# Principes et Mécanismes des Protocoles de Consensus dans les Blockchains

Raphaël PADIOU

4 mars 2025

## Préambule

*Ce cours est rédigé sur la base des explications fournies par Matthieu Rambaud dans le cadre du projet Artishow. L'objectif principal de ce projet est d'implémenter une amélioration du protocole de consensus Diem21 pour le rendre résistant aux corruptions des clés de signature, avec pour objectif « Hotstuff résistant au key exposure ». Ce travail s'inscrit dans le cadre du cursus ingénieur à Télécom Paris.*

*Ce document parcourt les bases essentielles à la compréhension des protocoles de blockchain abordés dans notre projet. Il ne constitue pas un cours exhaustif sur les blockchains, bien qu'il puisse être suffisant en guise de découverte.*

# Table des matières

# 1 Définitions

## 1.1 Fonction de Hachage

**Définition 1.1** (Fonction de Hachage)**.** Une **fonction de hachage** est une fonction publique et déterministe qui associe à une entrée $m$ de taille quelconque une sortie $h$ de longueur fixe (généralement 128 ou 256 bits) :

$$H(m) \to h$$

Cette fonction doit satisfaire certaines propriétés de sécurité essentielles.

**Définition 1.2** (Propriétés Mathématiques d'une Fonction de Hachage)**.** Une **fonction de hachage** doit satisfaire les propriétés suivantes :

- **Déterminisme** : Pour une même entrée $m$, la sortie $h$ est toujours la même.
- **Rapidité** : La fonction doit être rapide à calculer pour toute entrée $m$.
- **Préimage Résistance** : Il doit être **computationally infeasible** de retrouver une entrée $m$ donnée une sortie $h$.
- **Seconde Préimage Résistance** : Il doit être **computationally infeasible** de trouver une autre entrée $m' \neq m$ telle que $H(m') = H(m)$.
- **Résistance aux Collisions** : Il doit être **pratiquement impossible** de trouver deux entrées distinctes $m_1 \neq m_2$ telles que $H(m_1) = H(m_2)$.

**Définition 1.3** (Schéma de Signature Numérique)**.** Un **schéma de signature numérique** est un ensemble de trois algorithmes : `KeyGen(aléa)` : génère une clé privée sk et une clé publique vk. L'algorithme utilise un tirage aléatoire uniforme pour produire ces clés. `Sign(sk, m)` : prend une clé privée sk et un message $m$ pour produire une signature $\sigma$. `Verify(sigma, m, vk)` : prend une signature $\sigma$, un message $m$, et une clé publique vk pour vérifier la validité de la signature.

### 1.1.1 Exemple de Processus de signature d'un message

**Alice montre à tout le monde sa $\mathbf{vk}_A$** : Alice peut publier sa clé publique $\text{vk}_A$ afin que toute personne puisse vérifier ses signatures. Lorsqu'Alice est ajoutée au système, sa clé publique est ajoutée au tableau commun accessible à tous les participants.

- **Signature d'un message** : Alice souhaite signer un message $m$ avec sa clé privée $\text{sk}_A$. Elle utilise l'algorithme de signature pour produire la signature $\sigma$ :

$$\sigma = \texttt{Sign}(\text{sk}_A, m)$$

où $\sigma$ est la signature numérique d'Alice pour le message $m$.

- **Vérification de la signature** : Toute personne, disons Bob, peut vérifier la signature en utilisant la clé publique $\mathrm{vk}_A$ d'Alice. Bob applique l'algorithme de vérification :

$$\texttt{Verify}(\sigma_A, m, \mathrm{vk}_A)$$

Cela renverra soit "accept" soit "reject".

**Définition 1.4** (Inforgeabilité)**.** Si l'algorithme de vérification retourne "accept", cela prouve que, si Alice est honnête, c'est bien elle qui a généré $\sigma_A$. En d'autres termes, personne ne peut forger une signature valide sans connaître la clé privée $\mathrm{sk}_A$, assurant ainsi l'inforgeabilité de la signature.

**Définition 1.5** (Message Authentication Code (MAC))**.** Un **Message Authentication Code (MAC)** est un code utilisé pour assurer l'authenticité d'un message entre deux parties. Contrairement aux signatures numériques, un MAC n'est pas opposable à un tiers. Cela signifie que seul le destinataire prévu (par exemple, Bob) peut vérifier l'authenticité du message, mais il ne peut pas prouver à un tiers (comme un juge) que le message provient de l'expéditeur prétendu. Le MAC permet à Bob de se convaincre lui-même de l'authenticité du message sans pouvoir convaincre d'autres personnes.

## 1.2 Utilisation d'un Message Authentication Code (MAC)

**Définition 1.6** (Clé Symétrique de Session)**.** Une **clé symétrique de session** ssk est une clé secrète partagée entre deux parties (par exemple, Alice et Bob) pour une session de communication sécurisée. Cette clé est utilisée pour générer et vérifier des MACs.

**Définition 1.7** (Canal de Communication Public)**.** Un **canal de communication public** est un moyen de communication accessible à tous, où les messages peuvent être interceptés ou modifiés par des tiers. Cependant, il est souvent utilisé pour échanger des informations publiques ou pour établir des clés secrètes à l'aide de protocoles cryptographiques comme Diffie-Hellman.

**Définition 1.8** (Sens d'un Canal)**.** Le **sens d'un canal** fait référence à la direction de la communication entre deux parties. Un canal peut être unidirectionnel (messages envoyés dans une seule direction) ou bidirectionnel (messages échangés dans les deux directions). La sécurité et l'intégrité des messages peuvent varier en fonction du sens du canal.

### 1.2.1 Exemple de Processus d'envoi de message

— **Génération de la clé** : Alice et Bob utilisent le protocole Diffie-Hellman pour générer une clé symétrique de session ssk :

$$\text{ssk} = \texttt{DiffieHellman}()$$

— **Envoi du message** : Alice souhaite envoyer un message $m$ à Bob. Elle utilise l'algorithme de génération de MAC pour produire un tag :

$$\text{tag} = \texttt{Mac.Gen}(\text{ssk}, m)$$

Elle envoie ensuite le couple $(m, \text{tag})$ à Bob.

— **Vérification du message** : Lorsque Bob reçoit le couple $(m, \text{tag})$, il utilise l'algorithme de vérification de MAC :

$$\texttt{Mac.Verify}(m, \text{tag}, \text{ssk})$$

Concrètement, cela teste si tag == $\texttt{Mac.Gen}(\text{ssk}, m)$.

— **Si oui** : Cela prouve à Bob que le tag a bien été généré sur $m$ par Alice ou par lui-même. Cela fonctionne aussi pour lui-même car la clé symétrique est partagée, donc Bob peut également générer le même tag pour $m$ en utilisant ssk.

— **Sinon** : Cela ne prouve rien d'autre que le fait que le tag est invalide.

## 1.3 Clé Privée et Clé Publique

**Définition 1.9** (Clé Privée)**.** La **clé privée** sk est un secret détenu uniquement par l'entité (par exemple Alice) qui l'a générée. Elle est utilisée pour signer des messages dans le cadre du schéma de signature numérique.

**Définition 1.10** (Clé Publique)**.** La **clé publique** vk est une clé dérivée de la clé privée et est distribuée à tous ceux qui souhaitent vérifier les signatures. Elle est utilisée par les autres participants pour vérifier qu'une signature a bien été générée par l'entité correspondante.

**Définition 1.11** (Clé Symétrique)**.** Une **clé symétrique** est une clé unique utilisée à la fois pour le chiffrement et le déchiffrement des messages entre deux parties. Cette clé doit rester secrète et est partagée entre les deux parties pour assurer la confidentialité et l'intégrité des communications.

**Définition 1.12** (Diffie-Hellman)**.** Le **protocole Diffie-Hellman** est une méthode permettant à deux parties de générer une clé secrète partagée en utilisant un canal de communication public. Ce protocole permet à Alice et Bob de créer une clé symétrique de session ssk sans que cette clé soit jamais transmise directement sur le réseau.

Cependant, pour que le protocole soit sécurisé, il est crucial qu'Alice et Bob connaissent les clés publiques l'un de l'autre avant d'échanger des informations. Si cette condition n'est pas remplie, ils s'exposent au risque d'attaque de type "man-in-the-middle", où un attaquant pourrait intercepter et manipuler les communications entre les deux parties.

**Définition 1.13** (Attaque de type "Man-in-the-Middle")**.** Une **attaque de type "Man-in-the-Middle"** (MITM) est une attaque dans laquelle un attaquant intercepte et, potentiellement, modifie les communications entre deux parties légitimes, Alice et Bob, sans qu'elles en aient connaissance. L'attaquant se place ainsi "entre" les deux parties et peut écouter, altérer ou injecter des messages dans les échanges. Cette attaque est particulièrement dangereuse si les deux parties échangent des informations sensibles (comme des clés de chiffrement ou des mots de passe) sans vérifier l'identité de l'autre, notamment en l'absence de mécanismes d'authentification adéquats.

**Définition 1.14** (Bloc dans une Blockchain)**.** Un **bloc** dans une blockchain est une structure de données qui contient un ensemble de transactions validées. Chaque bloc comprend :

- Un **hash** du bloc précédent, assurant ainsi la continuité de la chaîne.
- Un ensemble de **transactions** ou d'autres données.
- Un **timestamp** indiquant le moment où le bloc a été créé.
- Un **nonce** utilisé pour les mécanismes de preuve de travail.

Le hash d'un bloc inclut l'information contenue dans tous les blocs précédents, garantissant ainsi l'intégrité de la chaîne.

## 2 Comparaison entre Signature Numérique et MAC

| **Caractéristique** | **Signature Numérique** | **Message Authentication Code (MAC)** |
|---|---|---|
| Authenticité | Oui | Oui |
| Intégrité | Oui | Oui |
| Inforgeabilité | Oui, grâce à `Verify` | Non |

Table 1 – Comparaison entre Signature Numérique et MAC

### 2.1 Une garantie plus forte, appelée "non-répudiation"

La non-répudiation est une garantie plus forte que l'inforgeabilité. En effet, alors que l'inforgeabilité garantit simplement qu'une signature ne peut pas être falsifiée par un attaquant sans la clé privée correspondante, la non-répudiation va plus loin en assurant qu'une partie, comme Alice, ne pourra pas nier avoir signé un message de manière valide. Cette garantie est essentielle dans des contextes juridiques ou contractuels où une partie pourrait vouloir contester sa propre signature pour se dégager de responsabilités.

Avant 2020, le standard EdDSA ne garantissait pas cette propriété de non-répudiation de manière solide. Dans certains cas, il était théoriquement possible pour une personne malveillante, ou une Alice corrompue, de contester la signature d'un message devant un juge. En effet, si Alice pouvait prouver l'existence d'une autre clé publique, générée de manière détournée, qui validait la signature incriminée, elle pourrait faire valoir qu'elle n'était pas responsable de cette signature. Cette vulnérabilité a été un problème majeur dans la crédibilité des signatures numériques avant la révision du standard EdDSA.

**Signature Numérique** : Une signature numérique, lorsqu'elle est correctement vérifiée par l'algorithme de vérification `Verify`, constitue une preuve solide de l'origine du message. Plus précisément, si Alice signe un message $m$ avec sa clé privée $\mathrm{sk}_A$, et que cette signature $\sigma$ est ensuite vérifiée avec sa clé publique $\mathrm{vk}_A$, il devient possible pour toute partie (comme Bob) de prouver de manière irréfutable, y compris devant un juge, que la signature a bien été générée par Alice.

Cette vérification ne repose pas simplement sur l'absence de falsification, mais sur l'impossibilité, dans le contexte d'une signature numérique correctement mise en œuvre, que quelqu'un d'autre qu'Alice puisse revendiquer la signature sans posséder sa clé privée. En ce sens, la signature numérique assure la non-répudiation : Alice ne peut pas légitimement nier avoir signé le message, car la signature est vérifiable de manière publique et sécurisée.

**Message Authentication Code (MAC)** : En revanche, un MAC ne garantit pas la non-répudiation. Le MAC repose sur une clé secrète partagée entre les deux parties (Alice et Bob) et permet de vérifier l'intégrité et l'authenticité d'un message. Toutefois, cette vérification n'est pas opposable à un tiers, comme un juge. Si Bob présente à un juge un message $m$ et un MAC associé généré avec la clé symétrique ssk, il est impossible de prouver de manière irréfutable que c'est Alice qui a effectivement généré ce MAC. En effet, comme la clé ssk est partagée, n'importe qui possédant cette clé, y compris Bob lui-même, pourrait générer un MAC valide pour le message $m$ et prétendre qu'il provient d'Alice.

Ainsi, bien que les MACs assurent l'authenticité et l'intégrité des messages dans des contextes où les parties sont de confiance, ils ne fournissent pas de preuve opposable à un tiers. En revanche, les signatures numériques offrent une garantie de non-répudiation grâce à l'usage d'une clé privée unique, ce qui rend impossible pour une partie de renier sa signature sans contester l'intégrité du système dans son ensemble.

En résumé, les signatures numériques assurent une preuve vérifiable et opposable grâce à l'algorithme de vérification, garantissant la non-répudiation. Les MACs, eux, bien qu'utiles pour l'authentification et l'intégrité, ne peuvent fournir une telle preuve dans un contexte juridique.

## 2.2 Exemples d'Applications

**Signatures Numériques** :

— **TLS 1.3** : Dans le protocole TLS 1.3, les signatures numériques sont utilisées pour authentifier les parties lors de l'établissement d'une connexion sécurisée. Lors de la phase d'handshake, les parties (client et serveur) signent des messages pour prouver leur identité et empêcher les attaques de type "man-in-the-middle".

**Message Authentication Codes (MACs)** :

— **TLS 1.3** : TLS 1.3 utilise des MACs pour garantir l'intégrité des messages échangés entre le client et le serveur. À chaque étape du protocole, un MAC est ajouté pour s'assurer que les données n'ont pas été modifiées en transit et que la communication provient bien de la partie authentifiée.

## 2.3 Avantages des MACs par rapport aux Signatures Numériques

Dans certains cas, les MACs peuvent être préférables aux signatures numériques :

- **Performance** : Les MACs sont généralement plus rapides à générer et à vérifier que les signatures numériques, car ils utilisent des algorithmes de hachage symétriques plus simples.
- **Simplicité** : Les MACs sont plus simples à implémenter et à utiliser dans des environnements où la non-répudiation n'est pas nécessaire.
- **Efficacité** : Dans des systèmes où les communications sont limitées à deux parties de confiance, les MACs peuvent offrir une solution plus efficace pour assurer l'authenticité et l'intégrité des messages sans la complexité des signatures numériques.

# 3 Blockchain

*(Disclaimer : Lorsque Ethereum sera mentionné, cela fera en réalité référence à la v1 du PoS.)*

## 3.1 Fonctionnement d'une Blockchain

Une blockchain est une base de données distribuée et sécurisée par des mécanismes cryptographiques. Voici comment elle fonctionne :

- **Création de Blocs** : Les transactions sont regroupées en blocs. Chaque bloc contient un ensemble de transactions validées, un hash du bloc précédent, un timestamp, et un nonce.
- **Chaînage des Blocs** : Chaque bloc est lié au bloc précédent par son hash. Cela crée une chaîne de blocs, où chaque bloc dépend de tous les blocs précédents.
- **Validation des Blocs** : Les blocs sont validés par des nœuds du réseau à l'aide de mécanismes de consensus comme la preuve de travail (PoW) ou la preuve d'enjeu (PoS). Ces mécanismes assurent que seuls les blocs valides sont ajoutés à la chaîne.
- **Sécurité et Intégrité** : La sécurité de la blockchain repose sur les propriétés des fonctions de hachage et les mécanismes de consensus. Toute tentative de modification d'un bloc antérieur invaliderait tous les blocs suivants, rendant la manipulation facilement détectable.
- **Décentralisation** : La blockchain est maintenue par un réseau décentralisé de nœuds, chacun possédant une copie complète de la chaîne. Cela élimine le besoin d'une autorité centrale et augmente la résilience du système.

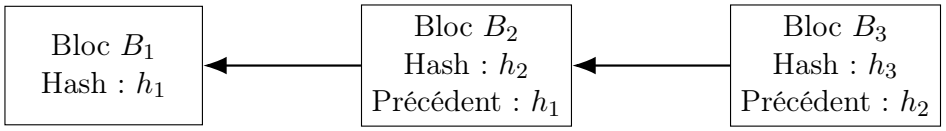


FIGURE 1 – Représentation d'une blockchain à 3 blocs.

En résumé, une blockchain est une chaîne de blocs sécurisée par des fonctions de hachage et des mécanismes de consensus, assurant l'intégrité et la sécurité des données de manière décentralisée.

## 3.2 Consensus

Le consensus est un mécanisme par lequel les nœuds d'un réseau blockchain s'accordent sur l'état actuel de la chaîne. Il est essentiel pour garantir que tous les participants du réseau ont la même version de la blockchain. Les mécanismes de consensus les plus courants sont la preuve de travail (PoW) et la preuve d'enjeu (PoS).

### 3.2.1 Preuve de Travail (PoW)

La preuve de travail est un mécanisme de consensus qui nécessite que les nœuds du réseau résolvent des problèmes mathématiques complexes pour valider les blocs. Voici comment cela fonctionne :

- **Calcul du Nonce** : Les mineurs (nœuds) doivent trouver un nonce (un nombre arbitraire) tel que le hash du bloc, combiné avec ce nonce, respecte une certaine condition (par exemple, le hash doit commencer par un certain nombre de zéros).
- **Validation** : Une fois le nonce trouvé, le bloc est diffusé au réseau. Les autres nœuds vérifient la validité du bloc en recalculant le hash avec le nonce proposé.
- **Récompense** : Le nonce valide reçoit une récompense sous forme de cryptomonnaie.

### 3.2.2 Preuve d'Enjeu (PoS)

La preuve d'enjeu est un mécanisme de consensus qui sélectionne les validateurs de blocs en fonction de la quantité de cryptomonnaie qu'ils possèdent et sont prêts à "staker" (mettre en jeu). Voici comment cela fonctionne :

- **Sélection des Validateurs** : Les validateurs sont choisis de manière pseudo-aléatoire en fonction de la quantité de cryptomonnaie qu'ils ont stakée. Plus un validateur stake de cryptomonnaie, plus il a de chances d'être sélectionné.
- **Validation des Blocs** : Les validateurs sélectionnés créent et valident les nouveaux blocs. Ils sont incités à agir honnêtement car ils risquent de perdre leur mise en cas de comportement malveillant.
- **Récompense** : Les validateurs reçoivent des récompenses sous forme de frais de transaction et/ou de nouvelles unités de cryptomonnaie.

### 3.2.3 Comparaison

La Proof of Work (PoW) assure la sécurité en exigeant une quantité substantielle de puissance de calcul, ce qui rend les attaques onéreuses et complexes à exécuter. En revanche, la Proof of Stake (PoS) se distingue par son efficacité énergétique, car elle ne repose pas sur une puissance de calcul intensive. De plus, la PoS favorise la participation honnête en alignant les intérêts des validateurs avec ceux du réseau, offrant ainsi une alternative plus durable et potentiellement plus sûre à long terme.

## 3.3 Principe de Leader et de Validateurs

Dans certains systèmes de blockchain, comme Ethereum, un leader est désigné pour proposer le prochain bloc à ajouter à la chaîne. Dans le cadre d'Ethereum, bien qu'il existe environ un million de validateurs éligibles, un sous-ensemble éphémère, appelé le *beacon committee*, est responsable de la proposition du bloc à chaque itération de vote. Ce comité, composé d'au moins 128 validateurs tirés au sort, participe à la validation des blocs proposés. Ainsi, le pourcentage total de pouvoir de vote est réparti entre les membres de ce comité, et le leader est sélectionné parmi eux pour proposer un bloc. Les autres nœuds du réseau, en dehors de ce comité, valident également le bloc proposé avant qu'il ne soit ajouté à la chaîne. Ce mécanisme permet de garantir la décentralisation tout en optimisant la participation au processus de validation.

Dans le cas de Diem, le réseau utilise un ensemble de validateurs pour proposer et valider les blocs. Tous les participants du réseau ne sont pas nécessairement des validateurs. Les validateurs sont des nœuds de confiance qui ont été sélectionnés pour leur fiabilité et leur capacité à maintenir l'intégrité du réseau. Les validateurs travaillent ensemble pour atteindre un consensus sur l'état de la blockchain.

## 3.4 Nœuds Corrompus et Hypothèse des Deux Tiers

La sécurité des blockchains repose sur l'hypothèse que deux tiers des nœuds sont honnêtes, permettant à un tiers d'être malveillants sans compromettre le réseau.

**Définition 3.1** (Nœud Malveillant)**.** Un **nœud malveillant** est un nœud du réseau qui tente de manipuler ou de corrompre la blockchain en ne respectant pas les règles du protocole. Ces nœuds peuvent essayer de valider des transactions invalides ou de créer des forks dans la chaîne.

**Définition 3.2** (Hypothèse des Deux Tiers)**.** L'**hypothèse des deux tiers** stipule qu'au moins plus de deux tiers des nœuds du réseau, pondérés par leur pouvoir de vote (c'est-à-dire la quantité de "stake" ou de fonds en dépôt), sont honnêtes et suivent les règles du protocole. Cette hypothèse est cruciale pour assurer que le consensus peut être atteint, même en présence de nœuds malveillants.

Cette hypothèse est fondamentale pour garantir que le consensus puisse être maintenu malgré la présence de nœuds malveillants. Par exemple, dans un système de preuve d'enjeu (PoS), où les validateurs sont choisis en fonction de la quantité de "stake" qu'ils détiennent, si moins d'un tiers des validateurs sont malveillants, ils ne pourront pas manipuler le consensus ni compromettre l'intégrité de la blockchain. Cependant, tant que plus de deux tiers des validateurs honnêtes, représentant une majorité pondérée par leur "stake", participent au processus de validation, le système reste sécurisé et fiable.

**Définition 3.3** (Fork)**.** Un **fork** est une situation où la blockchain se divise en deux chaînes concurrentes. Cela peut se produire lorsque des nœuds malveillants tentent de créer une version alternative de la chaîne en validant des blocs différents de ceux validés par les nœuds honnêtes.

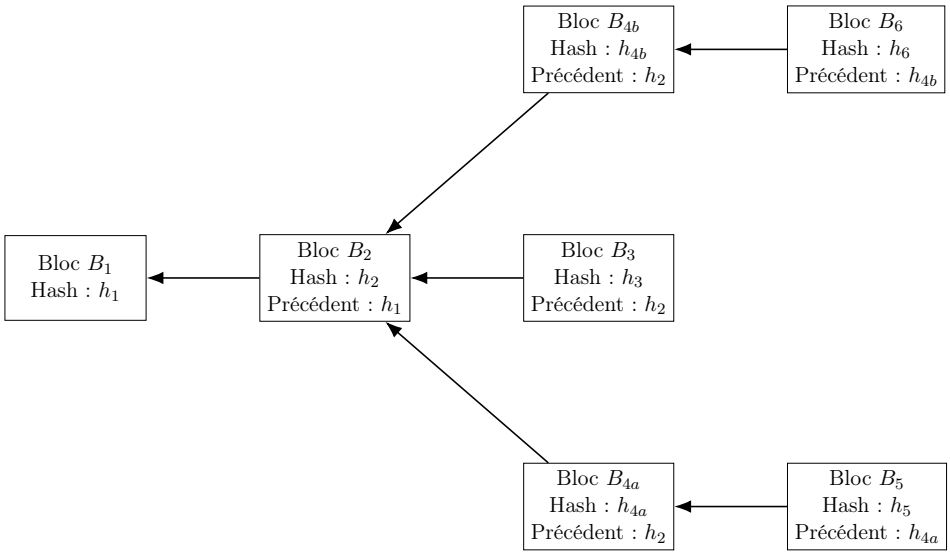


FIGURE 2 – Dans cet exemple, les blocs $B_{4a}$ et $B_{4b}$ pointent tous deux vers le bloc $B_2$, créant ainsi deux chaînes distinctes. Cette situation est non désirable car elle crée une incertitude sur la version correcte de la blockchain, ce qui peut entraîner des conflits et des pertes de données.

## 3.5 Clés Publiques, Type de Preuve et Sélection du Leader dans Ethereum

### 3.5.1 Clés Publiques

Dans Ethereum, chaque utilisateur possède une paire de clés cryptographiques : une clé privée et une clé publique. La clé privée est utilisée pour signer les transactions, tandis que la clé publique est utilisée par les autres participants pour vérifier ces signatures.

La clé publique est dérivée de la clé privée à l'aide d'algorithmes cryptographiques asymétriques, tels que l'algorithme de signature numérique à courbe elliptique (ECDSA).

- **Clé Privée** : La clé privée est un secret connu uniquement de l'utilisateur. Elle est utilisée pour signer les transactions et prouver l'identité de l'utilisateur.
- **Clé Publique** : La clé publique est dérivée de la clé privée et est partagée avec tous les participants du réseau. Elle est utilisée pour vérifier les signatures des transactions.

### 3.5.2 Type de Preuve Utilisé

Ethereum utilise deux types de mécanismes de consensus : la preuve de travail (PoW) et la preuve d'enjeu (PoS). Initialement, Ethereum utilisait uniquement la preuve de travail, mais il est en train de passer à la preuve d'enjeu avec la mise à jour Ethereum 2.0.

- **Preuve de Travail (PoW)** : Dans ce mécanisme, les mineurs résolvent des problèmes mathématiques complexes pour valider les blocs. Le premier mineur à résoudre le problème ajoute le bloc à la chaîne et reçoit une récompense.
- **Preuve d'Enjeu (PoS)** : Dans ce mécanisme, les validateurs sont sélectionnés pour proposer et valider les blocs en fonction de la quantité de cryptomonnaie qu'ils possèdent et sont prêts à "staker". Les validateurs sont incités à agir honnêtement car ils risquent de perdre leur mise en cas de comportement malveillant.

### 3.5.3 Mécanisme de Sélection du Leader

Dans Ethereum, le leader (ou proposeur de bloc) est sélectionné de manière aléatoire ou en fonction de critères spécifiques, selon le mécanisme de consensus utilisé.

- **Preuve de Travail (PoW)** : Le leader est le mineur qui résout le problème mathématique en premier. Ce processus est aléatoire et dépend de la puissance de calcul du mineur.
- **Preuve d'Enjeu (PoS)** : Le leader est sélectionné de manière pseudo-aléatoire en fonction de la quantité de cryptomonnaie que le validateur a stakée. Plus un validateur stake de cryptomonnaie, plus il a de chances d'être sélectionné comme leader.

# 4 Certificats et Messages dans les Protocoles de Consensus

## 4.1 Définitions

**Définition 4.1** (Bloc Courant)**.** Un **bloc courant** est le bloc actuellement considéré comme le plus récent et valide par un nœud dans le réseau. Il est utilisé comme référence pour ajouter de nouveaux blocs à la chaîne.

**Définition 4.2** (StatusMessage)**.** Un **StatusMessage** est un message envoyé par un nœud pour indiquer son état actuel dans le protocole de consensus. Il contient des informations telles que le numéro de vue, le hash du bloc courant, et d'autres métadonnées nécessaires pour synchroniser les nœuds du réseau.

**Définition 4.3** (Proposition de Bloc)**.** Une **proposition de bloc** est un message envoyé par un leader ou un nœud proposant un nouveau bloc à ajouter à la blockchain. Ce message contient le bloc proposé, le hash du bloc précédent, et d'autres informations nécessaires pour que les nœuds du réseau puissent valider et accepter la proposition.

**Définition 4.4** (Quorum Certificate)**.** Un **Quorum Certificate (QC)** est une preuve cryptographique qu'un certain nombre de nœuds (généralement une majorité qualifiée) ont validé un bloc ou une étape spécifique dans le protocole de consensus. Le QC est utilisé pour garantir que les décisions prises par le réseau sont sécurisées et acceptées par une majorité des participants.

**Définition 4.5** (Decision Certificate)**.** Un **Decision Certificate (DC)** est une preuve cryptographique qu'un bloc a été définitivement validé par le réseau. Il est obtenu lorsque suffisamment de nœuds ont accepté le bloc. Par exemple, un client peut avoir une entrée $b$ (le bloc) et une entrée $\pi$ (le Decision Certificate). La fonction suivante est utilisée pour vérifier si $b$ est un decided block :

$$\texttt{IsValid}(b, \pi) \rightarrow \text{accept/reject}$$

Si la fonction renvoie "accept", cela signifie que $b$ est un decided block et peut être inclus de manière permanente dans la blockchain.

**Définition 4.6** (Decided Block)**.** Un **Decided Block** est un bloc qui a été validé et accepté par le réseau, accompagné d'un Decision Certificate. Ce bloc est considéré comme final et ne peut plus être modifié ou annulé. Il est ajouté de manière permanente à la blockchain.

## 4.2 Propriété des Decided Blocks

Pour deux **Decided Blocks**, l'un doit nécessairement être un préfixe de l'autre. Cela garantit la cohérence et l'unicité de la chaîne de blocs.

## 4.3 Contre-exemple d'une Situation Non Désirable

Dans l'exemple précédent (3.4), les blocs $B_{4a}$ et $B_{4b}$ pointent tous deux vers le bloc $B_2$, créant ainsi deux chaînes distinctes. Cette situation est non désirable car elle crée une incertitude sur la version correcte de la blockchain, ce qui peut entraîner des conflits et des pertes de données.

## 4.4 Modèle, et sources de vulnérabilité

Un nœud sincère peut se tromper sur les blocs les plus récents (proposé, quorum-certified et decided) pour plusieurs raisons, dans un réseau où l'adversaire coordonne les nœuds corrompus et manipule les délais d'acheminement des messages :

- **Propagation des Messages** : La propagation des messages dans le réseau n'est pas instantanée, et un nœud sincère peut ne pas avoir reçu tous les messages pertinents concernant les derniers blocs (proposé, quorum-certified, et decided). En conséquence, il peut avoir une vision incomplète de l'état du réseau.
- **Latence Réseau** : La latence du réseau peut provoquer des retards dans la réception des messages, ce qui amène parfois un nœud sincère à considérer un bloc obsolète comme étant le plus récent. Cette situation peut se produire si le nœud n'a pas encore reçu une mise à jour plus récente des autres nœuds du réseau.
- **Forks Temporaires** : Des forks temporaires peuvent survenir lorsque plusieurs nœuds proposent simultanément des blocs différents. Un nœud sincère pourrait alors suivre une branche qui sera éventuellement abandonnée par le réseau. Ces forks, bien que temporaires, peuvent induire une confusion sur l'ordre correct des blocs.
- **Manipulation des Délais d'Acheminement** : L'adversaire peut manipuler les délais d'acheminement des messages entre les nœuds. En introduisant des délais artificiels, l'attaquant peut forcer certains nœuds à recevoir des informations dans un ordre incorrect ou à être retardés dans la réception des blocs les plus récents. Cela peut tromper un nœud sincère en lui faisant croire qu'un bloc obsolète est toujours valide.
- **Coordination des Nœuds Corrompus** : L'adversaire peut coordonner les nœuds corrompus pour envoyer des informations incorrectes et induire en erreur les nœuds sincères sur l'état des blocs les plus récents. Cela peut entraîner des divergences dans la perception du bloc valide ou des blocs abandonnés.

Ces erreurs peuvent être corrigées lorsque le nœud sincère reçoit les informations correctes des autres nœuds, ce qui lui permet de mettre à jour son état pour refléter les blocs les plus récents (proposé, quorum-certified et decided) du réseau.

### Préliminaire : Itérations de Vote dans Ethereum et Diem

Dans Ethereum et Diem, le processus de consensus se fait par itérations de vote, appelées *rounds*. Dans Ethereum, chaque itération, ou *epoch*, se déroule tous les 32 blocs, mais seuls les blocs numérotés $32 \times k$ (les *checkpoint blocks*) sont soumis au vote. Ces blocs servent de points de synchronisation pour le réseau.

Par la suite, nous simplifierons en considérant uniquement ces *checkpoint blocks* pour les processus de décision.

## 4.5 Decision Certificate dans Ethereum et Diem

**Ethereum (Proof of Stake).** Dans Ethereum, un bloc $b_r$ est considéré comme définitivement validé lorsqu'un ensemble de **plus de 2/3 des validateurs, pondérés par leur pouvoir de vote (c'est-à-dire leur "stake" en dépôt)** signe un message $\langle \text{vote}, b_{r-1} \rangle$. L'intégrité du pointeur vers le bloc précédent $b_{r-1}$ est assurée par le **Quorum Certificate**, qui comprend :

- Le **hash** du bloc précédent $H(b_{r-1})$.
- Un ensemble de **signatures** $\sigma_1, \ldots, \sigma_u$ représentant plus de **2/3 des validateurs, pondérés par leur pouvoir de vote**, ayant validé $b_{r-1}$.

**Diem (LibraBFT).** Diem repose sur un consensus **par rounds**, où un **leader** propose un bloc $b_r$ et les autres validateurs votent. Un bloc est validé lorsqu'il reçoit l'approbation de plus de **2/3 des validateurs, pondérés par leur pouvoir de vote**, via un mécanisme équivalent à Ethereum :

- Un **Quorum Certificate**, reliant $b_r$ à $b_{r-1}$, formé par un ensemble de signatures.
- Un processus de validation garantissant l'acceptation d'un bloc uniquement si le **précédent a bien été confirmé**.

Cette hypothèse est essentielle pour garantir que le consensus peut être atteint même en présence de nœuds malveillants. Par exemple, dans un système de preuve d'enjeu (PoS), si plus d'un tiers des validateurs sont malveillants, ils pourraient potentiellement manipuler le consensus et compromettre l'intégrité de la blockchain. Cependant, tant que la majorité des validateurs, pondérés par leur pouvoir de vote, sont honnêtes, le système peut résister aux attaques et maintenir la sécurité et l'intégrité des données. On précisera que **Ethereum** et **Diem** ont la même définition d'un **Decision Certificate**.

**Définition 4.7** (Decision Certificate, contexte particulier.)**.** Dans Ethereum, Diem et Moonshot, on dénifit un **Decision Certificate** pour un bloc $b_r$ du round $r$, noté $D_r(b_r)$, comme une structure de données composée des Quorum Certificates (QC) pour les blocs $b_r$ et $b_{r+1}$, où $b_{r+1}$ est le fils direct de $b_r$ et a été proposé et voté dans le round $r+1$ qui suit immédiatement $r$.

Formellement, un Decision Certificate est défini comme suit :

$$D_r(b_r) = (QC(b_r), QC(b_{r+1}))$$

où : - $QC(b_r)$ est le Quorum Certificate pour le bloc $b_r$, prouvant que plus de 2/3 des validateurs ont accepté ce bloc. - $QC(b_{r+1})$ est le Quorum Certificate pour le bloc $b_{r+1}$, prouvant que plus de 2/3 des validateurs ont accepté ce bloc, et qu'il a été proposé et validé dans le round $r + 1$.

Note : Dans Ethereum, un bloc avec un Decision Certificate est appelé un bloc "finalized".

## 4.6 Schéma illustratif des relations entre les blocs

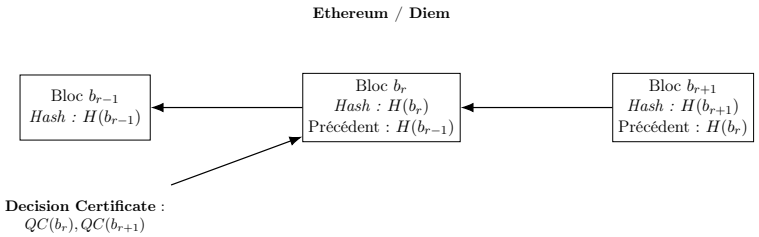


FIGURE 3 – Certificat de consensus : relation entre blocs avec Decision Certificate

Dans ce schéma :

- Chaque bloc $b_r$ **pointe vers son prédécesseur** $b_{r-1}$ grâce à un **Quorum Certificate (QC)** pour $b_r$, et vers son successeur $b_{r+1}$ avec un **Decision Certificate** qui inclut les QC pour $b_r$ et $b_{r+1}$.
- Le **Decision Certificate** assure que le bloc $b_r$ et son successeur immédiat $b_{r+1}$ ont été validés par un quorum de validateurs :
  - $QC(b_r)$ prouve que plus de 2/3 des validateurs ont accepté $b_r$.
  - $QC(b_{r+1})$ prouve que $b_{r+1}$ a été proposé et validé dans le round suivant $r + 1$, et que plus de 2/3 des validateurs l'ont accepté.
- La principale différence entre Ethereum et Diem réside dans leur **mécanisme de consensus** :
  - Ethereum repose sur un consensus *Proof of Stake*.
  - Diem utilise un consensus *par rounds avec un leader*.

## 4.7 Validation des Blocs dans Ethereum (Proof of Stake)

**Rappel :** Dans Ethereum, un bloc $b_r$ est proposé par un leader au round $r$. Pour qu'un nœud $i$ accepte de voter pour $b_r$, certaines conditions doivent être remplies.

## 4.8 Conditions Générales de Vote

Un nœud $i$ peut voter pour un bloc $b_r$ seulement si la proposition est accompagnée d'un ensemble $Fr$, défini comme suit :

$$Fr = \{\text{concaténation des StatusMsg signés } (\tfrac{2}{3} \text{ du pouvoir de vote pondéré par le "stake"})\}$$

Cela sous-entend que chaque nœud apporte une portion de vote proportionnelle à son "stake" (montant d'argent en dépôt). De plus, aucun des StatusMsg inclus dans cet ensemble ne doit déclarer avoir vu un bloc plus récent que $b_{r-1}$.

Ainsi, si $Fr$ est valide, le certificat $Cr_1$ peut être proposé, car il représente le bloc considéré comme le plus récent, validé par le consensus pondéré du réseau.

## 4.9 Conditions Spécifiques aux Protocoles

### 4.9.1 Conditions de Vote dans Ethereum

Soit $QC(b_{r_i})$ le QC pour le bloc le plus élevé (=le $r_i$ le plus grand) que $P_i$ a observé. Alors $P_i$ vote pour $b_r$ seulement si :

$$QC(b_{r'}) \leftarrow b_r \quad \text{avec} \quad r' \geq r_i$$

### 4.9.2 Conditions de Vote dans Diem/Moonshot

Soit $S$ l'ensemble des $2t+1$ StatusMsg reçus, où chaque $\text{StatusMsg}_i$ est de la forme :

$$\{\text{QC}(b_{r_j}), \text{déclaration signée de } P_j : \text{"pas vu de QC pour un bloc avec un round supérieur à } r_j\text{"}\}$$

Alors $P_i$ vote pour $b_r$ seulement si :

$$QC(b_{r'}) \leftarrow b_r \quad \text{avec} \quad r' \geq r_{\max} = \max_S(r_j)$$

Dans les deux protocoles, $r' \leq r-1$, ce qui signifie que le Quorum Certificate pour un bloc $b_r$ peut être associé à un bloc précédent $b_{r'}$ qui n'est pas nécessairement le bloc immédiatement précédent $b_{r-1}$. Cela permet une flexibilité dans la validation des blocs tout en assurant la sécurité du consensus.

## 4.10 Pourquoi $2/3$ ne suffit pas et pourquoi $> 2/3$ est nécessaire ?

Dans un protocole de consensus basé sur des validateurs, comme celui utilisé par Diem ou Ethereum, il est crucial d'assurer que les décisions prises sont irréversibles et acceptées par tous les nœuds honnêtes du réseau. Pour cela, un mécanisme de vote est mis en place, où les validateurs votent pour accepter ou rejeter un bloc.

Si un bloc $b_r$ est validé avec exactement 2/3 des votes, il est théoriquement possible qu'un tiers des validateurs ne soient pas d'accord. Dans le pire des cas, ces validateurs pourraient être malveillants et tenter de proposer un autre bloc concurrent $b'_r$. Cela pourrait entraîner une division (fork) de la blockchain, car il n'y aurait pas de consensus clair sur quel bloc doit être accepté.

Cependant, lorsque strictement plus de 2/3 des validateurs signent un QC pour $b_r$, alors chaque nœud honnête verra nécessairement ce QC et considérera $b_r$ comme le seul bloc valide pour ce round. Cela garantit :

— Qu'aucun autre bloc $b'_r$ ne pourra obtenir un QC concurrent, car il est impossible d'obtenir plus de 2/3 des votes pour deux blocs différents simultanément.

— Que tous les nœuds honnêtes auront vu et reconnu la décision, assurant ainsi un consensus unifié.

— Qu'un nœud qui n'a pas encore vu $b_r$ finira par le voir, car la majorité honnête des validateurs l'a signé, garantissant la propagation de l'information.

## 4.11 Pourquoi un bloc non décidé peut être oublié ?

Un bloc $b_r$ qui a été proposé mais qui n'a pas reçu de QC peut être ignoré sans conséquence. Cela signifie que même si un vote a eu lieu pour $b_r$, tant que le seuil strictement supérieur à 2/3 des signatures n'a pas été atteint, ce bloc reste **non décidé**. Dans ce cas, il peut être remplacé par un autre bloc $b_{r+1}$.

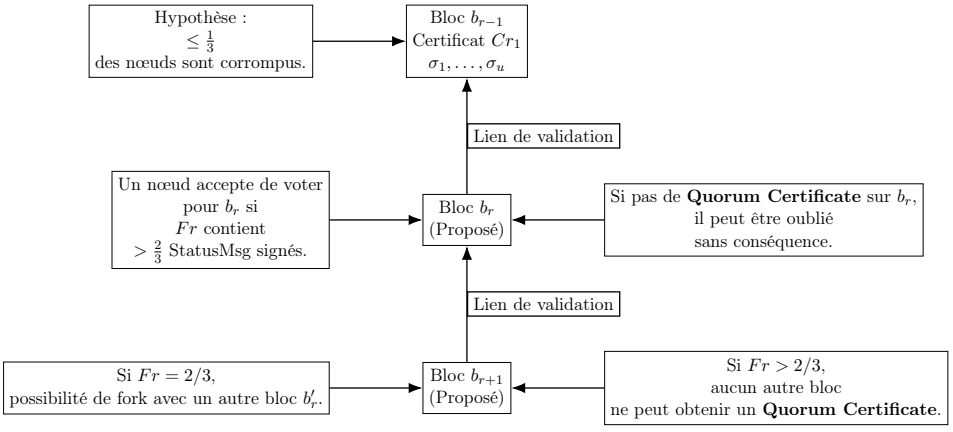


FIGURE 4 – Diagramme de validation des blocs dans un protocole de consensus. Les blocs sont validés séquentiellement, et un bloc est considéré comme décidé uniquement s'il obtient un **Quorum Certificate** avec plus de 2/3 des votes des validateurs. Les flèches indiquent la direction de la validation et les conditions nécessaires pour qu'un bloc soit accepté.

**Explication :**

— Un bloc $b_r$ peut être proposé et recevoir des votes, mais tant qu'il n'a pas de **Quorum Certificate**, il n'est pas officiellement *décidé*.

— Cela signifie qu'un autre bloc $b_{r+1}$ peut être proposé à sa place sans perturber le consensus.

— Les nœuds peuvent alors choisir $b_{r+1}$ et oublier $b_r$, car il n'y a aucun engagement définitif sur $b_r$.

— Ce mécanisme permet d'éviter les blocages et les forks en cas de retard ou de problème de synchronisation.