Introduction à la blockchain

Pascal Pares

Published with GitBook

Table des matières

Avertissement

Cette introduction est une présentation technique des mécanismes de création de la blockchain pour la monnaie Bitcoin. Elle est basée sur l'article fondateur de Satoshi Nakamoto et le site Wiki de bitcoin.org.

Les mécanismes de transactions pour les paiements en monnaie Bitcoin ne sont pas abordés : la structure de la transaction n'est pas décrite, notamment la somme et le fractionnement des montants, le principe de la signature électronique, les clés publiques et privées, et le stockage des transactions dans un bloc avec un arbre de Merkle.

Certaines notions sont illustrées par des extraits de code JavaScript pour *Node.js*.

Historique des révisions :

- 9 avril 2016 : Première publication
- 11 avril 2016 : Corrections mineures
- 25 avril 2016 : Révision de la section *La structure de la blockchain*
- 29 mai 2016 : Révision de la section *L'empreinte numérique d'un bloc*
- 22 octobre 2016 : Révision de la section *La fraude de la double-dépense*

1. Les objectifs de la blockchain

La blockchain est le registre des transactions de la monnaie Bitcoin. Ce registre est créé à partir des messages diffusés sur le réseau Bitcoin. Les choix de conception de la blockchain visent à répondre aux exigences suivantes :

- Les transactions enregistrées sont irréversibles, on ne peut les effacer du registre.
- Les transactions sont infalsifiables.
- Les transactions permettent un paiement sans tiers de confiance.
- La monnaie est créée sans autorité de contrôle.
- Les transactions sont publiques et vérifiables par tous, mais sont anonymes.
- Le système est protégé contre la fraude de la double-dépense.
- Le système vise le commerce électronique sur internet.

2. Le réseau Bitcoin

2.1 Un réseau de pair-à-pair

Le réseau Bitcoin est le réseau internet utilisé comme un réseau de pair-à-pair. Ce réseau de pair-à-pair vise à s'affranchir d'une autorité de contrôle pour la création monétaire, et à s'affranchir d'un tiers de confiance pour les paiements. Tous les participants du réseau Bitcoin ont ainsi le même statut ; aucun participant ne peut se prévaloir d'une quelconque légitimité supérieure. Chaque participant est considéré comme un pair vis-à-vis des autres.

2.2 Les messages

Deux types principaux de messages sont diffusés le plus largement possible sur Internet :

- la transaction, qui représente un paiement ;
- le bloc, qui enregistre une collection de transactions.

Lorsqu'une nouvelle transaction est signée, elle est diffusée sur le réseau Bitcoin. Elle sera ensuite collectée et enregistrée dans un bloc. Chaque bloc, une fois constitué, sera à son tour diffusé.

Tous ces messages sont ainsi publics et vérifiables. Ils permettent de notifier et donc de prendre à témoin l'ensemble des participants du réseau Bitcoin sur toute nouvelle information qui vient enrichir la blockchain.

Les messages sont transférés sur le réseau dans un format binaire, avec un codage des nombres sur 32 bits ou 256 bits en utilisant la convention Little-Endian (cf. la section *La structure de la blockchain*).

2.3 Les participants

Les participants au réseau sont

- soit des parties prenantes d'une transaction ;
- soit des "relais" qui participent à la diffusion des messages ;
- soit des "mineurs" qui participent à la création de blocs et à la diffusion des messages.

Nous utiliserons le terme de participant comme synonyme de nœud (au sens d'une machine connectée au réseau). Les nœuds qui ne sont pas uniquement des parties prenantes d'une transaction sont appelés les nœuds complets (full node).

> Les mécanismes décrits dans les sections qui suivent permettent d'assurer la diffusion des messages sur le réseau Bitcoin (de l'anglais "broadcast") ; cependant ils ne peuvent être imposés aux participants, ni par le protocole, ni par consensus, ils constituent plutôt un catalogue de bonnes pratiques afin d'optimiser les échanges. Ces mécanismes sont implémentés par le logiciel *Bitcoin Core*.

2.4 La diffusion par propagation

Pour diffuser un message, il faut atteindre les nœuds actifs du réseau Bitcoin. Chaque nœud établit une liste de quelques-uns de ses pairs en activité, qui utilisent la même version du protocole. Nous appellerons les pairs actifs sélectionnés (de l'anglais "known active peers"), les contacts. Pour diffuser un message, un nœud le transmet à ses contacts qui à leur tour vont relayer le message à leurs propres contacts, excepté s'ils l'ont déjà transmis. De proche ne proche le message sera propagé à tous les nœuds du réseau.

2.5 La veille périodique

La présence des nœuds actifs est périodiquement constatée par tous les nœuds. Chaque nœud diffuse un message de présence, en transmettant sa propre adresse IP, toutes les 24 heures, à chacun de ses contacts.

Chaque nœud surveille la présence de ses contacts sur le réseau Bitcoin. Toutes les 30 minutes un message est échangé, si aucune réponse n'est reçue dans les 90 minutes, alors ce contact est considéré comme absent.

2.6 La base de données d'adresses IP

Chaque nœud constitue sa base de données des adresses IP de tous les participants au réseau Bitcoin. Cette base est enrichie avec les adresses IP diffusée sur le réseau, et ceci dès l'introduction d'un nouveau participant sur le réseau.

La liste des contacts d'un nœud est établie avec un échantillon d'adresses IP issues de la base de données des adresses et sélectionnées de façon aléatoire.

2.7 La première découverte du réseau

Lors du premier accès au réseau, il est nécessaire d'amorcer une base d'adresses IP.

Des serveurs d'amorçage (seed servers), similaires à des serveurs de noms de domaines (DNS server), délivrent des listes d'adresses IP de nœuds du réseau Bitcoin. Ces listes sont établies périodiquement ou dynamiquement en scannant le réseau. La première liste des serveurs d'amorçage est figée par le logiciel Bitcoin Core à la date du téléchargement du logiciel. Si ces serveurs, ne sont plus opérationnels, une liste d'adresses IP de dernier recours est utilisée.

La première découverte du réseau, en utilisant uniquement les serveurs d'amorçage, pose un problème de sécurité. En effet pour ne pas dépendre d'un tiers de confiance, les serveurs d'amorçage ne disposent pas de certificat de sécurité, leur identité n'est donc pas authentifiée. Un pirate peut donc s'interposer entre un nouveau participant et le réseau Bitcoin, de façon à l'isoler du réseau en proposant ses propres adresses IP.

3. La structure de la blockchain

3.1 L'empilement des blocs

On peut représenter la blockchain comme une pile de blocs dont le sommet est le bloc le plus récent et la base le premier bloc créé par Satoshi Nakamoto, le 3 janvier 2009, le bloc *genesis*. Au fur et à mesure de l'émission des transactions sur le réseau, celles-ci sont enregistrées dans des blocs, lesquels sont empilés pour former la blockchain.

3.2 L'assemblage des blocs

Afin d'interdire toute altération ultérieure de blocs, ou la substitution d'un bloc à un autre bloc, chaque bloc est assemblé numériquement au bloc précédent par une empreinte numérique qui vérifie une preuve-de-travail.

3.3 L'empreinte numérique

Une fonction de hachage cryptographique transforme un message en un code numérique appelé l'empreinte numérique du message. Ce code reflète entièrement le message. La même fonction de hachage appliquée à un message altéré produira un code numérique distinct de son code initial.

3.4 L'algorithme SHA-256

L'algorithme SHA-256 (Secured Hash Algorithm) spécifie une méthode de hachage qui produit une empreinte numérique sur 256 bits. Elle est dite sécurisée car l'altération d'un seul bit du message, produit une empreinte numérique distincte. Il est donc impossible d'altérer un message tout en conservant la même empreinte.

La fonction *hash* ci-dessous applique l'algorithme SHA-256 au message passé en paramètre, le résultat est affiché dans sa représentation hexadécimale :

```
// Calculate a hash code from a binary buffer
var crypto = require("crypto");
var hash = function(buffer) {
    var f = crypto.createHash('sha256');
    var h = f.update(buffer);
    return h.digest();
};

> var message = new Buffer('hello world');
> message.toString('hex');
'68656c6c6f20776f726c64'
> var hashCode = hash(message);
> hashCode.toString('hex');
'b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9'
```

3.5 Le "nonce"

Un nonce est une valeur arbitraire adjointe à un message pour influer sur l'empreinte numérique obtenue ; on applique la fonction de hachage au message concaténé à un nonce codé sur 32 bits avec la convention Little-Endian.

> La convention Little-Endian ordonne les octets du poids le plus faible vers le poids le plus fort, la convention Big-Endian ordonne les octets du poids le plus fort vers le poids le plus faible. Par exemple le nombre 1 est représenté en hexadécimal sur 32 bits par 00000001 avec la convention Big-Endian et 01000000 avec la convention Little-Endian.
>
> Le nombre 1 sur 32 bits avec la convention Big-Endian :
>
> ```
> Octet 3 | Octet 2 | Octet 1 | Octet 0
> --------+---------+---------+--------
>      00 |      00 |      00 |      01
> ```
>
> Le nombre 1 sur 32 bits avec la convention Little-Endian :
>
> ```
> Octet 0 | Octet 1 | Octet 2 | Octet 3
> --------+---------+---------+--------
>      01 |      00 |      00 |      00
> ```

Dans le code ci-dessous, nous représentons le nonce avec un Buffer d'octets, indépendamment de la convention Little-Endian/Big-Endian propre au processeur, puis nous concaténons le message avec le nonce, pour obtenir une nouvelle empreinte numérique.

```
var numberToInt32LE = function (n) {
    var buffer = new Buffer(4);
    buffer.writeUInt32LE(n,0);
    return buffer;
};

> var nonce = numberToInt32LE (1000);
> nonce.toString('hex');
'e8030000'

> var message = new Buffer('hello world');
> var hashCode = hash(Buffer.concat([message,nonce]));
> hashCode.toString('hex');
'51b2583117a8e0c8551883127bba6bdc3fe6603ac58ce4ad5c1c2a6def9be485'
```

3.6 La preuve-de-travail

La preuve-de-travail vise

- à rendre difficile le calcul d'une empreinte numérique en exigeant du temps d'exécution du processeur ;
- à rendre aléatoire le temps nécessaire à ce calcul (cf. le principe du consensus).

L'algorithme SHA-256 produit un nombre qui peut être représenté sur 256 bits, soit un nombre entre 0 et $2^{256} - 1$. Si on considère la fonction de hachage basée sur l'algorithme SHA-256 comme une fonction pseudo-aléatoire, la probabilité d'obtenir un nombre qui commence par N bits à 0 pour sa représentation sur 256 bits est de $1/(2^N)$. Cela revient à segmenter l'intervalle des nombres de 0 à $2^{256} - 1$, en 2, 4, 8, ...2^N segments, et à évaluer la probabilité d'obtenir un nombre du premier segment.

Le tableau ci-dessous donne un échantillon de quelques probabilités :

Nombre de bits à zéro	Probabilité
1	1 / 2
2	1 / 4
3	1 / 8
4	1 / 16
5	1 / 32
16	1 / 65536

Une preuve-de-travail spécifie le nombre N de bits à 0 que doit satisfaire une empreinte numérique par l'adjonction d'un nonce. Pour satisfaire cette contrainte, il faut rechercher un nonce, avec une probabilité de succès de $1/(2^N)$. Cette recherche peut être faite simplement par une itération du nonce jusqu'à satisfaire la contrainte, ou par toute autre méthode, par exemple en tirant des valeurs au "hasard". Quelle que soit la méthode, elle nécessitera du temps d'exécution du processeur, un temps d'autant plus long que la probabilité de succès sera faible.

Pour tester l'empreinte numérique, nous utiliserons la fonction *toReverseHexaNotation* qui inverse la représentation hexadécimal d'un nombre codé sur 32 bits ou 256 bits avec la convention Little-Endian

```
Buffer.prototype.toReverseHexaNotation = function ()
{
    var hexa = "";
    for (var i = this.length-1; i >= 0; i--) {
       var digits =  this[i].toString(16);
        hexa += ("0" + digits).slice(-2); // Add "0" for single digit
    }
    return hexa;
};

/* Return a nonce value for a number of expected bits.
 * Warning: May take a while if the proof-of-work is difficult
 * proof is a string of '0' characters, each character is
 * a hexadecimal digit, so it is 4 zero-bits
 */
var calculateProofOfWork = function(proof, message) {
   var len = proof.length;
   for(var nonce=0;;nonce++){
        var nonceLE= numberToInt32LE(nonce);
        var hashCode = hash(Buffer.concat([message, nonceLE]));
        var result = hashCode.toReverseHexaNotation();
        if (result.substr(0, len) == proof)
            return nonceLE;
    }
};

> var message = new Buffer('hello world');
> var nonce = calculateProofOfWork('00', message);
> nonce.toString('hex');
'0e000000'
```

Une fois la valeur du nonce trouvée, la vérification de la preuve-de-travail est immédiate :

```
> var hashCode = hash(Buffer.concat([message, nonce]));
> hashCode.toReverseHexaNotation();
'003a669f5c15dd75046bae1661f6a7326bbb80ec217080bff5c97286dc03d7c6'
```

Le temps et l'effort de calcul pour trouver le nonce augmente à mesure que la probabilité de succès décroit :

```
var testProofOfWork = function (message, loops) {
    var proof = "0";
    for (var i = 0; i < loops; i++) {
        var t1 = Date.now();
        var nonce = calculateProofOfWork(proof, message);
        var t2 = Date.now();
        var hashCode = hash(Buffer.concat([message, nonce]));
        console.log ("hash:", hashCode.toReverseHexaNotation(),
                     "nonce:", nonce.toReverseHexaNotation(),
                     "elapsedTime:", t2-t1);
        proof += "0";
    }
};
```

```
> testProofOfWork(message, 5);
hash: 003a669f5c15dd75...80bff5c97286dc03d7c6 nonce: 0000000e elapsedTime: 1
hash: 003a669f5c15dd75...80bff5c97286dc03d7c6 nonce: 0000000e elapsedTime: 1
hash: 0001c1c7a764d470...76d596b136ef255ba911 nonce: 00001f0a elapsedTime: 254
hash: 00008e8dc456ce49...a36d96dfafab3f18fba0 nonce: 0000fca8 elapsedTime: 1486
hash: 000003aaa63ca127...35ba4de259c2625b617f nonce: 0008e973 elapsedTime: 12241
```

3.7 La cible de la preuve-de-travail

Le protocole Bitcoin a fait évoluer la preuve-de-travail. Elle n'est plus spécifiée comme un nombre de bits à 0 mais comme une valeur maximale que l'empreinte numérique peut atteindre. Cette valeur maximale est appelée la cible.

Cette cible est représentée sous une forme compacte de 32 bits, appelée *bits*. Cette représentation est basée sur les fonctions OpenSSL *BN_bn2mpi()* et *BN_mpi2bn()* :

- Les 23 bits de poids faible représentent une valeur.
- Le 24ième bit représente le signe négatif.
- L'octet de poids fort est utilisé pour établir un décalage à gauche en nombre d'octets.

La valeur correspondante s'obtient par le calcul suivant :

```
var bits = 0x04012345;
var sign = (bits & 0x00800000) >> 24;
var exponent = (bits & 0xFF000000) >> 24;
var mantissa = (bits & 0x007FFFFF);
var target = (Math.pow(-1,sign) * mantissa) << (8 * (exponent-3));
> target.toString(16)
'1234500'
```

La valeur limite choisie pour la cible, soit l'effort minimum de preuve-de-travail exigé, commence par 32 bits à 0, soit :

```
0x00000000FFFF0000000000000000000000000000000000000000000000000000
```

soit dans sa forme compacte sur 32 bits :

```
0x1d00ffff
```

3.8 L'indice de difficulté

L'indice de difficulté est calculé par le rapport de la valeur limite de la preuve-de-travail avec la cible courante. Cet indice donne une estimation de l'effort de calcul nécessaire pour satisfaire la preuve-de-travail. Plus l'indice est grand, plus l'effort de calcul est important. Une valeur de l'indice égale à 1 correspond à l'effort minimum.

```
0x00000000FFFF0000000000000000000000000000000000000000000000000000
/ 0x00000000000404CB000000000000000000000000000000000000000000000000
= 16307.420938523983
```

3.9 L'ajustement de la preuve-de-travail

La cible de la preuve-de-travail doit être ajustée de façon à générer un bloc en moyenne toutes les 10 minutes.

Si un bloc est généré en moyenne toutes les 10 minutes, alors sur une période de 14 jours, 2016 blocs doivent être générés (2016 blocs = 14 jours * 24 heures * 6 blocs par heure). Ainsi tous les 2016 blocs, la cible est ajustée en la multipliant par un facteur. Ce facteur est déduit de la différence de temps de création effectif des 2016 blocs avec la durée attendue de 14 jours.

3.10 L'empreinte numérique d'un bloc

L'empreinte numérique de chaque bloc est le hachage de la concaténation des éléments de son en-tête. Cet en-tête est constitué :

- du numéro de version de la structure du bloc (*version*) ;
- de l'empreinte numérique du bloc précédent (*previousBlockHash*) ;
- de l'empreinte numérique qui représente le contenu du bloc (*merkleRootHash*) ;
- de l'horodatage du bloc sur 32 bits (*time*);
- de la valeur compacte de la cible de la preuve-de-travail (*bits*) ;
- du nonce qui permet de satisfaire la preuve-de-travail (*nonce*).

> L'empreinte numérique du contenu du bloc est calculée via un arbre de Merkle, la racine de l'arbre de Merkle est l'empreinte numérique pour l'ensemble des transactions enregistrées.

```
// Header from block 125552
var header = {
    version: 1,
    previousBlockHash: '00000000000008a3a41b85b8b29ad444def299fee21793cd8b9e567eab02cd81'
    merkleRootHash: '2b12fcf1b09288fcaff797d71e950e71ae42b91e8bdb2304758dfcffc2b620e3',
    time: new Date ("Sat May 21 2011 17:26:31 GMT+0000 (UTC)"),
    bits: 440711666,
    nonce: 2504433986
};
```

Tous les éléments de l'en-tête sont considérés comme des nombres sur 32 bits ou 256 bits, encodés avec la convention Little-Endian. La fonction *serializeHeader* permet de compiler ces données dans des buffers, avec la convention Little-Endian :

```
var serializeHeader = function (header) {
  var buffers = [];
  buffers.push (numberToInt32LE(header.version));
  buffers.push (hexaNotationToInt256LE(header.previousBlockHash));
  buffers.push (hexaNotationToInt256LE(header.merkleRootHash));
  buffers.push (dateToInt32LE(header.time));
  buffers.push (numberToInt32LE(header.bits));
  buffers.push (numberToInt32LE(header.nonce));
  return Buffer.concat (buffers);
};

var dateToInt32LE = function (date) {
    var time = date.getTime() / 1000; // remove milliseconds
    return numberToInt32LE(time);
};

var hexaNotationToInt256LE = function (hexa)
{
    var bytes = new Array(32);
    for (var i = 0, j = 31, len = hexa.length; i < len; i+=2, j--) {
        bytes[j] = parseInt(hexa[i]+hexa[i+1],16);
    }
    return new Buffer(bytes);
};

> serializeHeader (header).toString('hex');
'0100000081cd02ab7e569e8bcd9317e2fe99f2de44d49ab2b8851ba4a308000000000000e320b6c2fffc8d75
>
```

L'empreinte numérique d'un bloc est calculée sur la concaténation des éléments de l'en-tête, puis une seconde empreinte est calculée sur le résultat, pour des raisons historiques et par prévention.

```
> var hashcode = hash ( hash (serializeHeader (header) ) ); // hash twice
> hashcode.toReverseHexaNotation();
'00000000000000001e8d6829a8a21adc5d38d0a473b144b6765798e61f98bd1d'
```

L'en-tête du bloc genesis est particulier car l'empreinte numérique du bloc précédent est 0.

```
// Header from block 0
var header = {
    version: 1,
    previousBlockHash: '0000000000000000000000000000000000000000000000000000000000000000'
    merkleRootHash: '4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b',
    time: new Date ("Sat Jan 03 2009 18:15:05 GMT+0000 (UTC)"),
    bits: 0x1d00ffff,
    nonce: 2083236893
};

> var hashcode = hash ( hash (serializeHeader (header) ) ); // hash twice
> hashcode.toReverseHexaNotation();
'000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f'
```

Toute altération d'un bloc conduirait ainsi à recalculer l'empreinte numérique de ce bloc et de tous les blocs qui le suivent. Le temps de recalcul sera d'autant plus grand que l'on remonte dans la blockchain.

En conclusion, l'assemblage des blocs possèdent les caractéristiques suivantes :

- Le temps nécessaire pour assembler deux blocs est aléatoire, et coûteux, il est en moyenne de 10 minutes.
- Il est facile de vérifier la validité de l'assemblage des blocs.
- Il est impossible d'altérer un bloc sans modifier son empreinte numérique, et celles des blocs qui le suivent.
- Il est impossible d'insérer un bloc sans recalculer les empreintes numériques des blocs qui le suivent.

4. La construction de la blockchain

4.1 Les mineurs

Les blocs de la blockchain sont créés et diffusés sur internet par les mineurs. Le caractère aléatoire de temps de calcul de la preuve-de-travail permet de distribuer les chances de création des blocs à tous les mineurs. Le mineur qui résoudra le premier une preuve-de-travail est, en quelque sorte, tiré au sort.

4.2 L'incitation

Pour inciter les mineurs à participer à la création de la blockchain, et investir leur puissance de calcul en preuve-de-travail, les mineurs peuvent prélever des frais sur les transactions enregistrées dans un bloc ou créer des Bitcoins.

Le terme de mineur a été choisi par analogie avec les chercheurs d'or qui par leurs efforts introduisent des pièces d'or en circulation. La création de bloc exige de la puissance de calcul, elle est donc rétribuée par une transaction spéciale, créée par le mineur pour lui-même.

4.3 Les instances de la blockchain

Au fur et à mesure de la création des blocs et de leur diffusion sur le réseau, chaque mineur assemble les blocs créés par lui-même et par ses pairs pour former sa propre instance de la blockchain.

La blockchain est ainsi stockée localement par chacun des mineurs.

4.4 Le protocole

Le protocole définit le format des messages diffusés ainsi que les règles de sélection des blocs, et de sélection de la blockchain parmi les instances concurrentes, afin que chaque mineur, bien que non contraint par une autorité de contrôle, parvienne à une instance identique de la blockchain :

- Un bloc est approuvé par un mineur, s'il vérifie trois conditions :
  - son empreinte numérique est calculée avec son en-tête ;
  - son empreinte numérique vérifie la preuve-de-travail ;
  - son contenu est valide.
- Chaque mineur ajoute au sommet de sa blockchain tout bloc approuvé.
- Chaque mineur diffuse par broadcast tout bloc qu'il crée de façon à faire valider et adopter ce bloc par ses pairs qui construiront alors la même instance de la blockchain.
- Si un mineur observe la mise en concurrence de plusieurs instances de la blockchain en recevant simultanément deux blocs candidats. Il est amené à conserver les deux instances, lesquelles divergeront à partir de ces deux blocs reçus. Le mineur doit privilégier la première l'instance qui correspond au premier bloc reçu.

- Si un mineur constate une progression plus rapide de l'une de ses instances alors il doit se reporter sur celle-ci. Un mineur doit toujours opter pour la plus longue de ses instances.
- Si un mineur constate qu'il ne peut assembler un bloc, cela signifie qu'il lui manque une instance de blockchain susceptible de devenir la plus longue. Il doit interroger ses pairs pour obtenir les blocs manquants ; c'est le cas d'un mineur qui a quitté le réseau volontairement ou du fait d'un incident d'exploitation.

4.5 Le consensus

Sans autorité de contrôle il n’est pas possible de vérifier que chaque mineur respecte le protocole. Le protocole est en fait imposé par consensus.

Un mineur qui n'observe pas le protocole, quel que soit sa raison (tentative de fraude, bogue, conception inadaptée, application de règles obsolètes, etc.), ne pourra assembler dans son instance de blockchain les nouveaux blocs créés par la majorité des mineurs. Autrement dit il sera mis en minorité vis à vis de ses pairs. De même les blocs qu'il crée ne s'assembleront pas à la plus longue instance de blockchain. C'est la raison pour laquelle un mineur ne peut disposer immédiatement de la transaction spéciale qui le rétribue pour son travail. Cette transaction devient valide lorsque suffisamment de blocs ont été ajoutés à la suite du bloc qu'il a créé.

La blockchain officiellement reconnue est donc la plus longue instance, soit celle adoptée à la majorité.

On peut noter ici que le comportement de la majorité des mineurs fait foi ; même si ce comportement est jugé arbitraire (comme se reporter sur l'instance de blockchain du premier bloc reçu) voire même dans certains cas erroné (si la majorité des mineurs utilise un logiciel commun bogué). Ce n'est donc par la spécification du protocole qui compte mais l'implémentation du protocole par le logiciel utilisé par la majorité des mineurs. C'est pourquoi pour amorcer le système, le partage d'un logiciel gratuit et ouvert est une condition nécessaire.

4.6 L'autorégulation

Le système se régule en fonction du trafic, par le consensus et par la mutualisation des ressources des mineurs.

L'ajustement de la preuve-de-travail tous les 2016 blocs est une règle du protocole. Comme toute règle consensuelle, si un mineur ne respecte cette preuve-de-travail ajustée, il sera mis en minorité, et ses blocs seront rejetés par la majorité. La fréquence de 10 minutes

correspond à une projection, estimée en 2008, des capacités de stockage des ordinateurs dans les années à venir, notamment pour conserver les en-têtes de blocs en mémoire vive.

Si la difficulté de la preuve-de-travail augmente, exigeant une puissance de calcul de plus en plus grande, les mineurs peuvent alors mutualiser leur ressource dans des pools de mineurs pour atteindre la puissance de calcul requise.

Enfin le nombre moyen de transactions par bloc est induit par les frais prélevés sur les transactions, car ceux-ci compensent le coût en électricité requis par la preuve-de-travail. La taille maximale d'un bloc est cependant fixée à 1 Mo afin de garantir une diffusion rapide des blocs sur le réseau.

5. La fraude de la double-dépense

5.1 Principe de la fraude

La double dépense consiste à émettre deux transactions qui dépensent le même avoir : la première transaction est émise pour payer un premier destinataire, la seconde transaction est émise pour payer un complice ou le pirate lui-même, afin de récupérer la somme dépensée.

Afin que la fraude ne soit pas immédiatement découverte, ces deux transactions doivent coexister dans deux instances concurrentes de la blockchain. La seconde instance doit devenir la plus longue blockchain pour que la fraude réussisse. Dans ce cas, la première transaction ne pourra pas être recyclée dans un autre bloc car elle sera jugée invalide car incompatible avec la seconde transaction et sera donc finalement rejetée. Si entre les deux transactions, le premier destinataire a accepté le paiement, il ne découvrira qu'après coup que cette transaction a été rejetée. Puisque les participants sont anonymes, il ne pourra se retourner contre le fraudeur.

5.2 Contrecarrer la fraude

Pour veiller à ne pas être victime de la fraude, le destinataire de paiement doit s'assurer que la transaction émise par le payeur est bien enregistrée dans un bloc de la plus longue des instances et que cette instance est pérenne. C'est pourquoi, il doit attendre qu'un nombre suffisant de blocs succèdent au bloc qui enregistre sa transaction, avant d'accepter le paiement.

La structure d'un bloc par l'arbre de Merkle permet de restituer un bloc allégé ; le destinataire de paiement peut ainsi vérifier une transaction sans avoir à télécharger le contenu complet des blocs.

Si le nombre de bloc successeurs est suffisamment élevé, et que le pirate ne dispose pas de plus de 51% de la puissance de calcul total alors le risque de double-dépense tend vers zéro (cf. la démonstration par Satoshi Nakamoto basé sur la probabilité de Poisson).

6. Conclusion

La blockchain est ainsi un registre répliqué par chaque mineur du réseau. Les principes de la preuve-de-travail et du consensus permettent de s'assurer qu'il évolue de façon synchrone. Ces principes offrent les bénéfices suivants :

- Il laisse au mineur une certaine latitude pour adopter la stratégie de son choix pour la création de blocs (nombre de transaction par bloc, fréquence d'émission des blocs, etc.).
- Un mineur peut joindre ou quitter le réseau à tout moment.
- Des transactions ou des blocs peuvent être perdus pour un mineur, ou être reçus en retard.
- La fraude est contrecarrée tant que la puissance de calcul détenue par les mineurs honnêtes est majoritaire.
- La fraude est dissuadée par le principe d'incitation à créer des blocs valides, car il autorise la création de Bitcoins ou le prélèvement de frais sur les transactions.
- La répartition des ressources matérielles parmi les participants du réseau offre une plus grande robustesse qu'un système centralisé, car elle permet une redondance des processeurs et du registre des transactions.
- La mise en concurrence des mineurs pour le calcul de la preuve-de-travail est en soi une parallélisation du calcul et une mutualisation des processeurs.