White Paper Bitcoin traduit en Français par COIN24.FR

Le bitcoin : un système de monnaie électronique en peer-to-peer

Satoshi Nakamoto

satoshin@gmx.com

https://coinjournal.net/fr/bitcoin/white-paper/

Résumé. Une version exclusivement peer-to-peer d'argent électronique permettrait aux paiements en ligne d'être envoyés d'une personne à une autre sans passer par une institution financière. Les signatures numériques fournissent une part de la solution, mais les avantages principaux sont perdus si une tierce personne est toujours requise pour empêcher les doubles dépenses.

Nous proposons une solution au problème de double dépense en utilisant un réseau peer-to-peer. Le réseau horodate les transactions en les découpant dans une chaîne basée sur des trames avec « preuve de travail » (proof-of-work), formant un enregistrement qui ne peut être modifié sans refaire la preuve de travail. La chaîne la plus longue ne sert pas seulement en tant que preuve de séquence d'événements témoins, mais prouve également qu'elle provient du groupement CPU le plus puissant. Tant que la majorité de la puissance CPU est contrôlée par des nœuds qui ne coopèrent pas pour pirater le réseau, ils vont générer la plus longue chaîne et distancer les pirates. Le réseau lui-même nécessite une structure minimaliste. Les messages sont diffusés sur une base de « meilleur effort », et les nœuds peuvent quitter et rejoindre le réseau à leur gré, en acceptant la plus longue chaîne de preuve de travail en tant que preuve de ce qui vient de se passer pendant leur absence.

Introduction

Le commerce sur internet en est venu à dépendre presque exclusivement sur des institutions financières servant de tiers de confiance dans le processus de paiement électronique. Bien que le système fonctionne suffisamment correctement pour la plupart des transactions, il souffre toujours des faiblesses inhérentes à son modèle basé sur la confiance. Les transactions totalement non-réversibles ne sont pas vraiment possibles, car les institutions financières ne

peuvent pas éviter la médiation de conflits. Le coût de la médiation augmente le coût des transactions, limitant la taille minimale de transaction utile, empêchant la possibilité des petites transactions courantes, avec de plus un coût important dans la perte de capacité à faire des paiements non-réversibles pour les services non-réversibles. La possibilité de réversibilité apporte une besoin de confiance. Les marchands doivent être méfiants vis-à-vis de leurs clients, leurs demandant beaucoup d'informations dont ils auraient besoin autrement. Un certain pourcentage de fraude est accepté car inévitable. Ces incertitudes de coûts et de paiements peuvent être évitées en personne en utilisant des monnaies physiques, mais aucun mécanisme n'existe pour créer des paiements sur un canal de communication sans un tiers de confiance.

Ce qui est nécessaire est un système de paiement électronique basé sur une preuve cryptographique au lieu d'un modèle basé sur la confiance, permettant à deux parties de réaliser des transactions directement entre elles sans avoir recours à un tiers de confiance. Les transactions qui sont informatiquement incapables d'être réversibles protègeraient les vendeurs de la fraude, et des routines de mécanisme de dépôt en mains tierces pourraient facilement être implémentées pour protéger les acheteurs. Dans cet article, nous proposons une solution au problème de double dépense en utilisant un serveur horodaté distribué en peer-to-peer pour générer une preuve informatique de l'ordre chronologique des transactions. Le système est sécurisé tant que des nœuds honnêtes contrôlent collectivement plus de puissance CPU que des nœuds de groupes pirates coopératifs.

Les transactions

Nous définissons une pièce électronique comme une chaîne de signatures numériques. Chaque propriétaire transfère la pièce au suivant en signant numériquement une empreinte de la transaction précédente ainsi que la clé publique du propriétaire suivant puis ajoute tout cela à la fin de la pièce. Un bénéficiaire peut examiner les signatures pour vérifier la chaîne de propriété.

Evidemment, le problème est que le bénéficiaire ne peut pas vérifier que l'un des propriétaires n'a pas dépensé deux fois la pièce. Une solution courante est

d’intégrer une autorité centrale de confiance, ou « hôtel des monnaies », qui vérifie chaque transaction pour éviter les doubles dépenses. Après chaque transaction, la pièce doit être retournée à l’hôtel des monnaies pour distribuer une nouvelle pièce, et seules les pièces issues directement de l’hôtel des monnaies sont approuvées comme n’étant pas issues d’une double dépense. Le problème avec cette solution est que le destin du système monétaire entier dépend de la compagnie qui dirige l’hôtel des monnaies, chacune des transactions y transitant tout comme dans une banque.

Nous avons besoin d’un moyen pour que le bénéficiaire sache que le propriétaire précédent ne vient pas de signer des transactions auparavant. Dans ce but, la toute première transaction est celle qui compte, donc nous ne nous soucions pas des tentatives ultérieures pour les doubles dépenses. Le seul moyen de confirmer l’absence d’une transaction est d’être au courant de toutes les transactions. Dans le modèle basé sur l’hôtel des monnaies, celui-ci était au courant de toutes les transactions et décidait laquelle était arrivé la première. Pour réaliser cela sans tiers de confiance, les transactions doivent être publiquement annoncées [1], et nous avons besoin d’un système dans lequel les participants se mettent d’accord sur un seul historique de l’ordre dans lequel elles ont été reçues. Le bénéficiaire a besoin de la preuve qu’à l’heure de chaque transaction, la majorité des nœuds s’accordent de la première transaction reçue.

Les serveurs d’horodatage

La solution que nous proposons commence par un serveur d’horodatage. Un serveur d’horodatage fonctionne en prenant un morceau de bloc de produits qui doivent être horodatés, puis il le partage à grande échelle, comme sur un journal ou un article Usenet [2-5]. L’horodatage prouve que la donnée a dû exister dans le temps, évidemment, afin d’être intégrée dans l’empreinte. Chaque horodatage inclue l’horodatage précédent dans son empreinte, formant une chaîne dont chaque horodatage additionnel vient renforcer le précédent.

Preuve de travail

Pour implémenter un serveur d’horodatage distribué sur un réseau peer-to-peer, nous allons avoir besoin d’utiliser un système de « preuve de travail » similaire

au système Hashcash de Adam Back, plutôt qu'un journal ou un article Usenet. La preuve de travail nécessite de rechercher une valeur telle que son empreinte, calculée par exemple en utilisant l'algorithme SHA-256, débute par un certain nombre de bits à 0. Le travail nécessaire est exponentiel avec le nombre de bits à zéro nécessaire et peut être vérifié en exécutant une seule 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. Lorsque l'effort CPU a été utilisé pour satisfaire la preuve de travail, le bloc ne peut plus être modifié sans devoir refaire ce travail. Et alors que des blocs suivants s'enchainent après celui-ci, le travail pour modifier le bloc nécessiterait de retravailler tous les blocs suivants.

La preuve de travail résout également le problème de déterminer la représentation dans les décisions majoritaires. Si la majorité était basée sur le mode une adresse IP égale un vote, cela pourrait être piraté par quelqu'un capable de s'allouer plusieurs IPs. La preuve de travail est essentiellement basée sur un CPU égale un vote. La décision majoritaire est représentée par la chaine la plus longue, laquelle a la plus grande preuve de travail. Si une majorité de la puissance de calcul CPU est contrôlée par des nœuds honnêtes, alors la chaîne honnête va croitre le plus rapidement et dépasser n'importe quelle chaîne concurrente. Pour modifier un ancien bloc, le pirate devrait refaire toute la preuve de travail du bloc et de tous les autres blocs suivants, puis rattraper et dépasser le travail des nœuds honnêtes. Nous montrerons plus tard que la probabilité qu'un pirate ayant peu de puissance CPU puisse rattraper diminue exponentiellement au fur et à mesure que des blocs s'ajoutent.

Pour compenser l'augmentation de la vitesse de calcul et l'intérêt changeant de faire fonctionner des nœuds du réseau, la difficulté de la preuve de travail est déterminée par une moyenne mobile visant un nombre moyen de blocs par heure. Si les blocs sont générés trop rapidement, la difficulté augmente.

Réseau

Les étapes dans le fonctionnement du réseau sont les suivantes :

1- Les nouvelles transactions sont diffusées à tous les nœuds.

2- Chaque nœud rassemble les nouvelles transactions dans un bloc.

3- Chaque nœud travaille à trouver une preuve de travail difficile pour ses blocs.

4- Quand le 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 à l'intérieur sont valides et n'ont pas déjà été dépensées.

6- Les nœuds expriment leur acceptation du bloc en travaillant sur la création d'un nouveau bloc dans la chaîne, avec comme empreinte précédente celle du bloc accepté.

Les nœuds considèrent toujours la chaîne la plus longue en tant que chaine correcte et vont travailler à son extension. Si deux nœuds diffusent des versions différentes du prochain bloc simultanément, certains nœuds vont potentiellement recevoir l'un ou l'autre en premier. Dans ce cas-là, ils travaillent sur le premier reçu, mais enregistrent également l'autre branche au cas celle-ci devienne plus longue. La poire sera coupée en deux quand la prochaine preuve de travail sera trouvée et qu'une des branches devienne plus longue ; les nœuds qui travaillaient sur l'autre branche vont ensuite passer sur la plus longue.

La diffusion des nouvelles transactions n'a pas nécessairement besoin d'atteindre tous les nœuds. Tant qu'elles atteignent plusieurs nœuds, elles vont intégrer un bloc en peu de temps. La diffusion des blocs est également tolérante aux messages perdus. Si un nœud ne reçoit pas un bloc, il va le demander quand il recevra le prochain bloc et se rendra compte qu'il en manque un.

Incitation

Par convention, la première transaction dans un bloc est une transaction spéciale qui commence une nouvelle pièce appartenant au créateur du bloc. Cela incite les nœuds à participer au réseau, et fournit un moyen de distribution initiale de

pièces dans la circulation, car il n'y a pas d'autorité centrale pour les distribuer. L'ajout régulier d'un montant constant de nouvelles pièces est analogue à un mineur d'or qui étend ses ressources pour ajouter de l'or en circulation. Dans notre cas, il s'agit de temps de puissance de calcul CPU et d'électricité qui est étendu.

L'incitation peut aussi être financée par des frais de transactions. Si la valeur de sortie d'une transaction est inférieure à la valeur d'entrée, la différence correspond au frais de transaction qui est ajouté à la valeur incitative du bloc contenant la transaction. Une fois qu'un nombre prédéterminé de pièces sont entrées en circulation, l'incitation passera sur un financement entièrement basé sur les frais de transaction, sans aucune inflation.

L'incitation peut encourager les nœuds à rester honnêtes. Si un pirate cupide est capable d'assembler plus de puissance CPU que les nœuds honnêtes, il faudrait qu'il choisisse entre escroquer les gens en volant des paiements ou générer des nouvelles pièces. Il doit trouver plus profitable de suivre les règles, celles-ci le favorisent avec plus de nouvelles pièces que quiconque combiné, plutôt que de saper le système et la valeur de sa propre fortune.

Economiser de l'espace disque

Une fois que la dernière transaction d'une pièce est enfouie sous suffisamment de blocs, la transaction dépensée précédemment peut être abandonnée pour économiser de l'espace disque. Pour faciliter cela sans briser l'empreinte du bloc, les transactions sont découpées dans un arbre de Merkle [7][2][5], dont seule la racine est intégrée dans l'empreinte du bloc. Les anciens blocs peuvent être compressés en coupant des branches de l'arbre. Les empreintes internes n'ont pas besoin d'être enregistrées.

Un en-tête de bloc sans transaction pèse environ 80 octets. En supposant des blocs générés toutes les 10 minutes, 80octets * 6 * 24 * 365 = 4.2Moctets par an. Avec des systèmes informatiques ayant typiquement 2Go de RAM en 2008, et grâce à la loi de Moore prédisant l'évolution actuelle de 1.2Go par an, le

stockage ne devrait pas être un problème même si les en-têtes de bloc doivent être gardés en mémoire.

La vérification de paiement simplifiée

Il est possible de vérifier les paiements sans utiliser un nœud réseau entier. Un utilisateur a seulement besoin de garder une copie des en-têtes de bloc de la chaîne de preuve la plus longue, laquelle peut être obtenue en la demandant aux nœuds du réseau jusqu'à qu'il soit certain d'avoir la plus longue chaîne, et qu'il obtienne la branche de Merkle liant la transaction au bloc sur lequel il est horodaté. Il ne peut pas vérifier la transaction lui-même, mais en la liant à une position dans la chaîne, il peut voir qu'un nœud du réseau l'a acceptée, et les blocs ajoutés en suivant vont confirmer que le réseau l'a accepté.

Ainsi faite, la vérification est fiable tant que les nœuds honnêtes contrôlent le réseau, mais devient plus vulnérable si le réseau est pris par des pirates possédant plus de puissance de calcul. Bien que les nœuds du réseau puissent vérifier les transactions d'eux même, la méthode simplifiée peut être dupée par une transaction créée par un pirate tant que ce dernier est capable de contrôler la puissance informatique du réseau. Une stratégie pour se protéger contre cela est d'accepter les alertes venant des nœuds du réseau quand ils détectent un bloc invalide, incitant le logiciel de l'utilisateur à télécharger l'intégralité du bloc et des transactions alertées afin de confirmer l'incohérence. Les entreprises qui reçoivent des paiements fréquents vont probablement toujours vouloir diriger leurs propres nœuds pour plus d'indépendance de sécurité et des vérifications plus rapides.

Combiner et fractionner les valeurs

Bien qu'il soit possible de gérer des pièces individuellement, cela serait peu pratique de réaliser des transactions fractionnées pour chaque centime lors d'un transfert. Afin de permettre aux valeurs d'être combinées et fractionnées, les transactions possèdent de multiples entrées et sorties. Habituellement, il y aura soit une seule entrée venant d'une transaction précédente plus grande, soit de multiples entrées combinant des montants plus faibles, et au plus deux sorties : une pour le paiement et l'autre pour rendre la monnaie, si besoin, 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 ici. Il n'y a jamais le besoin d'extraire l'historique complet d'une transaction.

Confidentialité

Le modèle bancaire traditionnel obtient un niveau de confidentialité en limitant l'accès à l'information aux parties impliquées et aux tiers de confiance. La nécessite d'annoncer les transactions publiquement exclue cette méthode, cependant la confidentialité peut être maintenue en brisant le flux d'information à un niveau précis : en conservant les clés publiques anonymes. Tout le monde peut voir que quelqu'un envoie un montant à quelqu'un d'autre, mais sans information liant cette transaction à quiconque. Cela est similaire au niveau d'information des échanges boursiers, dans lesquels l'heure et la taille de chaque échange, le « cours », est rendu public, mais sans révéler l'information des parties impliquées.

Comme protection supplémentaire, une nouvelle paire de clé devrait être utilisé pour chaque transaction afin d'éviter qu'elles soient liées à un propriétaire commun. Des relations sont toujours inévitables avec les transactions multi-entrées, qui révèlent nécessairement que les entrées appartiennent au même propriétaire. Le risque est que le propriétaire de la clé soit révélé, le lien pourrait révéler les autres transactions qui appartiennent à ce même propriétaire.

Calculs

Nous considérons le scénario d'un pirate essayant de générer une chaîne alternative plus rapidement que la chaine honnête. Même si cela se réalise, cela ne mène pas le système à un changement arbitraire, tels que la création de valeur de nulle part ou la prise d'argent qui n'appartient pas au pirate. Les nœuds ne vont pas accepter de transactions invalides en tant que paiement, et les

nœuds honnêtes ne vont jamais accepter un bloc les contenant. Un pirate peut seulement essayer de modifier l'une de ses propres transactions afin de récupérer l'argent qu'il vient récemment de dépenser.

La course entre les chaînes honnêtes et les chaînes pirates peut être caractérisée comme une Marche Aléatoire Binomiale. L'événement de réussite est lorsque la chaine honnête s'étend d'un bloc, augmentant son avance de +1, l'événement d'échec est lorsque la chaine pirate s'étend d'un bloc, réduisant l'écart a -1.

La probabilité qu'un pirate rattrape un retard donné est analogue au problème de la faillite d'un parieur. Imaginez qu'un parieur au crédit illimité commence avec un déficit et joue potentiellement une infinité d'essai pour atteindre un seuil de rentabilité. Il est possible de calculer la probabilité qu'il atteigne ce seuil, ou dans notre cas qu'un pirate rattrape la chaine honnête, comme suit :

p = la probabilité qu'un nœud honnête trouve le prochain bloc

q = la probabilité qu'un pirate trouve le prochain bloc

qz = la probabilité qu'un pirate parvienne à rattraper en partant de « z » blocs de retard.

D'après l'hypothèse que p > q, la probabilité chute exponentiellement quand le nombre de blocs que l'pirate doit rattraper augmente. Si les chances ne sont pas de son côté, s'il n'obtient pas un élan de chance, ses chances disparaissent alors qu'il sombre loin dans la chaîne.

Nous considérons maintenant la durée pour le destinataire dont une transaction a besoin d'attendre avant d'être suffisamment certain que l'émetteur ne puisse plus changer la transaction. Nous présumons que l'émetteur est un pirate qui souhaite faire croire au destinataire qu'il l'a payé pendant un certain temps, puis finit par se payer lui-même après ce temps-là. Le destinataire sera alerté quand cela arrivera, mais l'émetteur espère que ce sera trop tard.

Le destinataire génère une nouvelle paire de clé et rend cette clé publique à l'émetteur rapidement après la signature. Cela empêche l'émetteur de préparer une chaîne de bloc en avance de phase en travaillant continuellement jusqu'à ce qu'il soit suffisamment chanceux pour être loin devant, puis exécute la transaction à ce moment-là. Une fois la transaction envoyée, l'émetteur malhonnête commence à travailler secrètement 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 que z blocs aient été liés par la suite. Il ne connait pas l'état d'avancement du pirate, mais il suppose que les blocs honnêtes ont pris le temps moyen attendu par bloc, le potentiel avancement du pirate va se comporter telle une distribution de Poisson avec la valeur attendue suivante :

Pour obtenir la probabilité que le pirate puisse rattraper son retard, nous multiplions la densité de Poisson pour chaque quantité d'amélioration qu'il a pu faire par la probabilité qu'il puisse rattraper de ce point :

En réarrangeant l'équation pour éviter de sommer l'infini de la distribution…

Conversion en langage 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;

}
```

Après quelques calculs, on observe que la probabilité chute exponentiellement 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

Résultats pour P inférieur à 0.1%…

$P < 0.001$

q=0.10 z=5

q=0.15 z=8

q=0.20 z=11

q=0.25 z=15

q=0.30 z=24

q=0.35 z=41

q=0.40 z=89

q=0.45 z=340

Conclusion

Nous venons de proposer un système pour des transactions électroniques qui ne sont pas basé sur la confiance. Nous avons commencé avec le cadre habituel des monnaies faites à partir de signatures numériques, qui fournissent un fort contrôle de propriété, mais qui est incomplet sans un moyen d'évitement des doubles dépenses. Pour résoudre cela, nous proposons un réseau peer-to-peer utilisant des preuves de travail pour enregistrer un historique public de

transactions qui devient rapidement informatiquement impossible à changer pour un pirate dans le cas où les nœuds honnêtes sont contrôlés par la majorité de la puissance de calcul CPU. Le réseau est robuste de par sa simplicité non structurée. Les nœuds travaillent de concert avec peu de coordination. Ils n'ont pas besoin d'être identifiés, car les messages ne sont pas routés à un endroit particulier et ont simplement besoin d'être délivrés sur un mode de meilleur effort. Les nœuds peuvent quitter et rejoindre le réseau à leur convenance, acceptant la chaîne de preuve de travail en tant que preuve de ce qui vient d'arriver en leur absence. Ils votent avec leur puissance de calcul, et expriment l'acceptation des blocs valides en travaillant sur leur l'extension et le rejet des blocs invalides en refusant de travailler sur eux. N'importe quelles règles nécessaires et mesures incitatives peuvent être appliquées par un 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, no2, 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.