Bitcoin : un système de paiement électronique pair-à-pair

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

Traduction française de bitcoin.org/bitcoin.pdf par Arnaud-François Fausse
@AFFAUSSE

Résumé. Une version d'un système de paiement purement pair-à-pair permettrait des paiements en ligne directs d'une partie à l'autre sans passer par une institution financière. Les signatures digitales fournissent une partie de la solution, mais les principaux bénéfices sont perdus si un tiers de confiance est encore nécessaire pour éviter les doubles-paiements. Nous proposons une solution au problème du double-dépense en utilisant un réseau pair-à-pair. Le réseau horodate les transactions en les hachant en une chaîne continue de preuves-de-travail, formant un enregistrement de données qui ne peut pas être changé sans avoir à refaire la preuve-de-travail. La chaîne la plus longue non seulement sert de preuve par témoignage de la séquence des événements, mais prouve qu'elle est issue du plus grand groupe de puissance CPU. Aussi longtemps que la majorité de la puissance CPU est contrôlée par des nœuds non participant à une attaque du réseau, ils engendreront la plus longue chaîne et surpasseront les attaquants. Le réseau en lui-même exige une structure minimale. Les messages sont diffusés au mieux et les nœuds peuvent quitter et rejoindre le réseau à leur gré, en acceptant la plus longue chaîne de preuve-de-travail créée en leur absence.

1. Introduction

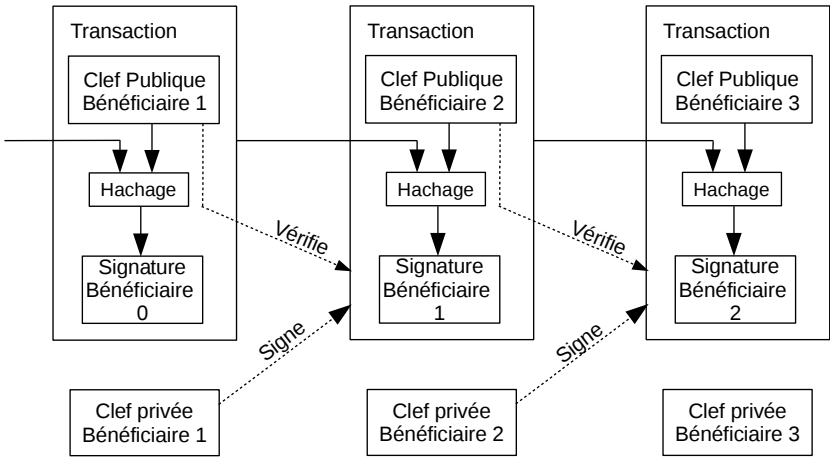
Le commerce sur Internet en est venu à reposer presque exclusivement sur les institutions financières agissant comme tiers de confiance afin de traiter les paiements électroniques. Alors que le système fonctionne suffisamment bien pour la plupart des transactions, il souffre de faiblesses inhérentes au modèle de confiance. Les transactions totalement irréversibles ne sont pas réellement possibles, car les institutions financières ne peuvent pas éviter les conflits de médiation. La coût de la médiation augmente les coûts de transaction, en limitant le montant minimum de la transaction et coupant ainsi la possibilité de transactions courantes à petit montant. De plus, il y a un coût plus important dans la perte de la capacité à faire des paiements irréversibles pour les services irréversibles. Avec la possibilité de réversibilité, la nécessité de la confiance s'étend. Les commerçants doivent se méfier de leurs clients, et les ennuyer en leur demandant plus d’information dont ils n’auraient pas besoin en procédant autrement. Un certain pourcentage de fraude est accepté comme inévitable. Ces coûts et incertitudes dans les paiements peuvent être évités par la présence et l’argent physiques, mais aucun mécanisme n’existe pour faire des paiements à travers un canal de communication sans un tiers de confiance.

Le besoin est d’avoir un système de paiement électronique basé sur une preuve cryptographique au lieu de la confiance, permettant à deux parties volontaires de réaliser entre elles des transactions sans le besoin d’un tiers de confiance. Des transactions calculatoirement incommodes à inverser protégeraient les vendeurs de la fraude, et des mécanismes habituels de dépôt pourraient être aisément implémentés pour protéger les acheteurs. Dans ce papier, nous proposons une solution au problème de la double-dépense en utilisant un serveur d’horodatage

distribué pair-à-pair afin d'engendrer calculatoirement la preuve de la chronologie des transactions. Le système est sûr tant que les nœuds honnêtes contrôlent collectivement plus de puissance CPU que celle de chacun des groupes de nœuds d'attaquants coopérants.

2. Transactions

Nous définissons une pièce (de monnaie) électronique comme une chaîne de signatures électroniques. Chaque propriétaire transfert la pièce au suivant en signant le hachage, de la transaction précédente, de la clef publique du prochain propriétaire et en ajoutant tout cela à la fin de la pièce. Un bénéficiaire peut vérifier les signatures pour vérifier la chaîne de propriété.



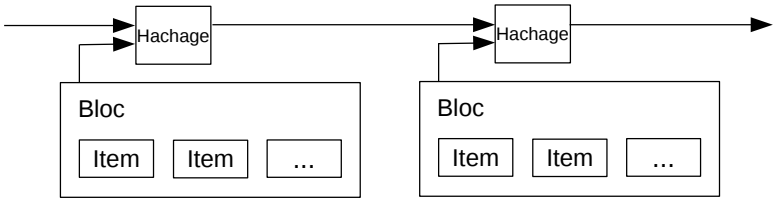
Le problème évidemment est que le bénéficiaire ne peut pas vérifier qu'un des propriétaires n'a pas dépensé deux fois la pièce. Une solution commune est d'introduire une autorité de confiance, ou émetteur de monnaie, qui vérifie chaque transaction concernant la double-dépense. Après chaque transaction, la pièce doit être renvoyée à l'émetteur de monnaie pour émettre une nouvelle pièce, et seules les pièces émises par l'émetteur sont réputées non dépensées deux fois. Le problème avec cette solution est que le destin de tout le système monétaire dépend de la société qui émet la monnaie, avec chaque transaction devant passer par elle, juste comme une banque.

Nous avons besoin d'un moyen pour le bénéficiaire de savoir que les précédents propriétaires n'ont pas signé de transactions précédentes. Pour nos fins, la transaction effectuée le plus tôt est celle qui compte, ainsi nous prêtons pas attention aux tentatives suivantes de double-dépense. Le seul moyen pour confirmer l'absence d'une transaction est d'être au courant de toutes les transactions. Dans le modèle d'un émetteur central de monnaie, ce dernier était au courant de toutes les transactions et décidait qui arrivait en premier. Pour accomplir pareille tâche sans un tiers de confiance, les transactions doivent être annoncées publiquement [1], et nous avons besoin d'un système pour les participants pour s'accorder sur une histoire unique de l'ordre dans lequel elles furent reçues. Le bénéficiaire a besoin de la preuve qu'au moment de chaque transaction, la majorité des nœuds était d'accord qu'elle était la première reçue.

3. Serveur d'horodatage

La solution que nous proposons commence avec un serveur d'horodatage. Un serveur d'horodatage fonctionne en prenant l'empreinte numérique d'un bloc d'items à horodater et à la publier largement, tel que dans un journal ou un forum sur Internet [2-5]. L'horodate prouve que les données ont dû exister à l'instant de

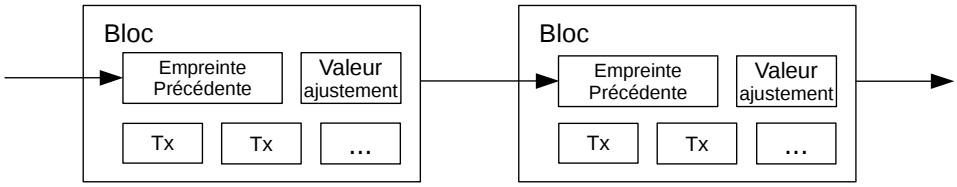
l'horodatage, évidemment, pour pouvoir obtenir leur empreinte numérique. Chaque horodate inclut l'horodate précédente dans son empreinte, formant une chaîne, avec chaque nouvelle horodate renforçant celles la précédant.



4. Preuve-de-travail

Pour implémenter un serveur d'horodatage distribué en pair-à-pair, nous avons besoin d'une preuve-de-travail similaire à celle d'Adam Back "Hashcash" [6], plutôt que d'un journal ou de publication sur un forum Internet. La preuve-de-travail implique la recherche d'une valeur qui une fois hachée, tel qu'avec le SHA-256, donne une empreinte numérique commençant par un nombre donné de bits à zéro. Le travail moyen demandé est exponentiel en fonction du nombre de bits à zéro exigés et peut être vérifié en exécutant un hachage unique.

Pour notre réseau d'horodatage, nous implémentons la preuve-de-travail par incrémentation d'une valeur d'ajustement dans le bloc jusqu'à trouver une valeur qui donne une empreinte avec le nombre de zéros requis. Une fois que la charge CPU a été dépensée pour satisfaire la preuve-de-travail, le bloc ne peut plus être changé sans refaire le travail. Étant donné que les blocs sont chaînés après le bloc considéré, le travail pour changer le bloc devrait inclure de refaire tous les blocs postérieurs.



La preuve-de-travail résout aussi le problème de la définition du processus de décision majoritaire. Si la majorité était basée sur une-adresse-IP-un-vote, elle pourrait être fraudée par quiconque capable d'allouer beaucoup d'IPs. La preuve-de-travail est par essence une-CPU-un-vote. La décision majoritaire est représentée par la chaîne la plus longue, qui a la plus grande preuve-de-travail investie. Si une majorité de la puissance CPU est contrôlée par des nœuds honnêtes, la chaîne honnête grandira la plus vite et dépassera toutes autres chaînes en compétition. Pour modifier un bloc passé, un attaquant aurait à refaire la preuve-de-travail du bloc et de tous les blocs après lui, et à ce moment là rattraper et surpasser le travail des nœuds honnêtes. Nous montrerons plus tard que la probabilité de rattrapage d'un attaquant plus lent diminue avec l'ajout des blocs subséquents.

Pour compenser l'augmentation de la vitesse du matériel et modifier l'intérêt de l'usage 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 calculés par heure. S'ils sont engendrés trop rapidement, la difficulté augmente.

5. Réseau

Les étapes pour faire fonctionner le réseau sont comme suit :

1) Les nouvelles transactions sont diffusées 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) Quand un nœud trouve une preuve-de-travail, il diffuse le bloc à tous les nœuds.
5) Les nœuds acceptent le bloc seulement si toutes les transactions sont valides et pas déjà dépensées.
6) Les nœuds expriment leur acceptation du bloc en travaillant à créer le prochain bloc de la chaîne, en utilisant l'empreinte numérique du bloc accepté comme l'empreinte précédente.

Les nœuds considèrent toujours la chaîne la plus longue comme la chaîne valide et continuent à travailler pour l'étendre. Si deux nœuds diffusent deux versions différentes du prochain bloc simultanément, les autres 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 deviendrait plus longue. Le lien sera rompu quand la prochaine preuve-de-travail est trouvée et une branche devient plus longue ; les nœuds qui étaient en train de travailler sur les autres branches commuteront alors sur la plus longue.

Les diffusions des nouvelles transactions n'ont pas besoin d'atteindre nécessairement tous les nœuds. Tant qu'elles atteignent beaucoup de nœuds, elles seront intégrées dans un bloc avant longtemps. Les diffusions de blocs sont aussi tolérantes aux pertes de messages. Si un nœud ne reçoit pas un bloc, il le demandera quand il recevra le prochain bloc et réalisera qu'il lui en manque un.

6. Prime de résultat

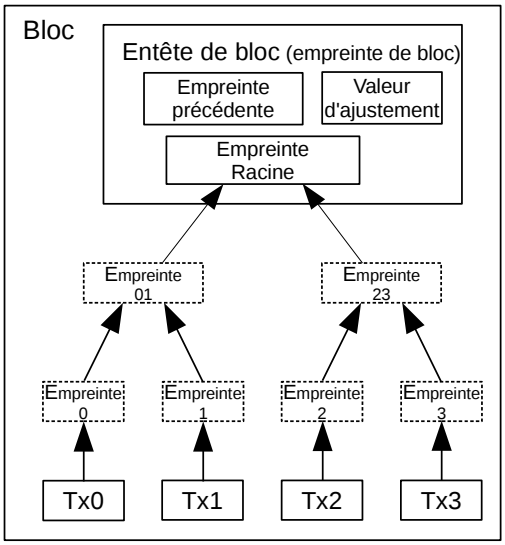
Par convention, la première transaction dans un bloc est une transaction spéciale qui commence par une nouvelle pièce détenue par le créateur du bloc. Cela ajoute une incitation pour les nœuds à supporter le réseau, et fournit un moyen initial de mettre des pièces en circulation puisqu'il n'y a pas d'autorité centrale d'émission de monnaie pour le faire. L'ajout stable d'un montant constant de nouvelles pièces est analogue aux chercheurs d'or dépensant des ressources pour ajouter de l'or en circulation. Dans notre cas, il s'agit de temps CPU et d'électricité qui sont dépensés.

La prime de résultat peut aussi être financée par des frais de transaction. Si la valeur sortie d'une transaction est inférieure à sa valeur d'entrée, la différence constitue les frais de transaction qui sont ajoutés à la prime de résultat du bloc contenant la transaction. Une fois mis circulation un nombre prédéterminé de pièces, la prime de résultat peut se convertir totalement en frais de transaction et être totalement non inflationniste.

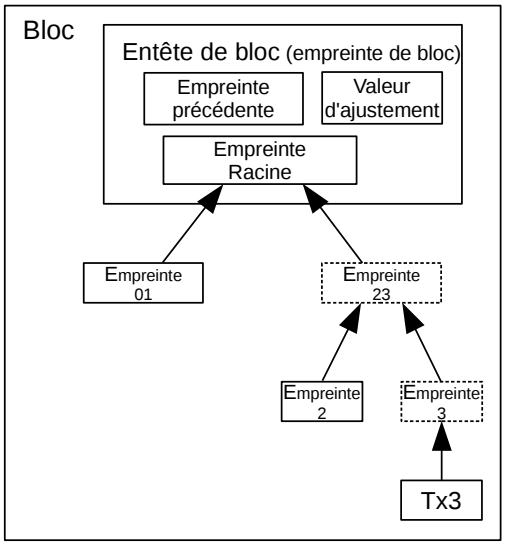
La prime de résultat peut aider à encourager les nœuds à rester honnêtes. Si un attaquant cupide était capable de réunir plus de puissance CPU que les nœuds honnêtes, il aurait à choisir entre escroquer les gens en récupérant frauduleusement ses paiements, ou, engendrer des nouvelles pièces. Il devrait trouver plus profitable pour jouer dans les règles, ces dernières le favorisant en lui offrant plus de nouvelles pièces que tout le reste du monde réuni, que de saper le système et la validité de sa propre fortune.

7. Demande d'espace disque

Une fois que la dernière transaction d'une pièce est enfouie en dessous de suffisamment de blocs, les transactions de dépenses la précédant peuvent être jetées pour sauver de l'espace disque. Pour faciliter cela sans casser l'empreinte numérique du bloc, les transactions sont hachées dans un arbre de Merkel [7][2][5], avec seulement la racine incluse dans l'empreinte numérique du bloc. Les anciens blocs peuvent alors être compactés par rognage des branches de l'arbre. Les empreintes intérieures de l'arbre n'ont pas besoin d'être stockées.



Transactions hachées dans un arbre de Merkel



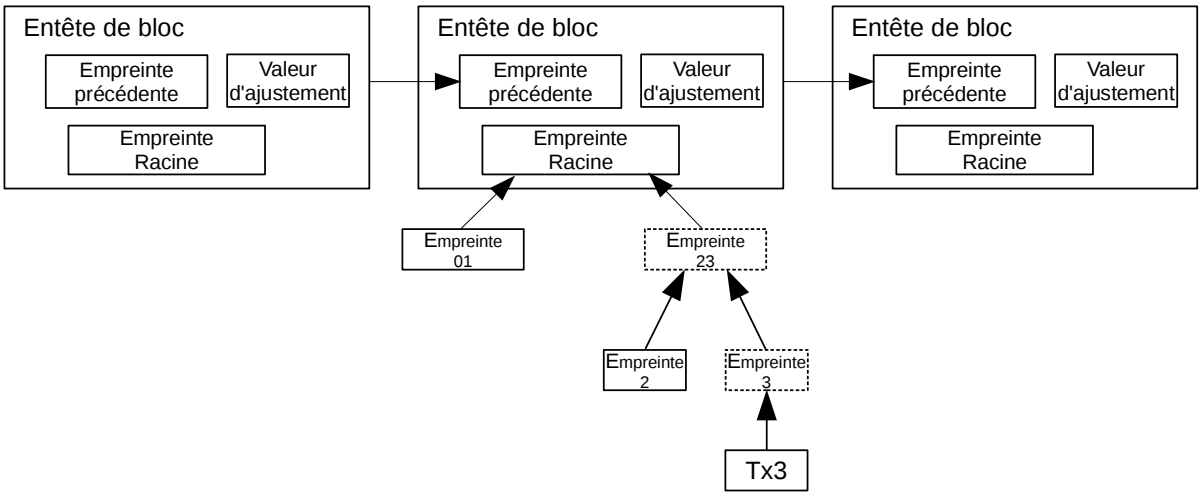
Après rognage de Tx0-2 du Bloc

Un entête de bloc sans transaction devrait être au environ de 80 octets. Si nous supposons les blocs engendrés toutes les dix minutes, 80 octets * 6 * 24 * 365 = 4,2 MOctets par an. Avec les ordinateurs typiquement vendus avec 2 GOctets de RAM en 2008, et la loi de Moore prédisant une croissance courante de 1,2 GOctets par an, le stockage ne devrait pas être un problème même si les entêtes de blocs doivent être gardés en mémoire.

8. Vérification de paiement simplifiée

Il est possible de vérifier des paiements sans faire fonctionner un nœud complet du réseau. Un utilisateur a seulement besoin de garder une copie des entêtes de bloc de la plus longue chaîne assurée par la preuve-de-travail, qu'il peut obtenir en requêtant les nœuds du réseau jusqu'à ce qu'il soit convaincu qu'il a la plus longue chaîne et obtienne la branche de Merkel liant la transaction au bloc l'horodatant. Il ne peut pas vérifier la transaction pour lui-même, mais en la liant à une place dans la chaîne, il peut voir que le réseau l'a acceptée, et les blocs ajoutés après le confirment.

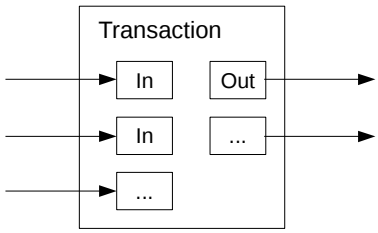
Chaîne à la plus grande preuve-de-travail



En tant que telle, la vérification est fiable tant que les nœuds honnêtes contrôlent le réseau, mais est plus vulnérable si le réseau est écrasé par la puissance d'un attaquant. Tant que les nœuds peuvent vérifier les transactions pour eux-mêmes, la méthode de vérification simplifiée peut être bernée par les transactions fabriquées d'un attaquant aussi longtemps que l'attaquant continue à surpasser le réseau. Une stratégie pour se protéger contre cela serait d'accepter des alertes des nœuds du réseau qui détectent un bloc invalide, provoquant le téléchargement du bloc complet et des transactions suspicieuses par le logiciel utilisateur pour confirmer la divergence. Les entreprises qui reçoivent fréquemment des paiements voudront probablement faire fonctionner leurs propres nœuds pour une sécurité plus indépendante et une vérification plus rapide.

9. Combinaison et séparation des valeurs

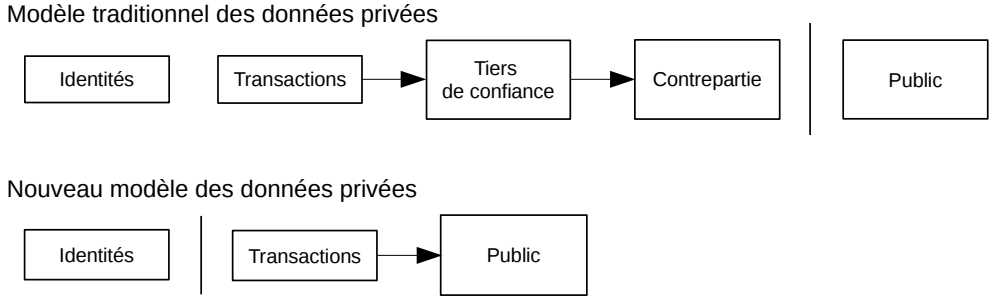
Bien qu'il soit possible de manipuler les pièces individuellement, il serait peu commode de faire une transaction séparée pour chaque centime dans un transfert. Pour autoriser les valeurs à être scindées ou combinées, les transactions contiennent entrées et sorties multiples. Normalement il y aura soit une entrée unique provenant d'une plus grosse et précédente transaction ou plusieurs entrées combinant des plus petits montants et au moins deux sorties : une pour le paiement, et une pour le rendu de la monnaie, le cas échéant pour le payeur.



Il doit être noté que la dissémination, où une transaction dépend de plusieurs transactions, et que ces transactions dépendent de bien plus, n'est pas un problème ici. Il n'y a jamais le besoin d'une copie complète et autonome de l'histoire des transactions.

10. Vie privée

Le modèle bancaire traditionnel réalise un niveau de protection de la vie privée en limitant l'accès aux informations aux personnes concernées et au tiers de confiance. La nécessité d'annoncer toutes les transactions publiquement écarte cette méthode, mais la protection de la vie privée peut encore être assurée en rompant le flux d'information à un autre endroit : en gardant les clefs publiques anonymes. Le public peut voir que quelqu'un est entrain d'envoyer un montant à quelqu'un d'autre, mais sans information liant la transaction à quelqu'un. Ceci est similaire au niveau d'information remis par la bourse, où les heures et montants des échanges, le "carnet d'ordres", est publique, mais sans dire qui sont les parties.



En guise de pare-feu additionnel, une nouvelle paire de clefs pourrait être utilisée pour chaque transaction afin de les garder non liées à un propriétaire commun. Toutefois, la liaison est inévitable avec les transactions multi-entrées, qui révèlent nécessairement que leurs entrées étaient détenues par un même propriétaire. Le risque est que si le propriétaire d'une clef est révélé, les liaisons peuvent révéler d'autres transactions qui ont appartenu au même propriétaire.

11. Calculs

Nous considérons le scénario d'un attaquant essayant d'engendrer une chaîne alternative plus rapidement que la chaîne honnête. Même si une telle tâche est accomplie, elle n'ouvre pas la porte à tous les changements arbitraires, tels que créer de la monnaie ex nihilo ou prendre de l'argent qui n'a 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 changer une de ses propres transactions pour récupérer l'argent qu'il a dépensé récemment.

La course entre la chaîne honnête et la chaîne de l'attaquant peut être caractérisée par un Chemin Aléatoire Binomial. L'événement de succès est que la chaîne honnête est étendue par un bloc, augmentant son avance de +1 et un événement d'échec est l'extension de la chaîne de l'attaquant réduisant l'écart de -1.

La probabilité qu'un attaquant gagne en partant d'un retard initial est analogue au problème de la ruine du joueur. Supposez un joueur, disposant d'un crédit illimité qui part avec un retard et joue potentiellement un nombre infini d'essais pour essayer d'atteindre le point d'équilibre. Nous pouvons calculer la probabilité qu'il atteigne un jour l'équilibre, 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 en partant avec z blocs de retard.

$$q_z = \begin{cases} 1 & si\ \ p \leqslant q \\ \left(\frac{q}{p}\right)^z & si\ \ p > q \end{cases}$$

Étant donnée notre hypothèse que p > q, la probabilité chute de manière exponentielle avec l'augmentation du nombre de blocs que l'attaquant doit rattraper. Avec la chance contre lui, s'il ne fait pas une avancée chanceuse dès le début, ses chances s'amenuisent, d'autant plus qu'il recule.

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

Le destinataire engendre une nouvelle paire de clefs et donne la clef publique à l'expéditeur juste avant la signature. Ceci évite que l'expéditeur prépare une chaîne de blocs à l'avance en travaillant dessus continuellement jusqu'à ce que la chance lui permette d'être suffisamment en avance, et d'exécuter alors la transaction. 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 jusqu'à ce que la transaction soit ajoutée à un bloc et z blocs ont été liés après. Il ne connaît pas l'avancement exact que l'attaquant à réalisé, mais en supposant que les blocs honnêtes ont pris le

temps moyen attendu par bloc, l'avancement potentiel de l'attaquant suivra une loi de Poisson avec la valeur attendue :

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

Pour obtenir la probabilité que l'attaquant puisse encore rattraper, nous multiplions la densité de la loi de Poisson pour chaque montant d'avancement il pourrait avoir, par la probabilité qu'il ait rattrapé depuis ce point :

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

En réarrangeant pour éviter de sommer la série infinie de la distribution :

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

En convertissant 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 obtenant quelques résultats, nous pouvons voir que la probabilité chute de manière 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 se passant de confiance. Nous avons commencé avec un cadre de fonctionnement ordinaire de pièces de monnaie établies par des signatures électroniques, qui offre un contrôle puissant de la propriété, mais qui est incomplet sans moyen d'éviter la double-dépense. Pour résoudre cela, nous avons proposé un réseau pair-à-pair utilisant la preuve-de-travail pour enregistrer une histoire publique des transactions, qui devient rapidement impraticable à un attaquant de modifier si les nœuds honnêtes contrôlent la majorité de la puissance CPU. Le réseau est robuste dans sa simplicité non structurée. Les nœuds travaillent tous ensemble avec peu de coordination. Ils n'ont pas besoin d'être identifiés, puisque les messages ne sont pas routés vers des destinations particulières et ont seulement besoin d'être livrés au mieux. Les nœuds peuvent quitter et rejoindre le réseau à leur gré, en acceptant comme preuve la chaîne de preuve-de-travail de ce qui s'est passé en leur absence. Ils votent avec leur puissance CPU, exprimant leur acceptation des blocs valides en travaillant à les étendre et à rejeter les blocs invalides en refusant de travailler dessus. Toutes les règles et primes de résultat nécessaires peuvent être imposées avec ce mécanisme de consensus.

References

[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.

Notes de traduction

Cette traduction a été réalisée avec le plus grand soin mais ne prétend en aucun cas être un substitut parfait du document original en anglais de Satoshi Nakamoto bitcoin.org/bitcoin.pdf . Ce document doit donc être utilisé avec la prudence qui s'impose et il convient d'utiliser les documents originaux pour réaliser toute œuvre dérivée.

Double-spending	:	Double-dépense
Peer-to-peer	:	Pair-à-air
Proof-of-work	:	Preuve-de-travail
Best effort	:	Au mieux
Coin	:	Pièce
Time stamp	:	Horodate
Time stamping	:	Horodatage
Hash	:	Empreinte numérique, hachage
IP	:	Internet Protocol (address) : Adresse Internet