A non-comparison oblivious sort and its application to private k-NN

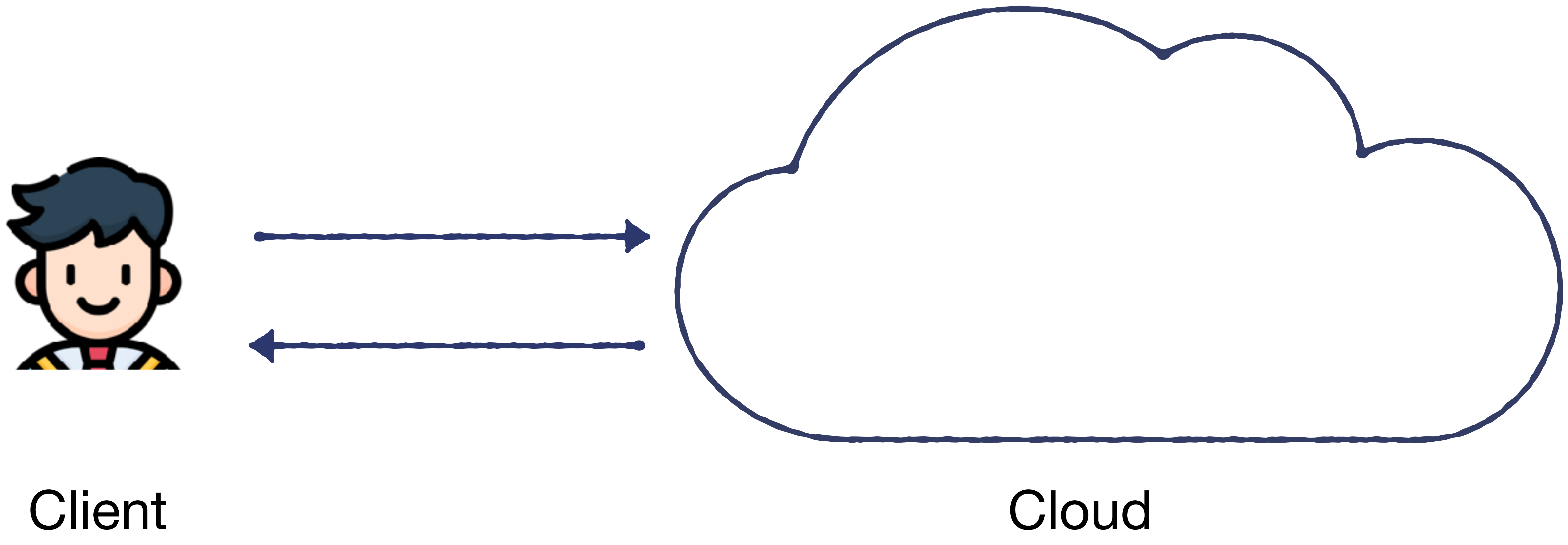
Sofiane Azogagh, Marc-Olivier Killijian and Félix Larose-Gervais

PETS'25 - July 17th 2025

Introduction and context

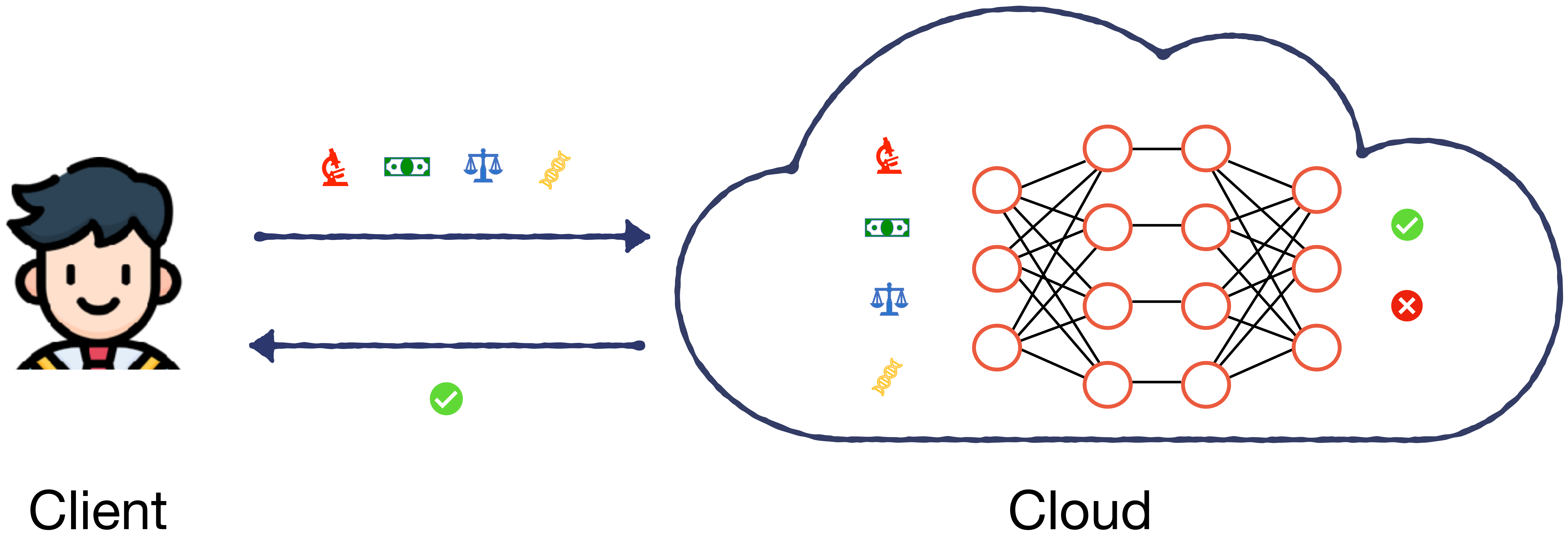
Context and security model

Outsourcing the computation



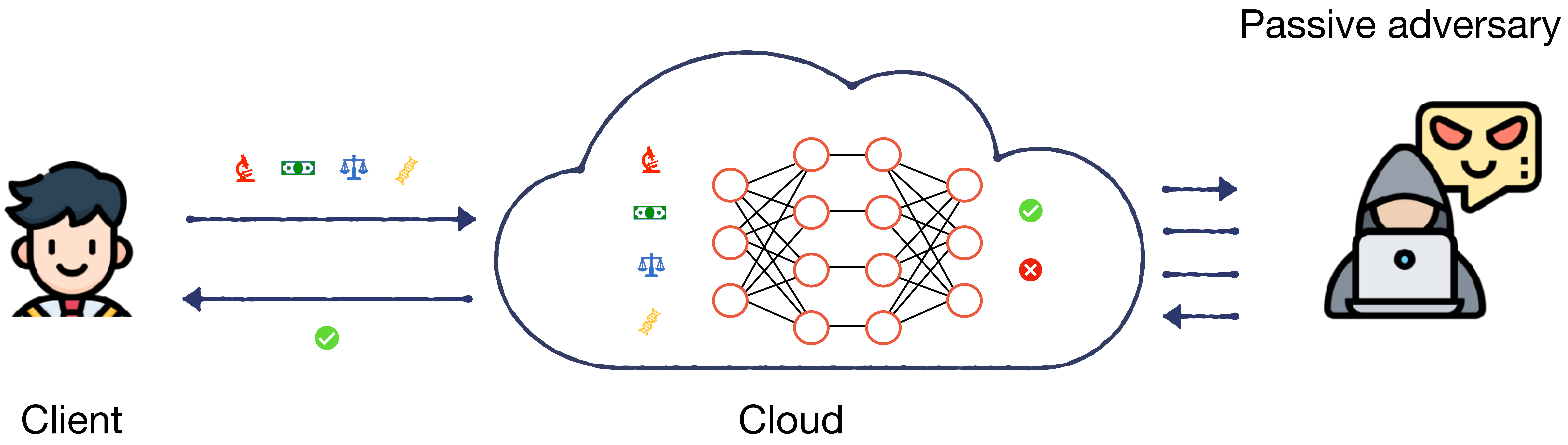
Context and security model

Outsourcing the computation



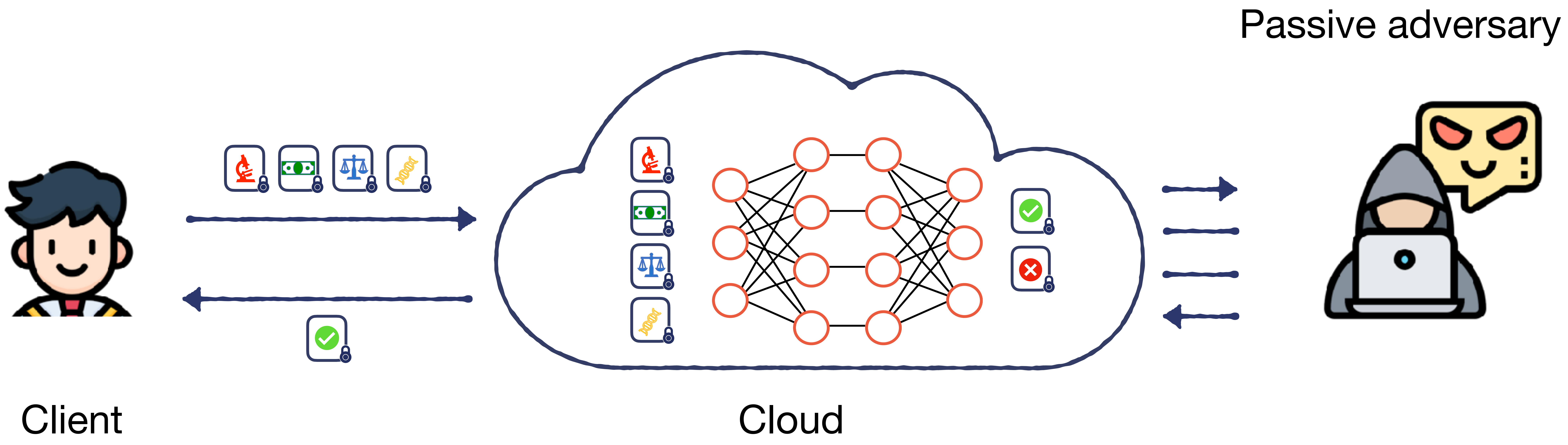
Context and security model

Outsourcing the computation



Context and security model

Outsourcing the computation



Oblivious algorithm

Just a definition

```
if a > b:
    max = a
else:
    max = b
```

Non oblivious

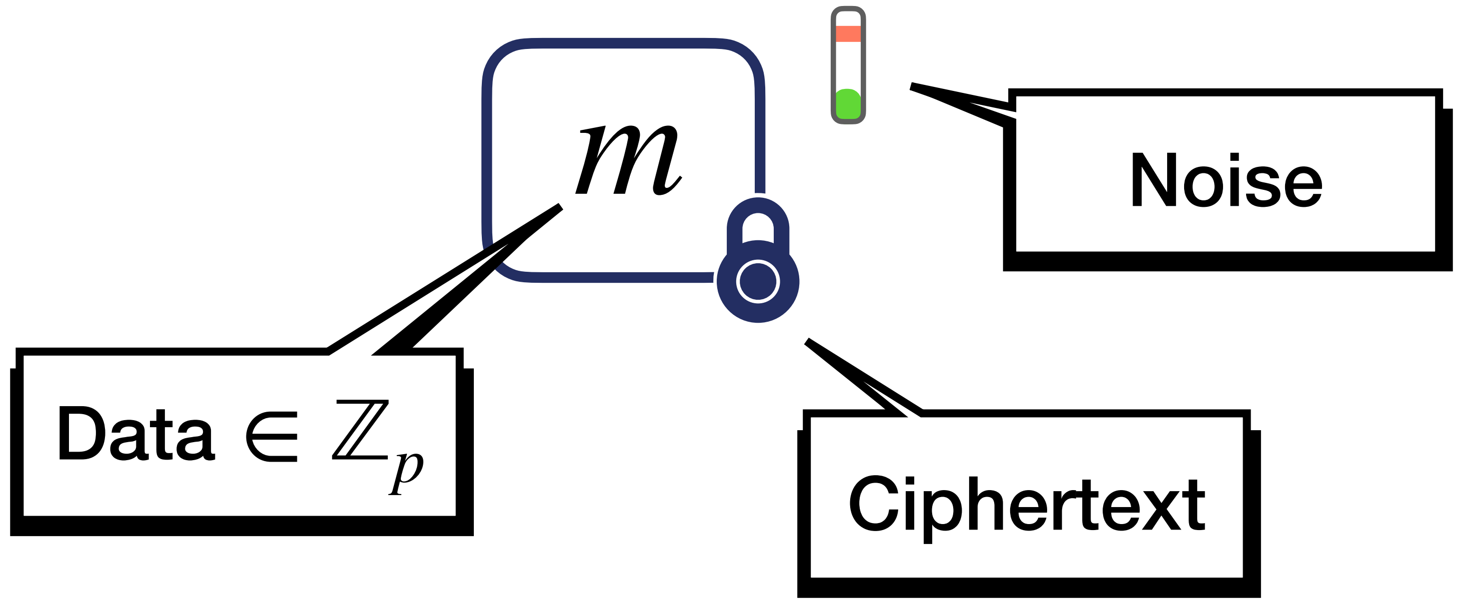
```
s = (a>b)
max = a*s + b*(1-s)
```

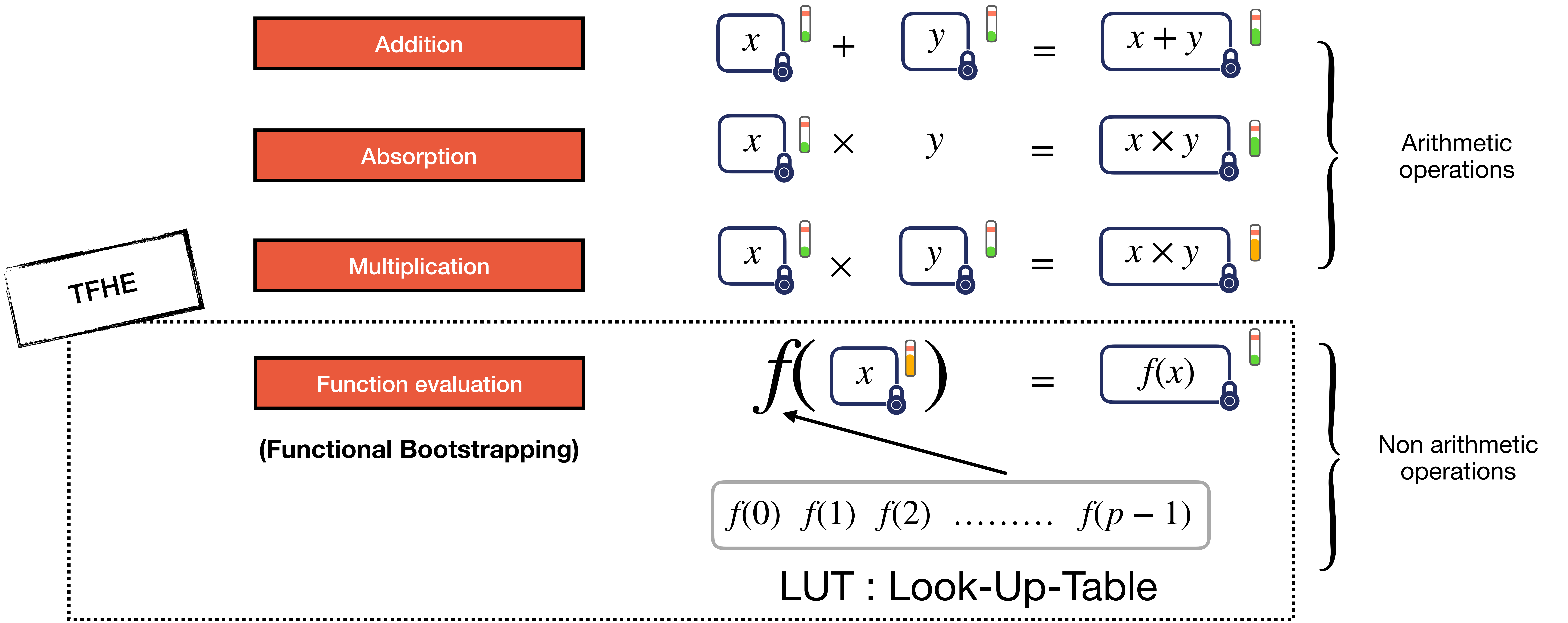
Oblivious

Oblivious algorithms are algorithms whose **access patterns** (e.g., which memory addresses they touch) and **control flow** (e.g., which branch they take) are **independent of the input data values.**

Fully Homomorphic Encryption

Fully Homomorphic Encryption





Chillotti et al. : TFHE: Fast Fully Homomorphic Encryption Over the Torus. J Cryptol 33, 34–91 (2020)

TFHE cryptosystem

Different types of ciphertexts

TFHE cryptosystem

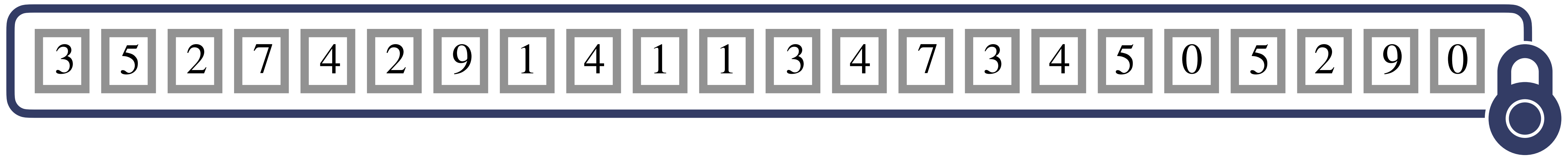
Different types of ciphertexts

LWE ciphertext

TFHE cryptosystem
Different types of ciphertexts

LWE ciphertext

RLWE ciphertext



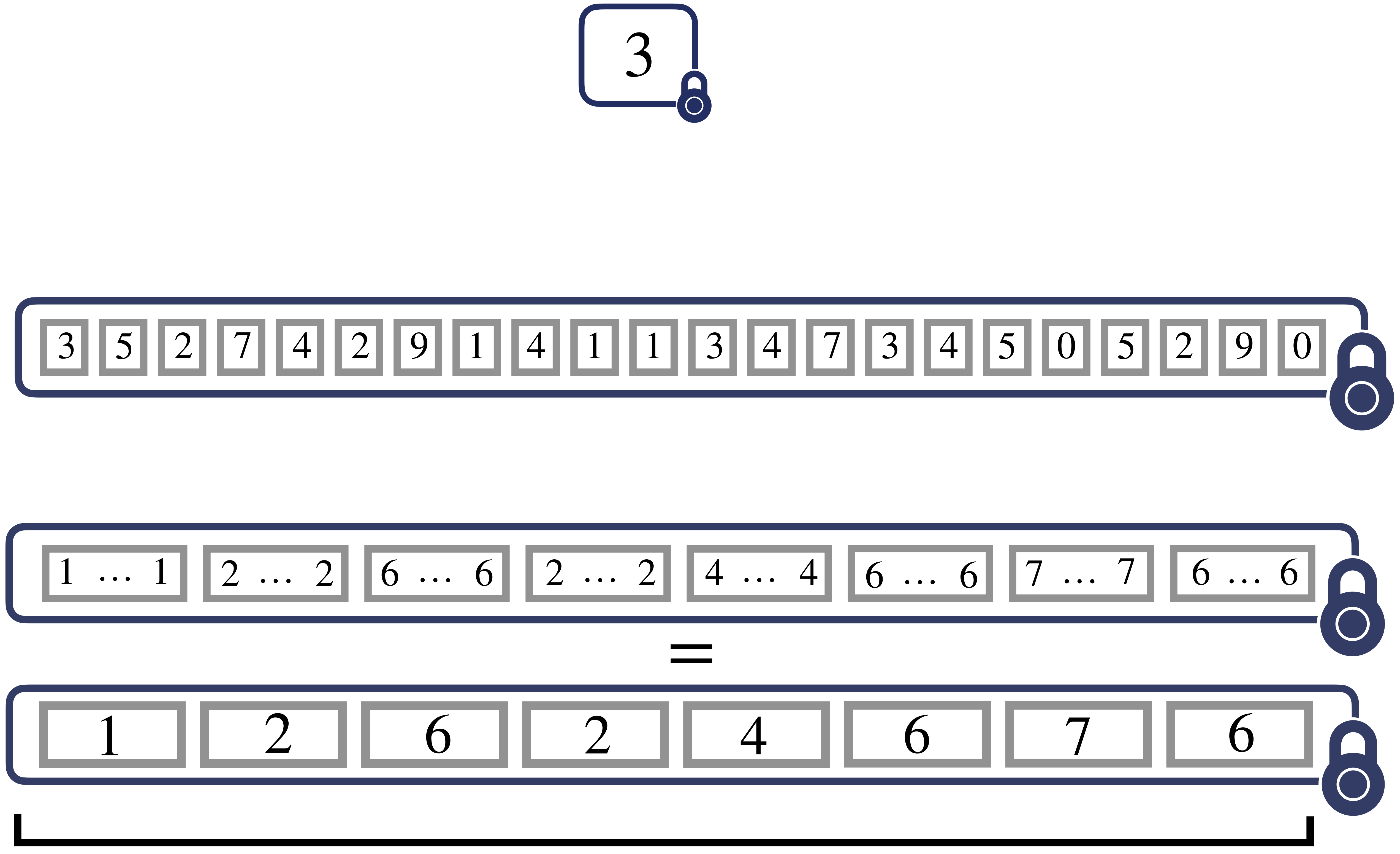
TFHE cryptosystem

Different types of ciphertexts

LWE ciphertext

RLWE ciphertext

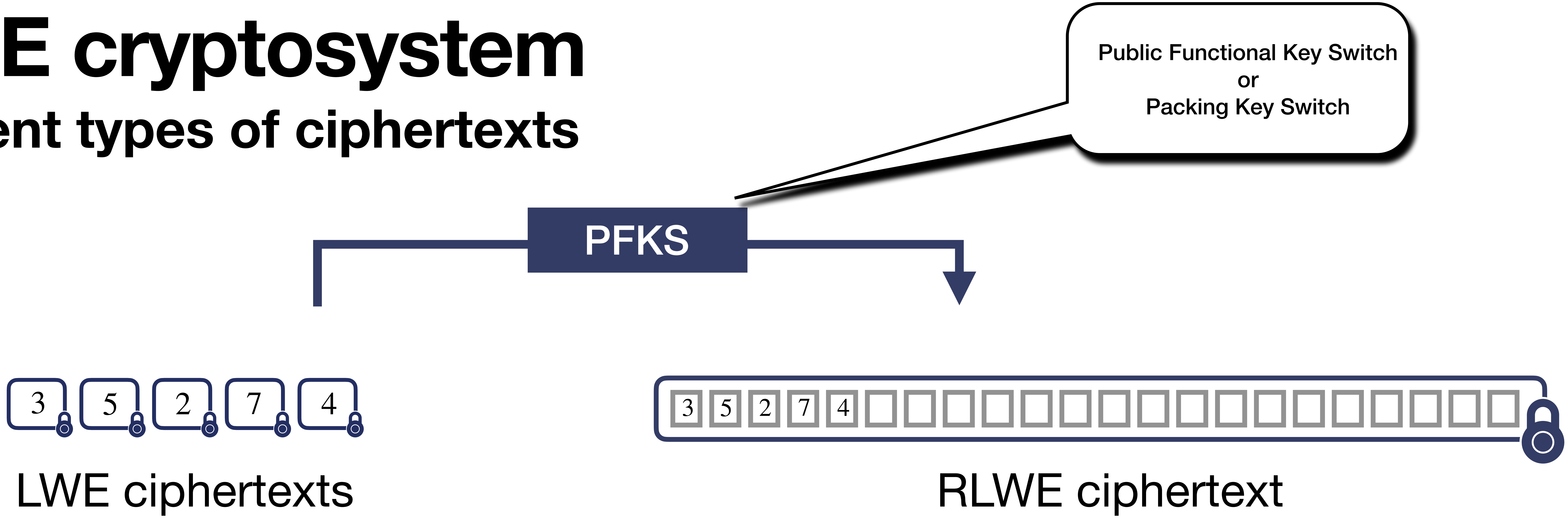
LUT ciphertext



p elements of $\mathbb{Z}_p$

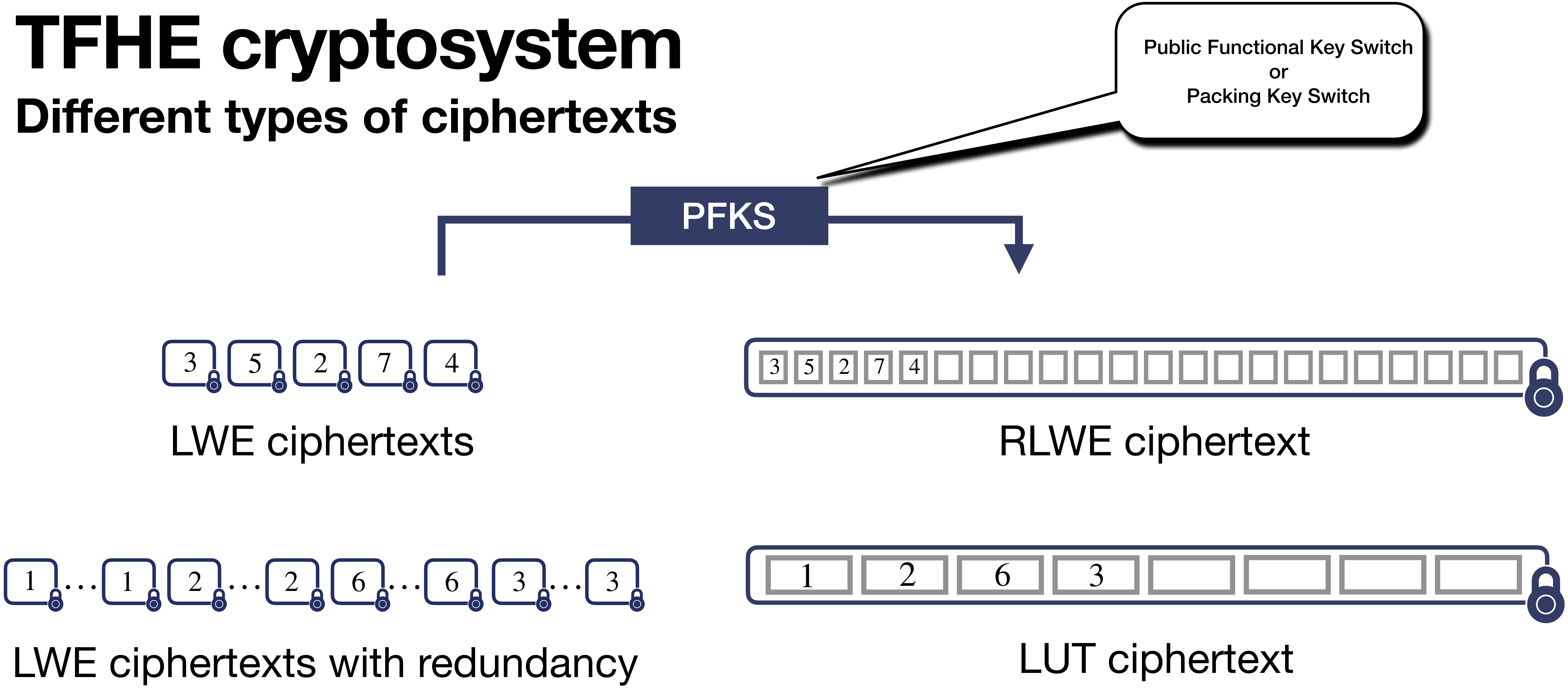
TFHE cryptosystem

Different types of ciphertexts



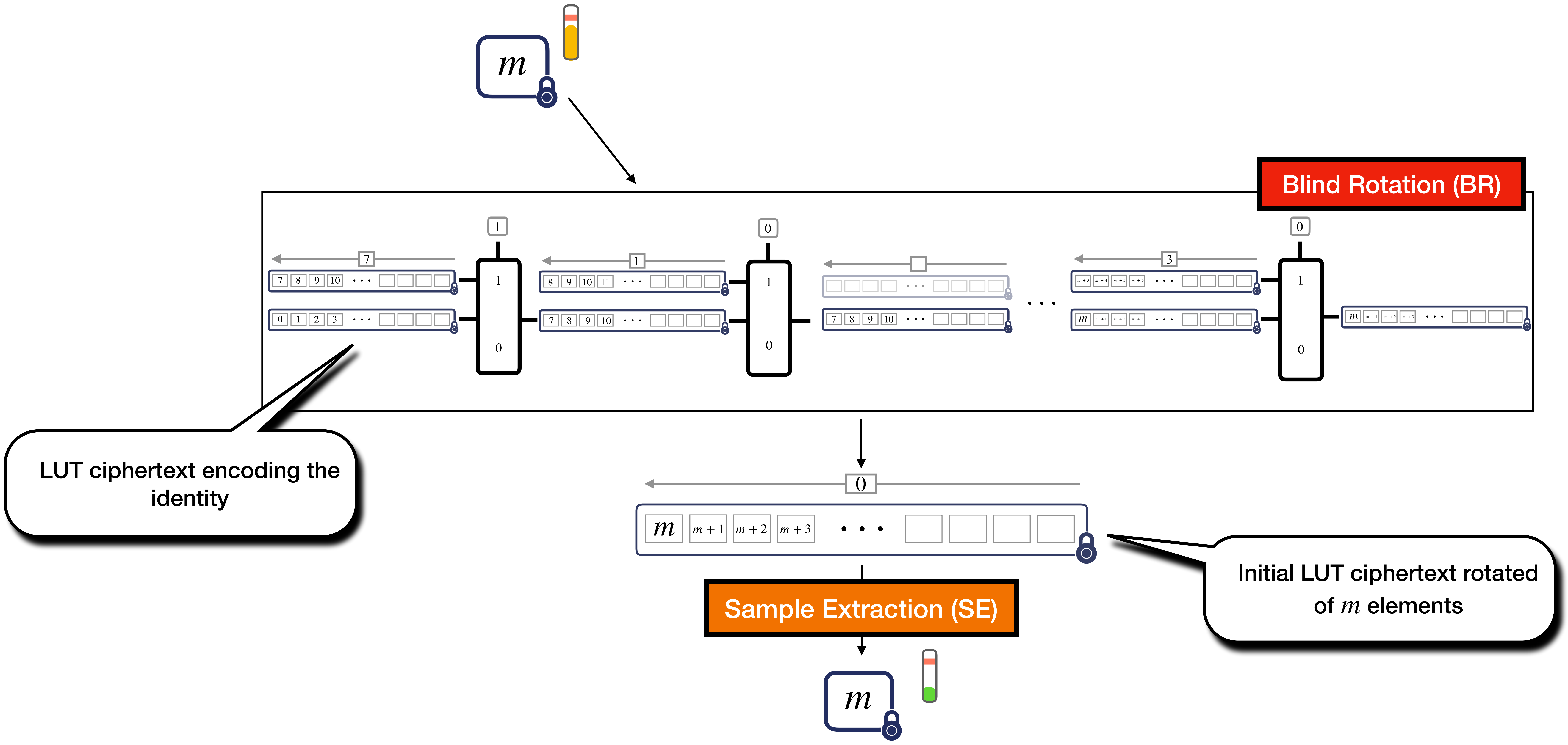
TFHE cryptosystem

Different types of ciphertexts

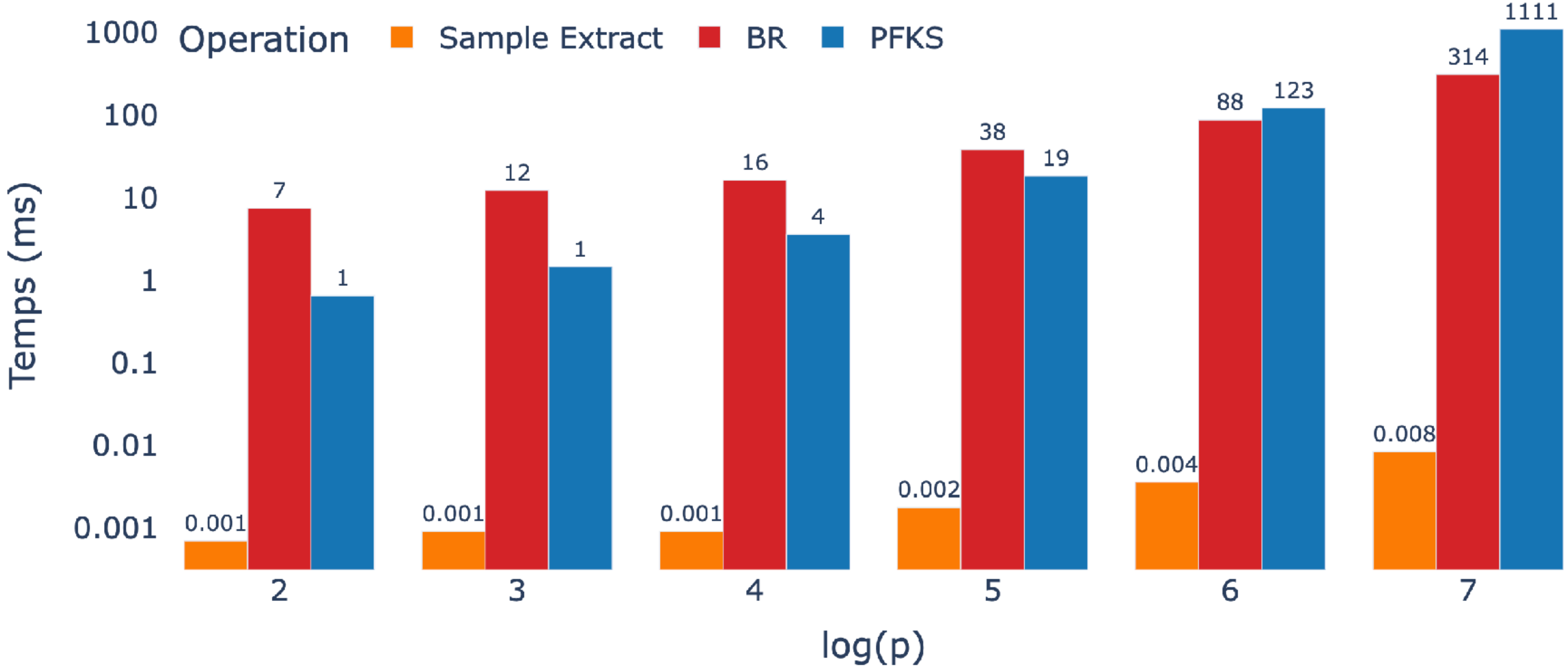


TFHE cryptosystem

High level view of the bootstrapping procedure



TFHE cryptosystem



RevoLUT : Rust Efficient Versatile Oblivious Look-Up-Table

ZAMA
TFHE-rs

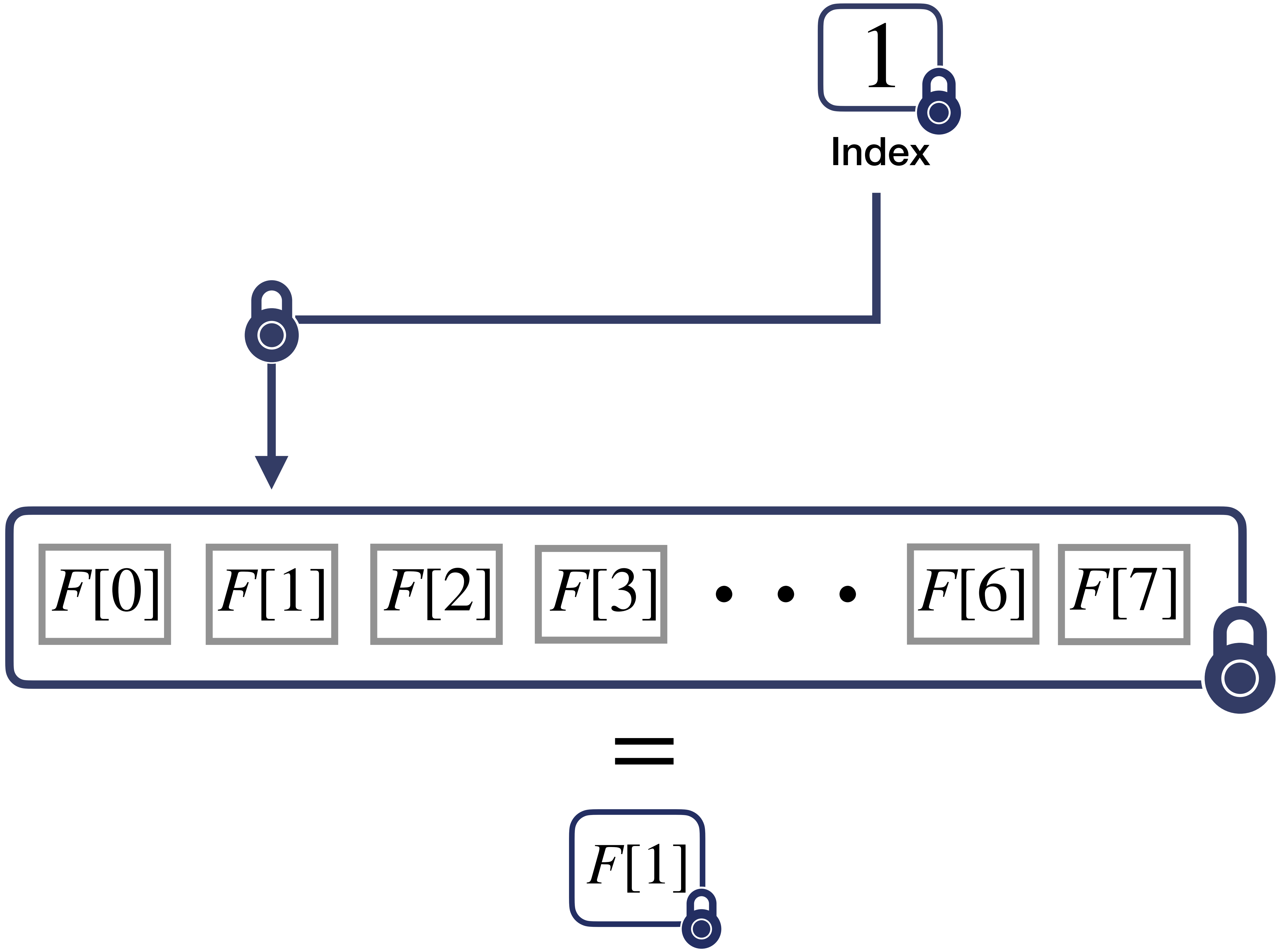
RevoLUT library

sofianeazogagh/revoLUT

How to blindly read in an encrypted array ?

Blind Array Access

$$t_{BAA} = 1 \times \text{BR}$$



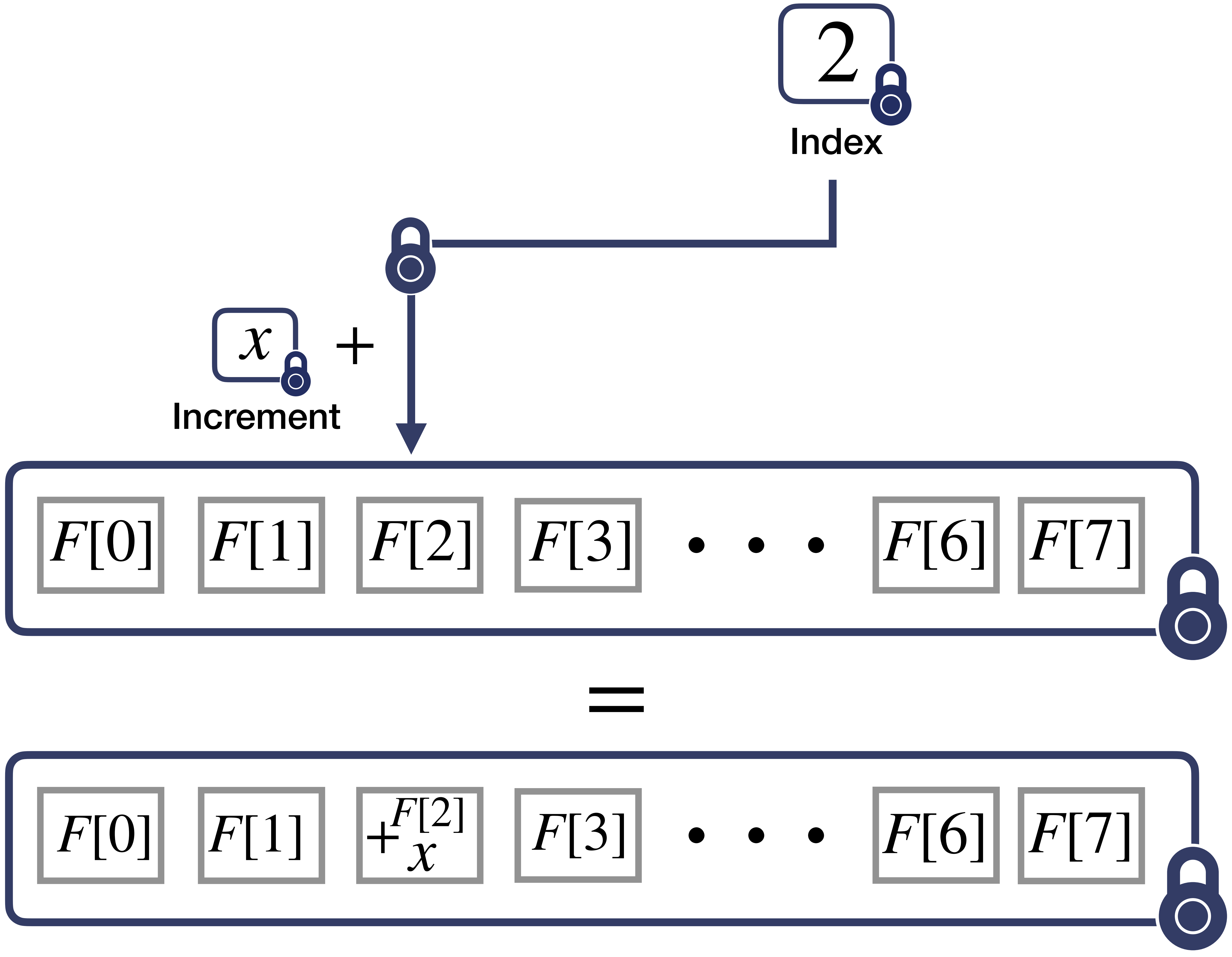
RevoLUT library

sofianeazogagh/revoLUT

How to blindly write in an encrypted array ?

Blind Array Add

$$t_{BAAdd} = 1 \times \text{BR} + 1 \times \text{PFKS}$$



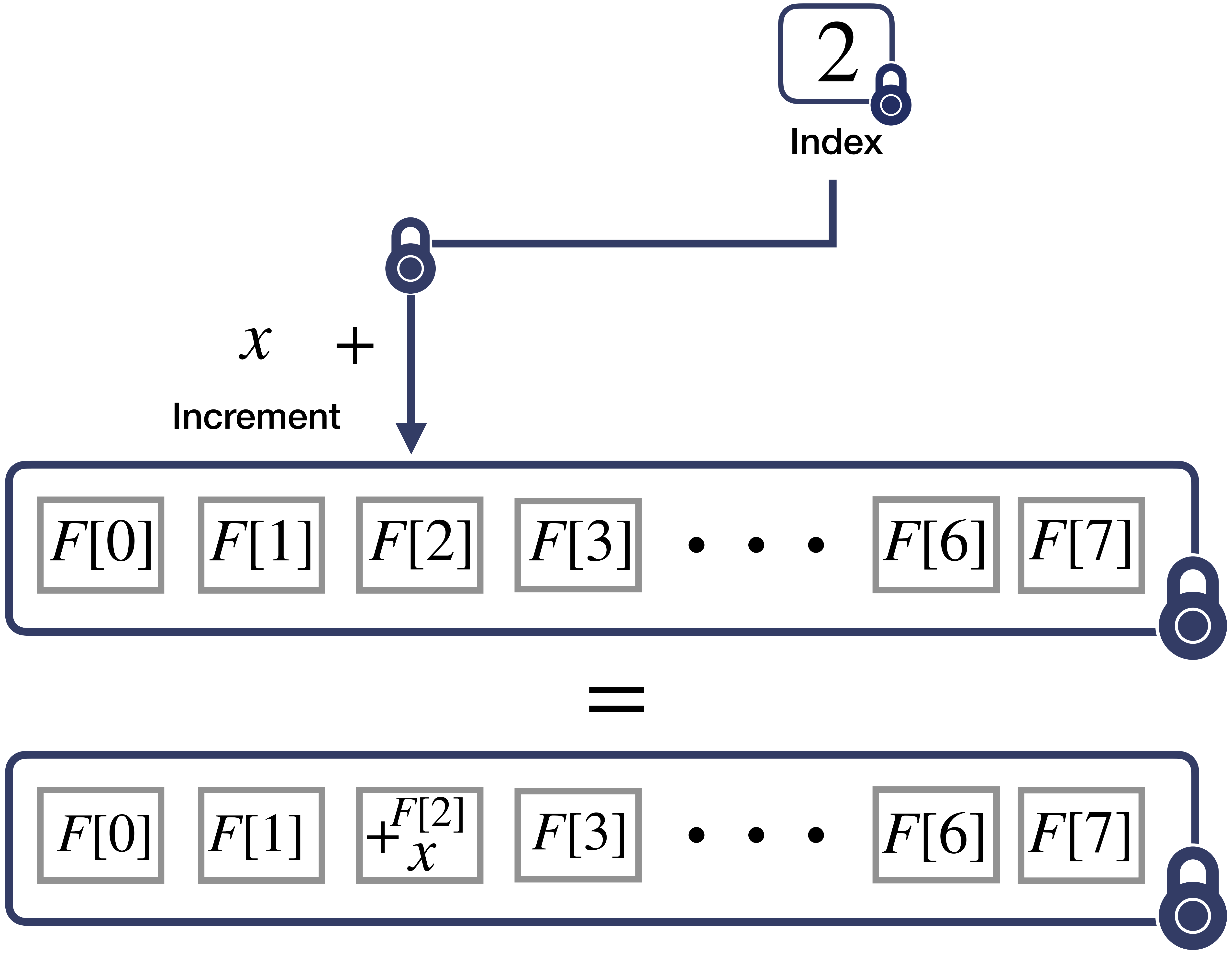
RevoLUT library

sofianeazogagh/revoLUT

How to blindly write in an encrypted array ?

Blind Array Add

$t_{BAAdd} = 1 \times \mathrm{BR}$



Blind Counting Sort (BCS)

Counting Sort

How it works in clear ?

Counting Sort

How it works in clear ?

$$\mathcal{O}(n + p)$$

number of items

items $\in \mathbb{Z}_p$

Counting Sort

How it works in clear ?

$$\mathcal{O}(n + p)$$

number of items

items $\in \mathbb{Z}_p$

$n = 8$ $\quad$ $p = 6$

3	2	5	2	4	1	5	1

Unsorted array A

Counting Sort

How it works in clear ?

$$\mathcal{O}(n + p)$$

number of items

items $\in \mathbb{Z}_p$

1) Count the occurrences : $\mathcal{O}(n)$

0	1	2	3	4	5
0	2	2	1	1	2

$n = 8$ $\qquad p = 6$

3	2	5	2	4	1	5	1

Unsorted array A

Counting Sort

How it works in clear ?

$$\mathcal{O}(n+p)$$

number of items

items $\in \mathbb{Z}_p$

$n = 8 \qquad p = 6$

3	2	5	2	4	1	5	1

Unsorted array A

1) Count the occurrences : $\mathcal{O}(n)$

0	1	2	3	4	5
0	2	2	1	1	2

2) Build the prefix sum : $\mathcal{O}(p)$

0	1	2	3	4	5
0	2	4	5	6	8

Counting Sort

How it works in clear ?

$$\mathcal{O}(n+p)$$

number of items

items $\in \mathbb{Z}_p$

$n = 8$ $\quad$ $p = 6$

3	2	5	2	4	1	5	1

Unsorted array A

1) Count the occurrences : $\mathcal{O}(n)$

0	1	2	3	4	5
0	2	2	1	1	2

2) Build the prefix sum : $\mathcal{O}(p)$

0	1	2	3	4	5
0	2	4	5	6	8

<u>Destination range</u> :

- $[0,2[$ should be filled of 1s
- $[2,4[$ should be filled of 2s
- $[4,5[$ should be filled of 3s
- $[5,6[$ should be filled of 4s
- $[6,8[$ should be filled of 5s

Counting Sort

How it works in clear ?

$$\mathcal{O}(n + p)$$

number of items

items $\in \mathbb{Z}_p$

1) Count the occurrences : $\mathcal{O}(n)$

0	1	2	3	4	5
0	2	2	1	1	2

$n = 8 \qquad p = 6$

3	2	5	2	4	1	5	1

Unsorted array A

2) Build the prefix sum : $\mathcal{O}(p)$

0	1	2	3	4	5
0	2	4	5	6	8

<u>Destination range</u> :

- $[0,2[$ should be filled of 1s
- $[2,4[$ should be filled of 2s
- $[4,5[$ should be filled of 3s
- $[5,6[$ should be filled of 4s
- $[6,8[$ should be filled of 5s

3) Rebuild the sorted array : $\mathcal{O}(n)$

0	1	2	3	4	5	6	7

Sorted array

Counting Sort

How it works in clear ?

$$\mathcal{O}(n+p)$$

number of items

items $\in \mathbb{Z}_p$

$n = 8$ $p = 6$

3	2	5	2	4	1	5	1

Unsorted array A

1) Count the occurrences : $\mathcal{O}(n)$

0	1	2	3	4	5
0	2	2	1	1	2

2) Build the prefix sum : $\mathcal{O}(p)$

0	1	2	3	4	5
0	2	4	5	6	8

<u>Destination range</u> :

- $[0,2[$ should be filled of 1s
- $[2,4[$ should be filled of 2s
- $[4,5[$ should be filled of 3s
- $[5,6[$ should be filled of 4s
- $[6,8[$ should be filled of 5s

3) Rebuild the sorted array : $\mathcal{O}(n)$

0	1	2	3	4	5	6	7
1	1	2	2	3	4	5	5

Sorted array

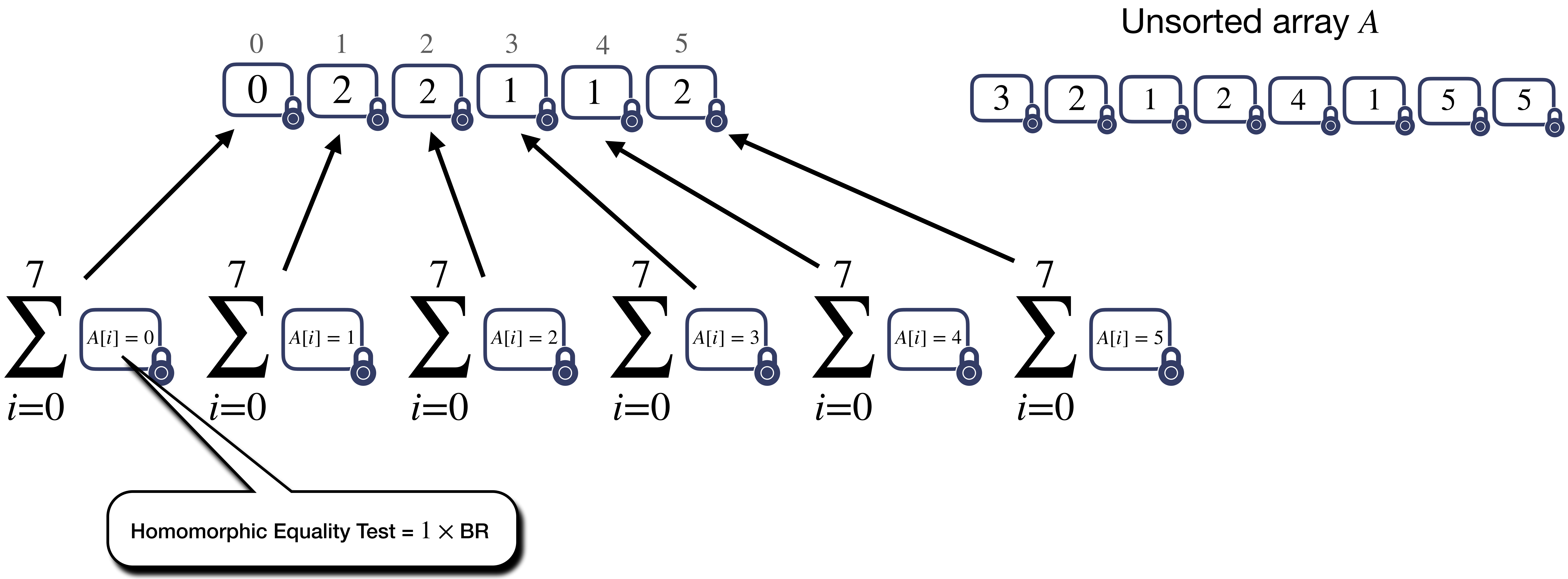
Counting Sort

The challenge of bringing it to the encrypted world

Counting Sort

The challenge of bringing it to the encrypted world

Count the occurences : $\mathcal{O}(n^2)$

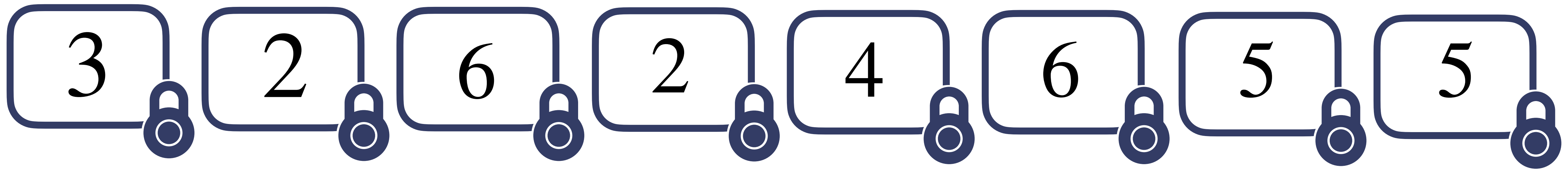


Blind Counting Sort (BCS)

How to blindly count occurrences ?

Blind Counting Sort (BCS)

How to blindly count occurrences ?



Blind Count

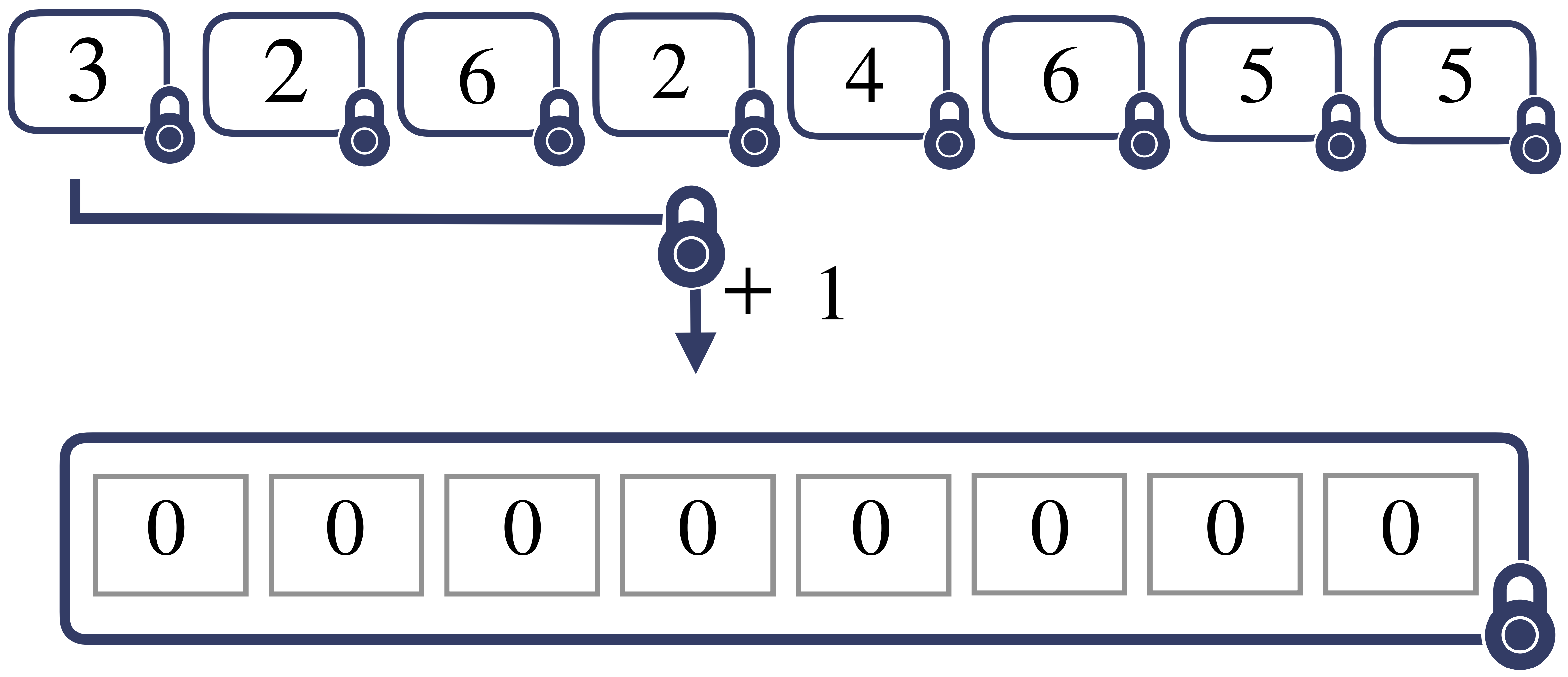
$\mathcal{O}(n)$

Blind Counting Sort (BCS)

How to blindly count occurrences ?

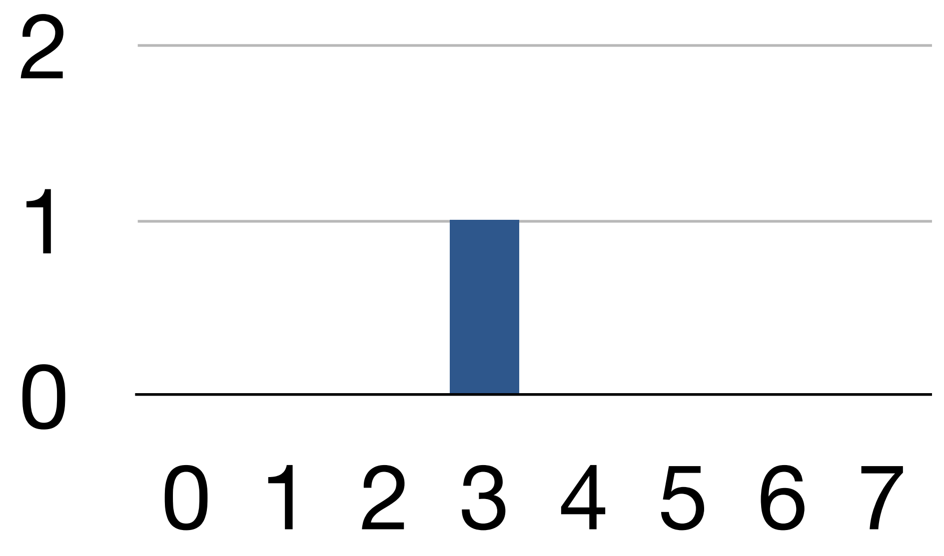
Blind Count

$\mathcal{O}(n)$



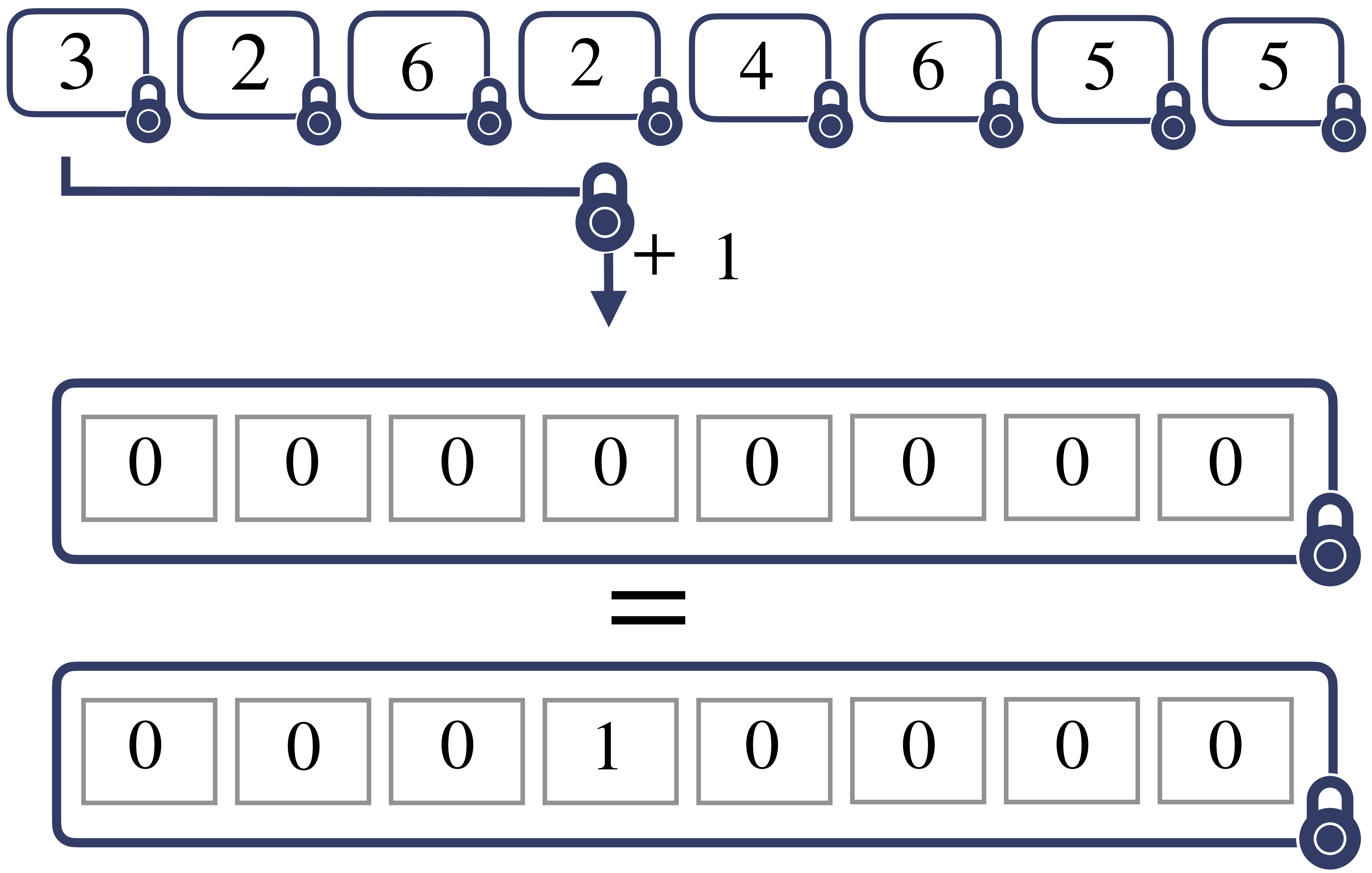
Blind Counting Sort (BCS)

How to blindly count occurrences ?



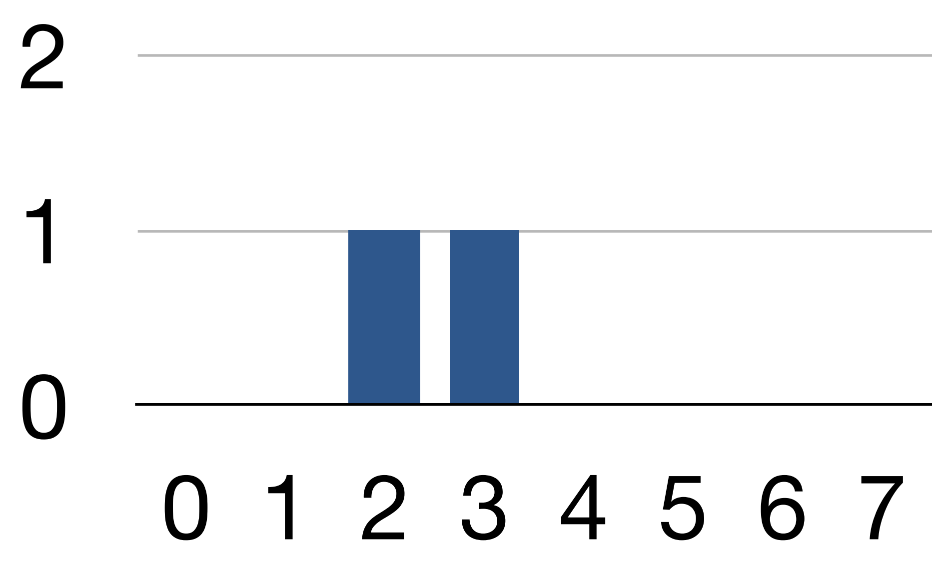
Blind Count

$\mathcal{O}(n)$



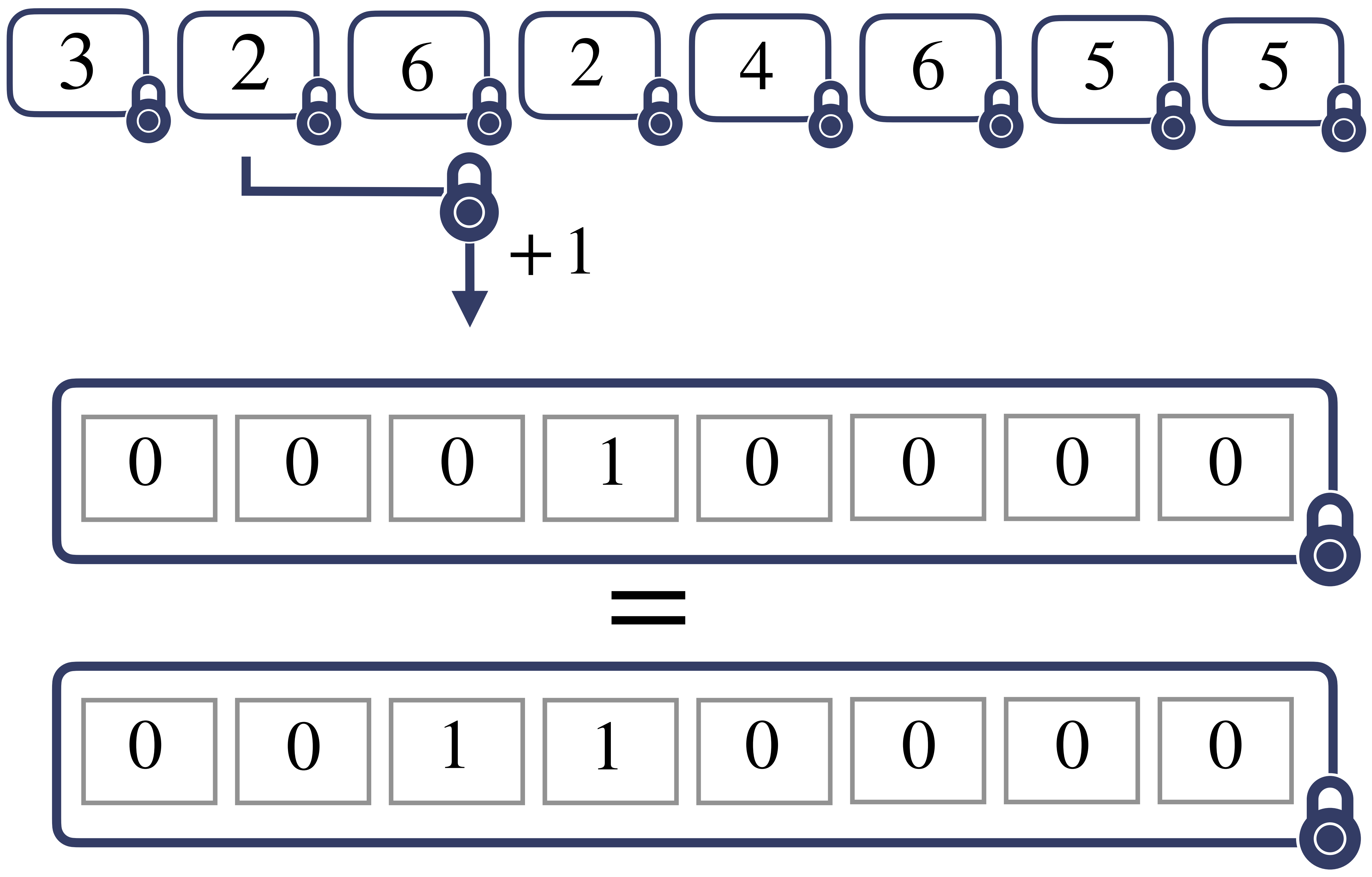
Blind Counting Sort (BCS)

How to blindly count occurrences ?



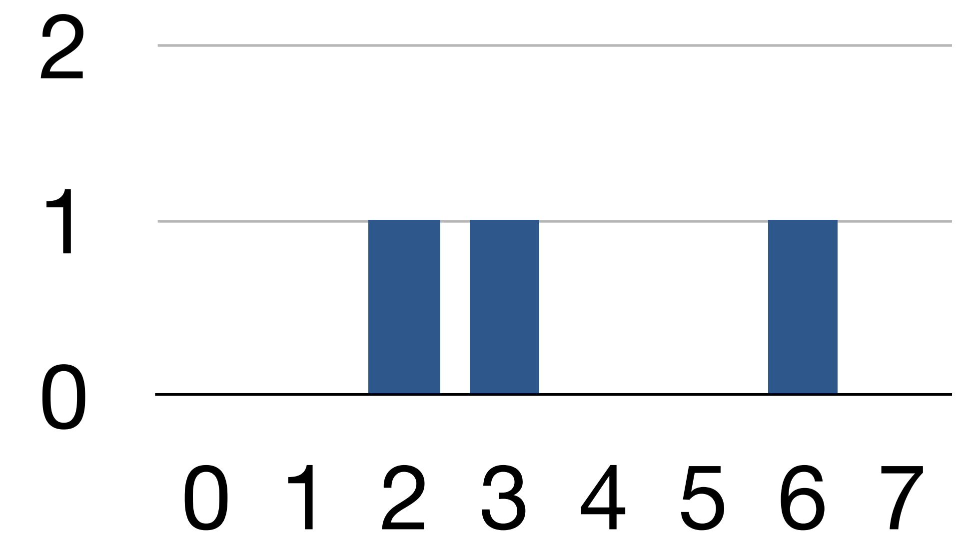
Blind Count

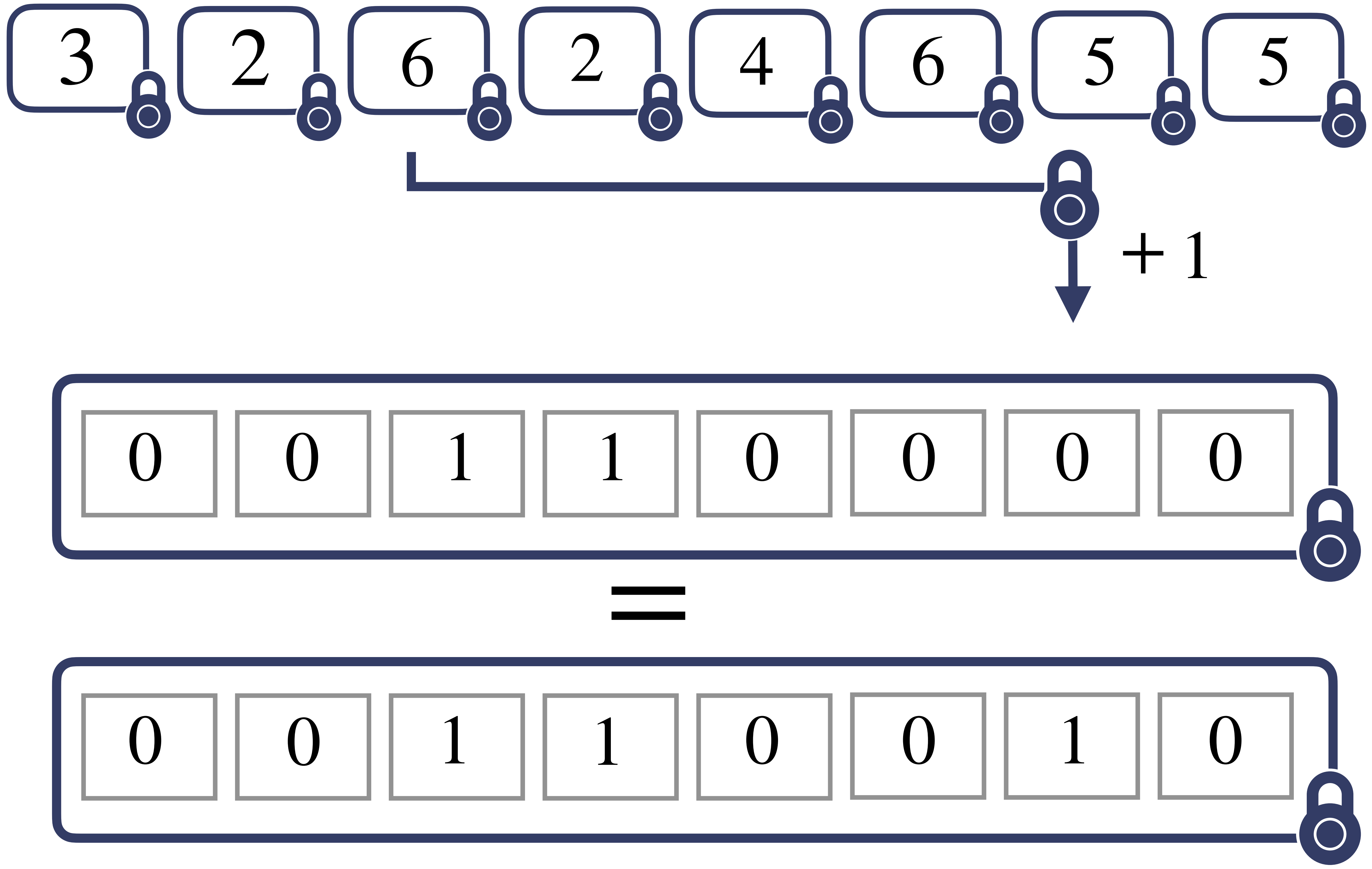
$\mathcal{O}(n)$



Blind Counting Sort (BCS)

How to blindly count occurrences ?



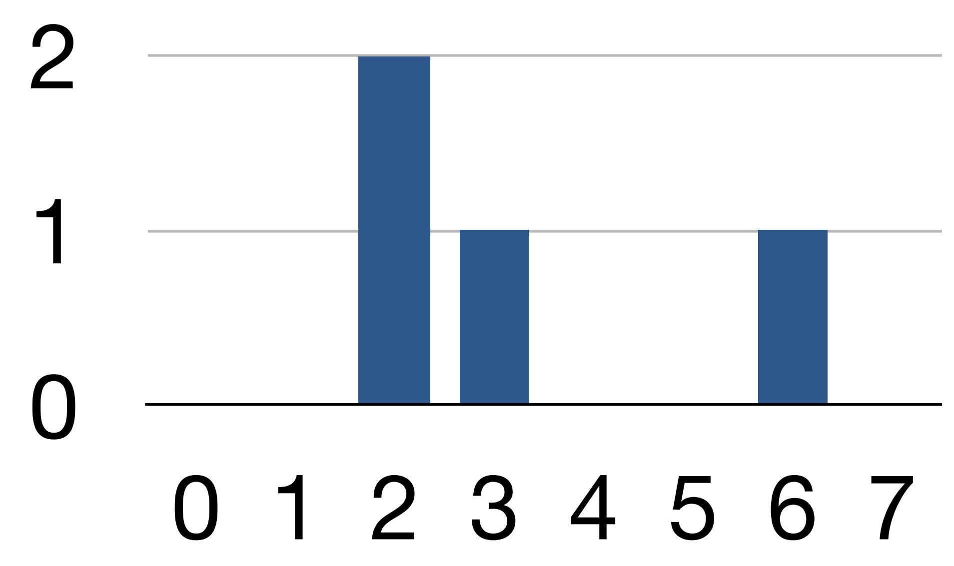


Blind Count

$\mathcal{O}(n)$

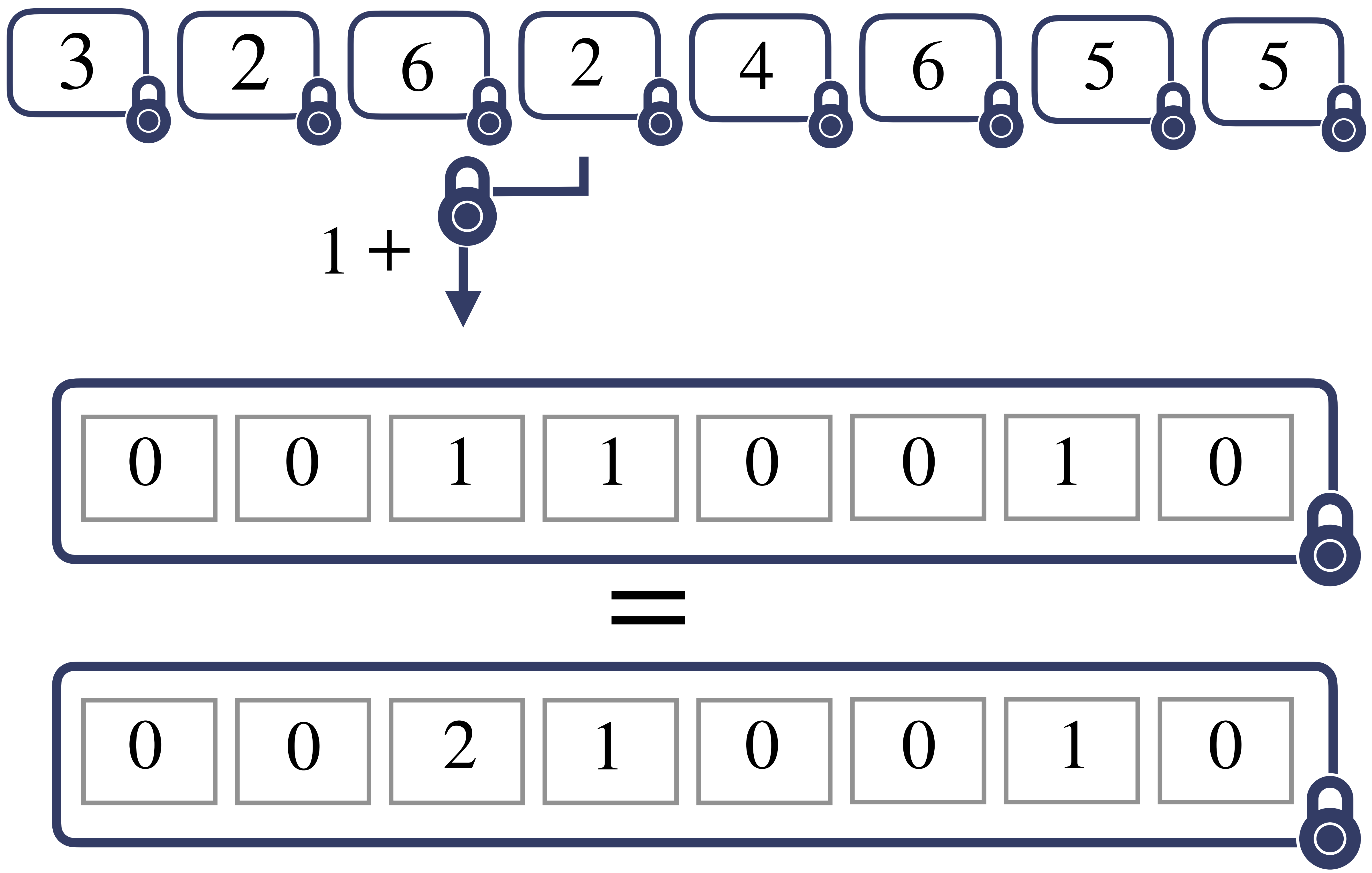
Blind Counting Sort (BCS)

How to blindly count occurrences ?



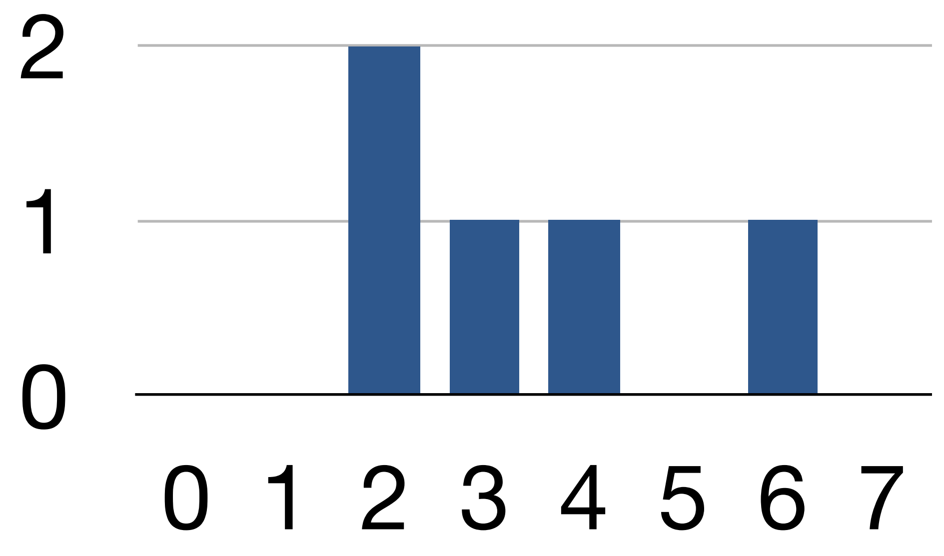
Blind Count

$\mathcal{O}(n)$



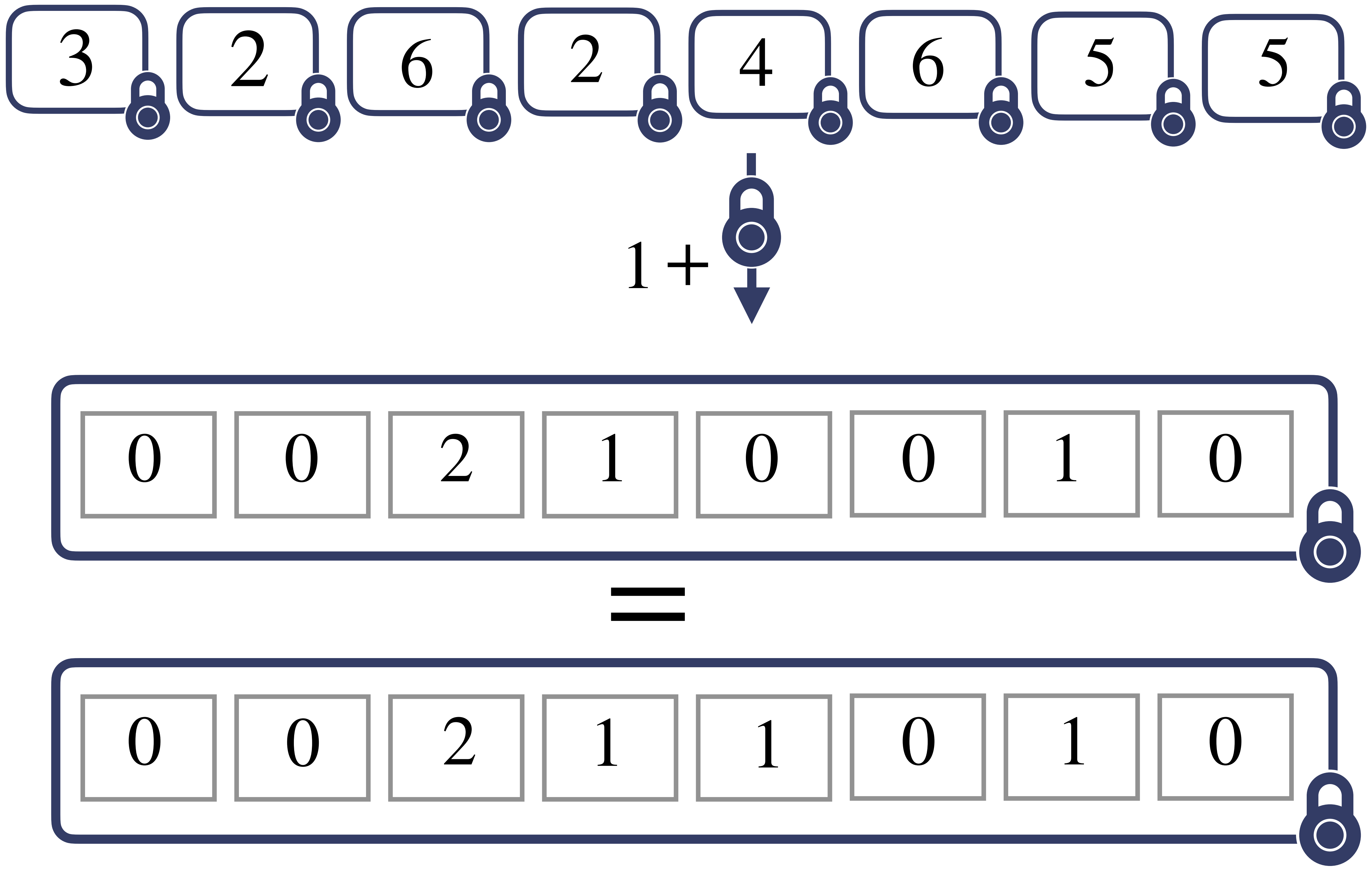
Blind Counting Sort (BCS)

How to blindly count occurrences ?



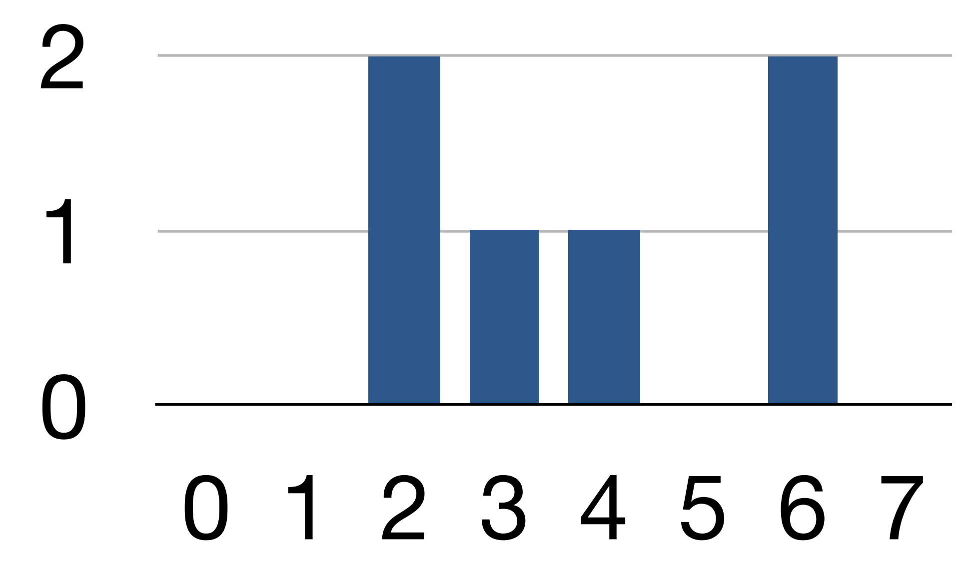
Blind Count

$\mathcal{O}(n)$



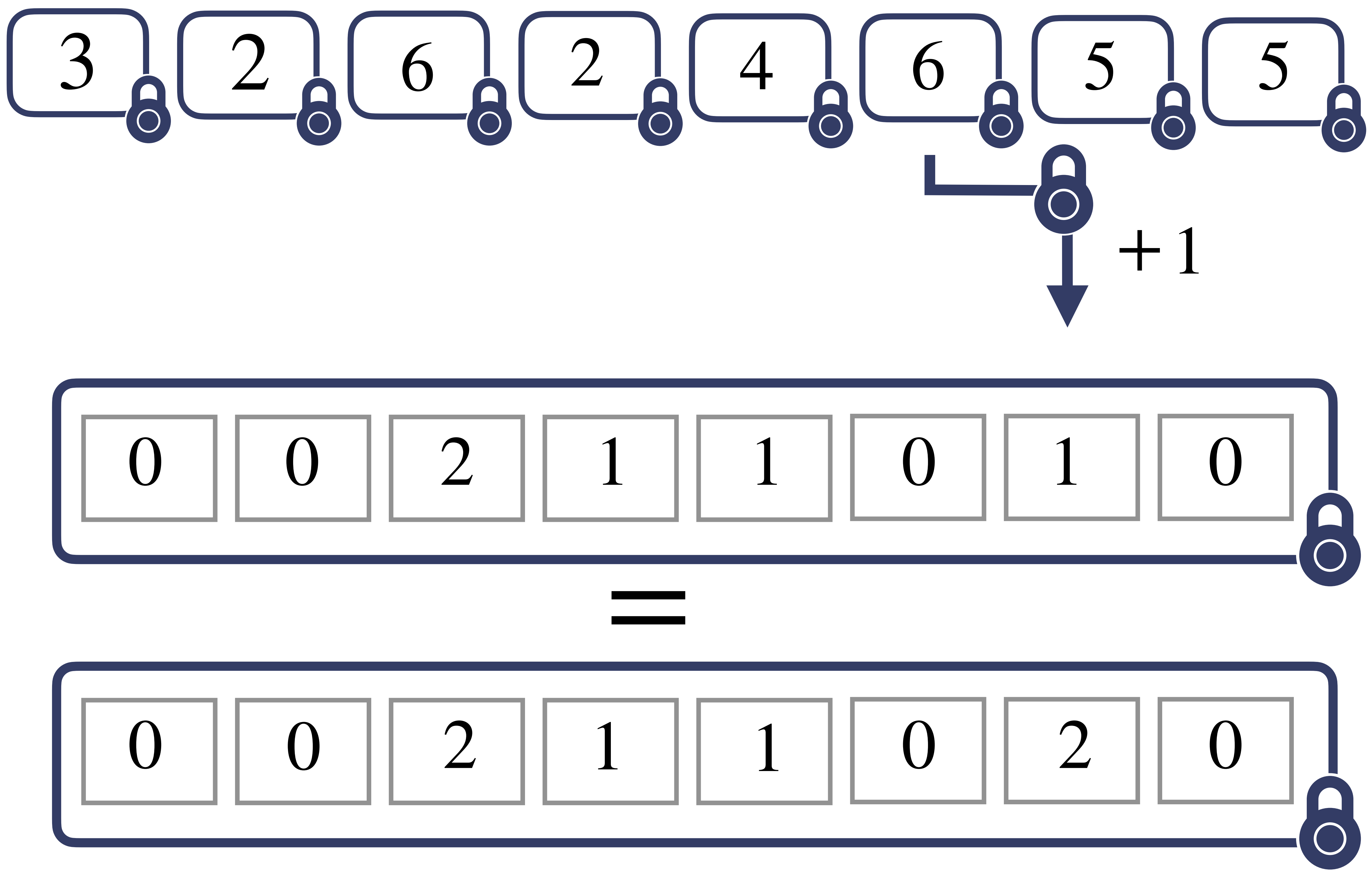
Blind Counting Sort (BCS)

How to blindly count occurrences ?



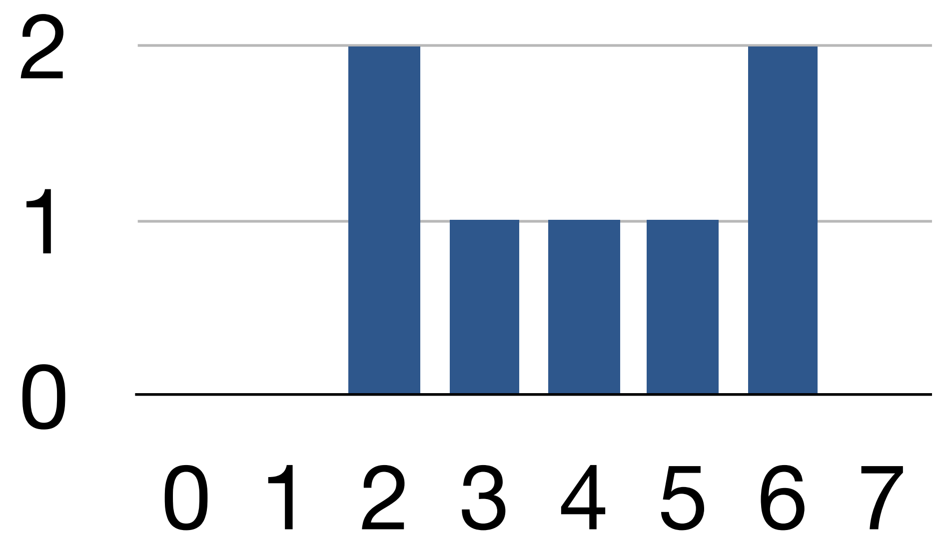
Blind Count

$\mathcal{O}(n)$



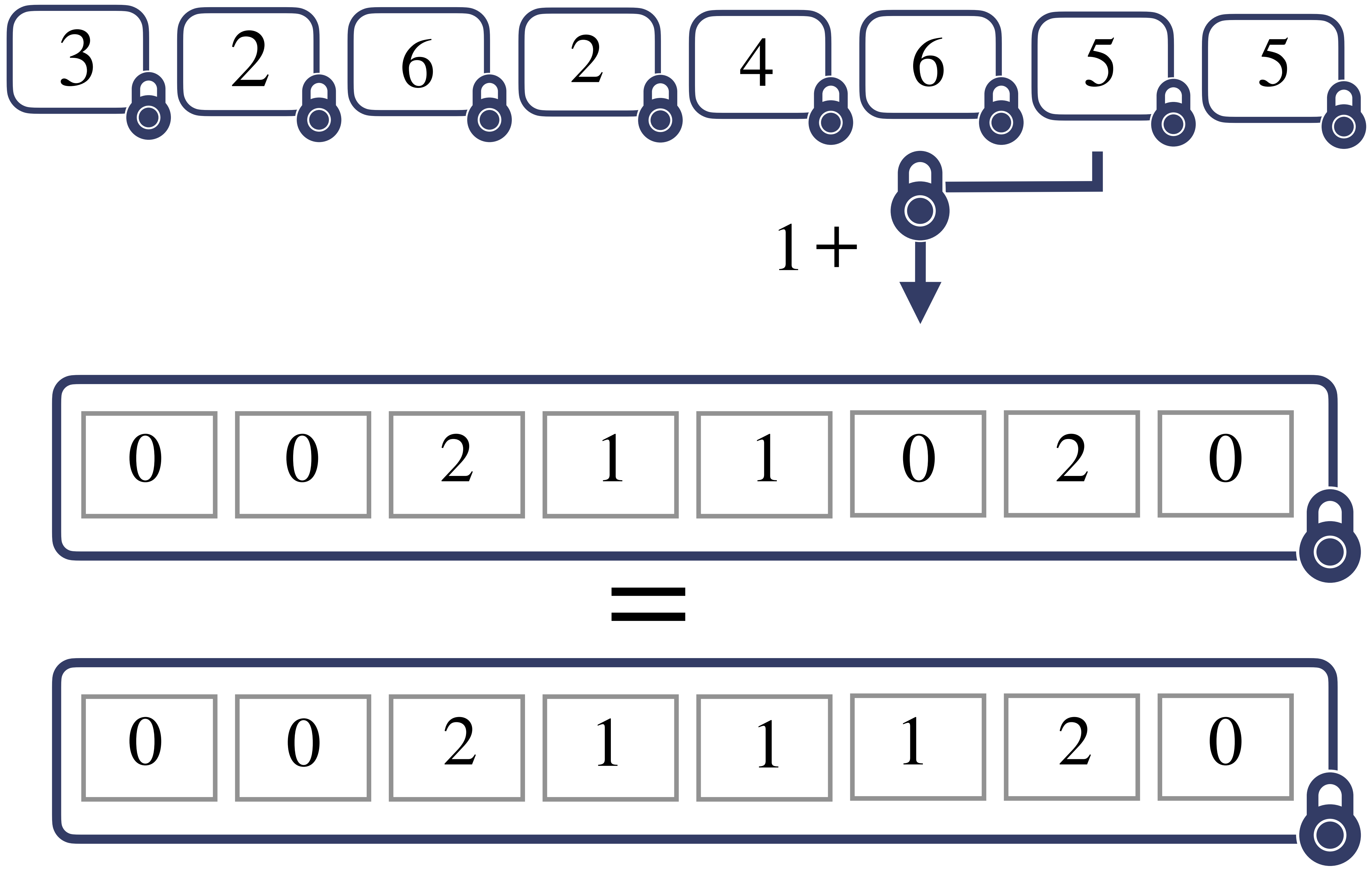
Blind Counting Sort (BCS)

How to blindly count occurrences ?



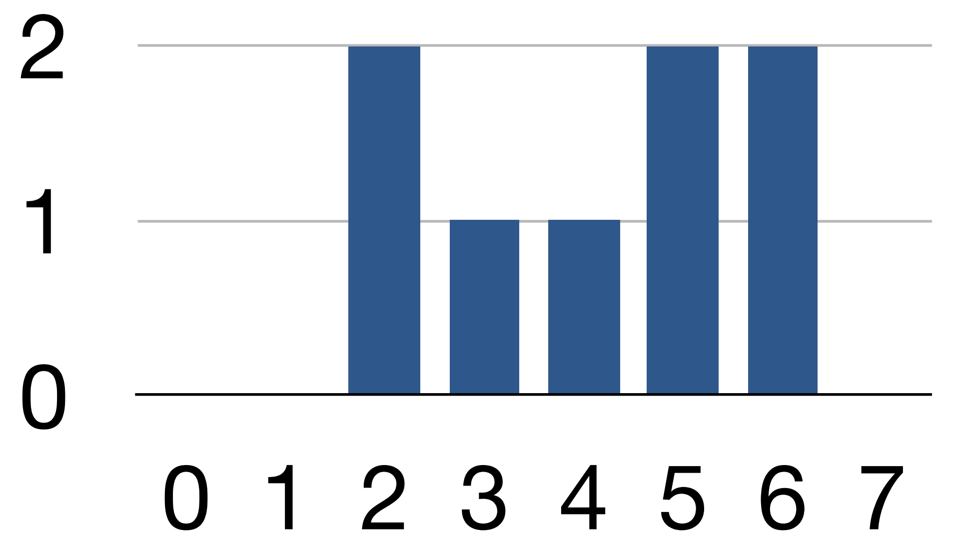
Blind Count

$\mathcal{O}(n)$



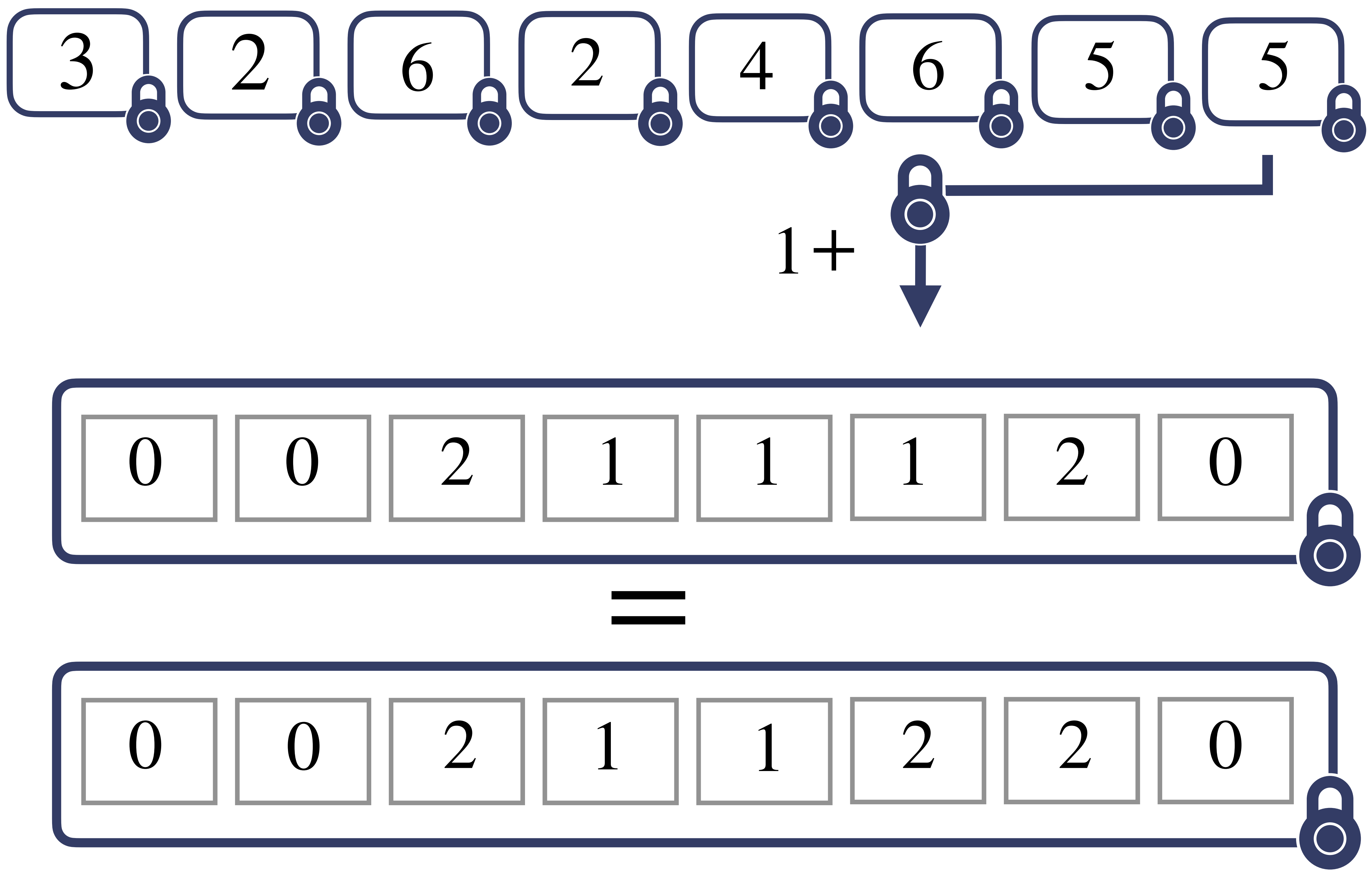
Blind Counting Sort (BCS)

How to blindly count occurrences ?



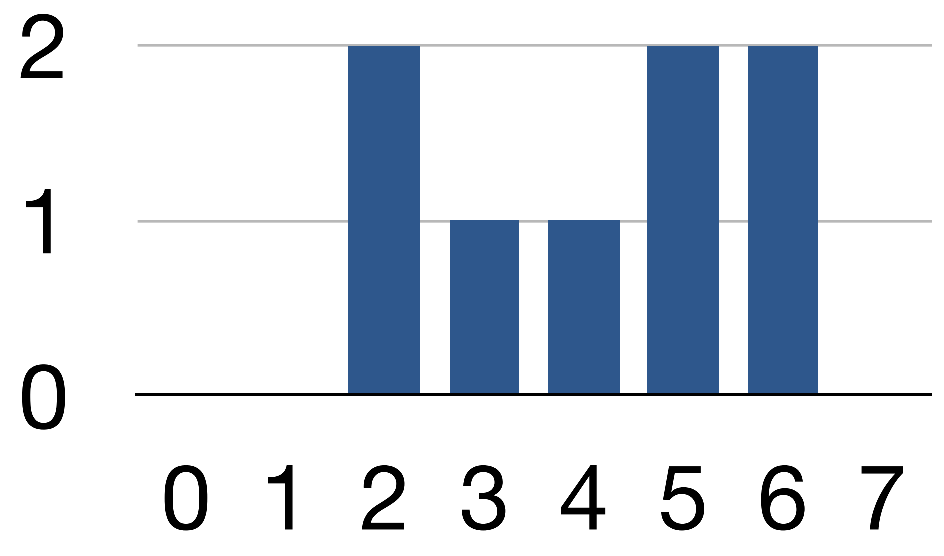
Blind Count

$\mathcal{O}(n)$



Blind Counting Sort (BCS)

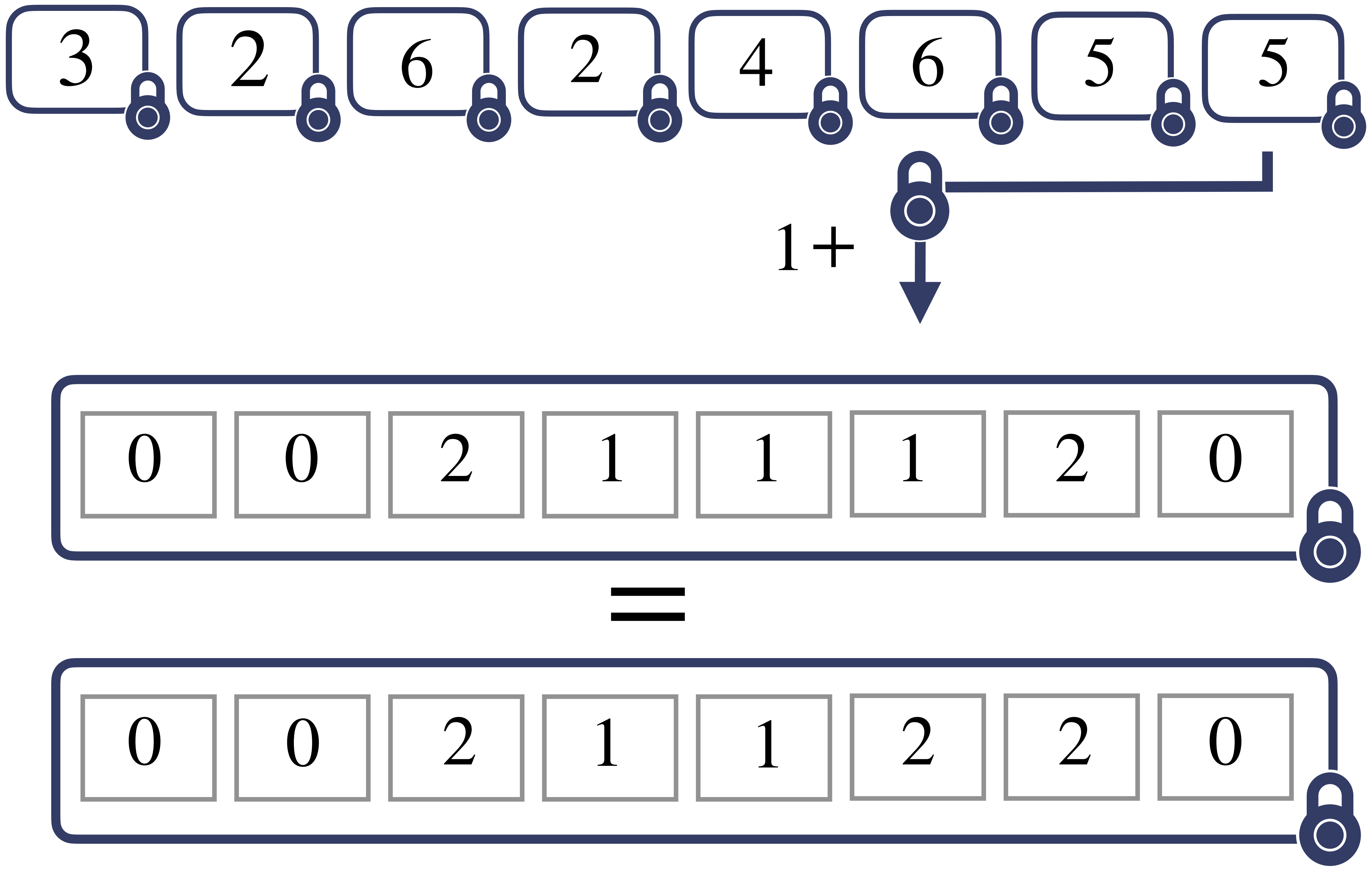
How to blindly count occurrences ?



Blind Count

$\mathcal{O}(n)$

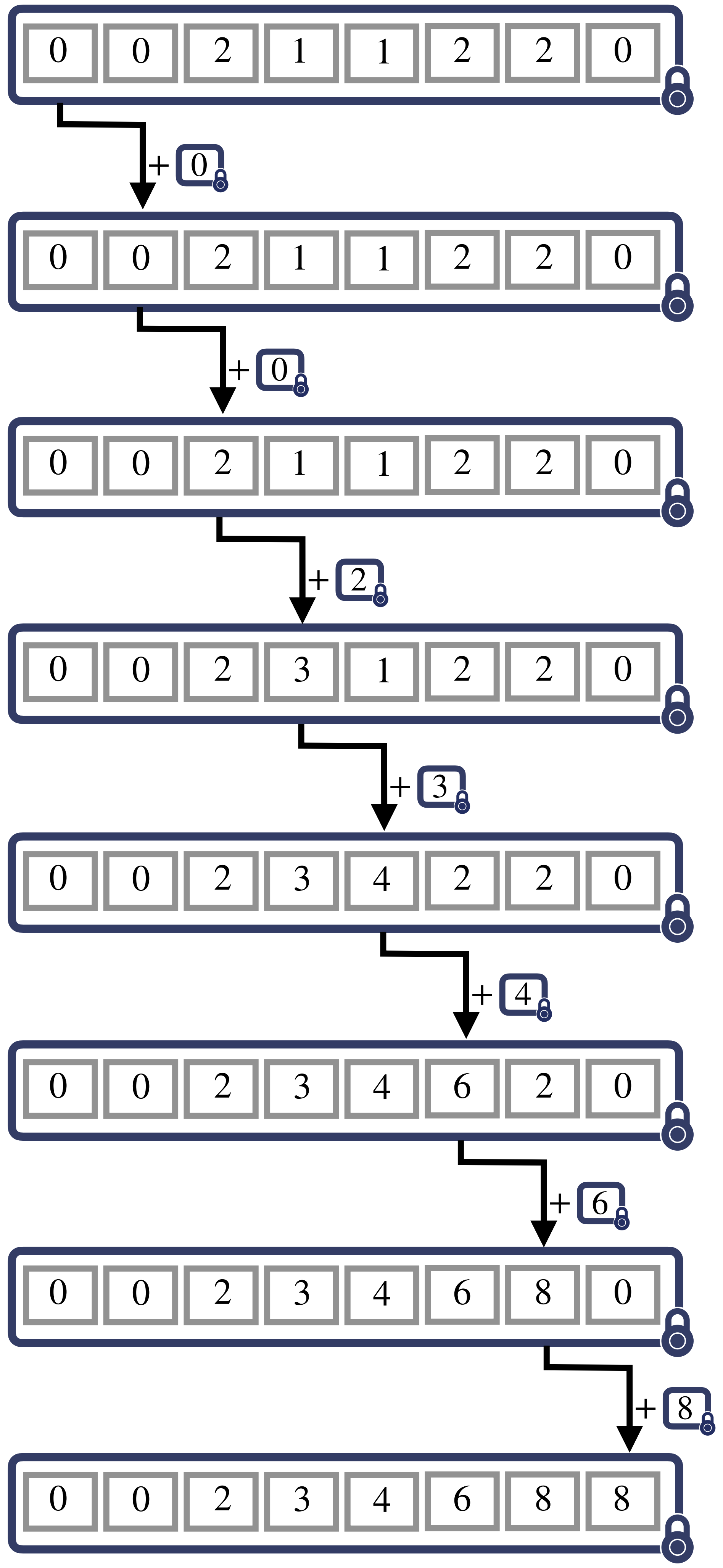
$n \times \text{BR}$

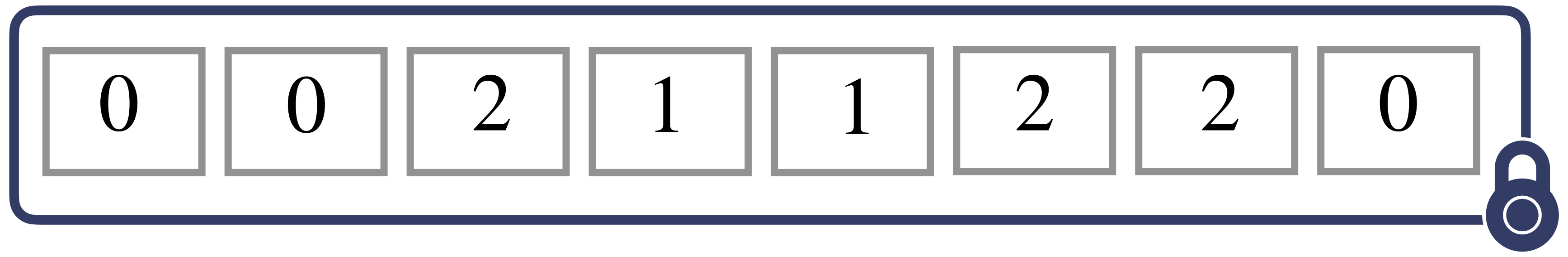


Counts

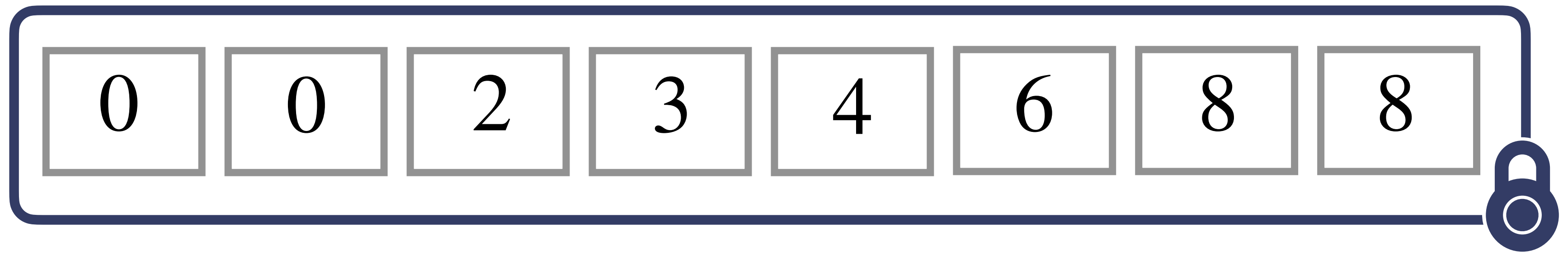
Blind Counting Sort (BCS)

Build a Prefix Sum





Counts



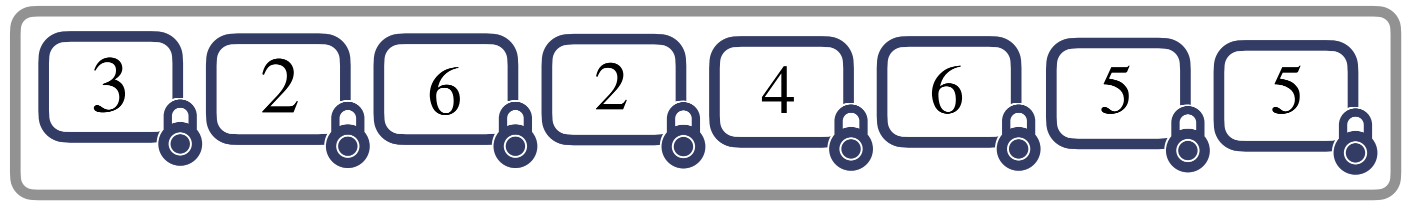
Prefix Sum

Blind Counting Sort (BCS)

Build the sorted array

Blind Counting Sort (BCS)

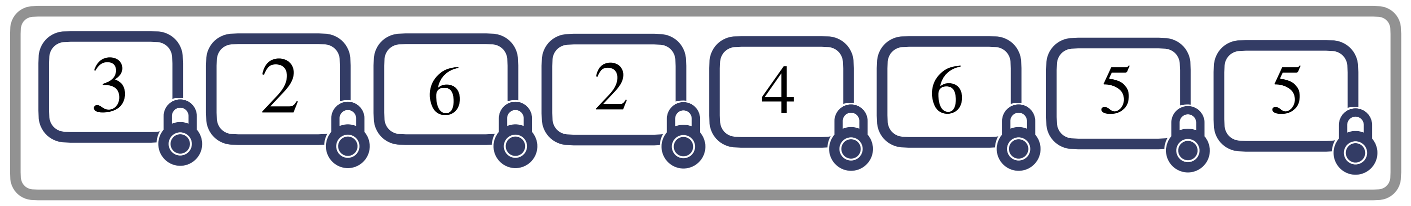
Build the sorted array



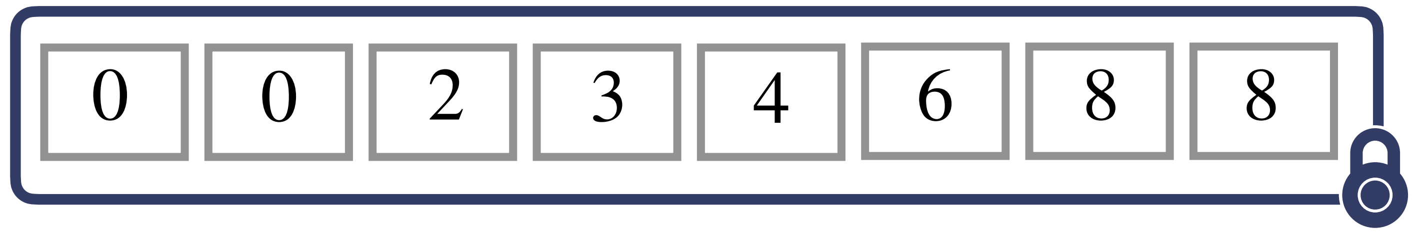
Unsorted array

Blind Counting Sort (BCS)

Build the sorted array



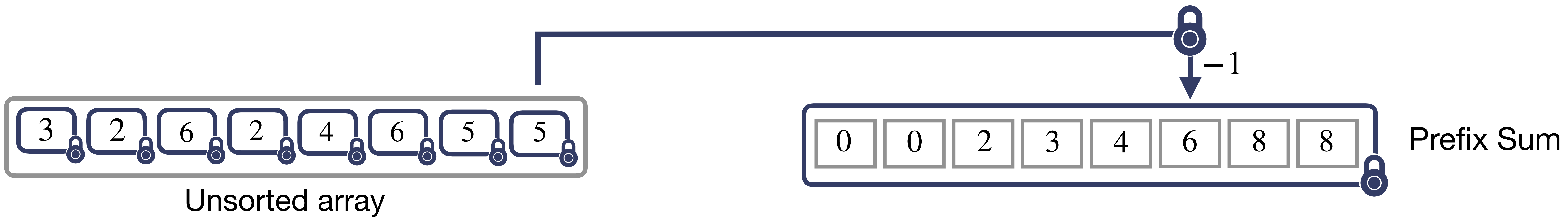
Unsorted array



Prefix Sum

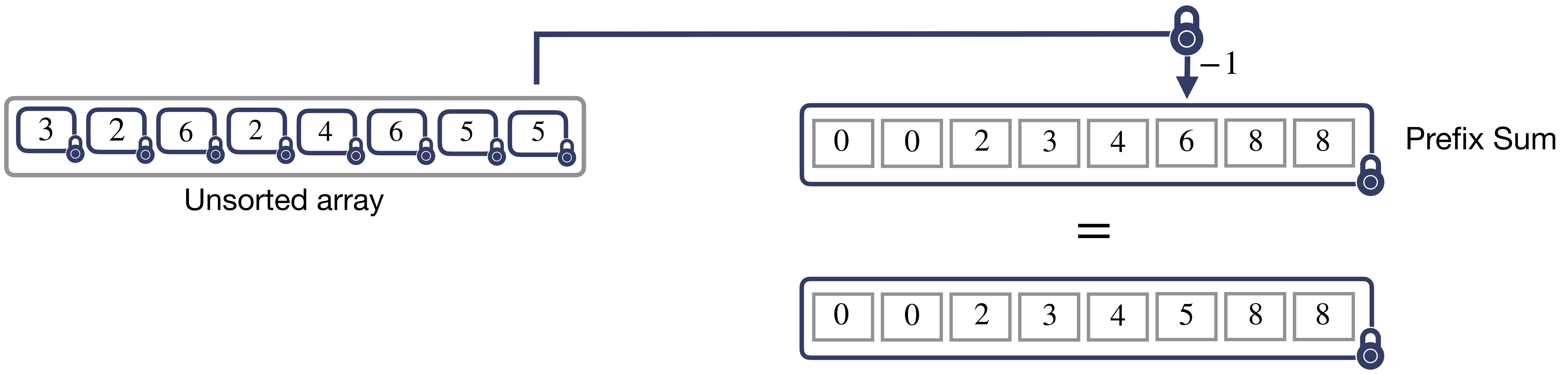
Blind Counting Sort (BCS)

Build the sorted array



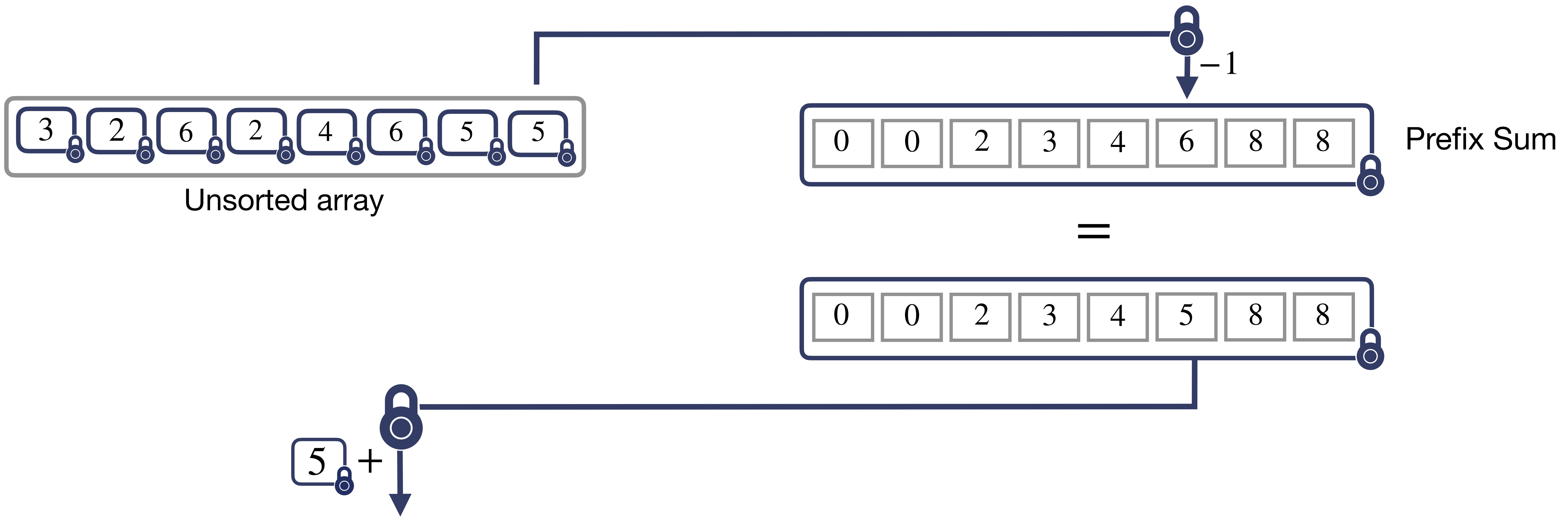
Blind Counting Sort (BCS)

Build the sorted array



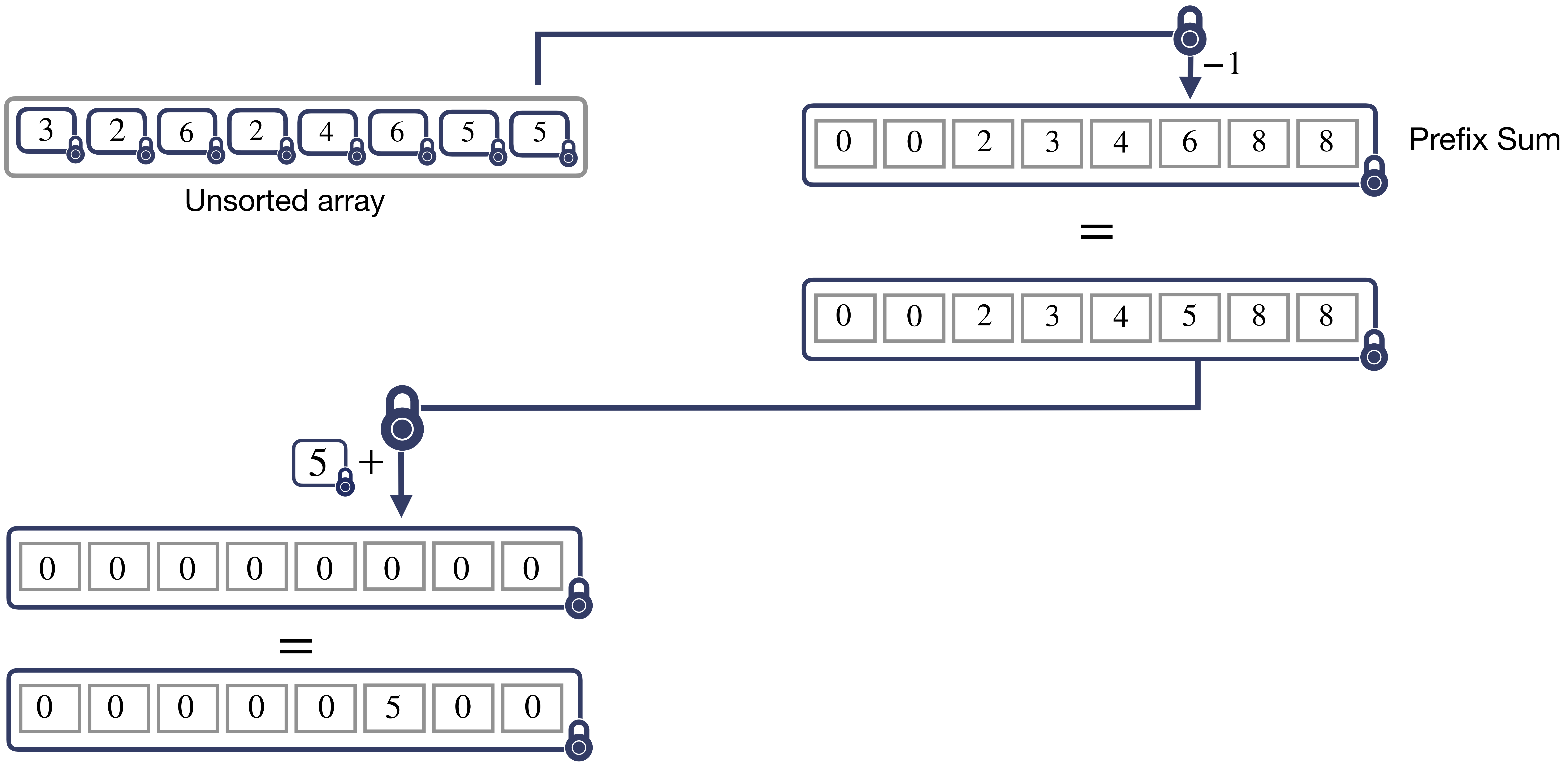
Blind Counting Sort (BCS)

Build the sorted array



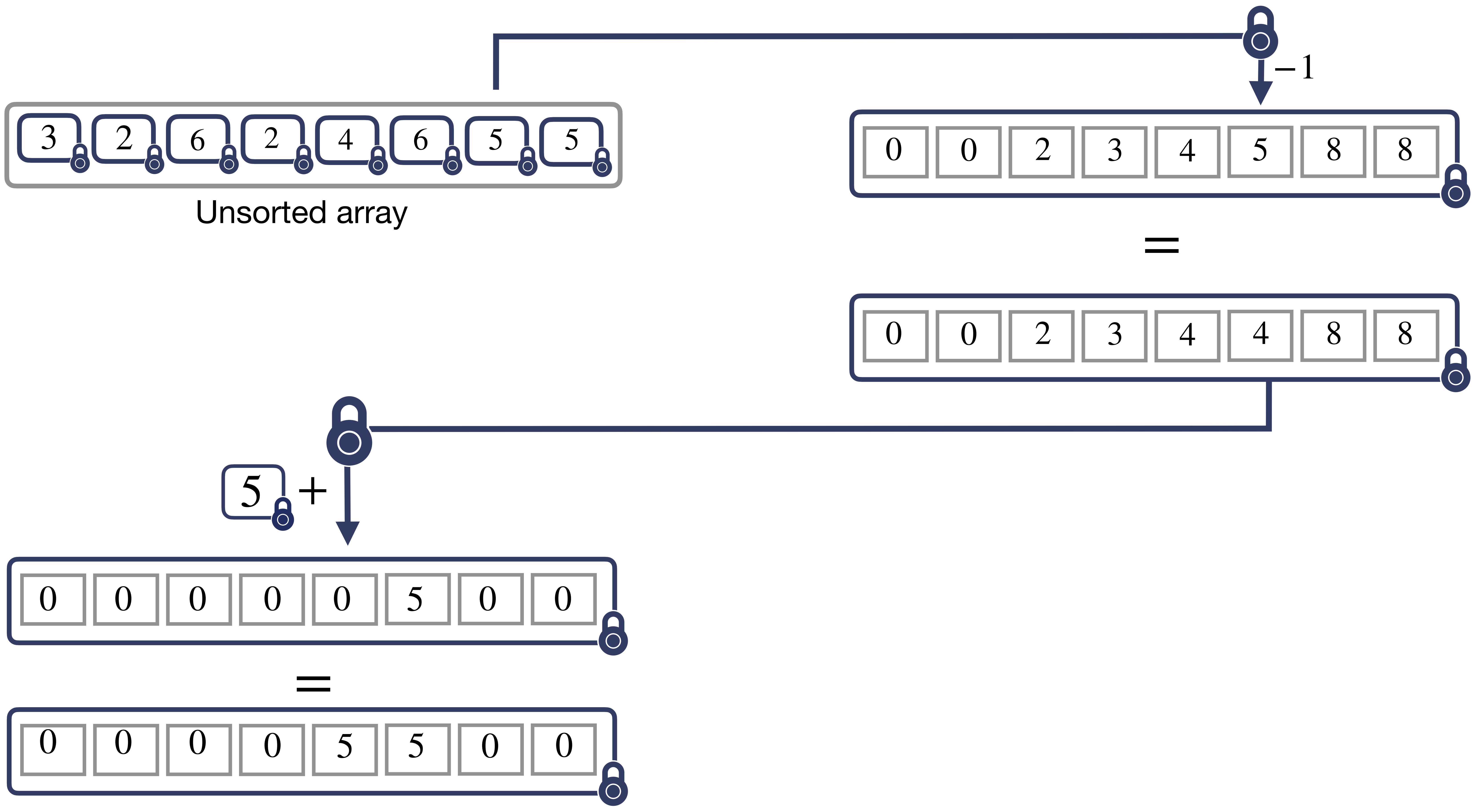
Blind Counting Sort (BCS)

Build the sorted array



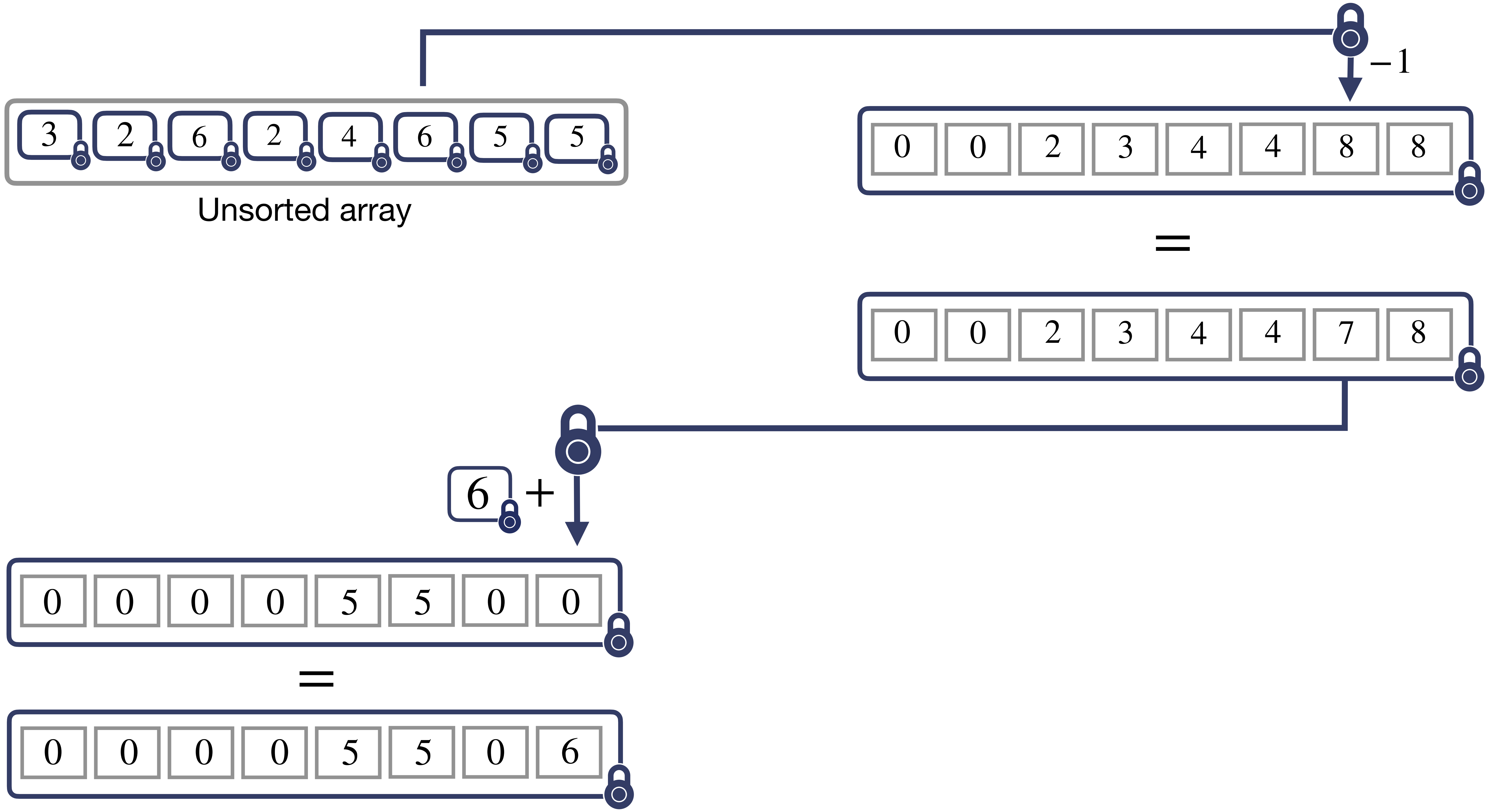
Blind Counting Sort (BCS)

Build the sorted array



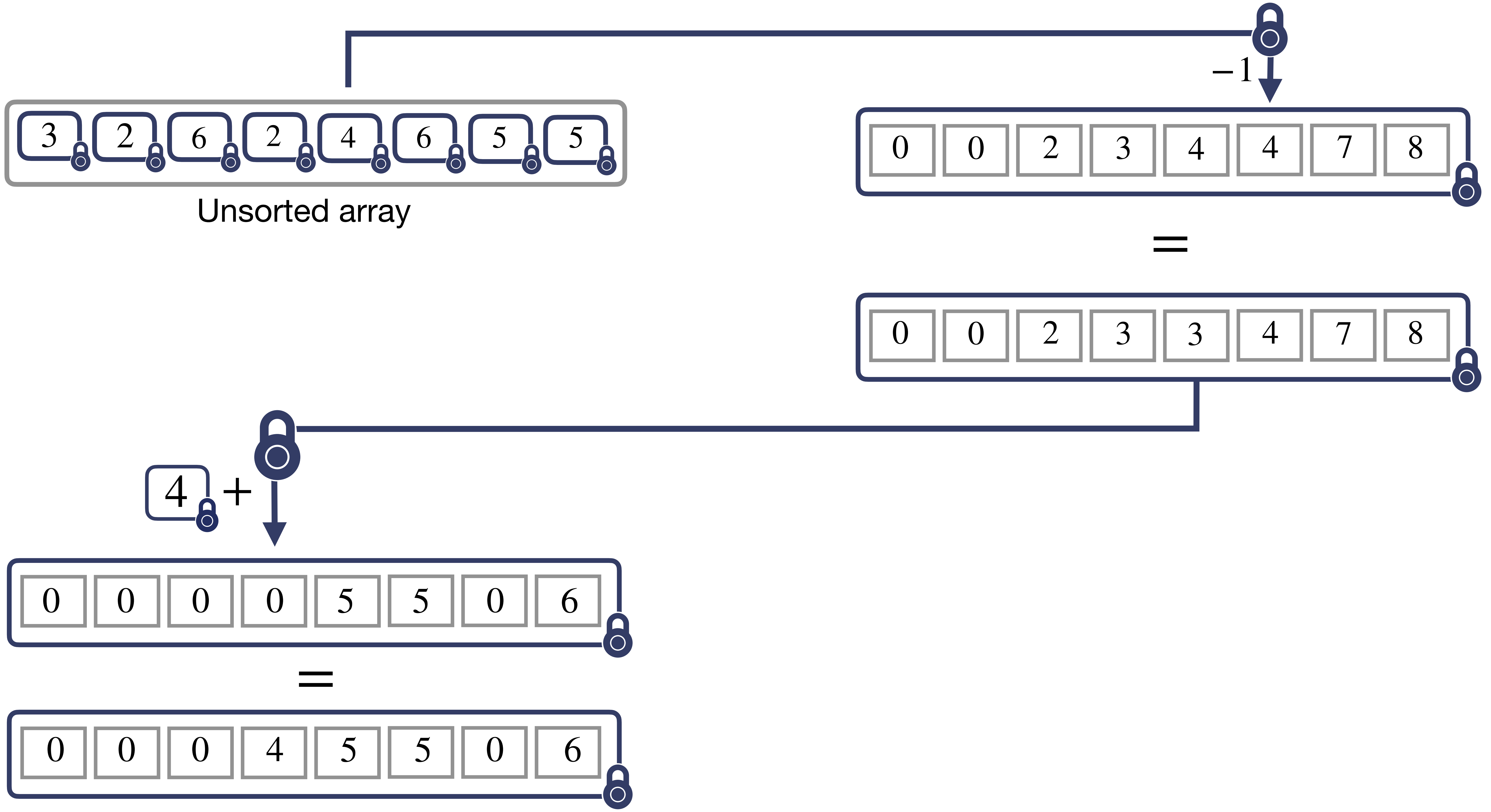
Blind Counting Sort (BCS)

Build the sorted array



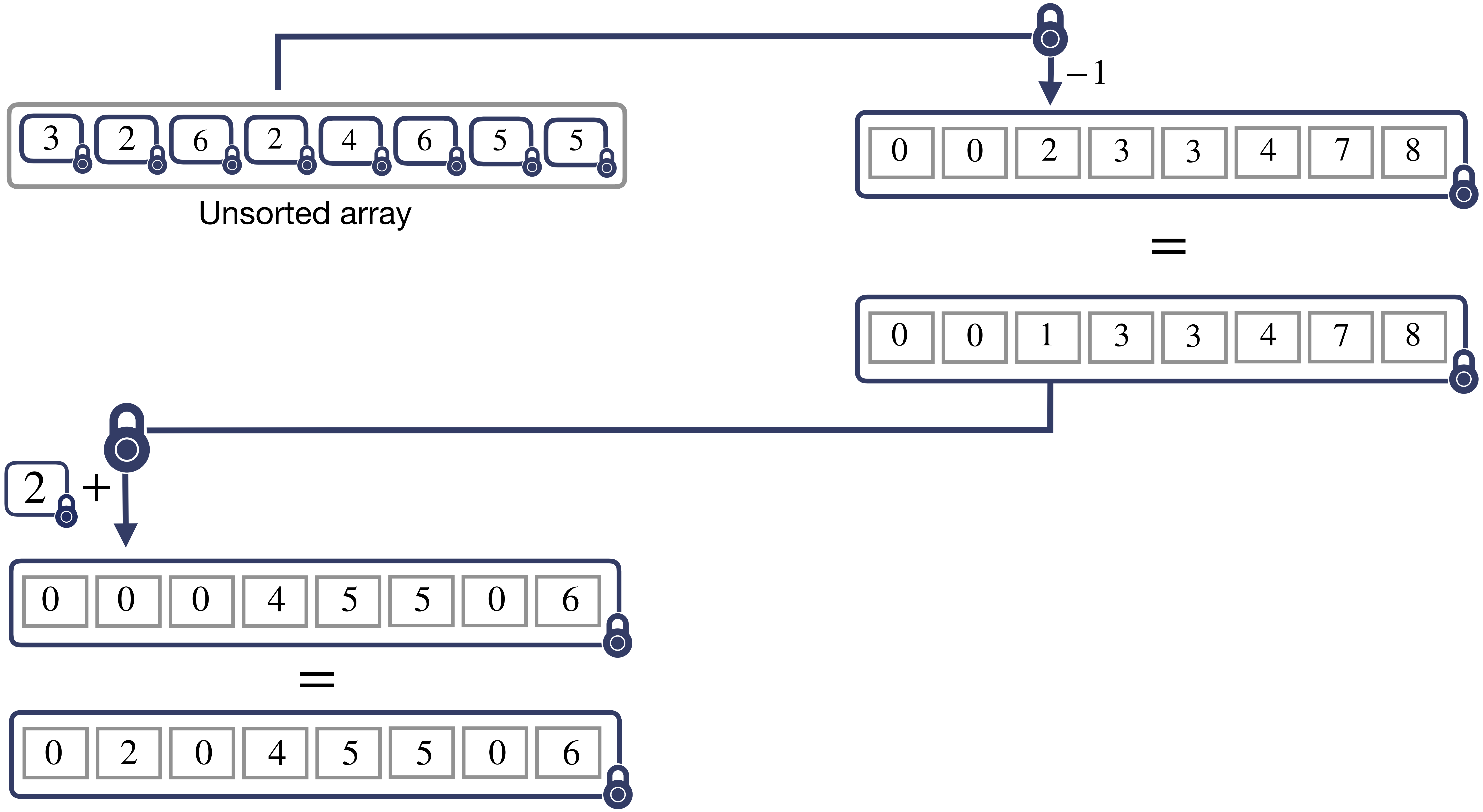
Blind Counting Sort (BCS)

Build the sorted array



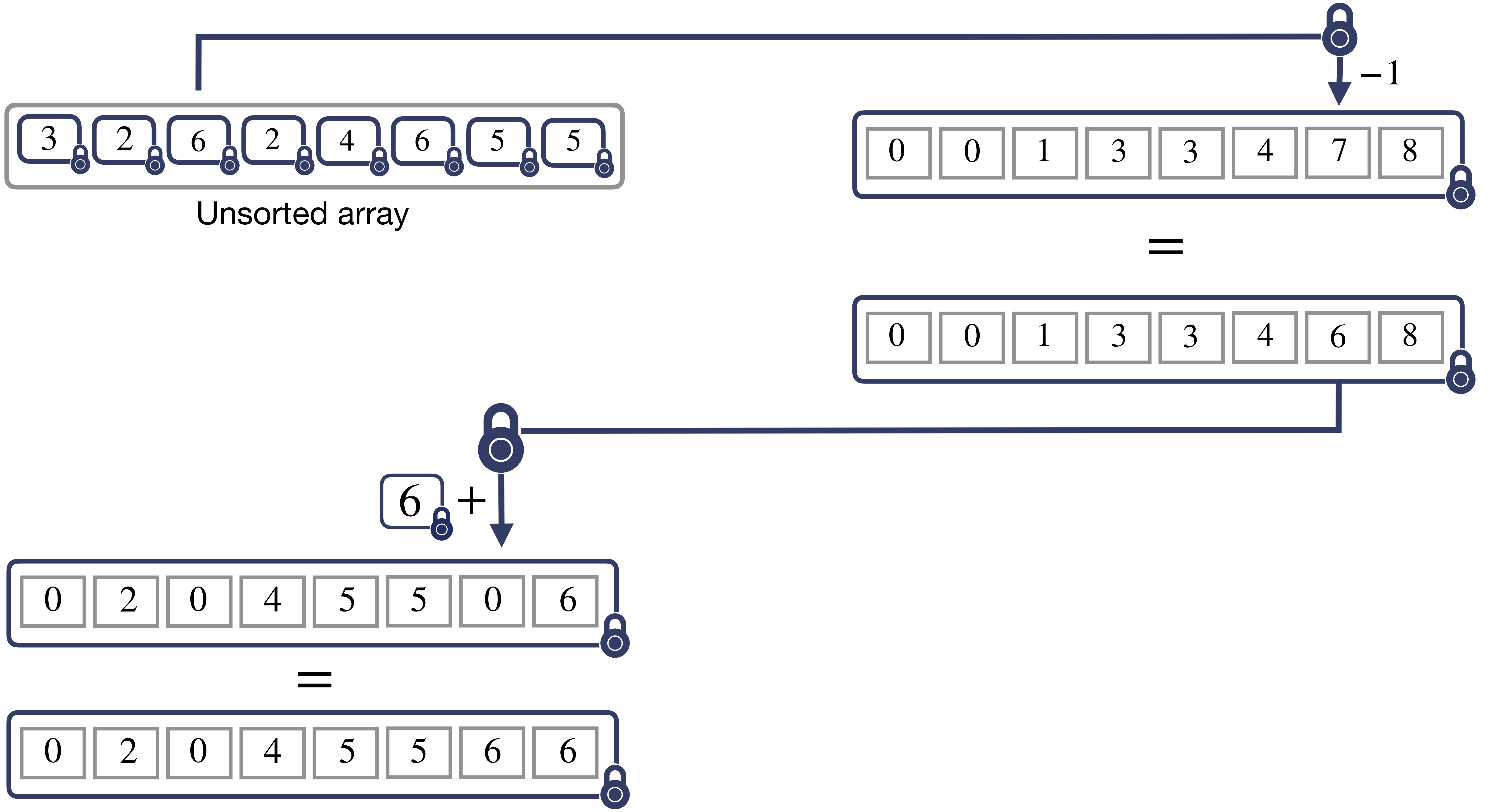
Blind Counting Sort (BCS)

Build the sorted array



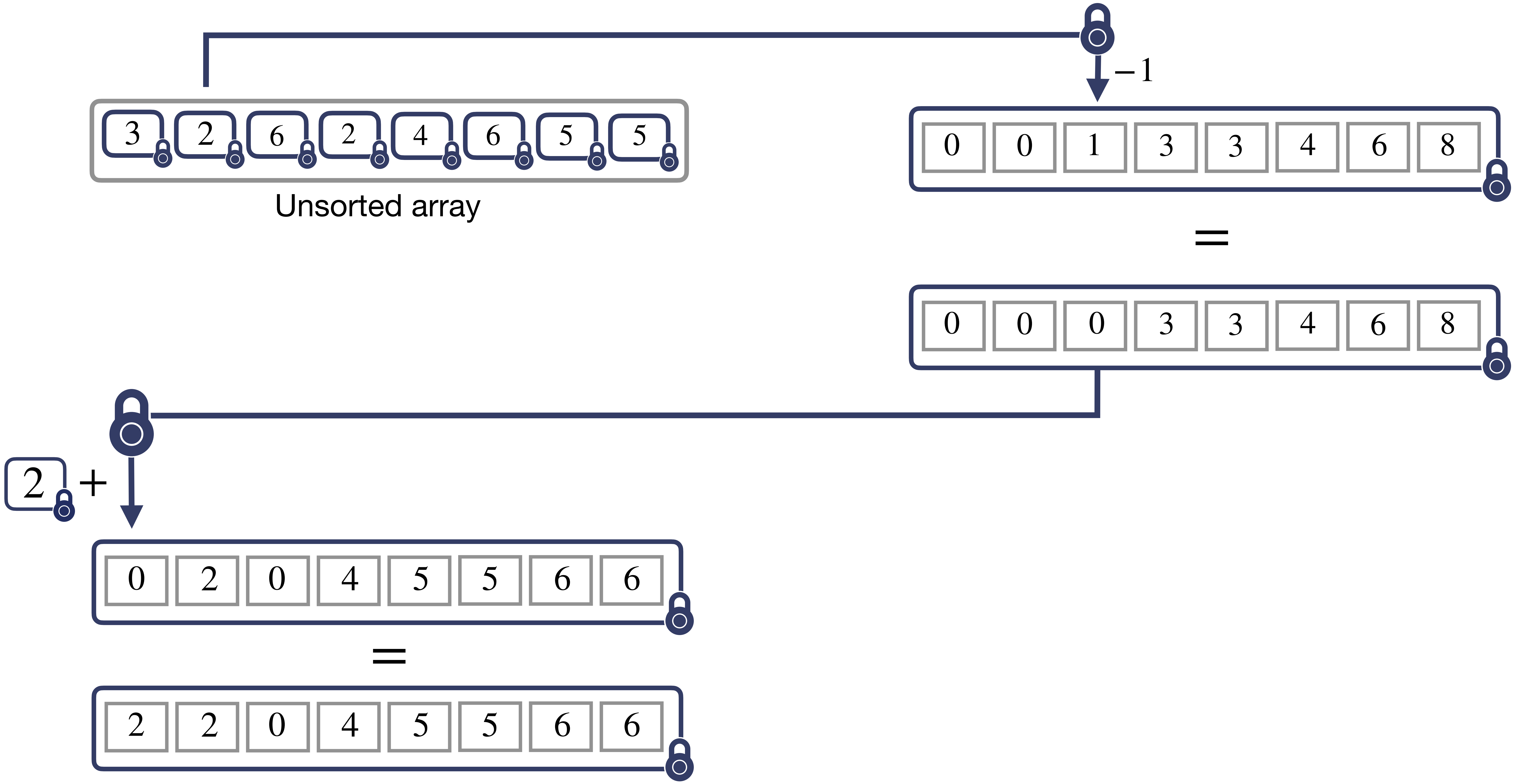
Blind Counting Sort (BCS)

Build the sorted array



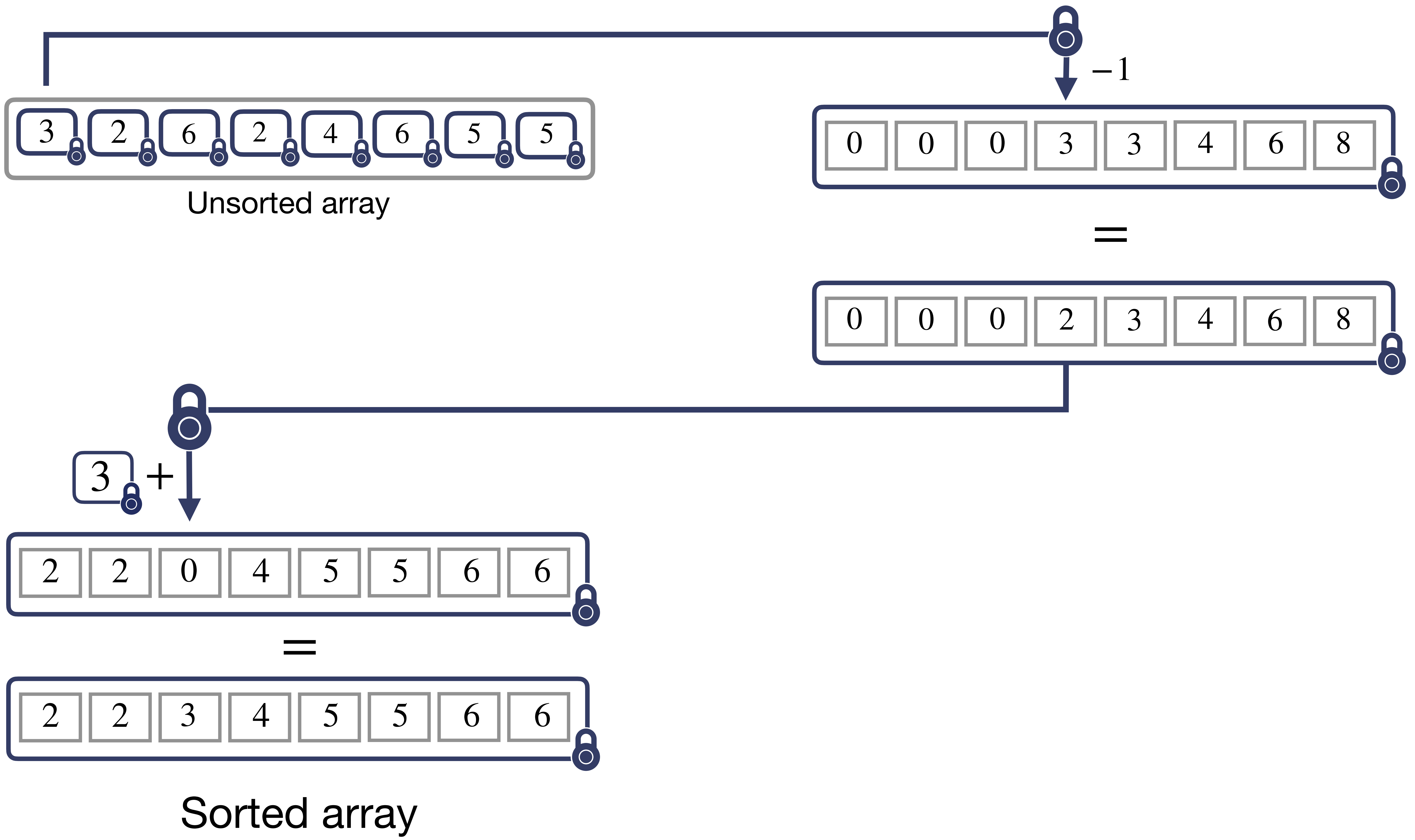
Blind Counting Sort (BCS)

Build the sorted array



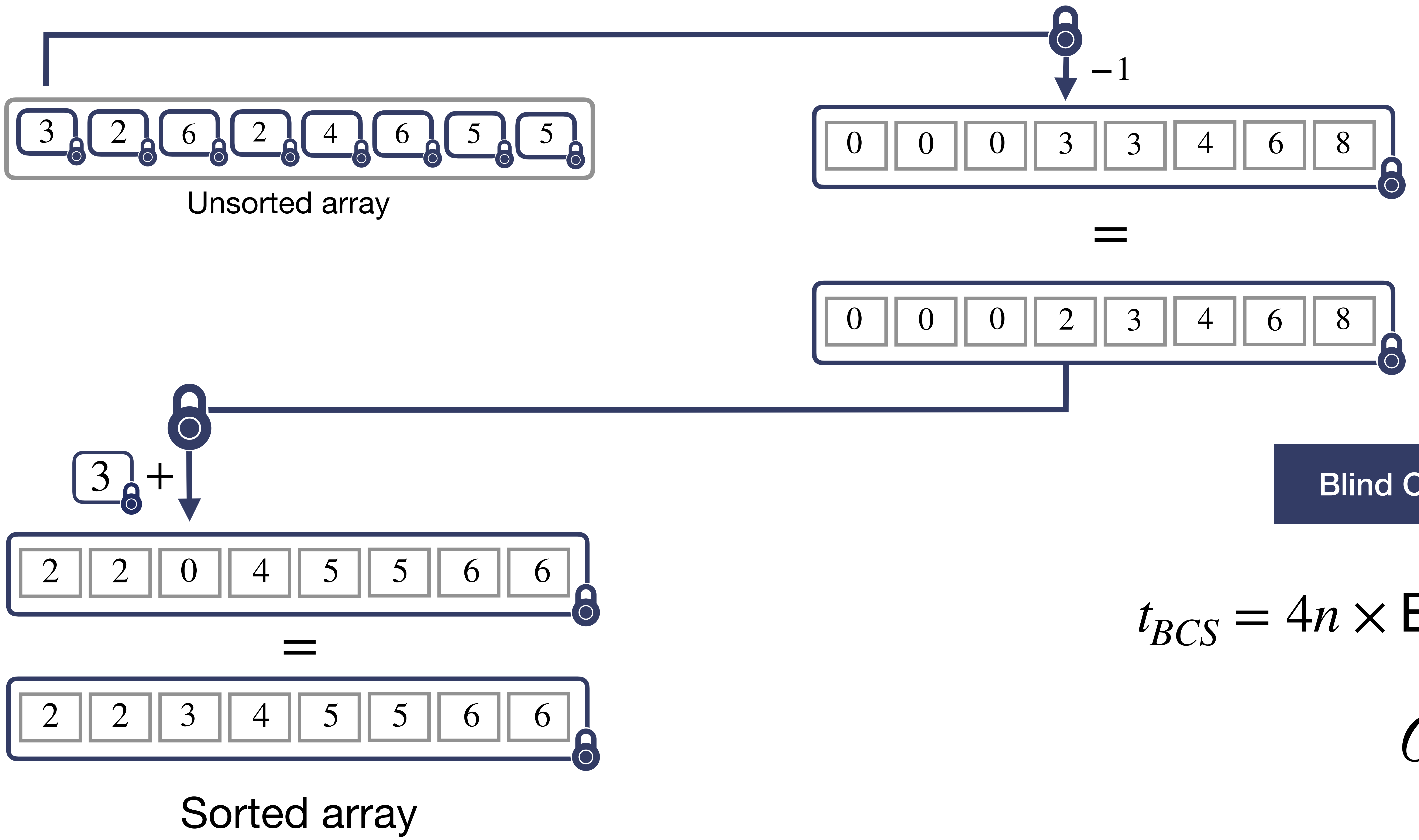
Blind Counting Sort (BCS)

Build the sorted array



Blind Counting Sort (BCS)

Build the sorted array



Blind Counting Sort

$$t_{BCS} = 4n \times \mathrm{BR} + 2n \times \mathrm{PFKS}$$

$$\mathcal{O}(n)$$

Blind Counting Sort (BCS)

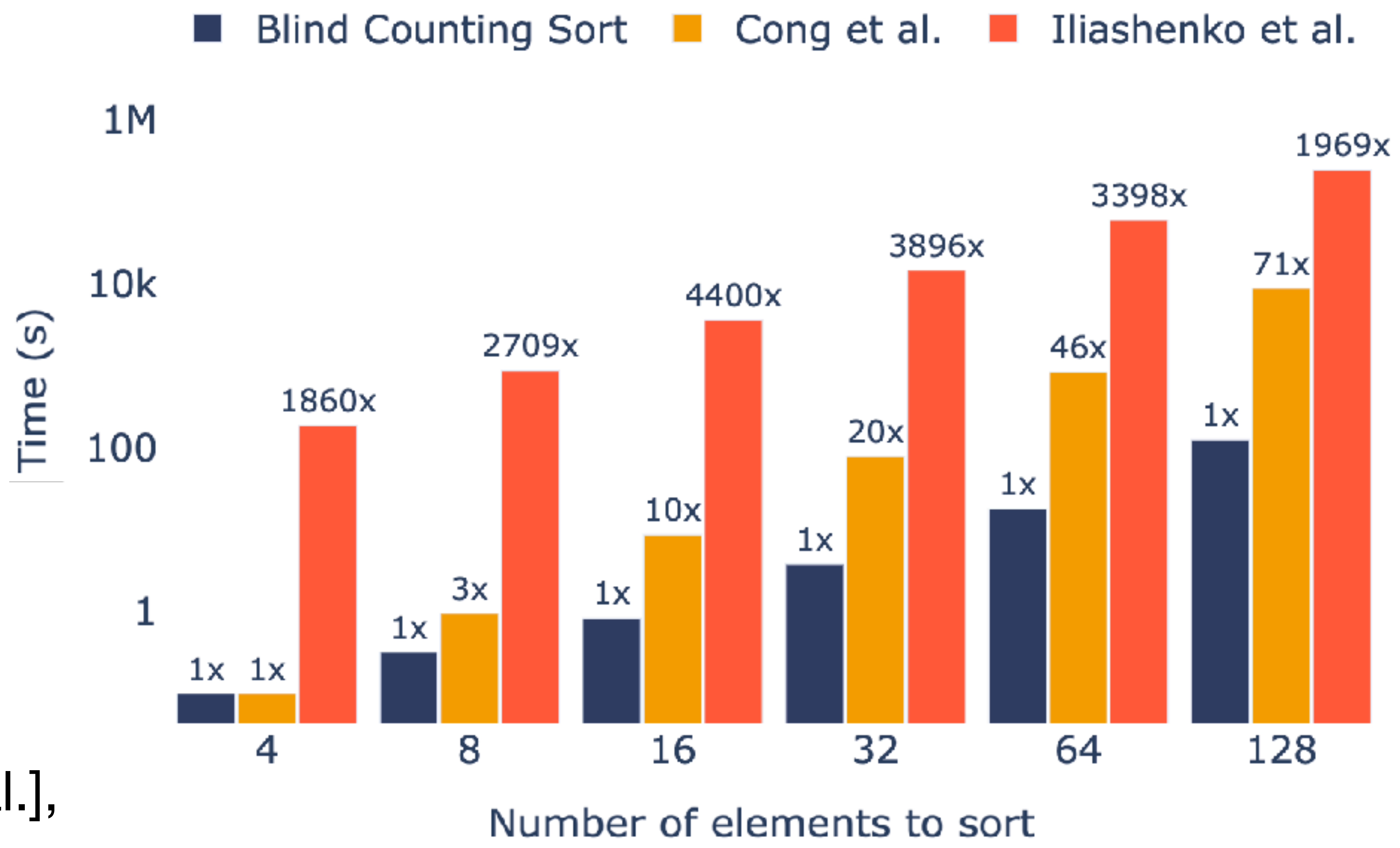
Comparison with the SOA

Iliashenko-Zucca. :

- Direct Sort ($\mathcal{O}(n^2)$)
- BGV scheme (SIMD compatible)
- Faster comparison

Cong et al. :

- Batcher Sort ($\mathcal{O}(nlog(n))$)
- TFHE scheme
- Build on the comparison method by [Zuber et al.], which needs **an extra bit of precision**.



[Iliashenko-Zucca.] : Faster homomorphic comparison operations for BGV and BFV, PoPETs, 2021(3), 246–264.

[Cong et al.] : Revisiting Oblivious Top-k Selection with Applications to Secure k-NN Classification, SAC 2024, LNCS 15516.

[Zuber et al.] : Efficient and accurate homomorphic comparisons, WHAC 2022, pp. 35–46.

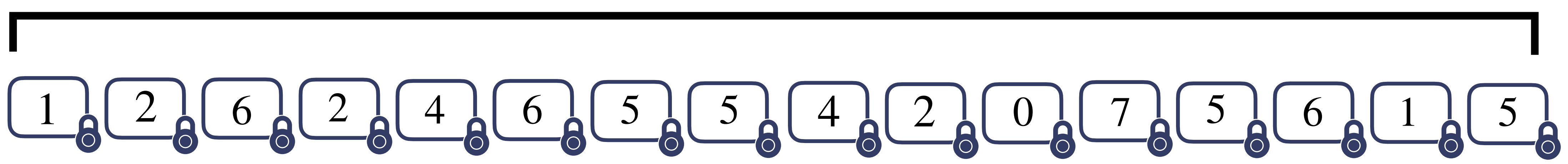
Blind Top-k

Blind Top-k

BCS used in a tournament

Blind Top-k

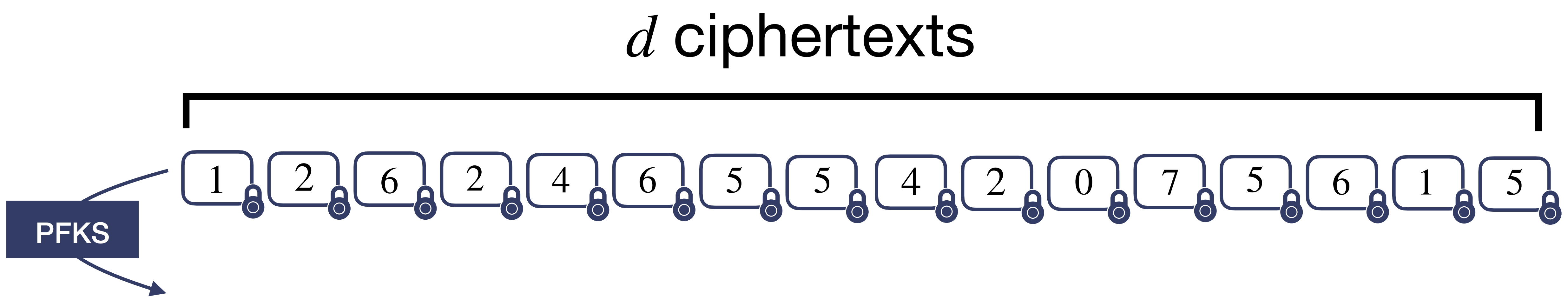
BCS used in a tournament



Blind Top-k

BCS used in a tournament

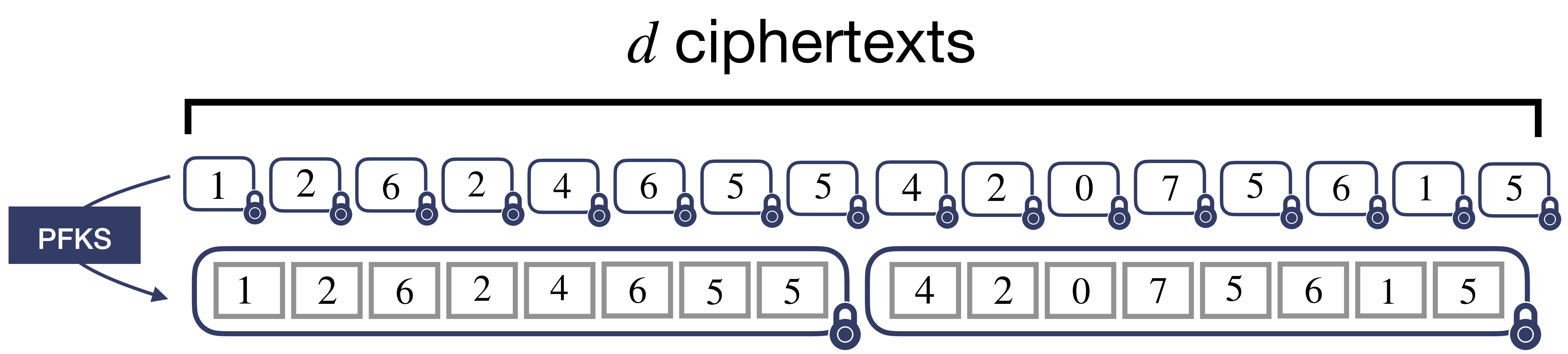
$k = 3$



Blind Top-k

BCS used in a tournament

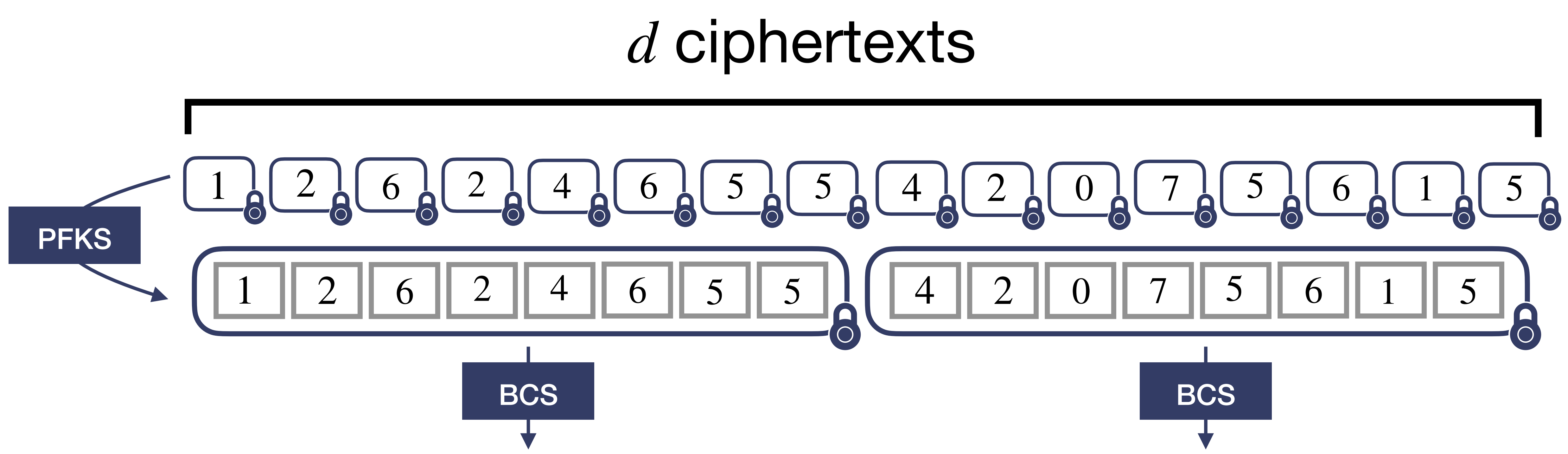
$k = 3$



Blind Top-k

BCS used in a tournament

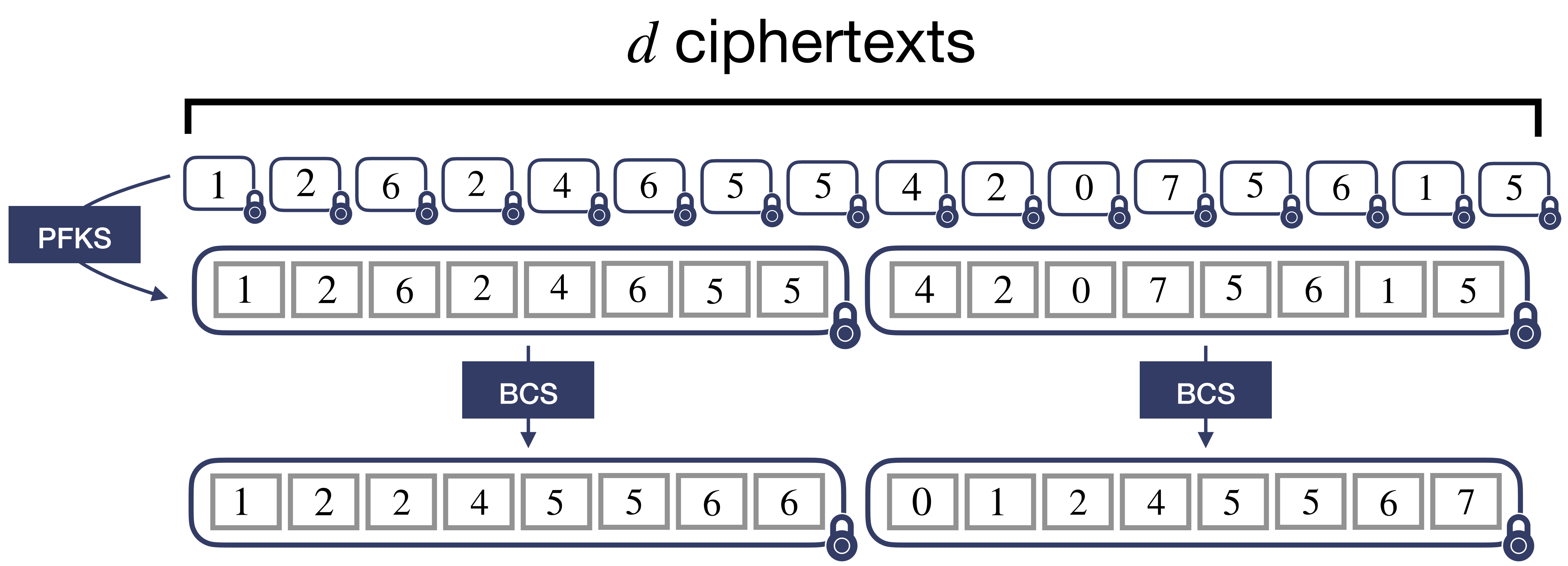
$k = 3$



Blind Top-k

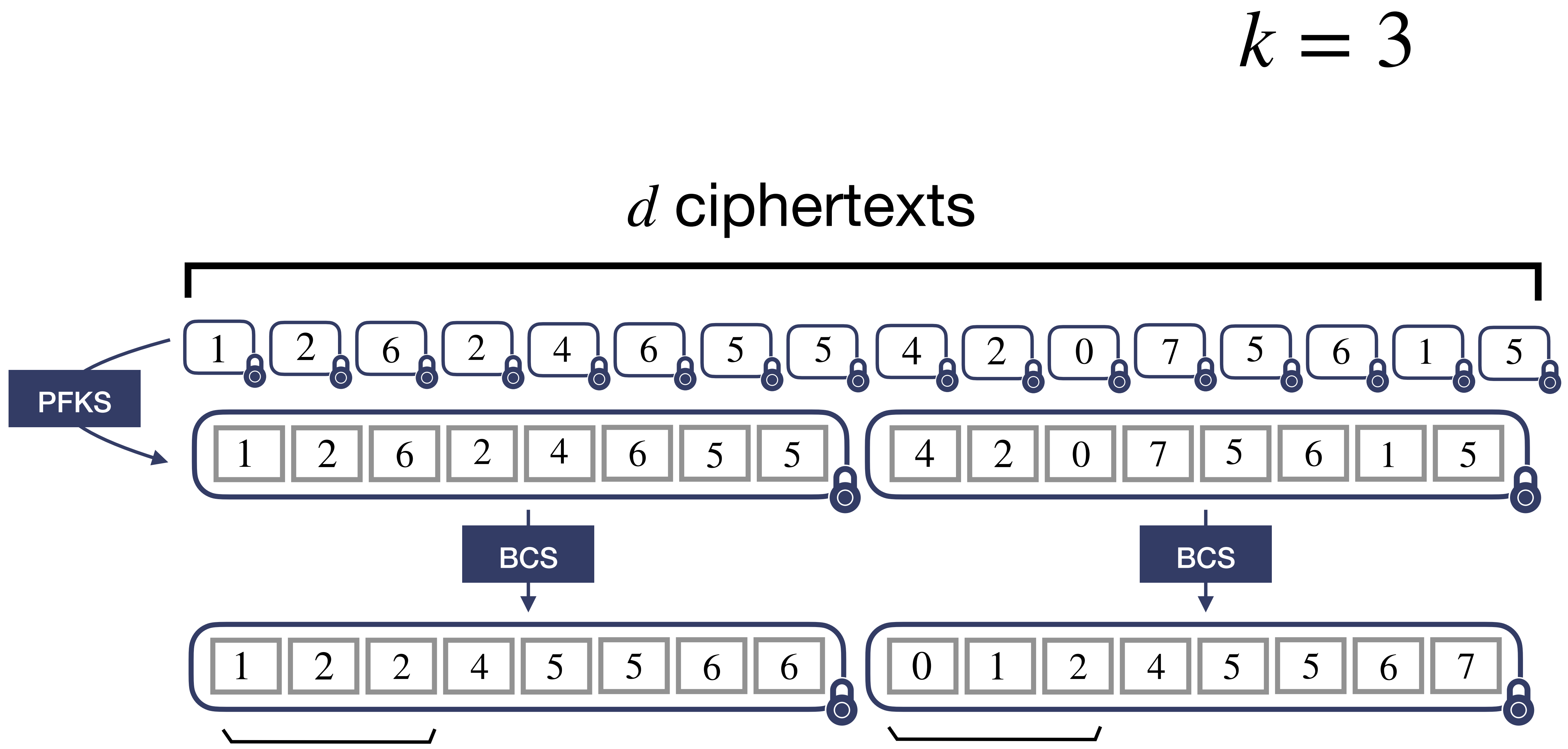
BCS used in a tournament

$k = 3$



Blind Top-k

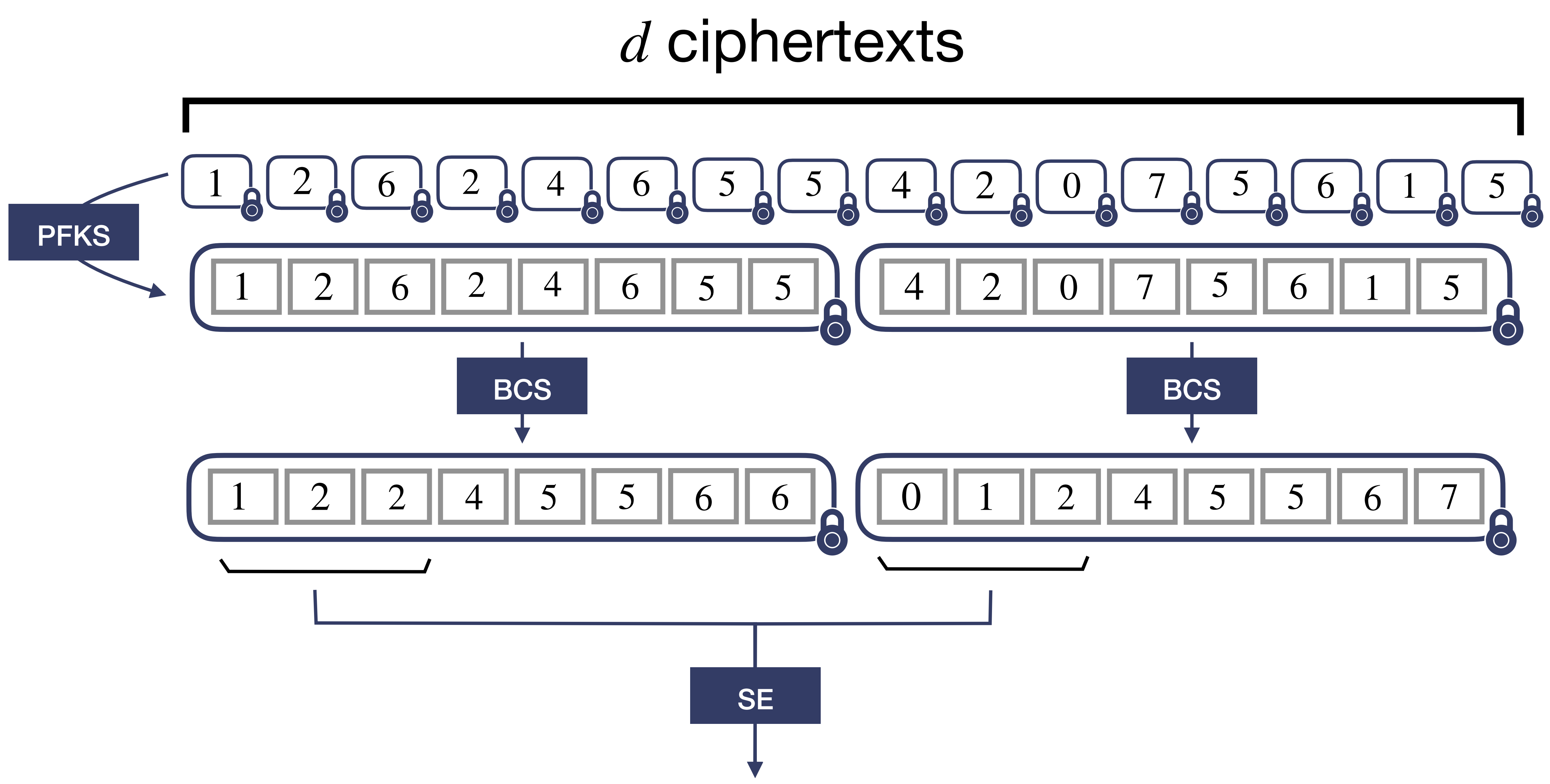
BCS used in a tournament



Blind Top-k

BCS used in a tournament

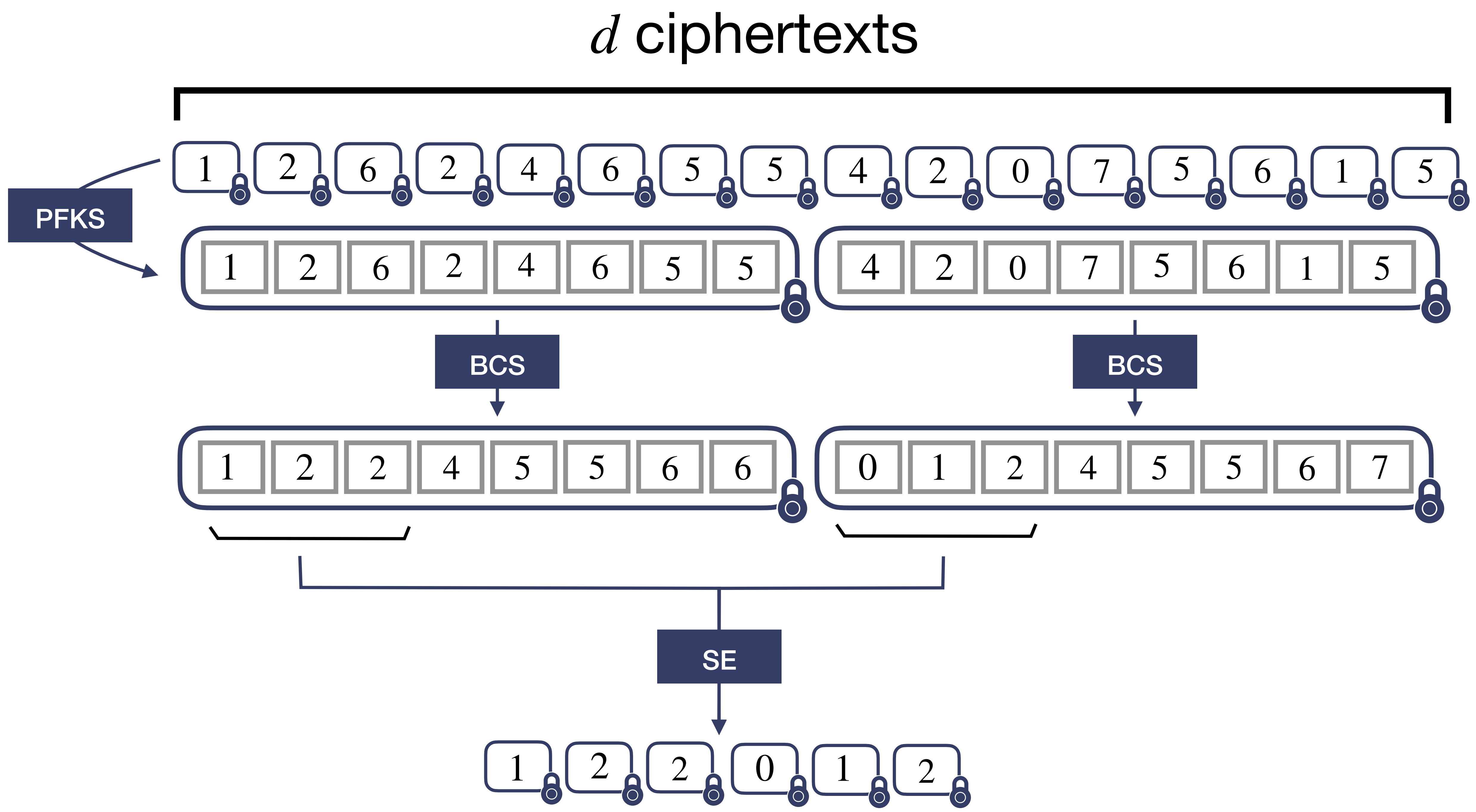
$k = 3$



Blind Top-k

BCS used in a tournament

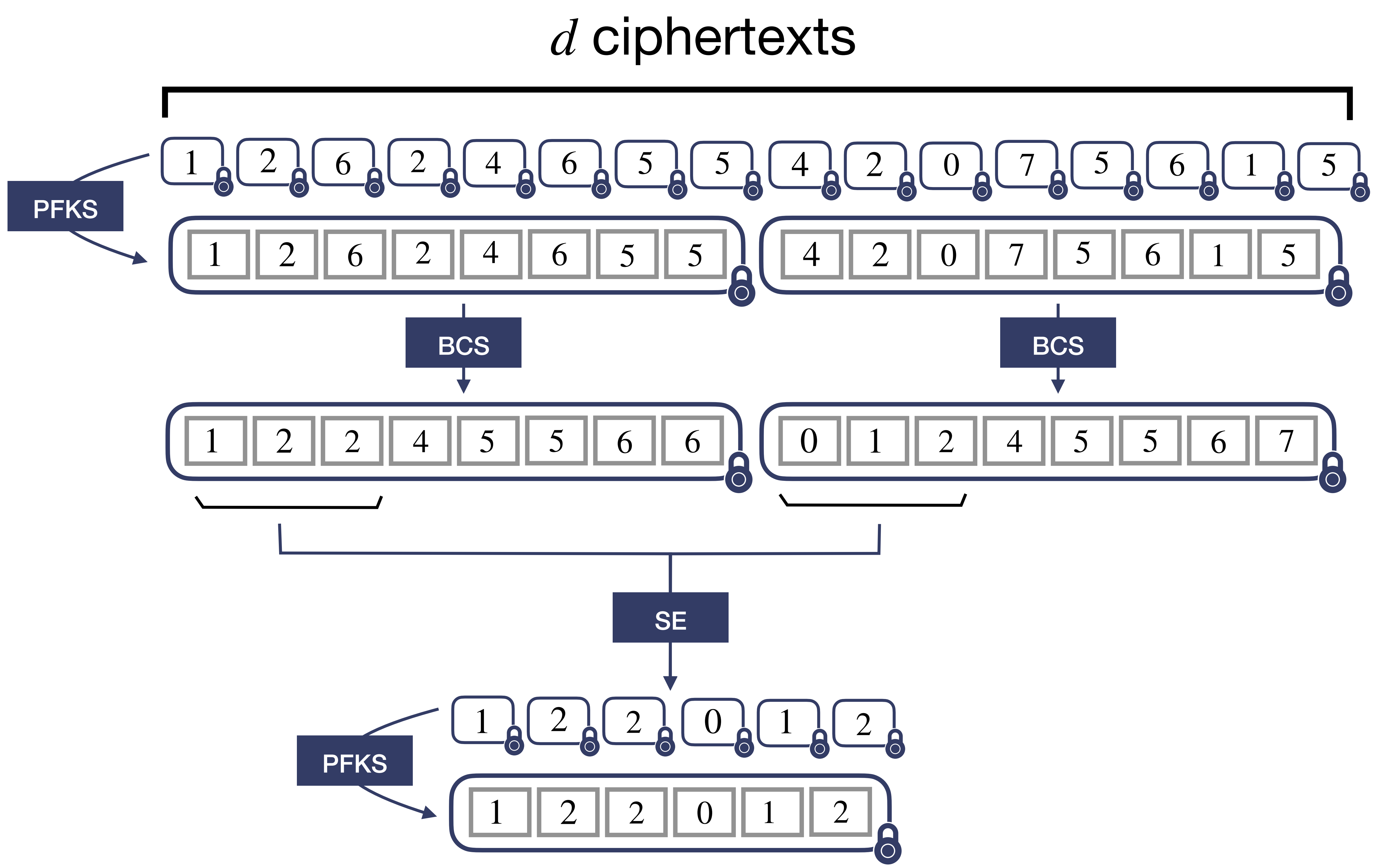
$k = 3$



Blind Top-k

BCS used in a tournament

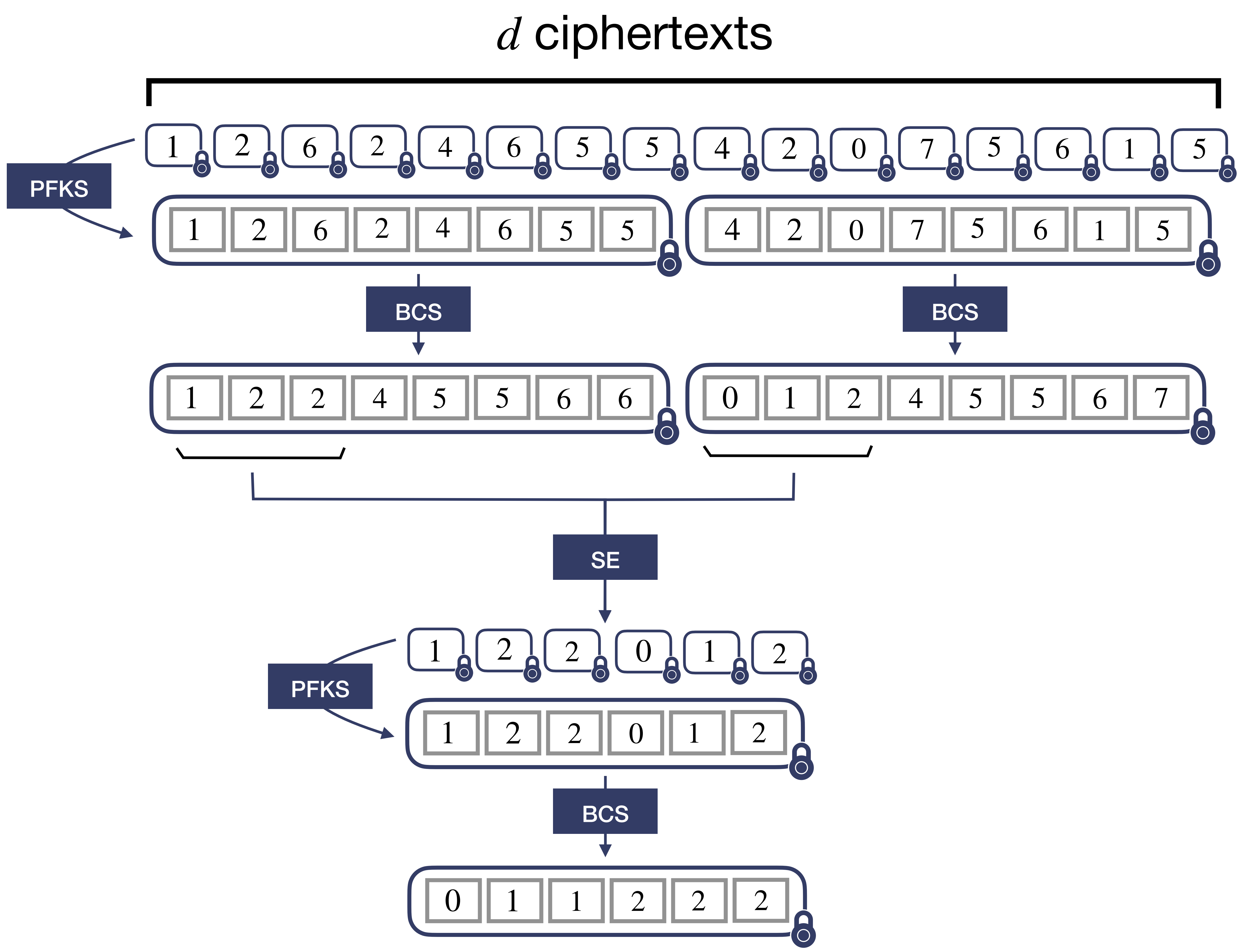
$k = 3$



Blind Top-k

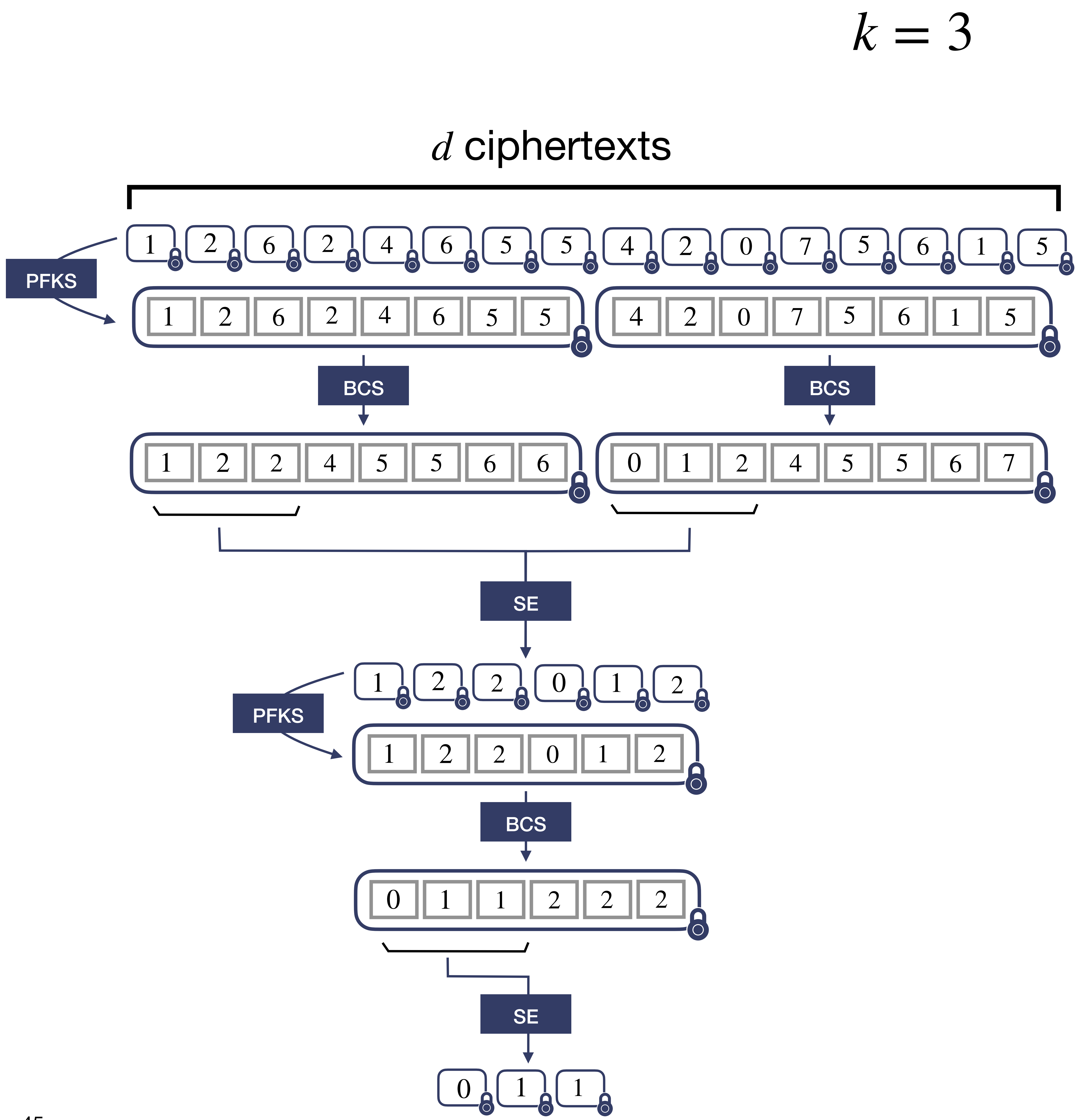
BCS used in a tournament

$k = 3$



Blind Top-k

BCS used in a tournament



Blind Top-k

BCS used in a tournament

Blind Top-k

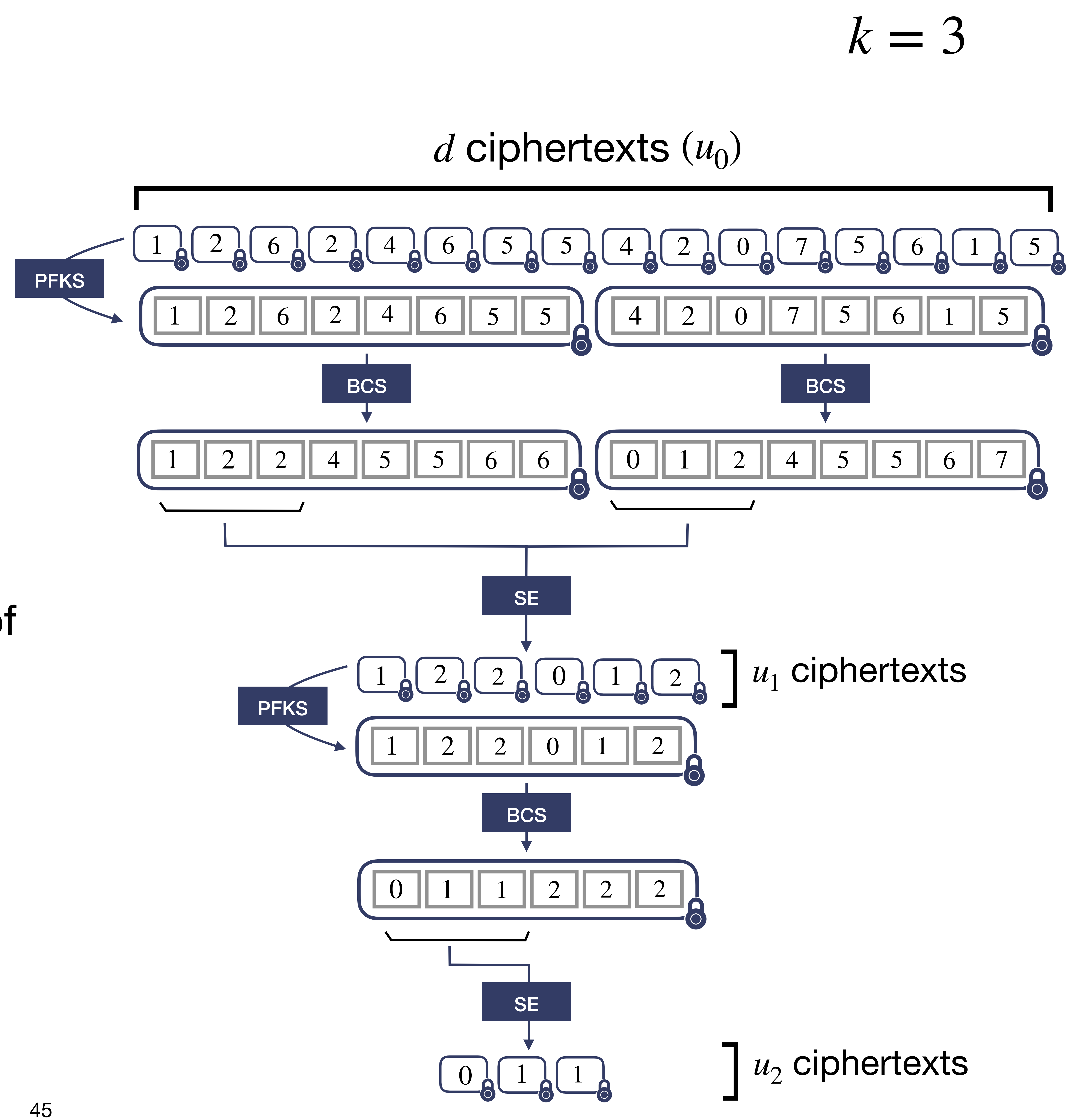
Total number of calls to BCS :

$$U = \left(\sum_{i=0}^{r-1} \left\lfloor \frac{u_i}{p} \right\rfloor \right) + 1$$

where r is the number of rounds and u_i the number of ciphertexts remaining after the i-th round.

Therefore, focusing on BR and PFKS, we have :

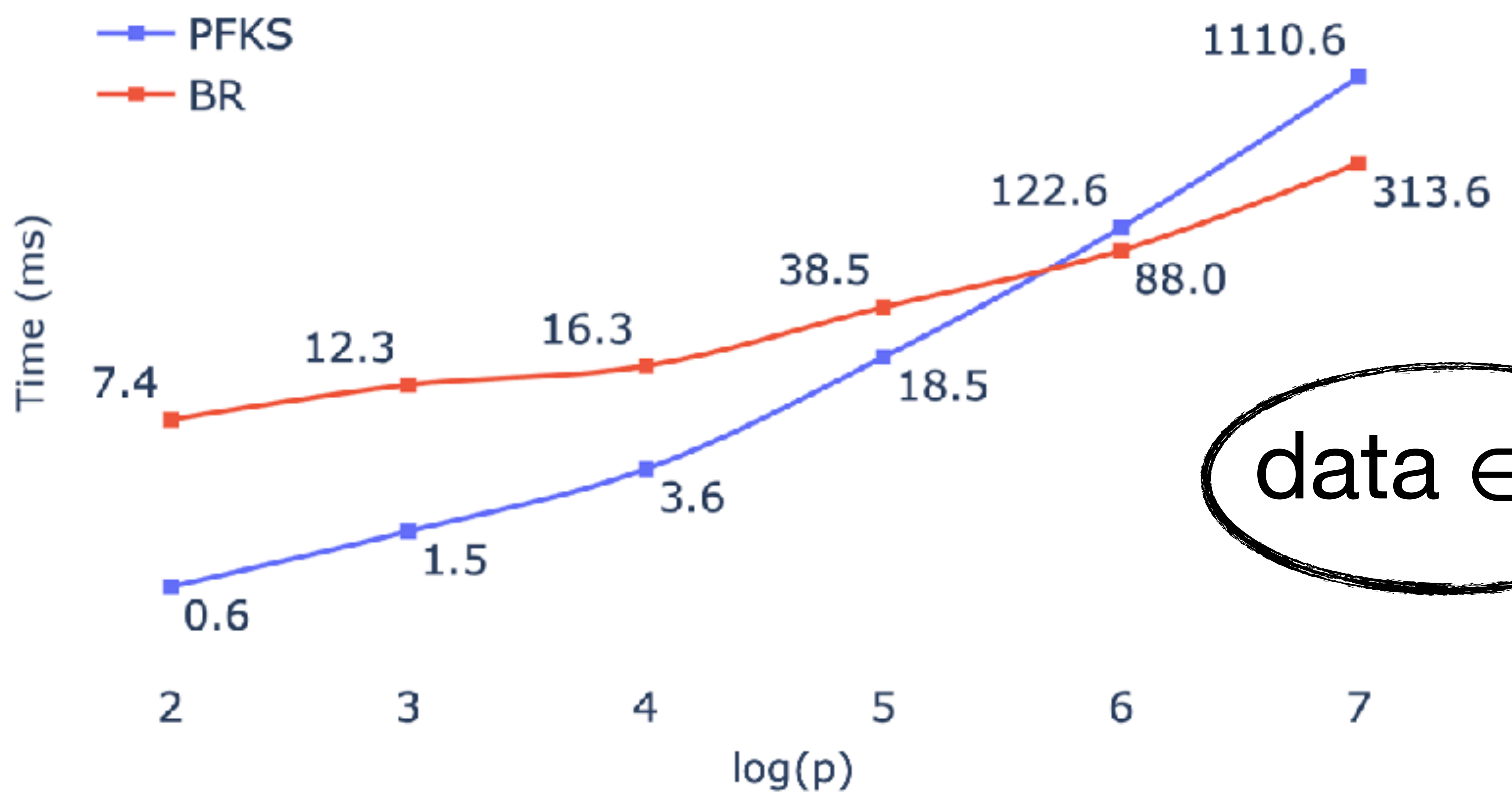
$$t_{BlindTopk} = 4pU \times \text{BR} + (2pU + r) \times \text{PFKS}$$



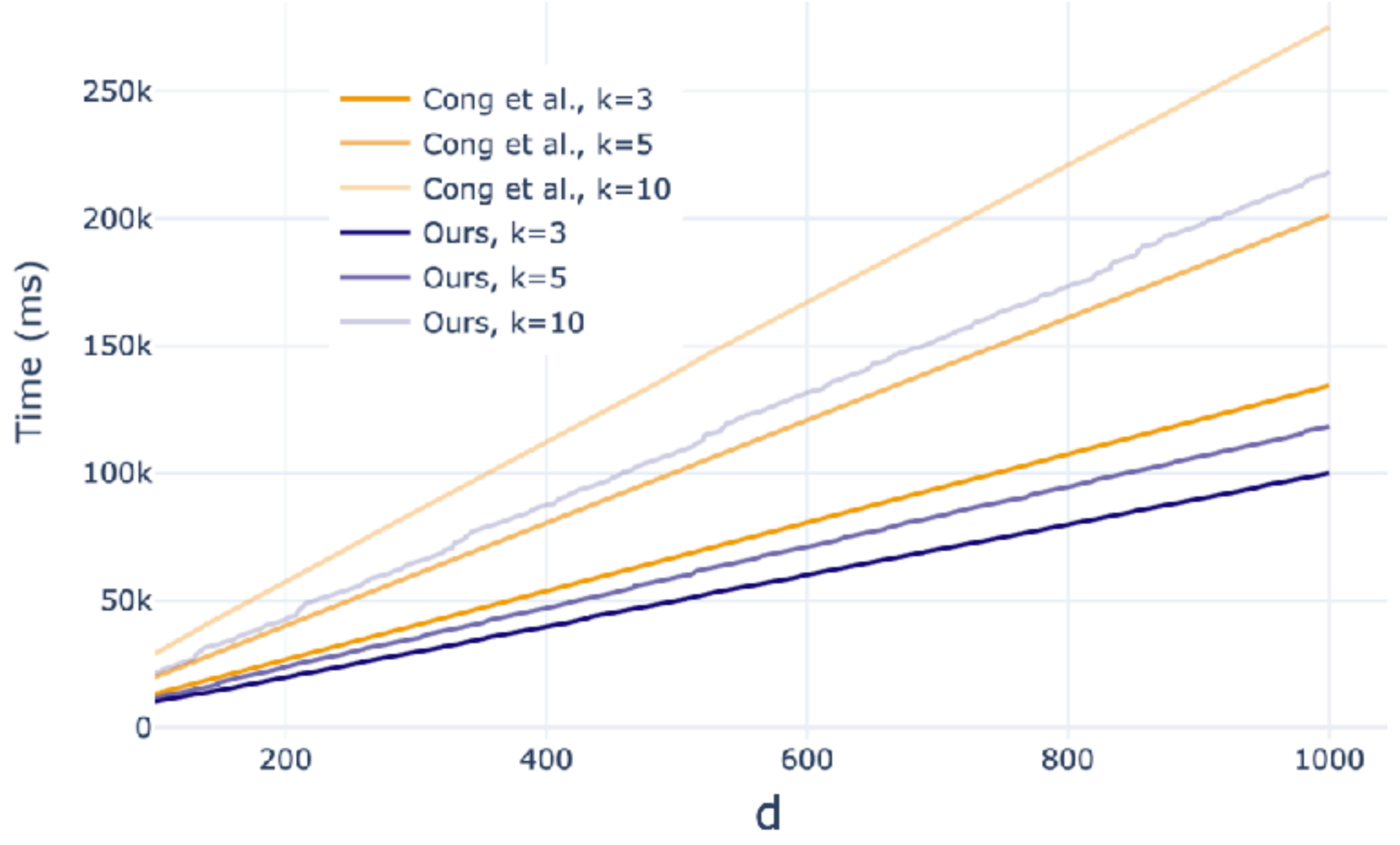
Blind Top-k

Comparison with the SOA

k	d	Cong et al.		Ours	
		BR	PFKS	BR	PFKS
3	40	186	372	190	148
	175	862	1724	995	668
	269	1332	2664	1565	1022
	457	2272	4544	2775	1794
	1000	4986	9972	6060	3846
5	40	250	500	210	156
	175	1196	2392	1215	810
	269	1856	3712	1860	1212
	457	3172	6344	3200	2054
	1000	6970	13940	7225	4582



data $\in \mathbb{Z}_{2^4}$

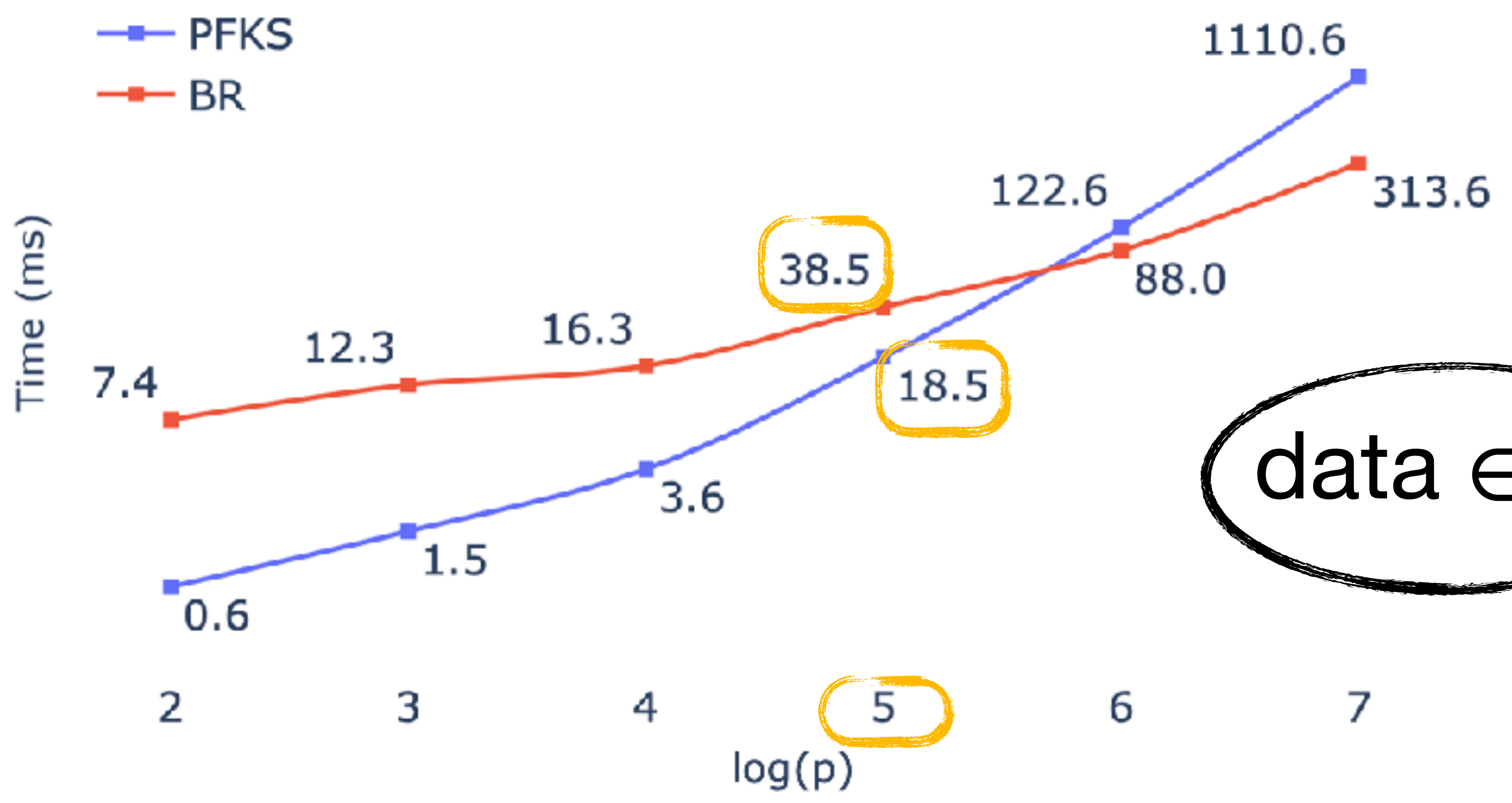


[Cong et al.] : Revisiting Oblivious Top-k Selection with Applications to Secure k-NN Classification, SAC 2024, LNCS 15516.

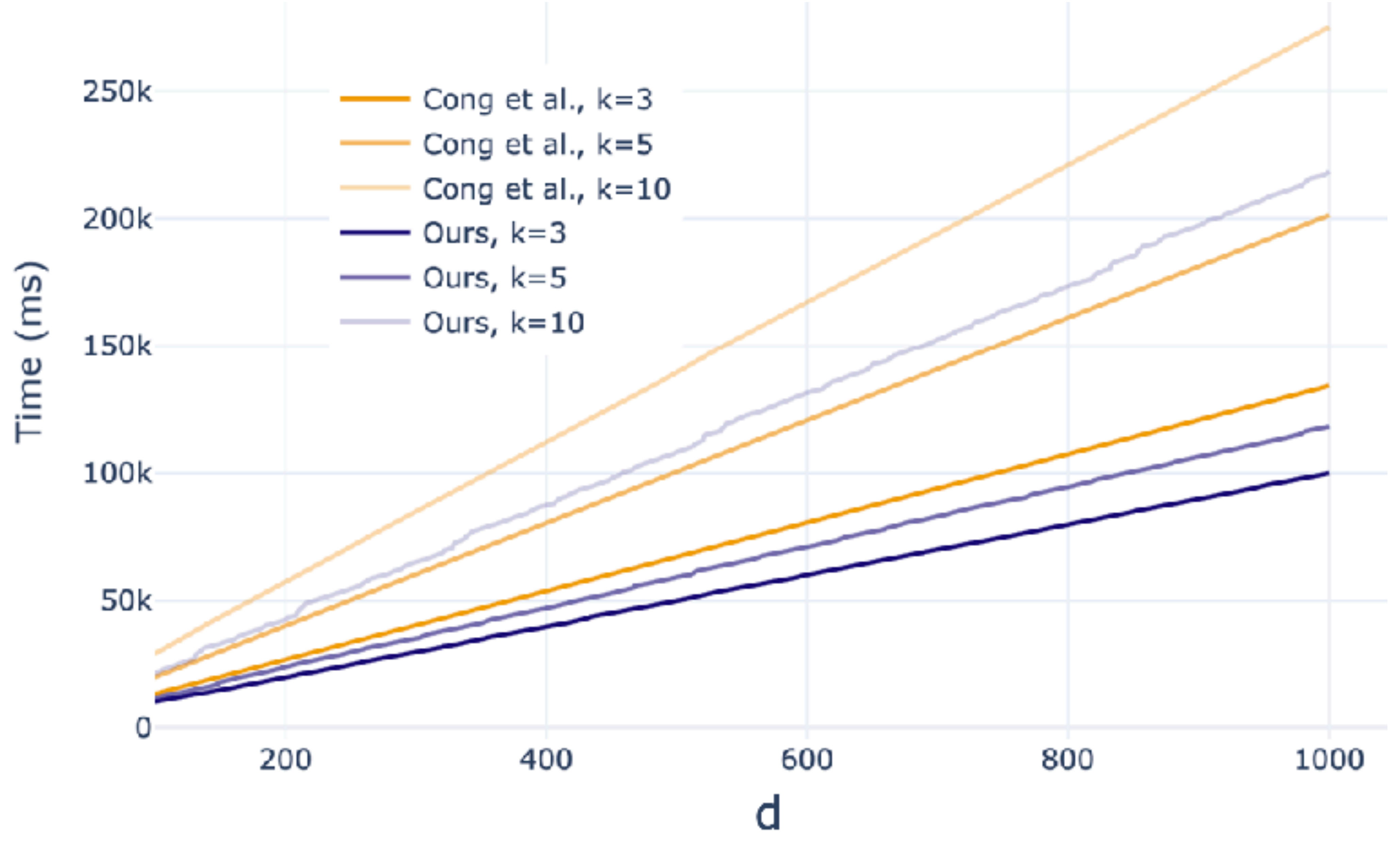
Blind Top-k

Comparison with the SOA

k	d	Cong et al.		Ours	
		BR	PFKS	BR	PFKS
3	40	186	372	190	148
	175	862	1724	995	668
	269	1332	2664	1565	1022
	457	2272	4544	2775	1794
	1000	4986	9972	6060	3846
5	40	250	500	210	156
	175	1196	2392	1215	810
	269	1856	3712	1860	1212
	457	3172	6344	3200	2054
	1000	6970	13940	7225	4582



data $\in \mathbb{Z}_{24}$

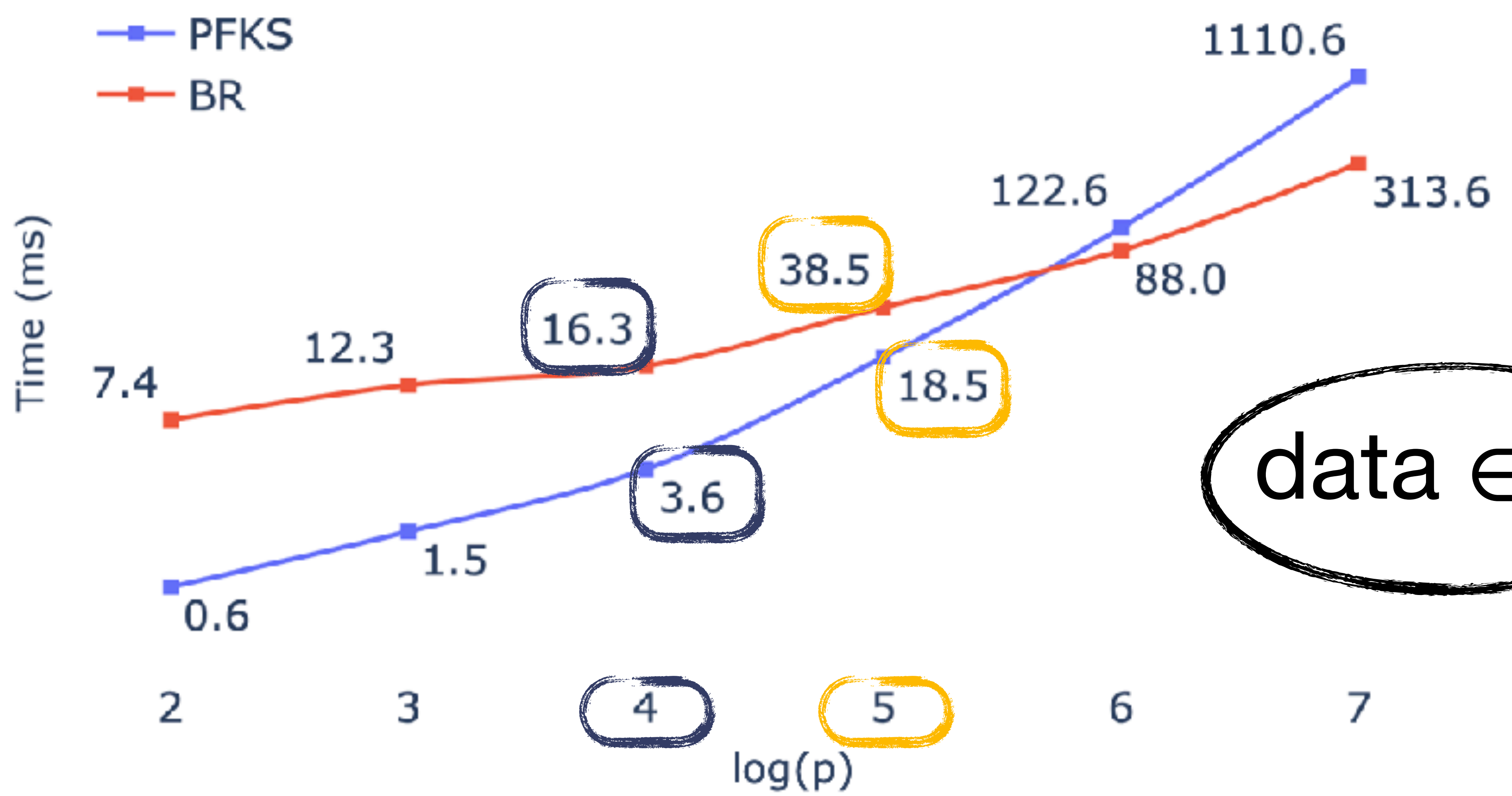


[Cong et al.] : Revisiting Oblivious Top-k Selection with Applications to Secure k-NN Classification, SAC 2024, LNCS 15516.

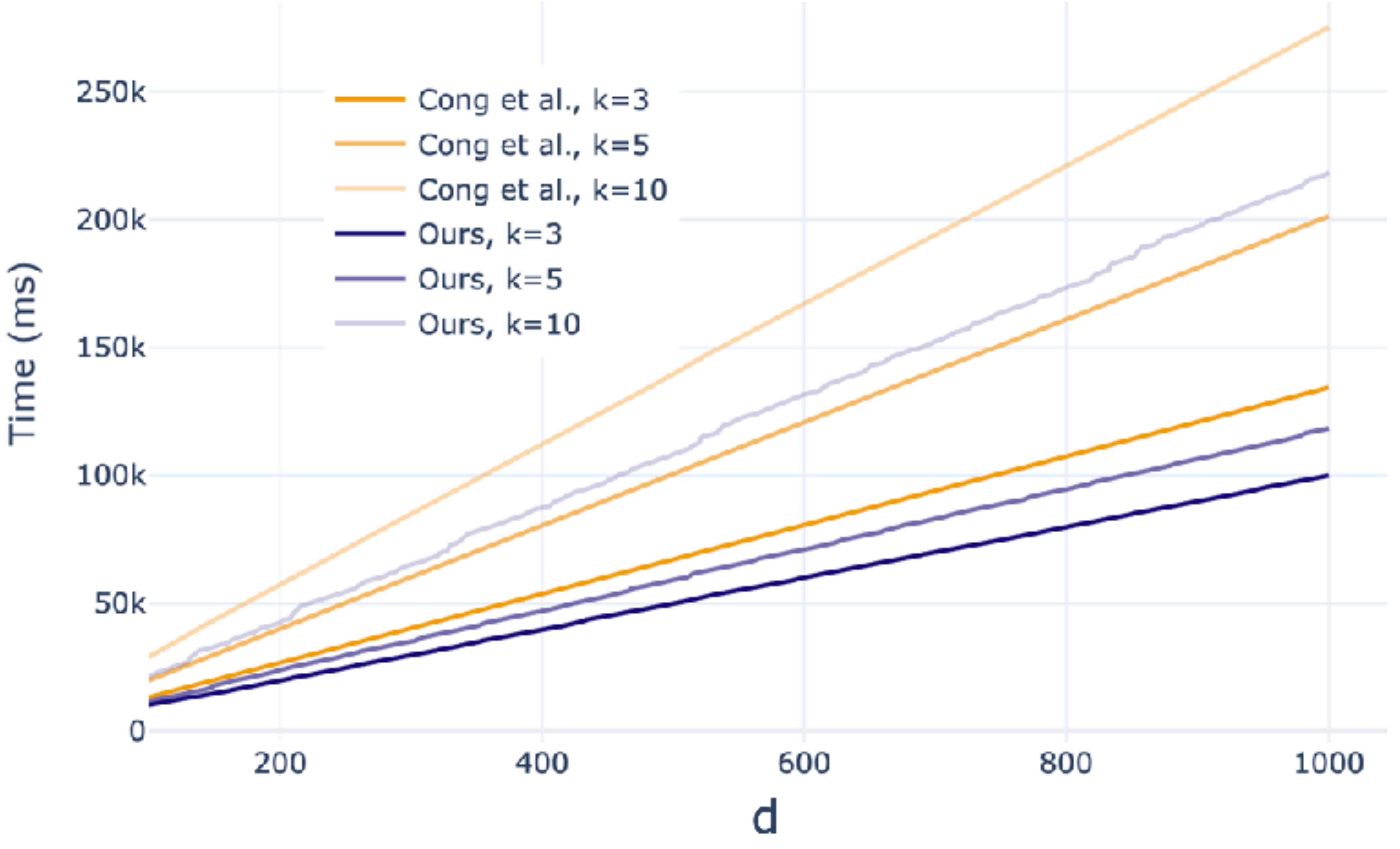
Blind Top-k

Comparison with the SOA

k	d	Cong et al.		Ours	
		BR	PFKS	BR	PFKS
3	40	186	372	190	148
	175	862	1724	995	668
	269	1332	2664	1565	1022
	457	2272	4544	2775	1794
	1000	4986	9972	6060	3846
5	40	250	500	210	156
	175	1196	2392	1215	810
	269	1856	3712	1860	1212
	457	3172	6344	3200	2054
	1000	6970	13940	7225	4582



data $\in \mathbb{Z}_{24}$



[Cong et al.] : Revisiting Oblivious Top-k Selection with Applications to Secure k-NN Classification, SAC 2024, LNCS 15516.

Private k-Nearest Neighbours

Private k-Nearest Neighbours

How it works in clear ?

Private k-Nearest Neighbours

How it works in clear ?

Client

Cloud

Private k-Nearest Neighbours

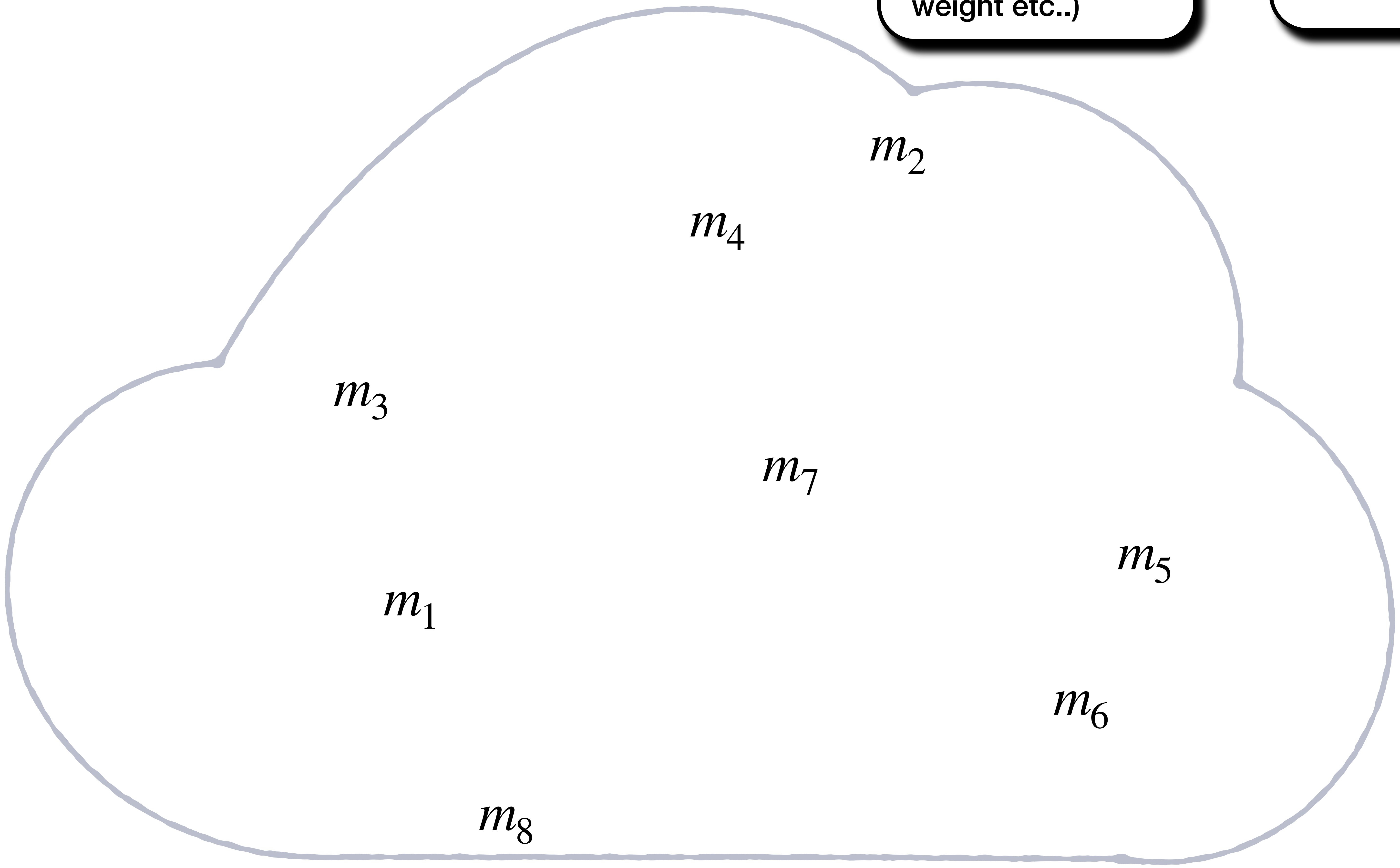
How it works in clear ?

$$m_i = (f_i, \ell_i)$$

Features (e.g age, weight etc..)

Label

Client



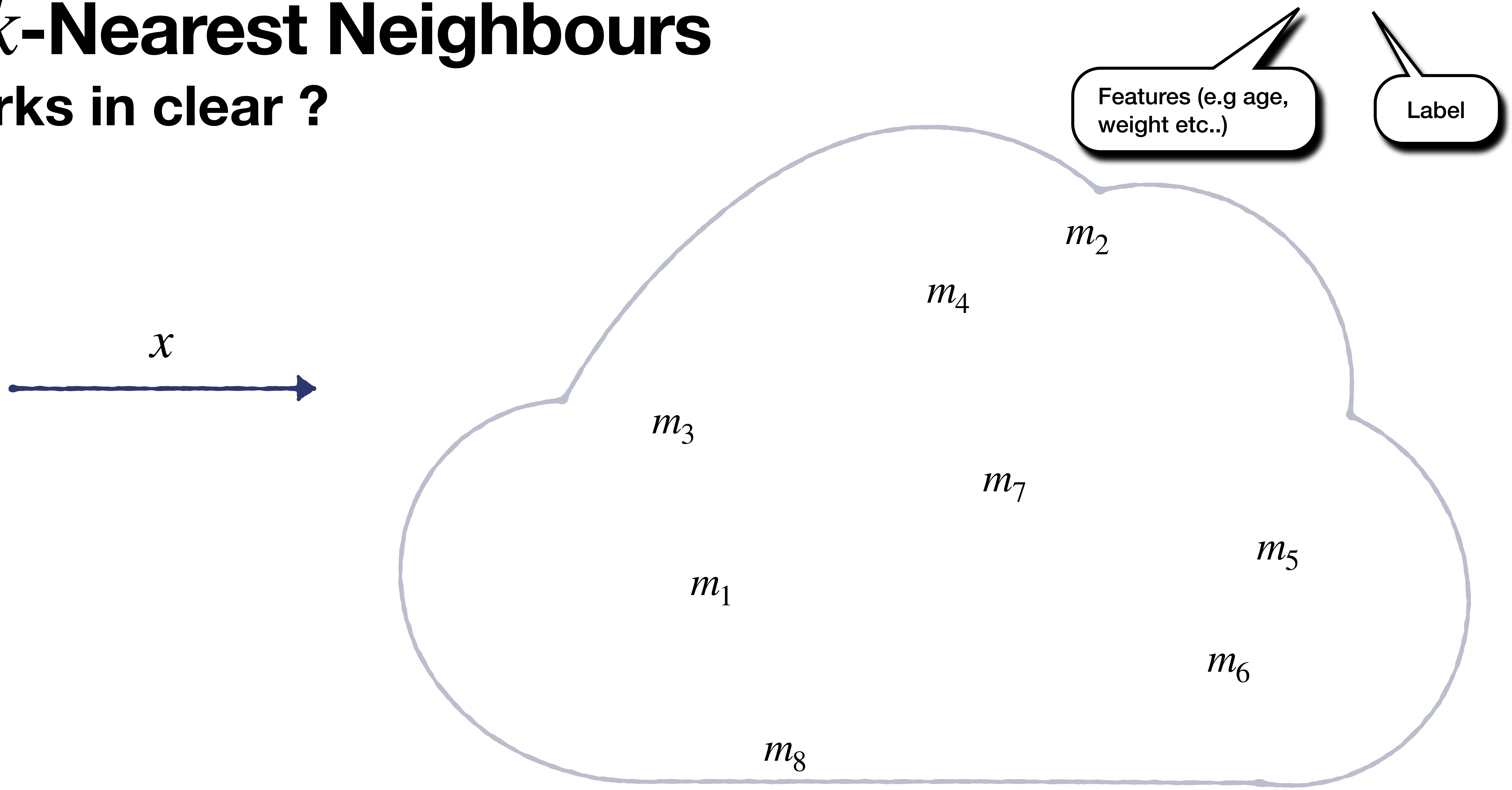
Cloud

Private k-Nearest Neighbours

How it works in clear ?

$$m_i = (f_i, \ell_i)$$

Client



Cloud

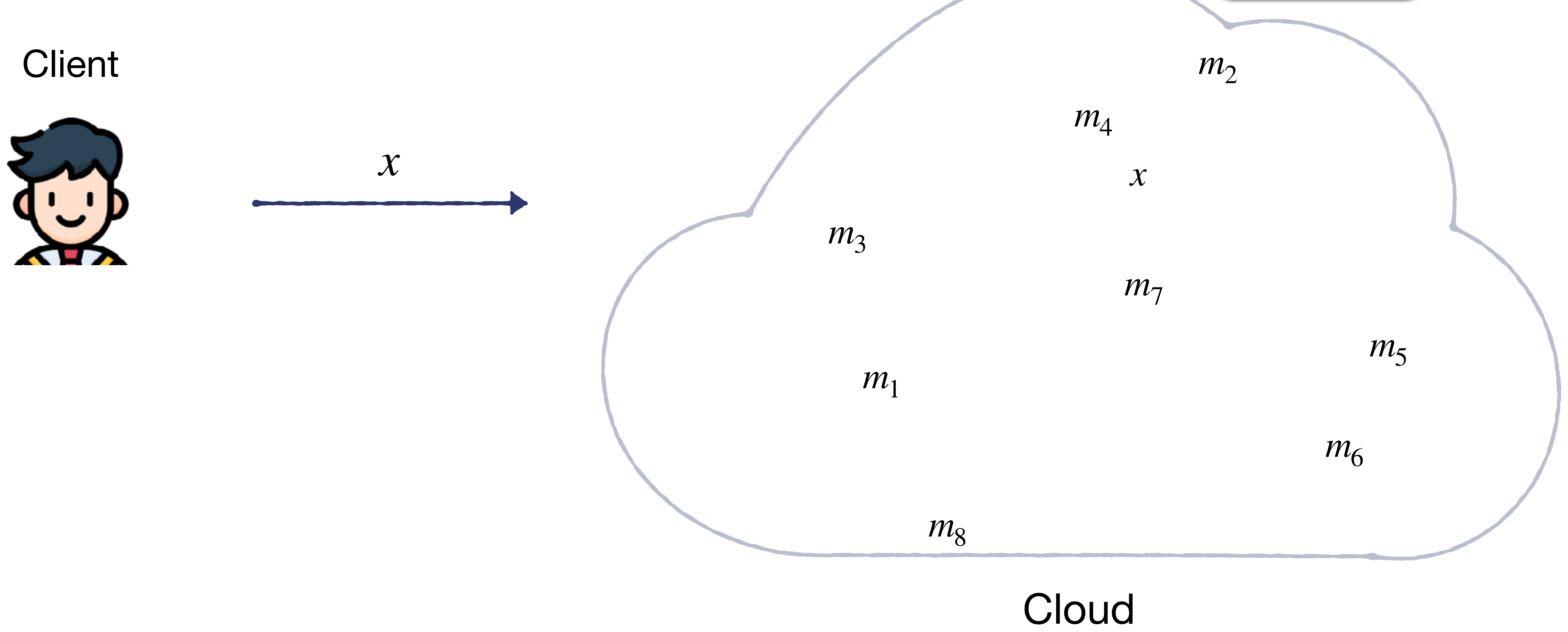
Private k-Nearest Neighbours

How it works in clear ?

$m_i = (f_i, \ell_i)$

Features (e.g age, weight etc..)

Label



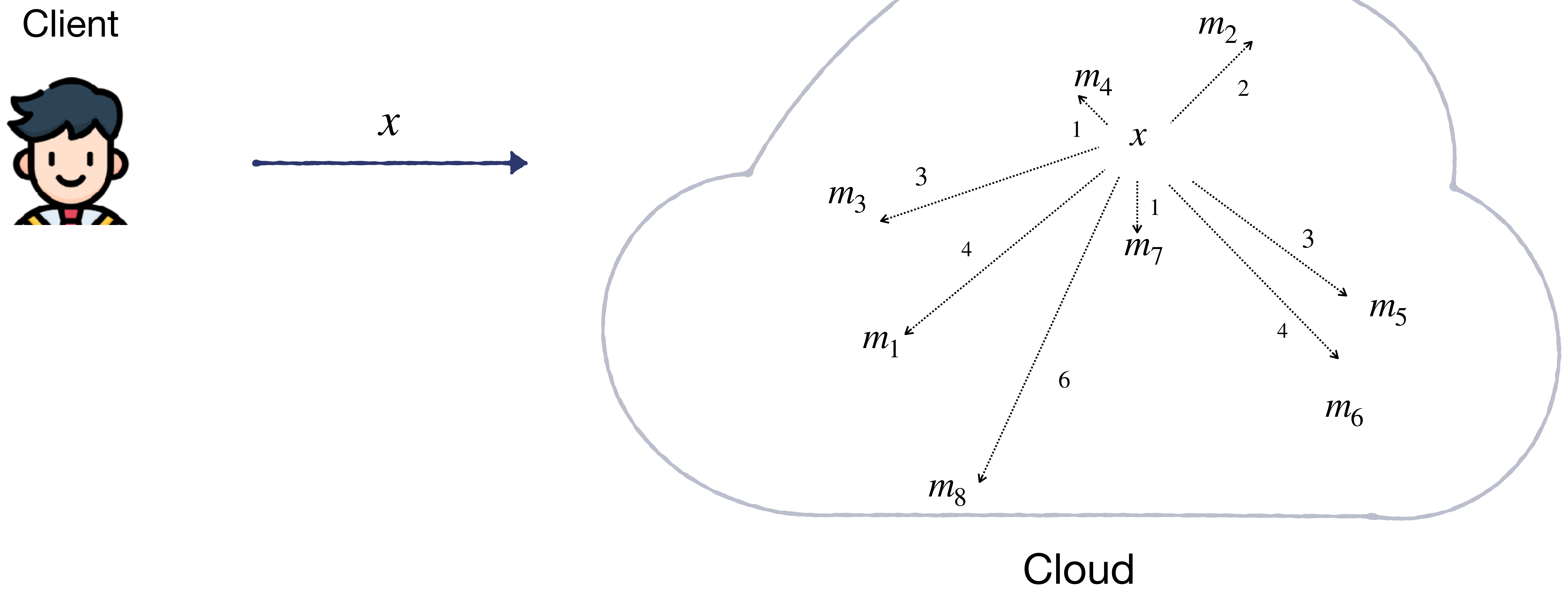
Private k-Nearest Neighbours

How it works in clear ?

$$m_i = (f_i, \ell_i)$$

Features (e.g age, weight etc..)

Label



Private k-Nearest Neighbours

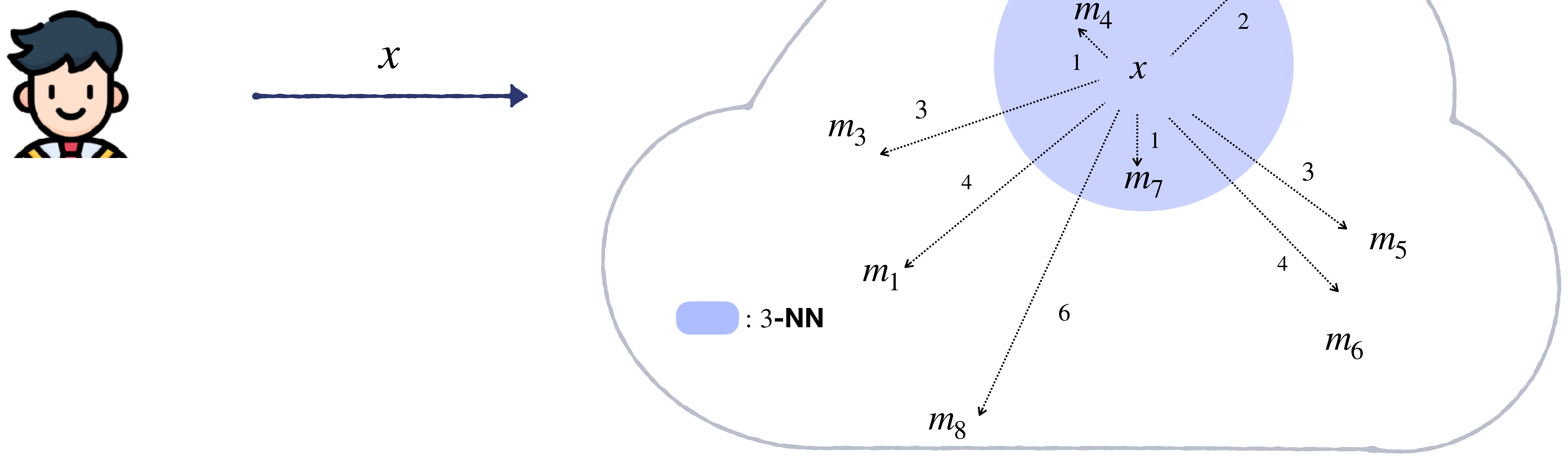
How it works in clear ?

$$m_i = (f_i, \ell_i)$$

Features (e.g age, weight etc..)

Label

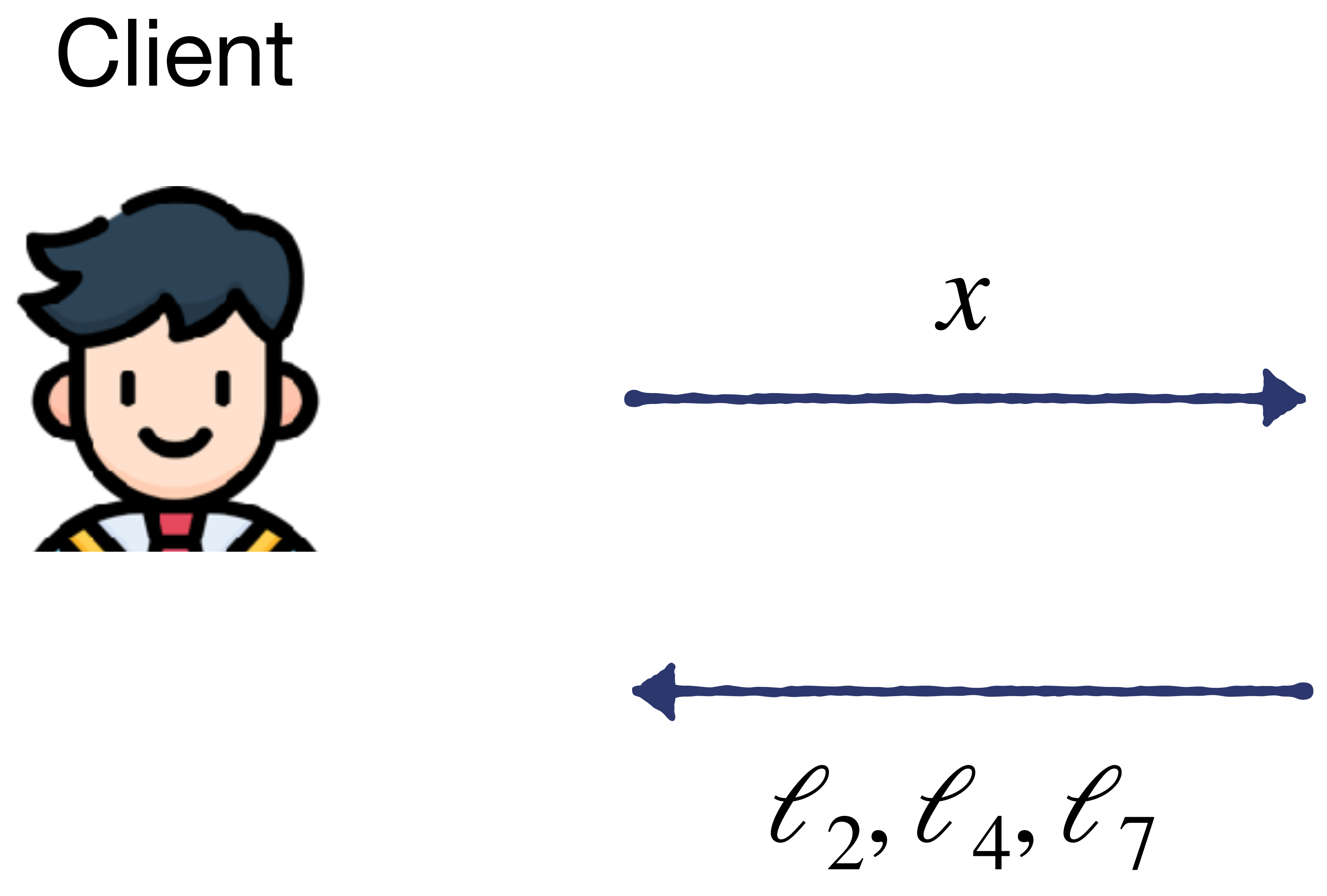
Client

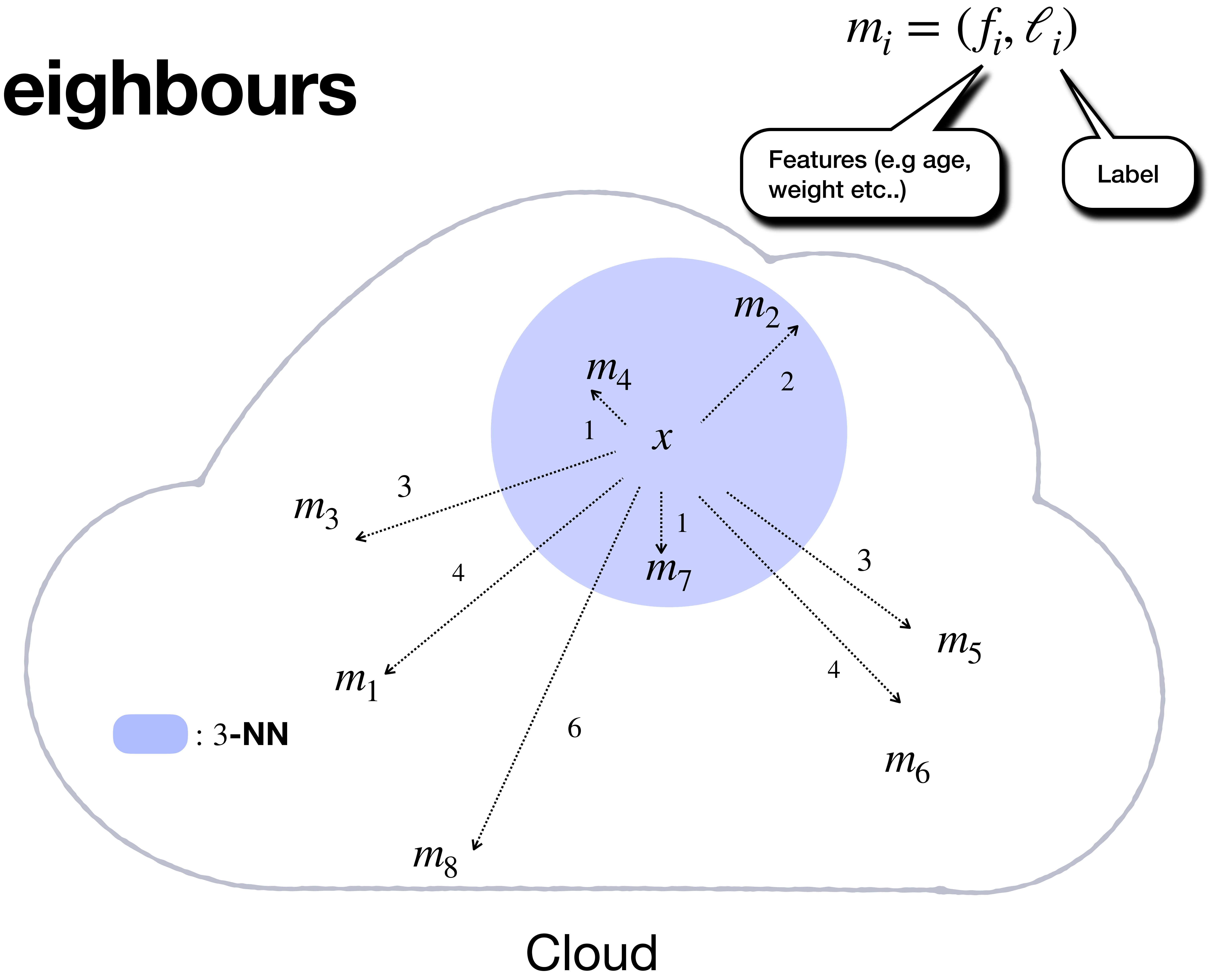


Cloud

Private k-Nearest Neighbours

How it works in clear ?





Private k-Nearest Neighbours

How it works in clear ?

$m_i = (f_i, \ell_i)$

Features (e.g age, weight etc..)

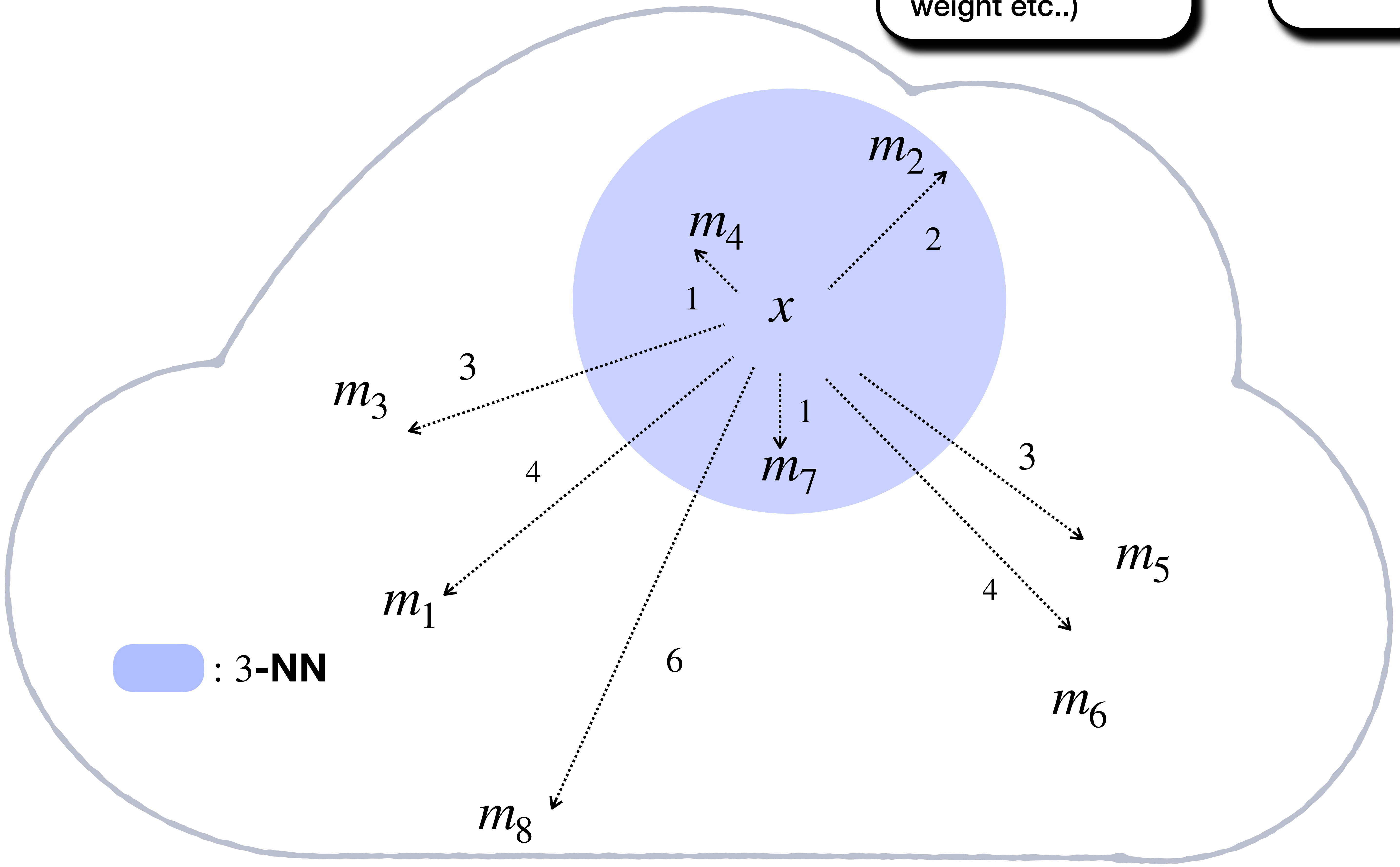
Label

Client

x

ℓ_2, ℓ_4, ℓ_7

$r \leftarrow$ Majority(ℓ_2, ℓ_4, ℓ_7)

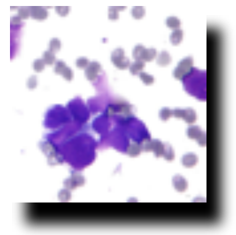


Cloud

Private k-Nearest Neighbours

Datasets used

$p = 2^4$



Dataset	Dimension	Message space	Dataset size	p_{dist}
Breast Cancer	30	$\mathbb{Z}_2$	569	2^4
MNIST	64	$\mathbb{Z}_2$	1797	2^5

A precision reduction is needed for the MNIST dataset as $p \leq p_{dist}$

Private k-Nearest Neighbours

Distances computation in $\mathbb{Z}_{p_{dist}}$

m_2

m_4

m_3

m_7

m_5

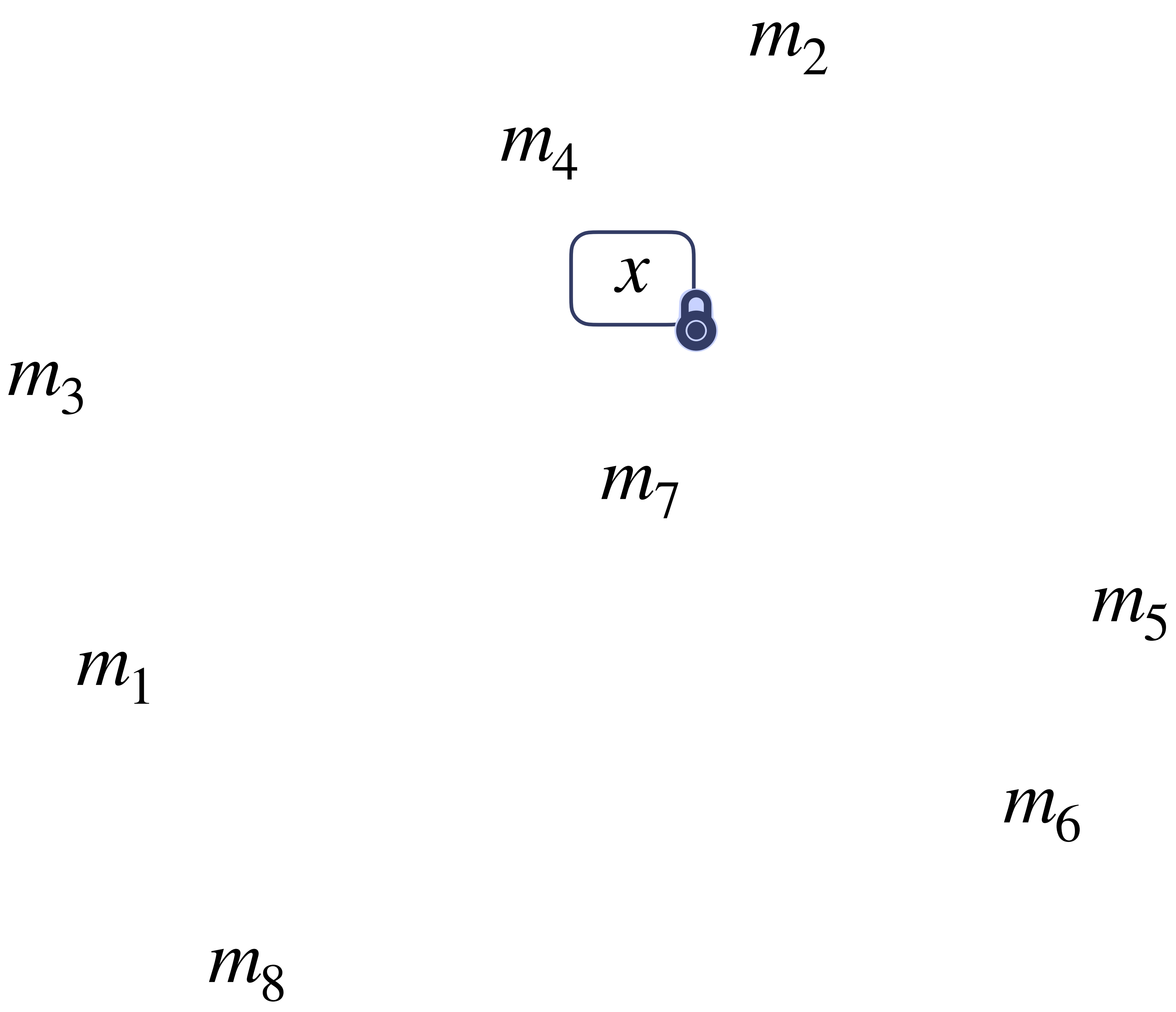
m_1

m_6

m_8

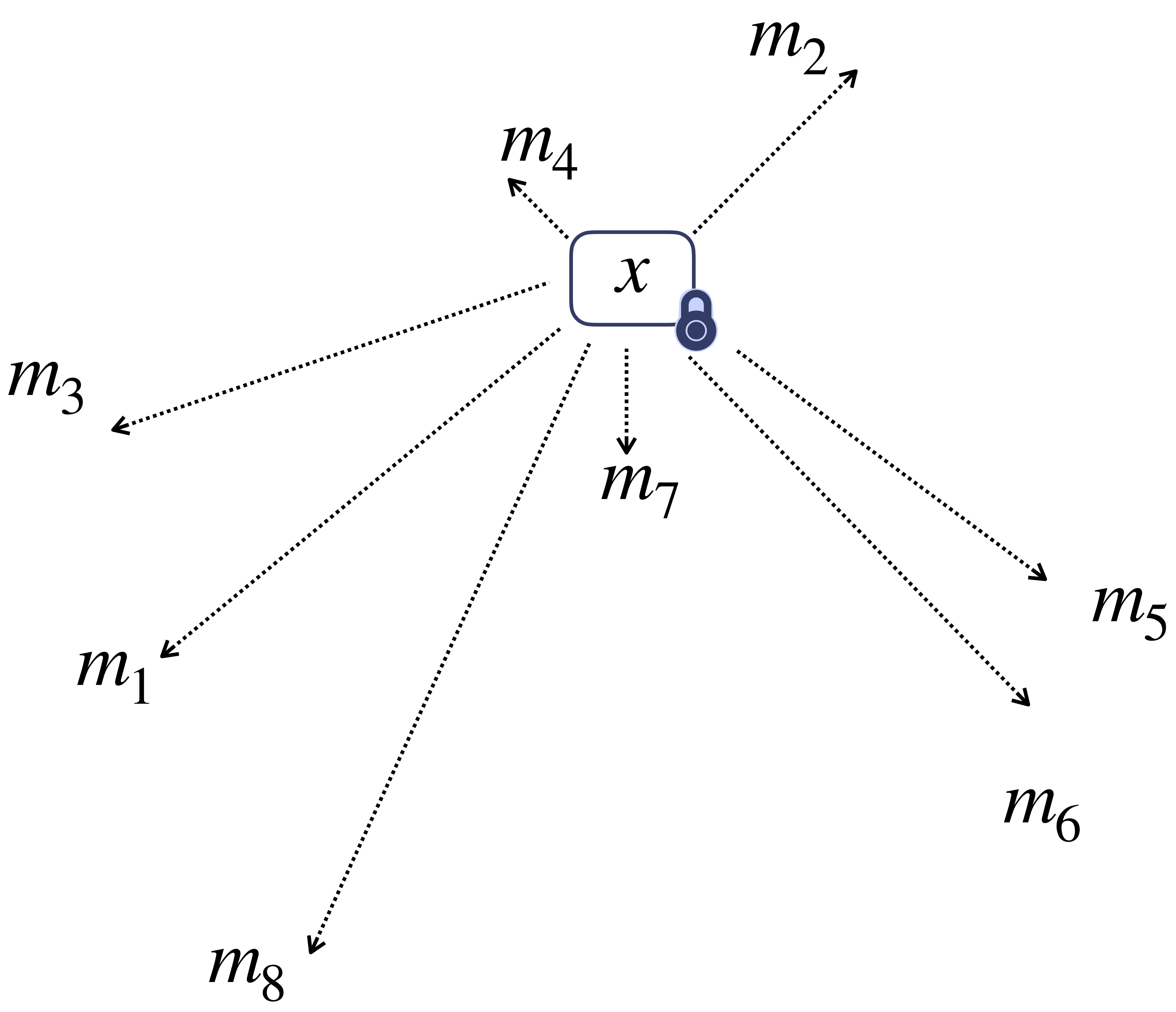
Private k-Nearest Neighbours

Distances computation in $\mathbb{Z}_{p_{dist}}$



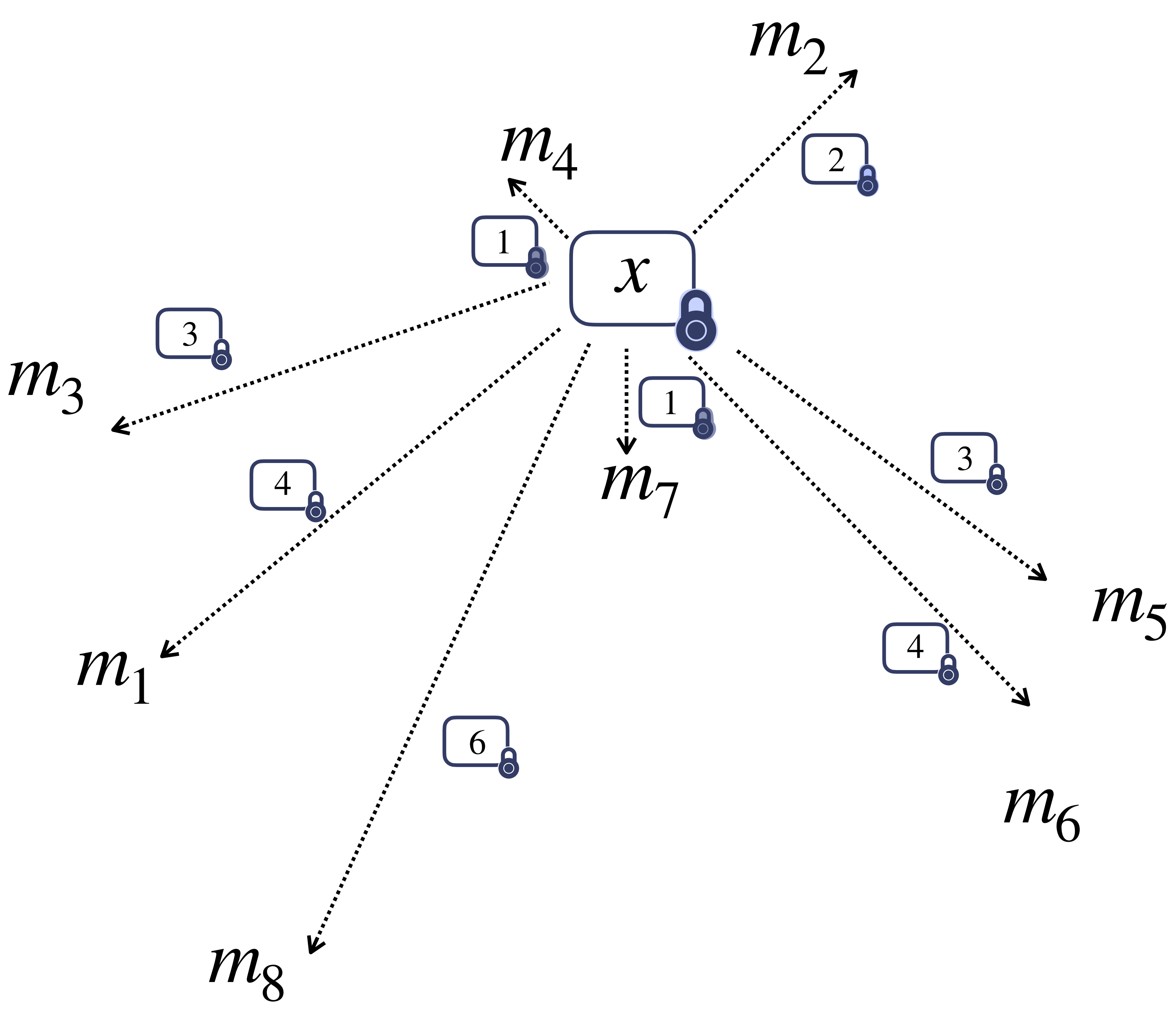
Private k-Nearest Neighbours

Distances computation in $\mathbb{Z}_{p_{dist}}$



Private k-Nearest Neighbours

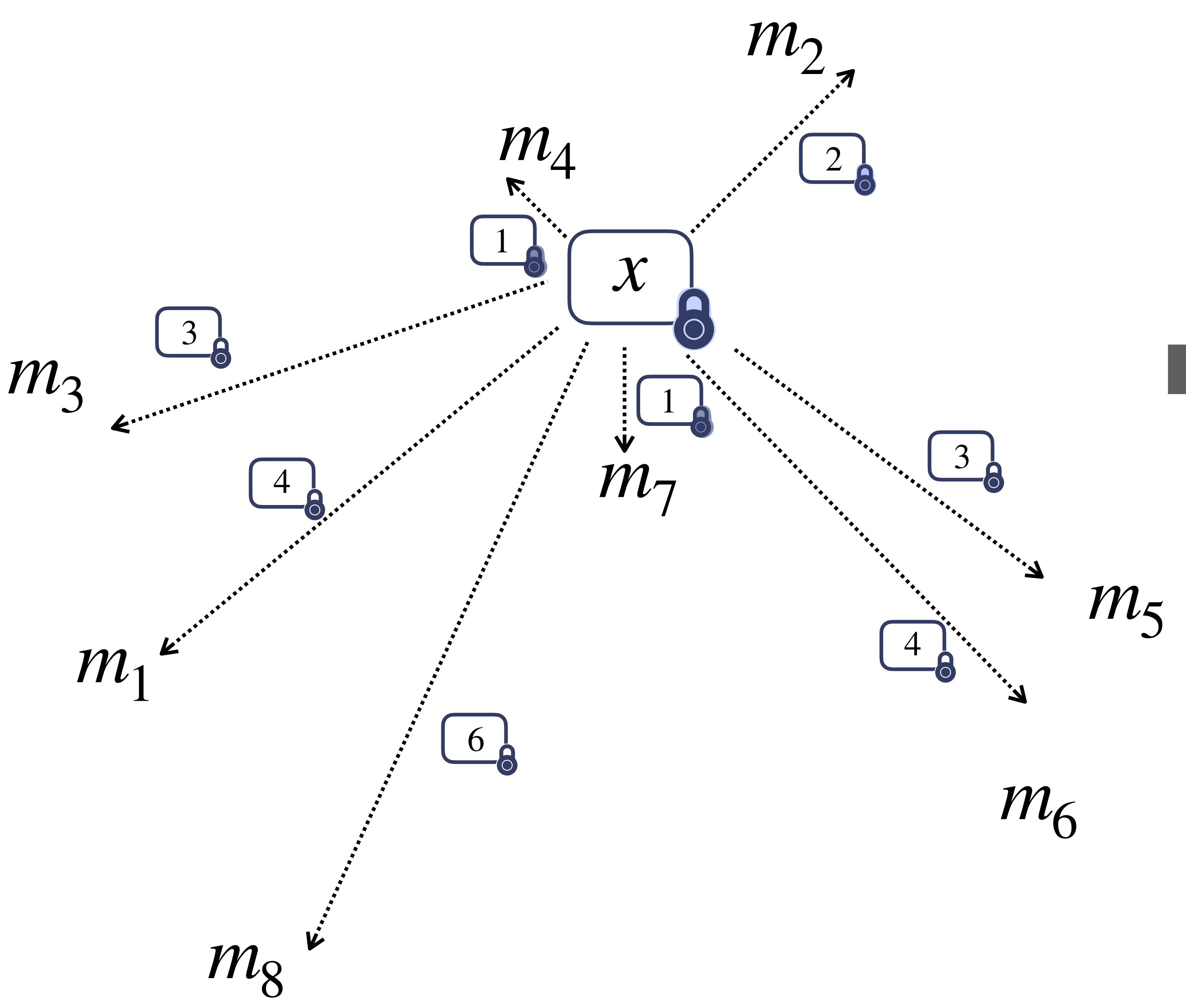
Distances computation in $\mathbb{Z}_{p_{dist}}$



Private k-Nearest Neighbours

Distances computation in $\mathbb{Z}_{p_{dist}}$

Precision reduction (if needed)



Distances plaintext from $\mathbb{Z}_{p_{dist}}$ to $\mathbb{Z}_p$

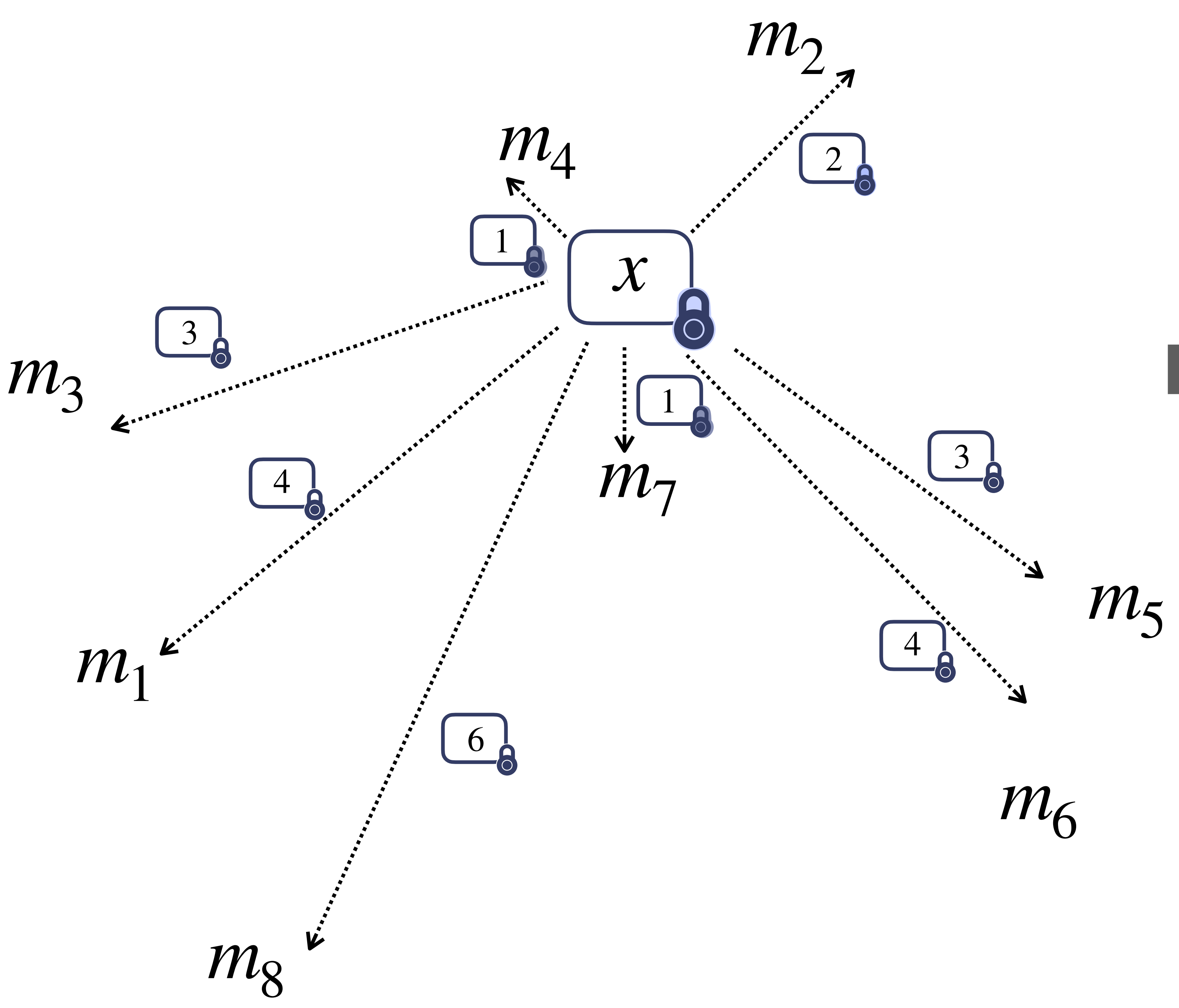
May introduce additional noise

Private k-Nearest Neighbours

Distances computation in $\mathbb{Z}_{p_{dist}}$

Precision reduction (if needed)

Top-k selection in $\mathbb{Z}_p$



Distances plaintext from $\mathbb{Z}_{p_{dist}}$ to $\mathbb{Z}_p$

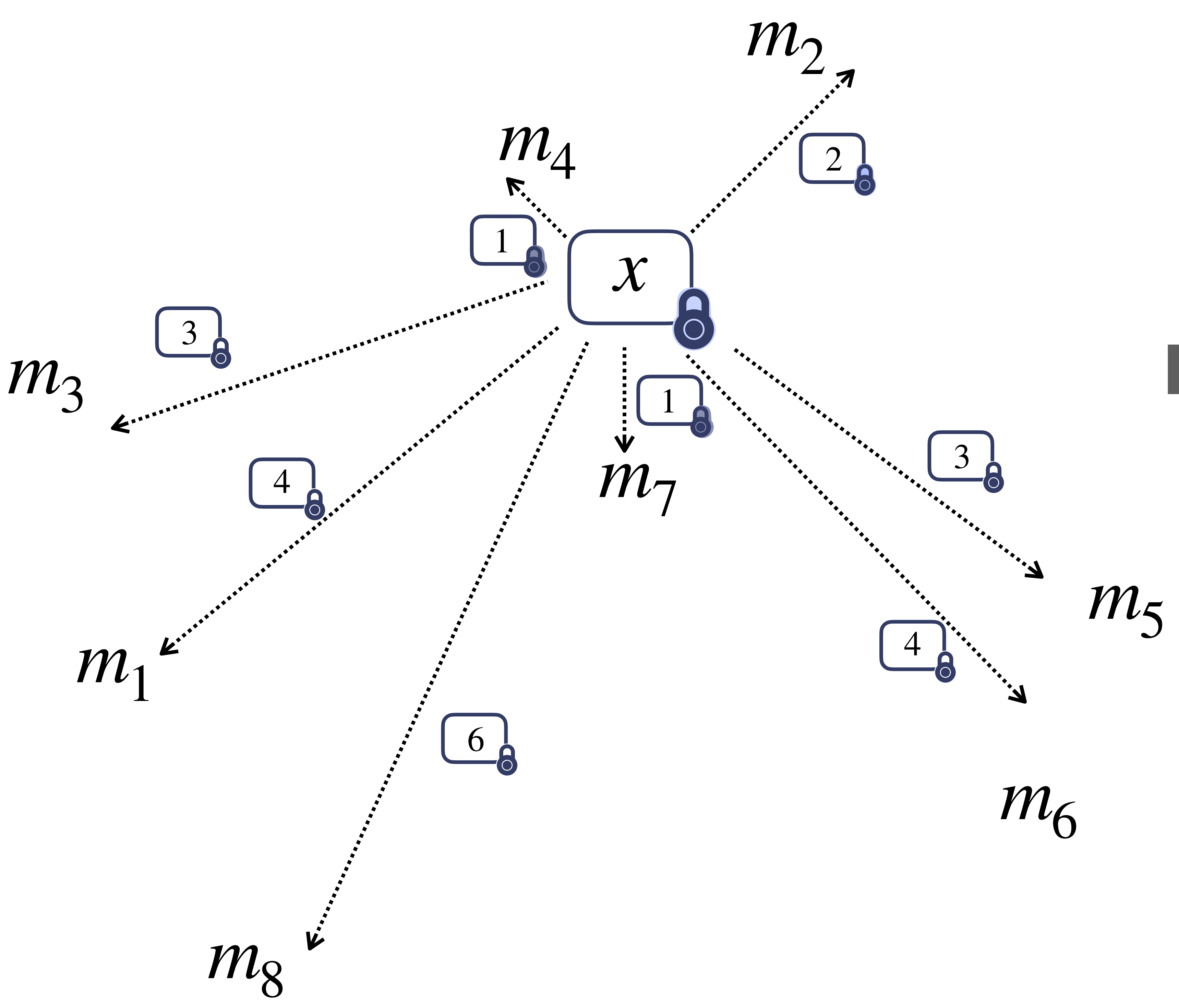
May introduce additional noise

Private k-Nearest Neighbours

Distances computation in $\mathbb{Z}_{p_{dist}}$

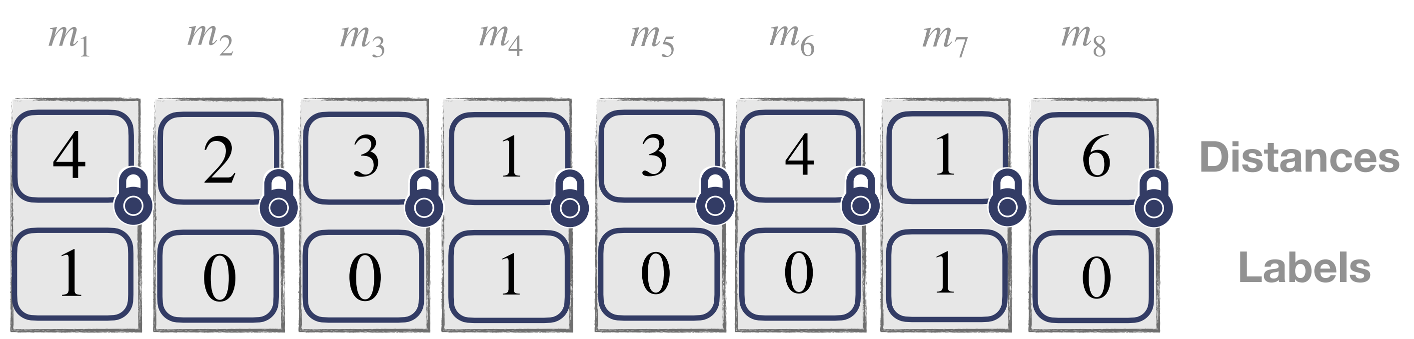
Precision reduction (if needed)

Top-k selection in $\mathbb{Z}_p$



Distances plaintext from $\mathbb{Z}_{p_{dist}}$ to $\mathbb{Z}_p$

May introduce additional noise

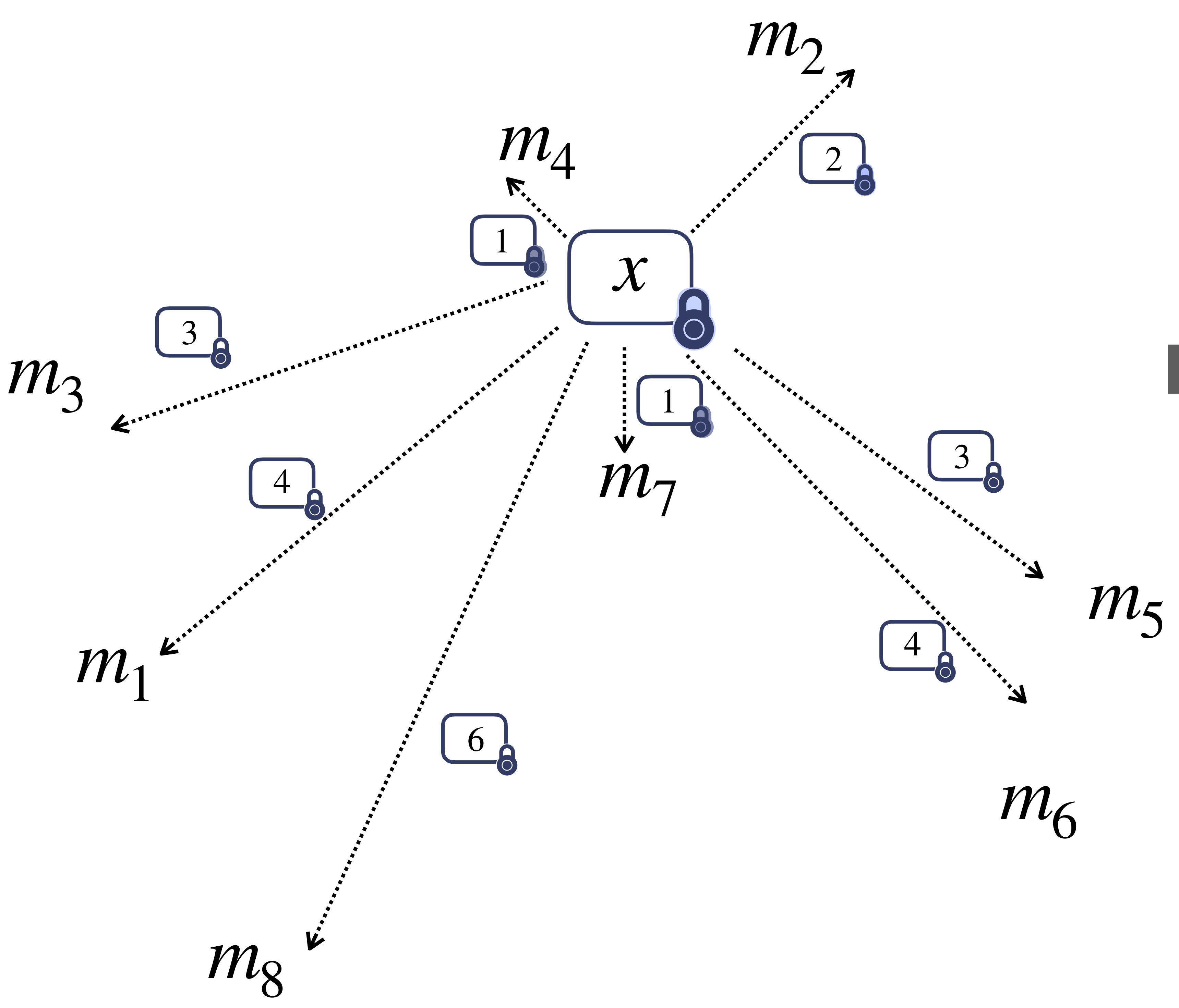


Private k-Nearest Neighbours

Distances computation in $\mathbb{Z}_{p_{dist}}$

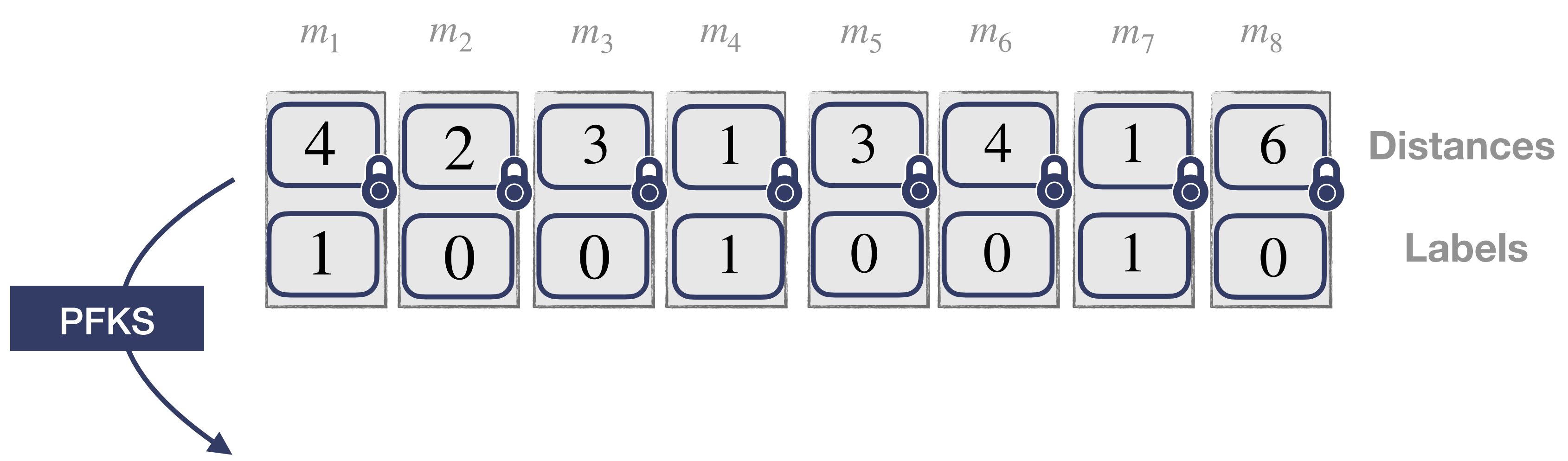
Precision reduction (if needed)

Top-k selection in $\mathbb{Z}_p$



Distances plaintext from $\mathbb{Z}_{p_{dist}}$ to $\mathbb{Z}_p$

May introduce additional noise

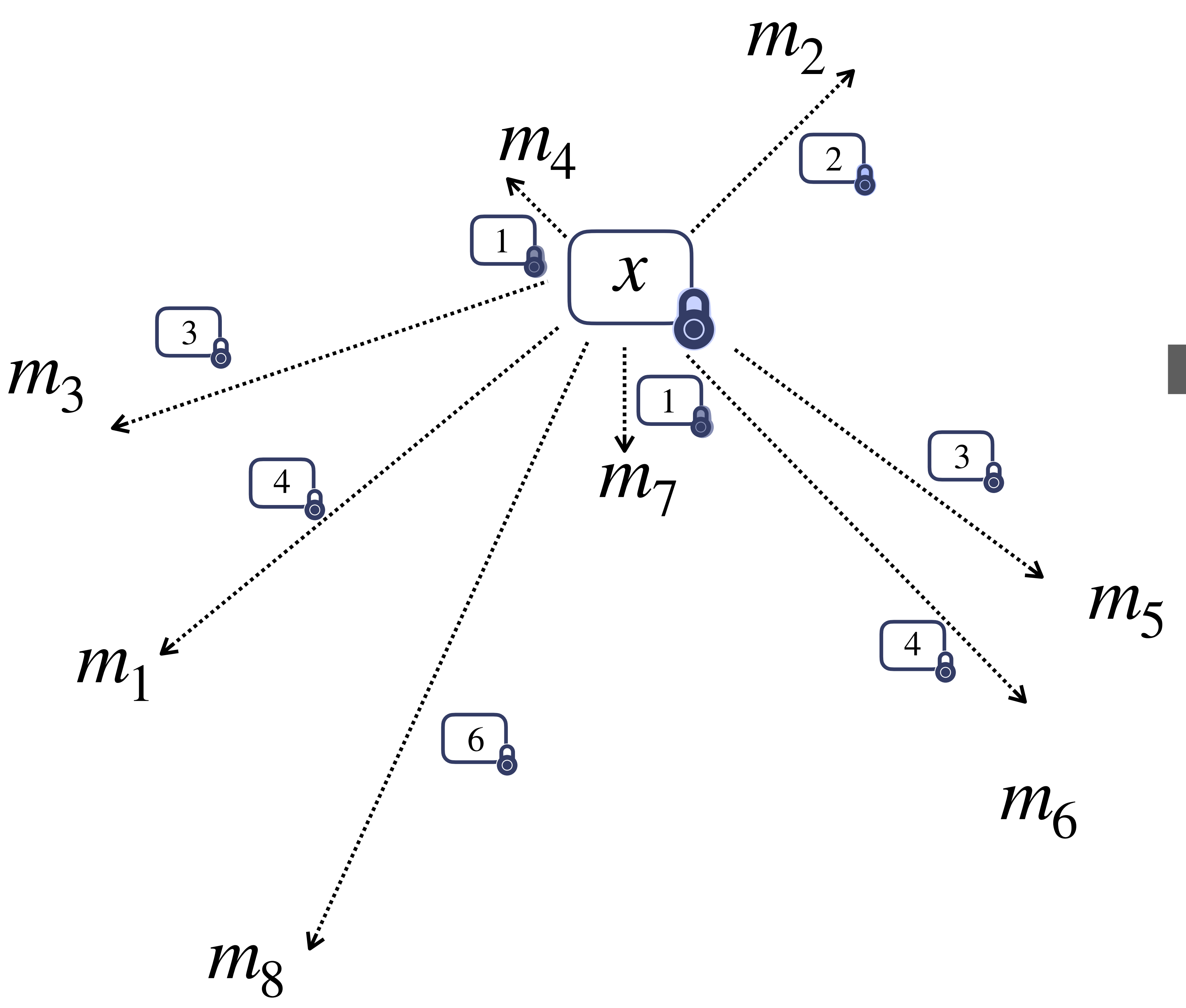


Private k-Nearest Neighbours

Distances computation in $\mathbb{Z}_{p_{dist}}$

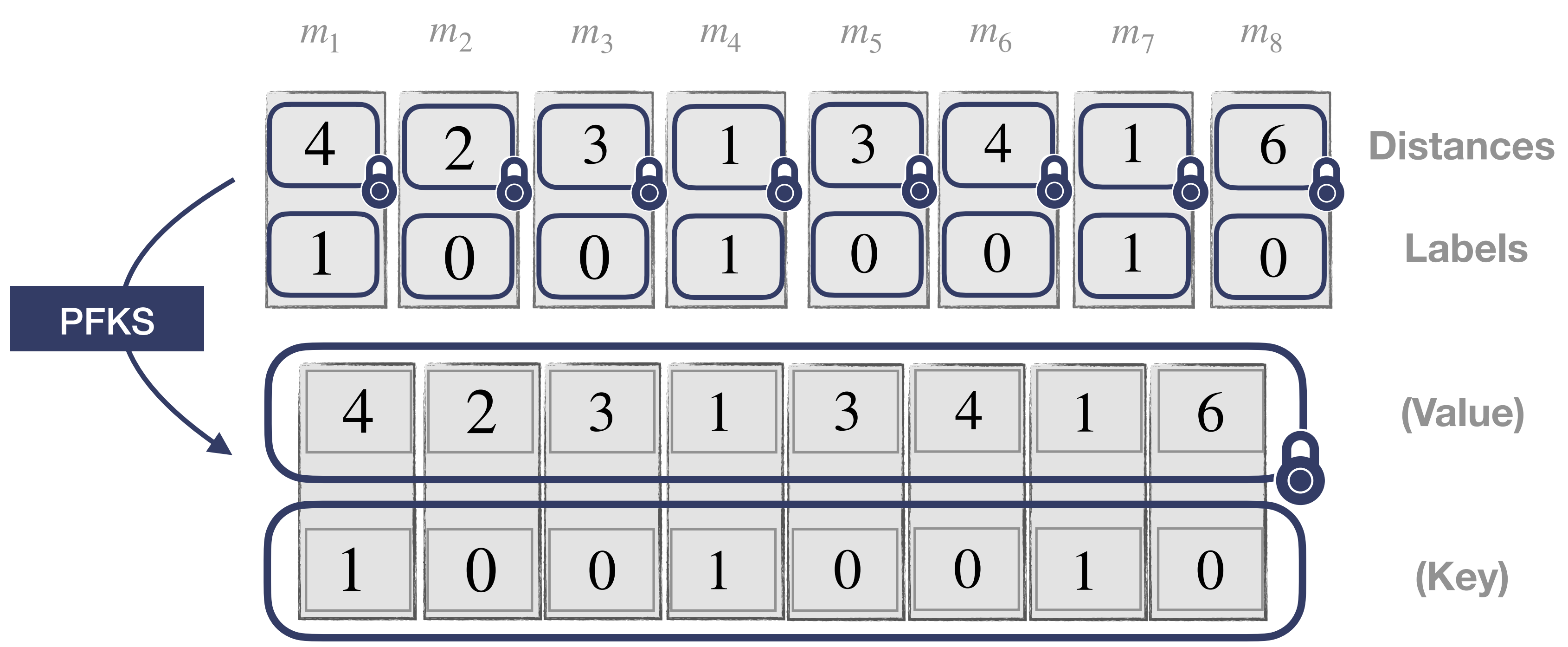
Precision reduction (if needed)

Top-k selection in $\mathbb{Z}_p$



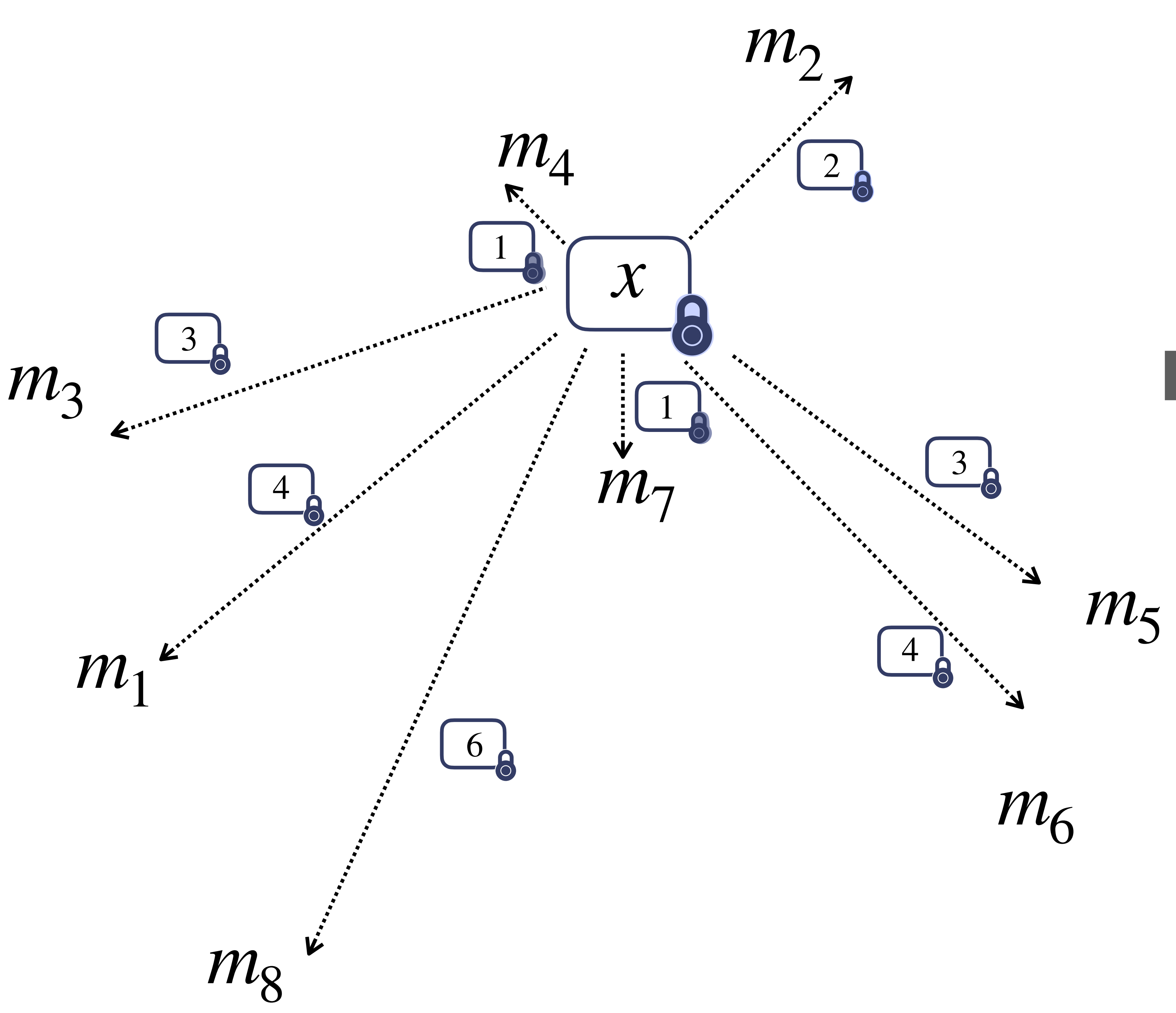
Distances plaintext from $\mathbb{Z}_{p_{dist}}$ to $\mathbb{Z}_p$

May introduce additional noise



Private k-Nearest Neighbours

Distances computation in $\mathbb{Z}_{p_{dist}}$

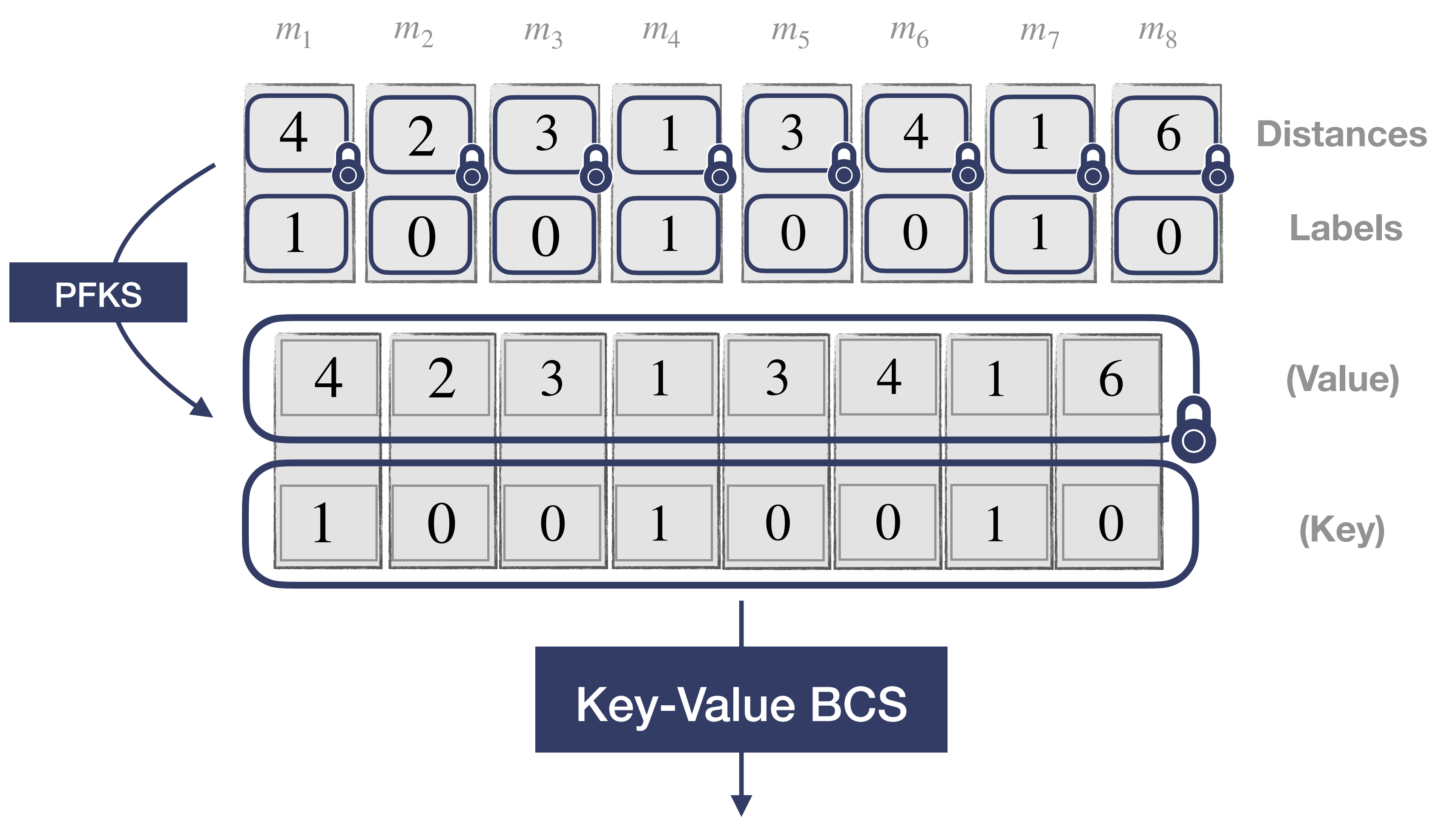


Precision reduction (if needed)

Distances plaintext from $\mathbb{Z}_{p_{dist}}$ to $\mathbb{Z}_p$

May introduce additional noise

Top-k selection in $\mathbb{Z}_p$

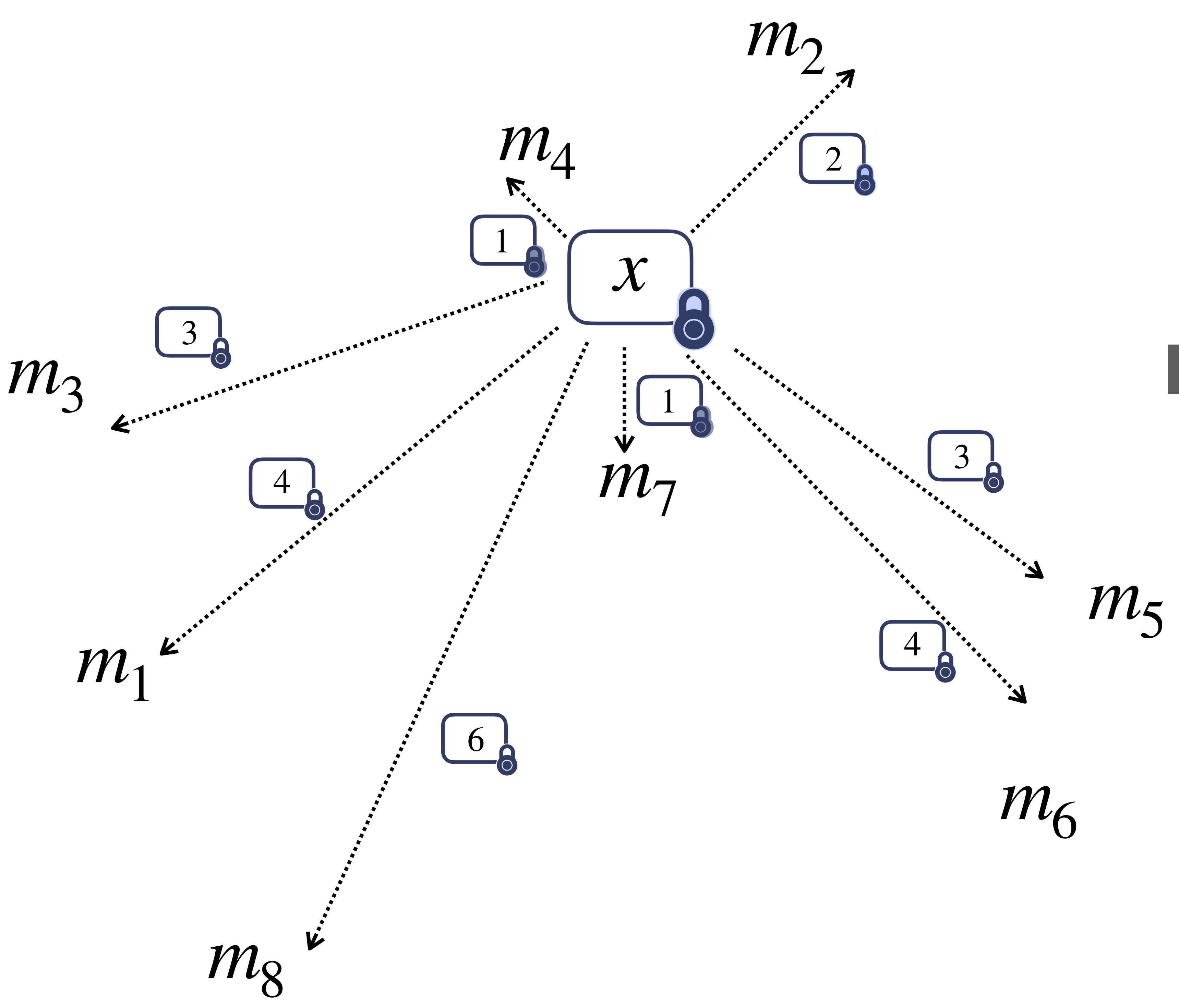


Private k-Nearest Neighbours

Distances computation in $\mathbb{Z}_{p_{dist}}$

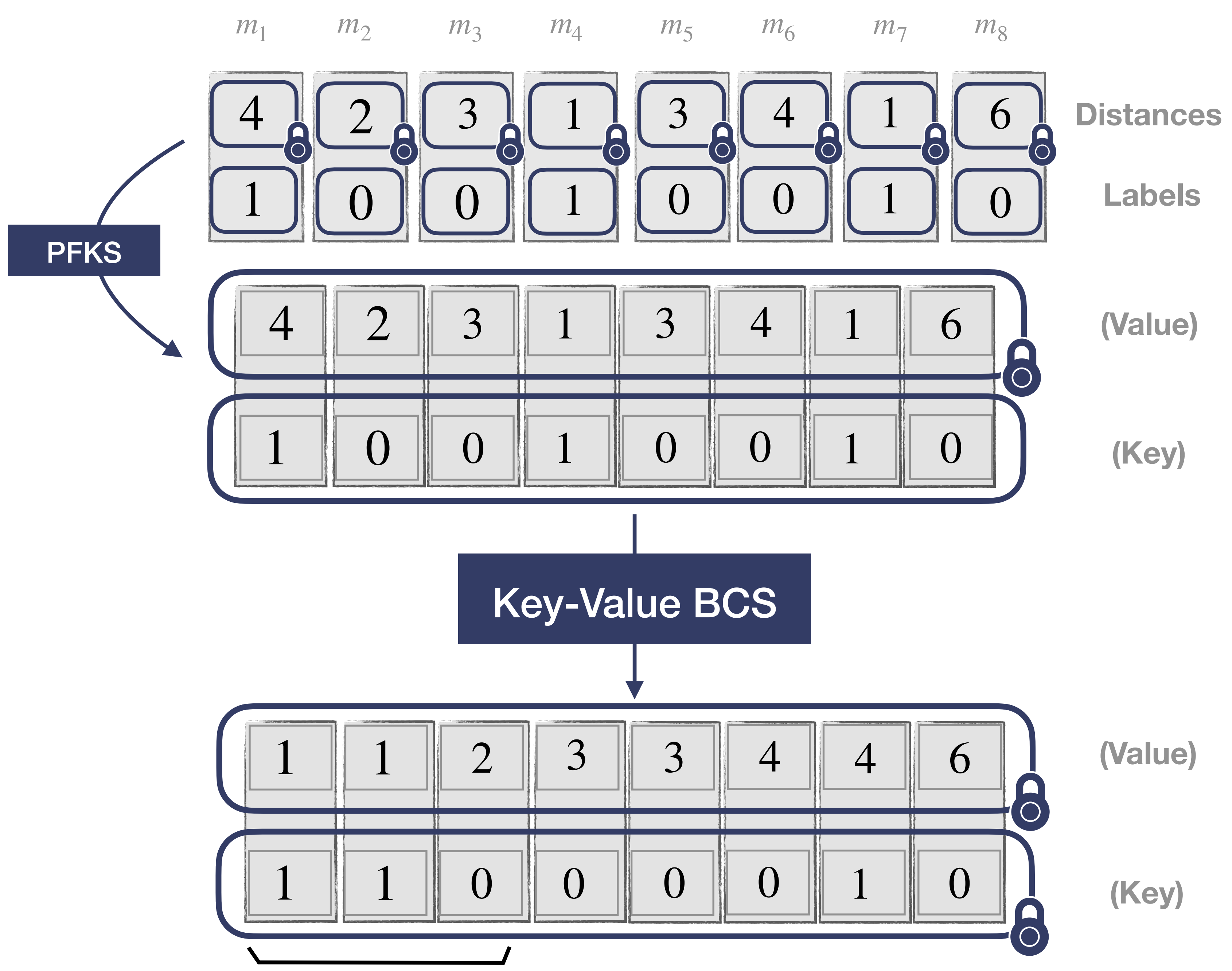
Precision reduction (if needed)

Top-k selection in $\mathbb{Z}_p$



Distances plaintext from $\mathbb{Z}_{p_{dist}}$ to $\mathbb{Z}_p$

May introduce additional noise

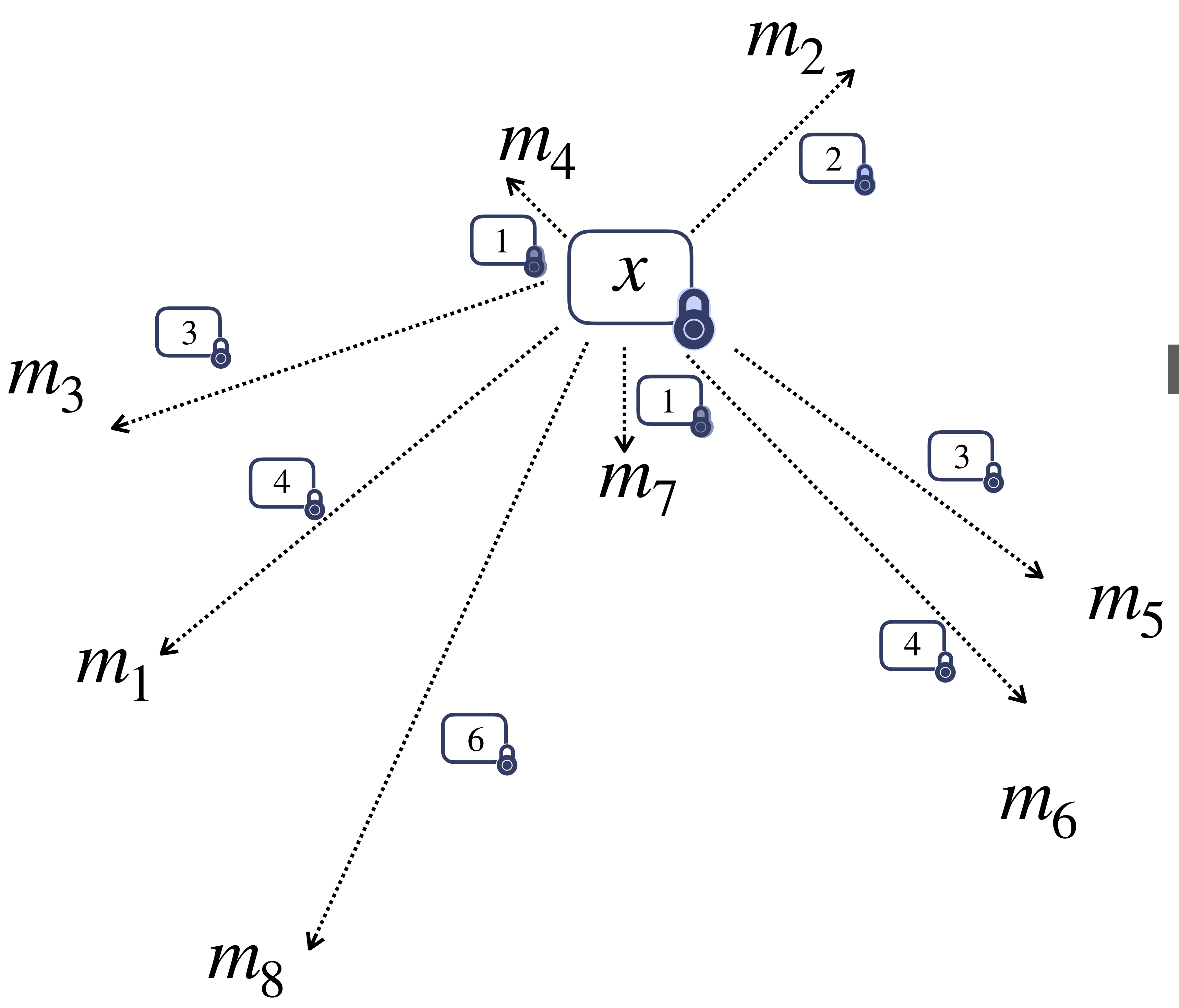


Private k-Nearest Neighbours

Distances computation in $\mathbb{Z}_{p_{dist}}$

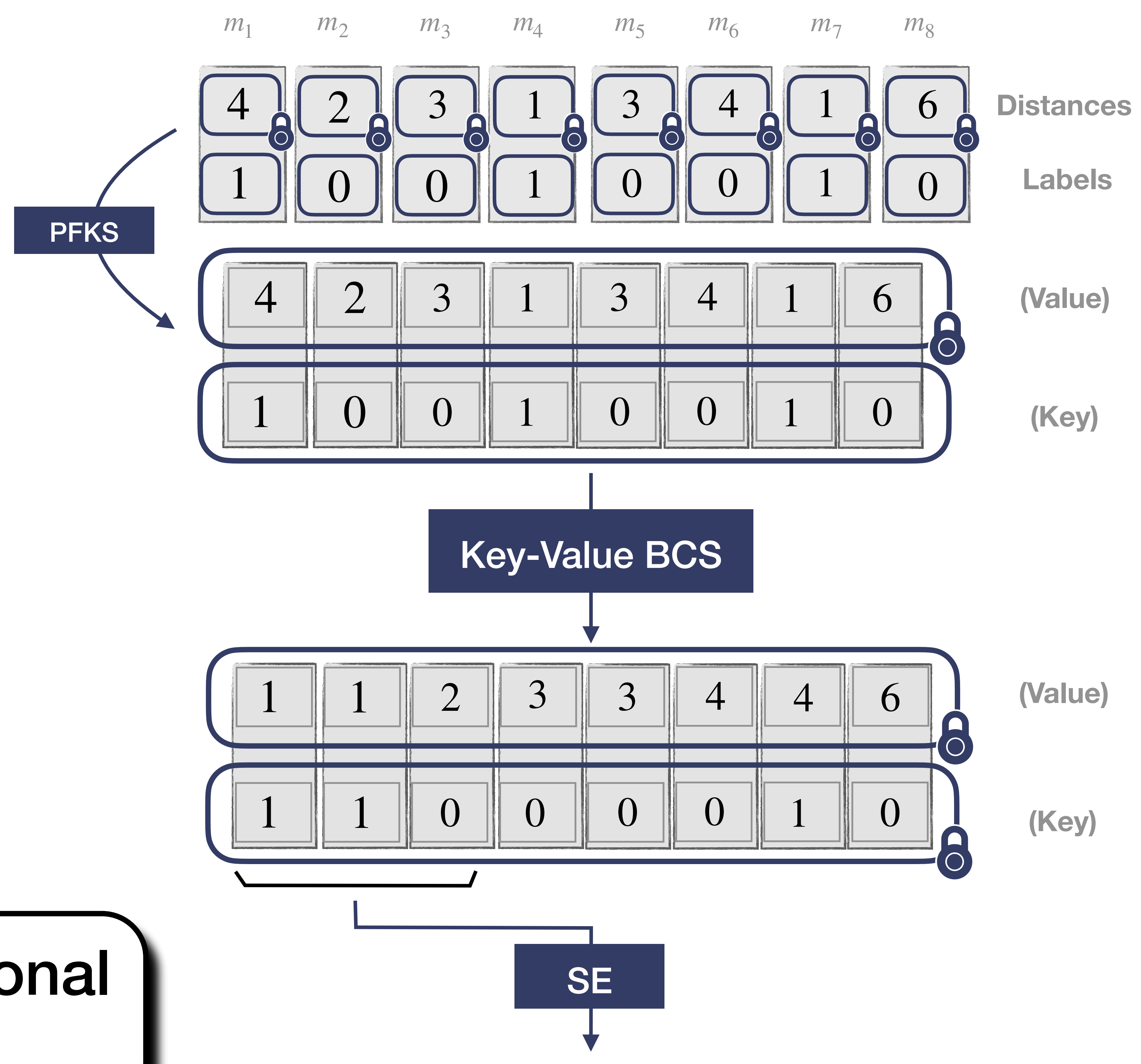
Precision reduction (if needed)

Top-k selection in $\mathbb{Z}_p$



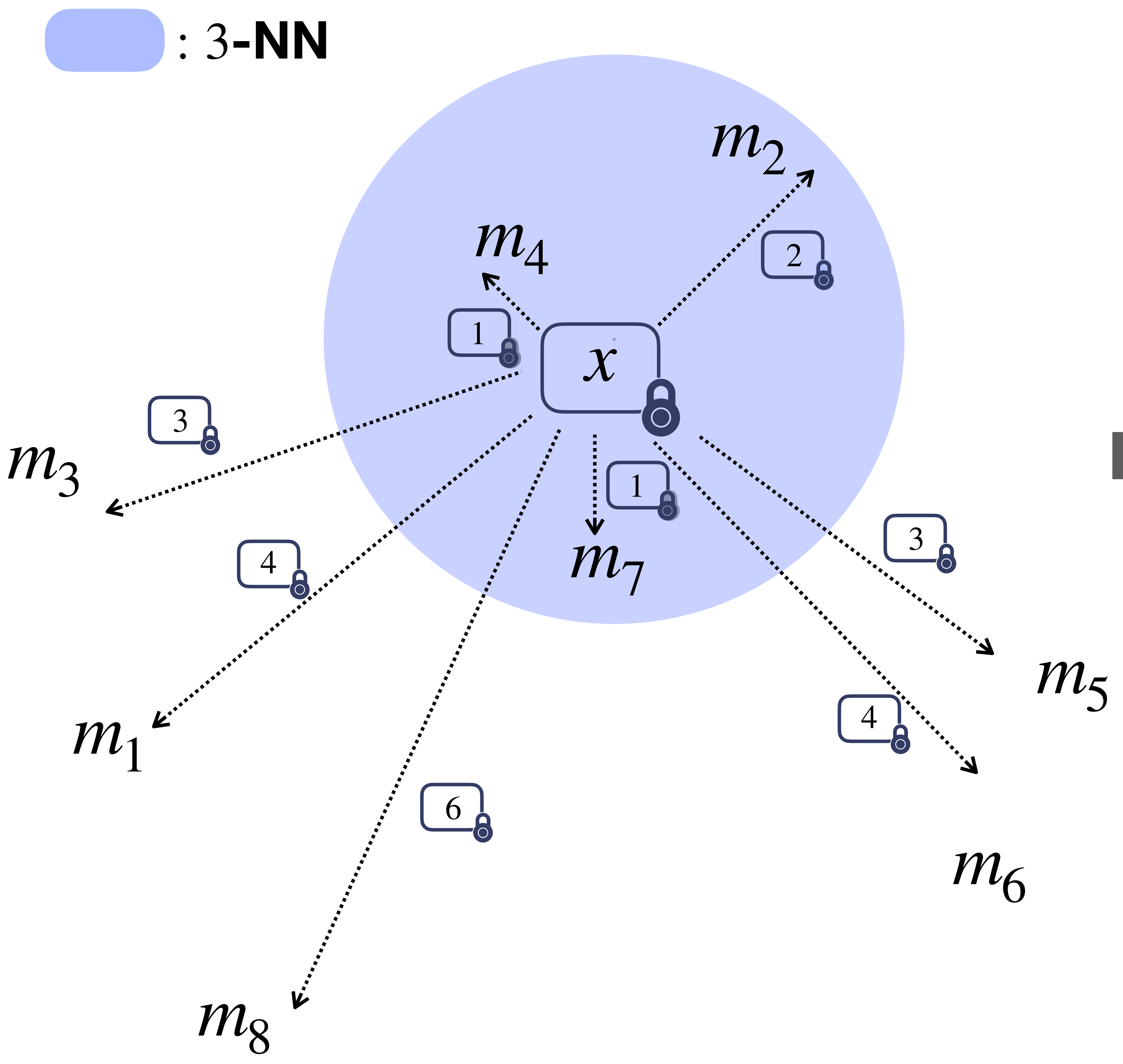
Distances plaintext from $\mathbb{Z}_{p_{dist}}$ to $\mathbb{Z}_p$

May introduce additional noise



Private k-Nearest Neighbours

Distances computation in $\mathbb{Z}_{p_{dist}}$

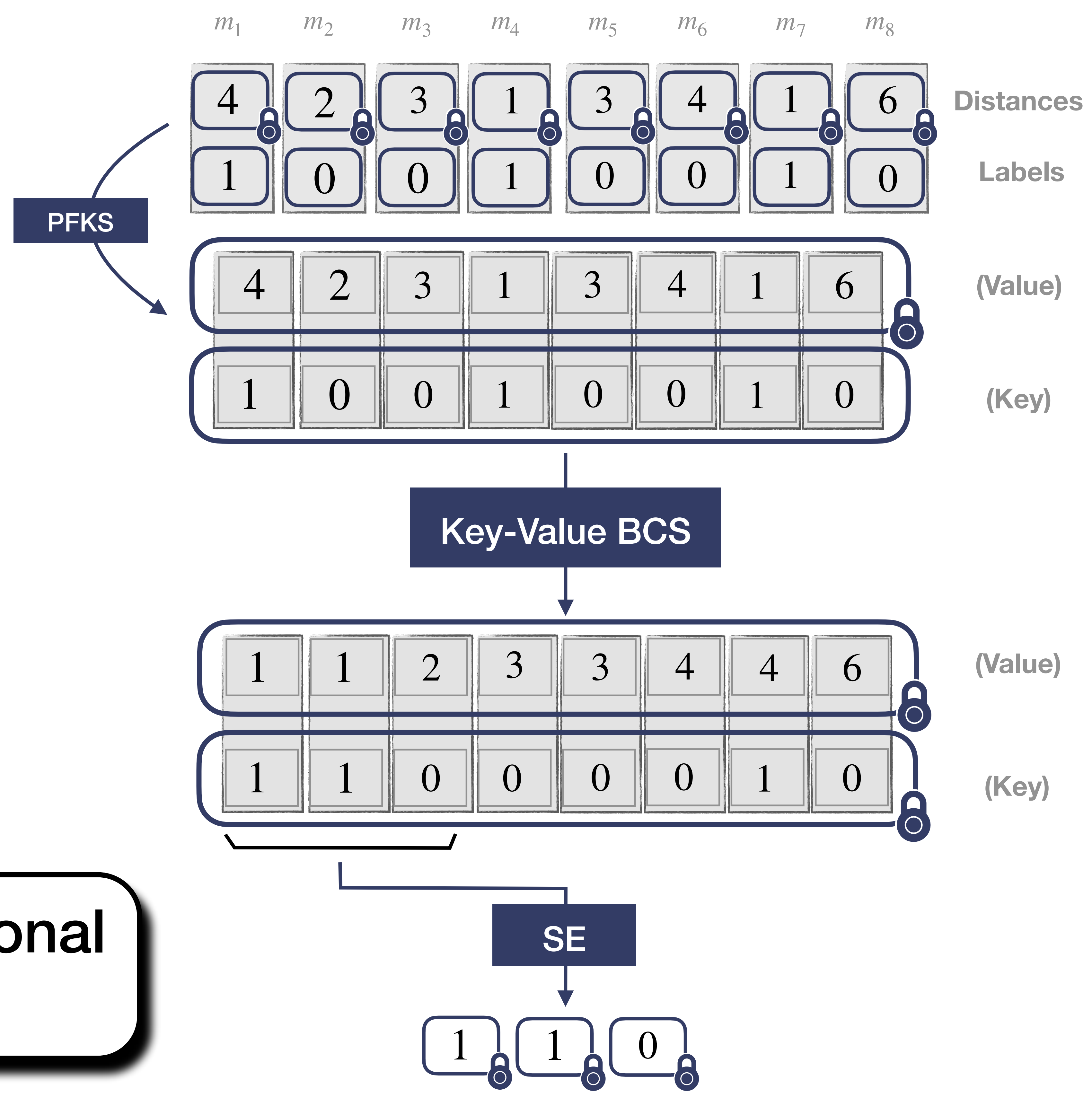


Precision reduction (if needed)

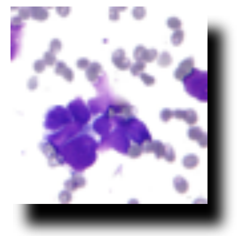
Distances plaintext from $\mathbb{Z}_{p_{dist}}$ to $\mathbb{Z}_p$

May introduce additional noise

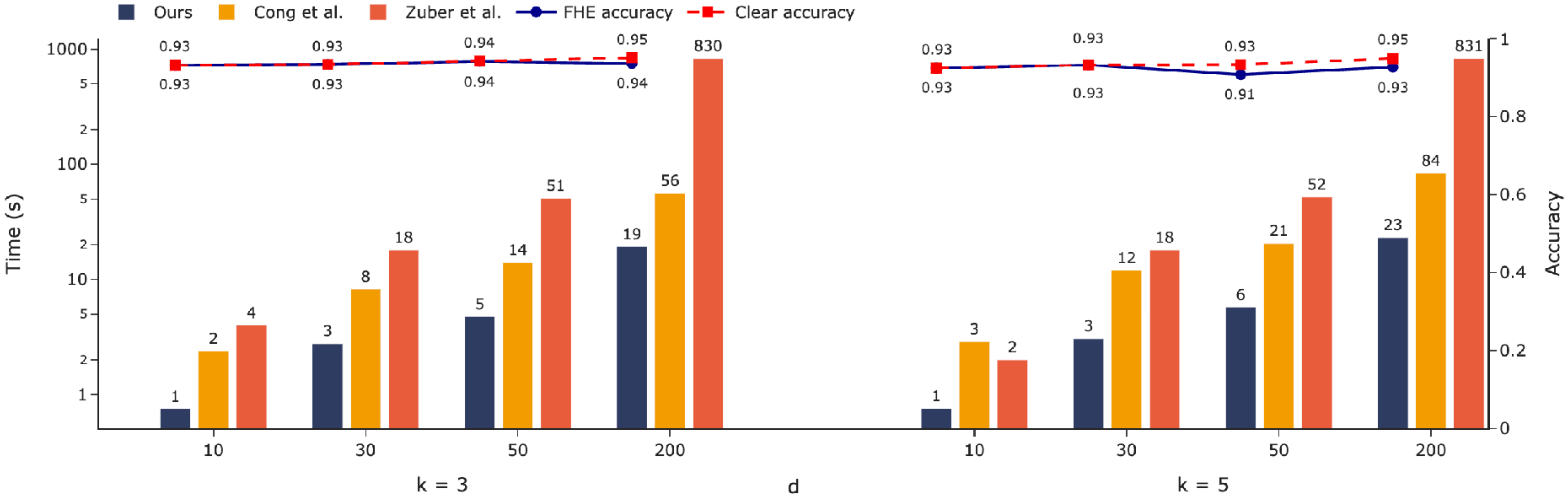
Top-k selection in $\mathbb{Z}_p$



Private k-Nearest Neighbours
Comparison with the SOA



Breast Cancer



[Zuber et al.] : Efficient homomorphic evaluation of k-NN classifiers, PoPETs, 2021(4), Martin Zuber and Renaud Sirdey.

[Cong et al.] : Revisiting Oblivious Top-k Selection with Applications to Secure k-NN Classification, SAC 2024, LNCS 15516.

Private k-Nearest Neighbours
Comparison with the SOA

MNIST

[Zuber et al.] : Efficient homomorphic evaluation of k-NN classifiers, PoPETs, 2021(4), Martin Zuber and Renaud Sirdey.

[Cong et al.] : Revisiting Oblivious Top-k Selection with Applications to Secure k-NN Classification, SAC 2024, LNCS 15516.

Conclusion

Conclusion

In this work, we :

- Designed the first **oblivious sorting algorithm** that operates without any comparisons
- Leveraged this sorting algorithm as a subroutine in a tournament-style to construct an **oblivious top-k** algorithm.
- Applied this oblivious top-k mechanism within a **privacy-preserving k-Nearest Neighbours model.**

Full paper: PoPETs 2025, paper no. 93

popets-2025-0093.pdf

sofianeazogagh/knn

Thank you !

Any questions ?

azogagh.sofiane@courrier.uqam.ca
sofianeazogagh.github.io