XAALISI : UN SYSTEME DE PAIEMENT ELECTRONIQUE PEER-TO-PEER

Abstrait. Une version purement poste à poste de l'argent électronique permettrait aux paiements en ligne d'être envoyés directement d'une partie à une autre sans passer par une institution financière. Les signatures numériques constituent une partie de la solution, mais les principaux avantages sont perdus si un tiers de confiance est toujours nécessaire pour éviter les doubles dépenses. Nous proposons une solution au problème de la double dépense en utilisant un réseau Peer-to-Peer. Le réseau horodate les transactions en les hachant dans une chaîne continue de preuves de travail basées sur le hachage, formant ainsi un enregistrement qui ne peut être modifié sans refaire la preuve de travail. La chaîne la plus longue sert non seulement à prouver la séquence d'événements observés, mais également à prouver qu'elle provient du plus grand pool de ressources processeur. Tant qu'une majorité de la puissance du processeur est contrôlée par des nœuds qui ne coopèrent pas pour attaquer le réseau, ils généreront les attaquants les plus longs en chaîne et en dépassement. Le réseau lui-même nécessite une structure minimale. Les messages sont diffusés au mieux, et les nœuds peuvent quitter et rejoindre le réseau à leur guise, en acceptant la plus longue chaîne de preuves de travail comme preuve de ce qui s'est passé pendant leur absence.

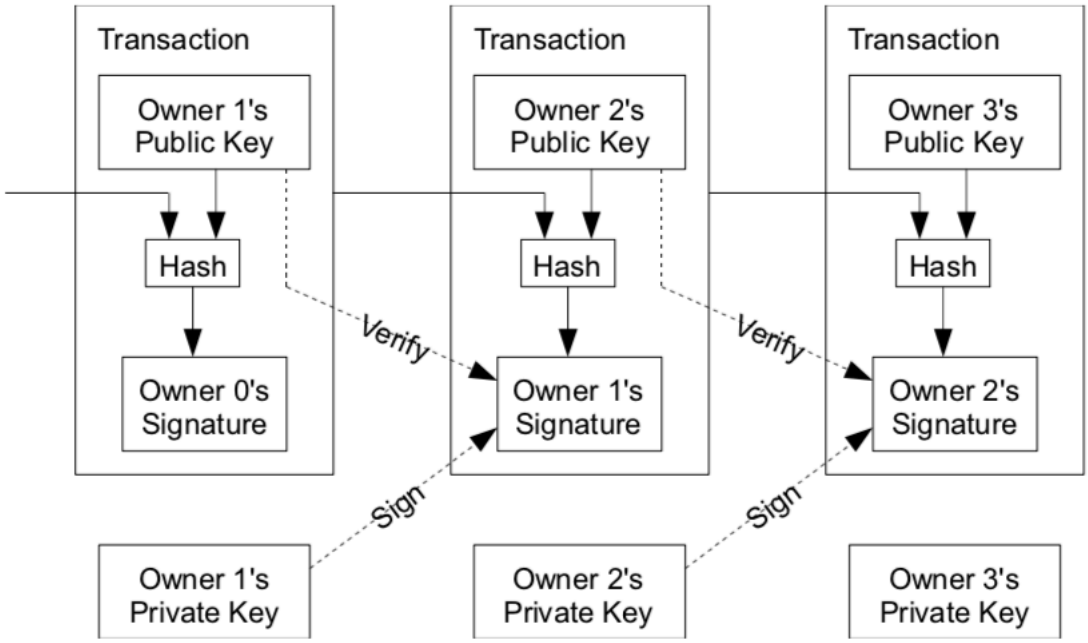
1. Introduction

Le commerce sur Internet a fini par compter presque exclusivement sur des institutions financières servant de tiers de confiance pour traiter les paiements électroniques. Bien que le système fonctionne assez bien pour la plupart des transactions, il souffre toujours des faiblesses inhérentes au modèle basé sur la confiance. Des transactions totalement irréversibles ne sont pas vraiment possibles, car les institutions financières ne peuvent éviter la médiation des différends. Le coût de la médiation augmente les coûts de transaction, limite la taille minimale de la transaction et empêche la possibilité de petites transactions occasionnelles. La perte de capacité à effectuer des paiements non réversibles pour des services non réversibles est également plus coûteuse. Avec la possibilité de renversement, le besoin de confiance augmente. Les commerçants doivent se méfier de leurs clients, les harcelant pour obtenir plus d'informations que nécessaire. Un certain pourcentage de fraude est accepté comme inévitable. Ces coûts et incertitudes de paiement peuvent être évités en personne en utilisant une monnaie physique, mais il n'existe aucun mécanisme pour effectuer des paiements sur un canal de communication sans une partie de confiance.

Ce qu'il faut, c'est un système de paiement électronique basé sur une preuve cryptographique plutôt que sur la confiance, permettant à deux parties intéressées de traiter directement l'une avec l'autre sans avoir besoin d'un tiers de confiance. Des transactions difficiles à effectuer sur le plan informatique protégeraient les vendeurs contre la fraude, et des mécanismes d'entiercement courants pourraient facilement être mis en place pour protéger les acheteurs. Dans cet article, nous proposons une solution au problème de double dépense en utilisant un serveur d’horodatage distribué entre homologues 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 processeur que tout groupe coopérant de nœuds attaquants.

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 un hachage de la transaction précédente et la clé publique du prochain propriétaire et en les ajoutant à la fin de la pièce. Un bénéficiaire peut vérifier les signatures pour vérifier la chaîne de propriété.



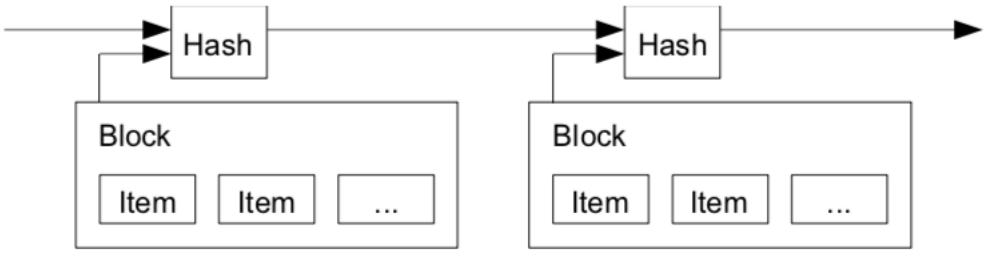
Le problème est bien sûr que le bénéficiaire ne peut pas vérifier que l'un des propriétaires n'a pas dépensé deux fois plus cher. Une solution courante consiste à mettre en place une autorité centrale de confiance, ou Monnaie, qui vérifie chaque transaction pour des dépenses doubles. Après chaque transaction, la pièce doit être retournée à la Monnaie pour émettre une nouvelle pièce. Seules des pièces émises directement de la Monnaie sont réputées ne pas être doublées. Le problème avec cette solution est que le sort de tout le système monétaire dépend de la société qui gère la Monnaie, chaque transaction devant y être traitée, comme une banque.

Nous devons trouver un moyen pour le bénéficiaire de savoir que les anciens propriétaires n'ont signé aucune transaction antérieure. Pour nos besoins, la transaction la plus ancienne est celle qui compte, nous ne nous soucions donc pas des tentatives ultérieures de doubler les dépenses. Le seul moyen de confirmer l'absence de transaction est de connaître toutes les transactions.

Dans le modèle basé sur la menthe, la menthe était au courant de toutes les transactions et décidait laquelle arrivait en premier. Pour ce faire sans une partie de confiance, les transactions doivent être annoncées publiquement [1] et nous avons besoin d'un système permettant aux participants de s'accorder sur un historique unique de l'ordre dans lequel ils ont été reçus. Le bénéficiaire a besoin de la preuve qu'au moment de chaque transaction, la majorité des nœuds ont convenu qu'il s'agissait du premier reçu.

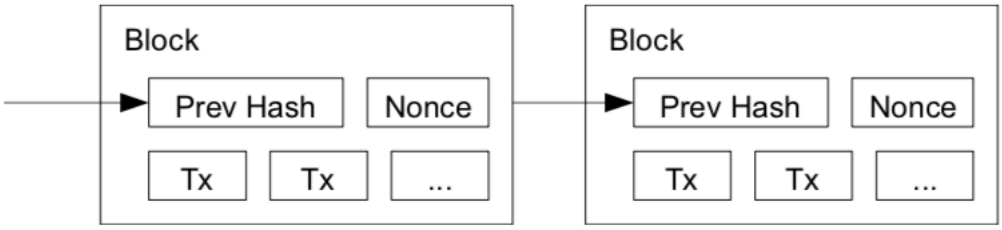
3. Serveur d'horodatage

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



4. Preuve de travail

Pour implémenter un serveur d'horodatage distribué sur une base d'égal à égal, nous devrons utiliser un système de preuve de travail similaire à Adam Back's Hashcash [6], plutôt que de journaux ou d'Usenet. La preuve de travail implique l'analyse d'une valeur qui, une fois hachée, telle que SHA-256, commence par un nombre de bits nuls. Le travail moyen requis est exponentiel



du nombre de bits nuls requis et peut être vérifié en exécutant un seul hachage.

Pour notre réseau d'horodatage, nous implémentons la preuve de travail en incrémentant un nonce dans le bloc jusqu'à ce qu'une valeur soit trouvée qui donne au bit de hachage les bits nuls requis. Une fois que le processeur a déployé des efforts considérables pour le rendre conforme à la preuve de travail, le bloc ne peut pas être modifié sans refaire le travail. Au fur et à mesure que les blocs ultérieurs sont enchaînés à la suite, le travail de modification du bloc inclurait la restauration de tous les blocs après.

La preuve de travail résout également le problème de la détermination de la représentation dans la prise de décision à la majorité. Si la majorité était basée sur une adresse IP, une voix, elle pourrait être subvertie par toute personne capable d'allouer de nombreuses adresses IP. La preuve de travail correspond essentiellement à un vote par unité centrale. La décision majoritaire est représentée par la chaîne la plus longue, qui a le plus grand effort de preuve du travail investi. Si la majorité de la puissance du processeur est contrôlée par des nœuds honnêtes, la chaîne honnête se développera le plus rapidement et dépassera toutes les chaînes concurrentes. Pour modifier un bloc passé, un

attaquant devrait refaire la preuve de travail du bloc et tous les blocs après, puis rattraper et surpasser le travail des nœuds honnêtes. Nous montrerons plus tard que la probabilité qu'un attaquant plus lent rattrape son retard diminue de manière exponentielle à mesure que des blocs ultérieurs sont ajoutés.

Afin de compenser l'augmentation de la vitesse du matériel et l'intérêt variable lié à l'exécution de nœuds au fil du temps, la difficulté de la preuve de travail est déterminée par une moyenne mobile ciblant un nombre moyen de blocs par heure. S'ils sont générés trop rapidement, la difficulté augmente.

5. Réseau

Les étapes pour exécuter le réseau sont les suivantes :

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

2) Chaque nœud collecte de nouvelles transactions dans un bloc.

3) Chaque nœud s'efforce de trouver une preuve de travail difficile pour son bloc.

4) Lorsqu'un nœud trouve une preuve de travail, il le diffuse à tous les nœuds.

5) Les nœuds acceptent le blocage uniquement si toutes les transactions qu'il contient sont valides et non déjà dépensées.

6) Les nœuds expriment leur acceptation du bloc en travaillant à la création du prochain bloc dans la

Chaîne, en utilisant le hachage du bloc accepté comme le hachage précédent.

Les nœuds considèrent toujours la chaîne la plus longue comme étant la bonne et continueront à l'étendre. Si deux nœuds diffusent simultanément des versions différentes du bloc suivant, certains nœuds peuvent recevoir l'un ou l'autre en premier. Dans ce cas, ils travaillent sur le premier qu'ils ont reçu, mais sauvegardent l’autre branche au cas où elle deviendrait plus longue. L'égalité sera brisée lorsque la prochaine preuve de travail sera trouvée et

qu'une branche deviendra plus longue ; les nœuds qui travaillaient sur l'autre branche passeront ensuite à la plus longue.

Les nouvelles transactions ne doivent pas nécessairement atteindre tous les nœuds. Tant qu'ils atteignent plusieurs nœuds, ils entrent dans un bloc avant longtemps. Les émissions bloquées sont également tolérantes aux messages supprimés. Si un nœud ne reçoit pas de bloc, il le demandera lorsqu'il recevra le prochain bloc et réalisera qu'il en a manqué un.

Incitation

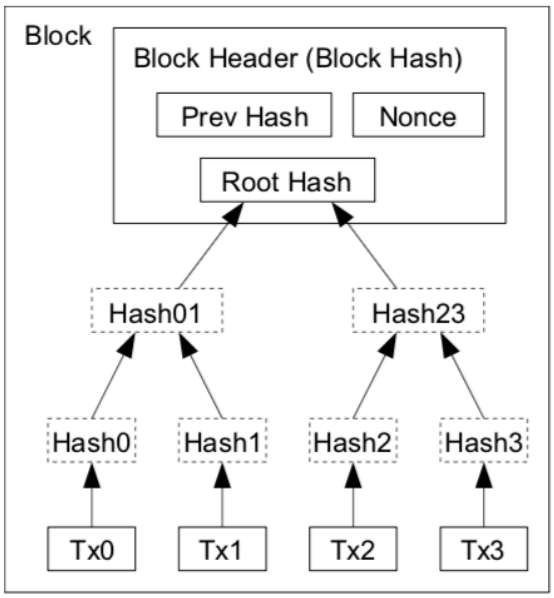
Par convention, la première transaction d'un bloc est une transaction spéciale qui démarre une nouvelle pièce appartenant au créateur du bloc. Cela incite les nœuds à prendre en charge le réseau et offre un moyen de distribuer initialement les pièces en circulation, car il n'existe aucune autorité centrale pour les émettre. L'ajout constant d'une quantité constante de nouvelles pièces de monnaie est analogue à celui utilisé par les orpailleurs qui dépensent des ressources pour ajouter de l'or à la circulation. Dans notre cas, c'est le temps de calcul et l'électricité qui sont dépensés.

L'incitatif peut également être financé avec des frais de transaction. Si la valeur de sortie d'une transaction est inférieure à sa valeur d'entrée, la différence correspond à des frais de transaction ajoutés à la valeur incitative du bloc contenant la transaction. Une fois qu'un nombre prédéterminé de pièces est entré en circulation, l'incitatif peut passer entièrement aux frais de transaction et être totalement exempt d'inflation.

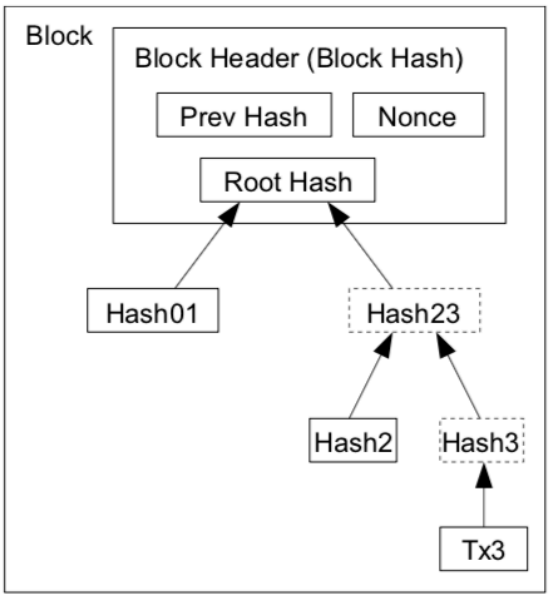
L'incitation peut aider à encourager les nœuds à rester honnêtes. Si un attaquant gourmand parvient à rassembler plus de puissance de traitement que tous les nœuds honnêtes, il devra choisir entre l'utiliser pour escroquer les gens en récupérant ses paiements ou l'utiliser pour générer de nouvelles pièces. Il devrait trouver plus rentable de respecter les règles, de telles règles qui le favorisent avec plus de pièces que toutes les autres combinées, plutôt que de saper le système et la validité de sa propre richesse.

6. Récupération de l'espace disque

Une fois que la dernière transaction dans une pièce de monnaie est enterrée sous un nombre suffisant de blocs, les transactions dépensées avant de pouvoir être rejetées pour économiser de l'espace disque. Pour faciliter ceci sans rompre le hachage du bloc, les transactions sont hachées dans un arbre de Merkle [7] [2] [5], avec seulement la racine incluse dans le hachage du bloc. Les vieux blocs peuvent ensuite être compactés en écrasant les branches de l'arbre. Les hachages intérieurs n'ont pas besoin d'être stockés.



Transactions Hashed in a Merkle Tree

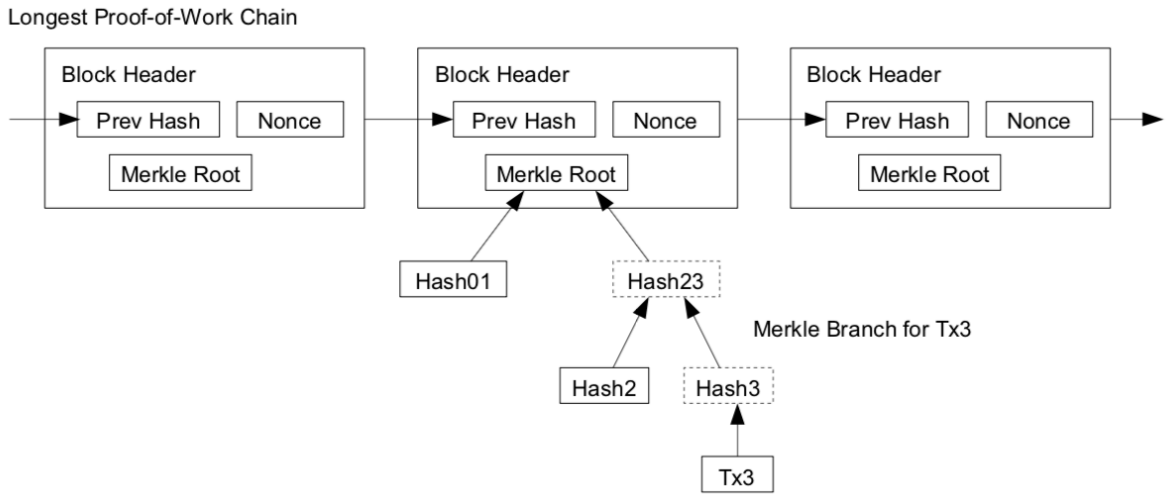


After Pruning Tx0-2 from the Block

Un en-tête de bloc sans transaction serait d'environ 80 octets. Si nous supposons que des blocs sont générés toutes les 10 minutes, 80 octets * 6 * 24 * 365 = 4,2 Mo par an. Avec les systèmes informatiques vendant généralement avec 2 Go de RAM à partir de 2008 et la loi de Moore prévoyant une croissance actuelle de 1,2 Go par an, le stockage ne devrait pas poser de problème, même si les en-têtes de bloc doivent être conservés en mémoire.

7. Vérification simplifiée des paiements

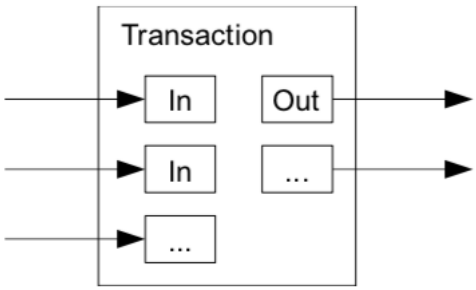
Il est possible de vérifier les paiements sans exécuter un nœud de réseau complet. Un utilisateur doit uniquement conserver une copie des en-têtes de bloc de la plus longue chaîne de preuve de travail, qu'il peut obtenir en interrogeant les nœuds du réseau jusqu'à ce qu'il soit convaincu qu'il possède la plus longue chaîne, et obtenir la branche Merkle qui relie la transaction au bloc. il ne peut pas vérifier la transaction pour lui-même, mais en le liant à un emplacement de la chaîne, il peut voir qu'un nœud de réseau l'a acceptée et les blocs ajoutés après l'avoir confirmée par le réseau.



En tant que telle, la vérification est fiable dans la mesure où des nœuds honnêtes contrôlent le réseau, mais elle est plus vulnérable si le réseau est maîtrisé par un attaquant. Bien que les nœuds du réseau puissent vérifier eux-mêmes les transactions, la méthode simplifiée peut être trompée par les transactions fabriquées par l'attaquant aussi longtemps que l'attaquant peut continuer à dominer le réseau. Une stratégie de protection contre ce problème consiste à accepter les alertes des nœuds du réseau lorsqu'ils détectent un bloc non valide, en invitant le logiciel de l'utilisateur à télécharger le bloc complet et des transactions alertées pour confirmer l'incohérence. Les entreprises qui reçoivent des paiements fréquents voudront probablement toujours utiliser leurs propres nœuds pour une sécurité plus indépendante et une vérification plus rapide.

8. Combinaison et partage de la valeur

Bien qu'il soit possible de manipuler des pièces individuellement, il serait difficile d'effectuer une transaction distincte pour chaque centime d'un transfert. Pour permettre de fractionner et de combiner une valeur, les transactions contiennent plusieurs entrées et sorties. Normalement, il y aura soit une entrée provenant d'une transaction précédente plus importante, soit plusieurs entrées combinant des montants plus faibles, et aux plus deux sorties : une pour le paiement et une pour renvoyer la modification, le cas échéant, à l'expéditeur.

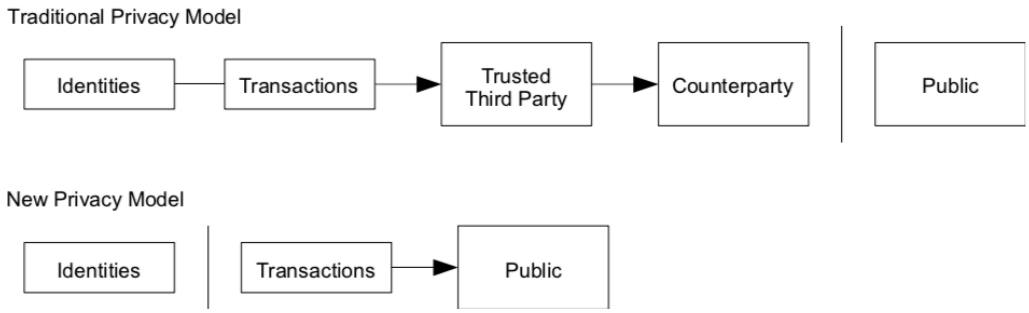


Il convient de noter que la ventilation, dans laquelle une transaction dépend de plusieurs transactions et de nombreuses autres, n’est pas un problème ici. Il n'est jamais nécessaire d'extraire une copie autonome complète de l'historique d'une transaction.

9. Confidentialité

Le modèle bancaire traditionnel atteint un niveau de confidentialité en limitant l'accès à l'information aux parties impliquées et au tiers de confiance. La nécessité d'annoncer publiquement toutes les transactions exclut cette méthode, mais la confidentialité peut toujours être préservée en interrompant le flux d'informations à un autre endroit : en gardant les clés publiques anonymes. Le public peut voir que quelqu'un envoie un montant à quelqu'un d'autre, mais sans information liant la transaction à qui que ce soit. Ceci est similaire au niveau d'information diffusé par les bourses, où l'heure et la taille

des transactions individuelles, la "bande", sont rendues publiques, mais sans préciser l'identité des parties.



En tant que pare-feu supplémentaire, une nouvelle paire de clés doit être utilisée pour chaque transaction afin d'empêcher qu'elles ne soient liées à un propriétaire commun. Certains liens sont toujours inévitables avec les transactions à entrées multiples, qui révèlent nécessairement que leurs entrées étaient la propriété du même propriétaire. Le risque est que si le propriétaire d'une clé est révélé, la liaison peut révéler d'autres transactions ayant appartenu au même propriétaire.

10. Calculs

Nous considérons le cas d'un attaquant essayant de générer une autre chaîne plus rapidement que la chaîne honnête. Même si cela est accompli, le système ne sera pas exposé à des changements arbitraires, tels que la création de valeur à partir de rien ou la prise d'argent qui n'a jamais appartenu à l'attaquant. Les nœuds n'accepteront pas une transaction non valide comme paiement et les nœuds honnêtes n'accepteront jamais un bloc les contenant. Un attaquant peut uniquement tenter de modifier l'une de ses propres transactions pour récupérer l'argent qu'il a récemment dépensé.

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

La probabilité qu'un attaquant rattrape un déficit donné est analogue au problème de la ruine d'un joueur. Supposons qu'un joueur avec un crédit illimité commence avec un déficit et joue potentiellement un nombre infini d'essais pour atteindre le seuil de rentabilité. Nous pouvons calculer la probabilité qu'il atteigne jamais le seuil de rentabilité, ou qu'un attaquant ne rattrape jamais 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 ne rattrape jamais z blocs derrière

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

Étant donné notre hypothèse selon laquelle p> q, la probabilité diminue de manière exponentielle à mesure que le nombre de blocs que l'attaquant doit rattraper augmente. Avec des chances contre lui, s'il ne fait pas un saut de chance en avant de bonne heure, ses chances deviennent infimes, alors qu'il est plus à la traîne.

Nous considérons maintenant combien de temps le destinataire d'une nouvelle transaction doit attendre avant d'être suffisamment certain que l'expéditeur ne peut pas modifier la transaction. Nous supposons que l'expéditeur est un attaquant qui veut faire croire au destinataire qu'il l'a payé pendant un certain temps, puis le remplace par un remboursement après un certain temps. Le destinataire sera averti lorsque cela se produira, mais l'expéditeur espère qu'il sera trop tard.

Le destinataire génère une nouvelle paire de clés et remet la clé publique à l'expéditeur peu de temps avant de signer. Cela empêche l'émetteur de préparer une chaîne de blocs à l'avance en y travaillant continuellement jusqu'à ce qu'il ait la chance d'avancer suffisamment longtemps avant d'exécuter la transaction à ce moment-là. Une fois la transaction envoyée, l'expéditeur malhonnête commence à travailler en secret sur une chaîne parallèle contenant une autre version de sa transaction.

Le destinataire attend que la transaction ait été ajoutée à un bloc et que z blocs aient été liés après celui-ci. Il ne connaît pas le montant exact des progrès de l'attaquant, mais en supposant que les blocs honnêtes prennent le temps moyen attendu par bloc, la progression potentielle de l'attaquant sera une distribution de Poisson avec la valeur attendue :

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

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

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

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

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

Conversion en code C ...

```
#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 lançant certains résultats, nous pouvons voir la probabilité diminuer 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
```

Résoudre pour P moins de 0,1% ...

```
P < 0.001
q=0.10   z=5
q=0.15   z=8
q=0.20   z=11
q=0.25   z=15
q=0.30   z=24
q=0.35   z=41
q=0.40   z=89
q=0.45   z=340
```

11. Conclusion

Nous avons proposé un système pour les transactions électroniques sans confiance. Nous avons commencé avec le cadre habituel des pièces constituées de signatures numériques, qui permet un contrôle strict de la propriété, mais qui est incomplet sans moyen de prévenir les doubles dépenses. Pour résoudre ce problème, nous avons proposé un réseau Peer-to-Peer utilisant une preuve de travail pour enregistrer un historique public des transactions qui devient rapidement impraticable pour un attaquant à changer si les nœuds honnêtes contrôlent la majorité de la puissance du processeur. Le réseau est robuste dans sa simplicité non structurée. Les nœuds fonctionnent tous en même temps avec peu de coordination. Ils n'ont pas besoin d'être identifiés, car les messages ne sont pas acheminés vers un endroit particulier et ne doivent être remis que dans la mesure du possible. Les nœuds peuvent quitter et rejoindre le réseau à leur guise, en acceptant la chaîne de validation du travail comme preuve de ce qui s'est passé pendant leur absence. Ils votent avec leur puissance CPU, exprimant leur acceptation des blocs valides en travaillant à leur extension et en rejetant les blocs non valides en refusant de travailler sur eux. Toutes les règles et incitations nécessaires peuvent être appliquées avec ce mécanisme de consensus.

www.xaalisi.com