Fonctions de hachage et integrité

Sorina Ionica

March 3, 2025

- **Confidentialité** : l'information est accésible qu'à ceux dont l'accés est autorisé.
- **Authentification**: vérifier l'identité de la personne/machine lors de la communication
- **L'integrité**: les données ne subissent pas d'altération.

Fonctions de hachage

Une fonction de hachage calcule une empreinte de n bits pour un message **M** de taille arbitraire.

- Cas d'usage : integrité des messages, stockage des mots de passe, signatures
- Exemples: SHA-1 (160 bits), MD5 (128 bits), SHA-2...

Fonctions de hachage

Formellement:

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

- Les valeurs $H(m)$ sont uniformément réparties dans $\{0,1\}^n$.
- Pour se faire les constructions utilisent des transformations sur le message permettant d'obtenir la diffusion et la confusion.
- H doit pouvoir se calculer rapidement.

Application: Integrité des messages

Pour détecter si un fichier a été modifié il suffit de recalculer son empreinte.

```
// Fichier code.c

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv)
{
  if (argc <2)
    {
      ...
    }
}
```

$\Rightarrow$ Hash de 160 bits
SHA-1(code.c)=
0xA51F07BB62EC44A3F118

Les mots-de-passe

- Au lieu de stocker le mot-de-passe, on stocke son hash $h = H(password)$.
- Pour s'authentifier, l'utilisateur envoie son mot-de-passe *password*.
- Côté serveur, on vérifie si $H(password)$ est égal au h stocké dans la base.

Les mots-de-passe

- Au lieu de stocker le mot-de-passe, on stocke son hash $h = H(password)$.
- Pour s'authentifier, l'utilisateur envoie son mot-de-passe *password*.
- Côté serveur, on vérifie si $H(password)$ est égal au h stocké dans la base.

Sécurité

Si l'attaquant récupere la base de donées, les mots-de-passe sont "chiffrés" : il a les valeurs des hash, mais ne peut pas calculer les mots-de-passe.

Notions de sécurité

Une bonne fonction de hachage doit être difficile à inverser.

Résistance à la préimage

Étant donné x, trouver M t.q. $H(M) = x$.

Résistance aux collisions

Trouver M_1 et M_2 t.q. $H(M_1) = H(M_2)$.

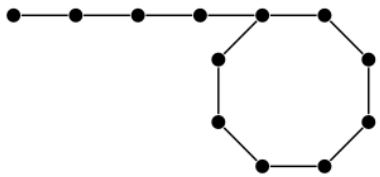
Algorithme rho de Pollard

- En entrée, un ensemble fini S.
- Une fonction $F : S \to S$ tel que :
  - $x_0 \in S$, $x_{i+1} = F(x_i)$
  - F est un générateur pseudo-aléatoire, c.à.d. les valeurs $F(m)$ sont reparties uniformement dans S.
- Idée: on sait qu'il existe i et j tel que

$$x_i \neq x_j \text{ mais } F(x_i) = F(x_j)$$

Algorithme rho de Pollard

m éléments hors du cycle, n la longueur du cycle



Méthode 1: Collisionement avec une mémoire "infinie"

Méthode 2: $\exists p$ tel que $x_p = x_{2p}$ avec $m \leq p \leq m + n$

Le paradoxe des anniversaires

Proposition

Si on choisit k balles parmi N balles de manière aléatoire et independante, alors la probabilité que deux soient identiques est $> \frac{1}{2}$ si $k \geq 1,17\sqrt{N}$.

Exemple : Si $N = 365$, alors la probabilité que parmi k personnes se trouvant dans une salle à un moment donné il y ait 2 avec le même jour d'anniversaire est $> \frac{1}{2}$ si $k > 23$.

Sécurité de SHA-1

En appliquant le paradoxe des anniversaires, quelle est la complexité d'un calcul de collision sur SHA-1?

Sécurité de SHA-1

En appliquant le paradoxe des anniversaires, quelle est la complexité d'un calcul de collision sur SHA-1?

Sécurité de SHA-1

En appliquant le paradoxe des anniversaires, quelle est la complexité d'un calcul de collision sur SHA-1?

Réponse: 2^{80} opérations.

Sécurité de SHA-1

En appliquant le paradoxe des anniversaires, quelle est la complexité d'un calcul de collision sur SHA-1?

Réponse: 2^{80} opérations.

En fait, depuis 2017, SHA-1 est morte : attaque en 2^{63} opérations.

`https://shattered.io`

Preuve de travail

Construction d'un point distingué d'une fonction cryptographique symbolisant la réalisation d'un effort

- Cet effort est mesuré par le temps de calcul et est correlé à un coût financier
- La preuve est verifiable rapidement
  - Hashcash "Bitcoin"
  - Contre-mesure au SPAM etc.

Preuve de travail

- Construction d'un point distingué d'une fonction cryptographique symbolisant la réalisation d'un effort
  - Qu'est qu'un point distingué? En entrée:
    - Une fonction de hachage cryptographique
    - Une donnée M.
  - M est un point distingué si $H(M)$ dispose de la distinction (i.e. une propriété choisie).

Qu'est qu'un point distingué?

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

- $H(M)$ doit être paire
- $H(M)$ doit débuter (ou terminer) par k bits à 0.
- $H(M)$ doit débuter par 0x12345 (Hexadécimal) etc.

Preuve de travail: algorithme

Algorithme

$M \leftarrow$ Random value in $\{0,1\}^*$
while $H(M)$ n'est pas distingué
$M \leftarrow F(M)$ $F(M)$ peut être $H(M)$, $M+1$ etc.

Quel effort ? Combien de tours de boucles pour calculer les distinctions suivantes?

- $H(M)$ doit être paire
- $H(M)$ doit débuter (ou terminer) par k bits à 0.
- $H(M)$ doit débuter par 0x12345 (Hexadécimal) etc.

Le consensus

Comment se mettre d'accord?
Comment avoir "confiance" dans cet accord?

Se mettre d'accord

- Centralisation de l'information
  - Tiers ou autorité centrale
  - Information partagée sur un serveur
  - Tout accès à une information centralisée sur un serveur est la même pour tous
- Confiance: Si le tiers de confiance fait autorité
  - Exemples:
    - Système d'identification français: l'autorité est l'état français.
    - Actes notariés: l'autorité est le notaire
    - Certification X.509: l'autorité est la racine de X.509 (par example Certigna)

Décentralisation de l'information

- Systèmes distribués (cf. cours Mr Cournier)
- Problèmes à résoudre
  - Comment un ensemble d'entités peuvent-elle se mettre d'accord localement sur une unique information ?
  - Quelles sont les conditions nécessaires et suffisantes à respecter pour que cela soit possible?
  - Comment avoir confiance dans une information locale sans autorité centrale?

Arbre de Merkle

- Vérification partielle de l'intégrité d'un ensemble de données
- Applications: Git, BitTorrent, Cryptomonnaies etc.
  - Les feuilles contiennent les données
  - Accès aux feuilles avec une complexité logarithmique
- En pratique ...
  - Arbre binaire

Arbre de Merkle

Merkle 1979

Exemple

Arbre de Merkle

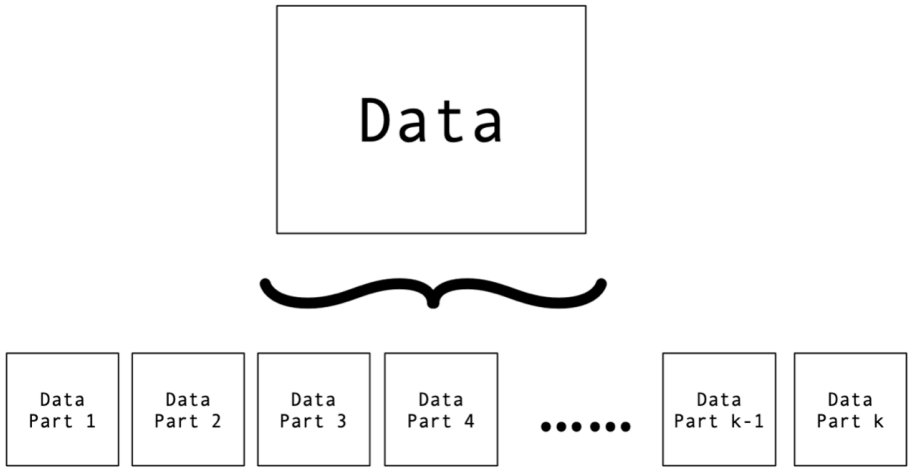
Merkle 1979

Exemple

Arbre de Merkle

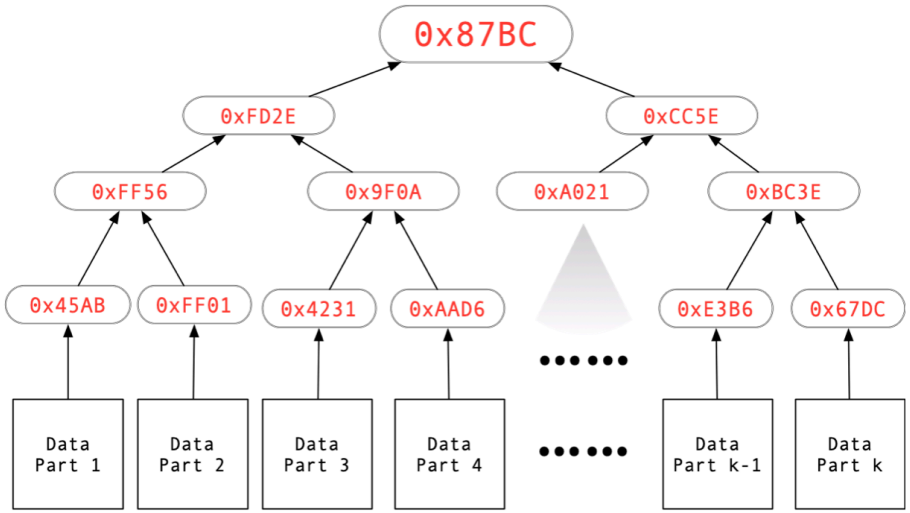
Merkle 1979

Exemple

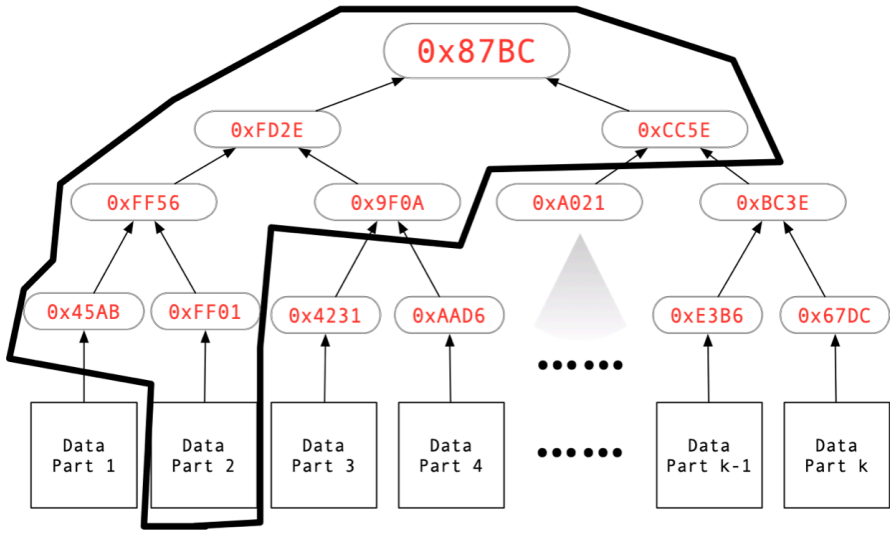


Arbre de Merkle

Calculer des valeurs de H recursivement et s'arrêter à la racine.



Arbre de Merkle



- But: garantir l'intégrité d'un bloc par rapport à l'ensemble des données
- Seul la racine doit être récupérée de manière sûre

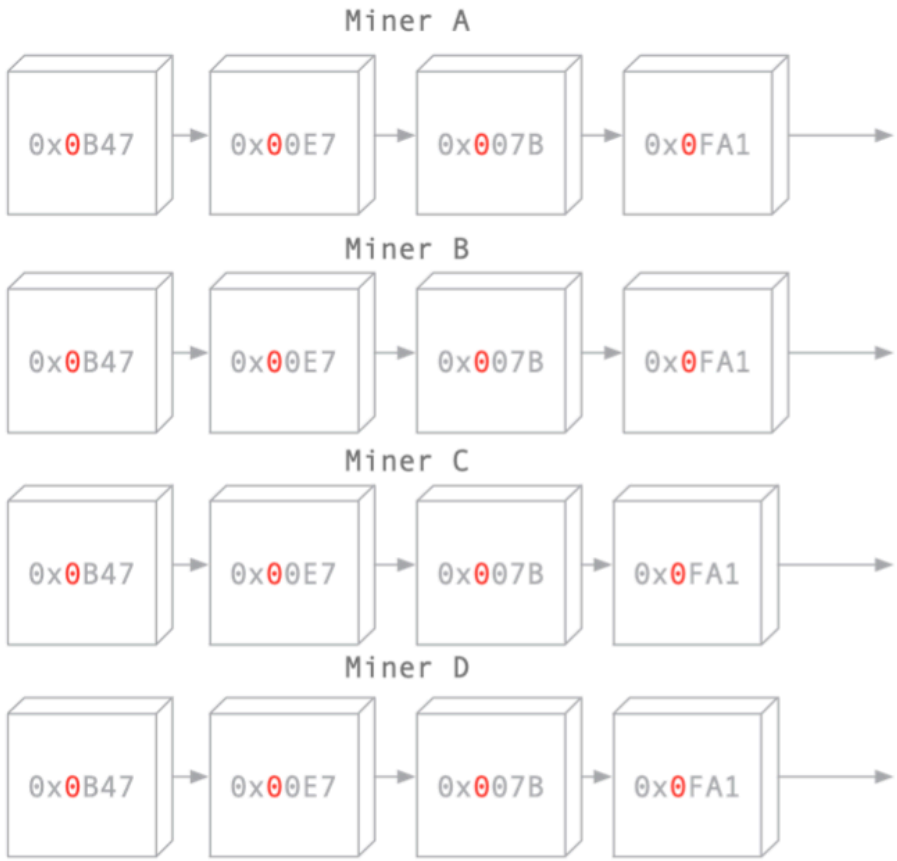
Bloc= donnée horodatée et hachée

Chaîne de bloc= le haché du bloc précédent + bloc+ preuve de travail

Exemple de consensus

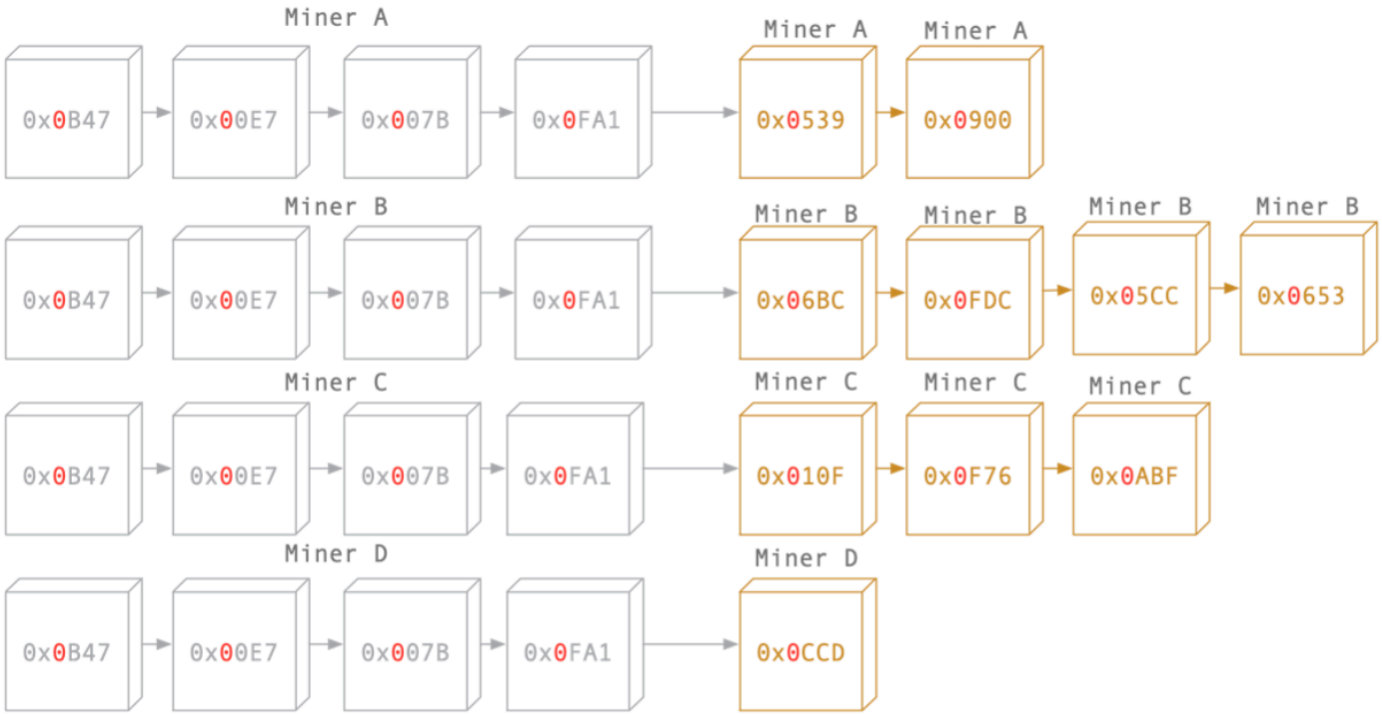
Chaîne de départ

Exemple de consensus



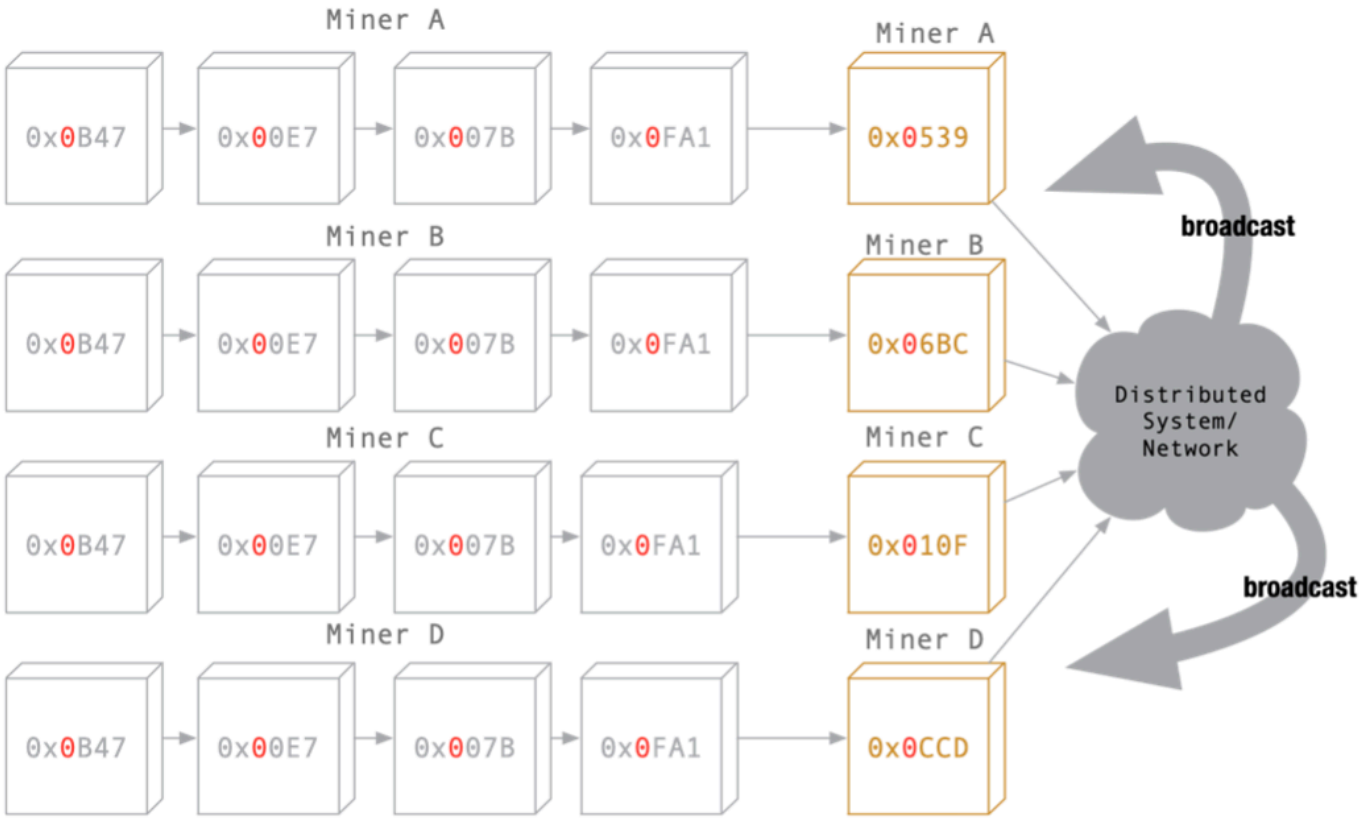
La chaîne de départ est commune à tout le monde.

Exemple de consensus



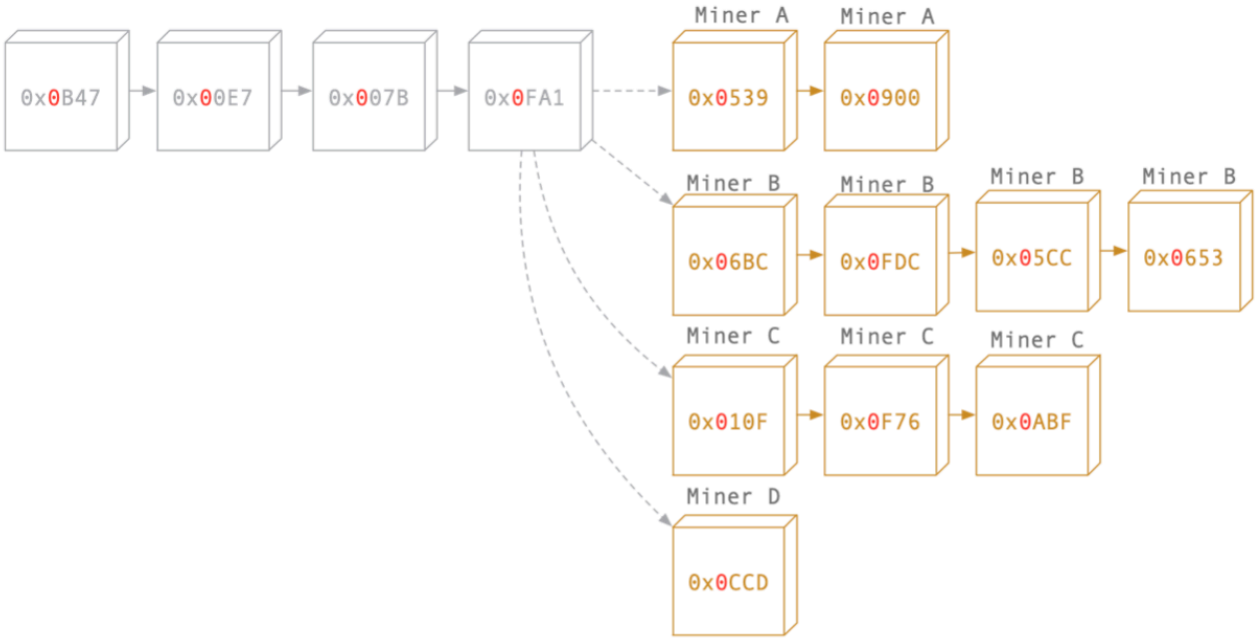
Chaque "mineur" calcule localement un prochain bloc possible.

Exemple de consensus



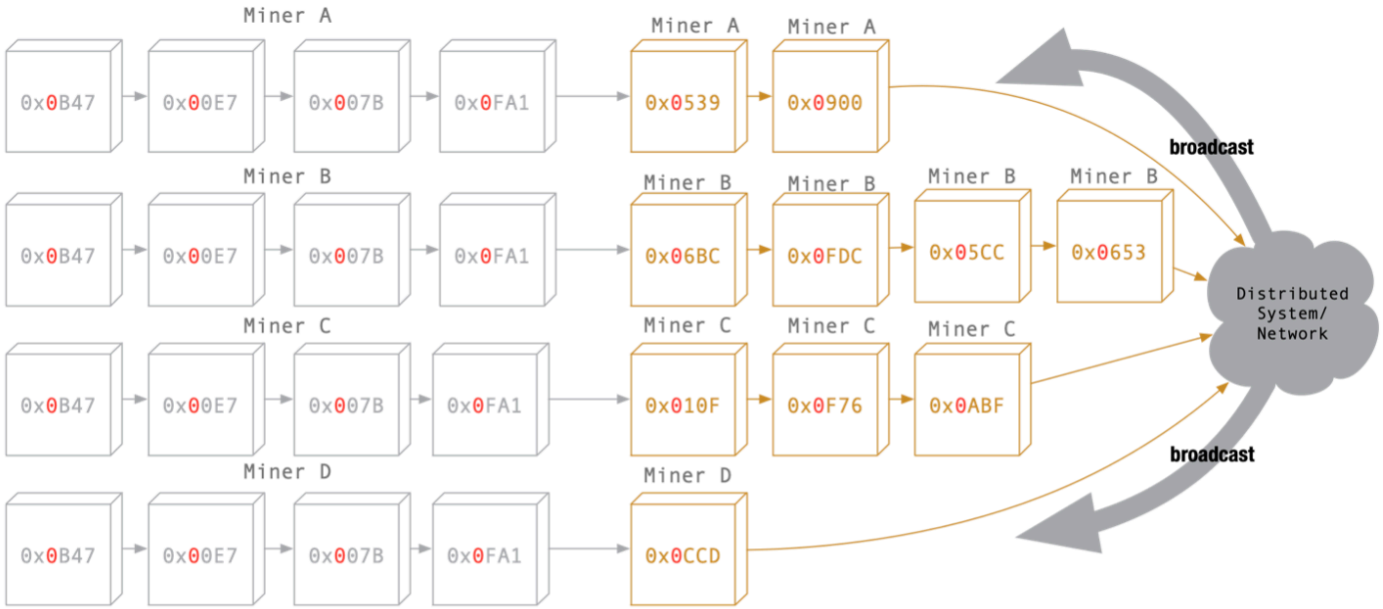
Chaque mineur diffuse sur le réseau à tous les mineurs (broadcast) les blocs calculés (cas particulier pséudo-synchrone).

Exemple de consensus



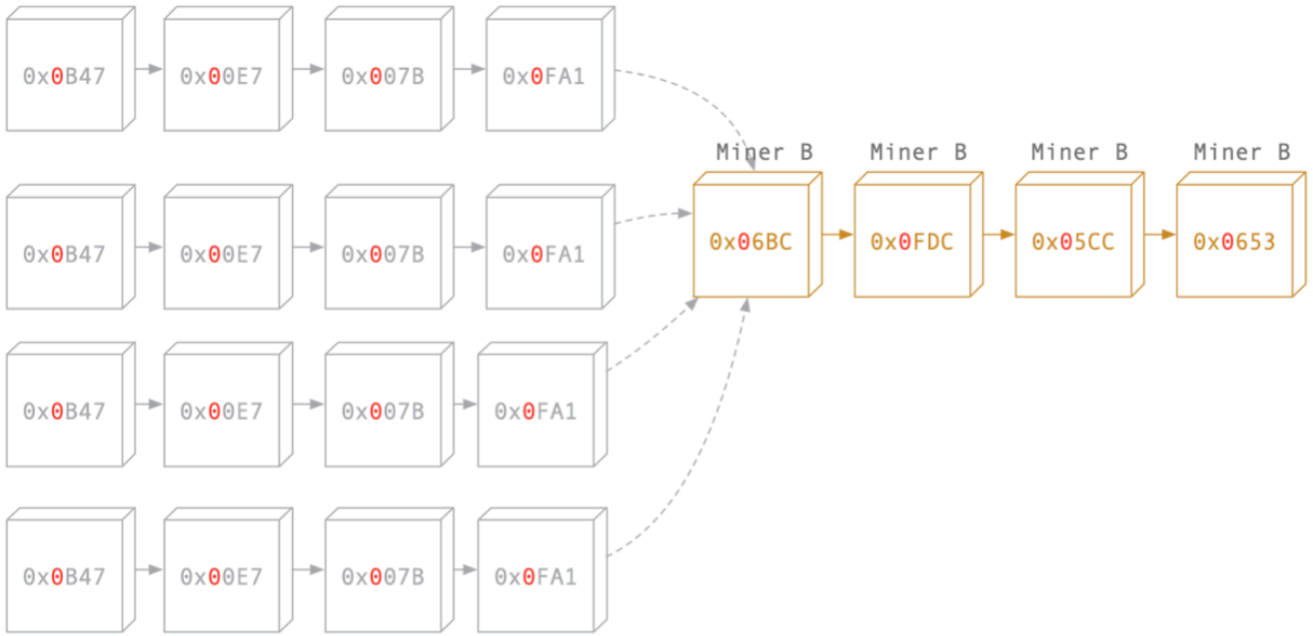
Cas plus réaliste où les mineurs ont fait progressé localement la chaîne en validant des blocs supplémentaires (cas asynchrone).

Exemple de consensus



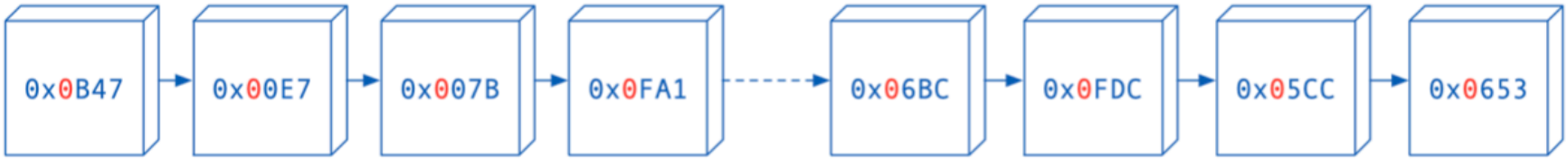
Chaque mineur reçoit plusieurs options de compléments possibles à sa chaîne de blocs locale provenant des autres trois mineurs.

Exemple de consensus



Chaque mineur applique la même règle : Je garde la chaîne la plus longue que je possède.
Cette chaîne représente le plus gros effort réalisé. Il est réalisé par le mineur B - gagnant !!

Exemple de consensus



Nouvelle chaîne commune à tous les mineurs. Le consensus est établi.