Bitcoin : un système de paiement électronique peer-to-peer

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Résumé. Une version purement peer-to-peer 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 constituent 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 double dépense en utilisant un réseau peer-to-peer. Le réseau horodatage les transactions en les hachant dans une chaîne continue de preuves de travail 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 des événements observés, mais aussi de preuve qu'elle provient du plus grand pool de puissance CPU. Tant qu'une majorité de la puissance du processeur est contrôlée par des nœuds qui ne coopèrent pas pour attaquer le réseau, ils générera la chaîne la plus longue et devancera les attaquants. Le réseau lui-même nécessite une structure minimale. Les messages sont diffusés au mieux et les nœuds peuvent quitter et rejoindre le réseau à volonté, en acceptant la plus longue chaîne de preuve de travail comme preuve de ce qui s'est passé pendant leur absence.

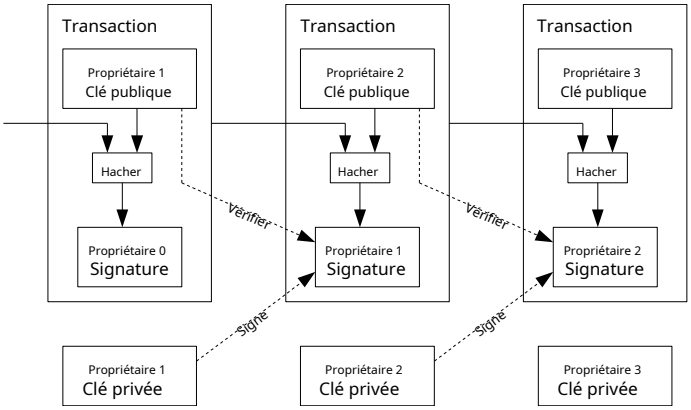
1. Introduction

Le commerce sur Internet repose presque exclusivement sur des institutions financières faisant office 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. Des 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 minimale pratique de la transaction et éliminant la possibilité de petites transactions occasionnelles, et il y a un coût plus important dans la perte de capacité à effectuer des paiements non réversibles pour des services non réversibles. Avec la possibilité de renversement, le besoin de confiance se répand. Les commerçants doivent se méfier de leurs clients, les harcelant 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 sur un canal de communication sans une partie de confiance.

Ce dont nous avons besoin, c'est d'un système de paiement électronique basé sur une preuve cryptographique au lieu de la confiance, permettant à deux parties consentantes de traiter directement l'une avec l'autre sans avoir besoin d'un tiers de confiance. Les transactions qui sont informatiquement impossibles à annuler protégeraient les vendeurs de la fraude, et des mécanismes d'entiercement de routine pourraient facilement être mis en œuvre pour protéger les acheteurs. Dans cet article, nous proposons une solution au problème de 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 tout groupe de nœuds attaquants coopérant.

2. Opérations

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

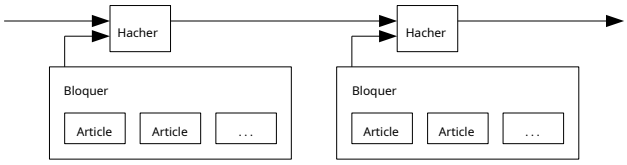


Le problème, bien sûr, est que le bénéficiaire ne peut pas vérifier que l'un des propriétaires n'a pas dépensé deux fois la pièce. Une solution courante consiste à introduire une autorité centrale de confiance, ou la menthe, qui vérifie chaque transaction pour les doubles dépenses. Après chaque transaction, la pièce doit être retournée à la Monnaie pour émettre une nouvelle pièce, et seules les pièces émises directement par la Monnaie sont dignes de confiance pour 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 le bénéficiaire de savoir que les propriétaires précédents n'ont signé aucune transaction antérieure. Pour nos besoins, la première transaction est celle qui compte, nous ne nous soucions donc pas des tentatives ultérieures de double dépense. 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 accomplir cela sans partie de confiance, les transactions doivent être annoncées publiquement [1], et nous avons besoin d'un système permettant aux participants de se mettre d'accord sur un historique unique de l'ordre dans lequel elles ont été reçues. Le bénéficiaire doit prouver qu'au moment de chaque transaction, la majorité des nœuds ont convenu qu'il s'agissait du premier reçu.

3. Serveur d'horodatage

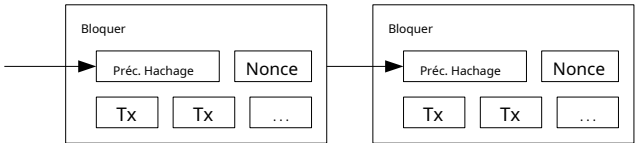
La solution que nous proposons commence par un serveur d'horodatage. Un serveur d'horodatage fonctionne en prenant un hachage d'un bloc d'éléments à horodater et en publiant largement le hachage, comme dans un journal ou une publication Usenet [2-5]. L'horodatage prouve que les données doivent avoir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



4. Preuve de travail

Pour implémenter un serveur d'horodatage distribué sur une base peer-to-peer, nous aurons besoin d'utiliser un système de preuve de travail similaire à Hashcash d'Adam Back [6], plutôt que des publications dans les journaux ou Usenet. La preuve de travail consiste à rechercher une valeur qui, lorsqu'elle est hachée, comme avec SHA-256, commence par un nombre de bits zéro. Le travail moyen requis est exponentiel en 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 qu'il satisfasse à la preuve de travail, le bloc ne peut pas être modifié sans refaire le travail. Comme les blocs ultérieurs sont enchaînés après lui, le travail pour changer le bloc inclurait de refaire tous les blocs après lui.



La preuve de travail résout également le problème de la détermination de la représentation dans la prise de décision majoritaire. Si la majorité était basée sur une adresse IP, une voix, elle pourrait être détournée 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 le plus grand effort de preuve de travail investi. Si une 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 après celui-ci, puis rattraper et dépasser le travail des nœuds honnêtes. Nous montrerons plus tard que la probabilité qu'un attaquant plus lent rattrape son retard diminue de façon exponentielle à mesure que les blocs suivants sont ajoutés.

Pour compenser l'augmentation de la vitesse du matériel 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 vite, la difficulté augmente.

5. Réseau

Les étapes pour faire fonctionner le réseau sont les suivantes :

1) Les nouvelles transactions sont diffusées à tous les nœuds.
2) Chaque nœud collecte de 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 n'ont pas déjà été 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 la chaîne la plus longue comme 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'une ou l'autre en premier. Dans ce cas, ils travaillent sur la première qu'ils ont reçue, mais sauvent l'autre branche au cas où elle s'allongerait. 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 tolèrent également les messages abandonnés. Si un nœud ne reçoit pas de bloc, il le demandera lorsqu'il recevra le prochain bloc et se rendra compte qu'il en a manqué un.

6. Incitatif

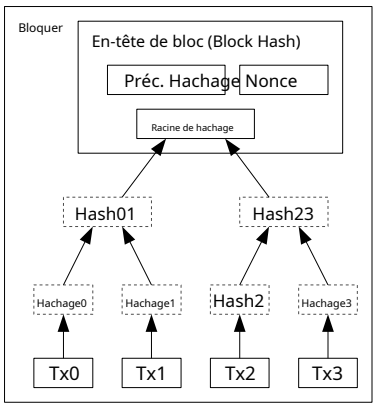
Par convention, la première transaction d'un bloc est une transaction spéciale qui démarre une nouvelle pièce appartenant au créateur du bloc. Cela incite les nœuds à prendre en charge le réseau et fournit un moyen de distribuer initialement les pièces en circulation, car il n'y a pas d'autorité centrale pour les émettre. L'ajout constant d'une quantité constante de nouvelles pièces est analogue aux mineurs d'or dépensant 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'incitatif peut également être financé 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'incitatif peut passer entièrement aux frais de transaction et être totalement exempt d'inflation.

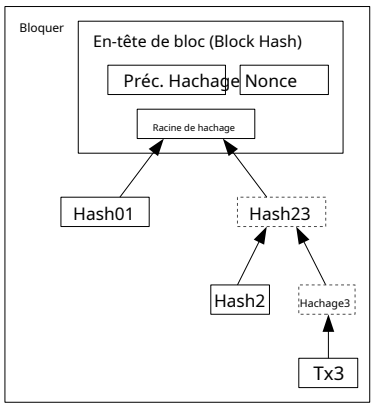
L'incitation peut aider à encourager les nœuds à rester honnêtes. Si un attaquant avide est capable d'assembler plus de puissance CPU que tous les nœuds honnêtes, il devra choisir entre l'utiliser pour frauder les gens en lui volant ses paiements, ou l'utiliser pour générer de nouvelles pièces. Il devrait trouver plus profitable de jouer selon les règles, de telles règles qui le favorisent avec plus de nouvelles pièces que tout le monde réuni, que de saper le système et la validité de sa propre richesse.

7. Récupération d'espace disque

Une fois que la dernière transaction d'une pièce est enfouie sous suffisamment de blocs, les transactions passées avant qu'elle ne soient 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 coupant les branches de l'arbre. Les hachages intérieurs n'ont pas besoin d'être stockés.



Transactions hachées dans un arbre Merkle

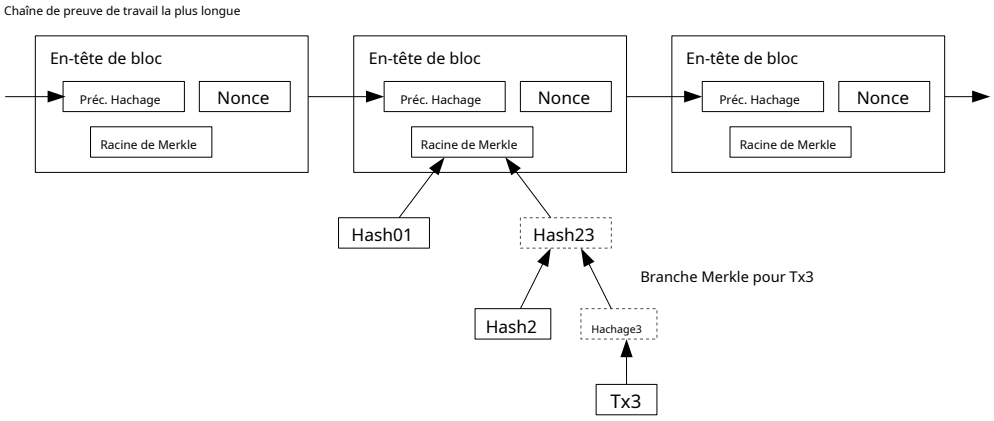


Après avoir élagué Tx0-2 du bloc

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.

8. Vérification de paiement simplifiée

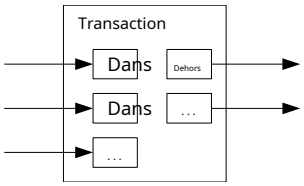
Il est possible de vérifier les paiements sans exécuter un nœud de réseau complet. Un utilisateur n'a besoin de conserver qu'une copie des en-têtes de bloc de la chaîne de preuve de travail la plus longue, qu'il peut obtenir en interrogeant les nœuds du réseau jusqu'à ce qu'il soit convaincu d'avoir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



En tant que telle, 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 pour se protéger contre cela serait d'accepter les alertes des nœuds du réseau lorsqu'ils détectent un bloc non valide, invitant le logiciel de l'utilisateur à télécharger le bloc complet et alertant les transactions 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.

9. Combiner et diviser la valeur

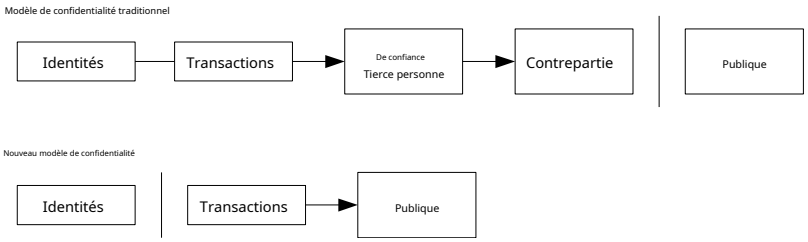
Bien qu'il soit possible de traiter les pièces individuellement, il serait difficile d'effectuer une transaction séparée pour chaque centime lors d'un transfert. Pour permettre à la valeur d'être divisée et combinée, les transactions contiennent plusieurs entrées et sorties. il y aura normalement soit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



Il convient de noter que le fan-out, où une transaction dépend de plusieurs transactions, et ces transactions dépendent de beaucoup d'autres, ne pose pas de problème ici. Il n'est jamais nécessaire d'extraire une copie autonome complète de l'historique d'une transaction.

10. Confidentialité

Le modèle bancaire traditionnel atteint un niveau de confidentialité en limitant l'accès aux informations aux parties concernées et au tiers 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 de les empêcher d'être liées à un propriétaire commun. Certains liens sont encore inévitables avec les transactions à intrants multiples, qui révèlent nécessairement que leurs intrants 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.

11. 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, tels que la création de valeur à partir de rien ou la prise d'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 réussite 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 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 le seuil de rentabilité, ou qu'un attaquant rattrape un jour la chaîne honnête, comme suit [8] :

p = probabilité qu'un nœud honnête trouve le prochain bloc$q$ =
probabilité que l'attaquant trouve le prochain bloc
q_z = probabilité que l'attaquant rattrape un jour les blocs z derrière

$$q_z = \begin{cases} 1 & si\ p \leq q \\ (q/p)^z & si\ p > q \end{cases}$$

Étant donné notre hypothèse selon laquelle $p > q$, la probabilité diminue de façon exponentielle à mesure que le nombre de blocs que l'attaquant doit rattraper augmente. Avec les chances contre lui, s'il ne fait pas un bond en avant chanceux dès le début, ses chances deviennent de plus en plus faibles 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 destinataire génère une nouvelle paire de clés et donne la clé publique à l'expéditeur peu de temps avant de signer. Cela empêche l'expéditeur de préparer une chaîne de blocs à l'avance en y travaillant en continu 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 soit ajoutée à un bloc et z les blocs ont été liés après lui. Il ne connaît pas la progression exacte de l'attaquant, mais en supposant que les blocs honnêtes aient pris le temps moyen attendu par bloc, la progression potentielle de l'attaquant sera une distribution de Poisson avec une valeur attendue :

$$\lambda = z \frac{q}{p}$$

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

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \begin{cases} (q/p)^{(z-k)} & \text{si } k \le z \\ 1 & \text{si } k > z \end{cases}$$

Réorganiser pour éviter de résumer la queue infinie de la distribution...

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{(z-k)}\right)$$

Conversion en code C...

```
# inclure <math.h>
double AttackerSuccessProbability(double q, int z) {

        double p = 1,0 - q; double lambda = z *
        (q / p); double somme = 1,0 ;

        int i, k;
        pour (k = 0; k <= z; k++) {

                double poisson = exp(-lambda); pour (i =
                1; i <= k; i++)
                        poisson *= lambda / i;
                somme -= poisson * (1 - pow(q / p, z - k));
        }
        somme de retour ;
}
```

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
```

Résoudre pour P moins de 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
```

12. Conclusion

Nous avons proposé un système pour les transactions électroniques sans s'appuyer sur la confiance. Nous avons commencé avec le cadre habituel des pièces fabriquées à partir de signatures numériques, qui offre un contrôle fort de la propriété, mais est incomplet sans un moyen d'éviter les doubles dépenses. Pour résoudre ce problème, nous avons proposé un réseau peer-to-peer utilisant une preuve de travail pour enregistrer un historique public des transactions qui devient rapidement impossible en termes de calcul pour un attaquant à modifier 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, accepter 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 sur leur extension et rejetant les blocs invalides en refusant de travailler dessus. Toutes les règles et incitations nécessaires peuvent être appliquées avec ce mécanisme de consensus.

Les références

[1] W. Dai, « b-money », http://www.weidai.com/bmoney.txt, 1998.

[2] H. Massias, XS Avila et J.-J. Quisquater, "Conception d'un service d'horodatage sécurisé avec des exigences de confiance minimales", Dans*20e Symposium sur la théorie de l'information au Benelux*, mai 1999.

[3] S. Haber, WS Stornetta, « Comment horodater un document numérique », Dans *Journal de Cryptologie*, tome 3, non 2, pages 99-111, 1991.

[4] D. Bayer, S. Haber, WS Stornetta, "Améliorer l'efficacité et la fiabilité de l'horodatage numérique", Dans *Séquences II : Méthodes en Communication, Sécurité et Informatique*, pages 329-334, 1993.

[5] S. Haber, WS Stornetta, « Noms sécurisés pour les chaînes de bits », Dans *Actes de la 4e conférence de l'ACM sur la sécurité informatique et des communications*, pages 28-35, avril 1997.

[6] A. Back, « Hashcash - une contre-mesure de déni de service », http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] RC Merkle, « Protocoles pour les cryptosystèmes à clé publique », dans *Proc. Colloque de 1980 sur la sécurité et la confidentialité*, IEEE Computer Society, pages 122-133, avril 1980.

[8] W. Feller, "Une introduction à la théorie des probabilités et ses applications," 1957.