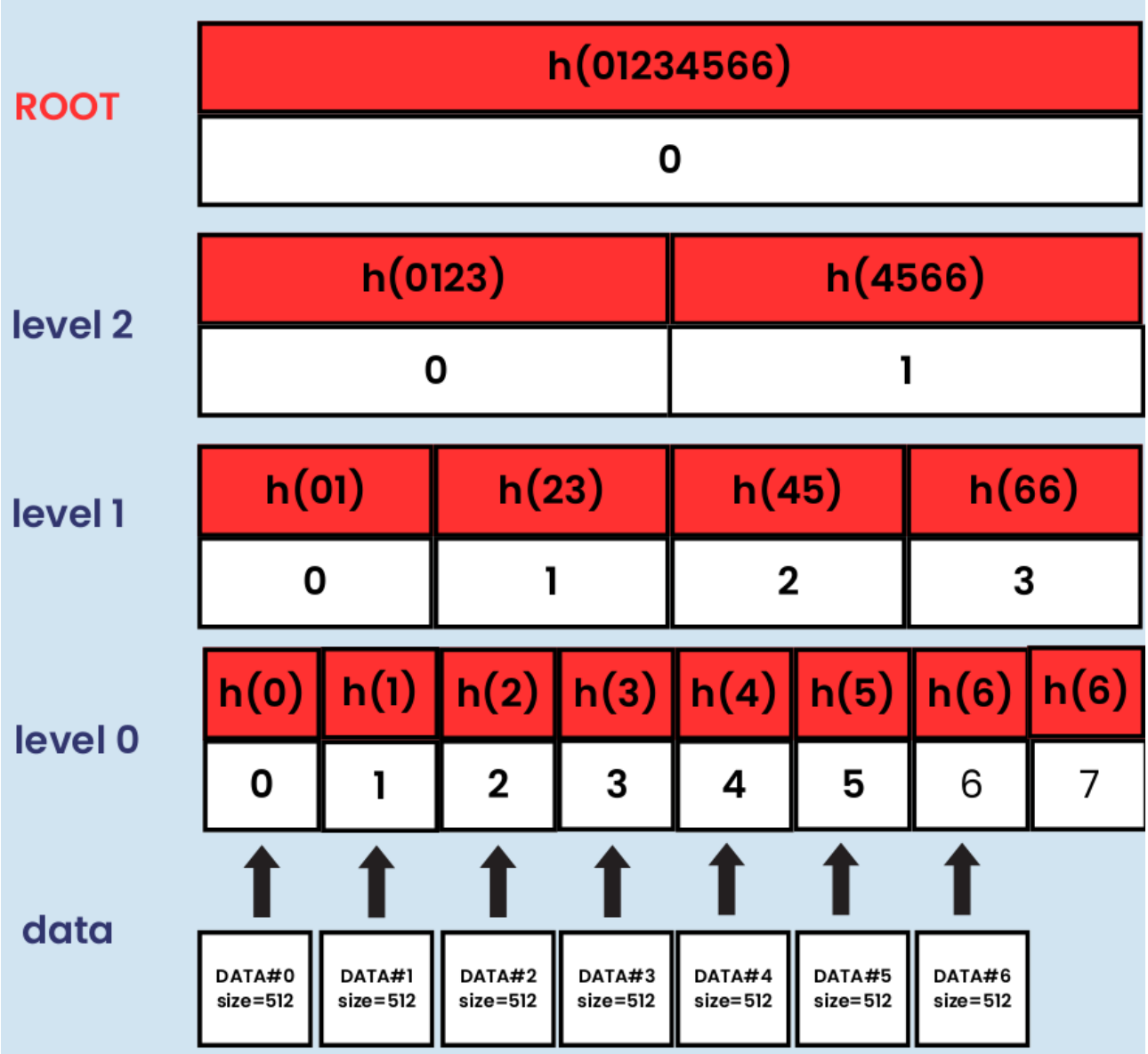
Introduction à la Cryptographie

BEN AMMAR Nader

Structure utilisé : Tableau 3D



Dans ce tableau 3D :

- Le premier indice représente le niveau de l'arbre, de la feuille (niveau le plus bas) à la racine (niveau le plus élevé).
- Le deuxième indice indique l'indice du nœud à ce niveau.
- Le troisième indice représente l'indice du caractère dans le haché du nœud.

Pour quelles raisons :

- **Efficacité en mémoire :** Un tableau occupe généralement moins d'espace en mémoire qu'un arbre binaire, car il n'y a pas de surcharge liée aux pointeurs nécessaires pour représenter les liens entre les nœuds.
- **Simplicité de mise en œuvre :** La manipulation d'un tableau est souvent plus simple et plus intuitive que celle d'un arbre binaire.

Contrôle d’intégrité global pour une partie :

Dans le contexte où l'on se concentre sur une seule partie des données, il reste possible de garantir l'intégrité globale de ces données en utilisant un processus efficace. Pour ce faire, nous exploitons une quantité relativement réduite d'informations supplémentaires, souvent désignées comme des empreintes « alternatives », qui remontent de la feuille à la racine de l'arbre de Merkle.

Voici une approche pour intégrer ces informations supplémentaires afin de vérifier l'intégrité globale pour chaque partie de DATA :

Tout d'abord, si nous avons accès à l'arbre de Merkle, nous pouvons calculer le hash racine (root hash) à partir de la partie de données que nous souhaitons vérifier, ainsi que des données supplémentaires.

on commence par level = 0,index = 0 :

1) Nous hachons la partie de données que nous voulons vérifier et stockons le résultat dans une variable appelée « current_hash ».
2) Ensuite, nous concaténons « current_hash » avec le hash du « sibling » du nœud actuel dans l'arbre de Merkle. La sélection du sibling dépend de la parité de l'index du nœud actuel : si l'index est pair, le sibling est l'indice suivant ; s'il est impair, c'est l'indice précédent.L'ordre de la concaténation est déterminé par la parité de l'index actuel : si l'index est pair, il est concaténé avant ; sinon, il est concaténé après.
3) Après avoir concaténé les deux hashes, nous calculons le hash de cette concaténation et stockons le résultat dans « current_hash ».
4) Nous répétons ce processus à partir de 2) en incrémentant le niveau (level) de l'arbre et en divisant l'index du nœud actuel par deux jusqu'à ce que nous atteignions la racine de l'arbre.

À la fin de ce processus, current_hash va contenir le merkle root hash de l’arbre.

Lorsque nous n'avons pas directement accès à l'arbre de Merkle, nous pouvons encore effectuer la vérification d'intégrité en récupérant sélectivement les empreintes alternatives nécessaires auprès d'un serveur ou d'une autre source. Cette approche est particulièrement utile lorsque l'arbre de Merkle est de grande taille en termes d'espace mémoire.

Voici comment procéder : on commence par level = 0,index = 0.

1) Si l'index du nœud actuel est pair, nous stockons l'indice suivant dans un tableau appelé « needed_index[level] » ; s'il est impair, nous stockons l'indice précédent. Nous répétons ce processus en incrémentant level et en divisant l'index du nœud actuel par deux jusqu'à ce que nous atteignions la racine de l'arbre.
2) Une fois que nous avons obtenu ce tableau, nous le transmettons au serveur ou à la source appropriée pour récupérer les empreintes correspondantes.
3) Avec les empreintes récupérées, triées par niveau (c'est-à-dire que tab[0] contient l'empreinte nécessaire pour le niveau 0, tab[1] l'empreinte nécessaire pour le niveau 1, etc.), nous pouvons ensuite effectuer la vérification d'intégrité en suivant un processus similaire à celui utilisé lorsque l'arbre de Merkle est accessible directement :
    - 3.1) Nous hachons la partie de données que nous voulons vérifier et stockons le résultat dans une variable appelée « current_hash ».
    - 3.2) Ensuite, nous concaténons « current_hash » avec l'empreinte correspondante dans le tableau « needed_hashes[level] ». L'ordre de la concaténation est déterminé par la parité de l'index actuel : si l'index est pair, il est concaténé avant ; sinon, il est concaténé après.
    - 3.3) Nous répétons ce processus à partir de 3.1) en incrémentant le niveau de l'arbre et en divisant l'index du nœud actuel par deux jusqu'à ce que nous atteignions la racine de l'arbre.