Comprendre Bitcoin
Rapport de projet INFRES357

Pierre Galland, Benoît de Laitre

19 avril 2015

Table des matières

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 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.

1.2 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 !

1.3 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 !

2 Un paiement Bitcoin

3 Diffuser les paiements

4 Empêcher le double-spend

5 Le minage