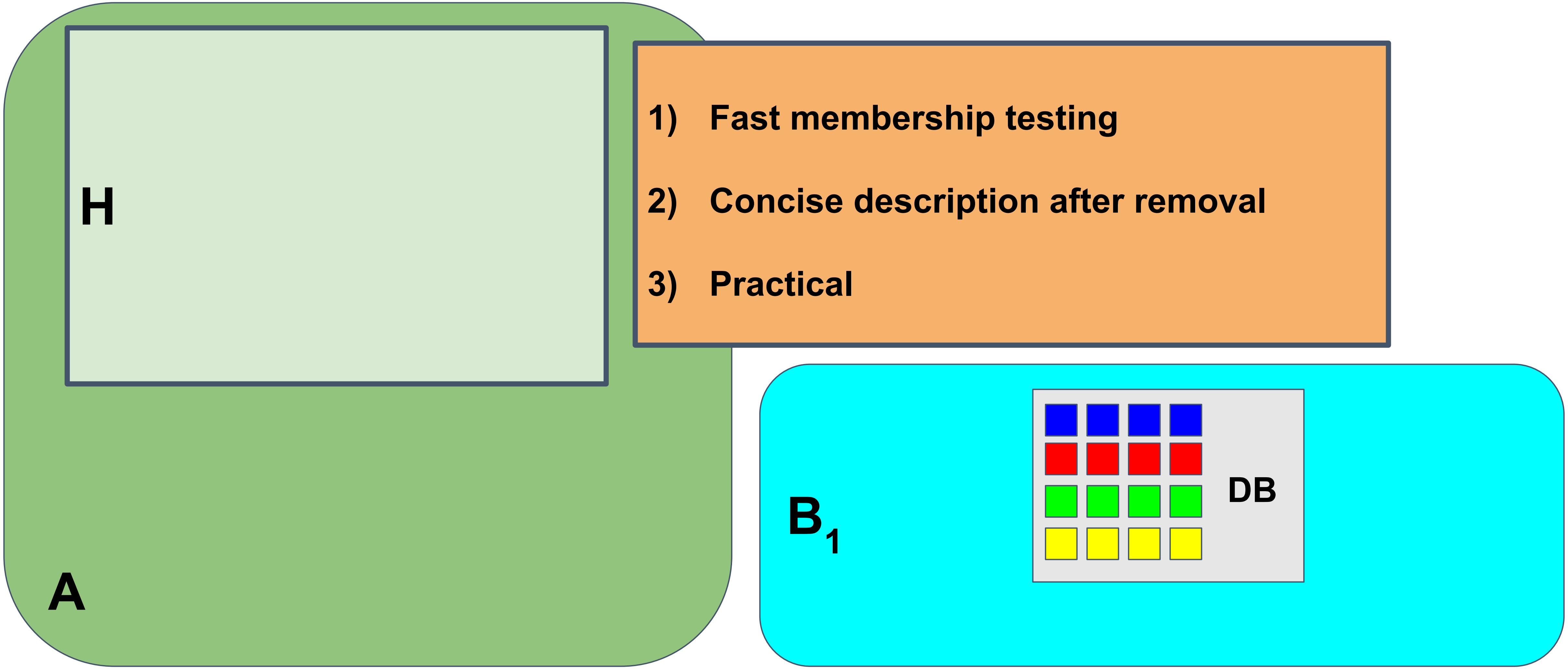
1) **Fast membership testing**
2) **Concise description after removal**
3) **Practical**

Modifying Client's Hint



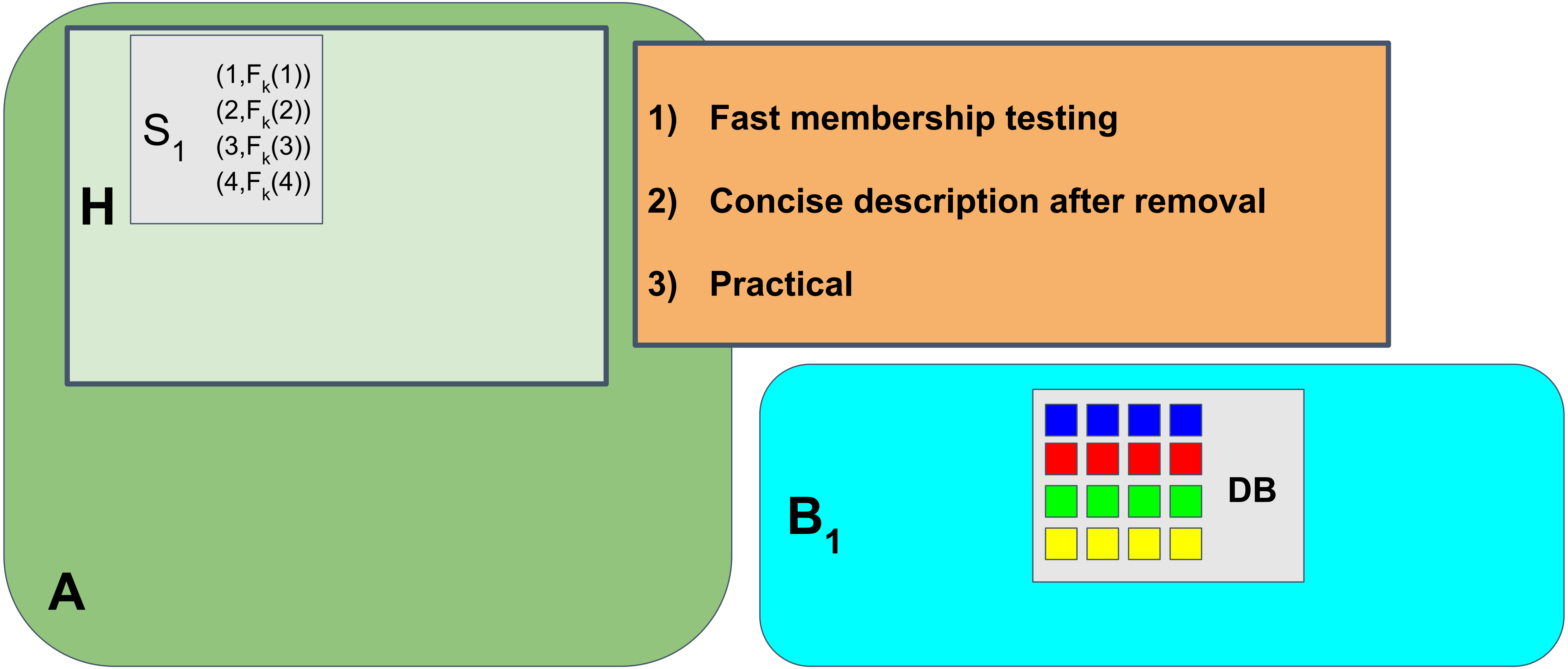
1) Fast membership testing
2) Concise description after removal
3) Practical

Modifying Client’s Hint



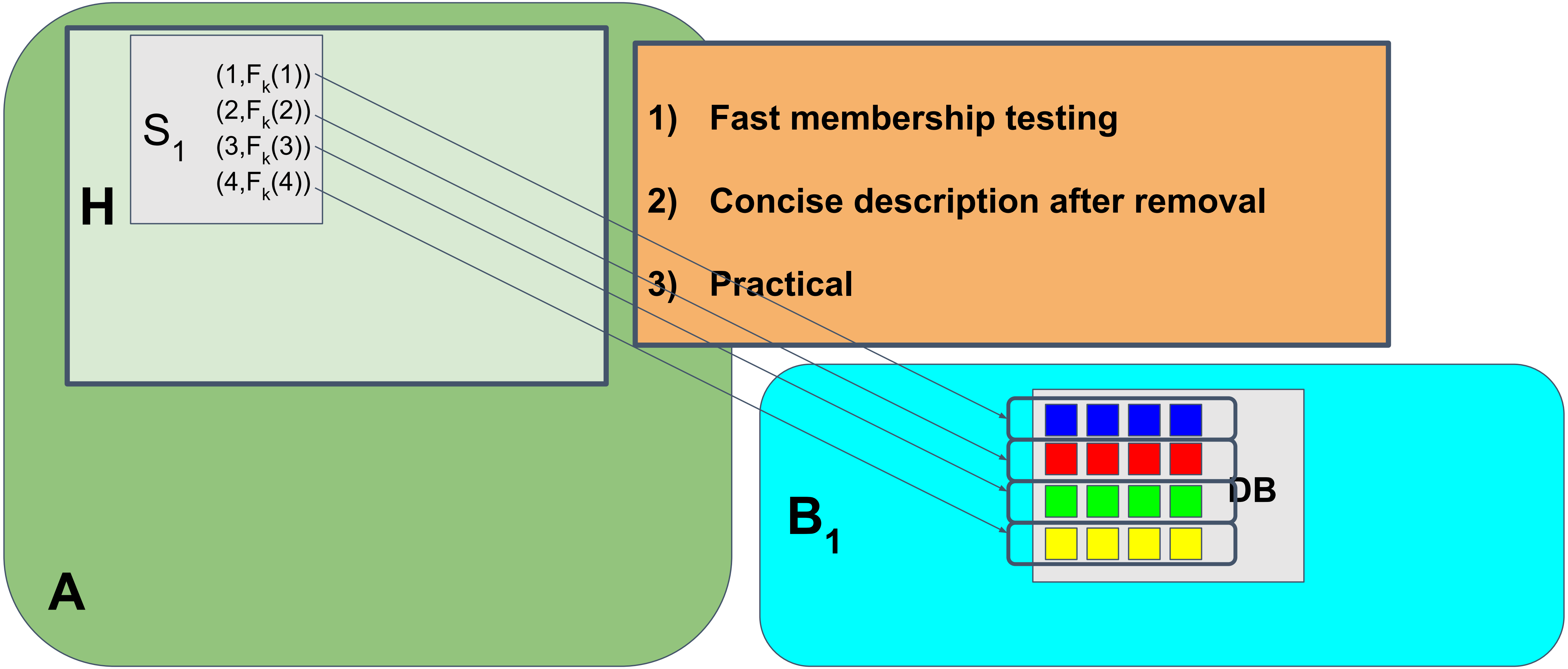
1) Fast membership testing
2) Concise description after removal
3) Practical

Modifying Client’s Hint



1) **Fast membership testing**
2) **Concise description after removal**
3) **Practical**

Modifying Client’s Hint

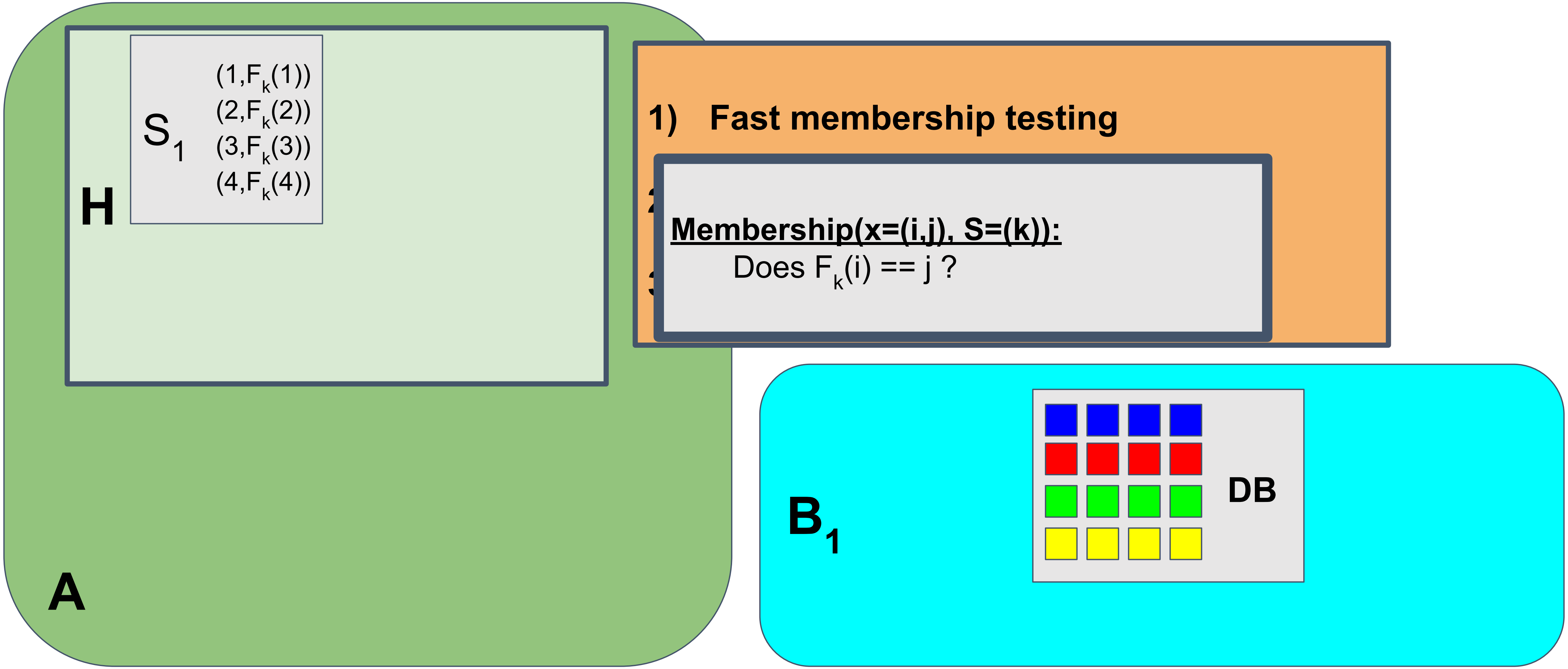


1) **Fast membership testing**

2) **Concise description after removal**

3) **Practical**

Modifying Client’s Hint

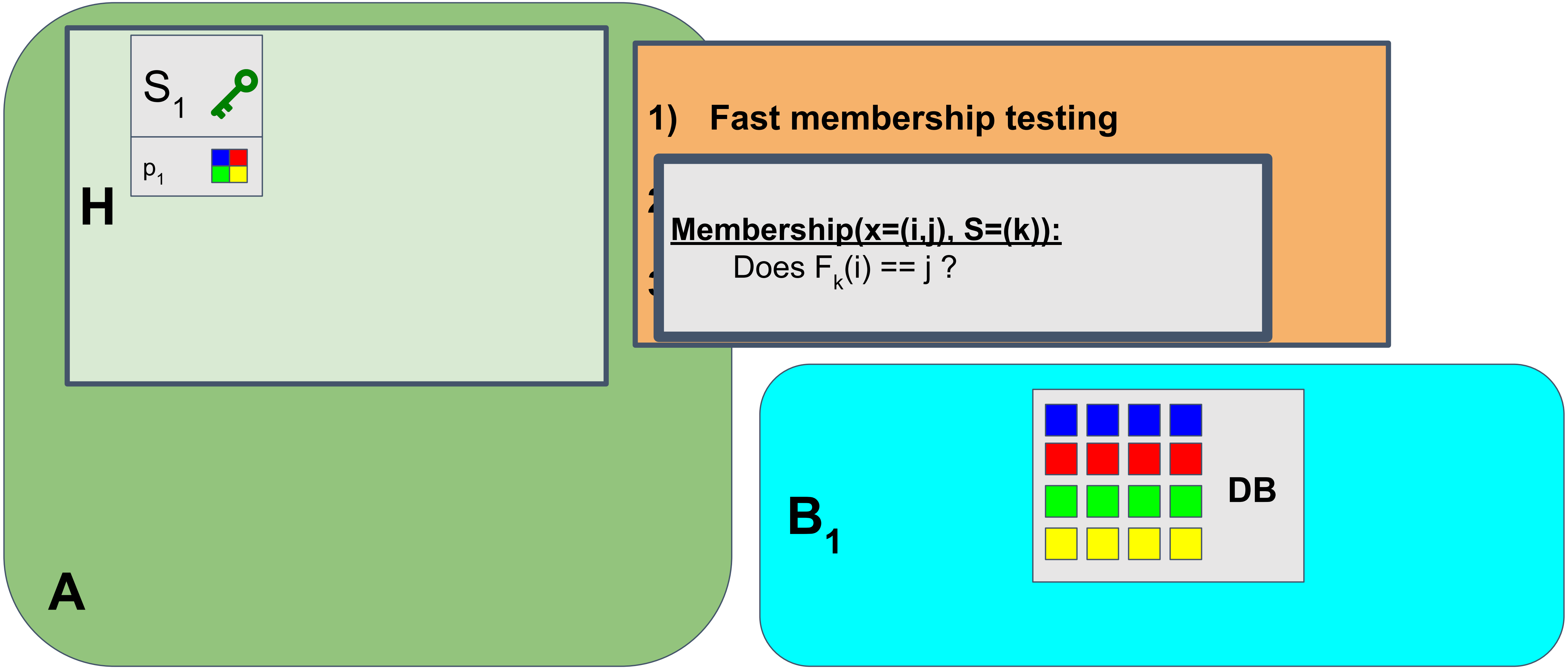


1) **Fast membership testing**

<u>Membership(x=(i,j), S=(k)):</u>

Does $F_k(i) == j$?

Modifying Client’s Hint

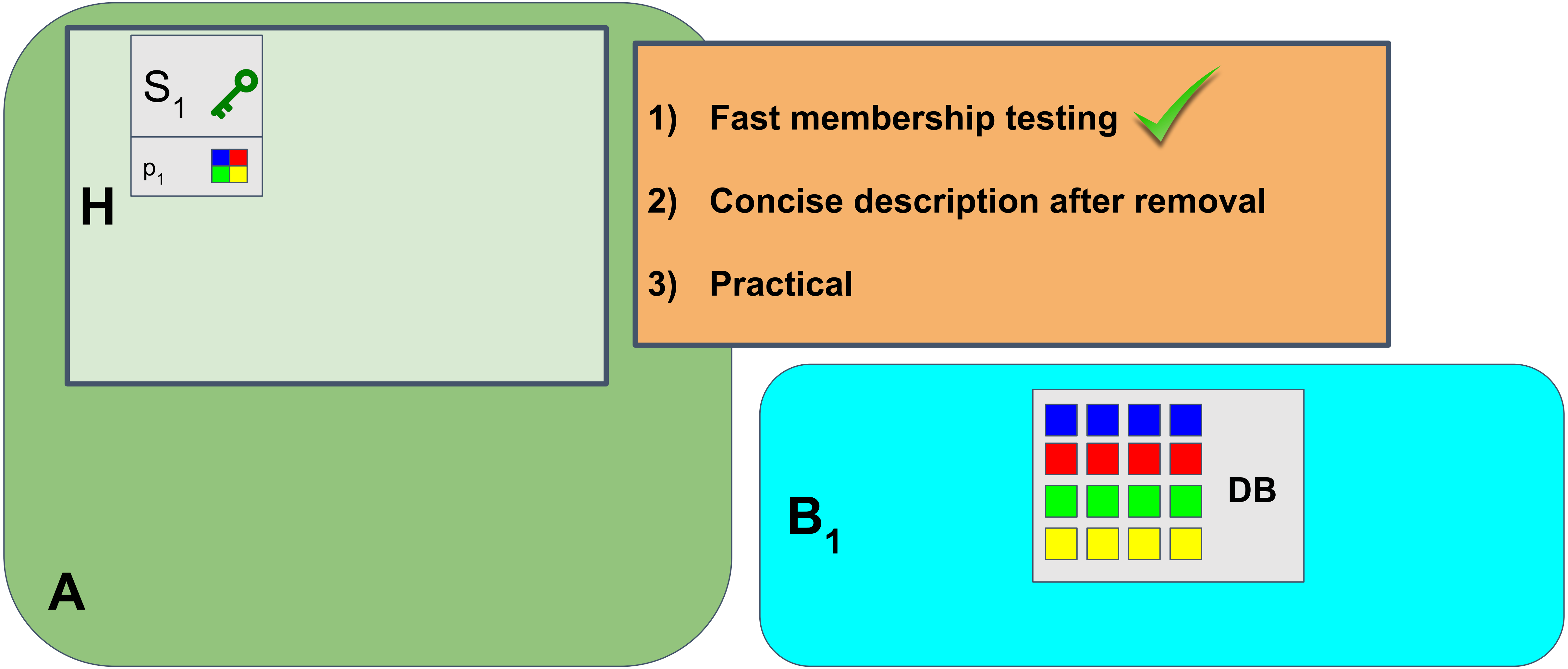


1) Fast membership testing

Membership(x=(i,j), S=(k)):

Does $F_k(i)$ == j ?

Modifying Client’s Hint

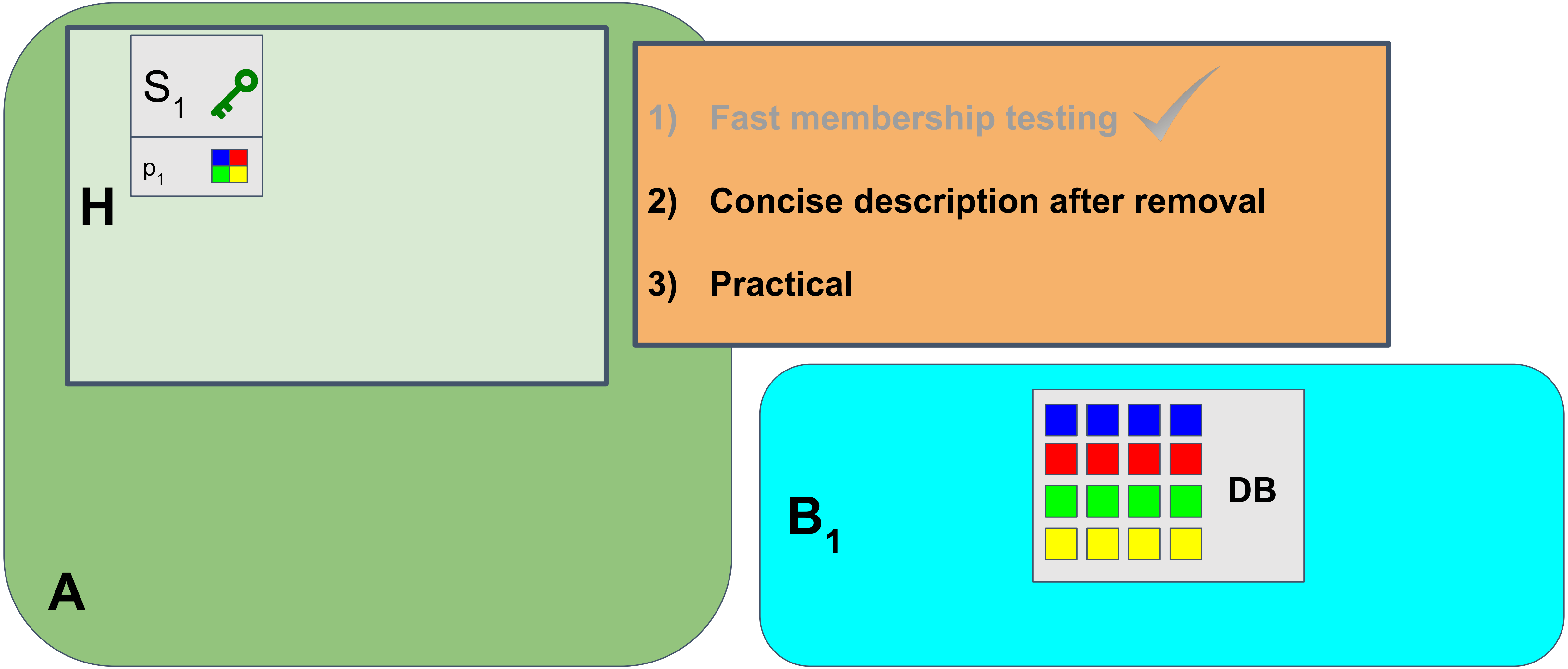


1) **Fast membership testing** ✓

2) **Concise description after removal**

3) **Practical**

Modifying Client's Hint

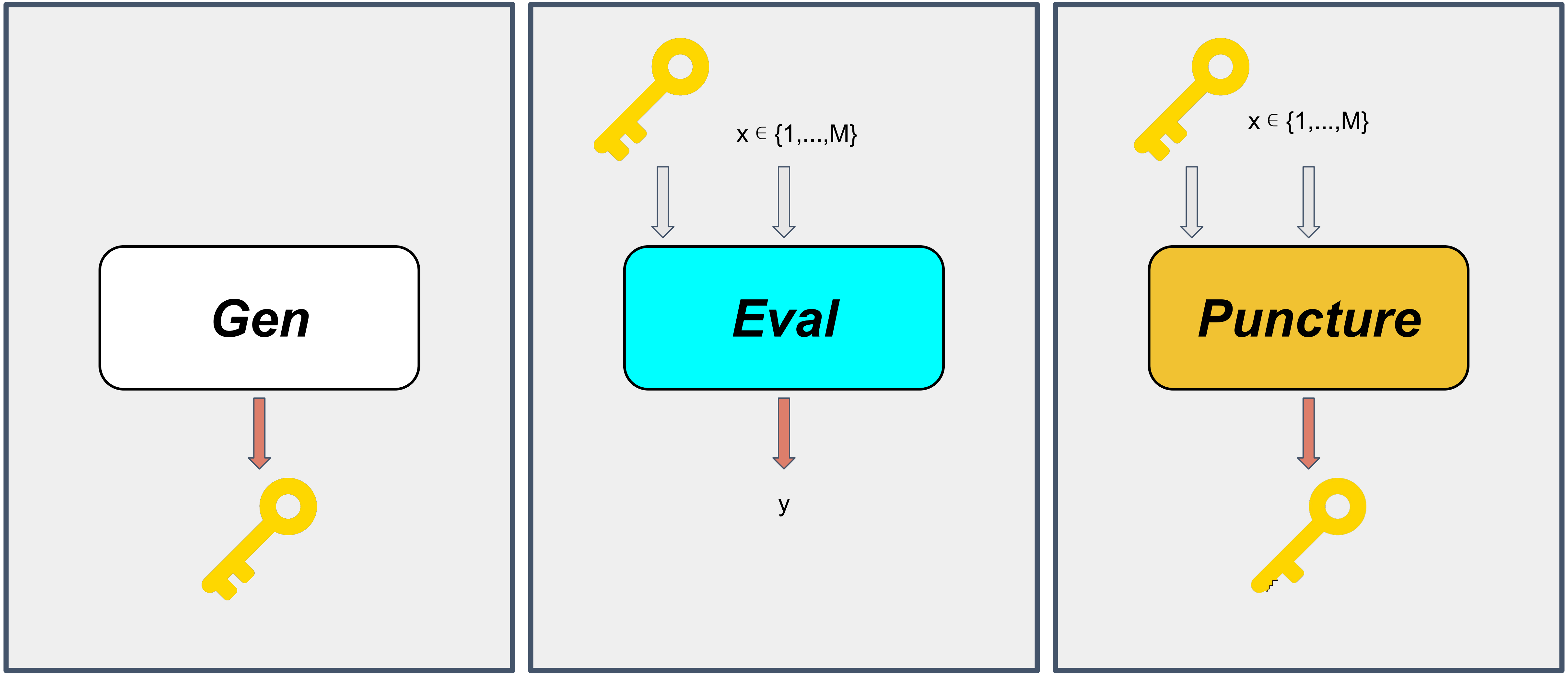


1) Fast membership testing ✓

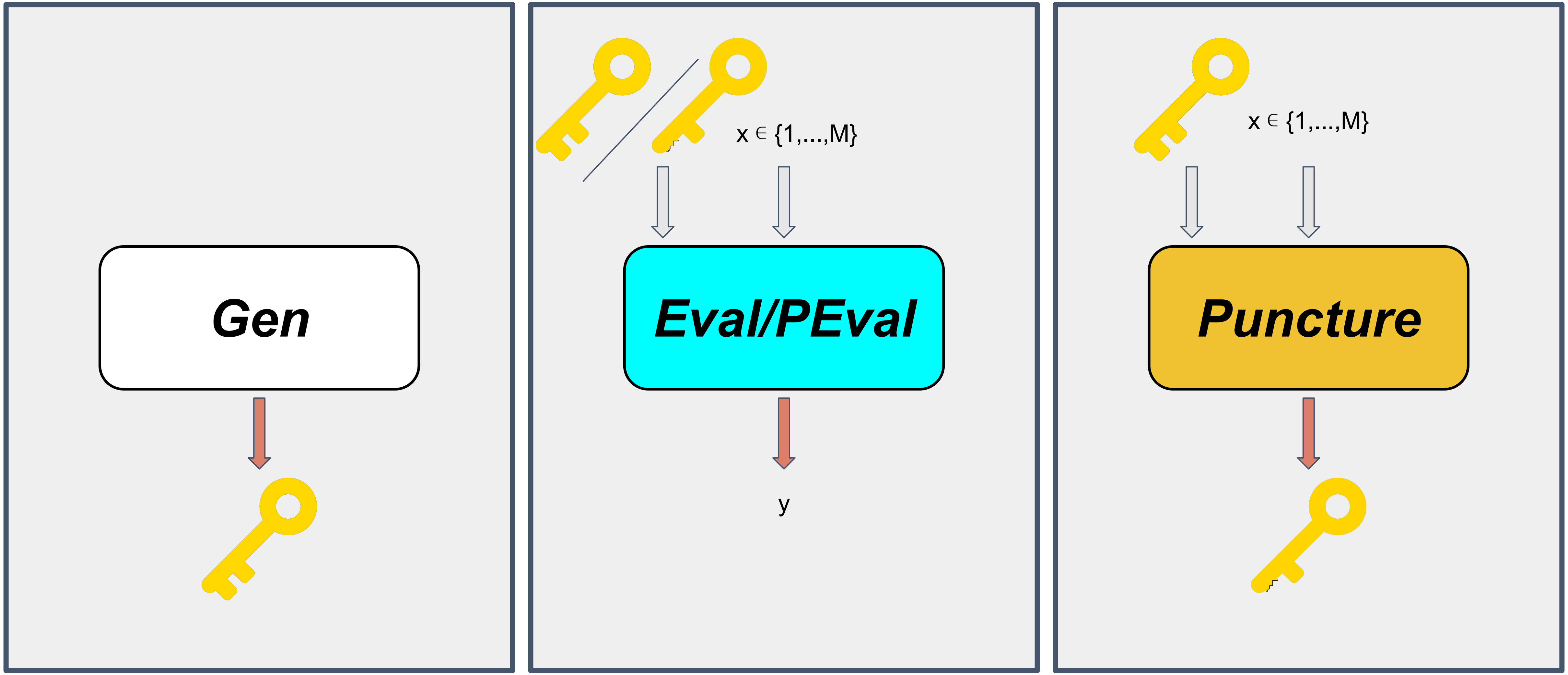
2) Concise description after removal

3) Practical

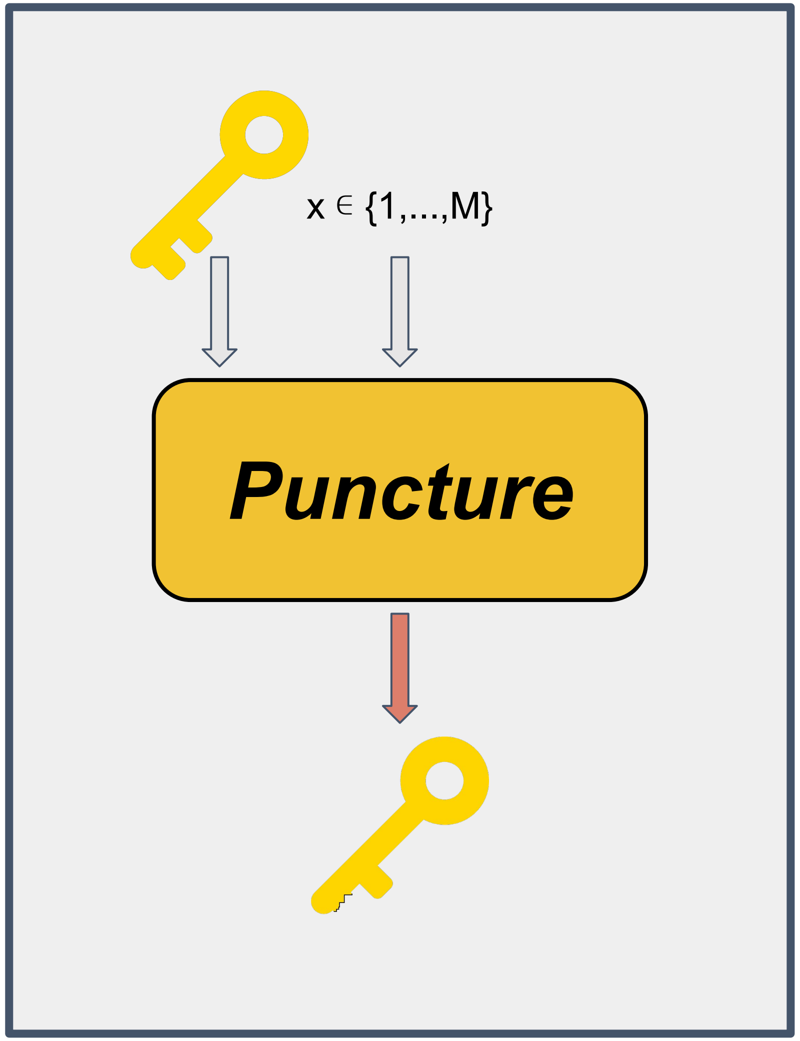
Tool: Puncturable Pseudorandom Function [GGM ‘84, BW ‘13, KPTZ ‘13, BGI ‘14, …]



Tool: Puncturable Pseudorandom Function [GGM ‘84, BW ‘13, KPTZ ‘13, BGI ‘14, …]



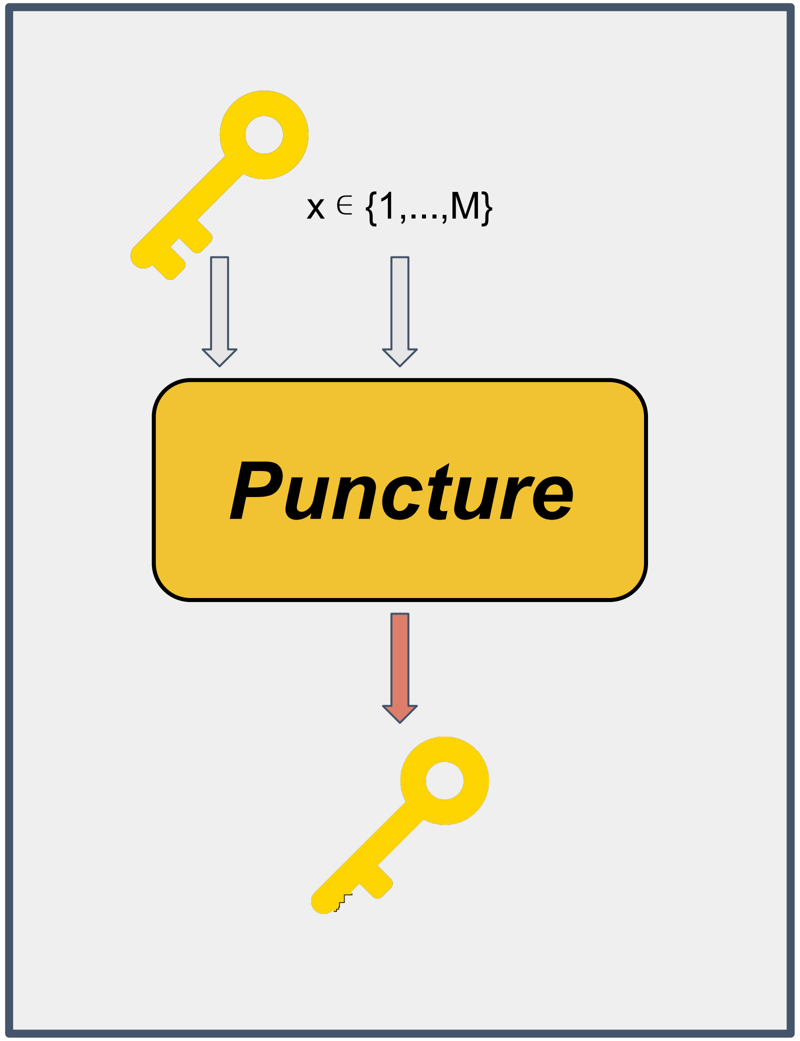
Tool: Puncturable Pseudorandom Function [GGM ‘84, BW ‘13, KPTZ ‘13, BGI ‘14, …]



Correctness:

For any input x’ ≠ x, punctured key evaluates to same output as original key

Puncture on Puncturable PRF [BW ‘13, KPTZ ‘13, BGI ‘14, …]



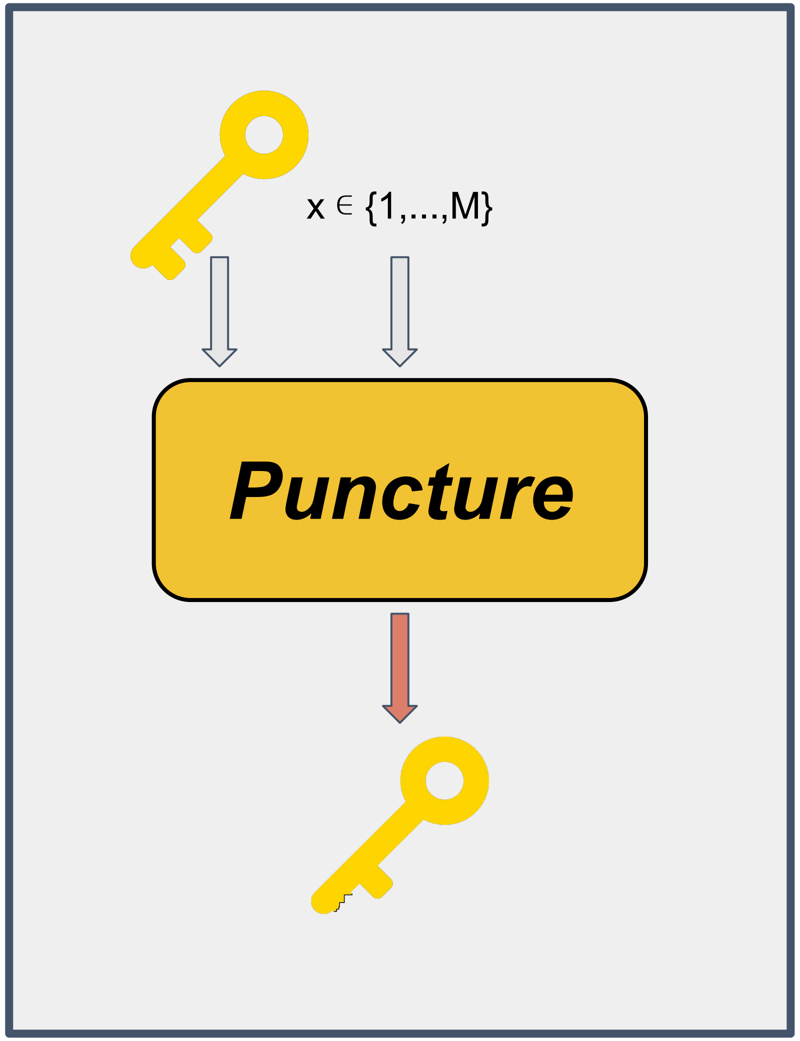
Correctness:

For any input x’ ≠ x, punctured key evaluates to same output as original key

Security:

New key contains no information about evaluation at punctured point x

Puncture on Puncturable PRF [BW ‘13, KPTZ ‘13, BGI ‘14, …]



Correctness:

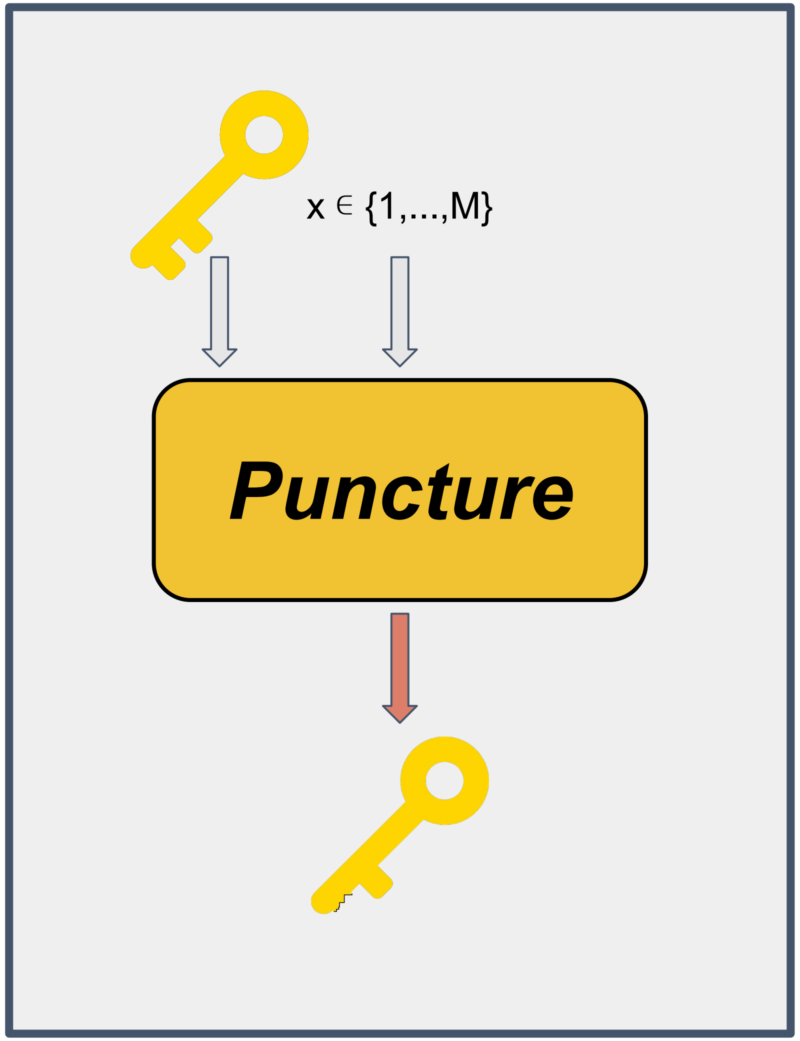
For any input x’ ≠ x, punctured key evaluates to same output as original key

Security:

New key contains no information about evaluation at punctured point x

Recall: if x = (i,j) and sets are made of tuples $(i,F_k(i))$, if we send to $\mathbf{B_0}$ k’ ← Puncture(k,i), it hides $F_k(i)$ but *does not hide i*

Puncture on Privately Puncturable PRF [BLW ‘15, BKM ‘17, CC ‘17, ...]



Correctness:

For any input x’ ≠ x, punctured key evaluates to same output as original key

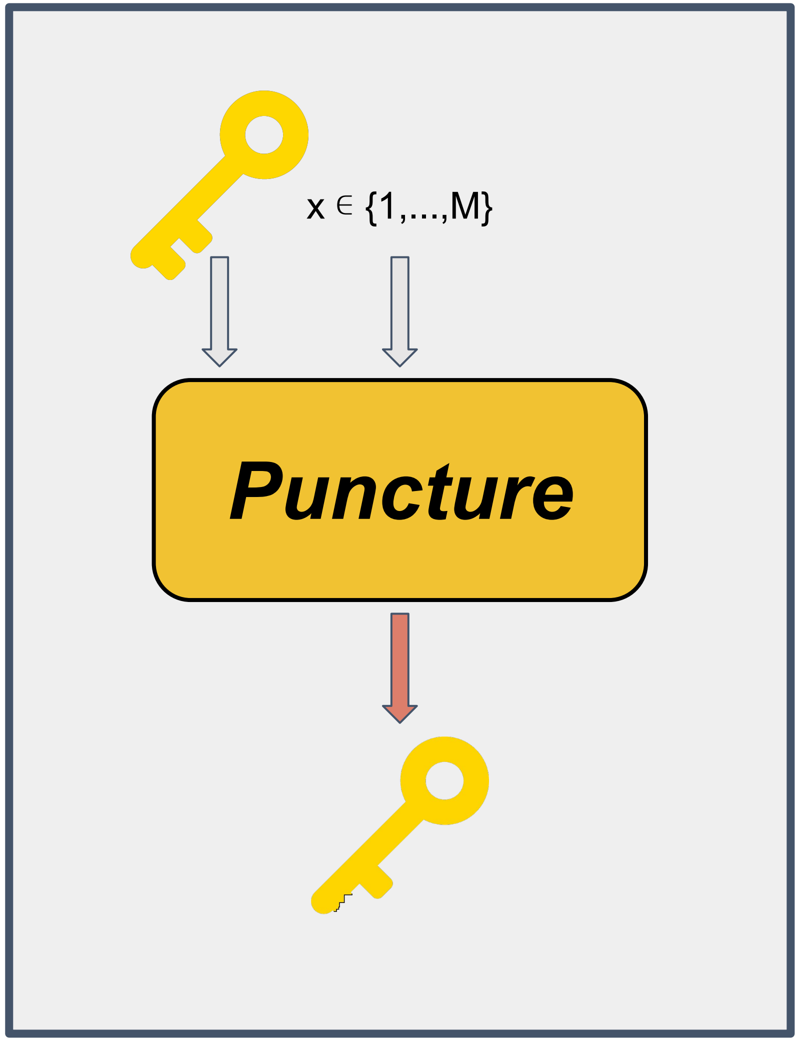
Security:

New key contains no information about evaluation at punctured point x

Privacy:

New key contains no information about punctured point x

Puncture on Privately Puncturable PRF [BLW ‘15, BKM ‘17, CC ‘17, ...]



Correctness:

For any input x’ ≠ x, punctured key evaluates to same output as original key

Security:

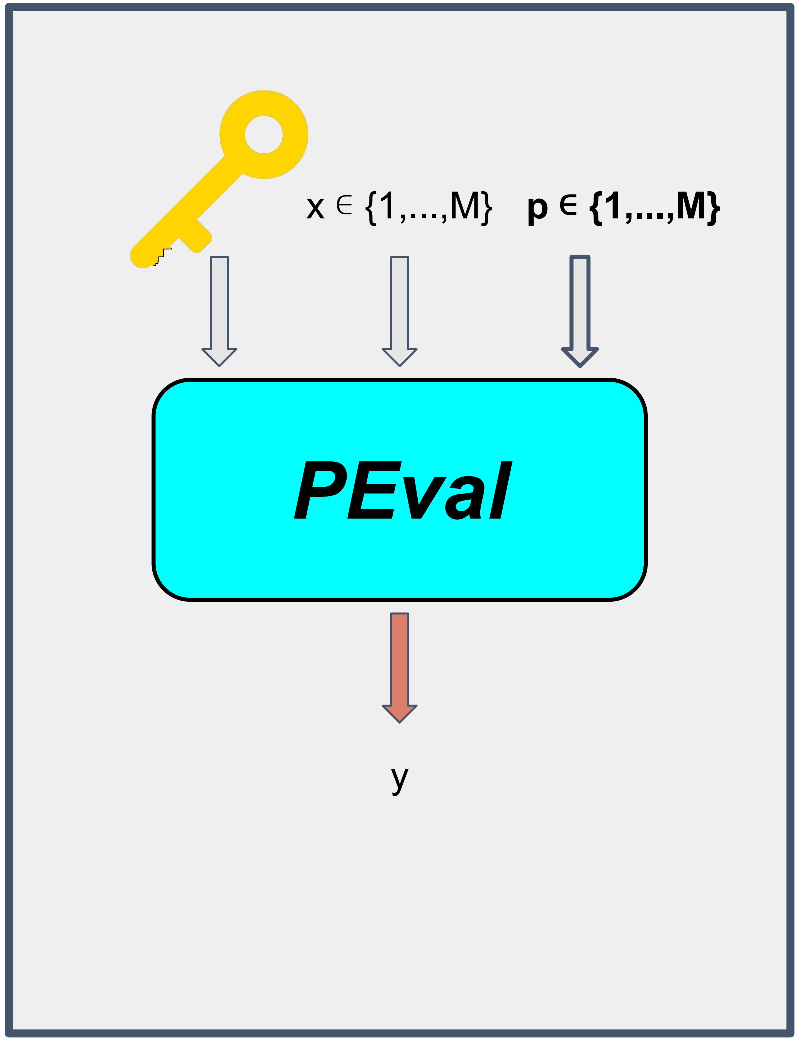
New key contains no information about evaluation at punctured point x

Privacy:

New key contains no information about punctured point x

[SACM ‘21]: First construction of sublinear time, polylog bandwidth PIR, using ppPRFs

Weak Privately Puncturable PRF



Weak Correctness:

For any input x' ≠ x, punctured key evaluates to same output as original key for a correct guess of the punctured point x

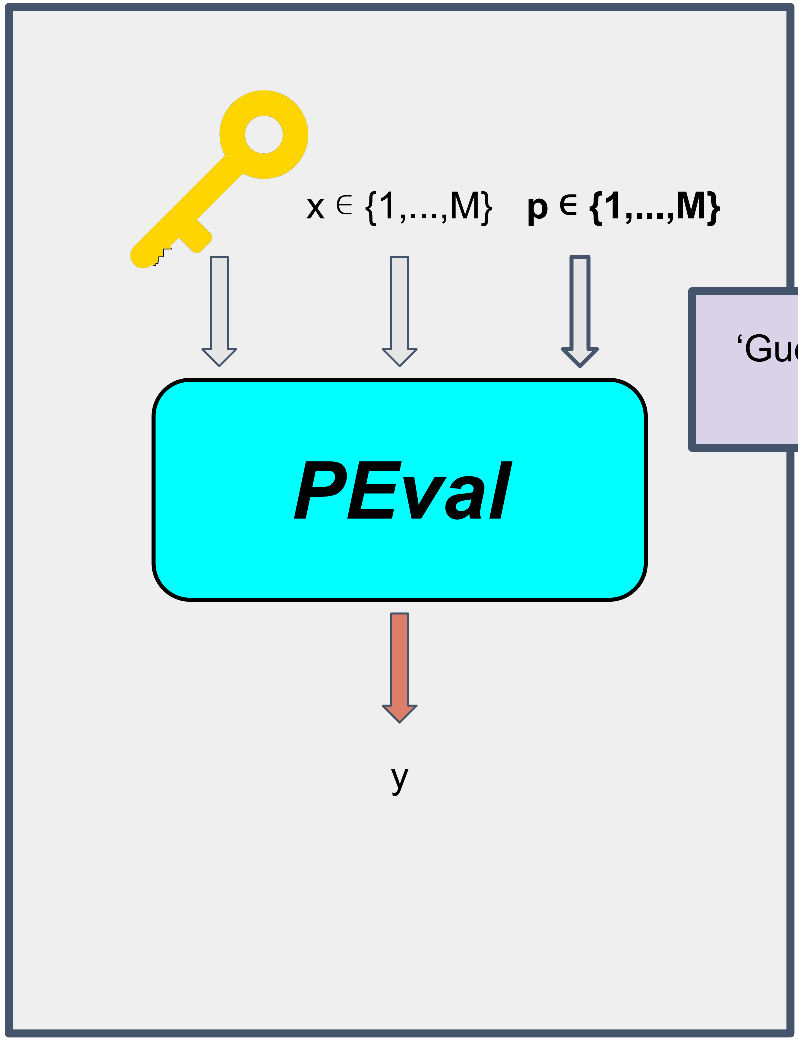
Security:

New key contains no information about evaluation at punctured point x

Privacy:

New key contains no information about punctured point x

Weak Privately Puncturable PRF



Weak Correctness:

For any input x’ ≠ x, punctured key evaluates to same output as original key for a correct guess of the punctured point x

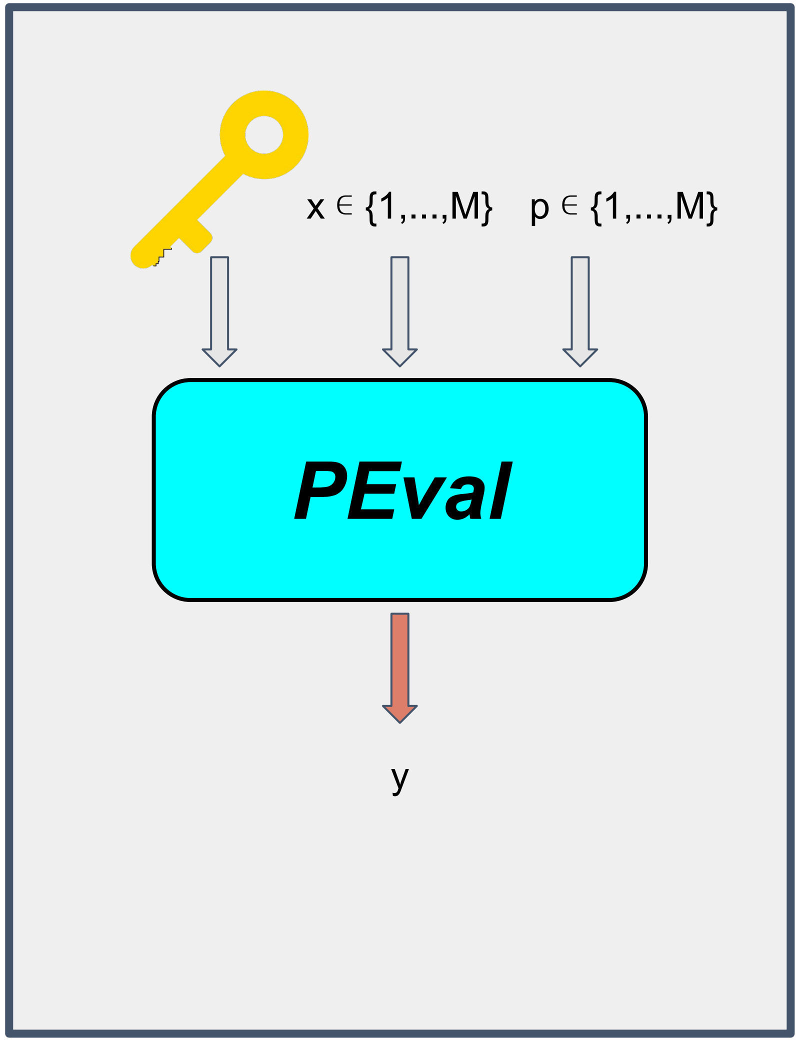
‘Guess’ of point that was punctured

ntains no information about evaluation at punctured point x

Privacy:

New key contains no information about punctured point x

Weak Privately Puncturable PRF



Weak Correctness:

For any input x' ≠ x, punctured key evaluates to same output as original key for a correct guess of the punctured point x

Security:

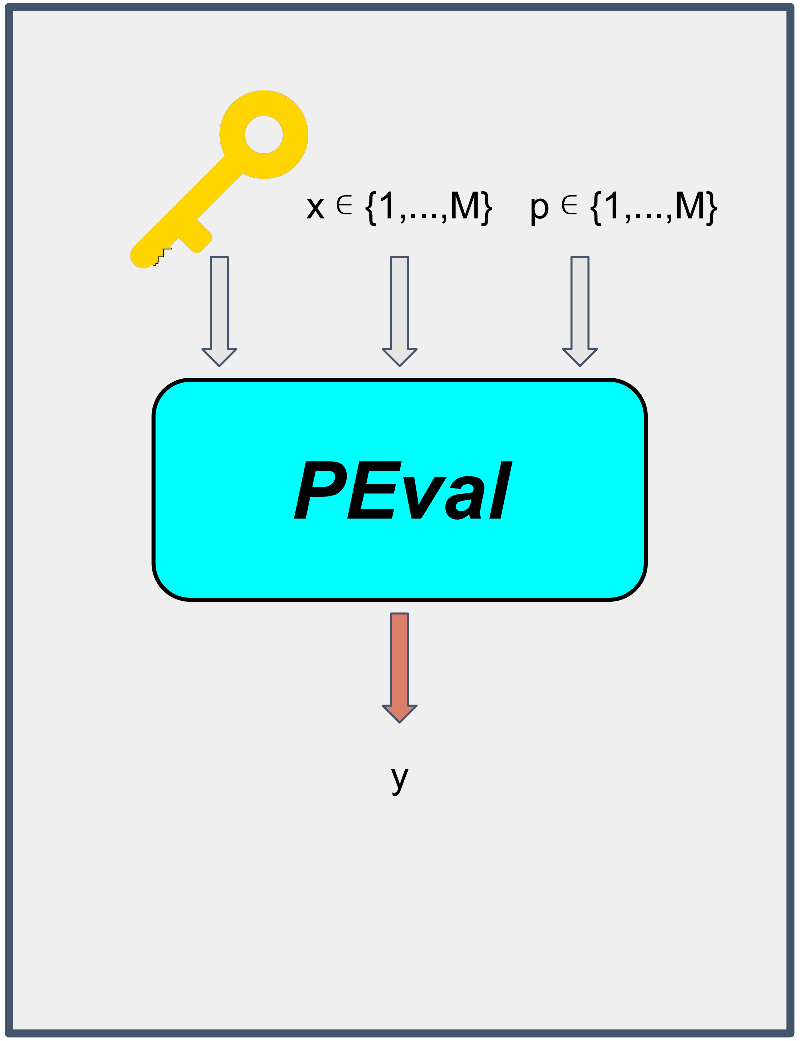
New key contains no information about evaluation at punctured point x

Privacy:

New key contains no information about punctured point x

Efficient Full Evaluation:

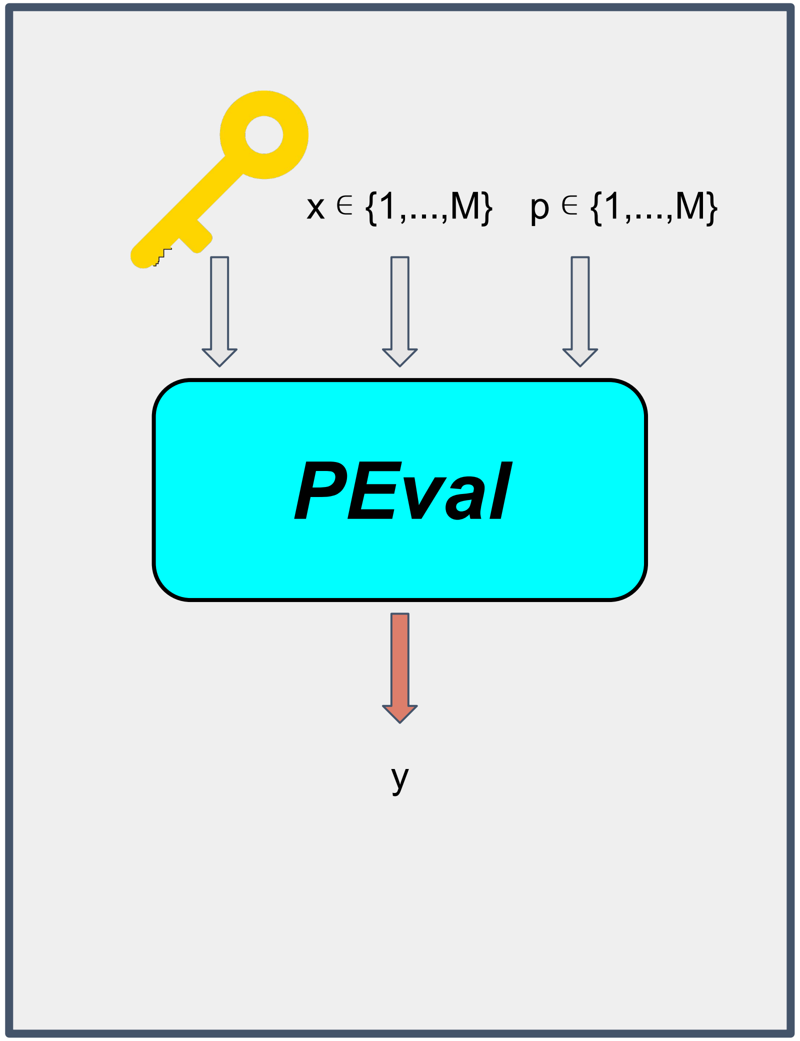
Weak Privately Puncturable PRF



Efficient Full Evaluation (Simplified):

Can compute a function over evaluations of entire domain in time quasi-linear in domain size.

Weak Privately Puncturable PRF



Efficient Full Evaluation (Simplified):

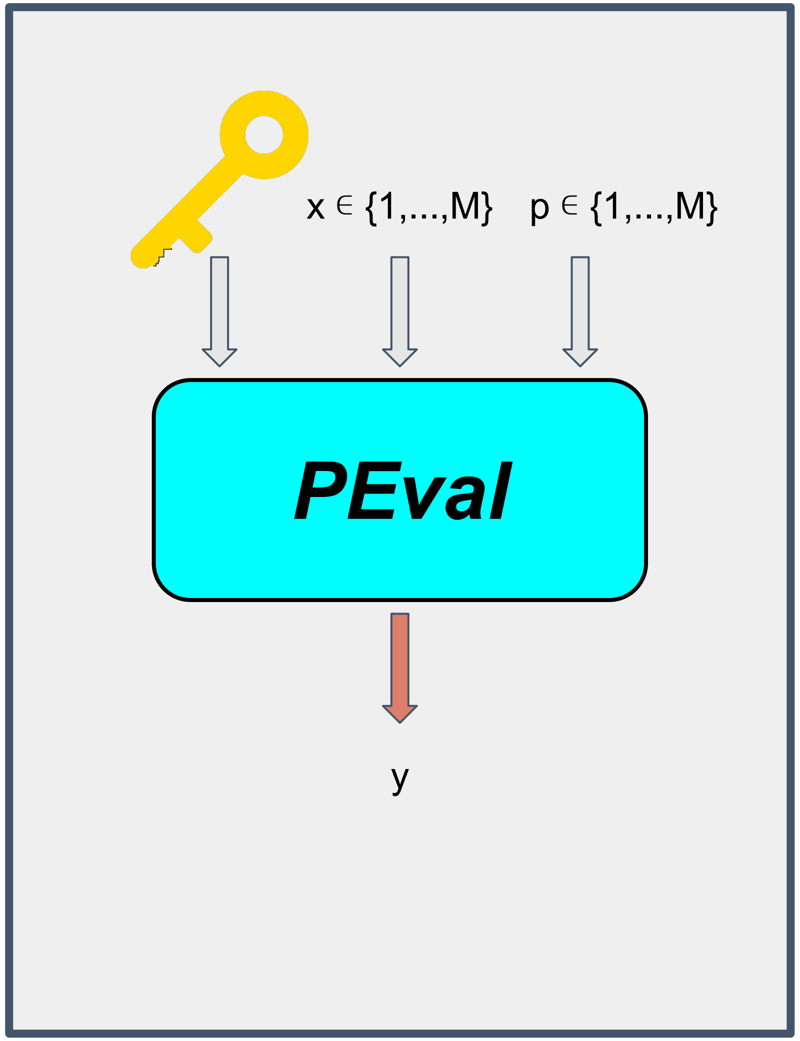
Can compute a function over evaluations of entire domain in time quasi-linear in domain size.

XOR:

Output array of length M where the i-th element is:

$$\bigoplus_{x \in \{1,...M\}} \mathrm{PEval}(k', x, i)$$

Weak Privately Puncturable PRF



Efficient Full Evaluation (Simplified):

Can compute a function over evaluations of entire domain in time quasi-linear in domain size.

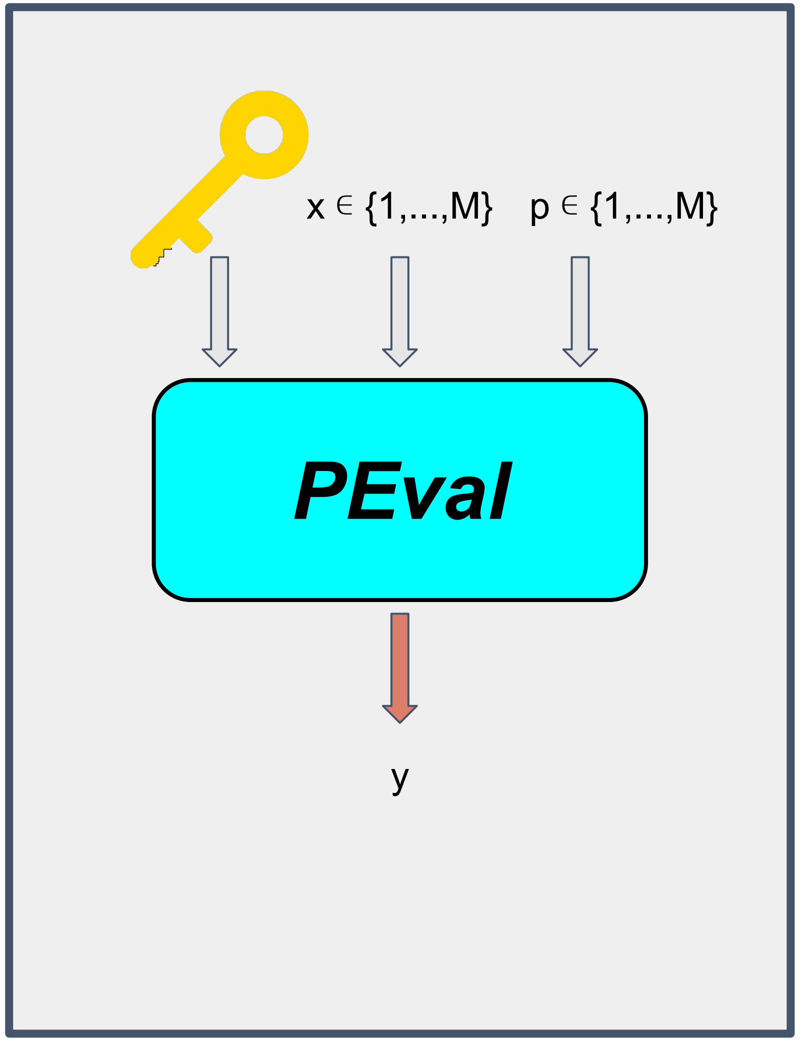
XOR:

Output array of length M where the i-th element is:

$$\bigoplus_{x \in \{1,\ldots M\}} \mathrm{PEval}(k', x, i)$$

At index equal to point that was punctured, xor will be consistent with original key (except for element punctured)

Weak Privately Puncturable PRF



Efficient Full Evaluation (Simplified):

Can compute a function over evaluations of entire domain in time quasi-linear in domain size.

XOR:

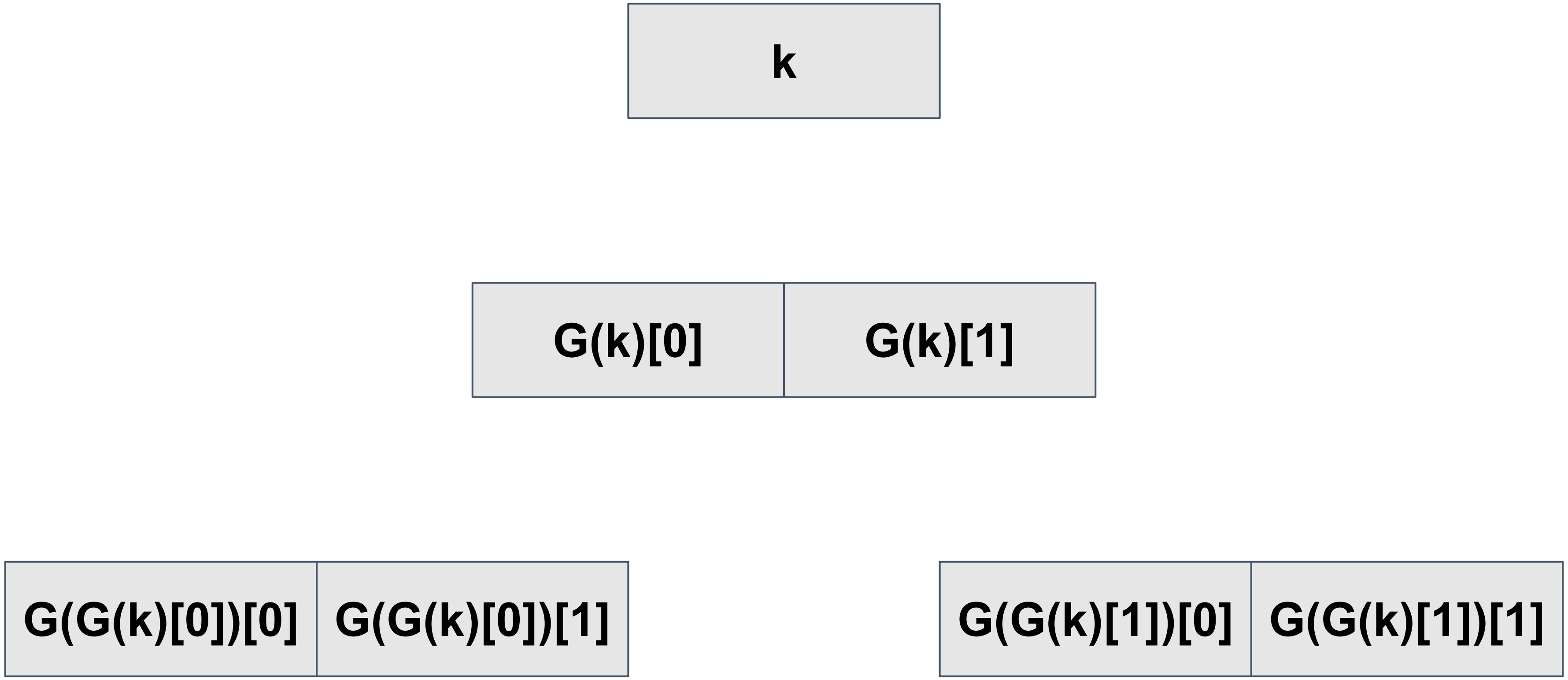
Output array of length M where the i-th element is:

$$\oplus_{x \in \{1,...M\}} \mathrm{PEval}(k', x, i)$$

At index equal to point that was punctured, xor will be consistent with original key (except for element punctured)

By Efficient Full Evaluation, we can output this array in quasi-linear time in M.

GGM PRF



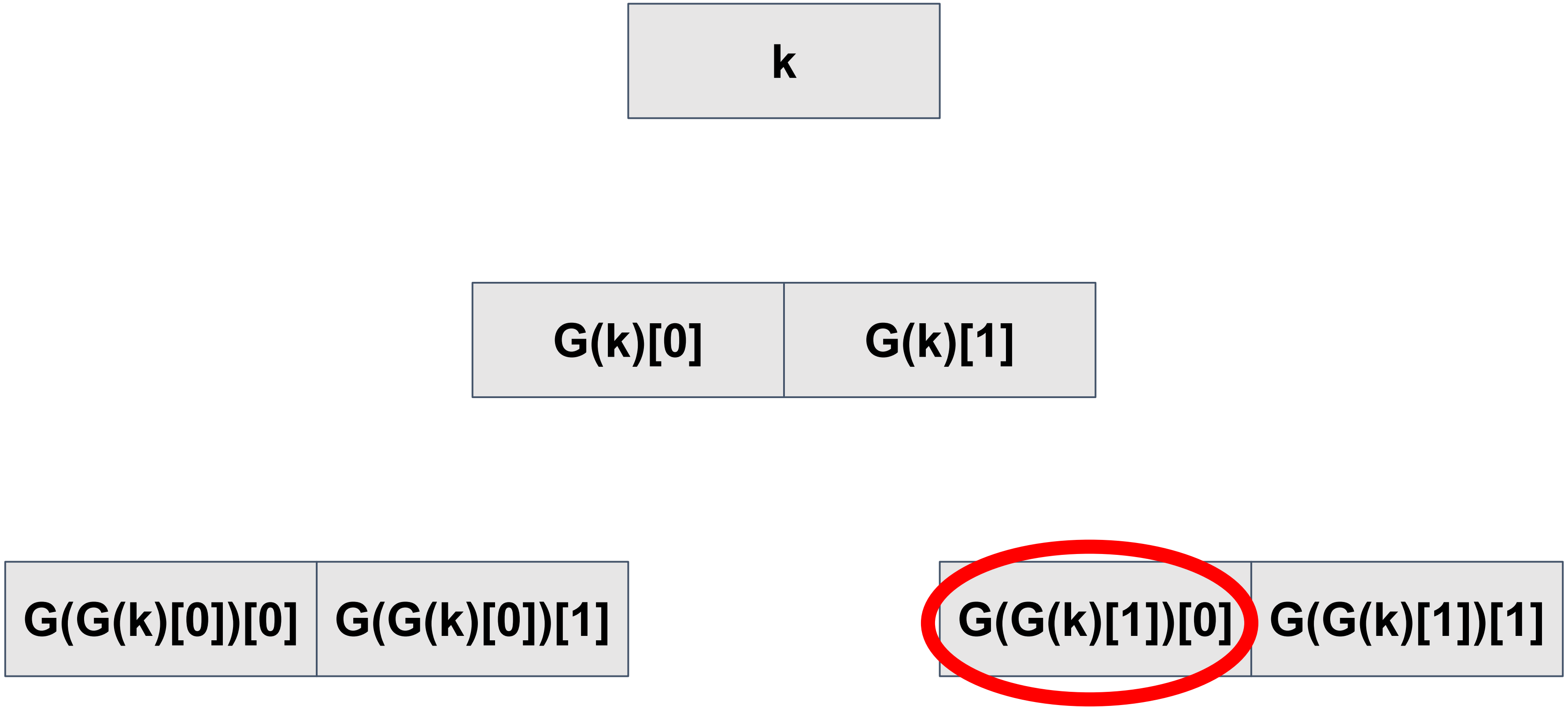
k

G(k)[0] G(k)[1]

G(G(k)[0])[0] G(G(k)[0])[1]

G(G(k)[1])[0] G(G(k)[1])[1]

GGM PRF



k

G(k)[0]

G(k)[1]

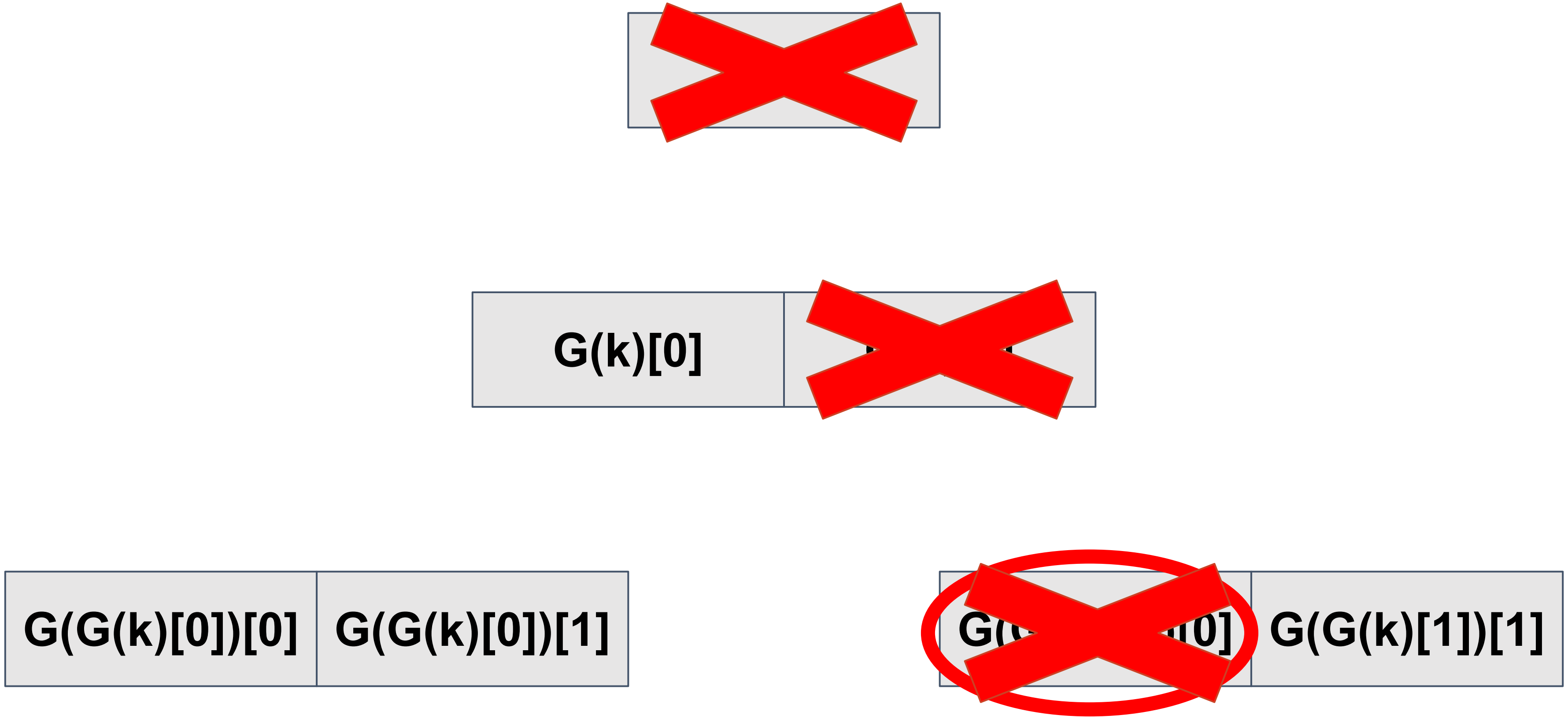
G(G(k)[0])[0]

G(G(k)[0])[1]

G(G(k)[1])[0]

G(G(k)[1])[1]

GGM PRF



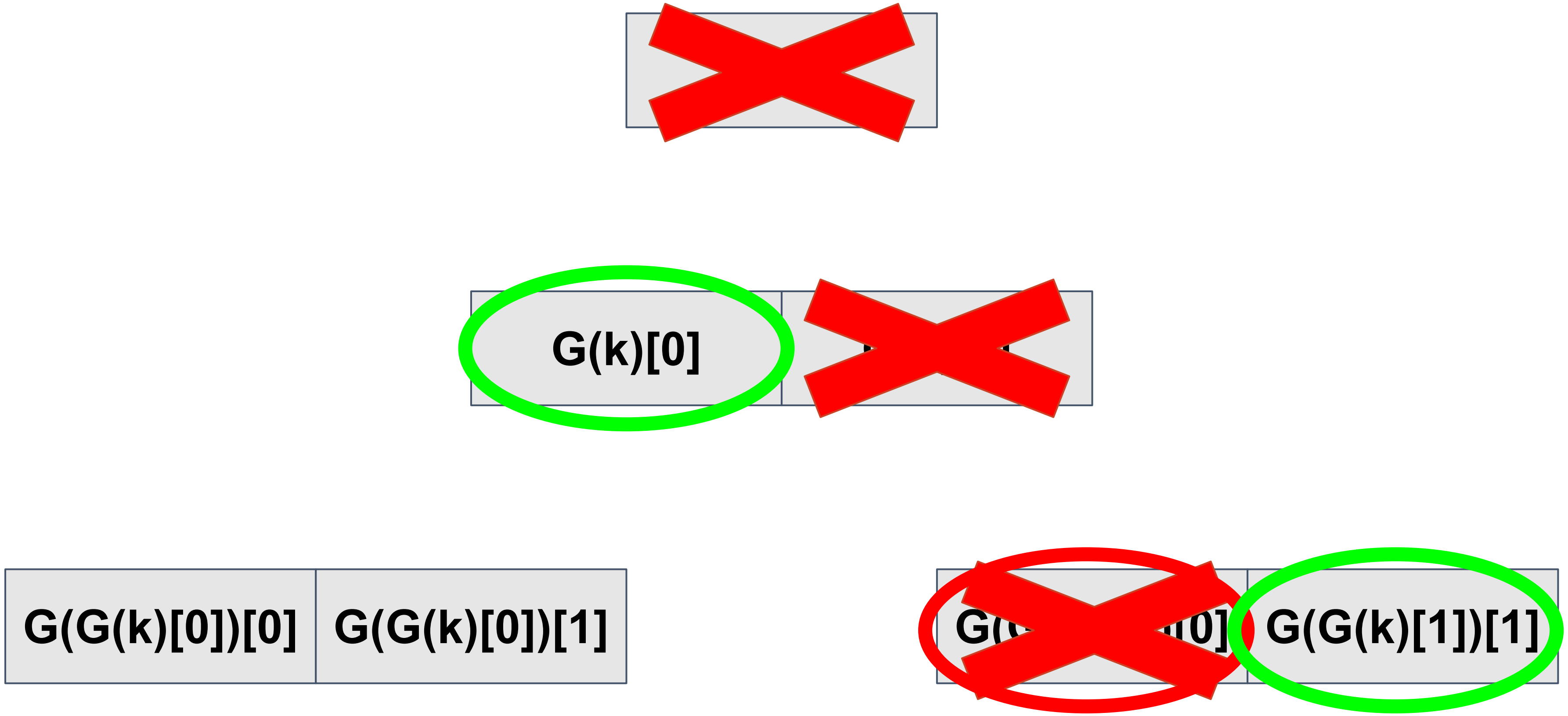
G(k)[0]

G(G(k)[0])[0]

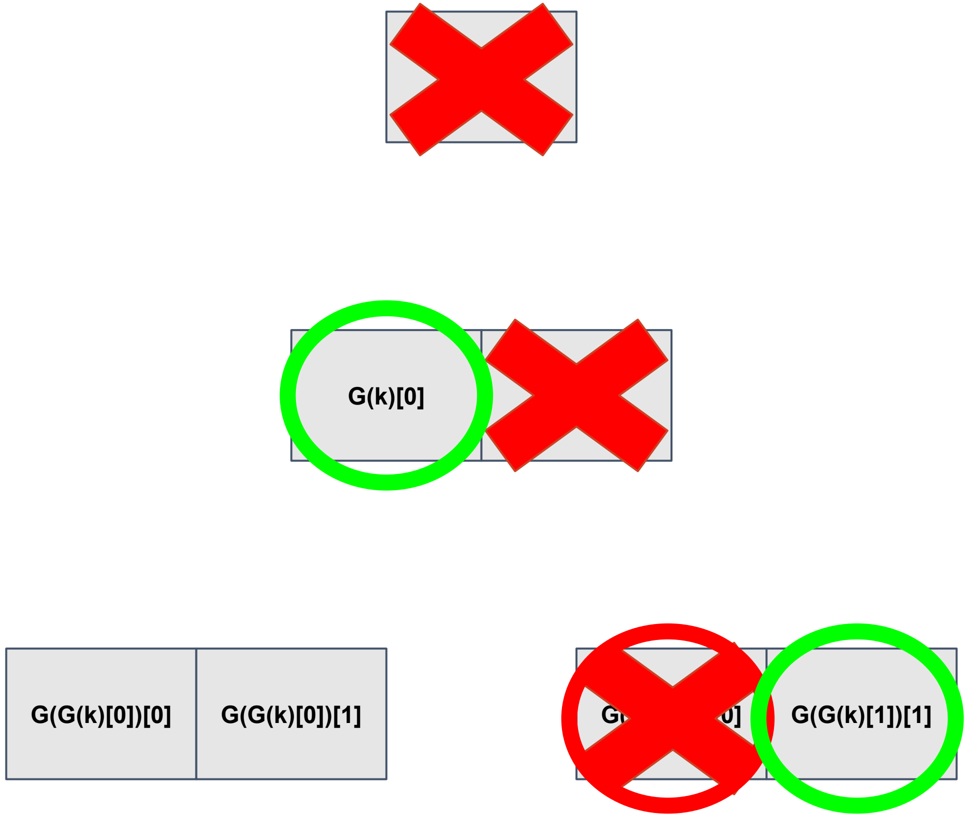
G(G(k)[0])[1]

G(G(k)[1])[1]

GGM PRF

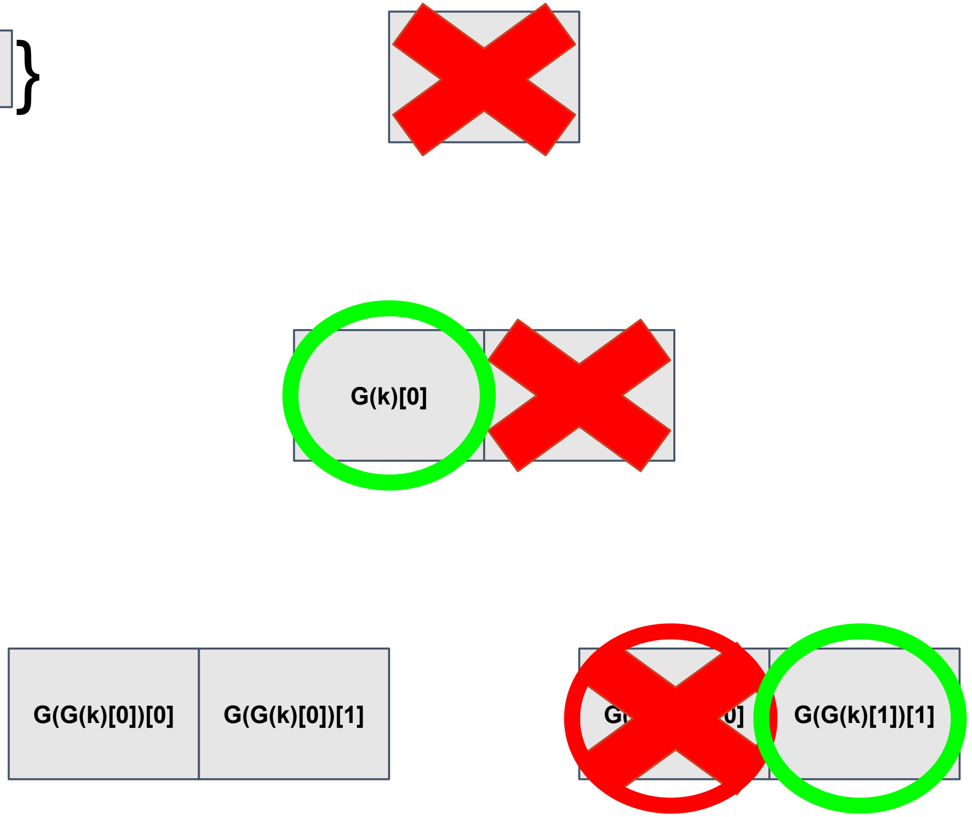


GGM PRF



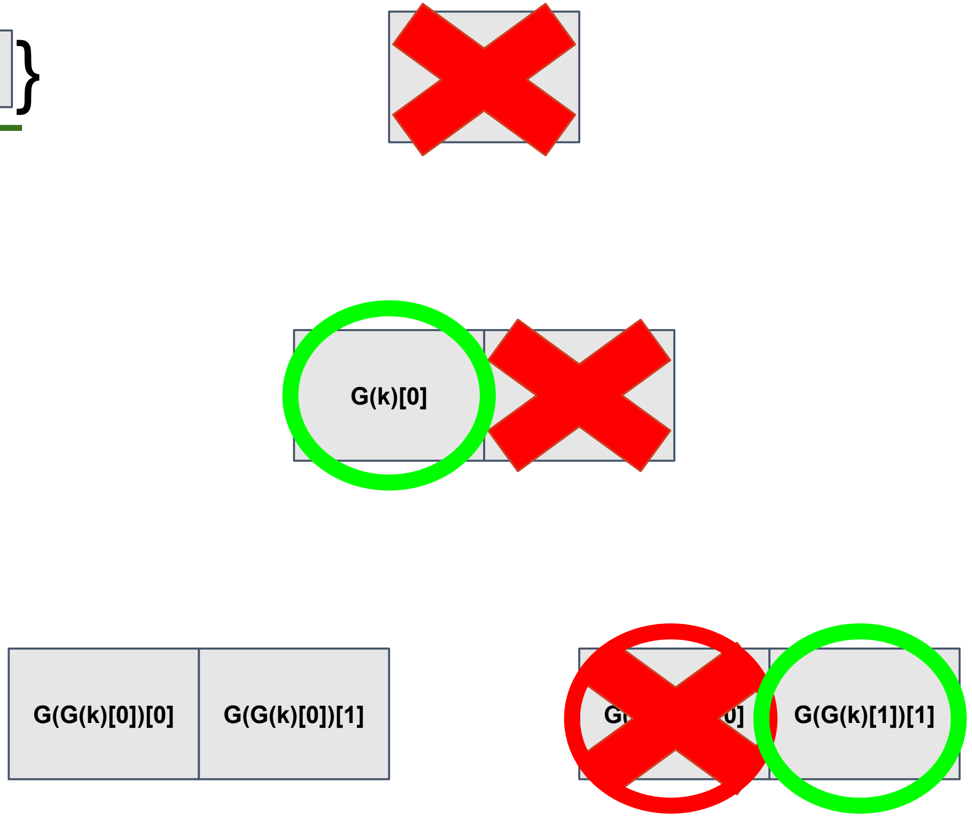
GGM PRF

Punctured Key: { 3, G(k)[0], G(G(k)[1])[1] }



GGM PRF

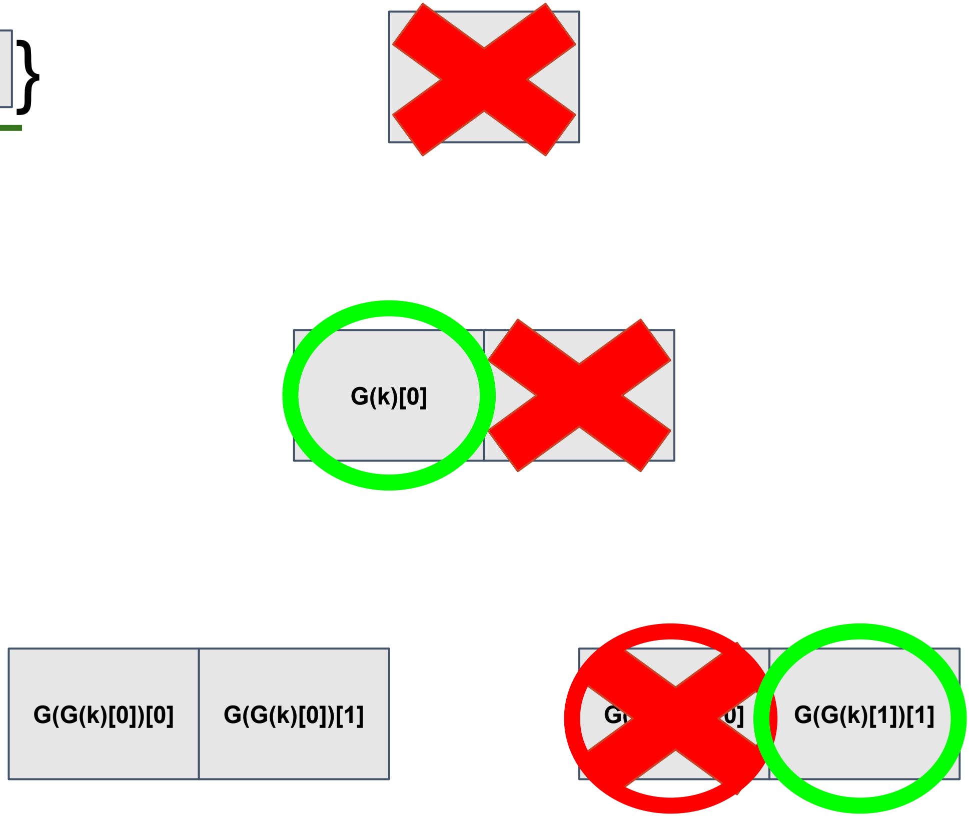
Punctured Key: $\{ \underbrace{3,}_{\text{index punctured}} \underbrace{\boxed{G(k)[0]}, \boxed{G(G(k)[1])[1]}}_{\text{Left-to-right ordering of nodes in adjacent path}} \}$



Weak Privately Puncturable PRF

Privately
Punctured Key: { ~~8~~, G(k)[0], G(G(k)[1])[1] }

Left-to-right ordering of nodes in adjacent path



Weak ppPRF

Privately

Punctured Key: $\{\cancel{3}, \overbrace{\boxed{G(k)[0]}, \boxed{G(G(k)[1])[1]}}^{\text{Left-to-right ordering of nodes in adjacent path}}\}$

Recipient can guess punctured index:

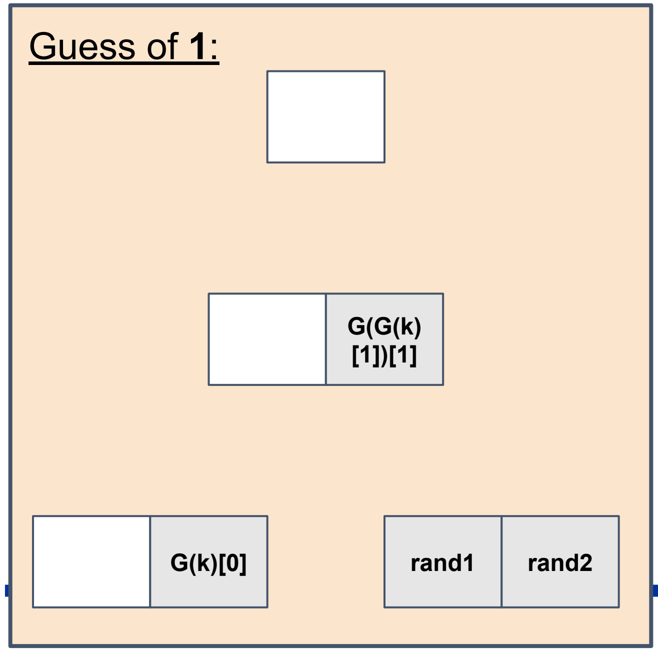
Weak ppPRF

Left-to-right ordering of nodes in adjacent path

Privately

Punctured Key: { ~~3~~, G(k)[0], G(G(k)[1])[1] }

Recipient can guess punctured index:

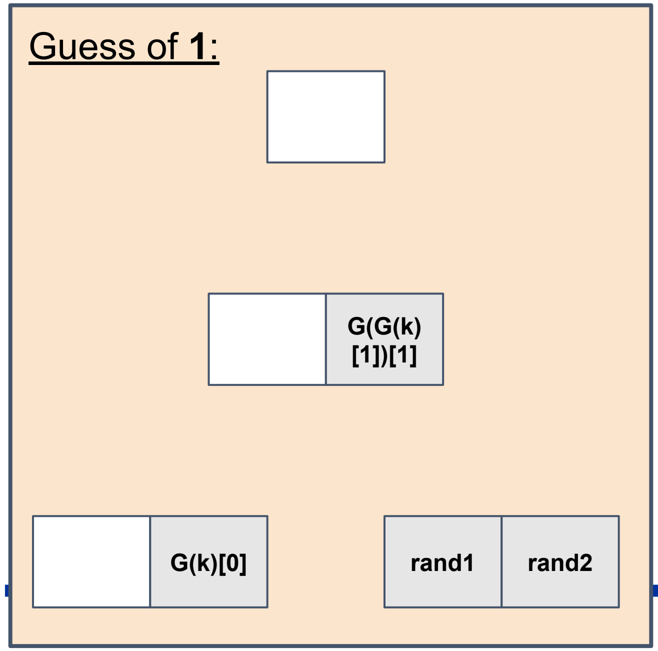


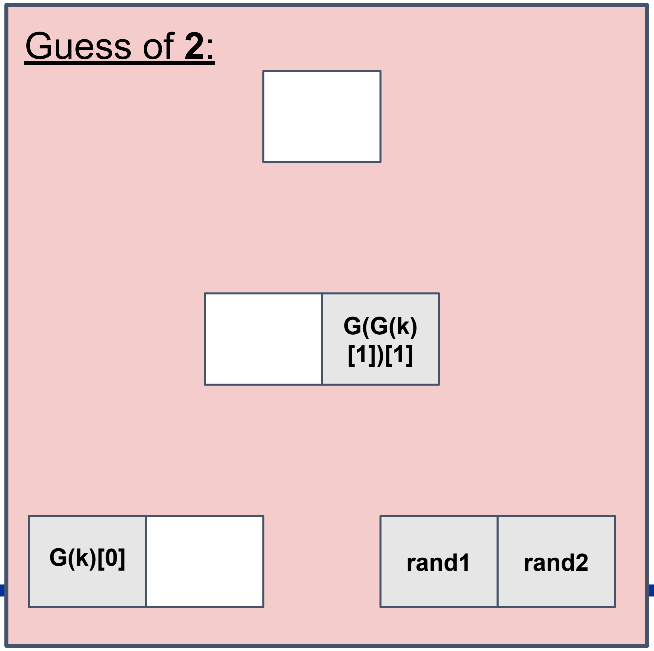
Weak ppPRF

Left-to-right ordering of nodes in adjacent path

Privately
Punctured Key: { ~~3~~, G(k)[0], G(G(k)[1])[1] }

Recipient can guess punctured index:





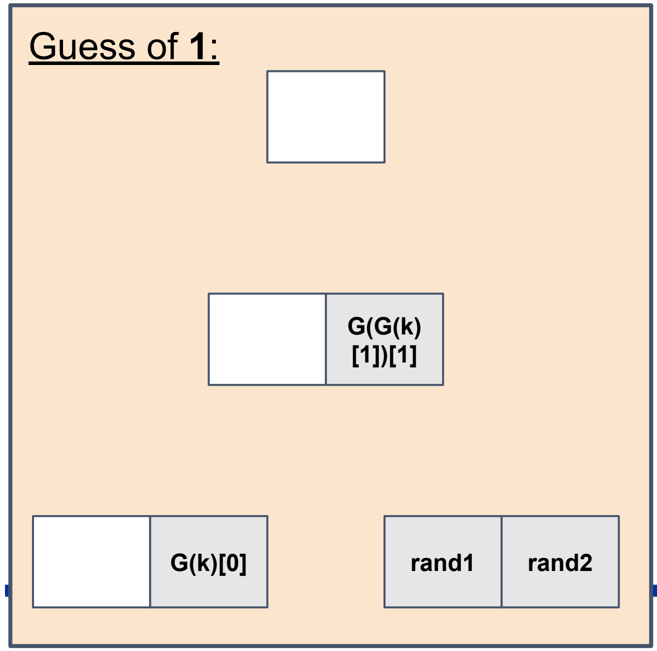
Weak ppPRF

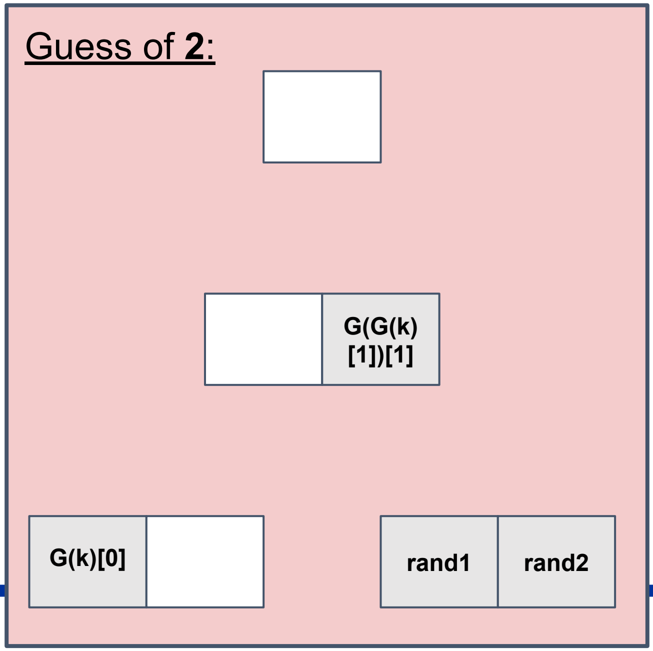
Left-to-right ordering of nodes in adjacent path

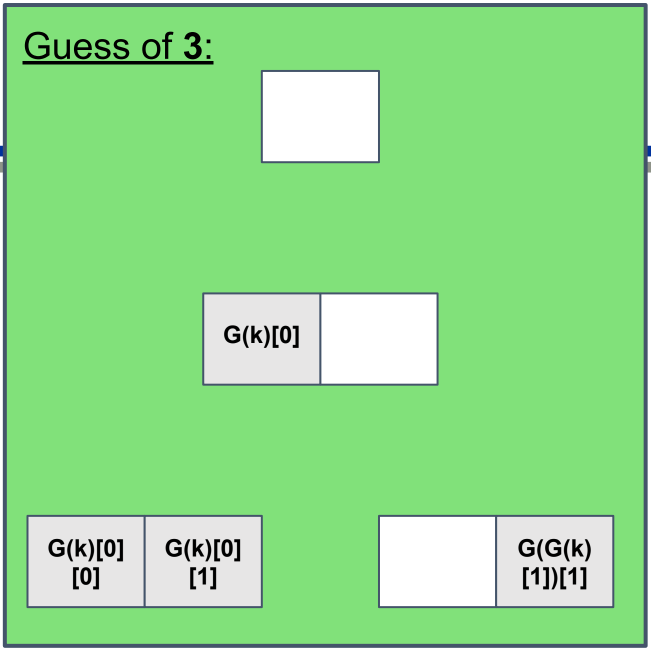
Privately

Punctured Key: { ~~3~~, G(k)[0] , G(G(k)[1])[1] }

Recipient can guess punctured index:


Guess of **1**:


Guess of **2**:

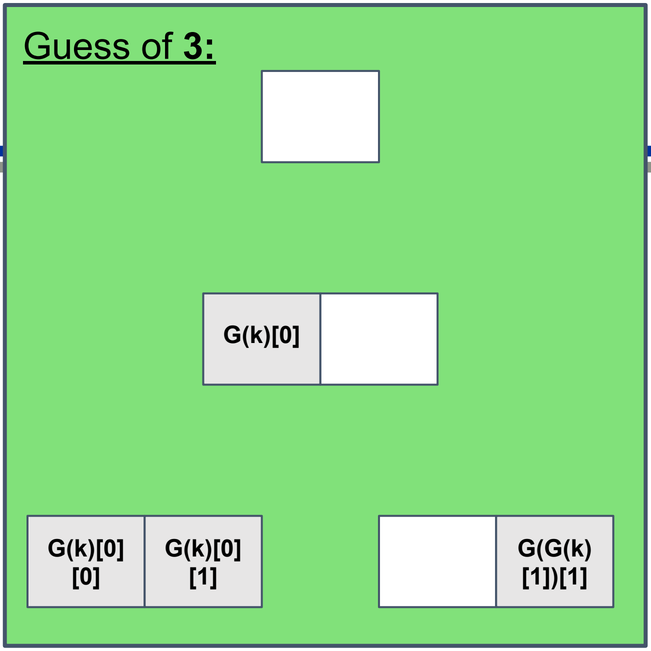

Guess of **3**:

Weak ppPRF

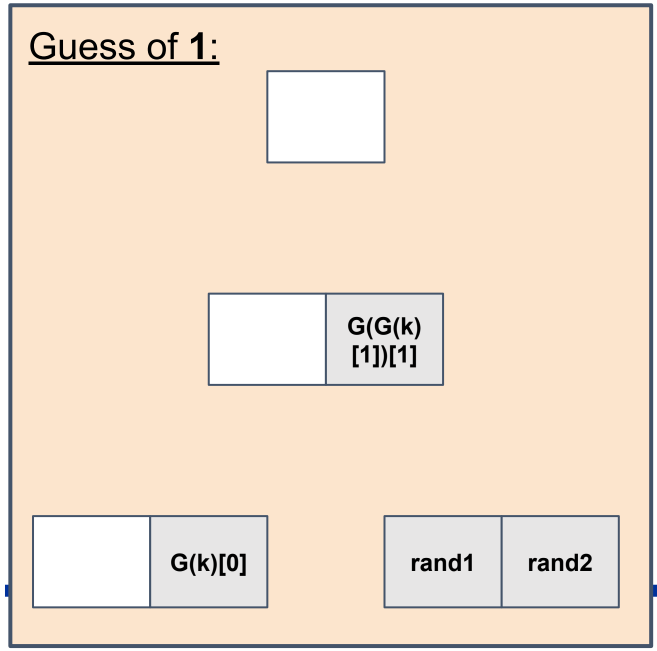
Left-to-right ordering of nodes in adjacent path

Privately Punctured Key: { ~~3~~, G(k)[0], G(G(k)[1])[1] }

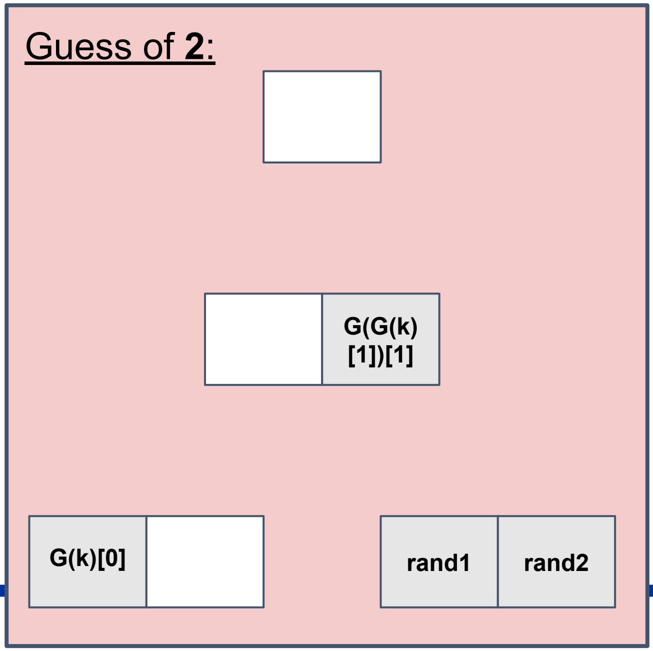
Recipient can guess punctured index:



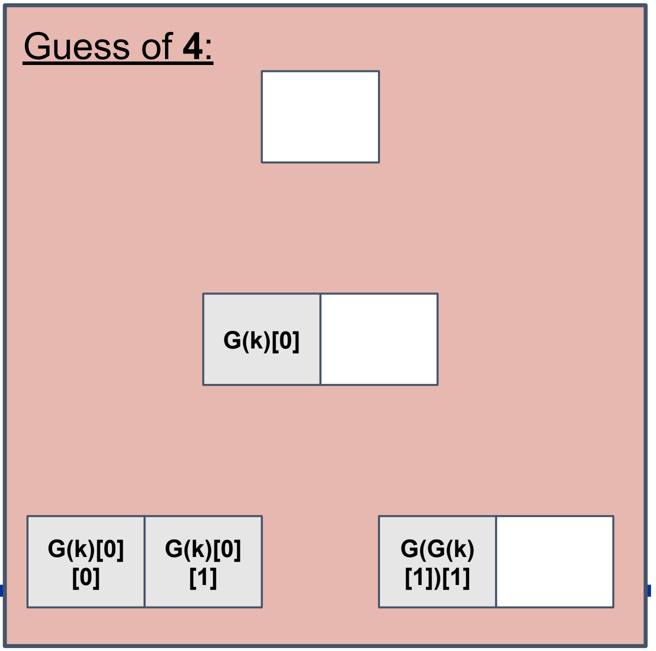
Guess of **3:**



Guess of **1**:

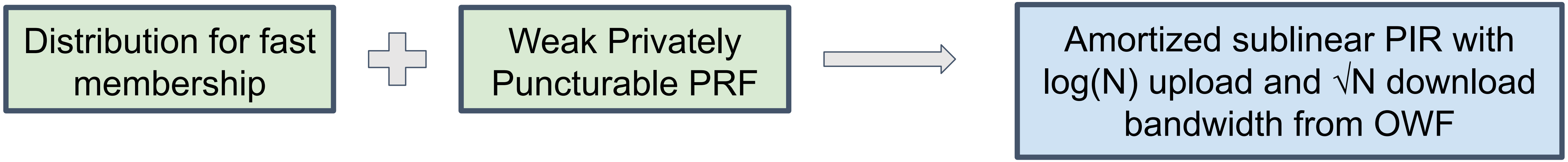


Guess of **2**:



Guess of **4**:

TreePIR

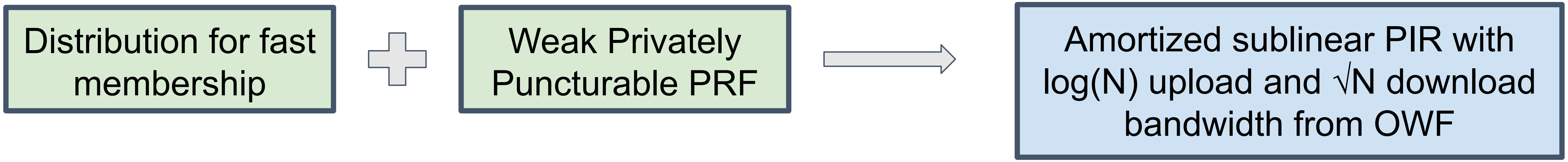


Distribution for fast membership

Weak Privately Puncturable PRF

Amortized sublinear PIR with log(N) upload and √N download bandwidth from OWF

TreePIR



Distribution for fast membership

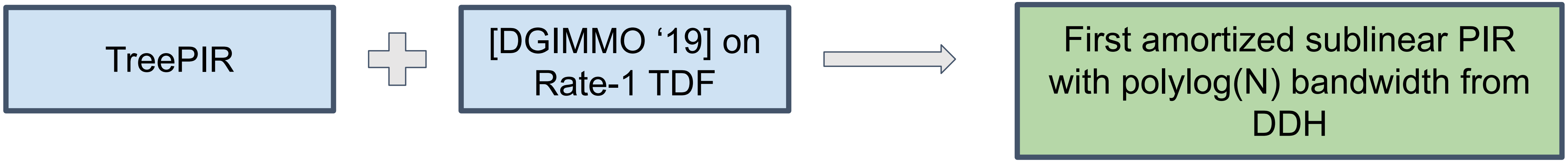
Weak Privately Puncturable PRF

Amortized sublinear PIR with log(N) upload and $\sqrt{N}$ download bandwidth from OWF

Database of 2^{32} bit entries

PIR Scheme	Client storage	Amortized query time	Online Bandwidth
TreePIR	1MB	3.5s	16.6KB
Checklist [KC21]	8GB	12.5s	0.5KB

TreePIR plus recursion

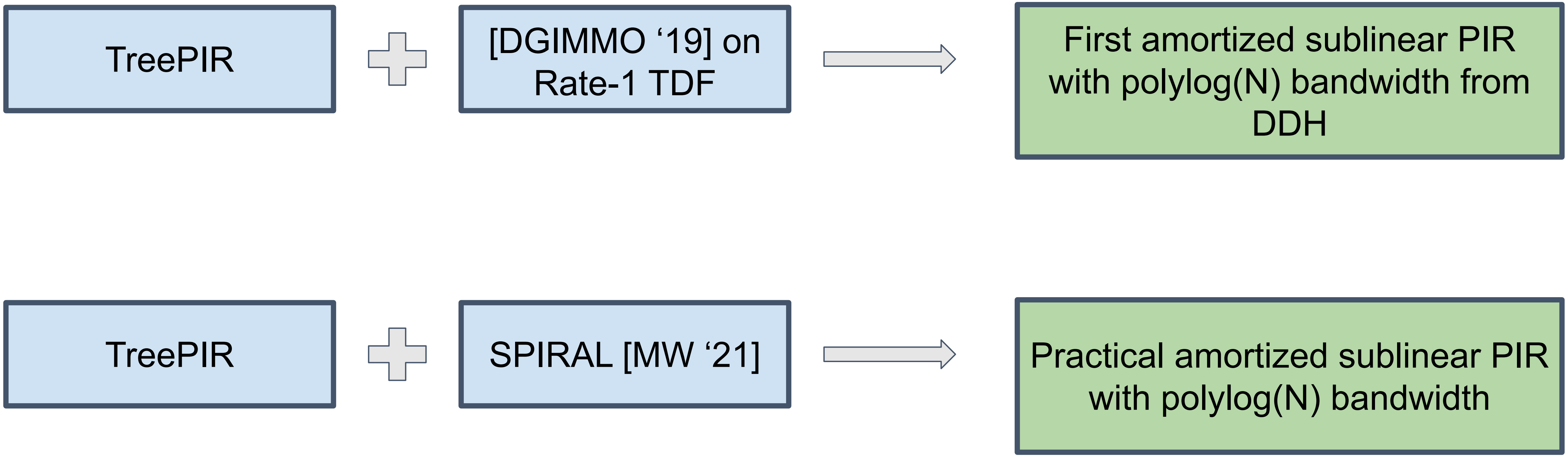


TreePIR

[DGIMMO ‘19] on Rate-1 TDF

First amortized sublinear PIR with polylog(N) bandwidth from DDH

TreePIR plus recursion



TreePIR + [DGIMMO '19] on Rate-1 TDF → First amortized sublinear PIR with polylog(N) bandwidth from DDH

TreePIR + SPIRAL [MW '21] → Practical amortized sublinear PIR with polylog(N) bandwidth

New works building on TreePIR

- Piano: Extremely Simple, Single-Server PIR with Sublinear Server Computation [ZLTS '23] (to appear IEEE S&P '24)
- Simple and Practical Amortized Sublinear Private Information Retrieval [MIR '23] (ePrint)

Sources for icons:

https://icon-library.com/icon/key-icon-png-7.html.html

https://www.onlygfx.com/magnifying-glass-clipart-png-transparent/

https://www.freepnglogos.com/images/tick-33835.html