Comprendre Bitcoin
Rapport de projet INFRES357

Pierre Galland, Benoît de Laitre

19 avril 2015

Table des matières

1 Le protocole Bitcoin

1.1 Briques de base

Le protocole de Bitcoin utilise plusieurs briques de bases de la cryptographie : le chiffrement asymétrique, la signature et le hachage. C'est d'ailleurs là que réside l'intérêt de ce protocole, c'est un assemblage astucieux de briques qui existent depuis de nombreuses années et qui sont très bien connues, pourtant cet assemblage crée une technologie totalement nouvelle.

1.1.1 Chiffrement asymétrique

Le principe du chiffrement asymétrique est que les clefs vont par paire, une clef publique et une clef privé (K_{Pub}, K_{Pri}). On peut voir les clefs comme des suites finies de caractères ou de chiffres, et les message aussi. Il est possible de chiffrer un message avec la clef privé et alors le message chiffré ne pourra être déchiffré qu'avec la clef publique correspondante. De même on peut chiffrer un message avec la clef publique et alors il ne pourra être déchiffré qu'avec la clef privé correspondante. Considérons l'exemple classique de Bob et Alice, ils possède chacun une paire clef publique/clef privée. La paire de clefs de Bob est (K^B_{Pub}, K^B_{Pri}) et celle d'Alice est (K^A_{Pub}, K^A_{Pri}).

Si Bob veut envoyer un message $mess$ à Alice qu'elle seule pourra lire, alors il chiffre son message avec le clef publique d'Alice, et il obtient le message chiffré $*mess*$:

$$chiffrer(mess,\ K^A_{Pub}) \rightarrow *mess*$$

Il envoie alors ce message $*mess*$ à Alice, qui quand elle le reçoit le déchiffre avec sa clef privée K^A_{Pri} :

$$déchiffrer(*mess*,\ K^A_{Pri}) \rightarrow mess$$

Si Alice essayait de déchiffrer le message $*mess*$ avec une autre clef que sa clef privée K^A_{Pri} alors cela ne marcherait pas, elle obtiendrait n'importe quoi (une suite de symbole qui n'a rien à voir avec le message $mess$). Donc comme on considère qu'Alice est la seule à connaître sa clef privée, elle est la seule à pouvoir déchiffrer le message.

$$déchiffrer(*mess*,\ K^{autre}_{Pri}) \rightarrow n'importe\ quoi$$

On peut également utiliser la paire clef publique/clef privée dans l'autre sens. Si Bob chiffre un message avec sa clef privée K^B_{Pri} alors on ne pourra le déchiffrer qu'avec sa clef publique K^B_{Pub}. En essayant de le déchiffrer avec une autre clef K^{autre}_{Pub} on obtiendrait n'importe quoi.

Ainsi si Alice veut que Bob prouve qu'il est bien Bob donc qu'il connaît la clef privée K^B_{Pri}, sans qu'il ait à révéler cette clef, alors elle peut créer un message $mess2$ et demander à Bob de le chiffrer avec sa clef K^B_{Pri} et de lui envoyer le résultat $*mess2*$. Alice va ensuite déchiffrer $*mess2*$ avec la clef publique de Bob K^B_{Pub} et si elle retombe bien sur $mess2$ cela prouve que Bob est bien Bob (qu'il connaît la clef K^B_{Pri}). Ce mécanisme sera très utile pour la signature, décrite plus loin.

1.1.2 Hachage cryptographique

On considère toujours nos messages comme des suites finies de caractères ou de chiffres (d'ailleurs in fine sur un ordinateur toute donnée est une suite finie de chiffres). Le but d'une fonction de hachage cryptographique h est de transformer chaque message de taille inférieure à T (où T est très très grand) en un message de taille beaucoup plus petite que T (par exemple dans Bitcoin en message de longueur 256 chiffres). Si j'ai un message $mess$ assez long alors $h(mess)$ sera beaucoup plus petit que $mess$. On appellera $h(mess)$ le hash de $mess$.

Prenons un exemple avec la fonction de hachage SHA-256, qui est utilisée dans Bitcoin. Je considère le message $mess$ suivant et je calcule son $h(mess)$:

$$mess = Un\ message\ pas\ très\ long,\ juste\ pour\ faire\ un\ exemple$$

$$h(mess) \rightarrow 6423e8584411fee28e5064799d8a230c64f999c448c769ab7d309baba9a33f42$$

Le but de la fonction de hachage est également que le hash d'un message puisse l'identifier de manière presque unique. Ainsi à priori si je considère deux messages différents alors leurs hashs sont également différents.

$$mess1 \neq mess2 \Rightarrow presque\ toujours\ h(mess1) \neq h(mess2)$$

Un autre point important est que si je connais seulement le hash du message $h(mess)$ ainsi que la fonction de hachage h mais que je ne connais pas le message $mess$ alors je ne puisse pas retrouver quel est le message $mess$. Il est impossible (cela demande trop de puissance de calcul) de retrouver le message à partir de son hash.

$$h\ et\ h(mess) \nrightarrow mess$$

Enfin si je considère un message $mess1$ et ma fonction de hachage h, il m'est impossible (une fois encore cela demande trop de puissance de calcul) de trouver un message $mess2$ tel que $h(mess1) = h(mess2)$.

Les fonctions de hachage jouent un rôle très important dans le protocol Bitcoin !

1.1.3 Signature

Le but de la signature électronique est de s'assurer que le message que l'on reçoit a bien été envoyé par la personne qui signe, et que le message n'a pas été modifié entre le moment de la signature et sa réception. Reprenons l'exemple de Bob et Alice, avec leurs paires de clefs respectives (K^B_{Pub}, K^B_{Pri}) et (K^A_{Pub}, K^A_{Pri}). Pour qu'Alice puisse être certaine que le message qu'elle reçoit de Bob est bien de lui et qu'il n'a pas été modifié en cours de route il faut que Bob signe son message :

Après avoir écrit son message $mess$, Bob calcul son hash $h(mess)$

$$mess \rightarrow h(mess)$$

Puis il utilise chiffre ce hash avec sa clef privée :

$$chiffrer(h(mess),\ K^B_{Pri}) \rightarrow *h(mess)*$$

Ce $*h(mess)*$ constitue la signature du message. Bob envoie à Alice le message avec sa signature : $(mess,\ *h(mess)*)$. Lorsque Alice reçoit le message et la signature elle vérifie que la signature est valide et qu'elle provient bien de Bob.

Alice reçoit donc $(mess1,\ *h(mess)*)$, on écrit $mess1$ car pour le moment on ne sait pas s'il s'agit bien du message original de Bob (il a peut-être été modifié par quelqu'un l'ayant intercepté). Alice calcule le hash de $mess1$:

$$mess1 \ \rightarrow \ h(mess1)$$

Puis elle déchiffre $*h(mess)*$ avec la clef publique de Bob. En le déchiffrant avec la clef publique de Bob K^B_{Pub} elle s'assure que $*h(mess)*$ a bien été chiffré avec la clef privé de Bob K^B_{Pri}. En effet s'il avait été chiffré avec une autre clef privée alors en le déchiffrant avec K^B_{Pub} on obtiendrait n'importe quoi. Comme à priori on n'est pas encore sûr que le $*h(mess)*$ que l'on déchiffre a bien été chiffré avec la clef privée de Bob, on note $?h(mess)?$ ce que l'on obtient :

$$déchiffrer(*h(mess)*,\ K^B_{Pub}) \ \rightarrow ?h(mess)?$$

Enfin on compare $h(mess1)$ et $?h(mess)?$, s'ils sont égaux alors on est sûr (modulo la solidité de la fonction de hachage et de l'algorithme de chiffrement) que le message est bien de Bob et qu'il n'a pas été modifié, s'ils sont différents alors on ne peut pas faire confiance à ce message.

On peut se convaincre que c'est la cas en reprenant le processus et en imaginant qu'un attaquant modifie le message et/ou la signature en cours de route, ou qu'il essaie d'envoyer le message avec sa paire de clefs. On verra qu'à la fin le test que fait Alice donne bien $h(mess1) \ \neq \ ?h(mess)?$ (en gardant à l'esprit qu'Alice connaît la clef publique de Bob).
En revanche si un attaquant connaît la clef privée de Bob alors il peut très bien se faire passer pour lui !

1.2 Une transaction Bitcoin

1.2.1 Identité et adresses

Dans le protocole Bitcoin la notion d'identité repose sur le chiffrement asymétrique donc un individu est une paire clef publique/clef privée. Sachant que l'on peut générer la clef publique à partir de la clef privée, un individu est une clef privée. On parle ici d'individus dans le protocole, car une personne réelle peut très bien avoir plusieurs clefs privées et donc correspondre à plusieurs individus dans le protocole. C'est d'ailleurs normalement le cas. Chaque personne réelle possédant des bitcoins a au moins une centaine de clefs privées, donc correspond à de nombreux individus théoriques dans le protocole.
Les individus sont identifiés dans le protocole par leur adresse. Une adresse est un hash de clef publique. Par exemple 3J98t1WpEZ73CNmQviecrnyiWrnqRhWNLy est une adresse. Pour vérifier qu'une adresse $addr$ correspond bien à la clef publique K_{Pub} on a juste à vérifier que $h(K_{Pub}) \ == \ addr$ où h est la fonction de hachage définie par le protocole.

1.2.2 Structure d'une transaction

Tout d'abord on va considérer qu'il existe un "livre de compte" partagé entre tous les utilisateurs de Bitcoin et que chaque transaction valide est inscrite dans ce livre de compte de manière chronologique. On verra par la suite quel procédé permet l'existence de ce livre de compte, qu'on

appelle la Blockchain.

Chacun peut donc accéder à l'historique de toutes les transactions valides qui ont été effectuées depuis la création de Bitcoin. Il est important de comprendre qu'il n'existe pas de pièce ou de billet de bitcoin, même virtuels. Il n'y a pas de fichiers qui représentent des bitcoins. C'est seulement la Blockchain (le livre de compte partagé) qui indique le nombre de bitcoins associé à chaque adresse. Même cette vision n'est pas tout à fait exacte. La Blockchain n'est que l'historique de toutes les transactions valides effectuées. Il est donc plus correct de dire qu'on peut calculer le nombre de bitcoin de chaque adresse à partir de la Blockchain.

Prenons un exemple : on suppose que Bob veut envoyer 1 btc à Alice (on utilisera l'abréviation btc pour désigner l'unité de monnaie bitcoin). Bob et Alice ont toujours chacun une clef privée, respectivement K^B_{Pri} et K^A_{Pri}, et ils ont également une adresse bitcoin : $addr_B = h(K^B_{Pub})$ et $addr_A = h(K^A_{Pub})$.

Pour que Bob puisse envoyer 1 btc, il faut qu'il dispose d'au moins 1 btc. Cela signifie qu'il faut qu'il existe dans la Blockchain une transaction qui envoie au moins 1 btc à l'adresse $addr_B$. Supposons que ce soit le cas, il y a dans la Blockchain une transaction Tx_1 qui envoie 1 btc à l'adresse $addr_B$. Bob va donc retransférer vers Alice l'argent qu'il a reçu de Tx_1. La transaction Tx de Bob contient donc les éléments suivants :

- Input
  - ⋆ $h(Tx_1)$
  - ⋆ K^B_{Pub}
  - • $signature(K^B_{Pri}, Tx^*)$
- Output
  - ⋆ value : 1 btc
  - ⋆ $addr_A$

FIGURE 1 – Transaction Tx'

La transaction contient donc 5 champs :

Dans la partie Input la transaction indique quelle transaction précédente est utilisée pour récupérer les bitcoins. Ici c'est la transaction Tx_1, c'est pour cela que le premier champ est $h(Tx_1)$. Le deuxième champ est la clef publique de Bob. En effet Bob veut utiliser les bitcoins donnés à $addr_B$ par Tx_1, il doit donc prouver qu'il possède bien l'adresse $addr_B$. Pour cela il fournit K^B_{Pub} et $signature(K^B_{Pri}, Tx^*)$. On peut donc vérifier d'une part que $h(K^B_{Pub}) == addr_B$, donc que l'adresse $addr_B$ correspond bien à la clef K^B_{Pub}, et d'autre part que $signature(K^B_{Pri}, Tx^*)$ est valide, donc que celui qui émet le message possède bien la clef privée K^B_{Pri}. Ce qui est signé est Tx^* c'est à dire la transaction Tx sans la signature (les éléments précédés d'une étoile sur la figure).

Dans la partie Output Bob indique qu'il souhaite donner 1 btc (champ value) à l'adresse $addr_A$, soit à Alice.

1.2.3 Sécurité de la transaction

Une fois que Bob a écrit sa transaction, il la diffuse sur le réseau Bitoin afin qu'elle soit intégrée dans la Blockchain et devienne ainsi une transaction valide, faisant partie de l'historique des transactions Bitcoin. Nous verrons par la suite que le réseau Bitcoin est un réseau peer-to-peer, donc lorsque Bob diffuse sa transaction, n'importe qui peut y avoir accès. La transaction est diffusée en clair sur le réseau.

Cependant pour pouvoir être intégrée dans la Blockchain une transaction doit être valide. Tous les champs seront vérifiés, il sera également vérifié que l'output de la transaction précédente (Tx_1 dans notre exemple) est bien dirigé vers l'adresse émettrice de la nouvelle transaction ($addr_B = h(K^B_{Pub})$ dans notre exemple), et qu'il est suffisant (supérieur à 1 btc dans notre exemple).

Il est donc primordial que si un attaquant entre en possession de la transaction il ne puisse pas la modifier afin par exemple de s'attribuer les fonds plutôt qu'à Alice, ou de changer le montant de la transaction, etc. Explorons quelques tentatives d'attaques et les raisons pour lesquelles elles échouent :

Changer le destinataire d'une transaction

Imaginons le scénario suivant : Bob a créé sa transaction, il la diffuse sur le réseau, en particulier Eve obtient une copie de la transaction. Eve veut essayer de modifier la transaction pour s'attribuer le 1 btc que Bob envoie à Alice, puis rediffuser cette nouvelle transaction modifiée sur le réseau, en espérant qu'elle soit intégrée dans la Blockchain. Eve possède l'adresse $addr_E$ et la clef privée K^E_{Pri}.

Eve fait un premier essai, elle crée la transaction Tx' en remplaçant l'adresse $addr_A$ par son adresse $addr_E$ dans l'output :

- Input
  - ⋆ $h(Tx_1)$
  - ⋆ K^B_{Pub}
  - • $signature(K^B_{Pri}, Tx^*)$
- Output
  - ⋆ value : 1 btc
  - ⋆ $addr_E$

FIGURE 2 – Transaction Tx'

Mais comme la signature $signature(K^B_{Pri}, Tx^*)$ est faite sur Tx^* et que Eve a changé un champs faisant partie de Tx^*, la signature n'est plus valide. Par conséquent cette transaction ne pourra jamais être acceptée dans la Blockchain. Dans un deuxième essai, Eve décide donc de changer la signature. Cependant comme elle ne connaît pas la clef privée de Bob, elle ne peut pas signer avec. Elle décide donc de signer avec sa clef privée K^E_{Pri}, mais à ce moment là elle doit aussi changer le deuxième champs qui contient K^B_{Pub} sinon la signature sera reconnue comme invalide. Elle va donc remplacer K^B_{Pub} par sa clef publique. Elle crée ainsi une transaction Tx''

- Input
  - ⋆ $h(Tx_1)$
  - ⋆ K^E_{Pub}
  - • $signature(K^E_{Pri}, Tx''^*)$
- Output
  - ⋆ value : 1 btc
  - ⋆ $addr_E$

FIGURE 3 – Transaction Tx''

Cette fois-ci la signature dans Tx'' est valide, car Eve a signé sur Tx''^*. Cependant il subsiste un problème et la transaction ne sera pas acceptée par la Blockchain. En effet cette transaction veut utiliser les fonds fournis à l'adresse $addr_B$ par la transaction Tx_1, or il va être vérifié que c'est bien la clef publique derrière $addr_B$ qui utilise ces fonds. On vérifiera donc que $h(K^E_{Pub}) == addr_B$ ce qui n'est pas le cas. Si Eve laissait la clef K^B_{Pub} dans le deuxième champs alors c'est la signature qui serait invalide cette fois, puisqu'elle n'est pas faite avec la clef privée correspondant à K^B_{Pub}. Eve est donc bloquée, elle ne peut pas détourner la transaction de Bob !

Changer le montant d'une transaction

Cette fois-ci Eve veut essayer de changer le montant de la transaction, le remplacer par 0.5 btc par exemple. Mais cette attaque échouera également car le champs value fait parti de Tx^* et est donc signé avec K^B_{Pub}, on ne peut pas le changer sans rendre la signature invalide. Or on a vu dans l'exemple précédent que l'on ne peut pas changer la signature. Cette attaque échoue donc également.

1.3 Blockchain et mineurs

1.4 La parade au double-spend

Références

[1] Bitcoin stackexchange. `http://bitcoin.stackexchange.com`.

[2] Bitcoin wiki. `https://en.bitcoin.it/wiki`.