Bitcoin : un système d'argent liquide électronique pair-à-pair

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

Traduction de Ludovic Lars

Résumé

Une version purement pair-à-pair d'argent liquide électronique permettrait aux paiements en ligne d'être envoyés directement d'une partie à l'autre sans passer par une institution financière. Les signatures numériques fournissent une partie de la solution, mais perdent leurs principaux avantages si un tiers de confiance est nécessaire pour empêcher la double dépense. 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 basées sur le hachage, formant un enregistrement qui ne peut être modifié sans reproduire la preuve de travail équivalente. La chaîne la plus longue sert non seulement de preuve du déroulement d'événements constatés, mais aussi de preuve qu'elle provient du plus grand regroupement de puissance de calcul (CPU). Tant que la majorité de la puissance de calcul est contrôlée par des nœuds qui ne coopèrent pas pour attaquer le réseau, ils génèrent la chaîne la plus longue et devancent les attaquants. Le réseau lui-même ne nécessite qu'une structure minimale. Les messages sont transmis au mieux, et les nœuds peuvent quitter et rejoindre le réseau à volonté, en acceptant la plus longue chaîne de preuves de travail comme preuve de ce qui s'est passé pendant leur absence.

1. Introduction

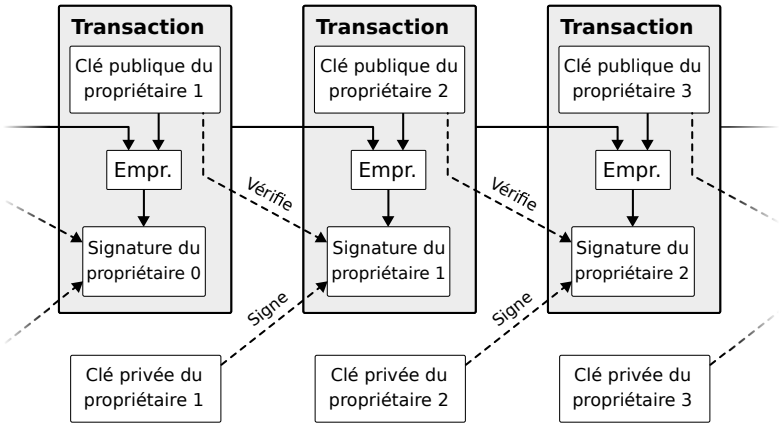
Le commerce sur Internet repose aujourd'hui presque exclusivement sur des institutions financières qui servent de tiers de confiance pour traiter les paiements électroniques. Bien que ce système fonctionne assez bien pour la plupart des transactions, il souffre toujours des faiblesses inhérentes à son modèle basé sur la confiance. L'irréversibilité totale des transactions n'est pas vraiment possible, car les institutions financières ne peuvent éviter la médiation des conflits. Le coût de cette médiation augmente le coût des transactions, limitant en pratique la taille minimale d'une transaction et supprimant la possibilité d'effectuer de petites transactions occasionnelles, et l'incapacité à effectuer des paiements irréversibles pour des services irréversibles engendre un coût encore plus important. Avec la possibilité de réversibilité, le besoin de confiance s'agrandit. Les commerçants doivent se méfier de leurs clients et les harceler pour obtenir plus d'informations que nécessaire. Un certain pourcentage de fraude est accepté comme inévitable. Ces coûts et ces incertitudes liées au paiement peuvent être évités en personne par l'utilisation d'une monnaie physique, mais il n'existe aucun mécanisme permettant d'effectuer des paiements sur un canal de communication sans tiers de confiance.

Ce dont nous avons besoin, c'est d'un système de paiement électronique basé sur des preuves cryptographiques plutôt que sur la confiance, qui permettrait à deux parties volontaires de réaliser directement des transactions entre elles sans avoir recours à un tiers de confiance. Les vendeurs seraient protégés de la fraude par l'impossibilité informatique d'inverser les transactions, et les acheteurs pourraient être facilement protégés par la mise en œuvre de mécanismes de dépôt fiduciaire routiniers. Dans ce document, nous proposons une solution au problème de la double dépense en utilisant un serveur d'horodatage distribué de 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 de calcul qu'un groupe de nœuds qui coopéreraient pour réaliser une attaque.

2. Transactions

Nous définissons une pièce de monnaie électronique comme une chaîne de signatures numériques. Chaque propriétaire transfère la pièce au suivant en signant numériquement l'empreinte 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.

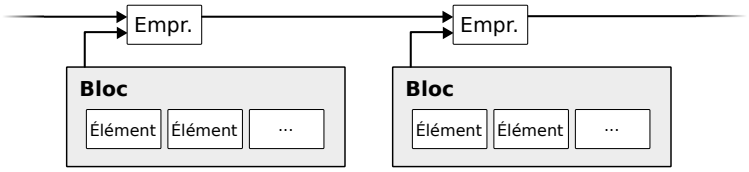


Le problème, bien sûr, est que le bénéficiaire ne peut pas vérifier que l'un des propriétaires n'a pas effectué de double dépense avec la pièce. Une solution courante consiste à introduire une autorité centrale de confiance, ou monnaierie, qui vérifie chaque transaction pour s'assurer qu'il n'y a pas de double dépense. Après chaque transaction, la pièce doit être renvoyée à la monnaierie pour que celle-ci émette une nouvelle pièce, et seules les pièces émises directement par la monnaierie peuvent être considérées comme n'ayant pas été dépensées deux fois. Le problème de cette solution est que le sort de l'ensemble du système monétaire dépend de l'entreprise qui gère la monnaierie, chaque transaction devant passer par elle, tout comme dans le cas d'une banque.

Nous avons besoin d'un moyen pour le bénéficiaire de savoir que les propriétaires précédents n'ont pas signé de transactions antérieures. À cette fin, c'est la transaction la plus ancienne qui compte, et 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 une monnaierie, la monnaierie est au courant de toutes les transactions et décide de celles qui arrivent en premier. Pour y parvenir sans tiers 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 a besoin de la preuve qu'au moment de chaque transaction, la majorité des nœuds convenait que cette transaction était la première reçue.

3. Serveur d'horodatage

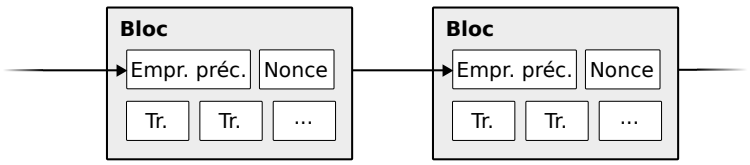
La solution que nous proposons se base sur un serveur d'horodatage. Un serveur d'horodatage fonctionne en calculant l'empreinte d'un bloc d'éléments à horodater et en publiant cette empreinte à grande échelle, par exemple dans un journal ou dans un message sur Usenet [2-5]. L'horodatage prouve que les données existaient à ce moment-là, de toute évidence, pour pouvoir être intégrées au hachage. Chaque horodatage inclut l'horodatage précédent dans son empreinte, formant ainsi une chaîne, au sein de laquelle chaque horodatage supplémentaire renforce le précédent.



4. Preuve de travail

Pour mettre en œuvre un serveur d'horodatage distribué sur une base pair-à-pair, nous avons besoin d'utiliser un système de preuve de travail similaire au système Hashcash [6] développé par Adam Back, plutôt que des articles de journaux ou des messages sur Usenet. La preuve de travail consiste à rechercher une valeur qui, une fois hachée, avec SHA-256 par exemple, commence par un nombre de bits nuls. Le travail moyen requis évolue de façon exponentielle avec le nombre de bits nuls exigés et peut être vérifié par l'exécution d'un seul hachage.

Pour notre réseau d'horodatage, nous mettons en œuvre la preuve de travail en incrémentant un nonce dans le bloc jusqu'à ce qu'une valeur soit trouvée qui donne à l'empreinte du bloc les bits nuls exigés. Une fois que l'effort de calcul nécessaire pour satisfaire la preuve de travail a été effectué, le bloc ne peut pas être modifié sans reproduire le travail. Comme les blocs suivants sont enchaînés après lui, le travail pour changer le bloc impliquerait de refaire celui de 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 le principe de vote par adresse IP (une adresse IP, une voix), elle pourrait être détournée par toute personne capable de s'octroyer de nombreuses adresses IP. La preuve de travail est essentiellement basée sur la puissance de calcul : un processeur, une voix. La décision majoritaire est représentée par la chaîne la plus longue, sur laquelle le plus grand effort de preuve de travail a été investi. Si la majorité de la puissance de la calcul est contrôlée par des nœuds honnêtes, la chaîne honnête se développera

plus rapidement et dépassera toutes les chaînes concurrentes. Pour modifier un ancien bloc, un attaquant devrait reproduire la preuve de travail de ce bloc et de tous les blocs suivants, puis rattraper et dépasser le travail des nœuds honnêtes. Nous montrerons plus bas que la probabilité qu'un attaquant plus lent rattrape son retard diminue de façon exponentielle à mesure que les nouveaux blocs sont ajoutés.

Afin de compenser l'augmentation de la vitesse du matériel et la variation de l'intérêt des nœuds actifs au fil du temps, la difficulté de la preuve de travail est déterminée par une moyenne mobile visant un nombre moyen de blocs par heure. Si ces blocs sont générés trop rapidement, la difficulté augmente.

5. Réseau

Les étapes nécessaires au fonctionnement du réseau sont les suivantes :

1) Les nouvelles transactions sont transmises à tous les nœuds.
2) Chaque nœud rassemble les nouvelles transactions dans un bloc.
3) Chaque nœud travaille pour trouver une preuve de travail difficile pour son bloc.
4) Lorsqu'un nœud trouve une preuve de travail, il transmet 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 de la chaîne et en utilisant l'empreinte du bloc accepté en tant qu'empreinte précédente.

Les nœuds considèrent toujours que la chaîne la plus longue est la chaîne correcte, et continuent à travailler pour la prolonger. Si deux nœuds transmettent simultanément des versions différentes du bloc suivant, certains nœuds peuvent recevoir l'une ou l'autre version en premier. Dans ce cas, ils travaillent sur la première version qu'ils ont reçue, mais conservent l'autre branche au cas où elle deviendrait plus longue. L'égalité est rompue lorsque la preuve de travail suivante est trouvée et qu'une branche devient plus longue ; les nœuds qui travaillaient sur l'autre branche passent alors sur la chaîne la plus longue.

La transmission de nouvelles transactions n'a pas besoin d'atteindre tous les nœuds. Du moment que les transactions atteignent de nombreux nœuds, elles entreront dans un bloc sous peu. La transmission des blocs est également tolérante à la perte de messages. Si un nœud ne reçoit pas un bloc, il le demandera lorsqu'il recevra le bloc suivant et se rendra compte qu'il en a manqué un.

6. Incitation

Par convention, la première transaction d'un bloc est une transaction spéciale qui crée une nouvelle pièce appartenant au créateur du bloc. Cela incite les nœuds à soutenir le réseau et permet la mise en circulation initiale des pièces, puisqu'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 dans la circulation. Dans notre cas, ce sont du temps de calcul et de l'électricité qui sont dépensés.

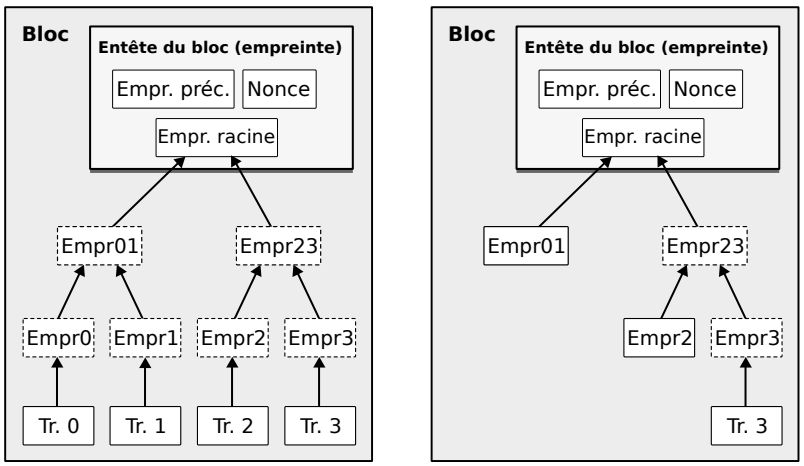
L'incitation peut également être financée par les frais de transaction. Si la valeur de sortie d'une transaction est inférieure à sa valeur d'entrée, la différence correspond à une commission qui est ajoutée à la valeur d'incitation du bloc contenant la transaction. Une fois qu'un nombre

prédéterminé de pièces a été mis en circulation, l'incitation peut être entièrement financée par les frais de transaction et ne plus requérir aucune inflation.

L'incitation peut contribuer à encourager les nœuds à rester honnêtes. Si un attaquant cupide est capable de réunir plus de puissance de calcul que l'ensemble des nœuds honnêtes, il aura à choisir entre l'utiliser pour escroquer des gens en leur récupérant ses paiements, ou l'utiliser pour générer de nouvelles pièces. Il devrait trouver plus rentable de respecter les règles du jeu, celles-ci lui permettant d'obtenir plus de nouvelles pièces que tous les autres réunis, plutôt que de saper le système et la validité de sa propre richesse.

7. Économie d'espace disque

Une fois que la dernière transaction concernant une pièce est enfouie sous un nombre suffisant de blocs, les transactions dépensées avant elle peuvent être abandonnées pour économiser de l'espace disque. Pour faciliter cette opération sans altérer l'empreinte du bloc, les transactions sont hachées dans un arbre de Merkle [7][2][5], dont seule la racine est comprise dans l'empreinte du bloc. Les anciens blocs peuvent ensuite être compactés en élaguant les branches de l'arbre. Les empreintes intermédiaires n'ont pas besoin d'être stockées.



Transactions hachées dans un arbre de Merkle

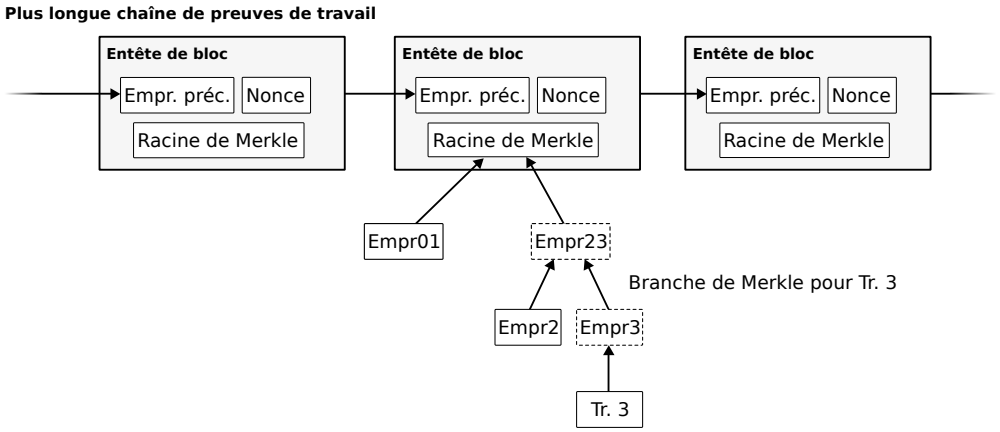
Après l'élagage des transactions 0 à 2 du bloc

Un entête de bloc sans transaction pèse environ 80 octets. Si nous supposons que les blocs sont générés toutes les 10 minutes, cela représente 80 octets * 6 * 24 * 365 = 4,2 Mo par an. Les systèmes informatiques étant généralement vendus avec 2 Go de RAM en 2008, et la loi de Moore prédisant une croissance actuelle de 1,2 Go par an, le stockage ne devrait pas poser de problème même si les entêtes de blocs doivent être conservés en mémoire.

8. Vérification de paiement simplifiée

Il est possible de vérifier les paiements sans faire fonctionner un nœud complet du réseau. Un utilisateur a seulement besoin de conserver une copie des entêtes des blocs de la plus longue chaîne de preuves de travail, qu'il peut obtenir en interrogeant les nœuds du réseau jusqu'à ce qu'il soit convaincu qu'il possède la plus longue chaîne, et obtenir la branche de Merkle liant la transaction au bloc dans lequel elle est horodatée. Il ne peut pas vérifier la transaction par

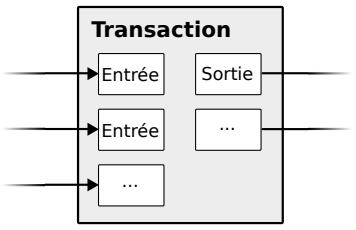
lui-même, mais en la reliant à un endroit de la chaîne, il peut voir qu'un nœud du réseau l'a acceptée, et les blocs ajoutés après le confirment.



De ce fait, la vérification est fiable tant que les 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 des transactions forgées par l'attaquant aussi longtemps que celui-ci maîtrise le réseau. Une stratégie pour se protéger serait d'accepter les alertes des nœuds du réseau lorsqu'ils détectent un bloc invalide, invitant le logiciel de l'utilisateur à télécharger le bloc complet et les transactions signalées pour confirmer l'incohérence. Les entreprises qui reçoivent fréquemment des paiements voudront probablement toujours faire fonctionner leurs propres nœuds afin d'obtenir une sécurité plus indépendante et une vérification plus rapide.

9. Combinaison et séparation de valeur

Bien qu'il soit possible de traiter les pièces séparément, il serait peu commode d'effectuer une transaction distincte pour chaque centime dans un transfert. Pour permettre aux valeurs d'être séparées et combinées, les transactions contiennent des entrées et des sorties multiples. Normalement, il y a soit une entrée unique provenant d'une transaction antérieure plus importante, soit des entrées multiples combinant des montants plus petits, et au maximum deux sorties : une pour le paiement, et une pour le retour de la monnaie, le cas échéant, au payeur.

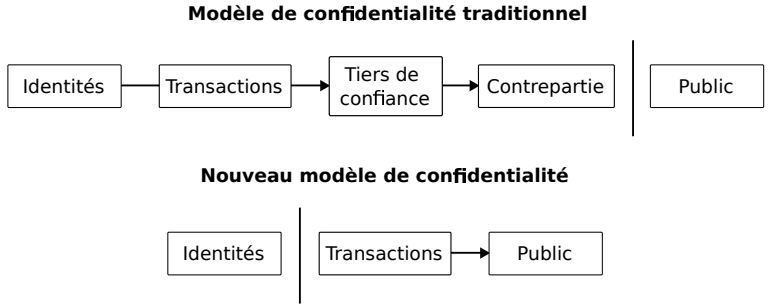


Il convient de noter que la dispersion, lorsqu'une transaction dépend de plusieurs transactions,

et que ces transactions dépendent de beaucoup d'autres, n'est pas un problème dans notre cas. 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 certain 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 ne dispose pas d'informations reliant la transaction à qui que ce soit.



Comme pare-feu supplémentaire, une nouvelle paire de clés devrait être utilisée pour chaque transaction afin de les empêcher d'être liées à un propriétaire commun. Certains liens sont toujours inévitables avec les transactions à entrées multiples, qui révèlent nécessairement que leurs entrées appartiennent au même propriétaire. Le risque est que si le propriétaire d'une clé est révélé, la liaison pourrait révéler d'autres transactions qui lui appartiennent.

11. Calculs

Nous considérons le scénario d'un attaquant qui tente de générer une chaîne alternative plus rapidement que la chaîne honnête. Même en cas de réussite, cela n'expose pas le système à des modifications arbitraires, comme la création de valeur *ex nihilo* ou l'appropriation d'argent n'ayant jamais appartenu à l'attaquant. Les nœuds ne vont pas accepter une transaction invalide comme paiement, et les nœuds honnêtes n'accepteront jamais un bloc les contenant. Un attaquant peut seulement essayer de modifier l'une de ses propres transactions afin de récupérer l'argent qu'il a récemment dépensé.

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

La probabilité qu'un attaquant rattrape un retard donné est analogue au problème de la ruine du joueur. Supposons qu'un joueur avec un crédit illimité commence avec un déficit et joue un nombre potentiellement infini d'essais pour essayer d'atteindre le seuil de rentabilité. Nous pouvons calculer la probabilité qu'il atteigne un jour le seuil de rentabilité, c'est-à-dire la probabilité qu'un attaquant rattrape un jour 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

q_z = probabilité que l'attaquant rattrape un jour un retard de z blocs

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

Étant donnée notre hypothèse que $p > q$, la probabilité diminue de façon exponentielle à mesure que le nombre de blocs que l'attaquant doit rattraper augmente. Avec les probabilités contre lui, s'il n'obtient pas une avancée chanceuse très tôt, ses chances deviennent de plus en plus faibles à mesure qu'il prend du retard.

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

Le destinataire génère une nouvelle paire de clés et donne la clé publique au payeur peu avant de signer. Cela empêche le payeur de préparer une chaîne de blocs à l'avance en travaillant dessus continuellement jusqu'à avoir la chance d'obtenir une avance suffisante, et d'effectuer la transaction à ce moment-là. Une fois la transaction envoyée, le payeur 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 ajoutés après ce bloc. 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 pour être créés, la progression potentielle de l'attaquant correspondra à une distribution de Poisson ayant pour espérance :

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

Afin d'obtenir la probabilité que l'attaquant puisse encore rattraper son retard, nous multiplions la fonction de 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 :

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

En réarrangeant pour éviter de sommer à l'infini la queue de la distribution...

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

Converti en code C :

```
#include
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 effectuant quelques calculs, on peut voir que la probabilité diminue 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
```

En résolvant 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
```

12. Conclusion

Nous avons proposé un système de transactions électroniques qui ne repose pas sur la confiance. Nous avons débuté par le cadre habituel des pièces fabriquées à partir de signatures numériques, qui fournit un contrôle fort de la propriété, mais qui reste incomplet sans moyen d'empêcher la double dépense. Pour résoudre ce problème, nous avons proposé un réseau pair-à-pair utilisant la preuve de travail pour enregistrer l'historique public des transactions qui devient rapidement impossible à modifier par un attaquant si les nœuds honnêtes contrôlent la majorité de la puissance de calcul. Le réseau est robuste par sa simplicité non structurée. Les nœuds travaillent tous en même temps avec peu de coordination. Ils n'ont pas besoin d'être identifiés, puisque les messages ne sont pas acheminés vers un endroit particulier et doivent seulement être délivrés au mieux. Les nœuds peuvent quitter et rejoindre le réseau à volonté, en acceptant la chaîne de preuves de travail comme preuve de ce qui s'est passé pendant leur absence. Ils votent avec leur puissance de calcul, exprimant leur acceptation des blocs valides en travaillant à leur extension et rejetant les blocs invalides en refusant d'y travailler. Toutes les règles et incitations nécessaires peuvent être appliquées grâce à ce mécanisme de consensus.

Références

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

[2] H. Massias, X.S. Avila et J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements", in *20th Symposium on Information Theory in the Benelux*, mai 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document", in *Journal of Cryptology*, volume 3, numéro 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, avril 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, avril 1980.

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