Bitcoin : Système de Monnaie Electronique en Pair-à-Pair.

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

Document original: www.bitcoin.org/bitcoin.pdf

Traducteurs: Benkebab, Grondilu, Mackila
Forum: http://bitcointalk.org/?topic=2465.0

Résumé. Un système de monnaie électronique entièrement en pair-à-pair permettrait d'effectuer des paiements en ligne directement d'un tiers à un autre sans passer par une institution financière. Les signatures numériques offrent une telle solution, mais perdent leur intérêt dès lors qu'un tiers de confiance est requis pour empêcher le double paiement. Nous proposons une solution au problème du double paiement en utilisant un réseau pair-à-pair. Le réseau horodate les transactions à l'aide d'une fonction de hachage qui les traduit en une chaîne continue de preuves de travail (des empreintes[1]), formant un enregistrement qui ne peut être modifié sans ré-effectuer la preuve de travail. La plus longue chaîne (d'empreintes) sert non seulement de preuve du déroulement des évènements constatés, mais également de preuve qu'elle provient du plus grand regroupement de puissance de calcul. Aussi longtemps que la majorité de la puissance de calcul (CPU) est contrôlée par des nœuds qui ne coopèrent pas pour attaquer le réseau, ils généreront la plus longue chaîne et surpasseront les attaquants. Le réseau en lui-même ne requiert qu'une structure réduite. Les messages sont diffusés au mieux, et les nœuds peuvent quitter ou rejoindre le réseau à leur gré, en acceptant à leur retour la chaîne de preuve de travail la plus longue comme preuve de ce qui s'est déroulé pendant leur absence.

1. Introduction

Le commerce sur Internet dépend aujourd'hui presque exclusivement d'institutions financières qui servent de tiers de confiance pour traiter les paiements électroniques. Bien que ce système fonctionne plutôt bien pour la plupart des transactions, il écope toujours des faiblesses inhérentes à son modèle basé sur la confiance. Les transactions totalement irréversibles n'y sont pas vraiment possibles, puisque les institutions financières doivent gérer la médiation de conflits. Le coût de cette médiation augmente les coûts des transactions, limitant en pratique la taille minimale d'une transaction et

---

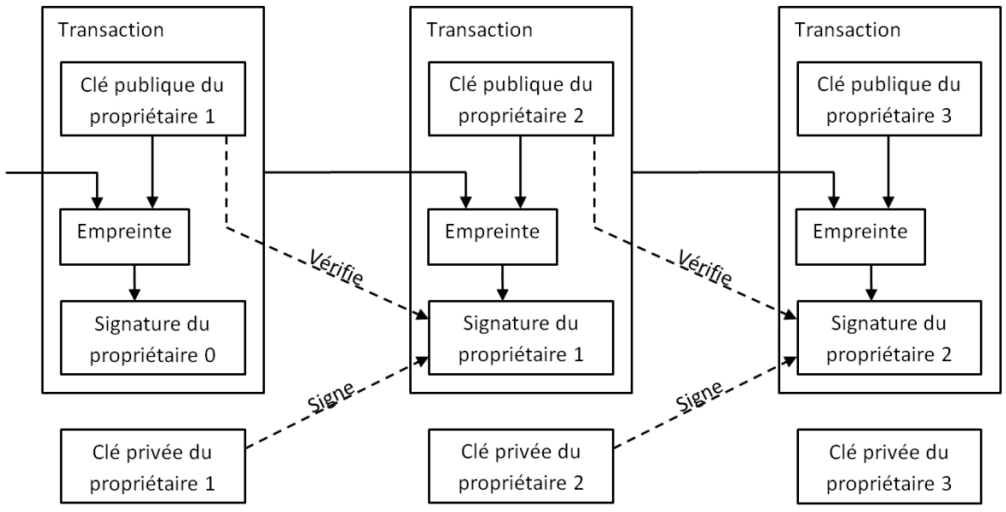
[1] N.d.t.: "en cryptographie, une empreinte est le résultat d'une fonction de hachage qui permet d'identifier la donnée initiale" (cf. http://fr.wikipedia.org/wiki/Empreinte)

empêchant la possibilité d'avoir des petites transactions peu coûteuses. L'impossibilité d'avoir des paiements non réversibles pour des services non réversibles engendre un coût encore plus important. Avec la possibilité d'inverser les transactions, le besoin de confiance augmente. Les marchands doivent se méfier de leurs clients, en les harcelant pour obtenir plus d'informations que nécessaire. Une certaine part de fraudes est acceptée comme inévitables. Tous ces coûts et incertitudes de paiement peuvent être évités par l'utilisation d'une monnaie physique, mais aucun mécanisme n'existe pour réaliser des paiements à travers un système de communication sans avoir recours à un tiers de confiance.

Ce dont nous avons besoin, c'est d'un système de paiement électronique basé sur des preuves cryptographiques au lieu d'un modèle basé sur la confiance, qui permettrait à deux parties qui le souhaitent de réaliser des transactions directement entre elles sans avoir recours à un tiers de confiance. Les transactions qui sont informatiquement impossibles à annuler protégeraient les vendeurs de fraudes éventuelles, et un système de compte séquestre pourrait facilement être implémenté pour protéger les acheteurs. Dans ce document, nous proposons une solution au problème de double-dépense en utilisant un serveur horodaté distribué en pair-à-pair pour générer des preuves informatiques de l'ordre chronologique des transactions. Le système est sécurisé tant que des nœuds honnêtes contrôlent ensemble plus de puissance de calcul qu'un groupe de nœuds qui coopèrerait pour réaliser une attaque.

2. Transactions

Nous définissons une pièce électronique comme une chaîne de signatures numériques. Tout propriétaire transfère cette pièce à un autre en signant numériquement une empreinte de la précédente transaction ainsi que la clé publique du nouveau propriétaire et les ajoute à la fin de la pièce. Tout bénéficiaire peut examiner les signatures pour vérifier la chaîne de propriété.

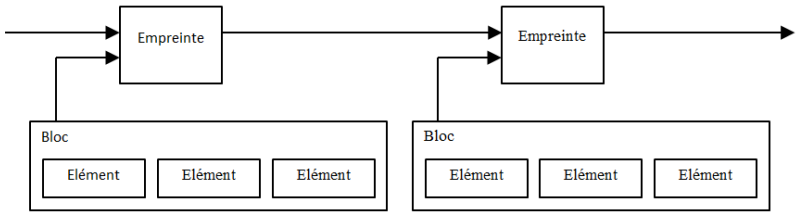


Le problème bien sûr est que le bénéficiaire ne peut vérifier qu'un des précédents propriétaire n'a pas fait de "double dépense" avec la pièce. Une solution courante est d'introduire une autorité centrale de confiance, ou hôtel des Monnaies, qui vérifierait chaque transaction pour éviter la "double dépense". Après chaque transaction, les pièces doivent être retournées à l'hôtel des Monnaies qui en crée une nouvelle, et seules les pièces issues directement de l'hôtel des Monnaies sont considérées comme n'ayant pas été dépensées deux fois. Le problème de cette solution est que le destin de tout le système monétaire repose sur l'entreprise qui dirige l'hôtel des Monnaies, et que chaque transaction doit passer par eux, comme une banque.

Nous avons besoin d'une méthode pour que le bénéficiaire puisse savoir si les précédents propriétaires n'ont pas signé de transactions précédentes. Pour cela, la transaction la plus ancienne doit être celle qui compte, nous n'avons pas à nous soucier des tentatives ultérieures pour dépenser la pièce en double. La seule manière de confirmer l'absence d'une transaction précédente est d'être au courant de toutes les transactions. Dans le modèle basé sur un hôtel des Monnaies, ce dernier était au courant de toutes les transactions et décidait donc laquelle arrivait en premier. Pour faire de même sans tierce partie, les transactions doivent être annoncées publiquement [1], et nous avons besoin d'un système pour que tous les participants se mettent d'accord sur un seul historique de l'ordre dans laquelle les transactions sont reçues. Le bénéficiaire a besoin d'une preuve qu'à chaque temps d'une transaction, la majorité des noeuds soient d'accord sur le fait qu'elle était la première reçue.

3. Serveur d'Horodatage

La base de la solution proposée est un serveur d'horodatage. Un serveur d'horodatage prend l'empreinte d'un ensemble d'éléments à horodater et publie cette empreinte, à la façon d'une annonce dans un journal ou d'un message sur un forum Usenet [2-5]. L'horodatage prouve que les données ont existé, afin d'être prises en compte dans l'empreinte. Chaque horodatage inclue l'horodatage précédent dans son empreinte, formant une chaîne dont chaque nouvel élément vient confirmer les précédents.

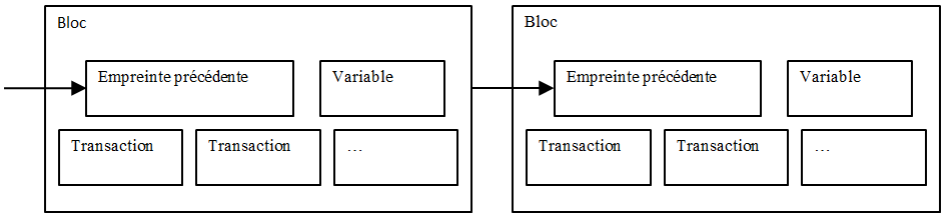


4. Preuve de travail

Pour implémenter un serveur d'horodatage distribué sur un réseau pair-à-pair, il faut

utiliser un système basé sur des preuves de travail tel que celui du système Hashcash de Adam Back [6], plutôt qu'un journal ou un message sur un forum Usenet. La preuve de travail nécessite de rechercher une valeur telle que son empreinte, calculée par exemple en utilisant SHA-256, débute par un certain nombre de bits à 0. Le travail requis pour est exponentiellement fonction du nombre de bits à 0 exigés, et peut être validé en effectuant un seul calcul d'empreinte.

Pour notre réseau d'horodatage, nous implémentons la preuve de travail en incrémentant une variable dans le bloc jusqu'à ce qu'une valeur donnant une empreinte ayant suffisamment de bits à 0 soit trouvée. Une fois que l'effort de calcul nécessaire à l'obtention de la preuve de travail a été effectué, il n'est plus possible de modifier le bloc sans refaire cet effort de calcul. Comme de nouveaux blocs sont chaînés à sa suite, l'effort de calcul nécessaire pour modifier un bloc inclue tout l'effort de calcul nécessaire pour modifier tous les blocs suivants.



La preuve de travail résout le problème du choix de la représentativité du vote. Si la majorité était basée sur des voix allouées par adresse IP, le vote pourrait être perverti par quelqu'un capable de s'octroyer beaucoup d'adresses. La preuve de travail est essentiellement basée sur la puissance de calcul (un processeur, une voix). La décision de la majorité est représentée par la chaîne la plus longue, celle qui a nécessité le plus de calculs de preuve de travail. Si la majorité de la puissance de calcul du réseau est contrôlée par des noeuds honnêtes, la chaîne légitime progresse le plus rapidement et distance les chaînes concurrentes. Afin de modifier un ancien bloc, un attaquant devrait recalculer les preuves de travail du bloc modifié et de tous les blocs suivants, pour rattraper et dépasser le travail fournit par les noeuds honnêtes. Nous démontrerons par la suite que la probabilité qu'un attaquant disposant de moins de puissance de calcul puisse rattraper diminue exponentiellement à chaque nouveau bloc ajouté.

Afin de compenser l'amélioration de la puissance de calcul du matériel et l'intérêt changeant de faire fonctionner des noeuds du réseau, la difficulté de la preuve de travail est déterminée par une moyenne de nombre de blocs à trouver par heure. Si ces blocs sont générés trop rapidement, la difficulté augmente.

5. Réseau

Les étapes mises en oeuvre pour faire fonctionner le réseau sont les suivantes :

1. **Les nouvelles transactions sont diffusées à tous les noeuds.**
2. **Chaque noeud regroupe les nouvelles transactions dans un bloc.**
3. **Chaque noeud travaille à la résolution de la preuve de travail sur son bloc.**
4. **Quand un noeud trouve une preuve de travail, il diffuse ce bloc à tous les noeuds.**
5. **Les noeuds n'acceptent le bloc que si toutes les transactions qu'il contient sont valides et n'ont pas déjà été dépensées.**
6. **Les noeuds expriment l'acceptation du bloc en travaillant sur un nouveau bloc dans la chaîne, ce nouveau bloc ayant comme empreinte précédente celle du bloc accepté.**

Les noeuds considèrent toujours la chaîne la plus longue comme étant la chaîne légitime, et travaillent à étendre celle-ci. Si deux noeuds diffusent deux versions différentes du bloc suivant simultanément, certains des noeuds vont recevoir l'une ou l'autre en premier. Dans cette situation, chacun travaille sur le bloc reçu en premier, mais conserve l'autre branche au cas ou celle-ci devienne la plus longue. Cette liaison sera rompue quand la preuve de travail suivante sera trouvée et qu'une branche deviendra plus longue que l'autre ; les noeuds qui travaillaient alors sur l'autre branche changeront pour la plus longue.

Les diffusions de nouvelles transactions n'ont pas besoin d'atteindre tous les noeuds. A partir du moment ou elles atteignent suffisamment de noeuds, elle se retrouveront dans un bloc en peu de temps. Les diffusions des blocs tolèrent la perte de messages. Si un noeud ne reçoit pas un bloc, il le demandera lors de la réception du bloc suivant, lorsqu'il réalisera qu'il lui en manque 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 noeuds à participer au réseau, et permet la distribution initiale de la monnaie, puisqu'il n'y a pas d'autorité centralisée pour le faire. Cet ajout régulier d'une quantité constante de monnaie se rapproche de l'effort fourni par des mineurs pour ajouter de l'or en circulation. Dans notre cas, l'effort se compose de puissance de calcul et de temps.

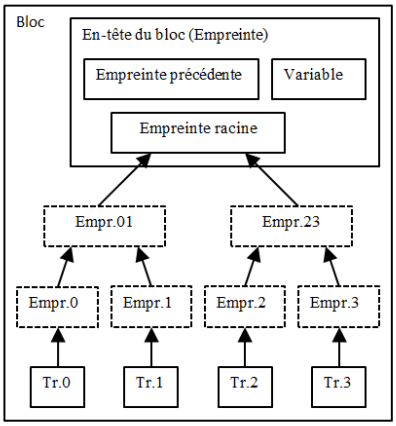
L'incitation peut aussi être financée par des frais de transaction. Si la valeur de sortie d'une transaction est inférieure à sa valeur d'entrée, la différence est correspond aux frais de transaction, qui sont ajoutés à la valeur d'incitation du bloc contenant cette transaction. Une fois que la quantité prédéterminée de monnaie sera entrée en circulation, l'incitation pourra passer sur un financement entièrement basé sur les frais de transaction, et ne provoquer aucune inflation.

L'incitation peut encourager les noeuds à rester honnêtes. Si un attaquant cupide a les moyens d'obtenir plus de puissance de calcul que l'ensemble des noeuds honnêtes, il peut choisir entre arnaquer les gens en récupérant ses paiements, ou utiliser sa puissance pour générer de la nouvelle monnaie. Il doit trouver plus

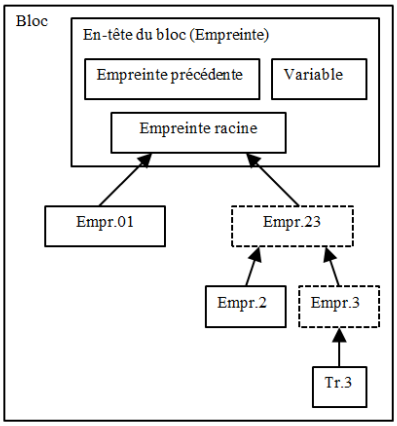
intéressant de jouer le jeu, ce qui le favorise avec plus de nouvelle monnaie que l'ensemble des autres noeuds, plutôt que de saper le système et la valeur de sa propre richesse.

7. Économiser l'Espace Disque

Une fois que la dernière transaction concernant une pièce est enfouie sous assez de blocs, les transactions passées peuvent être supprimées pour économiser de l'espace disque. Pour permettre cela sans invalider l'empreinte du bloc, les transactions sont résumées dans un arbre de Merkle [7][2][5], dont seule la racine est comprise dans l'empreinte du bloc. Les anciens blocs peuvent être compressés en coupant des branches de l'arbre. Les empreintes intermédiaires n'ont pas besoin d'être stockées.



Transactions résumées dans un arbre de Merkle

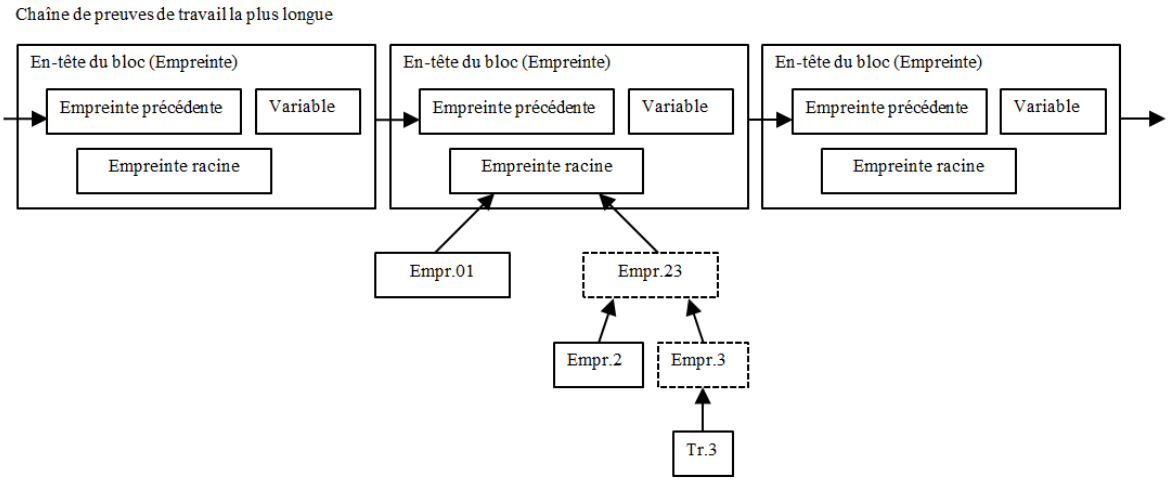


Après l'élagage des transactions 0 à 2

Un en-tête 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 = 4.2Mo par an. En 2008, les systèmes informatiques sont vendus avec en moyenne 2Go de capacité de mémoire vive, et, la Loi de Moore prédisant une croissance de 1.2Go par an, le stockage ne devrait donc pas poser de problème, même si l'ensemble des en-têtes de bloc devait être conservés en mémoire vive.

8. Vérification de Paiement Simplifié

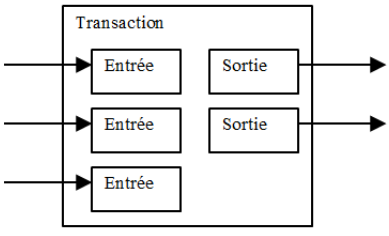
Il est possible de vérifier des paiements sans faire fonctionner un noeud complet du réseau. Un utilisateur n'a besoin de conserver qu'une copie des en-têtes de la chaîne de preuves la plus longue, ce qu'il peut obtenir à travers des requêtes sur les noeuds du réseau jusqu'à ce qu'il soit convaincu d'avoir la plus longue chaîne, puis en récupérant la branche de l'arbre de Merkle liant la transaction avec le bloc dans lequel elle est horodatée. L'utilisateur ne peut pas vérifier la transaction lui même, mais en la liant à sa place dans la chaîne, il peut voir qu'un noeud du réseau l'a acceptée, et les blocs suivants confirment davantage l'acceptation du réseau.



En tant que telle, la vérification est fiable tant que des noeuds honnêtes contrôlent le réseau, mais est plus vulnérable si le réseau est compromis par un attaquant disposant de plus de puissance de calcul. Bien que les noeuds du réseau puissent vérifier les transactions eux-mêmes, la méthode simplifiée peut être dupée par des transactions forgées par l'attaquant, tant que celui-ci a les moyens de dépasser la puissance de calcul du réseau. Une stratégie pour se protéger d'une telle attaque pourrait être de recevoir des alertes des noeuds du réseau lorsque ceux ci détectent un bloc invalide, demandant au logiciel de l'utilisateur de télécharger le bloc complet et les transactions suspectes pour vérifier l'incohérence. Les entreprises qui reçoivent fréquemment des paiements auront certainement intérêt à faire fonctionner leur propres noeuds afin d'obtenir une sécurité plus indépendante et des vérifications plus rapides.

9. Combinaison et Fractionnement de Valeur

Bien qu'il soit possible de traiter les pièces séparément, il serait peu pratique de générer une transaction différente pour chaque centime lors d'un transfert. Afin de permettre la combinaison et le fractionnement de la monnaie, les transactions comprennent de multiples entrées et sorties. Normalement, il y a soit une seule entrée depuis une grosse transaction précédente, ou plusieurs entrées combinant des montants plus faibles, et au maximum deux sorties : une pour le paiement, et l'autre pour renvoyer le change, s'il existe, à l'émetteur.

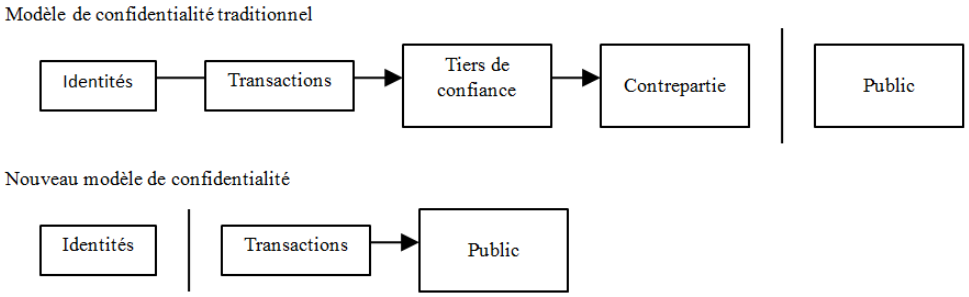


Il faut noter que la dispersion, lorsqu'une transaction dépend de plusieurs

transactions, et que ces transactions dépendent elles-mêmes de beaucoup plus de transactions, n'est pas un problème. Il n'y a jamais besoin de récupérer l'historique complet d'une transaction.

10. Confidentialité

Le système bancaire classique garantit un certain niveau de confidentialité en limitant l'accès aux informations aux parties concernées et au tiers de confiance. La nécessité de publier toutes les transactions exclue cette méthode, mais la confidentialité peut être obtenue en interrompant la circulation de l'information à un autre niveau : en gardant les clés publiques anonymes. Il est possible de voir que quelqu'un envoie un certain montant à quelqu'un d'autre, mais sans aucun lien avec des personnes. Ceci est similaire au niveau d'information disponible sur les marchés d'échange, ou la date et le montant de chacun des échanges, le "cours", est publique, mais sans révéler l'identité des parties.



Comme barrière supplémentaire, une nouvelle paire de clés peut être utilisée pour chaque transaction, pour éviter d'être reliées à un propriétaire commun. Une certaine relation est cependant inévitable avec les transactions multi-entrées, qui révèlent nécessairement que leurs entrées étaient possédées par le même propriétaire. L'évènement redouté étant que si le propriétaire d'une des clés est révélé, les liaisons permettent la révélation des autres transactions du même propriétaire.

11. Calculs

Considérons le cas d'un attaquant essayant de générer une chaîne alternative plus rapidement que la chaîne légitime. Même en cas de réussite, cela ne rendrait pas le système vulnérable à des modifications arbitraires, telles que la création monétaire à partir de rien, ou l'appropriation d'argent qui n'a jamais appartenu à l'attaquant. Les noeuds n'acceptent pas de transactions invalides comme paiement, et les noeuds honnêtes n'accepteront jamais un bloc contenant une de ces transactions. Un attaquant ne peut que modifier une de ses propres transactions afin de récupérer de l'argent qu'il vient de dépenser.

La course entre la chaîne légitime et la chaîne de l'attaquant peut être caractérisée comme une marche aléatoire binaire. L'évènement succès est

l'allongement de la chaîne légitime, augmentant son avance de +1, et l'évènement échec est l'allongement de la chaîne de l'attaquant, réduisant son retard de -1.

La probabilité qu'un attaquant rattrape son retard est analogue au problème de ruine du joueur. Imaginons un joueur ayant des crédits illimités, démarrant en négatif, et pouvant jouer un nombre infini de parties pour tenter d'atteindre le seuil de rentabilité. La probabilité qu'il y arrive, ou qu'un attaquant réussisse à rattraper la chaîne légitime, se calcule comme ceci [8] :

p = **probabilité qu'un noeud honnête trouve le prochain bloc**

q = **probabilité que l'attaquant trouve le prochain bloc**

q_z = **probabilité que l'attaquant réussisse à rattraper la chaîne avec** z **blocs de retard**

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

Etant donnée notre hypothèse $p > q$**, la probabilité diminue exponentiellement en fonction du nombre de bloc que l'attaquant a à rattraper. Avec les probabilités contre lui, s'il n'a pas une série chanceuse très tôt, ses chances deviennent infimes au fur et à mesure qu'il prend plus de retard.**

Nous nous intéressons maintenant au temps que le destinataire d'une nouvelle transaction doit attendre avant d'être suffisamment rassuré sur le fait que l'émetteur ne pourra pas modifier la transaction. Nous supposons que l'émetteur est un attaquant qui souhaite faire croire au destinataire qu'il a été payé depuis un certain temps, puis souhaite modifier la transaction pour récupérer l'argent de la transaction après un certain délai. Le destinataire sera alerté quand cela arrivera, mais l'émetteur espère que cela sera trop tard.

Le destinataire génère une nouvelle paire de clés et donne la clé publique à l'émetteur peu de temps avant la signature. Cela évite que l'émetteur prépare une chaîne de blocs en avance en travaillant dessus jusqu'à ce qu'il obtienne une avance suffisante, et qu'il effectue la transaction à ce moment là. Une fois la transaction émise, l'émetteur malhonnête commence à travailler sur une chaîne alternative contenant une version modifiée de la transaction.

Le destinataire attend que la transaction ait été ajoutée à un bloc et que z **blocs aient été ajoutés à la suite de celui-ci. Il ne sait pas quel est exactement l'état d'avancement de l'attaquant, mais en supposant que les blocs légitimes aient mis le temps moyen attendu par bloc pour être générés, l'avancement potentiel de l'attaquant est une distribution de Poisson ayant comme valeur attendue :**

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

Afin d'obtenir la probabilité que l'attaquant arrive encore à rattraper, nous multiplions la densité de Poisson pour chaque quantité de progression qu'il a pu obtenir par la probabilité qu'il rattrape depuis ce point :

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

En réarrangeant pour éviter de sommer à l'infini...

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

Converti 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 effectuant quelques essais, nous observons que la probabilité diminue exponentiellement selon 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

Solutions 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 ne reposant pas sur la confiance. Nous avons démarré avec un cadre habituel de pièces faites de signatures numériques, ce qui procure un contrôle fort de la propriété, mais reste incomplet sans un moyen d'empêcher les doubles dépenses. Pour résoudre ce problème, nous avons proposé un réseau pair-à-pair utilisant des preuves de travail pour enregistrer un journal public des transactions, qui devient rapidement inattaquable par le calcul si les noeuds honnêtes contrôlent la majorité de la puissance de calcul. Le réseau est robuste de par sa simplicité non structurée. Les noeuds travaillent de concert avec très peu de

coordination. Ils n'ont pas besoin d'être authentifiés, puisque les messages ne sont pas envoyés à un destinataire particulier, et n'ont besoin d'être délivrés qu'au mieux. Les noeuds peuvent quitter et rejoindre le réseau à volonté, en acceptant la chaîne de preuve de travail comme preuve de ce qu'il s'est passé pendant leur absence. Ils votent en utilisant leur puissance de calcul, en exprimant leur accord vis à vis des blocs valides en travaillant à les étendre, et en rejetant les blocs invalides en refusant de travailler dessus. Toutes les règles nécessaires et les mesures incitatives 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.