UNIVERSITÉ de Picardie Jules Verne

Projet de cryptographie

Licence informatique

Arbres de Merkle

Élaboré par :
BEN AMMAR Nader

Année universitaire : 2023-2024

Plan

Présentation général du projet

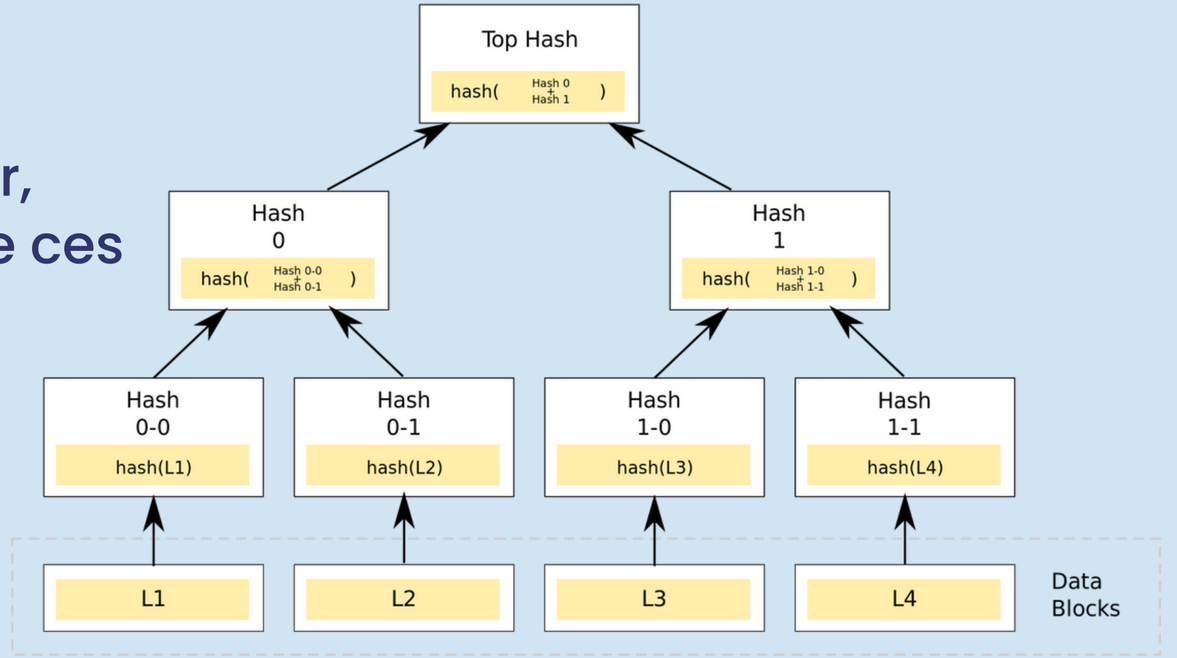
La structure utilisé

Les fonctionnalités

Présentation général du projet

Ce projet vise à expérimenter le contrôle d'intégrité de données en utilisant les arbres de Merkle, en se basant sur l'implémentation de la fonction de hachage TTH64.

L'objectif est de vérifier l'intégrité d'un ensemble de données, comme un fichier, même en ne possédant qu'une partie de ces données.



La structure utilisé

La structure utilisé:

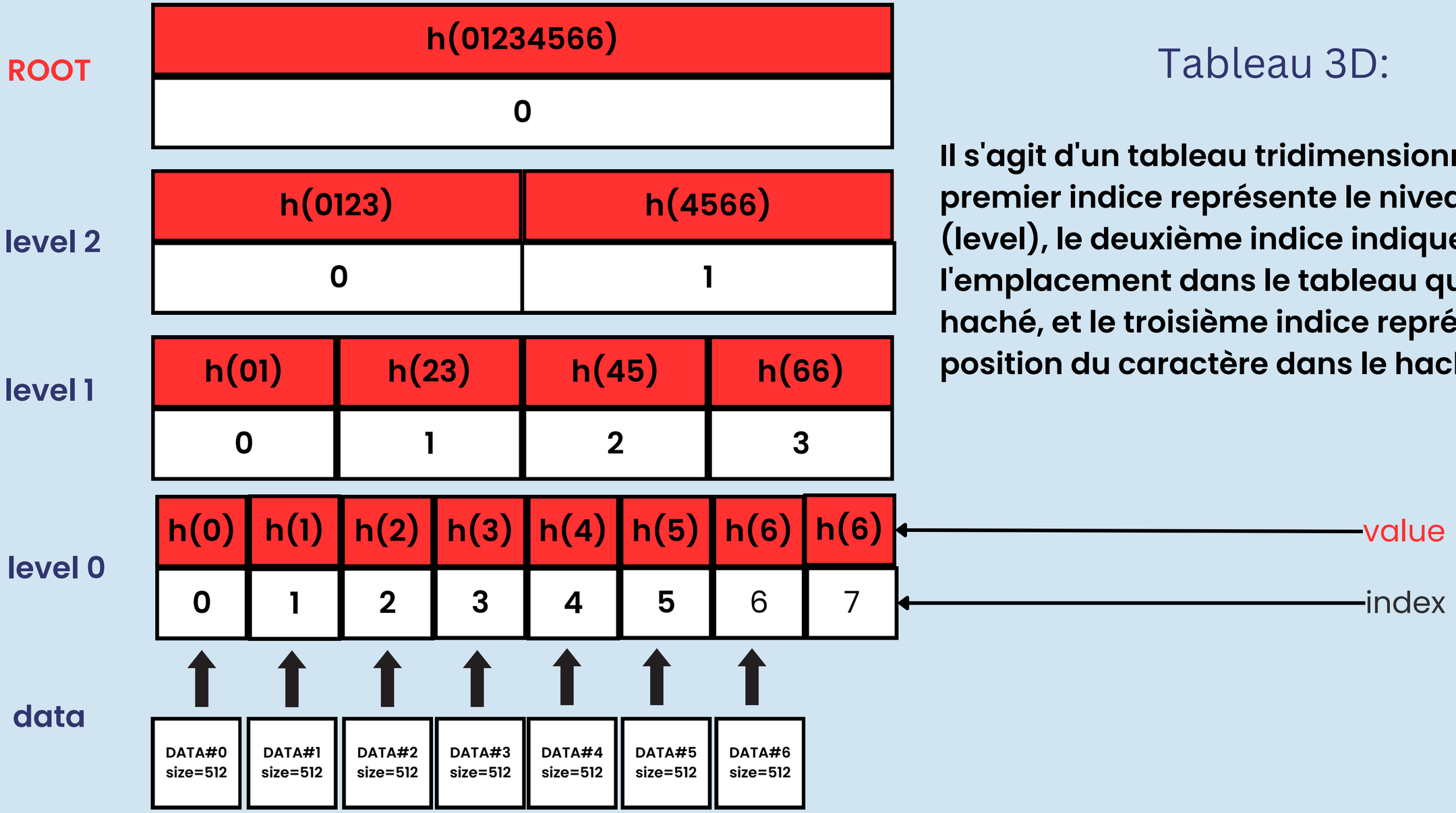


Tableau 3D:

Il s'agit d'un tableau tridimensionnel où le premier indice représente le niveau de l'arbre (level), le deuxième indice indique l'emplacement dans le tableau qui contient le haché, et le troisième indice représente la position du caractère dans le haché.

Les fonctionnalités

Extraction des blocs d’un fichier

FILE
size= 1025

=

```
3f9e6a3738379c2bb5c07a1c36b6a8f3f9b6aeeb19cc0d6f3f7ab4c9d6829baefab3f2b6b3f9b6c27b
99b4b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9
bc0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9b
c0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc
0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc0
b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc0b
6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc2f3f9e6a3738379c2bb5c07a1c36b6a8f3f9b6aeeb19cc0
d6f3f7ab4c9d6829baefab3f2b6b3f9b6c27b99b4b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a
6c2b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6
c2b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2
b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b
6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6
c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b2f3f9e6a3738379
c2bb5c07a1c36b6a8f3f9b6aeeb19cc0d6f3f7ab4c9d6829baefab3f2b6b3f9b6c27b99b4b6a6c2b6
aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6a
eeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6ae
eb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6aee
b3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6aeeb
3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6aeeb3f
9b6b3f9b6b3f9b6c27b9bc2f3f9e6a3738379c2bb5c07a1c36b6a8f3f9b6aeeb19cc0d6f3f7ab4c9d6
829baefab3f2b6b3f9b6c27b99b4b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c
2b6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2
b6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b
6aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b6
aeeb3f9b6b3f9b6c27b9bc0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b9bc0b6a6c2b6c0b6a6c2b6a
eeb3f9b6b3f9b6c27b9bc0b6a6c2b6aeeb3f9b6b3f9b6b3f9b6c27b2f
```

divisé sur 3 blocs

DATA#0
size=512

DATA#1
size=512

DATA#2
size=512

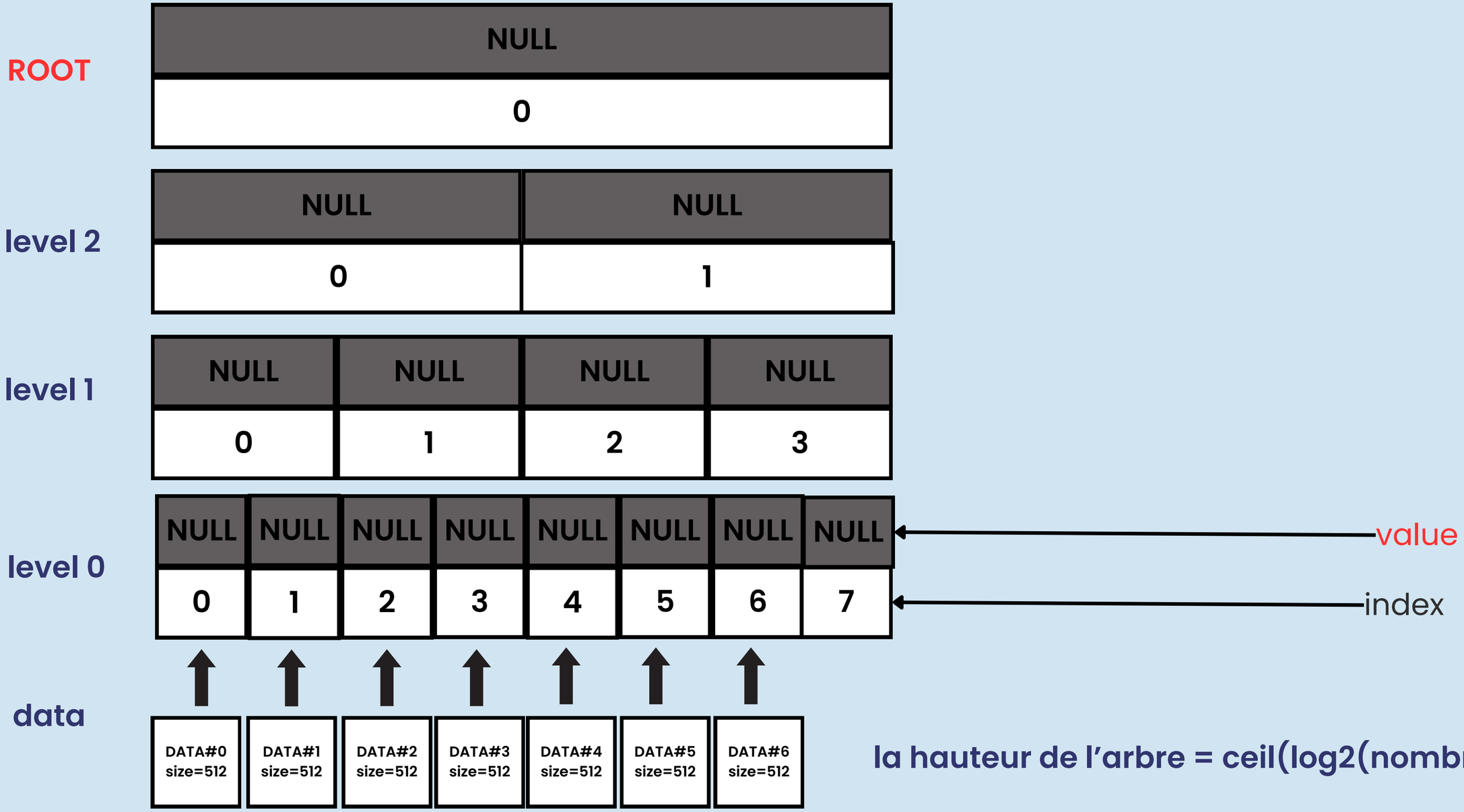
Padding

DATA#2
size=512

=

```
2f20000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
```

Construction de l’arbre de Merkle



la hauteur de l’arbre = ceil(log2(nombre de blocs)) +1

Construction de l’arbre de Merkle

ROOT

NULL
0

level 2

NULL	NULL
0	1

level 1

NULL	NULL	NULL	NULL
0	1	2	3

level 0

h(0)	h(1)	h(2)	h(3)	h(4)	h(5)	h(6)	h(6)
0	1	2	3	4	5	6	7

Remarque

lorsqu’un niveau a un nombre impaire de hachés, on duplique le dernier haché.

Construction de l’arbre de Merkle

ROOT

NULL
0

level 2

NULL	NULL
0	1

level 1

h(01)	h(23)	h(45)	h(66)
0	1	2	3

level 0

h(0)	h(1)	h(2)	h(3)	h(4)	h(5)	h(6)	h(6)
0	1	2	3	4	5	6	7

Construction de l’arbre de Merkle

ROOT

NULL
0

level 2

h(0123)	h(4566)
0	1

level 1

h(01)	h(23)	h(45)	h(66)
0	1	2	3

level 0

h(0)	h(1)	h(2)	h(3)	h(4)	h(5)	h(6)	h(6)
0	1	2	3	4	5	6	7

Construction de l’arbre de Merkle

ROOT

h(01234566)
0

level 2

h(0123)	h(4566)
0	1

level 1

h(01)	h(23)	h(45)	h(66)
0	1	2	3

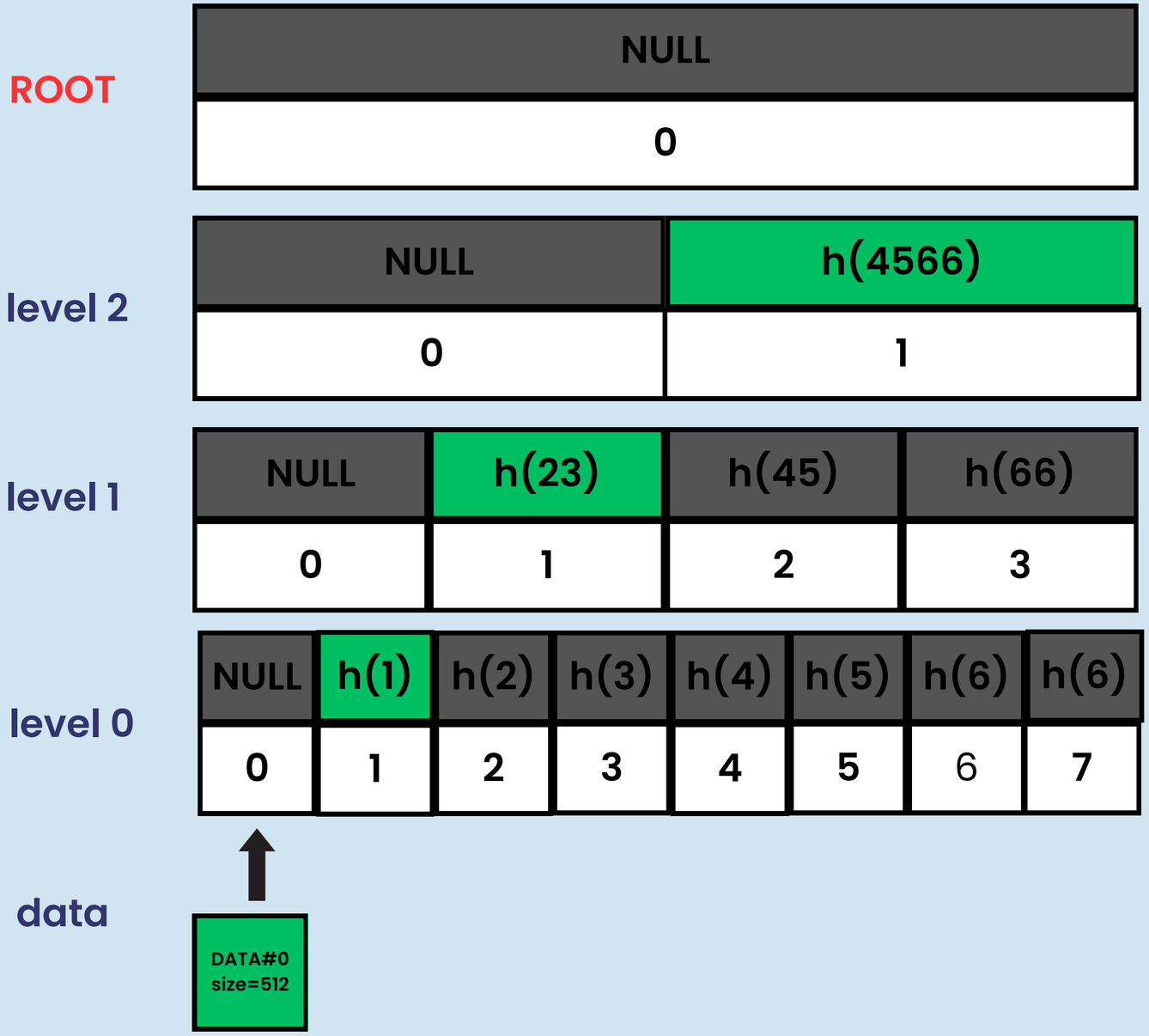
level 0

h(0)	h(1)	h(2)	h(3)	h(4)	h(5)	h(6)	h(6)
0	1	2	3	4	5	6	7

Remarque

lorsqu’un niveau a un nombre impaire de hachés, on duplique le dernier haché. exemple h(66).

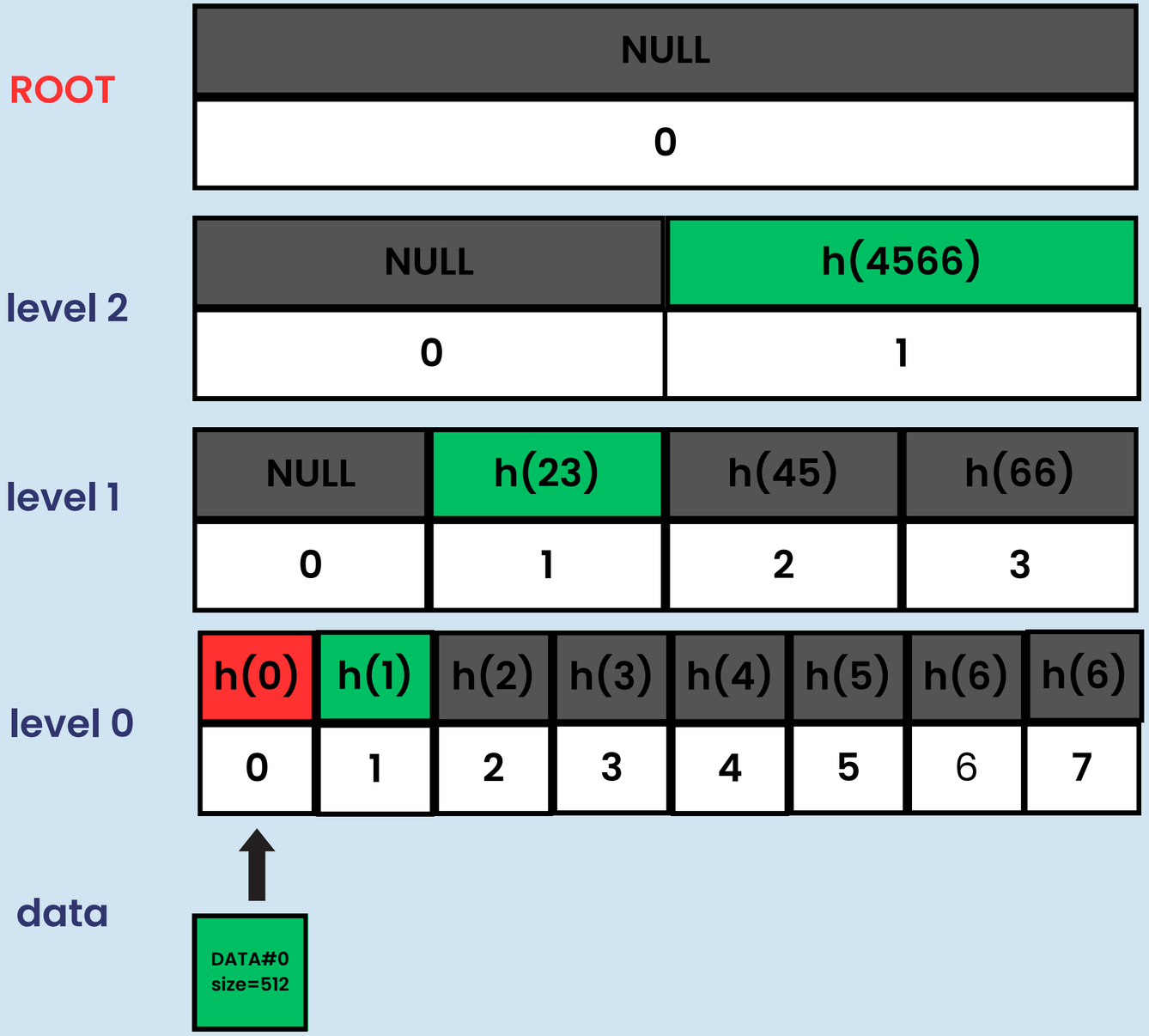
Contrôle d’intégrité global pour une partie



Exemple

Pour controler l’intégrité global pour la data#0 on aura besoin de seulement les informations en vert.

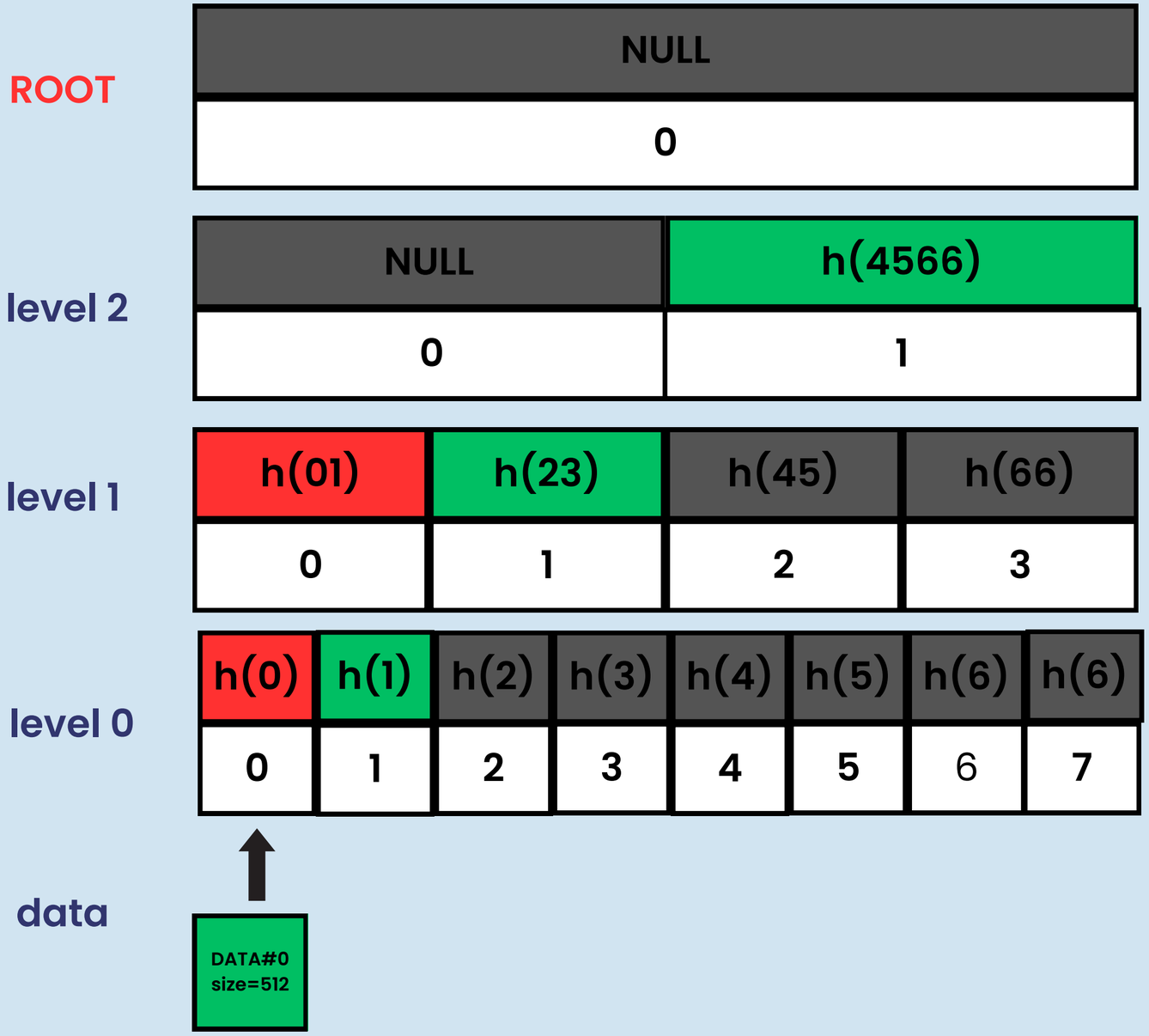
Contrôle d’intégrité global pour une partie



Exemple

Pour controler l’intégrité global pour la data#0 on aura besoin de seulement les informations en vert.

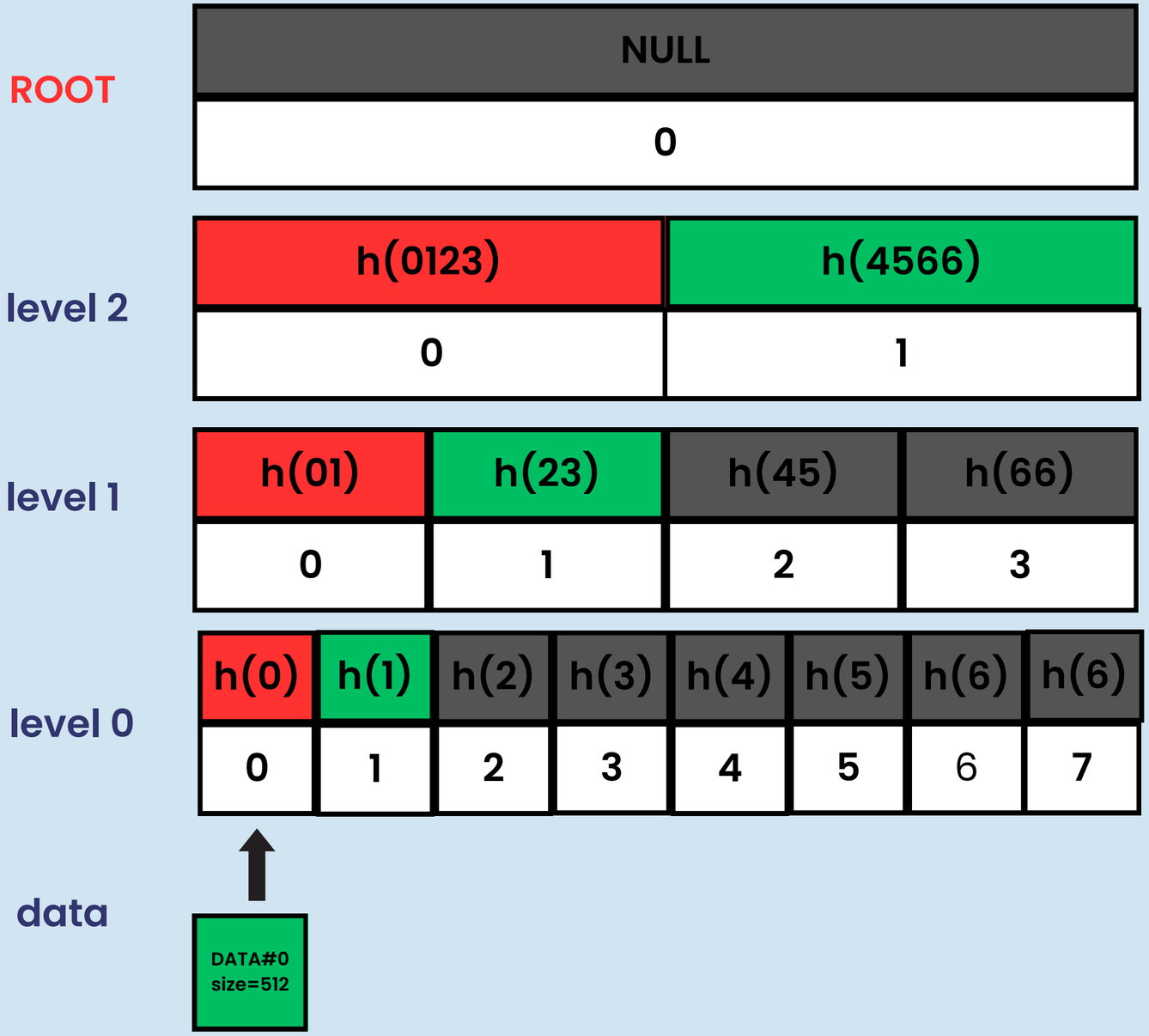
Contrôle d’intégrité global pour une partie



Exemple

Pour controler l’intégrité global pour la data#0 on aura besoin de seulement les informations en vert.

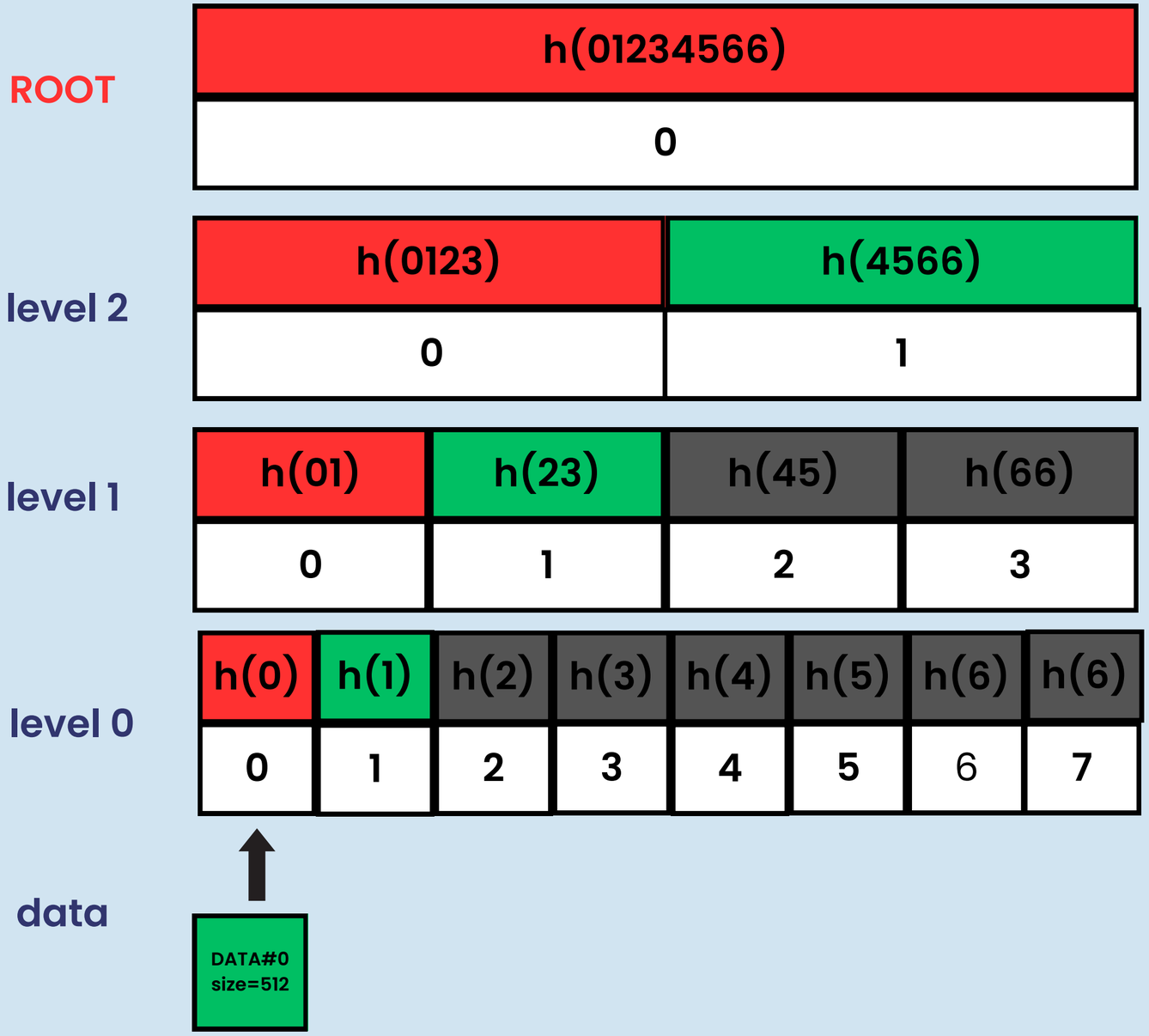
Contrôle d’intégrité global pour une partie



Exemple

Pour controler l’intégrité global pour la data#0 on aura besoin de seulement les informations en vert.

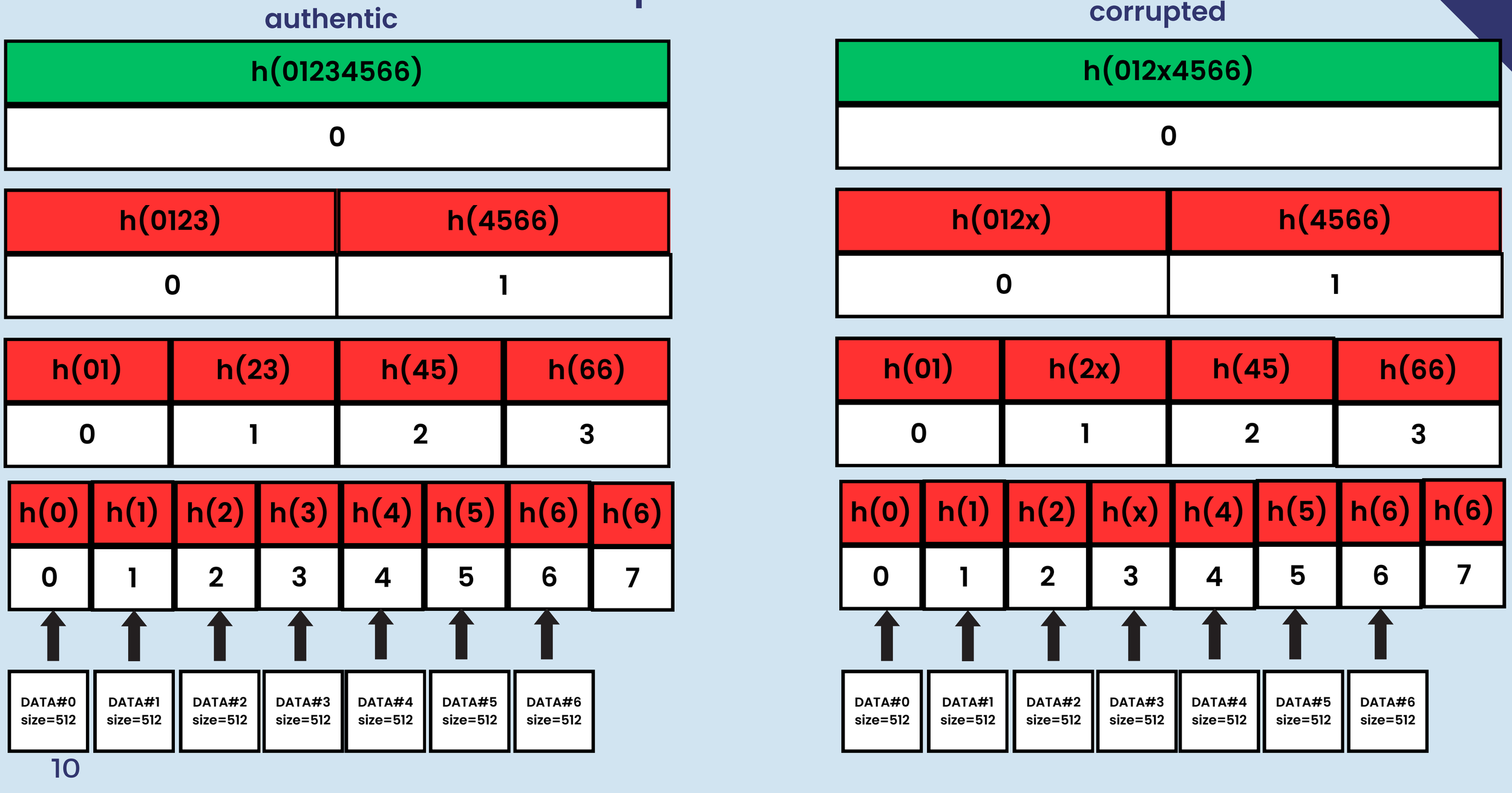
Contrôle d’intégrité global pour une partie



Exemple

Pour controler l’intégrité global pour la data#0 on aura besoin de seulement les informations en vert.

Trouver la donnée corrompu



Trouver la donnée corrompu

authentic

h(01234566)
0

h(0123)	h(4566)
0	1

h(01)	h(23)	h(45)	h(66)
0	1	2	3

h(0)	h(1)	h(2)	h(3)	h(4)	h(5)	h(6)	h(6)
0	1	2	3	4	5	6	7

DATA#0 size=512	DATA#1 size=512	DATA#2 size=512	DATA#3 size=512	DATA#4 size=512	DATA#5 size=512	DATA#6 size=512

corrupted

h(012x4566)
0

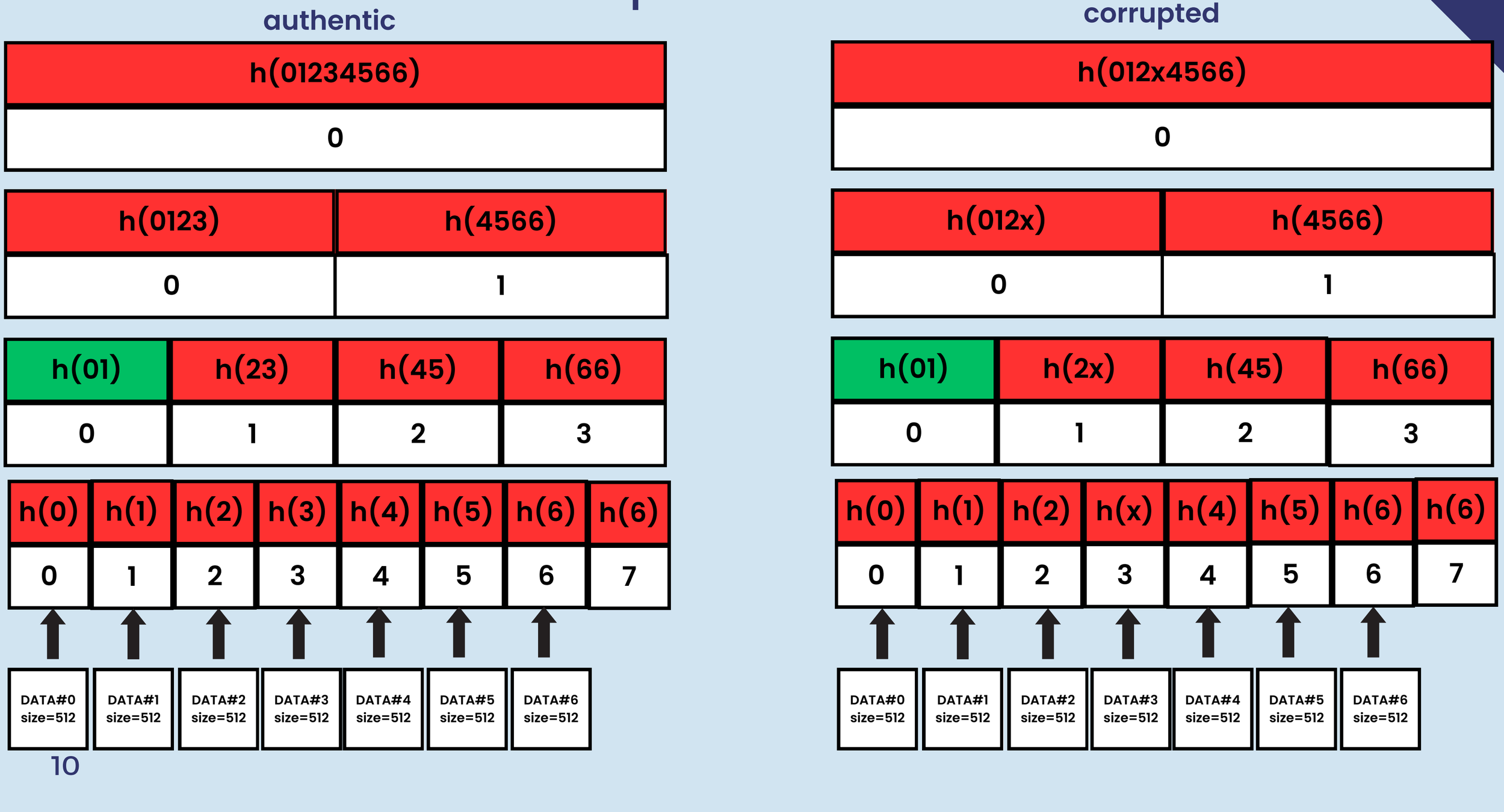
h(012x)	h(4566)
0	1

h(01)	h(2x)	h(45)	h(66)
0	1	2	3

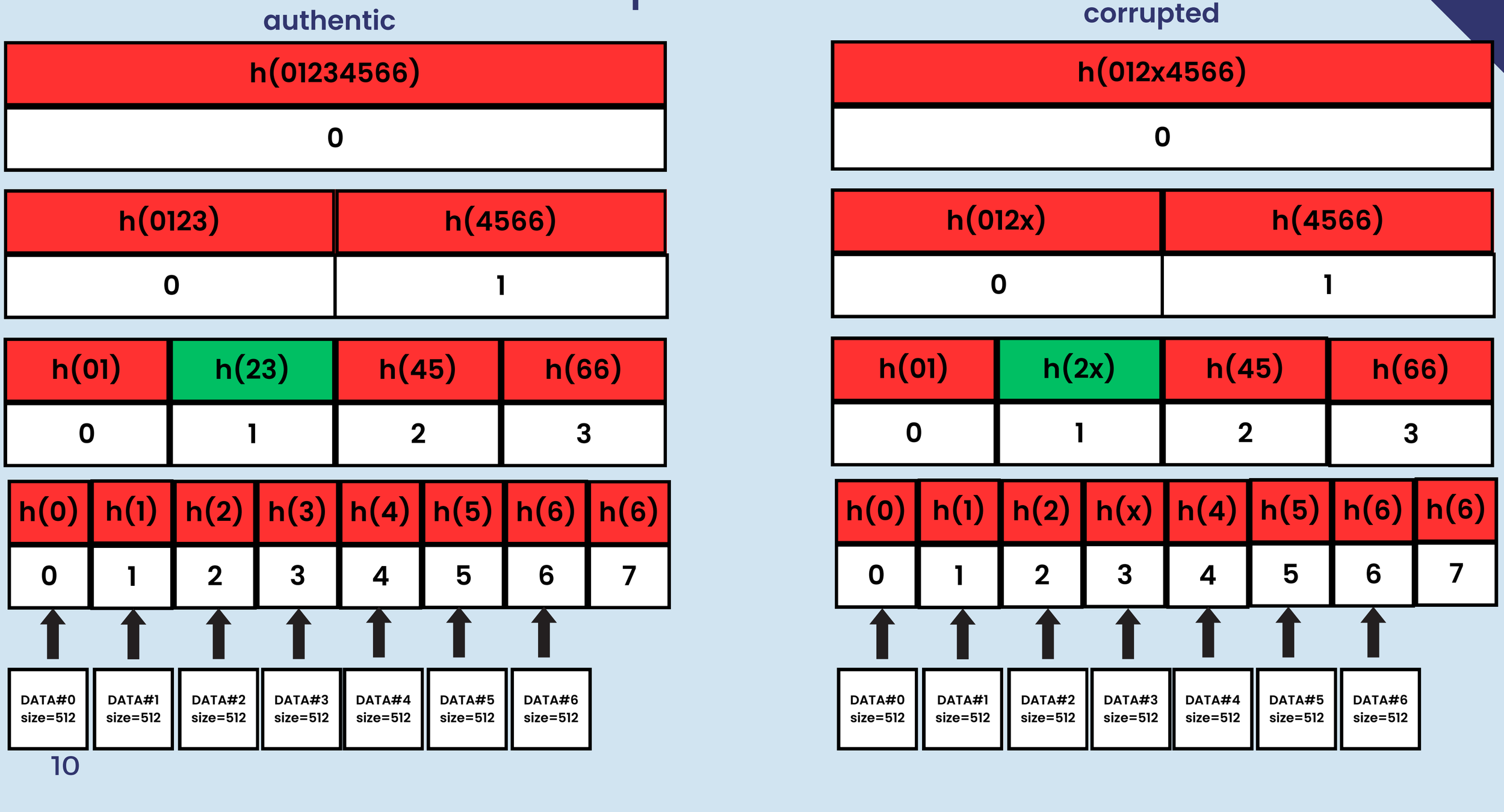
h(0)	h(1)	h(2)	h(x)	h(4)	h(5)	h(6)	h(6)
0	1	2	3	4	5	6	7

DATA#0 size=512	DATA#1 size=512	DATA#2 size=512	DATA#3 size=512	DATA#4 size=512	DATA#5 size=512	DATA#6 size=512

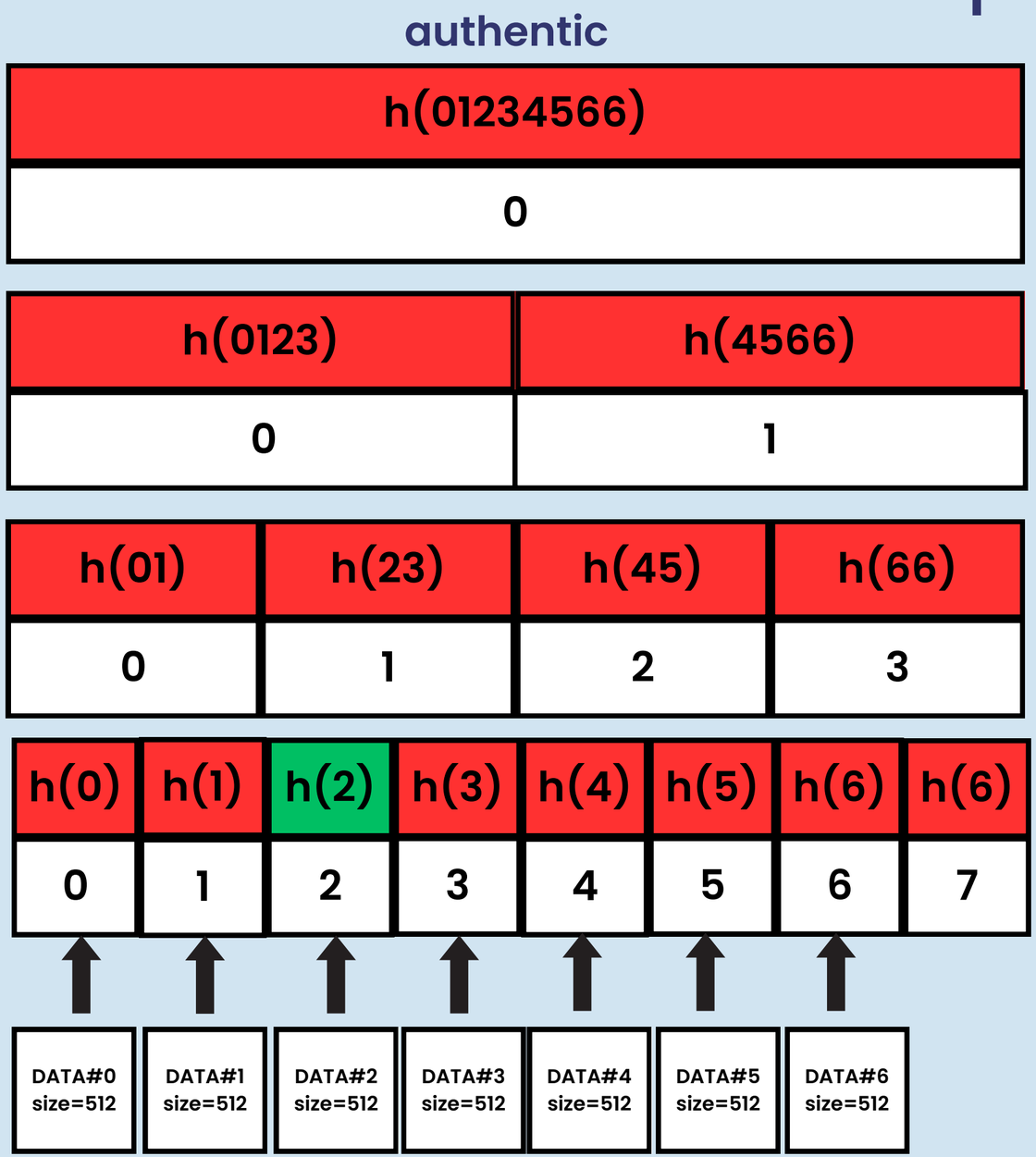
Trouver la donnée corrompu

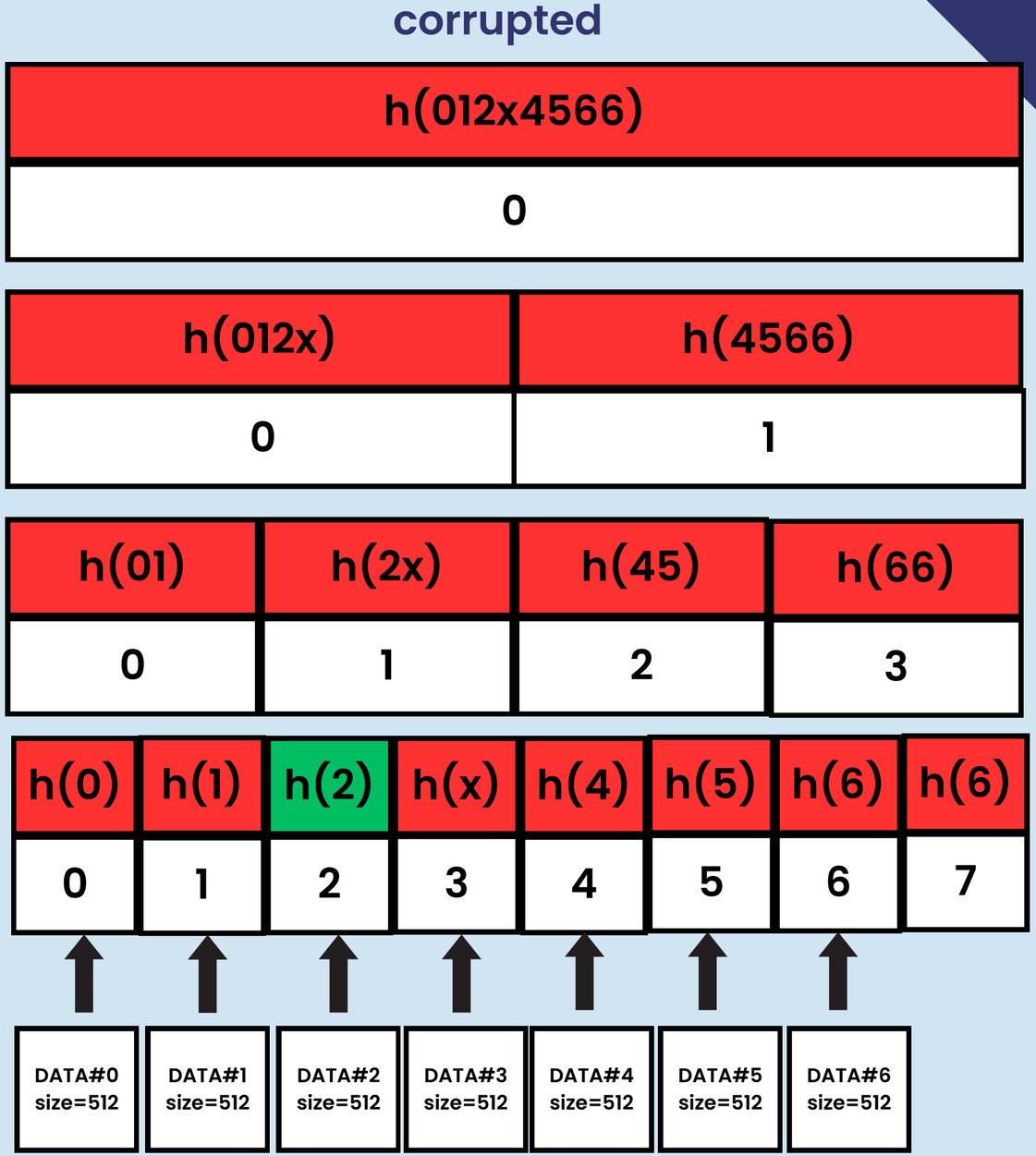


Trouver la donnée corrompu

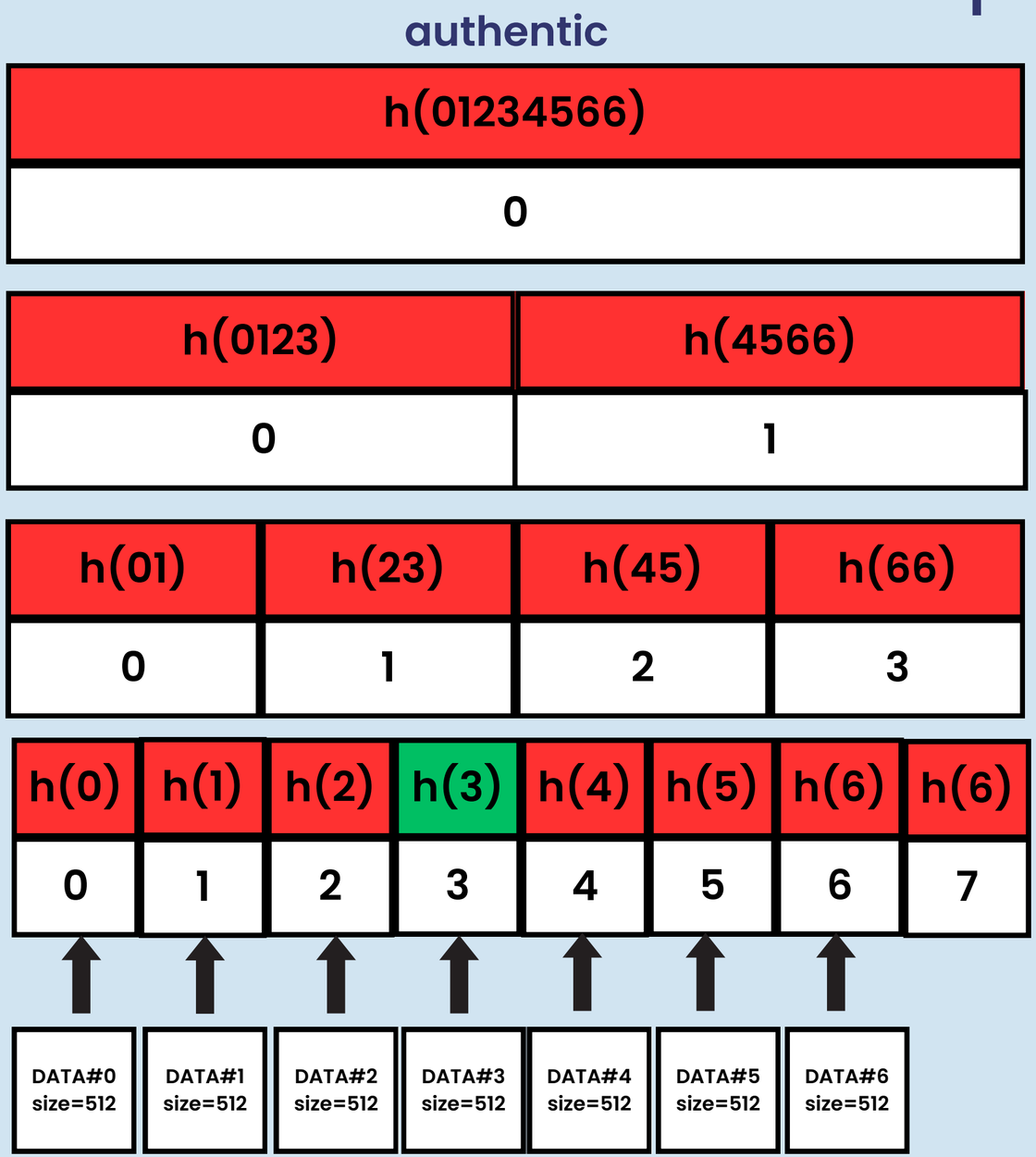


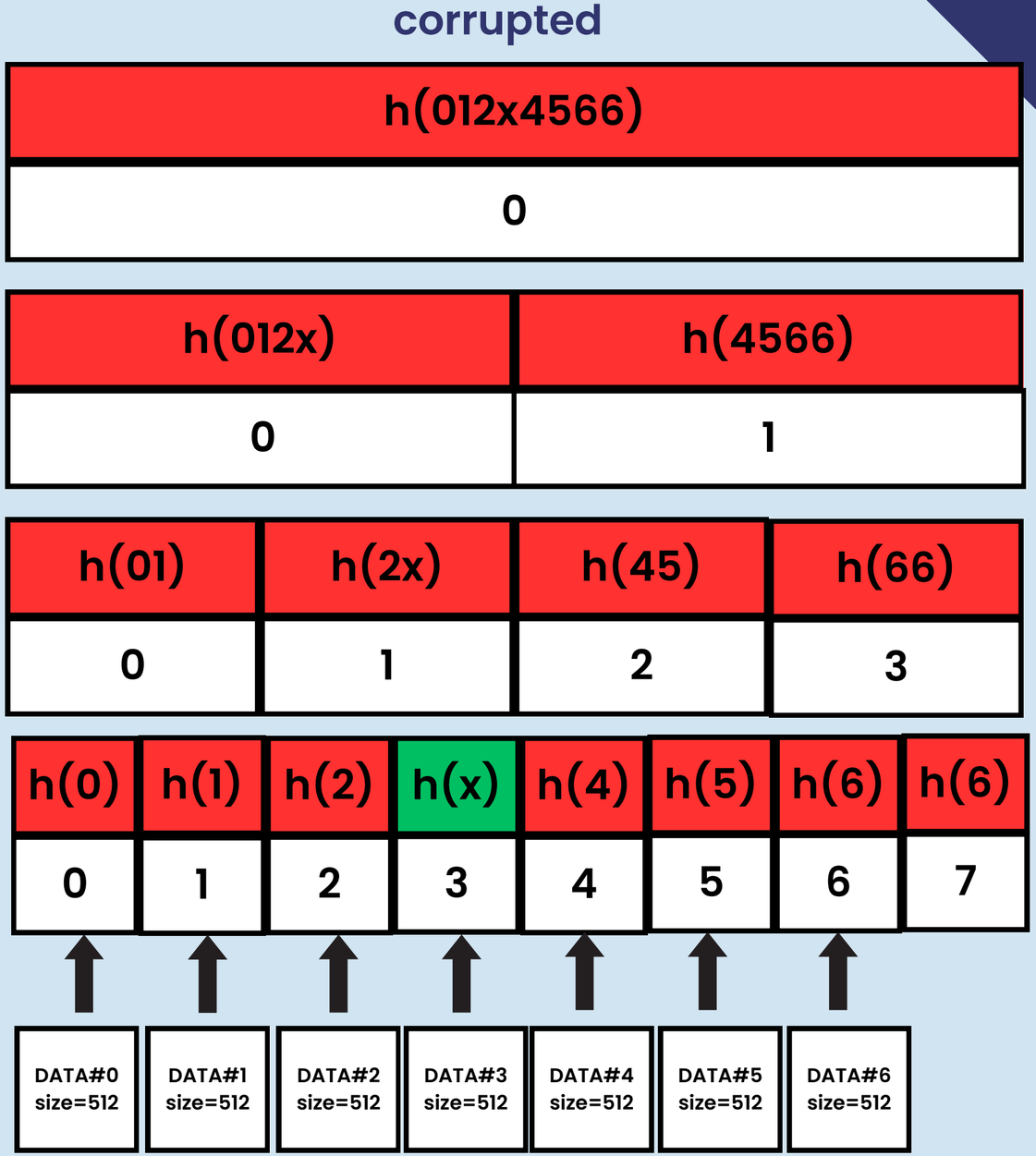
Trouver la donnée corrompu



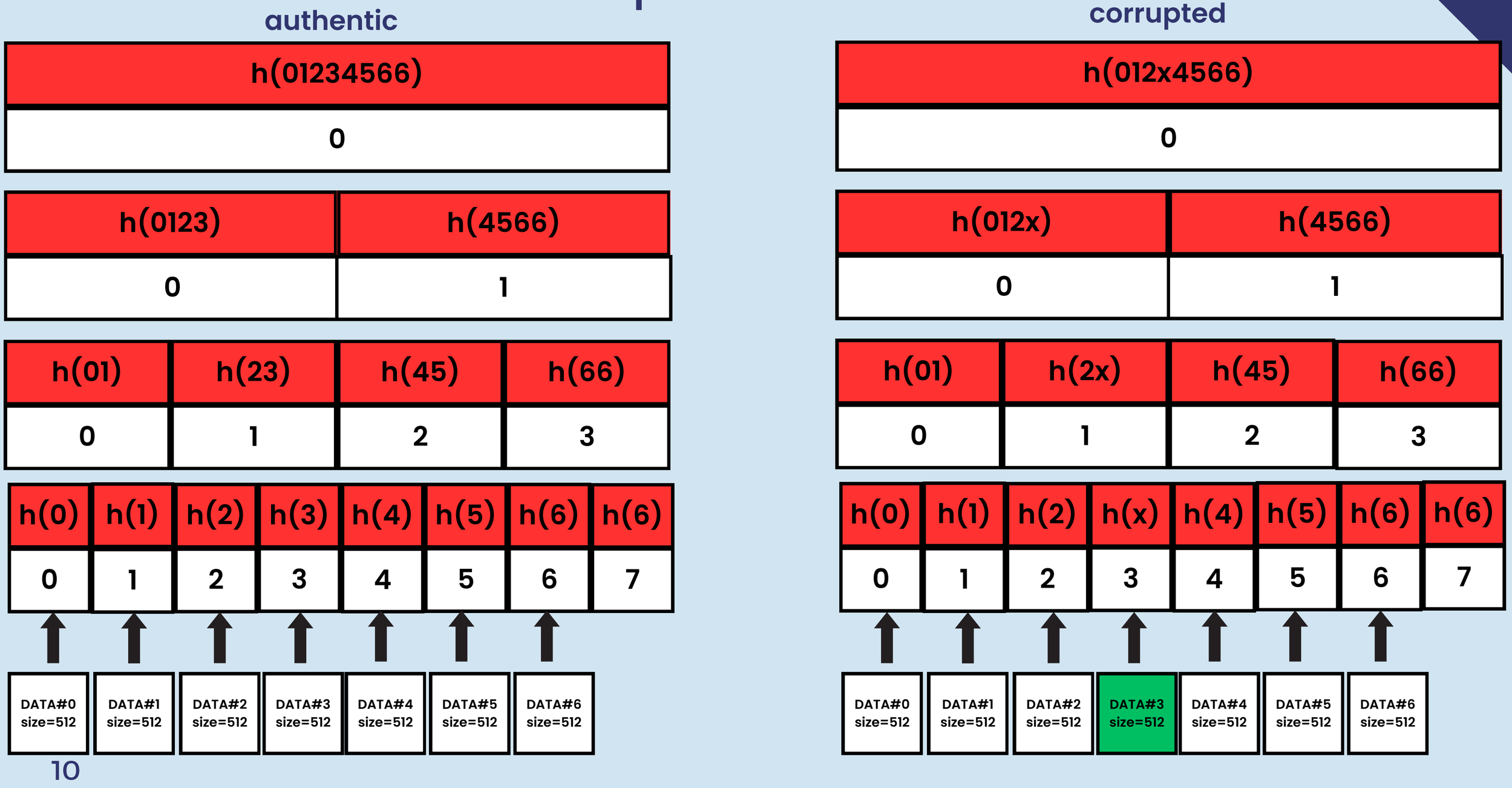


Trouver la donnée corrompu





Trouver la donnée corrompu



Merci pour votre attention

Des questions ?