Bitcoin : Analyse de la sécurité et de l'anonymat

Mathieu Lavoie [#1]

[#] *Département de génie logiciel et des technologies de l'information,*

École de technologie supérieure

[1] mathieu.lavoie.5@ens.etsmtl.ca

Introduction

Dans un système d'échange de monnaie en ligne, certains problèmes doivent être adressés. Il est primordial de vérifier la validité des transactions, l'identité réelle des acteurs impliqués dans la transaction et s'assurer que les transactions ne peuvent être dupliquées. Dans un système conventionnel, il y a une autorité de confiance qui gère les échanges entre utilisateurs. Cette autorité de confiance s'assure que les fonds sont bien transférés d'une personne à l'autre, que le payeur a les fonds nécessaires pour effectuer la transaction et que les deux acteurs sont authentifiés et que leur identité est confirmée.

Bitcoin a fait le gage qu'il est possible d'effectuer les vérifications nécessaires pour faire un système d'échange de monnaie, mais sans avoir d'autorité de confiance centrale. L'idée est qu'à l'intérieur d'un réseau pair à pair personne ne peut détenir plus de 50 % de la puissance de CPU. Ainsi, un système de validation par la majorité permet de vérifier la validité des transactions. Pour assurer l'intégrité des transactions et de l'historique des transactions, des mécanismes de hachage sont présents dans Bitcoin. Pour valider l'identité des acteurs impliqués dans les transactions, la cryptographie asymétrique et la signature des transactions sont nécessaires.

Plusieurs mécanismes propres à Bitcoin sont nécessaires pour contrer la fraude, l'usurpation d'identité et assurer l'intégrité des transactions. Dans Bitcoin, les transactions sont validées par le réseau pair à pairs. Dans un réseau pair à pair, chaque utilisateur du réseau est un nœud (*node*) du réseau. Un nœud reçoit et transmet l'information par les autres nœuds auxquels il est connecté. Lorsque la transaction est validée, elle n'est pas confirmée, un peu comme pour le fonctionnement de transaction de carte de crédit. Pour être confirmée, la transaction doit être ajoutée à un bloc de transaction. Ce bloc de transaction contient plusieurs transactions récentes qui n'ont pas été encore validées. Lorsqu'un nouveau bloc est créé, il est ajouté à la tête d'une liste chainée des blocs antérieurs nommée chaine de blocs. Cette chaine contient tous

les blocs de transactions depuis la création de Bitcoin et, par le fait même, toutes les transactions Bitcoin effectuées. Un bloc de transaction est créé en moyenne toutes les 10 minutes. L'utilisateur qui crée un nouveau bloc reçoit une prime en bitcoins du système ainsi que tous les frais de transactions que le bloc crée contiennent. Il est donc très intéressant pour les utilisateurs de créer de nouveaux blocs, car ils sont monétairement récompensés. Ce processus de création de bloc est appelé *mining.* Un total de 21 millions de bitcoins pourra être ainsi miné.

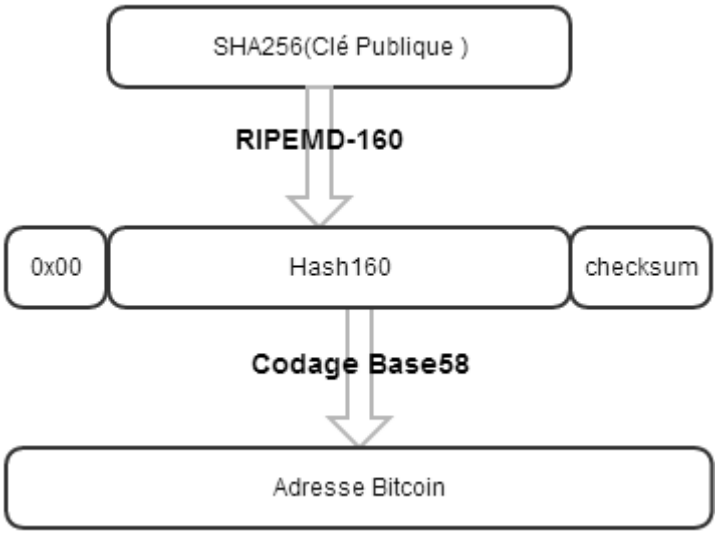
Le seul identifiant utilisé dans les transactions est l'adresse Bitcoin, dérivée de la clé publique du client. La cryptographie asymétrique est utilisée à la fois pour identifier les différents nœuds du réseau et pour signer les transactions. Puisqu'il n'existe pas d'autorité centrale et que les nœuds génèrent eux-mêmes leurs clés qui servent d'identifiants, Bitcoin est considéré comme anonyme. Cet anonymat n'est cependant pas complet : le journal des transactions est tout de même public. Une approche de regroupement par graphe des transactions, permettant d'obtenir des métadonnées intéressantes et de déduire des faits intéressants sur l'identité de certains membres du réseau, sera présentée dans la dernière section de ce rapport.

Dans la dernière section de ce rapport, une solution potentielle pour la détection de fraude en temps réel sera présentée. Cette solution a été présentée en 2012 dans l'article Two Bitcoins at the Price of One? Double-Spending Attacks on Fast Payments in Bitcoin. Certains systèmes de vente utilisant bitcoin utilisent cette approche pour détecter les failles. Ce service est exclusivement réservé aux clients de ces services et un compte utilisateur est nécessaire pour utiliser ce système. Ces systèmes ne sont pas open source et leur fonctionnement n'est pas entièrement connu et documenté. L'implémentation proposée à dernière section sera proposée sous forme d'un service anonyme et accessible publiquement.

Transactions

A. *Adresses*

Pour pouvoir effectuer des transactions, un client Bitcoin doit tout d'abord générer sa paire de clés asymétriques : une clé privée et une clé publique. L'algorithme de cryptographie asymétrique utilisée est ECDSA (Elliptic Curve Digital Signature Algorithm). L'adresse du client, qui lui sert d'identifiant est dérivée à partir de sa clé publique. Le schéma suivant illustre de quelle manière les adresses sont dérivées de la clé publique.



Pour déterminer l'adresse à partir de la clé publique, celle-ci est hachée avec SHA256. Cette empreinte est hachée de nouveau, cette fois-ci, avec RIPEMD-160. Le résultat est appelé *HASH160*. Un checksum de 4 octets est ajouté à l'empreinte. Un octet nul (0x00), indiquant la version présente du protocole, est préfixé. Le tout est encodé en Base58, pour finalement obtenir l'adresse Bitcoin.

Les adresses Bitcoin sont utilisées pour effectuer des transactions par les utilisateurs, mais au niveau du protocole, c'est la clé publique ou le HASH160 qui est utilisé comme identifiant. En effet, il est possible de récupérer directement le HASH160 en décodant l'adresse Bitcoin.

Rien n'empêche un utilisateur d'utiliser plusieurs adresses. En effet, tant que cet utilisateur possède les paires de clés pour ses adresses il pourra utiliser ses bitcoins. Si par contre cet utilisateur perd une clé

privée, l'adresse Bitcoin correspondante ne pourra plus être utilisée. Les bitcoins envoyés à cette adresse qui n'ont pas été dépensés sont irrécupérables.

B. *Format d'une transaction*

Une transaction Bitcoin est formée d'une ou plusieurs entrées de bitcoins et d'une ou plusieurs sorties. Une entrée d'une transaction est en fait une référence à une sortie d'une transaction antérieure.

Il existe communément deux types de transactions Bitcoin. Le premier type de transaction est la transaction de type Génération. Elle est la première transaction d'un bloc : c'est celle qui récompense l'utilisateur ayant réussi à créer le bloc valide. Le deuxième type de transaction est le type Adresses. C'est le type de transaction qui permet d'envoyer des bitcoins entre les utilisateurs du réseau.

Pour effectuer une transaction, un utilisateur doit prouver qu'il possède les bitcoins utilisées en entrées dans la transaction. Chaque entrée de la transaction doit contenir une référence vers la transaction qui a permis à l'utilisateur d'obtenir les bitcoins, c'est-à-dire la sortie d'une transaction antérieure qui a permis à l'utilisateur de recevoir les bitcoins. Ainsi, les entrées d'une transaction sont identifiées par le hachage de la transaction antérieur et l'index de la sortie. Par exemple, l'entrée 1 de la transaction représente les bitcoins obtenus lors de la transaction précédente identifiée par le hachage 0x123456789 à l'index de sortie 2. Les deux images suivantes illustrent une même transaction dans deux représentation différentes : au format JSON et dans un tableau résumé.


```
- in: [
    - {
        - prev_out: {
            hash: "cf4e2978d0611ce46592e02d7e7daf8627a316ab69759a9f3df109a7f2bf3ec3",
            n: 1
        },
        scriptSig: "30440220032d30df5ee6f57fa46cddb5eb8d0d9fe8de6b342d27942ae90a3231e0ba333e02203deee8060fdc7(
        04ae31c31bf91278d99b8377a35bbce5b27d9fff15456839e919453fc7b3f721f0ba403ff96c9deeb680e5fd341c0fc3a7b90c
    }
  ],
- out: [
    - {
        value: "0.01000000",
        scriptPubKey: "OP_DUP OP_HASH160 b0dcbf97eabf4404e31d952477ce822dadbe7e10 OP_EQUALVERIFY OP_CHECKSIG"
    },
    - {
        value: "2.99000000",
        scriptPubKey: "OP_DUP OP_HASH160 6b1281eec25ab4e1e0793ff4e08ab1abb3409cd9 OP_EQUALVERIFY OP_CHECKSIG"
    }
  ]
```

Inputs[?]

Previous output (index)[?]	Amount[?]	From address[?]	Type[?]	ScriptSig[?]
cf4e2978d061...:1	3	15vScfMHNrXN4QvWe54q5hwfVoYwG79CS1	Address	30440220032d30df5ee6f57fa46cddb5 04ae31c31bf91278d99b8377a35bbce

Outputs[?]

Index[?]	Redeemed at input[?]	Amount[?]	To address[?]	Type[?]	ScriptPubKey[?]
0	ae1933ba2b7d...	0.01	1H8ANdafjpqYntniT3Ddxh4xPBMCSz33pj	Address	OP_DUP OP_HASH160 b0dcbf97eabf4404e31d952477ce822 OP_EQUALVERIFY OP_CHECKS
1	20ed3fc9688a...	2.99	1Am9UTGfdnxabvcywYG2hvzr6qK8T3oUZT	Address	OP_DUP OP_HASH160 6b1281eec25ab4e1e0793ff4e08ab1a OP_EQUALVERIFY OP_CHECKS

Référence : Bitcoin Block Explorer[1]

Dans le cas précédent, Alice crée une nouvelle transaction pour envoyer de l'argent à deux adresses Bitcoin : 1H8AN[…]z33pj et 1Am9UT[…]UZT. Comme entrée de la transaction, Alice référence une transaction antérieure par laquelle elle a reçu 3 bitcoins. Pour référencer la transaction antérieure, le hachage de cette transaction sert d'identifiant. Dans le cas présent, il s'agit du hachage de transaction (cfe4e[…]3ec3). Cela semble demandant de vérifier toutes les transactions effectuées pour trouver la transaction ayant la bonne empreinte. Cependant, une liste des transactions qui n'ont pas encore été dépensées est gardée à jour par le réseau. Ainsi, l'ensemble de données à traiter est beaucoup moins grand.

Pour toute transaction, la somme des entrées doit égaler la somme des sorties. Il est cependant possible pour un utilisateur de renvoyer de l'argent vers lui-même. Par exemple, si Alice souhaite envoyer 0.1 bitcoin à Bob, elle peut référencer en entrée de la transaction une sortie d'une transaction précédente qui a envoyé cinq bitcoins à son adresse. En sortie de la transaction, Alice peut envoyer 0.1 bitcoin à l'adresse de Bob et 4.9 bitcoins à une ou plusieurs adresses qu'Alice possède. En envoyant le "change" de la transaction à différentes adresses qu'elle possède, Alice s'assure d'un meilleur anonymat.

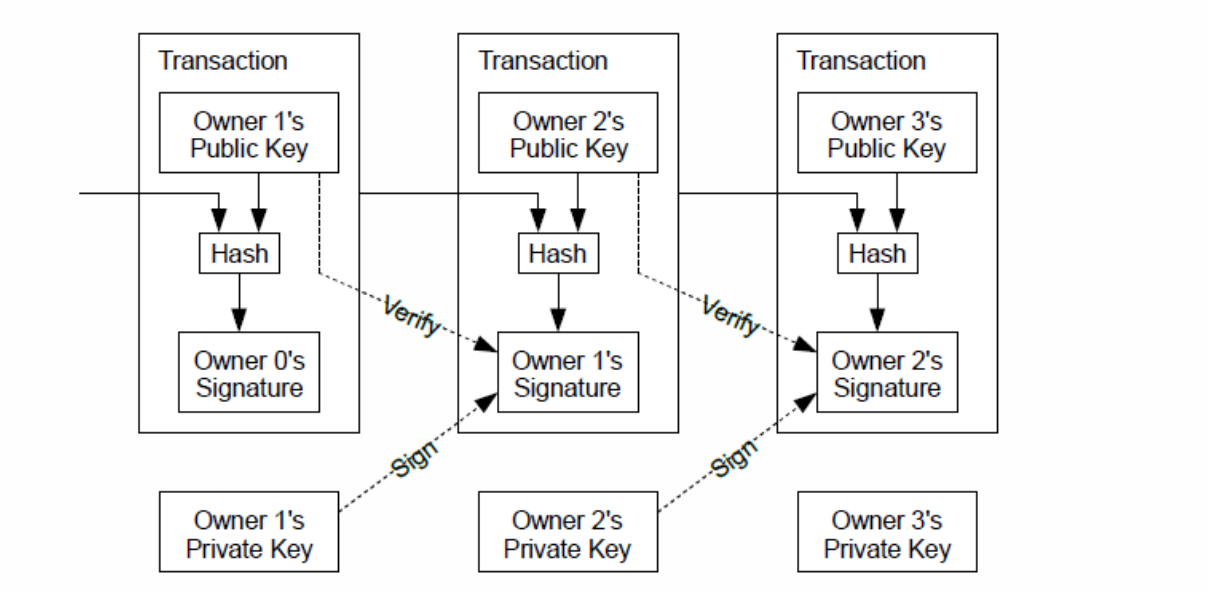
C. *Signature cryptographique*

Puisque les adresses sont publiques et que les transactions le sont aussi, un utilisateur malveillant, Ève, pourrait créer des transactions frauduleuses en usurpant l'identité d'un autre utilisateur. En effet, Ève

pourrait créer une transaction qui référencerait en entrée (*input*) une transaction antérieure où dix bitcoins ont été envoyés à Alice (et non pas à Ève). En sortie de cette transaction frauduleuse, les dix bitcoins sont transférés vers l'adresse d'Ève. Si aucun mécanisme d'authentification et de validation de l'identité n'était présent, un utilisateur malveillant pourrait transférer les bitcoins des autres utilisateurs vers son adresse.

Pour éviter qu'un utilisateur puisse effectuer des transactions frauduleuses au nom d'un autre utilisateur, les transactions doivent être signées pour vérifier l'identité du payeur. Chaque utilisateur doit prouver, pour chacune des entrées d'une transaction, qu'il possède vraiment l'adresse source et les bitcoins envoyés vers cette adresse.

Dans les deux figures précédentes, deux champs assurent cette vérification de l'identité des utilisateurs. Pour dépenser les trois bitcoins reçus par la transaction cfe4e[…]3ec3, Alice doit prouver qu'elle a l'adresse qui possède ces trois bitcoins. Pour faire cette preuve, Alice devra signer, avec sa clé privée, la transaction précédente (celle où elle a reçu trois bitcoins) et mettre cette signature dans le champ *ScriptSig*. Alice ajoute aussi dans ce champ sa clé publique pour que les utilisateurs du réseau puissent vérifier la signature et valider son identité. De cette manière, Ève ne peut pas faire de transaction en utilisant l'adresse d'Alice. En effet, Ève ne pourra pas signer la transaction puisqu'elle ne possède pas la clé privée d'Alice. La figure suivante illustre le processus de signature et le mécanisme de hachage mis en place pour éviter qu'un attaquant puisse modifier une transaction sans en modifier le hachage.



Référence : Bitcoin: A Peer-to-Peer Electronic Cash System [3]

Dans la figure précédente, la transaction antérieure (qui est utilisée comme entrée dans la transaction courante) et la clé publique du prochain propriétaire (owner 3's Public Key) sont hachées. L'envoyeur (owner 2) signe le tout avec sa clé privée pour approuver le transfert de bitcoins au prochain propriétaire. Ainsi seulement le nouveau propriétaire (owner 3) pourra à son tour transférer ces bitcoins, car il est le seul à posséder la clé privée. (Owner 3's Private Key)

Dans les sorties de la transaction, il y a un champ nommé *ScriptPubKey*. Ce champ indique de quelle façon l'utilisateur qui reçoit les bitcoins devra valider, à son tour, son identité pour pouvoir dépenser ces bitcoins. Dans la presque totalité des cas, l'utilisateur devra signer la transaction pour pouvoir dépenser les bitcoins reçus. En effet, L'opérateur OP_CHECKSIG indique que les personnes ayant reçu des bitcoins devront signer la transaction pour dépenser leur bitcoins reçus.

Bloc de transaction

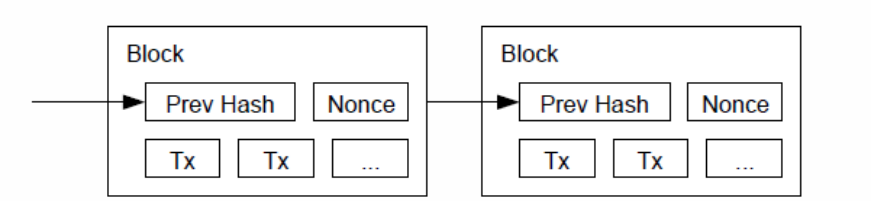
Un bloc de transaction est une unité de la chaine de transactions. Un bloc intègre dans la chaine de transactions un nombre de transactions non confirmées, mais validées. Pour faire accepter un nouveau bloc et ainsi confirmer les transactions qu'il contient, une preuve de travail (calcul cryptographique) appelée *mining* doit être effectuée par un nœud du réseau. Lorsqu'un nouveau nœud est créé (par le processus de *mining*), celui-ci est diffusé sur le réseau et est ajouté à la tête de la chaine des transactions.

A. *Mining*

Le processus de Mining consiste à prendre de nouvelles transactions qui n'ont pas été confirmées, c'est-à-dire qu'ils n'ont pas été intégrés dans la chaine de blocs. Les utilisateurs du réseau peuvent donc créer un nouveau bloc et ajouter ces nouvelles transactions à ce bloc. Tous peuvent créer un bloc, mais seulement le premier utilisateur qui résoudra un défi mathématique précis pourra le faire accepter par le réseau. Ce défi mathématique consiste simplement à trouver un bloc dont le hachage commence par un nombre précis de bits à 0. Pour modifier le hachage du bloc, les utilisateurs font varier le Nonce du bloc pour arriver à une solution du défi mathématique. Le premier utilisateur qui réussit le défi mathématique reçoit des bitcoins en récompense.

B. *Structure d'un bloc*

Un bloc de transaction inclut un ensemble de transactions qui n'ont pas encore été confirmées et ajoutées à la liste des transactions. Le bloc contient aussi le hachage du bloc précédent dans la chaine. Lorsqu'un bloc est créé, il n'est pas inclus dans la chaine. Puisque ce nouveau bloc doit devenir le nouveau bloc de tête de la chaine, le hachage du bloc précédent correspond au hachage du bloc présentement à la tête de la chaine. Finalement, un nonce est inclus dans le bloc. Ce nonce doit modifier le hachage du bloc courant de manière à ce que ce dernier commence par un certain nombre de « 0 ». La figure suivante résume bien la structure des blocs et l'interdépendance entre les blocs consécutifs.



Référence : Bitcoin: A Peer-to-Peer Electronic Cash System [3]

Comme le hachage du bloc précédent est inclus dans le bloc, il est impossible d'altérer les blocs antérieurs, car la chaine serait alors brisée. Ainsi, l'intégrité des blocs antérieurs est assurée et, par conséquent, l'intégrité des transactions effectuées l'est également.

C. *Nonce*

Puisque tous les nœuds peuvent créer des blocs et qu'un seul nœud peut être accepté comme étant la nouvelle tête du bloc, il y a une sorte de loterie qui s'organise. En effet, le premier client qui réussit à faire un bloc de transactions dont le hachage commence par un nombre prédéterminé de bits à « 0 » et à l'annoncer sur le réseau gagne cette « loterie ».

Puisque le hachage du bloc précédent et que la liste des transactions sont fixes, la seule partie variable du bloc varier est le nonce. Le but est donc de trouver, par essais-erreur, un nonce qui fait en sorte que le hachage global du bloc commence par un nombre prédéfini de bits à « 0 », c'est-à-dire de trouver un hachage dont la valeur est inférieure à la valeur limite définie par un coefficient de difficulté.

D. *Redevances pour l'effort de calcul*

Lorsqu'un nœud du réseau trouve un nouveau bloc respectant le critère du nombre de bits à « 0 », une redevance pour l'effort de calcul effectué (*mining*) lui est accordée. Ces bitcoins sont accordées par une transaction de type *Generation.* Cette transaction est la première du bloc. Cette redevance est réduite de moitié environ tous les 4 ans. Elle était initialement à 50 bitcoins et est maintenant à 25 bitcoins.

En plus des redevances en bitcoin pour miner des blocs, les frais de transactions sont aussi envoyés vers l'adresse du nœud qui a été en mesure de faire accepter le premier le bloc. Lorsque la totalité des bitcoins sera minée, la redevance pour avoir créé un nouveau bloc sera uniquement constituée des frais de transactions.

Les frais de transaction sont laissés à la discrétion de celui qui crée une transaction. Ces frais sont en fait un pourboire laissé au nœud qui réussit à créer un nouveau bloc valide incluant la transaction. Ainsi, les nœuds qui minent des blocs ont avantage à intégrer les transactions leur laissant un pourboire intéressant. Présentement, les transactions qui n'incluent pas de frais (pourboire) sont également incluses dans les nouveaux blocs. En effet, les frais de transactions sont généralement de l'ordre de 0.0001 bitcoin. La récompense de 25 bitcoins obtenus lors de la génération d'un bloc représente une somme beaucoup plus

attrayante que la somme des frais des transactions incluses dans le bloc crée. Lorsque les 21 millions de bitcoins seront minés, la redevance pour miner un bloc sera uniquement composée des frais de transaction. Ainsi, les transactions qui ne contiennent pas de frais de transaction ne seront plus intégrées dans les nouveaux blocs. Pour faire accepter ses nouvelles transactions, un utilisateur devra donner un pourboire sous forme de frais de transaction comme prime pour l'utilisateur qui réussit à intégrer sa transaction non confirmée dans un bloc valide. Il sera beaucoup plus intéressant pour les utilisateurs d'ajouter aux blocs les transactions leur rapportant le plus en frais de transactions et de négliger les transactions qui incluent peu ou pas de frais.

Transaction[?]	Fee[?]	Size (kB)[?]	From (amount)[?]
4a6a36271d...	0	0.168	Generation: 25 + 0.01842515 total fees

Référence : Bitcoin Block Explorer[1]

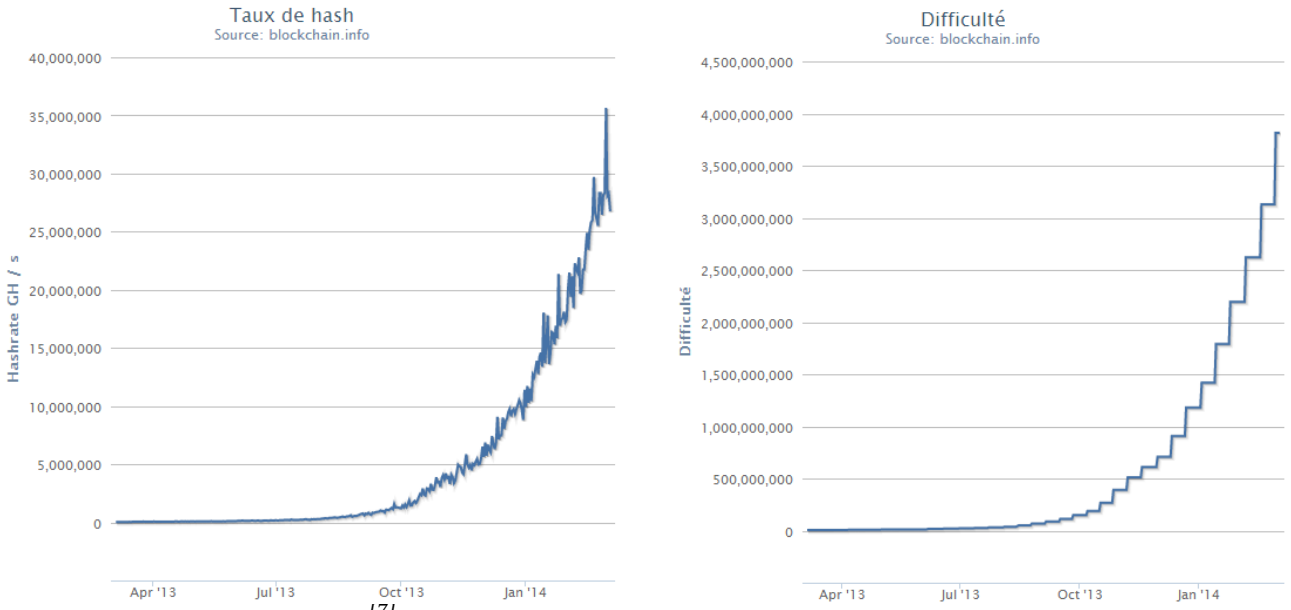
La première transaction d'un bloc est illustrée par la figure précédente. L'utilisateur ayant miné ce bloc a reçu la somme totale de 25.01842515 bitcoins. Les frais de transactions sont actuellement négligeables par rapport à la redevance de 25 bitcoins obtenue pour avoir miné le bloc.

E. *Création de bloc à temps constant*

Un bloc doit être généré aux dix minutes pour confirmer les nouvelles transactions. Le délai entre deux blocs doit être assez court pour confirmer rapidement les nouvelles transactions, mais assez long pour réduire les conflits entre les nouveaux blocs minés. Un conflit entre deux blocs se produit lorsque deux clients sont minés pratiquement en même temps. Il y a donc sur le réseau deux nouveaux blocs en concurrence pour être intégré en tête de la chaine de blocs. Ce cas sera expliqué à la section *Chaine de blocs — Embranchement de la chaine*.

La difficulté pour la création d'un bloc doit donc être ajustée toutes les deux semaines. La difficulté est la valeur maximale que les hachages des blocs minés peuvent avoir. Les hachages des blocs minés doivent donc commencer par un nombre minimal de bits à « 0 ». Ce nombre de bits à « 0 » est donc ajusté selon la puissance de calcul du réseau pour conserver la moyenne du nombre de blocs créé de six par heure.

Puisque la puissance de calcul du réseau augmente exponentiellement, la difficulté est ajustée en conséquence. Les deux graphiques suivants montrent la progression de la puissance de calcul du réseau sur une période d'un et de l'ajustement de la difficulté qui a été fait durant cette même période.



Référence : Blockchain.info [7]

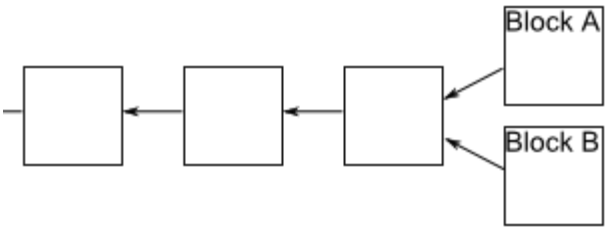
Pour miner des blocs, la majorité des utilisateurs s'inscrivent à des services de groupe de mining. En effet, la difficulté est trop grande pour qu'un seul utilisateur ait une chance de miner un bloc seul. Ainsi, certains sites web offrent la possibilité de s'inscrire à un groupe de mining et de distribuer la charge de calcul entre tous les utilisateurs du groupe. Lorsque le groupe réussit à miner un bloc, les bitcoins reçus sont répartis équitablement entre les utilisateurs du groupe.

Chaine de blocs

La chaine de blocs est l'historique complet de toutes les transactions effectuées sur le réseau Bitcoin depuis le début de l'existence de ce réseau. Il s'agit d'un journal public de toutes les transactions effectuées. Les transactions sont contenues dans les blocs et les blocs forment une liste chainée qui compose la chaine de blocs.

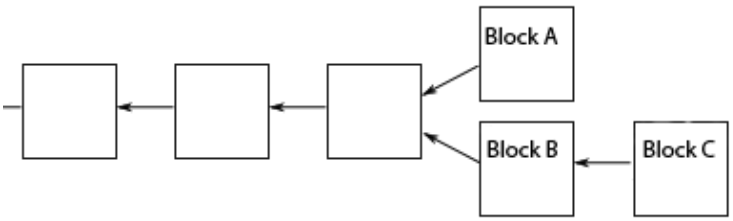
A. Embranchement de la chaine.

Puisque le réseau est un réseau pair à pairs, l'information est diffusée par tous les pairs. La propagation de l'information n'est pas prédictible et les délais de transmission sont variables. Si deux blocs ont été minés (bloc_a et bloc_b) par deux utilisateurs à quelques secondes d'intervalles, les deux utilisateurs propageront le bloc trouvé sur le réseau pairs à pairs. Il est donc possible que certains utilisateurs reçoivent le bloc_a avant le bloc_b et que d'autres utilisateurs reçoivent l'information dans l'ordre inverse (bloc_b suivi de bloc_a). Les utilisateurs gardent les deux blocs. Un embranchement est donc créé dans la chaine, tel qu'illustré par la figure suivante.



Ainsi, les deux blocs créés référencent le bloc précédant accepté dans la chaine. Les utilisateurs continuent à créer de nouveaux blocs de transaction basée sur le premier des deux blocs reçus.

Par exemple, Alice reçoit les deux blocs valides dans l'ordre suivant : bloc_a suivi de bloc_b. Alice essayera de miner un nouveau bloc basé sur le bloc_a. Si Bob, au contraire, reçoit les blocs dans l'ordre bloc_b suivi de bloc_a, il essayera de miner un bloc basé sur le bloc_b. Si Bob trouve une solution, il diffuse son nouveau bloc (bloc_c) lié au bloc_b. La structure de la chaine sera alors:



Dès que les utilisateurs du réseau reçoivent le nouveau bloc de Bob, ils changent immédiatement pour la branche la plus longue. Dans le cas présent, Alice commencera alors à créer de nouveaux blocs basés sur le bloc_c. Ainsi, la chaîne de bloc la plus longue d'inclura pas le bloc_a. Si certaines transactions avaient été incluses dans le bloc_a mais n'avaient pas été incluses dans le bloc_b, ces transactions sont retournées dans l'ensemble des transactions non confirmées. En effet, elles n'existent pas dans la chaine de blocs : c'est comme si elles n'avaient jamais été intégrées dans un bloc.

Un commerce qui utiliserait Bitcoin aurait donc avantage à attendre quelques blocs pour confirmer la vente et remettre le produit au client. Cette problématique sera plus longuement discutée dans la section *Sécurité – Double Spending*.

B. Intégrité de la chaine

Pour éviter qu'un utilisateur puisse altérer un bloc de la chaine, chacun des blocs de la chaine inclut le hachage du précédent. Ainsi, si un bloc est altéré, la chaine est tout simplement brisée. Un utilisateur malveillant ne pourrait donc pas mentir au reste du réseau sur le montant reçu dans les transactions précédentes et sur la composition des blocs présents dans la chaine. Un utilisateur malveillant ne pourrait pas enlever ou rajouter un bloc au milieu de la chaine.

La nécessité d'intégrer le hachage du bloc précédent évite qu'un utilisateur malveillant puisse précalculer une séquence de bloc et utiliser cette séquence précalculée pour effectuer un *double spending*. Ce cas sera présenté plus en détail dans la section *Sécurité – Double Spending.* En effet, puisque toute la chaine doit inclure le hachage du précédent, un changement au hachage du premier bloc de la chaine précompilé invalidera tous les autres blocs de la chaine précalculée.

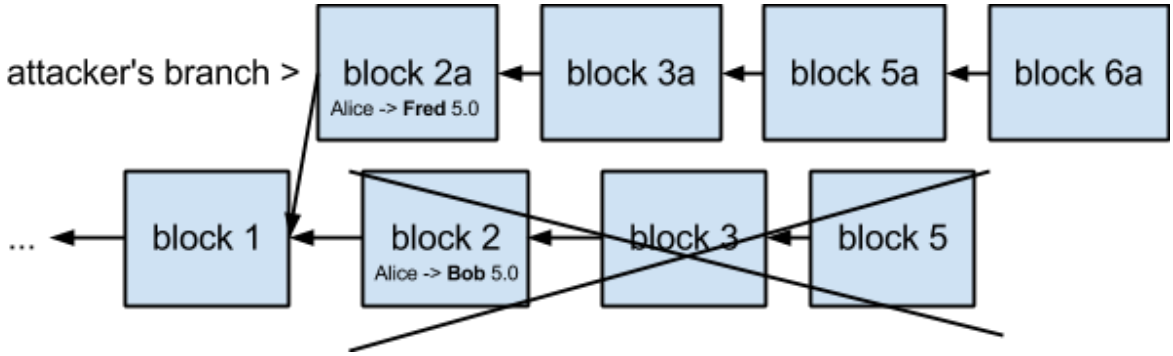
Sécurité

Il y a quelques attaques possibles sur Bitcoin. Les attaques ne sont pas liées à des défauts dans la conception du système, mais plutôt à une utilisation incorrecte faite par les utilisateurs. En effets, les utilisateurs de Bitcoin doivent connaitre quelques notions importantes pour éviter de faire des erreurs qui pourraient leur couter cher.

A. Double Spending

L'attaque la plus connue est sans doute le *double spending*. Elle consiste à générer deux transactions lors d'un achat par Bitcoin : une transaction légitime et une transaction qui envoie l'argent vers l'adresse de l'attaquant. Par exemple, Alice crée une transaction pour envoyer cinq bitcoins à Bob (transaction_1) et une autre envoyant cinq bitcoins à elle-même (transaction_2). Puisque dans un réseau pairs à pairs l'information est propagée de manière non déterministe, certains utilisateurs vont recevoir la transaction_2 avant la transaction_1. Ces utilisateurs vont donc ignorer la transaction_1 puisque, de leur point de vue, les bitcoins ont déjà été dépensés lors de la transaction_2. Ainsi, certains utilisateurs vont essayer d'inclure la transaction_2 dans un nouveau bloc de transaction et le reste du réseau tentera d'inclure la transaction_1. Une course s'engage alors pour la génération du prochain bloc de transactions. Le bloc suivant, il contiendra soit la transaction_1 ou soit la transaction_2, dépendamment de quel utilisateur aura réussi à miner le bloc. Il n'est pas possible de savoir quelle transaction sera acceptée tant que le prochain bloc n'est pas miné. Il y a donc un délai moyen de dix minutes pour confirmer une transaction bitcoin. Il s'agit ici d'une approche naïve où l'attaquant a une chance sur deux que la transaction_2 soit acceptée, plutôt que la transaction légitime. Cette attaque est seulement valable si le vendeur n'attend pas le prochain bloc de la chaine avant de procéder à la vente.

Il existe un moyen plus avancé pour effectuer une attaque de type *double spending.* Si l'attaquant a à sa disposition une grande puissance de calcul, il peut essayer de battre le réseau de vitesse pour faire accepter dans la chaine la transaction qu'il souhaite. L'illustration suivante illustre une attaque de *double spending* réussie.



Référence : Imponderable Things (Scott Driscoll's Blog).[4]

Comme illustrée dans la figure, Alice effectue deux transactions, une envoyant les cinq bitcoins à Bob et l'autre envoyant ces mêmes cinq bitcoins à Fred. Elle diffuse sur le réseau uniquement la transaction vers Bob. Alice, qui détient une grande puissance de calcul, mine le bloc suivant en incluant la transaction vers Fred dans son bloc (bloc 2a). Le reste du réseau tentera d'inclure la transaction vers Bob dans le prochain bloc (bloc 2). Lorsque le bloc 2 est miné, Alice, au lieu de changer vers la branche la plus longue de la chaine de bloc, tel que dicté par Bitcoin, continuera à miner sur son embranchement. Alice créera sur sa branche alternative les blocs 2a, 3a, 5a et 6a. Lorsque Bob validera que sa transaction soit bien incluse dans le Bloc 2, il procèdera à la vente. Lorsque la vente sera complétée, Alice annoncera tous les blocs de sa chaine alternative sur le réseau. Puisque la branche d'Alice est plus longue, les utilisateurs du réseau accepteront la chaine d'Alice et commencerons à miner des blocs basés sur l'embranchement d'Alice. Les transactions de l'autre embranchement sont retournées dans l'état *non confirmé.* Bob a donc donné un produit à Alice, puisqu'il ne recevra jamais les bitcoins de la part d'Alice. En effet, les bitcoins ont plutôt été transférés à Fred. Alice a donc réussi, en fraudant Bob, à obtenir un bien ou un service de Bob sans le payer et réutiliser ces bitcoins pour payer Fred. Ainsi, les bitcoins ont été dépensés deux fois, mais une seule des deux dépenses sera conservée dans la chaine de blocs.

Dans l'illustration précédente, même si Bob est prudent et attend 3 blocs supplémentaires après la confirmation de la transaction, il n'est pas complètement à l'abri d'un *double spending*, mais il rend la tâche beaucoup plus difficile à l'attaquant. Le principe de sécurité de Bitcoin est basé sur le principe qu'une seule personne ne peut pas contrôler plus de 50 % de la puissance de calcul du réseau. La probabilité qu'Alice puisse créer sa propre chaine alternative plus rapidement que le reste du réseau est de plus en plus faible, lorsque la longueur de la chaine à créer augmente. Cette situation est décrite par une loi probabiliste binomiale. Si la variable « q » représente le pourcentage de la puissance de calcul que détient

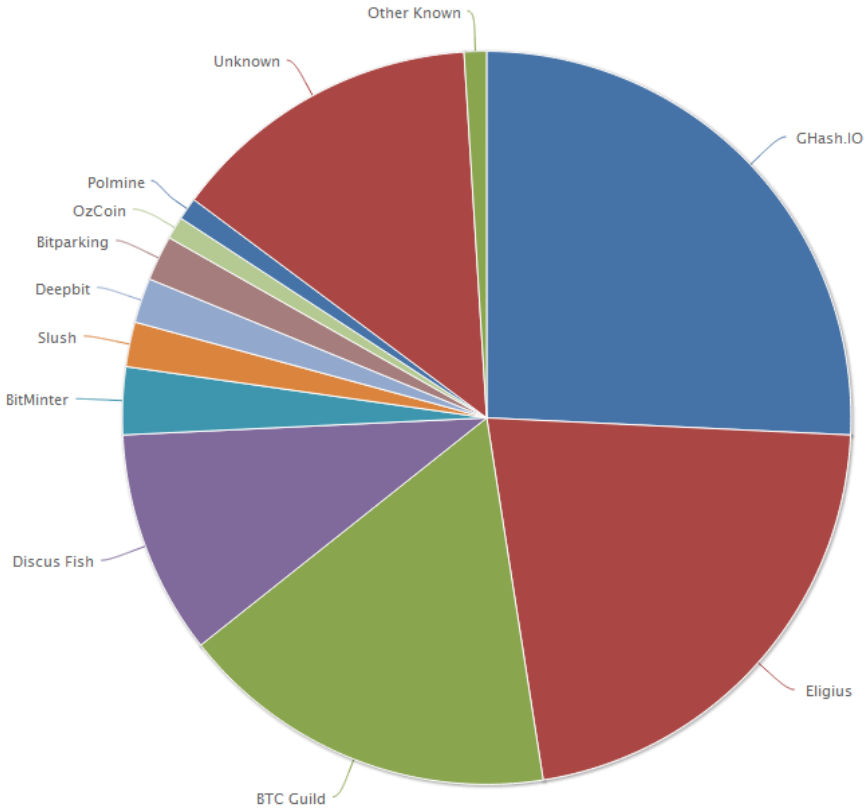
une entité et « z » le nombre de blocs que Bob attend avant de procéder à la transaction, la probabilité que Bob soit victime d'un *double spending* est décrite par la distribution suivante.

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

Si Alice détient 10 % de la puissance de calcul du réseau, Bob devrait attendre cinq blocs avant de procéder à la vente pour avoir une probabilité de *double spending* inférieure à 0.1 %. Dépendamment du risque que le vendeur est prêt à accepter, il devra attendre un certain nombre de blocs.

Le problème de *double spending* est bien réel, car certaines entités possèdent plus de 10 % de la puissance du réseau. Le graphique représente la distribution de l'ensemble de la puissance de calcul.



Référence : Blockchain.info [7]

Les *groupes de mining* Discus Fish, BTC Guild, Elgius et Ghash.IO représentent respectivement 10 %, 17 %, 22 % et 26 % des ressources du réseau Bitcoin. Si un attaquant est en mesure de contrôler les ordinateurs d'un de ces 4 *groupes*, la possibilité d'une attaque de *double spending* sera bien réelle.

B. Perte de clé privée

Le point de faiblesse le plus important pour un usager est la protection de ses clés privées. Le moyen le plus simple pour voler les bitcoins d'un utilisateur est d'obtenir sa clé privée. En effet, pour effectuer une transaction, il est nécessaire de la signer avec la clé privée de l'adresse Bitcoin pour que le réseau puisse valider la transaction avec la clé publique associée à cette même adresse. Si un attaquant réussit à copier la clé privée d'un utilisateur, il pourra signer les transactions en son nom et dépenser l'argent de l'utilisateur comme il le souhaite. Ainsi, voler la clé privée d'un utilisateur équivaut à voler l'identité de cet utilisateur.

Il n'existe aucun mécanisme de protection contre le vol de clé privée : l'utilisateur est responsable de conserver confidentiellement sa clé privée. Ainsi, lors de plusieurs attaques contre des sites web utilisant Bitcoin, les attaquants ont mis la main sur les clés privées du site web et ont transféré les bitcoins vers des adresses qui leur appartenaient. Les sites piratés n'ont eu aucun recours possible pour récupérer leurs bitcoins.

Dans cette même optique, de plus en plus de logiciels malveillants implémentent des fonctionnalités pour voler les clés privées contenues dans les portefeuilles présent sur les ordinateurs des victimes. Par exemple, le botnet *Pony* a récemment publié une mise à jour qui ajoute cette fonctionnalité. Puisque la majorité des logiciels clients Bitcoin stockent les clés privées dans un fichier nommé *wallet.dat*, le logiciel malveillant vérifie si ce fichier existe dans le répertoire par défaut. Le fichier wallet.dat n'est pas crypté par défaut et la majorité des utilisateurs n'active pas cette fonctionnalité. Ainsi, la clé privée de l'utilisateur, en texte clair, est volée par le logiciel malveillant et les fonds sont détournés.

Une bonne manière de protéger son portefeuille électronique consiste à déplacer le fichier *wallet.dat* dans un répertoire autre que le répertoire défaut ainsi que crypter son portefeuille avec un mot de passe fort.

C. *Distribution de la puissance de calcul*

Comme mentionné précédemment, la sécurité de Bitcoin repose sur un gage simple : personne ne peut contrôler plus de 50 % de la puissance de calcul du réseau. En effet, dans cette situation, un utilisateur pourrait effectuer des attaques de type *double spending* avec un taux de réussite de 100 %. Lors d'un double spending, l'attaquant peut donc choisir la branche qui sera conservée dans la chaine de bloc. Même sans avoir 50 % du réseau, cet utilisateur peut causer d'importants problèmes sur le réseau. En effet, le tableau suivant indique le nombre de blocs à attendre après une confirmation de la transaction pour s'assurer d'avoir moins de 0.1 % de chances d'être victime d'un *double spending*.

$$P < 0.001$$

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

Q est la fraction de la puissance de calcul que possède une entité sur le réseau Bitcoin et z est le nombre de blocs à attendre pour s'assurer d'être sous la probabilité de *double spending* de 0.1%.

Anonymat

Bitcoin est considéré comme anonyme puisque le seul mécanisme d'identification est basé sur la cryptographie asymétrique où les paires de clés ne sont pas générées par une entité centrale, mais bien par les utilisateurs. Un utilisateur génère donc une paire de clés et dérive son adresse Bitcoin à partir de sa clé publique. Ensuite, il pourra envoyer et recevoir de l'argent à l'adresse qu'il a lui-même générée.

Un utilisateur peut donc générer et utiliser plusieurs adresses, tant qu'il possède les clés privées pour chacune de ses adresses. Pour rester le plus anonyme possible, il est recommandé d'utiliser une adresse différente pour chacune des transactions de bitcoins. Ainsi, il est très difficile d'associer une adresse Bitcoin à un utilisateur et de voir, dans la chaine de blocs, quelles transactions cet utilisateur a faites.

Il n'est cependant pas impossible d'associer plusieurs adresses à un même utilisateur. Si un utilisateur veut effectuer une transaction et que ses avoirs sont fractionnés sur plusieurs adresses, il devra spécifier en entrée de la transaction plusieurs adresses Bitcoin d'où proviendront les fonds nécessaires. Ainsi, puisque l'utilisateur est en mesure de signer chacune des transactions précédentes pour chacune des adresses en entrée, il est possible d'associer toutes ces adresses à cet utilisateur. De cette manière, il est possible d'effectuer un graphe de toutes les transactions effectuées et d'utiliser ces données pour déterminer les adresses qui appartiennent à un même utilisateur. Dans l'exemple suivant, il est possible de déterminer que les trois adresses différentes en entrée de la transaction appartiennent au même utilisateur. Puisqu'il a réussi à signer les 3 entrées, cet utilisateur possède minimalement les trois clés privées associées à ces adresses.

Inputs[?]

Previous output (index)[?]	Amount[?]	From address[?]
70121b411ef9...:0	10.84610803	1AN7Eg3gwC2M9BnobEoR1scTW9drDvCMJC
8cdef4e63cc7...:0	11.53919337	128VjD4wVYDusSGHAecYM7sZH6W5ikYVe1
00fce5ada369...:0	59.2051576	1F6gYcNnfQjTWzGV44AGvAr6j3h98QV2BV

Référence : Bitcoin Block Explorer[1]

Preuve de concept - Implémentation

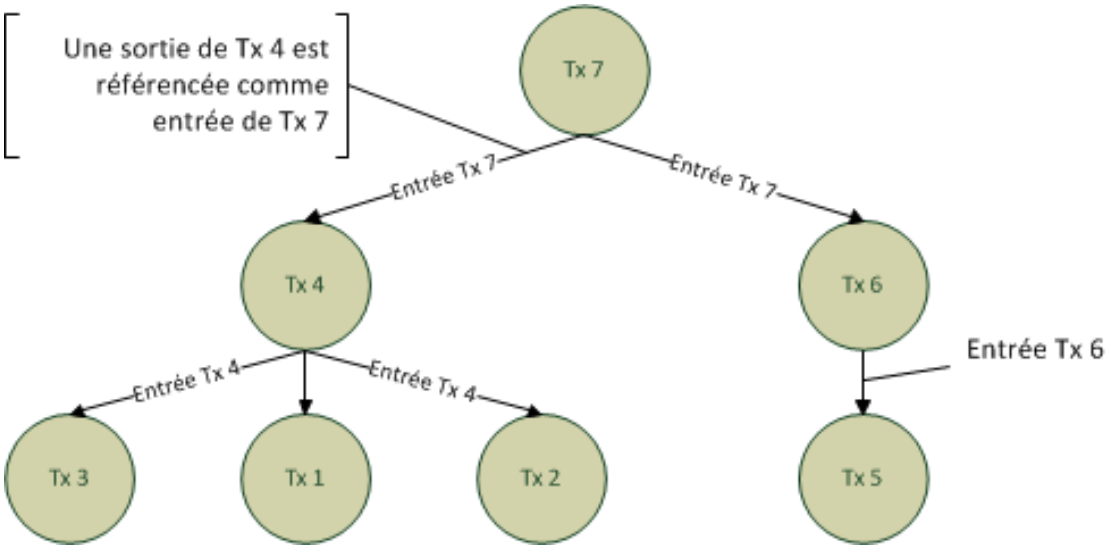
Cette section présentera de façon plus appliquée les problèmes de sécurités liés à Bitcoin. Deux problèmes seront abordés de manière plus pratique (*hands on*) pour démontrer les limitations du protocole.

Dans un premier lieu, un logiciel de parcours de graphe d'une transaction sera présenté. Ce logiciel permet d'illustrer le parcours d'une partie des bitcoins utilisés dans une transaction; de sa génération à sa dépense dans une transaction spécifiée.

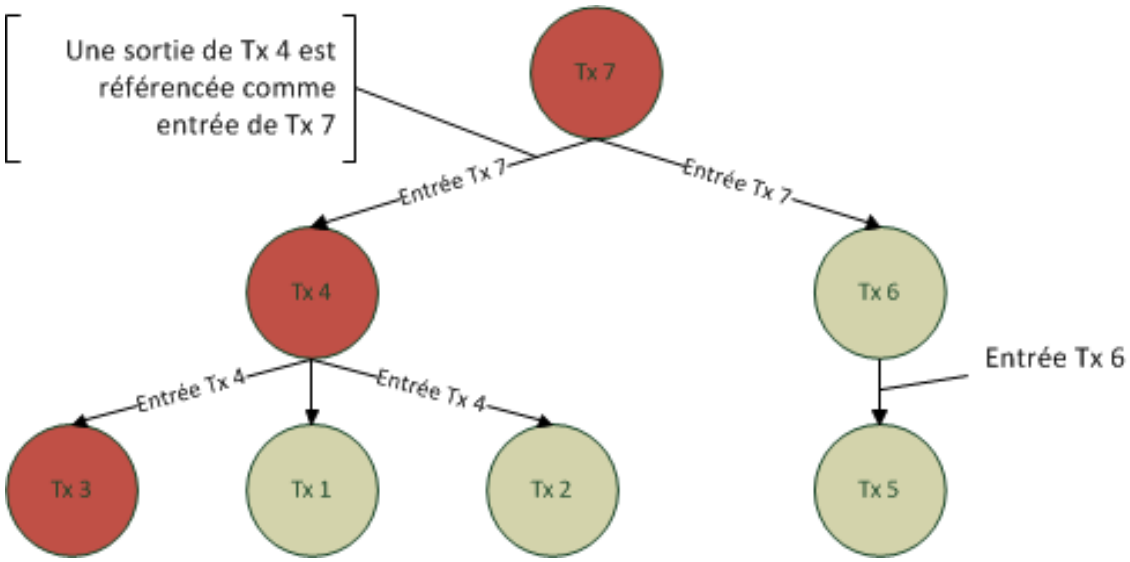
La deuxième preuve de concept est en fait une proposition de solution au problème de détection de fraude en temps réel. L'implémentation de la solution n'a pas été effectuée dans le cadre de ce projet, mais le concept ainsi qu'une analyse préliminaire sera présentée.

A. Anonymat - Graphe des transactions

Comme expliquée précédemment, une transaction est un échange de bitcoins de zéro à plusieurs adresses Bitcoin en entrée vers une adresse ou plus en sortie. La somme des bitcoins spécifiées en entrée de la transaction doit égaler la somme des bitcoins en sortie. La seule transaction où cette règle n'est pas respectée est la transaction de type *Generation* qui récompense un utilisateur qui a réussi à créer un bloc valide. Chaque entrée d'une transaction référence une sortie d'une transaction antérieure qui n'a pas encore été dépensée. Ainsi, il est possible de partir d'une transaction récente et de représenter les sorties des transactions antérieures liées à cette transaction récente. De cette façon, il est possible de faire une représentation sous forme d'arbre tel qu'illustrée par la figure suivante.



Pour la preuve de concept développée, seulement la branche telle qu'illustrée dans le graphe suivant a été suivie. Avec plus de 55 millions de transactions effectuées depuis la création de Bitcoin, il n'est pas réaliste d'itérer dans toutes les branches d'une transaction pour trouver toutes les transactions antérieures impliquées dans le graphe d'une transaction spécifiée. Ainsi, seulement la branche des sorties à l'index 0 a été suivie par l'utilitaire. Pour illustrer graphiquement le chemin parcouru, il a été illustré sur le graphe précédent. Les transactions surlignées en rouge représentent le chemin parcouru par l'utilitaire.



L'utilitaire en question permet d'explorer ce chemin et d'afficher à l'écran le chemin d'une partie des bitcoins utilisé dans la transaction spécifiée. La figure suivante est une capture d'écran de l'utilitaire durant son fonctionnement. La sortie complète de l'utilitaire est présentée à l'annexe 1.

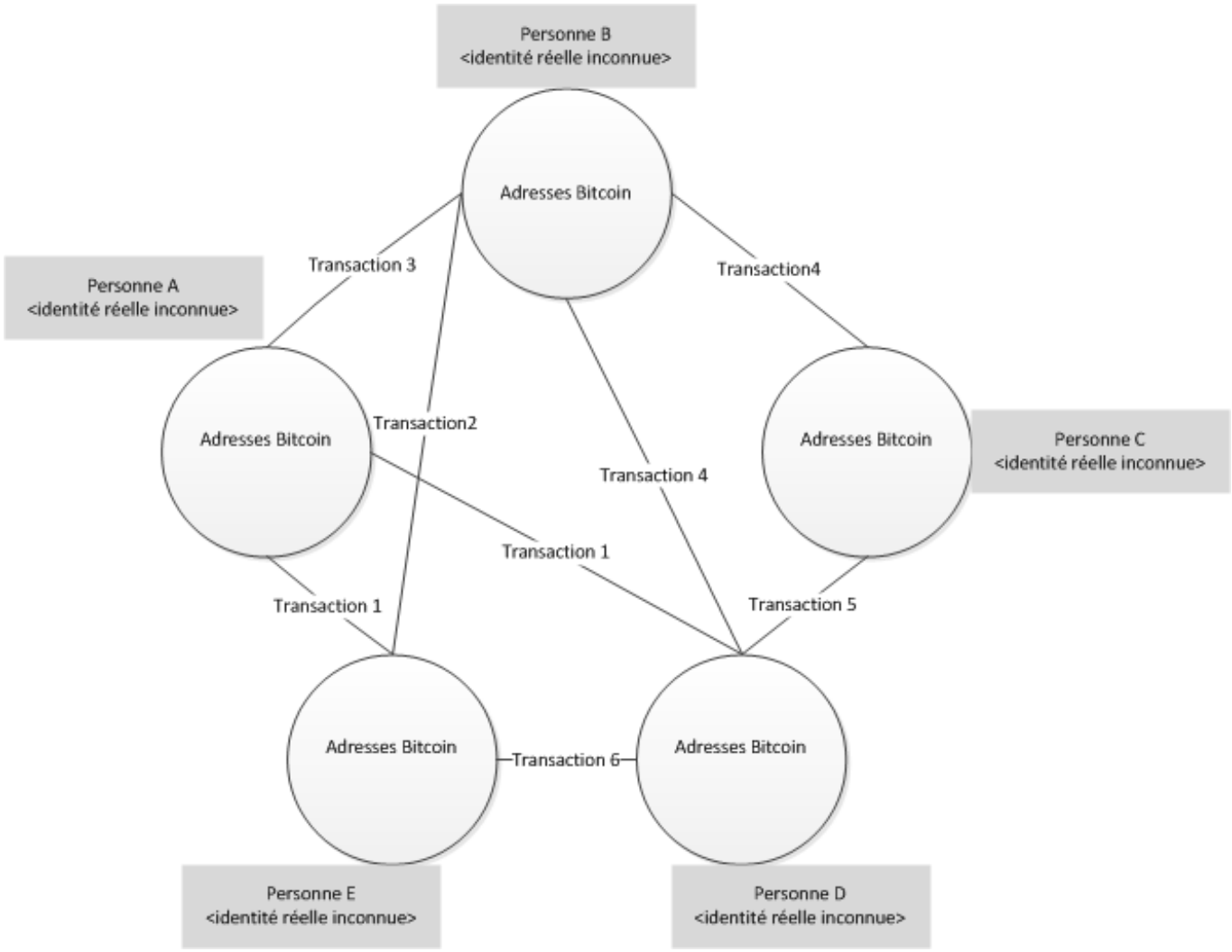
```
Enter Tx number or hash  [54690175]
10249073
Starting graph for tx number: 10249073
from 1Q7N6QRucb8i9QPRyyNJA6VXT5u7fgKSwi to 17jXiGKfd1oM79ahN2YXZvQ6HDMuAziuHa, Tx Number: 10249073
from 1vUtWheYHgYb4TVAq5BqHXndnuggK3C8j to 1Q7N6QRucb8i9QPRyyNJA6VXT5u7fgKSwi, Tx Number: 10060286
from 1Kpk33taYjJdmUvLAtAdZ6BKoL9DxXsMkH to 1vUtWheYHgYb4TVAq5BqHXndnuggK3C8j, Tx Number: 10058283
from 19rk9MwuwDaio6dAvqSSm3iHfxbVGWUJE2 to 1Kpk33taYjJdmUvLAtAdZ6BKoL9DxXsMkH, Tx Number: 10014692
from 13KU9VvTSfcu8Ccymt8YWgkqoiM8v3kfLv to 19rk9MwuwDaio6dAvqSSm3iHfxbVGWUJE2, Tx Number: 10012816
from 1E86A5E6ANEVPuayP2XLGUzsXjaxT5MbRm to 13KU9VvTSfcu8Ccymt8YWgkqoiM8v3kfLv, Tx Number: 9987197
from 1MdYC22Gmjp2ejVPCxyYjFyWbQCYTGhGq8 to 1E86A5E6ANEVPuayP2XLGUzsXjaxT5MbRm, Tx Number: 9985797
Mined by 1MdYC22Gmjp2ejVPCxyYjFyWbQCYTGhGq8, Tx Index : 9741630


 Press a key to continue :
```

L’utilitaire utilise une API web de *blockchain.info* pour obtenir les informations sur les transactions et sur la chaine de bloc. Le fonctionnement de l’utilitaire est décrit par le pseudocode suivant.

Assigner l’entrée utilisateur à tx_courante
Tant que tx_courante.entrées n’est pas vide :
    Afficher information tx_courante
    Assigner tx_courante.entrées[0] à tx_courante

Lors du parcours de la branche tel qu'illustré précédemment, il est possible de déterminer plusieurs informations importantes à propos des acteurs impliqués dans les transactions parcourues. Lorsqu'une transaction est effectuée, l'envoyeur doit signer chacune des entrées de la transaction. Ces entrées peuvent provenir de plusieurs adresses Bitcoin différentes. Pour que la transaction soit valide, l'envoyeur doit signer les entrées avec chacune des clés privées associées aux adresses Bitcoin utilisées en entrées. En d'autres mots, l'envoyeur doit prouver qu'il possède toutes les adresses d'où proviennent les fonds de la transaction. Il est donc possible d'assumer que toutes les adresses utilisées en entrée d'une transaction valide appartiennent donc toutes à une même personne.

Il est ainsi possible de regrouper des adresses ensemble et de les associer à une personne dont l'identité est inconnue. De cette façon, il est possible de représenter le réseau à l'aide d'un graphe où un nœud représente un utilisateur du réseau et où une transaction représente une arête. Chaque nœud contient une liste d'adresse Bitcoin appartenant à l'utilisateur. De cette manière, il est possible de voir graphiquement les utilisateurs actifs du réseau. Il est aussi possible d'utiliser le nombre de transactions entre deux nœuds pour déterminer les liens entre deux utilisateurs du réseau.



Il est possible pour un utilisateur du réseau d'associer les personnes desquelles il a reçu ou envoyé des bitcoins à leur adresse Bitcoin. Cet utilisateur est donc en mesure d'associer une identité réelle à certains nœuds du graphe, mais cette information n'est pas d'une grande utilité pour un simple utilisateur. Avec le peu d'information que possède un utilisateur du réseau, il ne sera pas en mesure de faire quoi que ce soit d'intéressant avec les simples bribes d'information qui possède.

Cependant, certaines organisations ont des intérêts beaucoup plus grands à connaitre l'identité réelle des utilisateurs de Bitcoin. Pour arriver à leurs fins, ces organisations utilisent plusieurs méthodes pour associer un nœud du graphe à une identité réelle. Deux méthodes intéressantes sont présentées dans l'article *Bitcoin Transaction Graph Analysis*[10] :l'exploration du web et l'utilisation de données statistiques.

La première méthode consiste à explorer le web et les forums web. Plusieurs utilisateurs s'échangent leurs adresses sur ces forums ou l'affiche simplement dans leur signature.

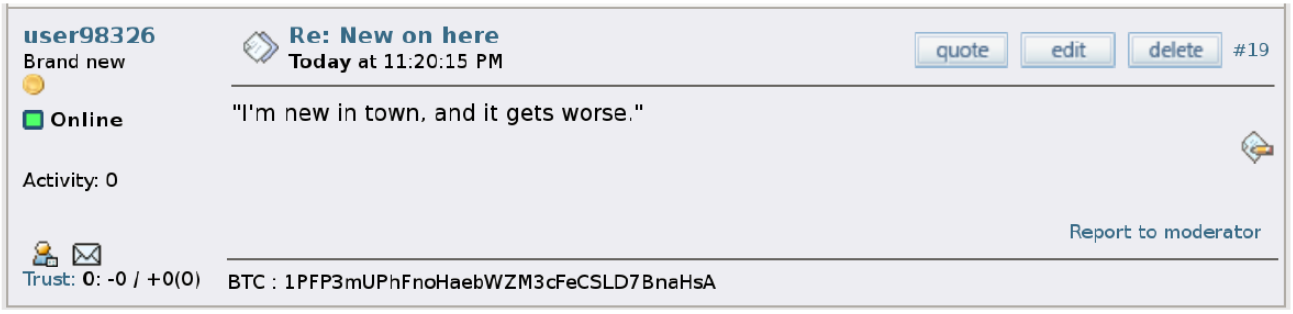


Figure 3: A typical user signature line that includes a bitcoin address for 'tipping'.

Référence : Bitcoin Transaction Graph Analysis[10]

Il est possible d'aller chercher davantage d'informations en lisant certaines conversations. Par exemple, si un utilisateur connu écrit à un autre utilisateur qu'il planifie le payer par Bitcoin à midi, il est possible de regarder les transactions de cet utilisateur et de vérifier s'il a effectivement effectué une transaction aux alentours de midi. Si la transaction est identifiée dans la chaine de blocs, il est possible d'obtenir l'identité de l'autre utilisateur impliqué dans la transaction. En corrélant une conversation sur internet avec les transactions Bitcoin, il est possible de découvrir de nouvelles identités.

L'approche statistique consiste à corréler des observations du graphe à des situations « réelles ». Par exemple, les nœuds ayant un large volume de transaction seront probablement des sites populaires dans le réseau Bitcoin. En utilisant ces données, les auteurs de l'article *Bitcoin Transaction Graph Analysis*[10] ont réussi à déterminer qu'un des nœuds ayant un important flux de transaction était en fait un site de pari Bitcoin : *legacy.satoshidice.com*.

B. Système de détection de fraudes

Dans l'article *Two Bitcoins at the Price of One? Double-Spending Attacks on Fast Payments in Bitcoin*[11] à la section 5.2, il est proposé d'utiliser des observateurs sur le réseau Bitcoin pour détecter rapidement les tentatives de *Double Spend*. Les auteurs de l'article ne proposent malheureusement pas d'implémentation concrète de leur idée.

Le concept proposé est d'ajouter des observateurs de confiance sur le réseau Bitcoin. Ces nouveaux nœuds du Réseau doivent être répartis à différent endroit dans le monde pour avoir une bonne représentation de la propagation de l'information à l'intérieur du réseau. Les nouveaux nœuds de confiance sont reliés entre elles sur un réseau de confiance. Sur ce réseau de confiance, ces nœuds partageront de l'information pour effectuer la détection de fraude en temps réel. Le diagramme de l'annexe 2 illustre une topologie réseau qui pourrait être utilisée pour ce type de système.

Les commerçants désirant vérifier une transaction pour une tentative de fraude doivent simplement utiliser un service web en spécifiant le montant, l'adresse source et une adresse de destination. Les nœuds liés par le réseau de confiance seront interrogés. Chaque nœud devra dire s'il possède une transaction dans son *memory pool* (liste des transactions qui ne sont pas encore confirmées) qui répond à ces critères :

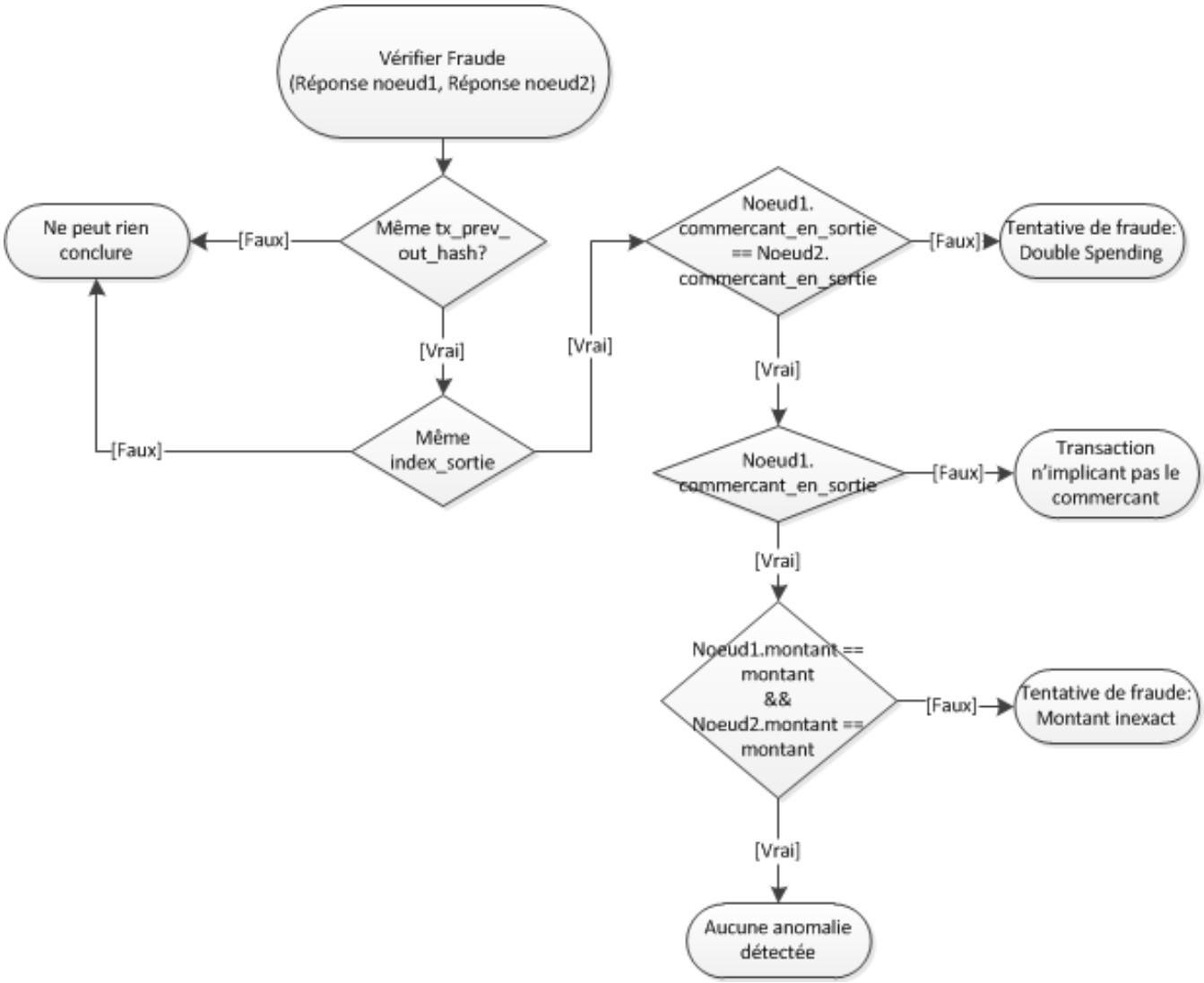
1) L'adresse de l'envoyeur est dans la liste des entrées d'une transaction.
2) Une sortie de cette transaction réfère l'adresse du commerçant
3) Cette sortie de la transaction est du bon montant.

Les réponses peuvent donc avoir ces formats :

1) NULL
2) {string tx_prev_out_hash, int index_sortie, bool commercant_en_sortie, int montant}

- Si le critère 1 n'est pas respecté, alors le nœud retourne NULL, car il n'a pas encore reçu de transaction en lien avec cette vente.
- Sinon, la transaction retournée sera celle qui respectera (en ordre de priorité) les critères {1,2 ,3}, critères {1,2} ou finalement seulement le critère 1.
- La valeur booléenne commercant_en_sortie sera vraie si les critères {1,2} sont respectés.
- La valeur du montant sera vérifiée seulement si les critères {1,2} sont respectés.

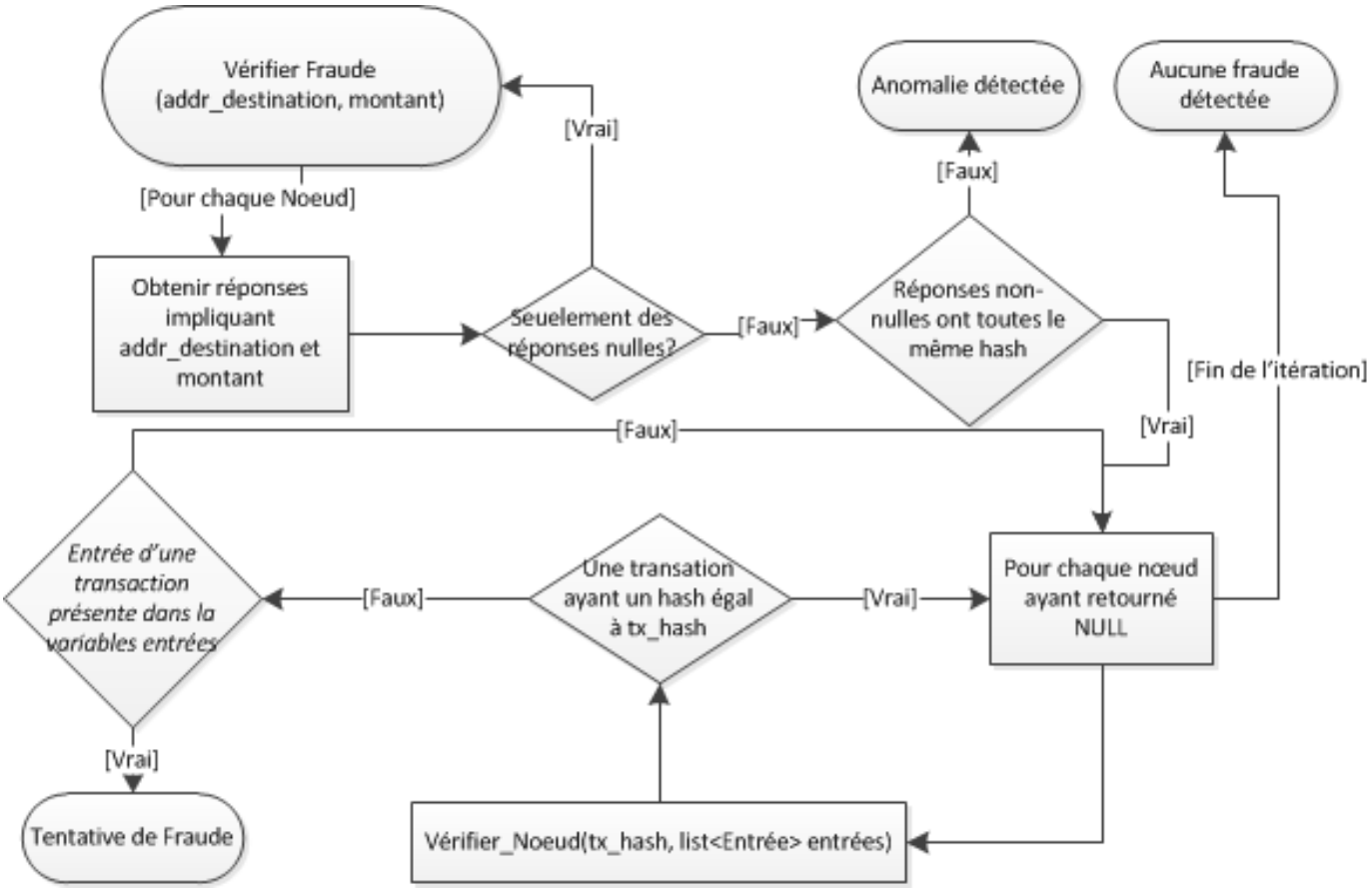
Le serveur central recueille les réponses des différents nœuds. Il recontacte les nœuds qui ont retourné NULL, jusqu'à ce que tous les nœuds aient reçu la transaction par propagation et qu'ils retournent un résultat. Lorsque le serveur central il vérifie alors la validité de la transaction. Un *double spend* consiste à essayer de dépenser deux fois la même sortie d'une transaction antérieure qui n'a pas encore été dépensée. En d'autres mots, il y a tentative de fraude si on détecte qu'une même sortie de transaction antérieure est envoyée pour certains nœuds au commerçant et que pour d'autres nœuds cette sortie n'est pas envoyée au commerçant. Il y a tentative de fraude s'il existe deux nœuds qui se contredisent. C'est-à-dire que deux nœuds ont retourné les mêmes valeurs pour tx_prev_out_hash et index_sortie, mais que la variable booléenne est différente. Pour chacune des paires de réponses des nœuds possibles, le diagramme suivant illustre le processus de validation à effectuer :



Il est recommandé pour le commerçant de générer une nouvelle adresse pour chaque vente. Il pourra ainsi conserver une association {adresse Bitcoin, numéro de vente}. De cette façon, l'API fournie pourrait être utilisée sans fournir l'adresse Bitcoin source. L'API assume que seul l'acheteur connait l'adresse Bitcoin pour la vente. Le serveur central interrogera ses nœuds de confiance pour obtenir une de ces réponses possibles :

1) NULL
   - Retournée si aucune transaction impliquant l'adresse de destination n'a été trouvée
2) {string tx_hash, list<Entree> entrees}
   - Retournée le hachage de la transaction impliquant l'adresse de destination et ses entrées.
   - . Le format de l'objet Entrée est le suivant : Objet Entree {string tx_prev_out_hash, int index_sortie}

Le diagramme de flux suivant illustre la prise de décision en fonction des différents scénarios possibles :



Si tous les nœuds retournent NULL, la transaction n'a pas encore été effectuée, le processus de vérification doit attendre que la transaction soit diffusée sur le réseau Bitcoin et recommencer la vérification ultérieurement.

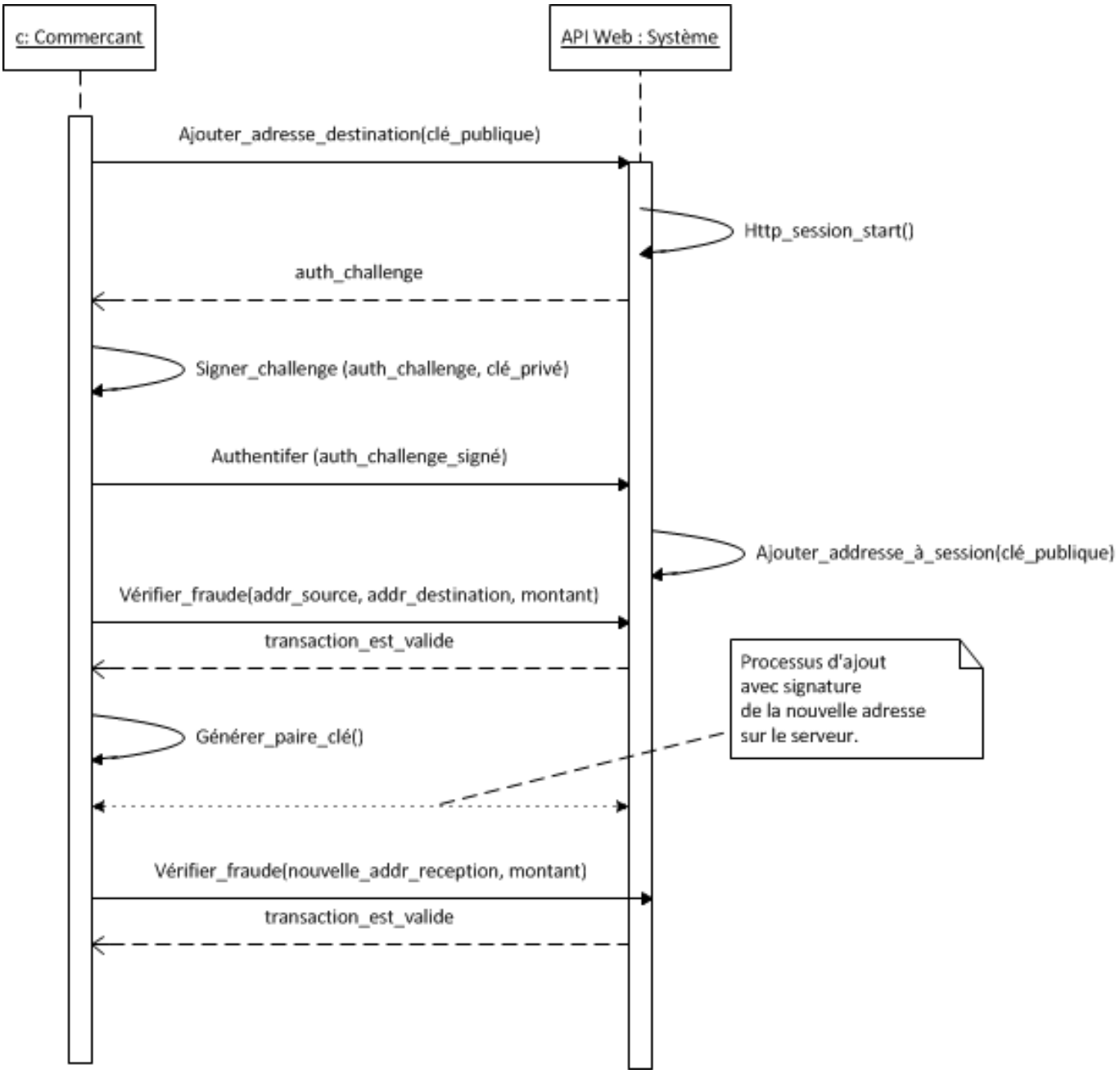
Si plusieurs hachages de transaction (tx_hash) différents sont retournés par les nœuds, le paiement de la vente est considéré comme suspicieux et cette vente sera annulée.

Pour chacun des nœuds ayant retourné NULL :

1) le hachage de la transaction ainsi que les entrées de la transaction sont passés en paramètres.
2) Si le nœud détecte une transaction non confirmée qui dépense des entrées de la liste reçue en paramètre, il vérifie le hachage de cette transaction.
3) Si le hachage est différent de celui passé en paramètre, il signale la tentative de fraude.

Si aucun des nœuds ayant préalablement retourné NULL ne signale la tentative de fraude, la transaction est considérée comme légitime.

Un dernier point important avec l'API externe est la confidentialité. Il est primordial que seul le détenteur de l'adresse de destination puisse vérifier les tentatives de fraude le concernant. Pour valider l'identité du demandeur, l'appel de l'API sera modifié. En effet, l'appel à l'API externe nécessitera de fournir la clé publique plutôt qu'une adresse de destination. Le Diagramme UML illustre le fonctionnement de la mesure de protection.



Pour pouvoir vérifier la fraude sur une transaction, l'utilisateur doit prouver qu'il possède l'adresse Bitcoin de destination qui recevra les fonds d'une vente. L'utilisateur envoie sa clé publique au serveur, celui-ci lui renvoie un défi à signer (auth_challenge). Ce défi peut être simplement une chaine bits générée aléatoirement. Ce défi sera signé par le client avec la clé privée et sera renvoyé au serveur. Le serveur pourra alors vérifier la signature avec la clé publique reçue préalablement. Si la signature est correcte, alors le serveur déterminera l'adresse Bitcoin à partir de la clé publique reçue et permettra à cet utilisateur de vérifier les tentatives de fraudes sur cette adresse Bitcoin. Lors d'un appel à Vérifier_fraude, le système vérifiera que l'utilisateur a préalablement réussi à signer le défi avec la clé privée associée avec cette adresse. En d'autres mots, le serveur s'assure que l'utilisateur a prouvé qu'il possède l'adresse de destination avant d'enclencher le processus de détection de fraude.

Les deux appels à *Vérifier_fraude* illustrent les deux approches de détection de fraude expliquées au début de cette section. Le deuxième prototype de *Vérifier_fraude* qui ne prend pas d'adresse source en paramètre nécessite de créer une nouvelle adresse de réception par vente. Pour vérifier la fraude sur cette adresse exclusive, le commerçant devra effectuer le même processus de signature pour prouver qu'il possède bien cette adresse. Dans le diagramme précédent, cette étape a été remplacée par une note pour éviter de surcharger le diagramme.

Conclusion

Le fonctionnement de Bitcoin est relativement simple, considérant qu'il s'agit d'un système qui traite des sommes d'argent importantes. La cryptographie utilisée dans le protocole est simple, efficace et reconnue. L'intégrité des transactions est assurée par un système de hachage des transactions et des blocs de transactions. La chaine de bloc permet à tout le réseau de valider et d'enregistrer ces transactions. Bref, il n'est pas possible d'altérer l'intégrité du journal des transactions (chaine de bloc).

Le seul identifiant utilisé sur le réseau est l'adresse Bitcoin qui est dérivée de la clé publique, ainsi le réseau est pseudoanonyme, même si l'historique des transactions est public. En effet, il est souvent nécessaire d'utiliser de l'information acquise à l'extérieur du réseau Bitcoin. La méthode la plus efficace pour associer un utilisateur Bitcoin à son identité réelle est d'utiliser le web. Plusieurs utilisateurs s'échangent leurs adresses sur les réseaux sociaux ou sur des forums. Il est possible d'utiliser cette information pour trouver des corrélations entre des adresses Bitcoin et des données utilisateurs (nom d'utilisateur, pseudonyme, courriel, etc.).

En résumé, Bitcoin est sécuritaire s'il est utilisé correctement par les utilisateurs. Les utilisateurs du réseau doivent connaitre certains principes du protocole pour éviter de tomber les pièges. Premièrement, un utilisateur devrait protéger sa clé privée, puisqu'elle sert à valider son identité sur le réseau. Il devrait être nécessaire pour un utilisateur de crypter son portefeuille électronique (qui contient ses clés privées) avant de pouvoir commencer à utiliser le client (logiciel) Bitcoin. Deuxièmement, l'utilisateur doit être conscient des risques et des conséquences d'une attaque de type *double spending*. Il doit au minimum savoir qu'il doit attendre la création de quelques nouveaux blocs avant de considérer la transaction comme étant complété. Connaitre les rudiments de Bitcoin et les bonnes pratiques en matière de sécurité permet à l'utilisateur d'utiliser ce système de transfert d'argent pratiquement sans risques.

Même si Bitcoin est fréquemment utilisé comme un nouveau mode de paiement pour le commerce électronique, il reste très peu utilisé par commerce ayant pignon sur rue. En effet, les délais pour la confirmation d'une transaction sont trop longs que la technologie puisse être utilisée dans les commerces. Il n'est pas possible de faire attendre un client dix minutes (un nouveau bloc) avant de pouvoir confirmer la transaction. L'idée de mettre des nœuds de confiance sur le réseau pour écouter les diffusions des

transactions sur le réseau pour détecter les fraudes n'est pas nouvelle, mais peu d'implémentations concrètes existent. Certaines compagnies, comme Coinbase, offrent des services de détection de fraude sur Bitcoin, mais le fonctionnement n'est pas documenté et connu du public. La majorité de ces services demande un compte utilisateur. La solution proposée dans ce rapport respecte davantage l'esprit de Bitcoin, car elle ne demande pas de fournir des informations personnelles pour utiliser le service. En effet, l'authentification est effectuée par un défi cryptographique qui prouve que l'utilisateur vérifie la fraude sur des transactions dans lesquelles il est impliqué.

Références

[1] Bitcoin Block Explorer. 2014. En ligne. <http://blockexplorer.com/>

[2] Bitcoin Wiki. 2014. En ligne. <https://en.bitcoin.it>

[3] Satoshi Nakamoto. 2009. « Bitcoin: A Peer-to-Peer Electronic Cash System» En ligne. 9p. < https://bitcoin.org/bitcoin.pdf> Consulté le 21 février 2014

[4] Imponderable Things (Scott Driscoll's Blog). 2013. « How Bitcoin Works Under the Hood» <http://www.imponderablethings.com/2013/07/how-bitcoin-works-under-hood.html>

[5] Trustware SpiderLabs. 2014. « Look What I Found: Pony is After Your Coins! » <http://blog.spiderlabs.com/2014/02/look-what-i-found-pony-is-after-your-coins.html>

[6] Bitcoin watch. 2014. En ligne. <http://bitcoinwatch.com/>

[7] Blockchain. 2014. En ligne. <http://blockchain.info>

[8] Coinbase. 2014. «Bitcoin Wallet - Coinbase ». En ligne. <https://coinbase.com/>

[9] Philip Koshy, Diana Koshy, Patrick McDaniel. 2014. « An Analysis of Anonymity in Bitcoin Using P2P Network Traffic » En ligne. 17p. <http://fc14.ifca.ai/papers/fc14_submission_71.pdf> Consulté le 18 avril 2014

[10] Michael Fleder, Michael S. Kester, Sudeep Pillai. 2014. « An Analysis of Anonymity in Bitcoin Using P2P Network Traffic » En ligne. 8p. <http://people.csail.mit.edu/spillai/data/papers/bitcoin-transaction-graph-analysis.pdf> Consulté le 19 avril 2014

[11] Ghassan O. Karame, Elli Androulaki. 2012. « Two Bitcoins at the Price of One? Double-Spending Attacks on Fast Payments in Bitcoin» En ligne. 17p. <http://eprint.iacr.org/2012/248.pdf> Consulté le 14 avril 2014

Annexe 1 – Anonymat : graphe des transactions – Sortie de l'utilitaire

```
Enter Tx number or hash  [54690175]
10432485
Starting graph for tx number: 10432485
from 1mKNwa56LwtHarnhLp45eJonJxD8yutCS to 16wjNVUJ5Q9m1J3uvyPztyAY4NZp67DwKm, Tx Number: 10432485
from 1L6oVRSrvUTUxdcc2KdykANbqKXWL5dmd1 to 1mKNwa56LwtHarnhLp45eJonJxD8yutCS, Tx Number: 10409843
from 1ETWSA9f7bSgoMSmoxtsEjYkhJ5jm15mTF to 1L6oVRSrvUTUxdcc2KdykANbqKXWL5dmd1, Tx Number: 10303923
from 1NxkSCLhzwgh4ykEZ2aTv71xpijWBqnj9G to 1ETWSA9f7bSgoMSmoxtsEjYkhJ5jm15mTF, Tx Number: 10302529
from 13mnxtpQpazEv6AMrveoBLkDQmXBc9MP2T to 1NxkSCLhzwgh4ykEZ2aTv71xpijWBqnj9G, Tx Number: 10302014
from 19EPpzcDQUW1xNzHBKo2XwNqGXKu36rHQL to 13mnxtpQpazEv6AMrveoBLkDQmXBc9MP2T, Tx Number: 10301585
from 197iqjQEoaRgBUFU8yG8U617pTs7hGmdhE to 19EPpzcDQUW1xNzHBKo2XwNqGXKu36rHQL, Tx Number: 10301020
from 1FkNWaDa2axDPXpiba7qiJaqwsCC8yFD3M to 197iqjQEoaRgBUFU8yG8U617pTs7hGmdhE, Tx Number: 10299270
from 16BnGdQWFAk8K9f9jDTZop99GAzNixuKV1 to 1FkNWaDa2axDPXpiba7qiJaqwsCC8yFD3M, Tx Number: 10297662
from 1E19maaoRRNps4fu5b8gVtFhZmrw5Jwky1 to 16BnGdQWFAk8K9f9jDTZop99GAzNixuKV1, Tx Number: 10296605
from 1Mf98CUFCKp1ZZ3kKymuRTQHUerm94yc4i to 1E19maaoRRNps4fu5b8gVtFhZmrw5Jwky1, Tx Number: 10295824
from 1PswyzvBrS7UttgXdCrs7qJ2kGj4hRob1M to 1Mf98CUFCKp1ZZ3kKymuRTQHUerm94yc4i, Tx Number: 10295016
from 17vmyaNBxutR17wQMgT7QUC5exda9YvYy1 to 1PswyzvBrS7UttgXdCrs7qJ2kGj4hRob1M, Tx Number: 10293980
from 1ZUgRCCHnXmgo5fFxpzrKsxcXnXnKnVLx to 17vmyaNBxutR17wQMgT7QUC5exda9YvYy1, Tx Number: 10293137
from 1363deXwSCNfYAaLGUYRhTepUhbjtnj3rS to 1ZUgRCCHnXmgo5fFxpzrKsxcXnXnKnVLx, Tx Number: 10291156
from 1JcAyKpQoxDgsWe8W7ndAzpQfktwxrdZd6 to 1363deXwSCNfYAaLGUYRhTepUhbjtnj3rS, Tx Number: 10287169
from 1AcuxKs8gKxQtoyjdcS3eJ9P7RZaED5TuK to 1JcAyKpQoxDgsWe8W7ndAzpQfktwxrdZd6, Tx Number: 10281680
from 16rivWj6t6L6xSiKuMjXdSAQCjp6zgtTi1 to 1AcuxKs8gKxQtoyjdcS3eJ9P7RZaED5TuK, Tx Number: 10280714
from 19g4LfVxx2UVF2aBkyRrtEjhTg4dUCn6EK to 16rivWj6t6L6xSiKuMjXdSAQCjp6zgtTi1, Tx Number: 10279878
from 13TKGBL2X92UYU2oDtZwEgPoxr5qEykiSS to 19g4LfVxx2UVF2aBkyRrtEjhTg4dUCn6EK, Tx Number: 10265796
from 1MVggmNkCkorvZx93Qyjv1waUA6DQ6gG8a to 13TKGBL2X92UYU2oDtZwEgPoxr5qEykiSS, Tx Number: 10251303
from 17jXiGKfd1oM79ahN2YXZvQ6HDMuAziuHa to 1MVggmNkCkorvZx93Qyjv1waUA6DQ6gG8a, Tx Number: 10249988
from 1Q7N6QRucb8i9QPRyyNJA6VXT5u7fgKSwi to 17jXiGKfd1oM79ahN2YXZvQ6HDMuAziuHa, Tx Number: 10249073
from 1vUtWheYHgYb4TVAq5BqHXndnuggK3C8j to 1Q7N6QRucb8i9QPRyyNJA6VXT5u7fgKSwi, Tx Number: 10060286
from 1Kpk33taYjJdmUvLAtAdZ6BKoL9DxXsMkH to 1vUtWheYHgYb4TVAq5BqHXndnuggK3C8j, Tx Number: 10058283
from 19rk9MwuwDaio6dAvqSSm3iHfxbUGWUJE2 to 1Kpk33taYjJdmUvLAtAdZ6BKoL9DxXsMkH, Tx Number: 10014692
from 13KU9UvTSfcu8Ccymt8YWgkqoiM8v3kfLv to 19rk9MwuwDaio6dAvqSSm3iHfxbUGWUJE2, Tx Number: 10012816
from 1E86A5E6ANEUPuayP2XLGUzsXjaxT5MbRm to 13KU9UvTSfcu8Ccymt8YWgkqoiM8v3kfLv, Tx Number: 9987197
from 1MdYC22Gmjp2ejUPCxyYjFyWbQCYTGhGq8 to 1E86A5E6ANEUPuayP2XLGUzsXjaxT5MbRm, Tx Number: 9985797
Mined by 1MdYC22Gmjp2ejUPCxyYjFyWbQCYTGhGq8, Tx Index : 9741630


 Press a key to continue :

All those addressed belong to the same persone :
1HJENkRZ4jCKMosNUXPSrLPkHRKqjqjqDr, 1mKNwa56LwtHarnhLp45eJonJxD8yutCS

All those addressed belong to the same persone :
12QdNDShREbTWFk1yRLvxXuURmrcFQY2yw, 147WqBcXcUeBDzXWgTjYw5CybPrpQRQxVb, 1Buf5zuiQ1ZNWpyHUmr5GmzvtLQz
1P8JUE, 1HtyQrvk7bfNfP3KX29gfGgrTd7Q98YmQb, 1L6oVRSrvUTUxdcc2KdykANbqKXWL5dmd1, 1MhoMAKtdtxw4NzGZPjg
YrHJQZfPEiFMpM

All those addressed belong to the same persone :
1ETWSA9f7bSgoMSmoxtsEjYkhJ5jm15mTF, 1KkykkJHgnUsA9gRd4d25uWHdY6nDPVq9q
```

Annexe 2 - Système de détection de fraudes : Diagramme réseau