Le livre blanc Bitcoin de Satoshi Nakamoto

2022-03-17

# Le livre blanc Bitcoin de Satoshi Nakamoto

Ceci est le livre blanc original, reproduit dans son intégralité exactement tel qu'il a été publié par Satoshi Nakamoto en octobre 2008 et traduit (par assistance informatique) par Serafim Dos Santos en 2022 pour l’édition ouverte de *Maîtriser Bitcoin* (voir version originale *Mastering Bitcoin*).

# Bitcoin - Un système de paiement électronique pair à pair

Satoshi Nakamoto

*satoshin@gmx.com*

www.bitcoin.org

**Résumé.** Une version purement pair à pair de la monnaie électronique permettrait d'envoyer des paiements en ligne directement d'une partie à une autre sans passer par une institution financière. Les signatures numériques fournissent une partie de la solution, mais les principaux avantages sont perdus si un tiers de confiance est toujours requis pour éviter les doubles dépenses. Nous proposons une solution au problème de la double dépense en utilisant un réseau pair à pair. Le réseau horodate les transactions en les hachant dans une chaîne continue de preuves de travail (proof-of-work) basées sur le hachage, formant un enregistrement qui ne peut pas être modifié sans refaire la preuve de travail. La chaîne la plus longue sert non seulement de preuve de la séquence d'événements observés, mais aussi de preuve qu'elle provient du plus grand bassin de puissance CPU. Tant que la majorité de la puissance du processeur est contrôlée par des nœuds qui ne coopèrent pas pour attaquer le réseau, ils généreront la plus longue chaîne et devanceront les attaquants. Le réseau lui-même nécessite une structure minimale. Les messages sont diffusés dans la mesure du possible, et les nœuds peuvent quitter et rejoindre le réseau à volonté, acceptant la plus longue chaîne de preuve de travail comme preuve de ce qui s'est passé pendant leur absence.

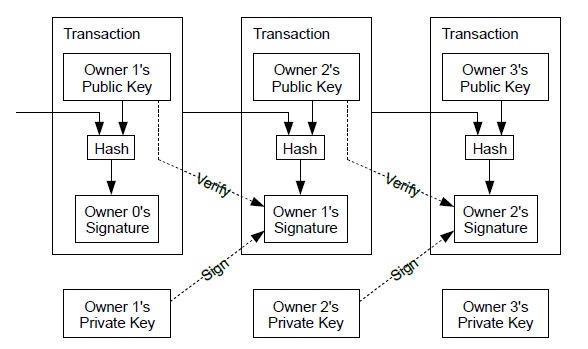
## Introduction

Le commerce sur Internet en est venu à s'appuyer presque exclusivement sur des institutions financières servant de tiers de confiance pour traiter les paiements électroniques. Bien que le système fonctionne assez bien pour la plupart des transactions, il souffre toujours des faiblesses inhérentes au modèle basé sur la confiance. Les transactions totalement irréversibles ne sont pas vraiment possibles, car les institutions financières ne peuvent éviter la médiation des différends. Le coût de la médiation augmente les coûts de transaction, limitant la taille pratique minimale des transactions et supprimant la possibilité de petites transactions occasionnelles, et il y a un coût plus important dans la perte de capacité à effectuer des paiements irréversibles pour des services irréversibles. Avec la possibilité d'inversion, le besoin de confiance se répand. Les commerçants doivent se méfier de leurs clients et les harceler pour obtenir plus d'informations qu'ils n'en auraient autrement besoin. Un certain pourcentage de fraude est accepté comme inévitable. Ces coûts et incertitudes de paiement peuvent être évités en personne en utilisant de la monnaie physique, mais aucun mécanisme n'existe pour effectuer des paiements via un canal de communication sans une partie de confiance.

Ce qu'il faut, c'est un système de paiement électronique basé sur la preuve cryptographique au lieu de la confiance, permettant à deux parties consentantes d'effectuer des transactions directement l'une avec l'autre sans avoir besoin d'un tiers de confiance. Les transactions impossibles à annuler sur le plan informatique protégeraient les vendeurs contre la fraude, et des mécanismes de séquestre de routine pourraient facilement être mis en œuvre pour protéger les acheteurs. Dans cet article, nous proposons une solution au problème de la double dépense en utilisant un serveur d'horodatage distribué pair à pair pour générer une preuve informatique de l'ordre chronologique des transactions. Le système est sécurisé tant que les nœuds honnêtes contrôlent collectivement plus de puissance CPU que n'importe quel groupe coopérant de nœuds attaquants.

## Transactions

Nous définissons une pièce électronique (ou unité de cryptomonnaie) comme une chaîne de signatures numériques. Chaque propriétaire transfère la pièce au suivant en signant numériquement un résultat de hachage de la transaction précédente et la clé publique du propriétaire suivant et en les ajoutant à la fin de la pièce. Un bénéficiaire peut vérifier les signatures pour vérifier la chaîne de propriété.

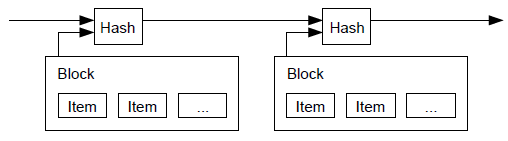


Le problème est bien sûr que le bénéficiaire ne peut pas vérifier que l'un des propriétaires n'a pas dépensé deux fois la pièce. Une solution courante consiste à introduire une autorité centrale de confiance, ou "maison ou hôtel de la monnaie", qui vérifie chaque transaction pour les doubles dépenses. Après chaque transaction, la pièce doit être retournée à l’hôtel de la monnaie pour émettre une nouvelle pièce, et seules les pièces émises directement par l’hôtel de la monnaie sont réputées ne pas être dépensées en double. Le problème avec cette solution est que le sort de l'ensemble du système monétaire dépend de la société qui gère la monnaie, chaque transaction devant passer par elle, tout comme une banque.

Nous avons besoin d'un moyen pour que le bénéficiaire sache que les propriétaires précédents n'ont signé aucune transaction antérieure. Pour nos besoins, la première transaction est celle qui compte, donc nous ne nous soucions pas des tentatives ultérieures de doubler les dépenses. La seule façon de confirmer l'absence d'une transaction est d'être au courant de toutes les transactions. Dans le modèle basé sur la monnaie, la monnaie était au courant de toutes les transactions et décidait laquelle arrivait en premier. Pour y parvenir sans partie de confiance, les transactions doivent être annoncées publiquement [1], et nous avons besoin d'un système permettant aux participants de s'entendre sur un historique unique de l'ordre dans lequel elles ont été reçues. Le bénéficiaire doit avoir la preuve qu'au moment de chaque transaction, la majorité des nœuds ont convenu qu'il s'agissait du premier reçu.

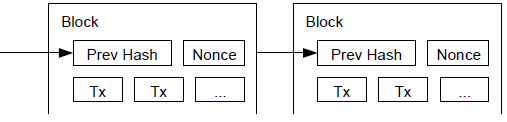
## Serveur d'horodatage

La solution que nous proposons commence par un serveur d'horodatage. Un serveur d'horodatage fonctionne en prenant un hachage d'un bloc d'éléments à horodater et en publiant largement le hachage, comme dans un journal ou une publication Usenet [2-5]. L'horodatage prouve que les données doivent avoir existé à l'époque, évidemment, pour entrer dans le hachage. Chaque horodatage inclut l'horodatage précédent dans son hachage, formant une chaîne où chaque horodatage supplémentaire renforce ceux qui le précèdent.



## Proof-of-Work (ou Preuve de travail)

Pour implémenter un serveur d'horodatage distribué sur une base pair à pair, nous devrons utiliser un système de preuve de travail similaire à Hashcash d'Adam Back [6], plutôt que des articles de journaux ou Usenet. La preuve de travail implique la recherche d'une valeur qui, lorsqu'elle est hachée, comme avec SHA-256, le hachage commence par un certain nombre de bits zéro. Le travail moyen requis est exponentiel dans le nombre de bits zéro requis et peut être vérifié en exécutant un seul hachage. Pour notre réseau d'horodatage, nous implémentons la preuve de travail en incrémentant un nonce dans le bloc jusqu'à ce qu'une valeur soit trouvée qui donne au hachage du bloc les bits zéro requis. Une fois que l'effort CPU a été dépensé pour satisfaire la preuve de travail, le bloc ne peut pas être modifié sans refaire le travail. Comme les blocs ultérieurs sont enchaînés après celui-ci, le travail de modification du bloc comprendrait la refonte de tous les blocs après celui-ci.



La preuve de travail résout également le problème de la détermination de la représentation dans la prise de décision à la majorité. Si la majorité était basée sur une adresse IP, un vote, elle pourrait être renversée par toute personne capable d'allouer de nombreuses adresses IP. La preuve de travail est essentiellement un processeur, un vote. La décision majoritaire est représentée par la chaîne la plus longue, qui a investi le plus grand effort de preuve de travail. Si la majorité de la puissance du processeur est contrôlée par des nœuds honnêtes, la chaîne honnête se développera le plus rapidement et dépassera toutes les chaînes concurrentes. Pour modifier un bloc passé, un attaquant devrait refaire la preuve de travail du bloc et de tous les blocs qui le suivent, puis rattraper et dépasser le travail des nœuds honnêtes. Nous montrerons plus tard que la probabilité qu'une attaque plus lente rattrape le système, diminue de manière exponentielle à mesure que des blocs ultérieurs sont ajoutés.

Pour compenser l'augmentation de la vitesse matérielle et l'intérêt variable pour l'exécution des nœuds au fil du temps, la difficulté de la preuve de travail est déterminée par une moyenne mobile ciblant un nombre moyen de blocs par heure. S'ils sont générés trop rapidement, la difficulté augmente.

## Réseau

Les étapes pour exécuter le réseau sont les suivantes :

1. Les nouvelles transactions sont diffusées à tous les nœuds.
2. Chaque nœud collecte les nouvelles transactions dans un bloc.
3. Chaque nœud travaille à trouver une preuve de travail difficile pour son bloc.
4. Lorsqu'un nœud trouve une preuve de travail, il diffuse le bloc à tous les nœuds.
5. Les nœuds n'acceptent le bloc que si toutes les transactions qu'il contient sont valides et non déjà dépensées.
6. Les nœuds expriment leur acceptation du bloc en travaillant à la création du bloc suivant dans la chaîne, en utilisant le hachage du bloc accepté comme hachage précédent.

Les nœuds considèrent toujours que la chaîne la plus longue est la bonne et continueront à travailler pour l'étendre. Si deux nœuds diffusent simultanément des versions différentes du bloc suivant, certains nœuds peuvent recevoir l'un ou l'autre en premier. Dans ce cas, ils travaillent sur la première branche qu'ils ont reçue, mais conservent l'autre branche au cas où elle deviendrait plus longue. L'égalité sera rompue lorsque la prochaine preuve de travail sera trouvée et qu'une branche s'allongera ; les nœuds qui travaillaient sur l'autre branche passeront alors à la plus longue.

Les diffusions de nouvelles transactions n'ont pas nécessairement besoin d'atteindre tous les nœuds. Tant qu'ils atteignent de nombreux nœuds, ils entreront dans un bloc avant longtemps. Les diffusions en bloc sont également tolérantes aux messages abandonnés. Si un nœud ne reçoit pas de bloc, il le demandera lorsqu'il recevra le bloc suivant et réalisera qu'il en a manqué un.

## Incitatif

Par convention, la première transaction d'un bloc est une transaction spéciale qui démarre une nouvelle pièce appartenant au créateur du bloc. Cela ajoute une incitation pour les nœuds à prendre en charge le réseau et fournit un moyen de distribuer initialement les pièces en circulation, car il n'y a pas d'autorité centrale pour les émettre. L'ajout régulier d'une quantité constante de nouvelles pièces est analogue aux mineurs d'or qui dépensent des ressources pour ajouter de l'or à la circulation. Dans notre cas, c'est le temps CPU et l'électricité qui sont dépensés.

L'incitation peut également être financée par des frais de transaction. Si la valeur de sortie d'une transaction est inférieure à sa valeur d'entrée, la différence est une commission de transaction qui s'ajoute à la valeur incitative du bloc contenant la transaction. Une fois qu'un nombre prédéterminé de pièces est entré en circulation, l'incitation peut passer entièrement aux frais de transaction et être totalement exempte d'inflation.

L'incitation peut aider à encourager les nœuds à rester honnêtes. Si un attaquant cupide est capable d'assembler plus de puissance CPU que tous les nœuds honnêtes, il devrait choisir entre l'utiliser pour frauder les gens en volant leur paiements, ou l'utiliser pour générer de nouvelles pièces. Il devrait trouver plus avantageux de jouer selon les règles que de saper le système et la validité de sa propre richesse. De telles règles le favorise avec plus de nouvelles pièces plus que tout les nœuds réunis.

## Récupération d'espace disque

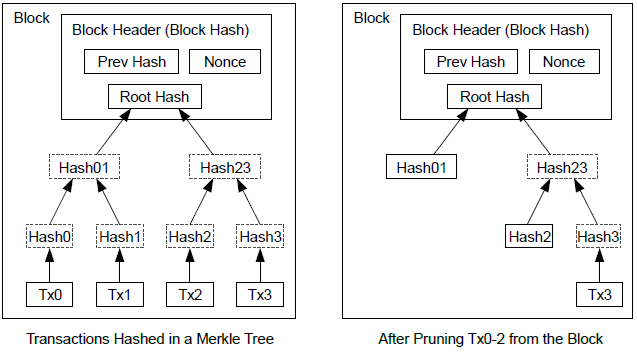
Une fois que la dernière transaction d'une pièce est enterrée sous suffisamment de blocs, les transactions passées avant peuvent être supprimées pour économiser de l'espace disque. Pour faciliter cela sans casser le hachage du bloc, les transactions sont hachées dans un arbre Merkle

[7]

[2]

[5]

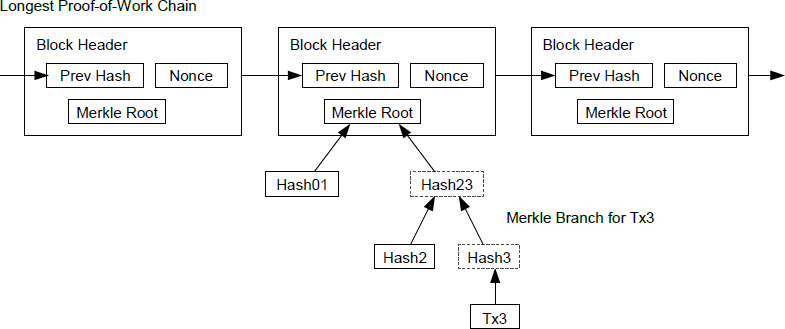
, avec seulement la racine incluse dans le hachage du bloc. Les vieux blocs peuvent ensuite être compactés en écrasant les branches de l'arbre. Les hachages intérieurs n'ont pas besoin d'être stockés.



Un en-tête de bloc sans transactions serait d'environ 80 octets. Si nous supposons que les blocs sont générés toutes les 10 minutes, 80 octets \* 6 \* 24 \* 365 = 4,2 Mo par an. Avec des systèmes informatiques se vendant généralement avec 2 Go de RAM à partir de 2008 et la loi de Moore prédisant une croissance actuelle de 1,2 Go par an, le stockage ne devrait pas être un problème même si les en-têtes de bloc doivent être conservés en mémoire.

## Vérification de paiement simplifiée

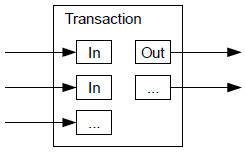
Il est possible de vérifier les paiements sans exécuter un nœud de réseau complet. Un utilisateur n'a besoin que de conserver une copie des en-têtes de bloc de la chaîne de preuve de travail la plus longue, qu'il peut obtenir en interrogeant les nœuds du réseau jusqu'à ce qu'il soit convaincu qu'il a la chaîne la plus longue, et d'obtenir la branche Merkle reliant la transaction au bloc dont il est horodaté. Il ne peut pas vérifier la transaction par lui-même, mais en la liant à un endroit de la chaîne, il peut voir qu'un nœud du réseau l'a acceptée, et les blocs ajoutés après confirment que le réseau l'a acceptée aussi.



Ainsi, la vérification est fiable tant que des nœuds honnêtes contrôlent le réseau, mais est plus vulnérable si le réseau est maîtrisé par un attaquant. Alors que les nœuds du réseau peuvent vérifier les transactions par eux-mêmes, la méthode simplifiée peut être trompée par les transactions fabriquées par un attaquant, tant que l'attaquant peut continuer à maîtriser le réseau. Une stratégie de protection contre cela consisterait à accepter les alertes des nœuds du réseau lorsqu'ils détectent un bloc invalide, incitant le logiciel de l'utilisateur à télécharger le bloc complet et les transactions alertées pour confirmer l'incohérence. Les entreprises qui reçoivent des paiements fréquents voudront probablement toujours exécuter leurs propres nœuds pour une sécurité plus indépendante et une vérification plus rapide.

## Combiner et diviser la valeur

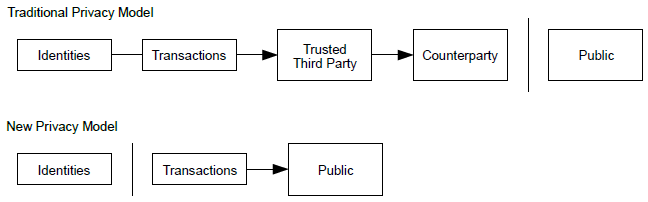
Bien qu'il soit possible de gérer les pièces individuellement, il serait difficile d'effectuer une transaction distincte pour chaque centime d'un transfert. Pour permettre à la valeur d'être fractionnée et combinée, les transactions contiennent plusieurs entrées et sorties. Normalement, il y aura soit une seule entrée provenant d'une transaction précédente plus importante, soit plusieurs entrées combinant des montants plus petits, et au plus deux sorties : une pour le paiement et une renvoyant la monnaie, le cas échéant, à l'expéditeur.



Il convient de noter que la diffusion, où une transaction dépend de plusieurs transactions, et ces transactions dépendent de beaucoup d'autres, n'est pas un problème ici. Il n'est jamais nécessaire d'extraire une copie autonome complète de l'historique d'une transaction.

## Confidentialité

Le modèle bancaire traditionnel atteint un niveau de confidentialité en limitant l'accès aux informations des parties concernées et au tierce partie de confiance. La nécessité d'annoncer publiquement toutes les transactions exclut cette méthode, mais la confidentialité peut toujours être préservée en interrompant le flux d'informations à un autre endroit : en gardant les clés publiques anonymes. Le public peut voir que quelqu'un envoie un montant à quelqu'un d'autre, mais sans information liant la transaction à qui que ce soit. Ceci est similaire au niveau d'information publié par les bourses, où l'heure et la taille des transactions individuelles, la "bande", sont rendues publiques, mais sans dire qui étaient les parties.



En tant que pare-feu supplémentaire, une nouvelle paire de clés doit être utilisée pour chaque transaction afin d'éviter qu'elles ne soient liées à un propriétaire commun. Certains liens sont encore inévitables avec les transactions multi-entrées, qui révèlent nécessairement que leurs entrées appartenaient au même propriétaire. Le risque est que si le propriétaire d'une clé est révélé, la liaison pourrait révéler d'autres transactions ayant appartenu au même propriétaire.

## Calculs

Nous considérons le scénario d'un attaquant essayant de générer une chaîne alternative plus rapidement que la chaîne honnête. Même si cela est accompli, cela n'ouvre pas le système à des changements arbitraires, comme créer de la valeur à partir de rien ou prendre de l'argent qui n'a jamais appartenu à l'attaquant. Les nœuds n'accepteront pas une transaction invalide comme paiement, et les nœuds honnêtes n'accepteront jamais un bloc les contenant. Un attaquant ne peut essayer de modifier qu'une de ses propres transactions pour récupérer l'argent qu'il a récemment dépensé.

La course entre la chaîne honnête et une chaîne attaquante peut être caractérisée comme une marche aléatoire binomiale. L'événement de succès est la chaîne honnête prolongée d'un bloc, augmentant son avance de +1, et l'événement d'échec est la chaîne de l'attaquant prolongée d'un bloc, réduisant l'écart de -1.

La probabilité qu'un attaquant rattrape un déficit donné est analogue à un problème de la ruine du joueur (Gambler's Ruin). Supposons qu'un joueur avec un crédit illimité commence avec un déficit et joue potentiellement un nombre infini d'essais pour essayer d'atteindre le seuil de rentabilité. Nous pouvons calculer la probabilité qu'il atteigne jamais le seuil de rentabilité, ou qu'un attaquant rattrape jamais la chaîne honnête, comme suit

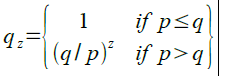
[8]

:

p = probabilité qu'un nœud honnête trouve le bloc suivant

q = probabilité que l'attaquant trouve le bloc suivant

qz = probabilité que l'attaquant rattrape jamais z blocs derrière



Étant donné notre hypothèse que p > q, la probabilité chute de façon exponentielle à mesure que le nombre de blocs que l'attaquant doit rattraper augmente. Avec les chances contre lui, s'il n’a pas un coup de chance deavant dès le début, ses chances deviennent infiniment petites à mesure qu'il prend du retard.

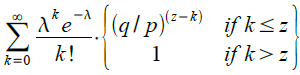
Nous considérons maintenant combien de temps le destinataire d'une nouvelle transaction doit attendre avant d'être suffisamment certain que l'expéditeur ne peut pas modifier la transaction. Nous supposons que l'expéditeur est un attaquant qui veut faire croire au destinataire qu'il l'a payé pendant un certain temps, puis le changer pour se rembourser après un certain temps. Le destinataire sera alerté lorsque cela se produira, mais l'expéditeur espère qu'il sera trop tard.

Le récepteur génère une nouvelle paire de clés et donne la clé publique à l'expéditeur peu de temps avant la signature. Cela évite à l'expéditeur de préparer une chaîne de blocs à l'avance en y travaillant continuellement jusqu'à ce qu'il ait la chance d'aller assez loin, puis d'exécuter la transaction à ce moment-là. Une fois la transaction envoyée, l'expéditeur malhonnête commence à travailler en secret sur une chaîne parallèle contenant une version alternative de sa transaction.

Le destinataire attend que la transaction ait été ajoutée à un bloc et que z blocs aient été liés après celle-ci. Il ne connaît pas la progression exacte de l'attaquant, mais en supposant que les blocs honnêtes ont pris le temps moyen attendu par bloc, la progression potentielle de l'attaquant sera une distribution de Poisson avec une valeur attendue :

"eq2"

Pour obtenir la probabilité que l'attaquant puisse encore rattraper maintenant, nous multiplions la densité de Poisson pour chaque quantité de progrès qu'il aurait pu faire par la probabilité qu'il puisse rattraper à partir de ce point :



Réorganiser pour éviter de faire la somme de la queue infinie de la distribution…​

"eq4"

Conversion en code C…​

#include <math.h>  
double AttackerSuccessProbability(double q, int z)  
{  
 double p = 1.0 - q;  
 double lambda = z \* (q / p);  
 double sum = 1.0;  
 int i, k;  
 for (k = 0; k <= z; k++)  
 {  
 double poisson = exp(-lambda);  
 for (i = 1; i <= k; i++)  
 poisson \*= lambda / i;  
 sum -= poisson \* (1 - pow(q / p, z - k));  
 }  
 return sum;  
}

En exécutant certains résultats, nous pouvons voir la probabilité chuter de façon exponentielle avec z.

q=0.1  
z=0 P=1.0000000  
z=1 P=0.2045873  
z=2 P=0.0509779  
z=3 P=0.0131722  
z=4 P=0.0034552  
z=5 P=0.0009137  
z=6 P=0.0002428  
z=7 P=0.0000647  
z=8 P=0.0000173  
z=9 P=0.0000046  
z=10 P=0.0000012

q=0.3  
z=0 P=1.0000000  
z=5 P=0.1773523  
z=10 P=0.0416605  
z=15 P=0.0101008  
z=20 P=0.0024804  
z=25 P=0.0006132  
z=30 P=0.0001522  
z=35 P=0.0000379  
z=40 P=0.0000095  
z=45 P=0.0000024  
z=50 P=0.0000006

Résolution pour P inférieur à 0,1 %…​

P < 0.001  
q=0.10 z=5  
q=0.15 z=8  
q=0.20 z=11  
q=0.25 z=15  
q=0.30 z=24  
q=0.35 z=41  
q=0.40 z=89  
q=0.45 z=340

## Conclusion

Nous avons proposé un système de transactions électroniques sans reposer sur la confiance. Nous avons commencé avec le cadre habituel des pièces fabriquées à partir de signatures numériques, qui offre un fort contrôle de propriété, mais qui est incomplet sans un moyen d'éviter les doubles dépenses. Pour résoudre ce problème, nous avons proposé un réseau pair à pair utilisant la preuve de travail pour enregistrer un historique public des transactions qui devient rapidement impossible à modifier par un attaquant si des nœuds honnêtes contrôlent la majorité de la puissance du processeur. Le réseau est robuste dans sa simplicité non structurée. Les nœuds fonctionnent tous en même temps avec peu de coordination. Ils n'ont pas besoin d'être identifiés, car les messages ne sont pas acheminés vers un endroit particulier et ne doivent être livrés que dans la mesure du possible. Les nœuds peuvent quitter et rejoindre le réseau à volonté, acceptant la chaîne de preuve de travail comme preuve de ce qui s'est passé pendant leur absence. Ils votent avec leur puissance CPU, exprimant leur acceptation des blocs valides en travaillant à les étendre et rejetant les blocs invalides en refusant de travailler dessus. Toutes les règles et incitations nécessaires peuvent être appliquées avec ce mécanisme de consensus.

## Références

[1]

W. Dai, "b-money,"

http://www.weidai.com/bmoney.txt

, 1998.

[2]

H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In 20th Symposium on Information Theory in the Benelux, May 1999.

[3]

S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In Journal of Cryptology, vol 3, no 2, pages 99-111, 1991.

[4]

D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In Sequences II: Methods in Communication, Security and Computer Science, pages 329-334, 1993.

[5]

S. Haber, W.S. Stornetta, "Secure names for bit-strings," In Proceedings of the 4th ACM Conference on Computer and Communications Security, pages 28-35, April 1997.

[6]

A. Back, "Hashcash - a denial of service counter-measure,"

http://www.hashcash.org/papers/hashcash.pdf

, 2002.

[7]

R.C. Merkle, "Protocols for public key cryptosystems," In Proc. 1980 Symposium on Security and Privacy, IEEE Computer Society, pages 122-133, April 1980.

[8]

W. Feller, "An introduction to probability theory and its applications," 1957.

# Licence

Ce livre blanc a été publié en octobre 2008 par Satoshi Nakamoto. Il a ensuite été (2009) ajouté comme documentation de support au logiciel bitcoin, qui porte la même licence MIT. Il a été reproduit dans ce livre, sans modification autre que la mise en forme, selon les termes de la licence MIT :

The MIT License (MIT) Copyright (c) 2008 Satoshi Nakamoto

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.