Faculté des Sciences & Techniques

Université de Limoges

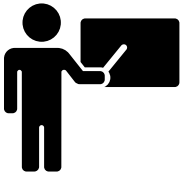
Master 1ère année

Blockchain et Crypto-monnaie — P-F. Bonnefoi

Version du 9 mai 2020

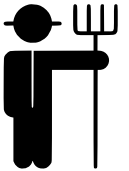
Monnaie ?

Chaîne de confiance ?



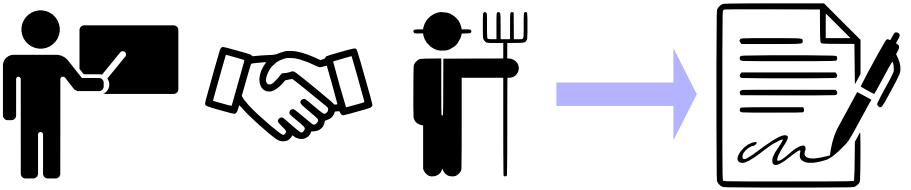
Imaginons que vous soyez plutôt bon à expliquer les maths (pour l'informatique c'est plus dur sans ordinateur).
Votre voisin, qui est fermier, vous demande d'enseigner à ses enfants les mathématiques.

Pour vous récompenser de vos efforts, il vous offre un **poulet**.

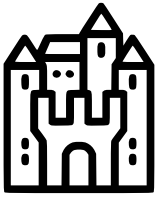


Vous pouvez enseigner à plusieurs étudiants **simultanément**.
D'autres fermiers vous demandent d'enseigner les maths à leurs enfants.
Vous avez de **plus en plus d'étudiants** et de **plus en plus de poulets**...

Vous ne pouvez **plus manger assez de poulet à la fois**, mais vous ne voulez pas arrêter d'enseigner, il est temps de **reporter l'obtention du poulet** à plus tard, par exemple pour les jours où vous n'enseignerez pas.

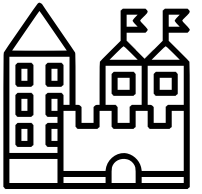


Bon pour un poulet.
Signé : le prof & le fermier



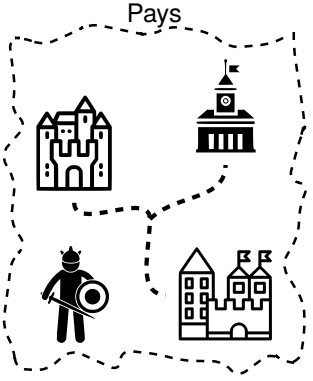
Vous vous rendez compte que maintenant grâce aux **bons** que vous accumulez vous pouvez les échanger contre d'autres biens.
Dans votre ville tout le monde se connait, et il est facile de vérifier que le **bon est valable** et quand une personne se présente avec un de vos bons, le fermier donne un poulet au porteur sans hésitation. S'il a un doute, il vous demande de **confirmer** si vous avez bien donné ce bon à la personne qui réclame le poulet.

Maintenant vous aimeriez voyager dans une autre ville, mais malheureusement vos bons ne servent à rien là bas, car personne ne vous connait, ni vos bons, ni le fermier qui les a signé.



Une solution serait de trouver un voyageur provenant de cette ville avec lequel pouvoir échanger vos bons contre les siens, mais comment lui faire confiance...

Les villes décident de s'unir et de **former un pays** !



Un **gouvernement** est établi :

- ▷ il établit des **règles partagées** entre les différentes villes qui composent le pays ;
- ▷ il dispose de **représentants** dans chaque ville ;
- ▷ il lève un **impôt** pour s'occuper des routes entre les villes, garantir son autorité, rétribuer ses représentants *etc.*
- ▷ il unifie le système de bons et les transforme en **argent**.
  Cet **argent** est accepté par les habitants du pays car :
  - ◇ les **impôts en nature** sont convertibles en argent ;
  - ◇ il existe en **quantité limitée** et ne peut être contrefait ;
  - ◇ **tant qu'il peut servir à payer les impôts, il possède une valeur**.

Le **gouvernement assure les échanges** entre les habitants du pays, et ils font confiance à sa **bonne gestion**.
Est-ce que la **confiance** régnera ? Est-ce que **l'argent** sera toujours lié à un **échange** ?

Certains habitants avaient néanmoins envie de pouvoir s'échanger des bons directement entre eux, sans passer par un tiers de confiance.

Qu'est-ce qu'un **bon** ? l'enregistrement d'une transaction, d'un accord entre deux individus de s'échanger un bien contre un service.

Comment rendre le bon **valable** entre deux villes ?
- ▷ donner une **identité unique** à chaque habitant ;
- ▷ permettrent aux habitants de **partager** toutes les transactions réalisées ;

Mais...
Si je **créais des bons** avec **d'autres bons** ?
Si j'échangeais plusieurs bons pour un poulet contre un bon pour un cheval ?

Comment changer un bon pour un autre ? Comment les combiner ? Comment suivre tous ces échanges ?
⟹ On les **enregistrant tous** depuis le début !

Comment être sûr que les échanges ont **bien été autorisés** ?
⟹ En **signant** chaque transaction par l'individu qui la réalise !

Comment **démarrer** le système ? Comment le **protéger** ? Comment **inciter** à son usage ?
⟹ En garantissant sa **transparence**, son **unicité** et créant un **système de récompense** pour ceux qui le soutiennent.

Crypto-monnaie ?

blockchain ?

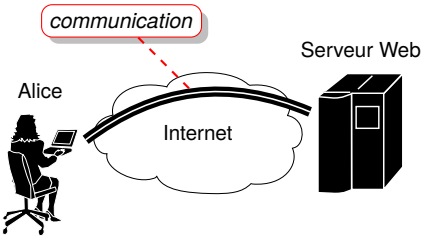
Cryptographie ?

Sécurité ?

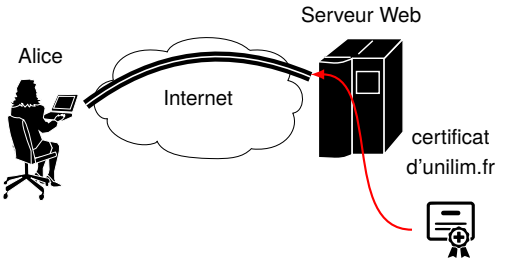
Certificats ?

La connexion httpS : un «*tiers de confiance*» est nécessaire

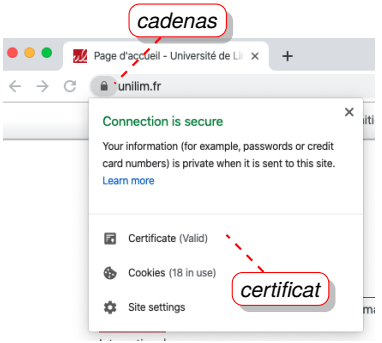
Alice veut consulter le site de l'Université de Limoges :
Elle se connecte à https://www.unilim.fr/

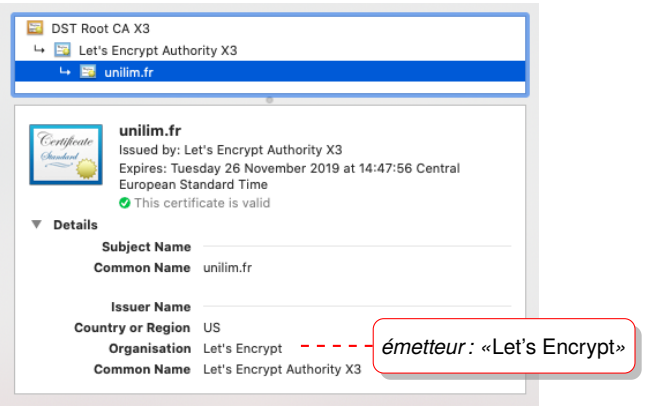


Dans la communication, le serveur Web envoie son «*certificat*» :



Dans son navigateur, elle voit un cadenas qui lorsqu'elle clique dessus, lui indique que le «*certificat*» est valide :



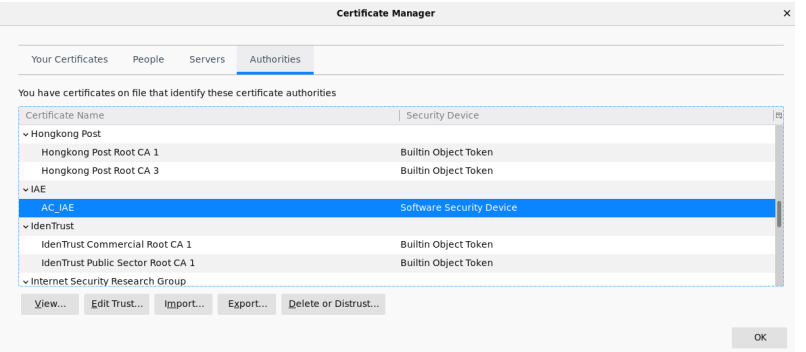


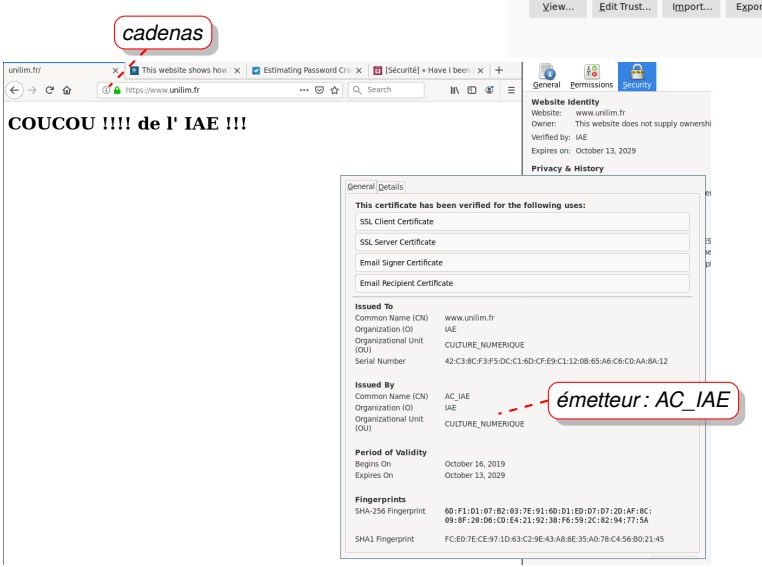
- ☐ Le certificat **authentifie** le site «*www.unilim.fr*» ;
- ☐ Le certificat est émis par l'organisation «*Let's Encrypt*» ;

⟹ Let's Encrypt est un **tiers de confiance** entre Alice et «*www.unilim.fr*» !

Alice s'est absentée et a laissé son poste de travail sans protection.

À son insu, j'installe un nouveau **tiers de confiance**, «*AC_IAE*» dans son navigateur Firefox :





Lorsqu'elle revient et consulte le site «*www.unilim.fr*», elle trouve un message «*Coucou !!!*» à la place de la page de l'Université.

⟹J'ai ajouté un «*Tiers de confiance*» qui donne de la **confiance** à mon mon **faux site** «*www.unilim.fr*» à l'aide d'un **faux certificat** !

⟹**La confiance d'Alice a été trahie !**

*Que faire pour ne pas **dépendre***

d'un tiers de confiance ?

$\Longrightarrow$ Répartir la confiance

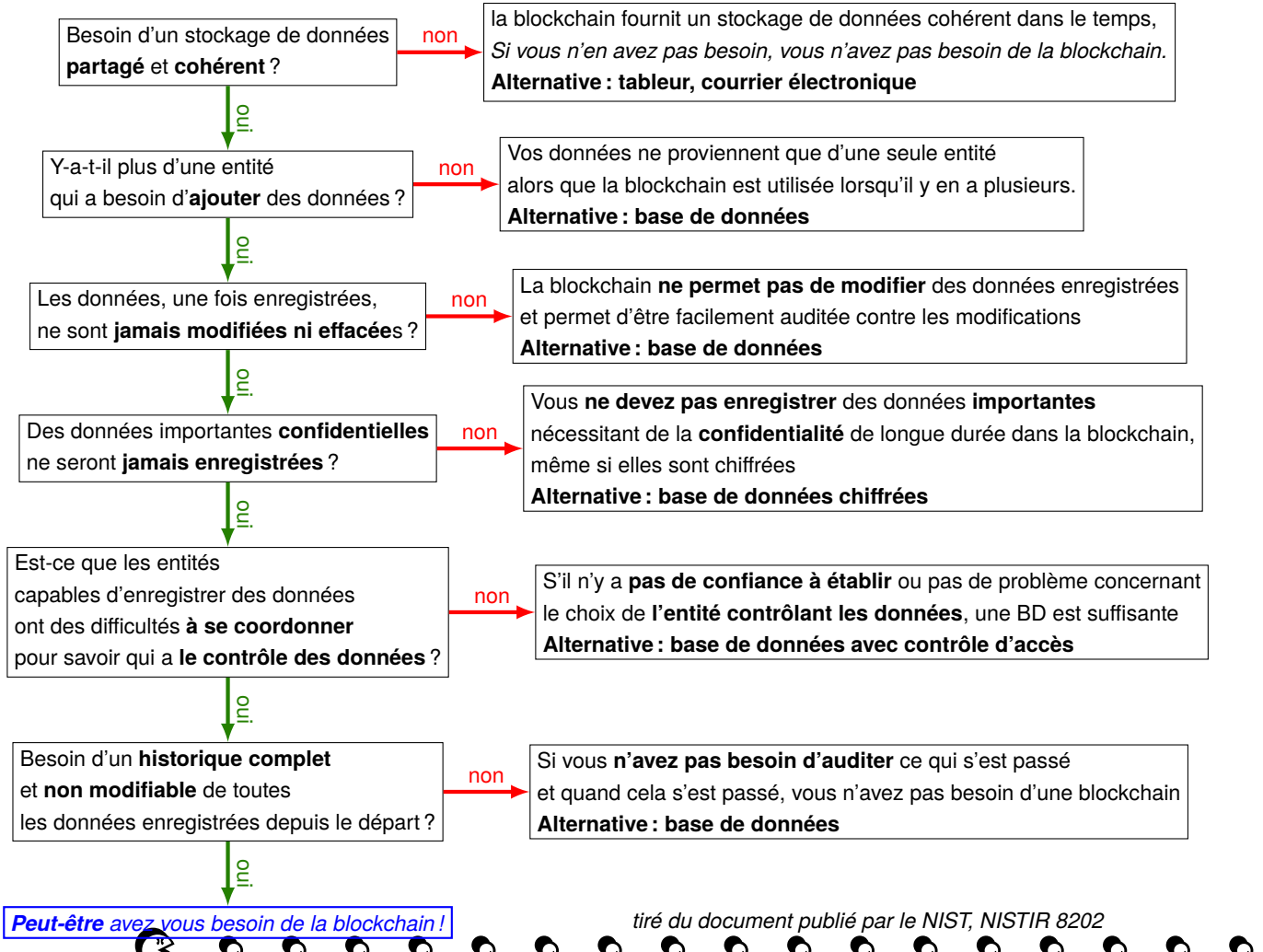
$\Longrightarrow$ Transparence

Définition

Une «*blockchain*» est une **liste chaînée** où chaque élément de la liste contient des **données** et un **hash** de l'élément précédent dans la liste.

L'ajout d'un élément dans la liste requiert **l'autorisation de la majorité** des utilisateurs.

- une **liste chaînée** ? une liste d'éléments avec un **lien** entre chacun des éléments de cette liste ;
- un **hash** ? une valeur assurant **l'intégrité** des données sur laquelle est calculée ;
- **l'intégrité** ? Avoir **la preuve** que les données sur lesquelles est déterminée cette intégrité **n'ont pas été modifiées** ;
- comment calculer un **hash** ? Utiliser une **fonction de hachage cryptographique** prenant en entrée les données à traiter et donnant en sortie une valeur :
  - **identique** à celle calculée précédemment, si les valeurs **n'ont pas été modifiées** depuis le calcul précédent ;
  - **différente** de celle calculée précédemment, si les valeurs **ont été modifiées** ;
- comment **assurer l'intégrité** ?
  - **mémoriser le hash** calculé sur les données initiales ;
  - permettre à l'utilisateur de **recalculer un hash** sur les données courantes ;
  - c**omparer les deux hashs** :
    - s'ils sont **différents** $\Longrightarrow$ données **modifiées**
    - s'ils sont **identiques** $\Longrightarrow$ données **non modifiées**
- comment **mémoriser le hash** d'un élément de la liste et **empêcher sa modification** ? en l'intégrant dans l'élément suivant de la liste chaînée ;
- qu'est-ce que le **lien** dans la liste chaînée ? le **hash** contenu dans l'élément **courant** de la chaîne est **calculé** sur l'élément **précédent**.



tiré du document publié par le NIST, NISTIR 8202

Alors, pour faire fonctionner la blockchain,
il faut de la «*Cryptographie*» ?

3. Les bases de la cryptographie

a. Vocabulaire

La cryptographie est une discipline consistant à manipuler des données de telle façon que les services suivants puissent être fournis :

Intégrité

Objectif : s'assurer que les données n'ont pas été modifiées sans autorisation.
Remarque : dans les faits, la cryptographie ne s'attache pas vraiment à empêcher une modification de données, mais plutôt à fournir un moyen sûr de détecter une modification malveillante.

Confidentialité

Objectif : ne permettre l'accès aux données qu'aux seules personnes autorisées.

Preuve (authentification et non-répudiation)

Objectif : fournir un moyen de preuve garantissant la véritable identité des entités ainsi que l'imputation de leurs actions.

Introduction

Depuis l'Egypte ancienne, l'homme a voulu pouvoir échanger des informations de facon **confidentielle**.
En grec :
Cryptographie : (κρυπτο – γραφην) écriture cachée / brouillée.

Il existe de nombreux domaines où ce besoin est vital :

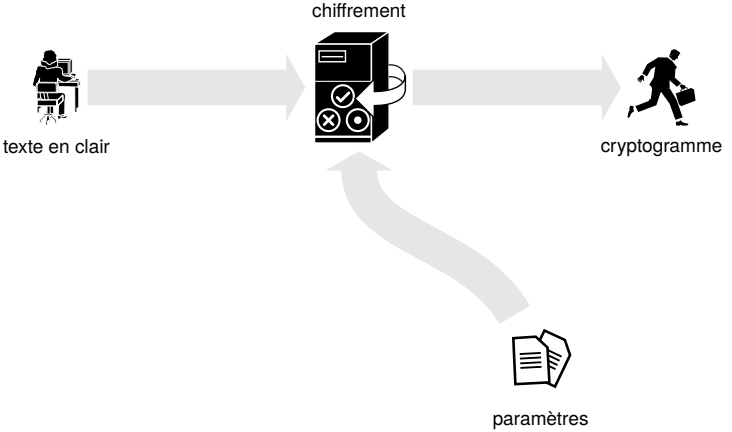
- ▷ **militaire** (sur un champ de bataille ou bien pour protéger l'accès à l'arme atomique) ;
- ▷ **commercial** (protection de secrets industriels) ;
- ▷ **bancaire** (protection des informations liées à une transaction financière) ;
- ▷ **vie privée** (protection des relations entre les personnes) ;
- ▷ **diplomatique** (le fameux «*téléphone rouge*» entre États-Unis et Union soviétique) ;
- ▷ ...

Définitions

Pour assurer la protection des accès à une information, on utilise des techniques de **chiffrement**.
Ces techniques s'appliquent à des **messages** lisibles appelés également «texte en clair».

Le fait de «*cacher*» un message de telle facon à le rendre secret s'appelle **chiffrement**.
La méthode inverse, consistant à retrouver le message original, est appelé **déchiffrement**.

Les messages à chiffrer, appelés «*texte en clair*», sont transformés grâce à une **méthode de chiffrement paramétrable**.
Si la méthode est connue de tous, ce sont les **paramètres** qui constituent la protection : ils servent à chiffrer/déchiffrer le message.



Ce **cryptogramme** est ensuite envoyé à son destinataire.

On appelle **cryptanalyse** les techniques employées pour déchiffrer un cryptogramme, **sans connaître** la méthode et/ou ses paramètres.

Le chiffrement est aussi appelé **cryptographie**.

L'ensemble des techniques de cryptographie et de cryptanalyse est appelé **cryptologie**.

Deux types de chiffrement :

Symétrique

Asymétrique

Chiffrement à clé symétrique

Principe

La même clé doit être employée pour chiffrer ou déchiffrer le message : on parle de clé symétrique ou secrète.

Le chiffrement consiste alors à appliquer un algorithme avec la clé secrète sur les données à chiffrer. Le déchiffrement se fait à l'aide de cette **même clé secrète**.

Remarques

La qualité d'un crypto système symétrique se mesure par rapport :

- à des propriétés statistiques des textes chiffrés ;
- à la résistance aux classes d'attaques connues.

En pratique

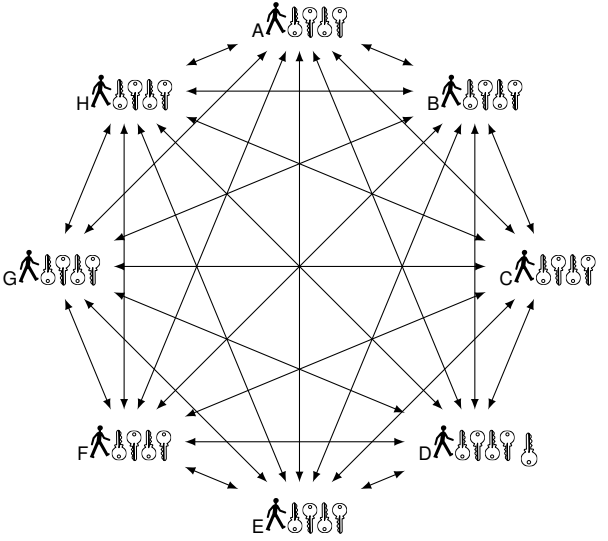
Tant qu'un crypto système symétrique n'a pas été cassé, il est bon, après il est mauvais !

Les limites du chiffrement symétrique

La multiplication des clés

Pour établir un canal de communication entre deux individus :

- Il faut qu'il soit chiffré avec une clé partagée entre les deux individus ;
- Il est ainsi confidentiel pour ceux qui ne possède pas la clé de chiffrement.



Pour que deux canaux de communications soient indépendants l'un de l'autre, c-à-d qu'une personne accède à l'un mais pas à l'autre, il faut que ces deux canaux utilisent des **clés différentes**.

Il est possible qu'un des interlocuteurs connaissent plusieurs clés utilisées dans différents canaux le reliant à des utilisateurs différents.

Exemple l'utilisateur D possède une clé pour chaque lien (avec H, G, F, E et C).

Problème comment échanger toutes ces clés ?

Pas d'intégrité et d'identification de l'auteur

Si Alice, Bob et Cédric partage le même lien de communication alors ils partagent la même clé de chiffrement symétrique.

Bob

clé secrète

clé secrète

Cédric

Alice

clé secrète

Message

message chiffré par Bob

message chiffré par Cédric

Message
(modifié par Cédric)

❶ Bob chiffre le message à destination d'Alice

❷ Cédric intercepte le message, le modifie, et le chiffre à nouveau avec la clé secrète

Problème

Chacun peut intercepter et modifier les messages qui s'échangent.

Et l’asymétrique alors ?

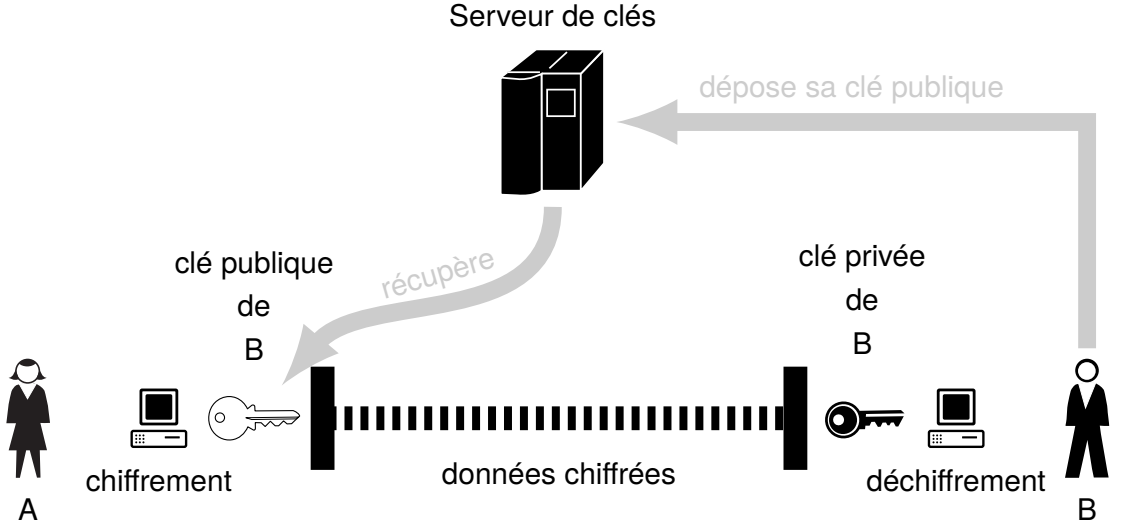
Chiffrement à clé asymétrique

Principe

Il utilise :

- une **clé publique** connue de tous ;
- une **clé privée** connue seulement du destinataire du cryptogramme.

*Ces chiffrements à «*clé publique*» ont été découverts par James Ellis (Angleterre) en 1969 et par Whitfield Diffie (Etats unis) en 1975.*



L'idée de la conception de tels algorithmes revient à Diffie et Hellman en 1976.

Des clé et des cadenas

Alice :

▷ crée une **clé aléatoire** (la clé privée) ;

▷ puis fabrique un **grand nombre de cadenas** (clé publique) qu'elle met à disposition dans un casier accessible par tous (le casier joue le rôle de canal de communication non sécurisé).

Bob :

▷ prend un cadenas (ouvert) ;

▷ ferme une valise contenant le document qu'il souhaite envoyer à Alice ;

▷ envoi la valise à Alice, propriétaire de la clé publique (le cadenas).

Cette dernière pourra ouvrir le cadenas et la valise avec sa clé privée.

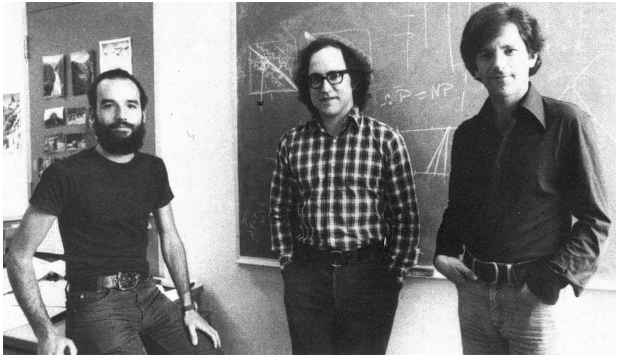
Les contraintes pour un tel algorithme

Il faut trouver un couple de fonctions f (fonction unidirectionnelle) et g (fonction de «*backdoor*») :
C'est un problème mathématique difficile !
Au départ, le système à clé publique n'a d'abord été qu'une idée dont la faisabilité restait à démontrer.

Des algorithmes ont été proposés par des mathématiciens

Un des premiers algorithmes proposé repose sur la **factorisation du produit de deux grands nombres entiers**.
Suivant la taille des nombres, cette factorisation peut demander un temps de calcul de plusieurs années : le problème est résolu !

Cet algorithme a été proposé par Rivest, Shamir et Adleman en 1977, ce qui a donné naissance à RSA.



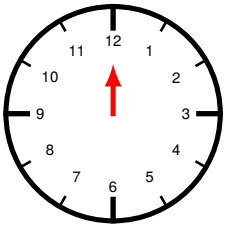
L'idée générale est la suivante :

- la fonction f est l'**exponentiation modulaire** ;
- la **clé publique** est la valeur c utilisée en combinaison avec le produit n de deux grands nombres entiers ;
- la **clé privée** est la valeur z ;
- g consiste en la **factorisation** de n.

Seul Bob, qui connaît z et g peut déchiffrer le message chiffré.

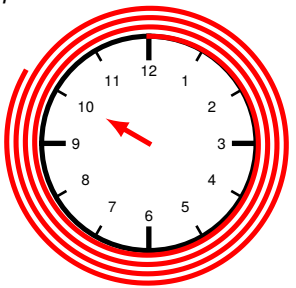
Les nombres modulaires : $x \bmod p$

Il est 0h ou 12h :



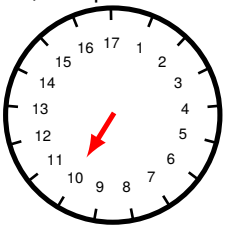
Et si on avance de 46h ?

on tourne de 46h autour du cadran...
On fait 3 tours : $3 * 12 = 36$
plus $46 - 36 = 10$ *h*



Ce qui fait $46 \bmod 12$? *10h !*

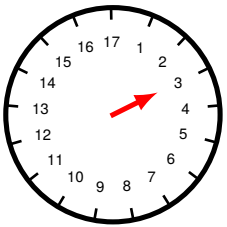
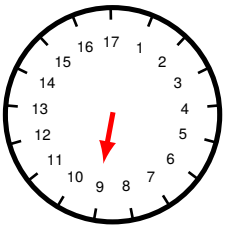
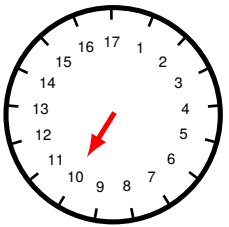
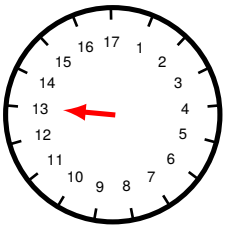
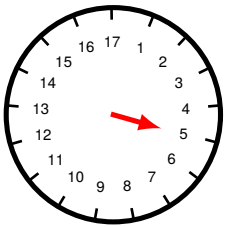
Et si on utilise un **nombre premier**, comme 17, à la place de 12 ?



et 3, la «*racine primitive*» modulo 17, c-à-d n'ayant pas de facteur en commun...

Les résultats de l'exponentiation modulaire :

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...
$3^n \bmod 17$	3	9	10	13	5	15	11	16	14	8	7	4	12	2	6	1	...

...

Les valeurs sont **également distribuées** autour du cadran...ce qui donne l'impression qu'elles sont **aléatoires**.
La **procédure inverse** qui permet de passer de 12, par exemple, à la valeur de x telle que : $3^x \bmod 17 = 12$ est **dure** !

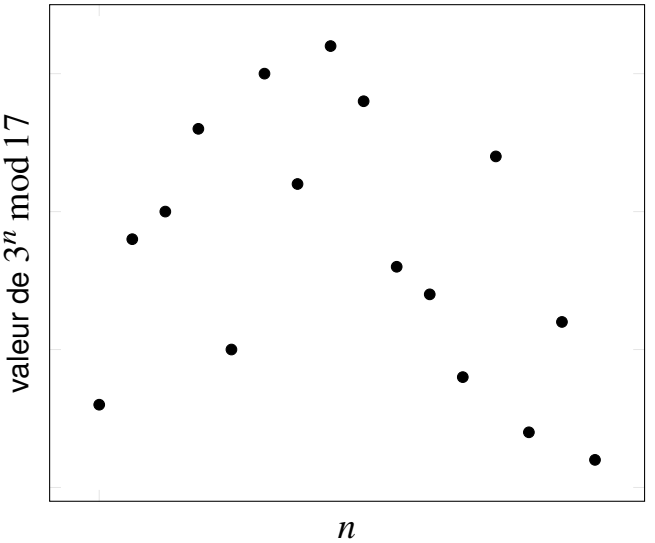
$\Longrightarrow$ ***pourquoi ne pas s'en inspirer pour définir un crypto-système ?***

Chiffrement : trouver une opération facile à réaliser mais dure à inverser

On vérifie le «*caractère aléatoire*» des résultats de l'exponentiation modulaire :

On recommence

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...
$3^n \bmod 17$	3	9	10	13	5	15	11	16	14	8	7	4	12	2	6	1	...



▷ Pourquoi la **procédure inverse** qui permet de passer de 12, par exemple, à la valeur de x telle que : $3^x \bmod 17 = 12$ est-elle **dure** ?

⟹ parce-qu'il faut **essayer toutes les valeurs possibles** avant de trouver la bonne, ce qui peut prendre beaucoup de temps !

Crypto-système : recherche d'un problème difficile à résoudre

$3^{29} \bmod 17$ --- facile ---> 12

$3^{?} \bmod 17$ <--- difficile --- 12

Problème du «*logarithme discret*» :
$\Longrightarrow$ pour trouver l'exposant, il faut **essayer les différentes valeurs possibles** !

Est-ce facile ?

Pour des valeurs petites, comme 17 oui... mais si on utlise des nombre plus grands comme :

```
41 699 392 957 415 060 122 123 251 743 926 118 047 280 755 942 727 859 562 221 144 422 770 879 634
528 234 687 327 545 978 537 253 643 190 435 384 223 451 790 600 743 186 710 706 948 084 596 275 495
353 648 241 532 947 688 879 507 515 517 193 627 711 579 879 475 350 089 816 821 087 217 733 022 854
019 999 144 696 637 566 134 923 661 835 848 181 145 153 671 156 026 923 341 312 533 527 164 598 580
084 075 991
```

ce nombre premier a été choisi sur 1024bits.

Il faut **beaucoup, beaucoup de temps** pour y arriver ! *Suivant la taille, plusieurs années avec des ordinateurs puissants !*

Et si on essaye d'aller plus loin ?

La fonction $\Phi(n)$ calcule le nombre d'entiers inférieur à n qui ne partagent pas de diviseur supérieur à 1 avec n.
Exemple : $\Phi(8) = 4$, *car* $\boxed{1}, 2, \boxed{3}, 4, \boxed{5}, 6, \boxed{7}$, *il y en a 4 ayant cette propriété.*

On peut arriver à :

- $\Phi(n) = n - 1$ si n est premier, sinon Φ(n) est **long à calculer** (il faut **énumérer** les différents entiers) ;
- $\Phi(a * b) = \Phi(a) * \Phi(b)$;
- si on choisit **deux nombres premiers** p_1 et p_2 et on calcule $n = p_1 * p_2$,
  on a $\Phi(n) = \Phi(p_1) * \Phi(p_2) = (p_1 - 1) * (p_2 - 1)$
- Théorème d'Euler : $m^{k*\Phi(n)+1} = m \bmod n$
  On essaye de trouver $e * d = k * \Phi(n) + 1$, soit $d = \frac{k*\Phi(n)+1}{e}$,
  $\Longrightarrow$ d'où notre **cryptosystème** : $\boxed{\text{message}^e \bmod n = \text{chiffré}}$ et $\boxed{\text{chiffré}^d \bmod n = \text{message}}$

Chiffrement asymétrique : clé publique et clé privée

Chiffrement

Alice prépare ses valeurs :
$p_1 = 53$
$p_2 = 59$
$n = 53 * 59 = 3127$
$\Phi(n) = 52 * 58 = 3016$
$e = 3$
$d = \frac{2*(3016)+1}{3} = 2011$

Alice utilise avec Bob :
$n = 3127$
$e = 3$ *clé publique*

et conserve **secrétement** :
$n = 3127$
$d = 2011$ *clé privée*

Bob **veut envoyer un message** à Alice :
$m = 89$ où 89 correspond à une lettre par exemple ;

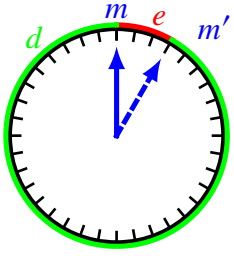
Il calcule :
$m^e \bmod n \Longrightarrow 89^3 \bmod 3127 = 1394$
et transmets à Alice la valeur 1394

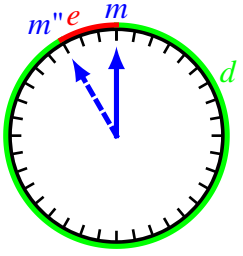
Déchiffrement

Alice reçoit 1394.
Elle calcule :
$1394^{2011} \bmod 3127 = 89$
et retrouve le message m de Bob !

Un attaquant obtenant les valeurs
$n = 3127$, 1394 *et* $e = 3$
doit calculer $\Phi(3127)$ *pour trouver* d
Ce qui lui prendrait trop de temps pour n *très grand !*

Chiffrement ? Déchiffrement ? Ce ne serait pas la même chose ?





On peut **permuter** l'usage de e et d...

$\text{message}^{e*d} \bmod n = \text{message}^{d*e} \bmod n = \text{message}$

C-à-d créer un message qui **peut être déchiffrer avec la clé publique** (e et n) mais qui ne peut être **créer qu'avec la clé privée** (d et n) !

Seule la personne **ayant la clé privée** peut chiffrer un message choisi **déchiffrable par la clé publique** associée...
$\Longrightarrow$On peut **authentifier** la personne ! C-à-d **garantir son identité**, car elle est la seule à posséder cette clé privée !

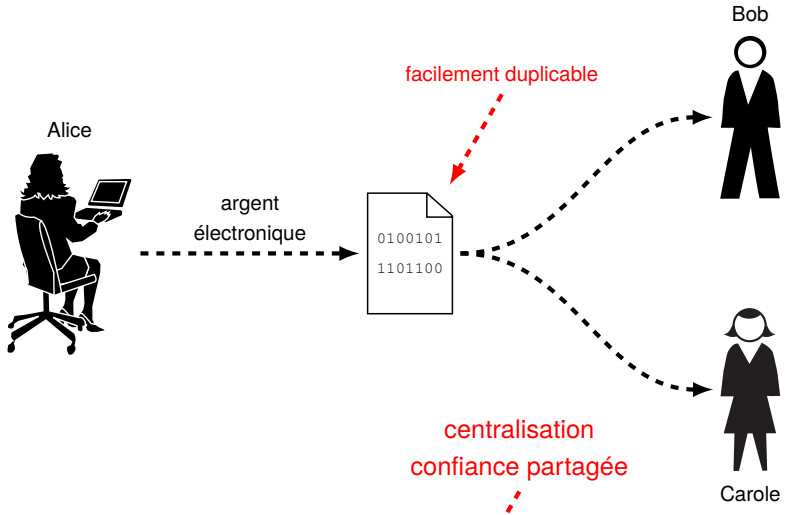
Et la Crypto-monnaie dans tout ça ?

Alice envoie de l'argent à Bob

Argent fiat

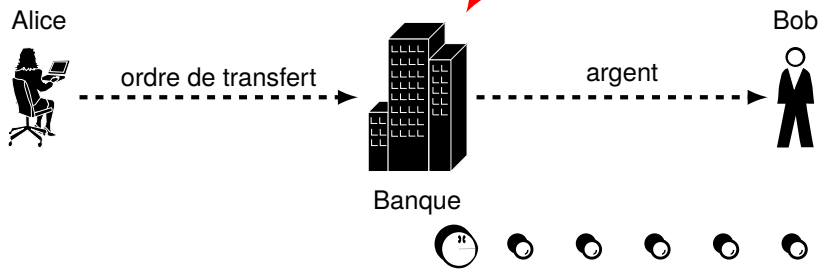
Alice donne de l'argent sous forme de cash à Bob $\Longrightarrow$ Alice ne possède plus l'argent et Bob l'obtient.

Problème de l'argent électronique ? Dépenser plusieurs fois la même somme !



- ☐ Alice envoie de l'argent sous forme de fichier électronique à Bob ;
- ☐ Bob n'a **pas de preuve** qu'Alice a effacé sa copie de ce fichier ;
- ☐ Alice peut **envoyer une copie du fichier** à Carole.

Solution ?

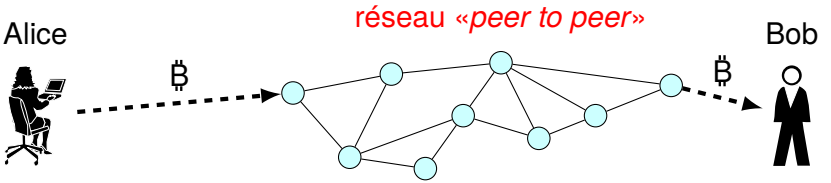


Utiliser un «*tiers de confiance*» qui :

- ▷ vérifie qu'Alice a bien la somme ;
- ▷ **valide** la transaction et **crédite** Bob ;
- ▷ **enregistre la transaction** : Alice et Bob ont échangé de l'argent.

Alice envoie de l'argent à Bob

Solution décentralisée ? Le «*BitCoin*» ₿



- **Distribué** : le livre de compte ou «*ledger*», la «*blockchain*», est dupliqué entre différents ordinateurs⟹**n'importe quel ordinateur** connecté à Internet peut le télécharger ;

- **Cryptographique** :
  - **s'assure** que l'expéditeur, Alice, possède bien les bitcoins qu'elle veut envoyer ;
  - **décide** comment la transaction va être **ajoutée** au blockchain ;

- **Immuable** : la blockchain ne peut qu'être **étendue**⟹ toute transaction enregistrée ne peut être modifiée ou effacée ;

- **Preuve de travail** : certains nœuds du réseau «*peer to peer*» sont appelés «*mineurs*» :
  - ils cherchent la solution d'un **puzzle cryptographique** ;
  - s'ils la trouvent : il peuvent **ajouter la transaction** au blockchain ;
  - ils sont **rémunérés** ;
  - ils **créent** de la monnaie.

Allez ! Encore un peu de Cryptographie...

Qu'est-ce qu'une fonction de hachage ?

Une **fonction de hachage** est une fonction qui fait correspondre **à toute information** une valeur appelée «***hash***».

Exemple : pour accéder rapidement à un livre dans une base de données, on utilise une fonction de hachage pour accéder directement au livre recherché :

f : {*ensemble des titres de livres*} $\mapsto$ {*ensemble des hashs*}

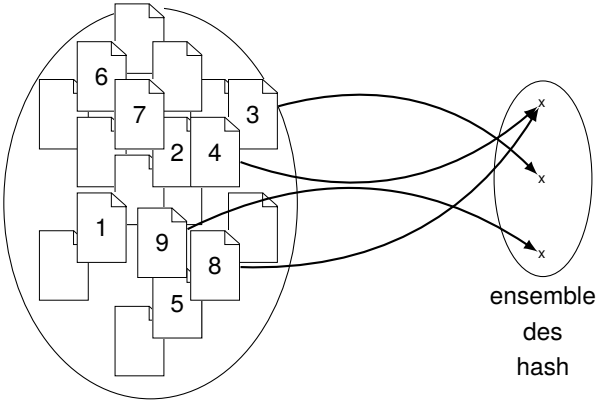
f(*20 000 Lieues sous les mers*) $\rightarrow 598$

f(*Les androïdes rêvent-ils de moutons électriques ?*) $\rightarrow 698593$

*Ainsi, on ne parcours pas tous les titres, mais on va **directement** à la valeur indiquée par la fonction de hachage.*

*Si deux livres possèdent le même hash, on parle de «**collision**». Une bonne fonction de hachage limite les collisions.*

Fonction de hachage cryptographique

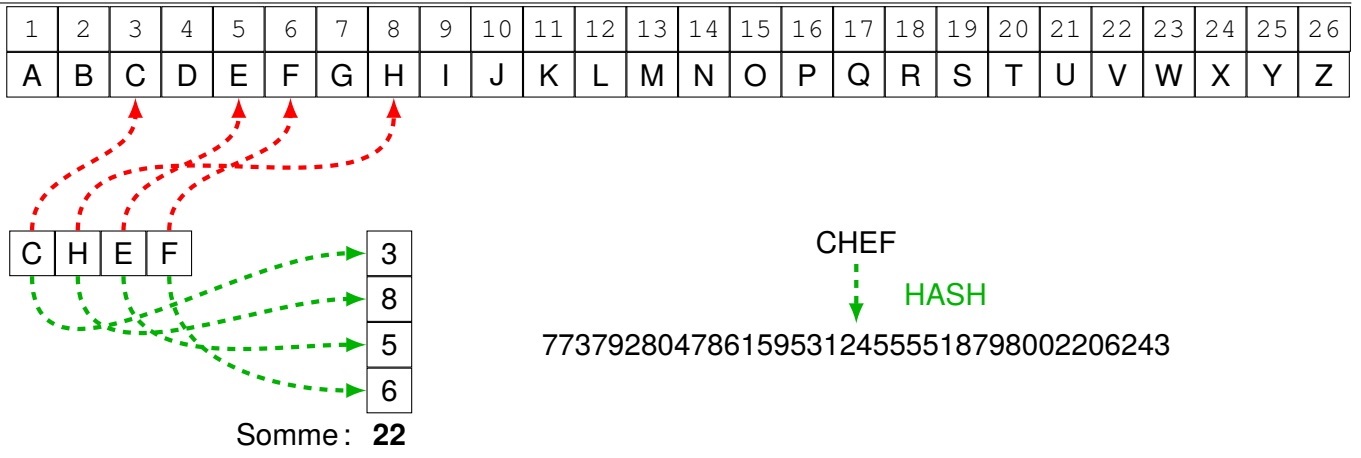


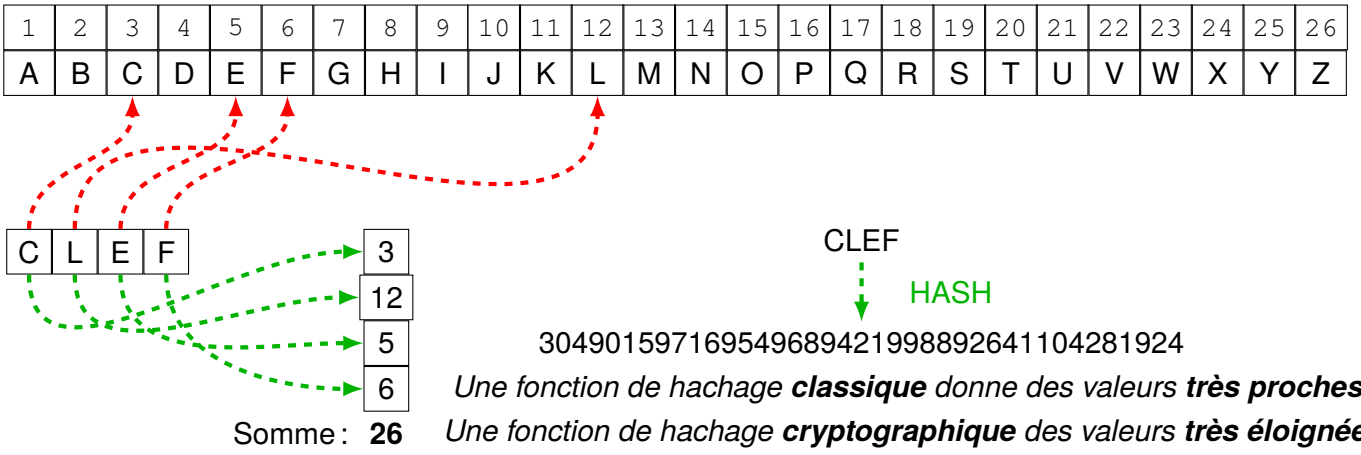
Une **fonction de hachage cryptographique** possède les propriétés suivantes :

- **pour tout document électronique** elle calcule un hash ;
- depuis le hash il est **impossible** de retrouver le document ;
- deux documents **proches** possédent des hashs **très différents** ;
- pour un **hash connu**, il est difficile, voire **impossible** de trouver le document permettant de l'obtenir ;
- il est très difficile, voire **impossible**, de trouver deux documents différents ayant le **même hash** (collision).
- le hash est de taille **fixe** et **limitée** (le document peut être de taille quelconque).

*On parle aussi **d'empreinte** de documents.*

Fonction de hachage cryptographique



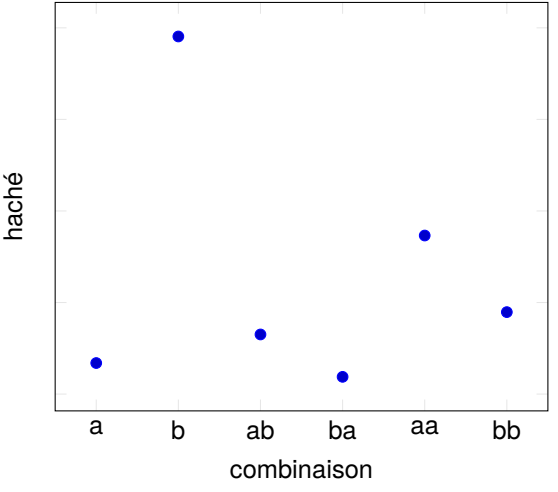


*Une fonction de hachage **classique** donne des valeurs **très proches**...*

*Une fonction de hachage **cryptographique** des valeurs **très éloignées** !*

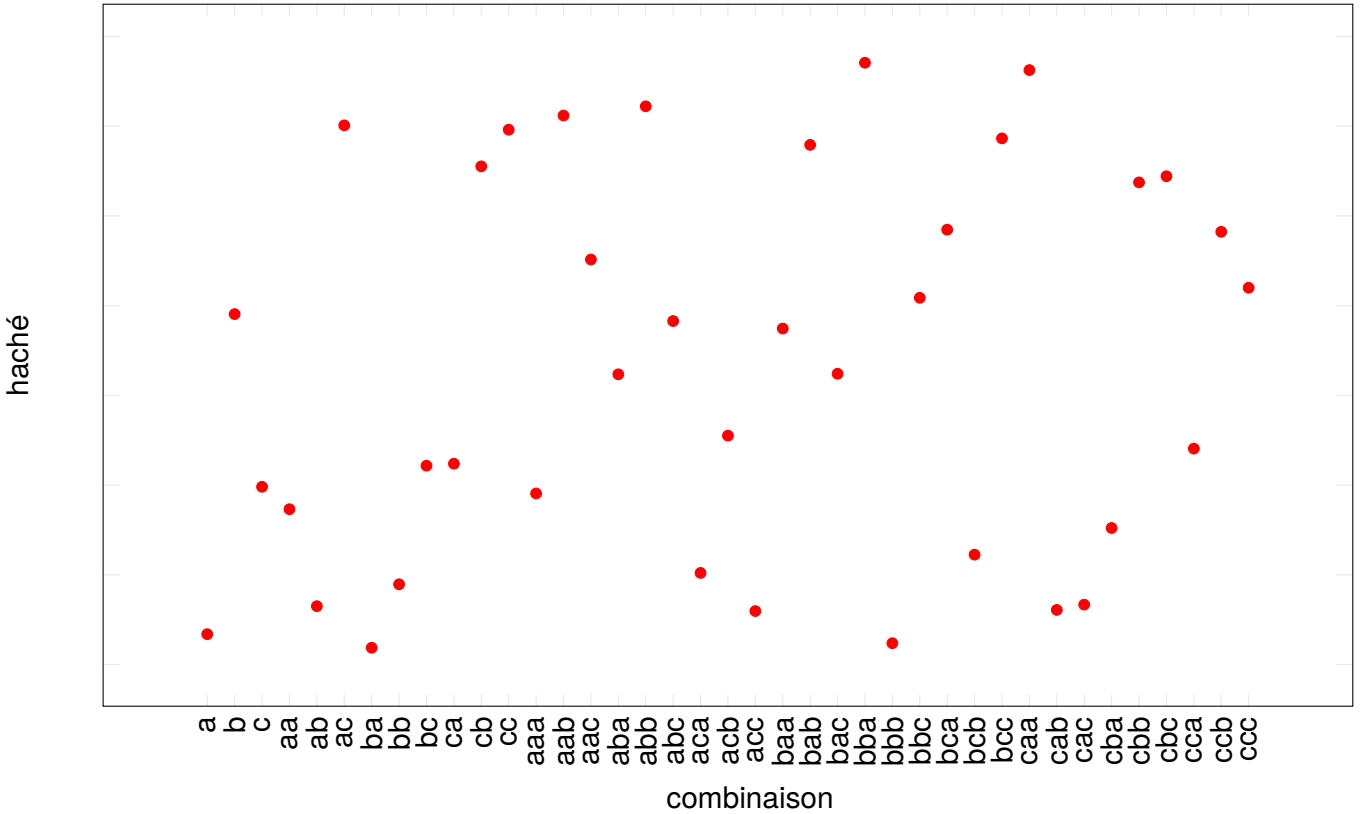
Imaginons que l'on veuille voir quelles sont les valeurs associées par la **fonction de hachage** aux différentes combinaisons possibles obtenues à partir des lettres«*a*» et «*b*», d'au plus 2 lettres :

combinaison	**haché**
a	169552370019632401730582715598587264 97
b	195289424170611159128911017612795795343
ab	325606555493056888658533171298094888 00
ba	941680395931154527312999502951431136 4
aa	865905563437731851846768541823880583 86
bb	447630314541533955979929907481056088 28

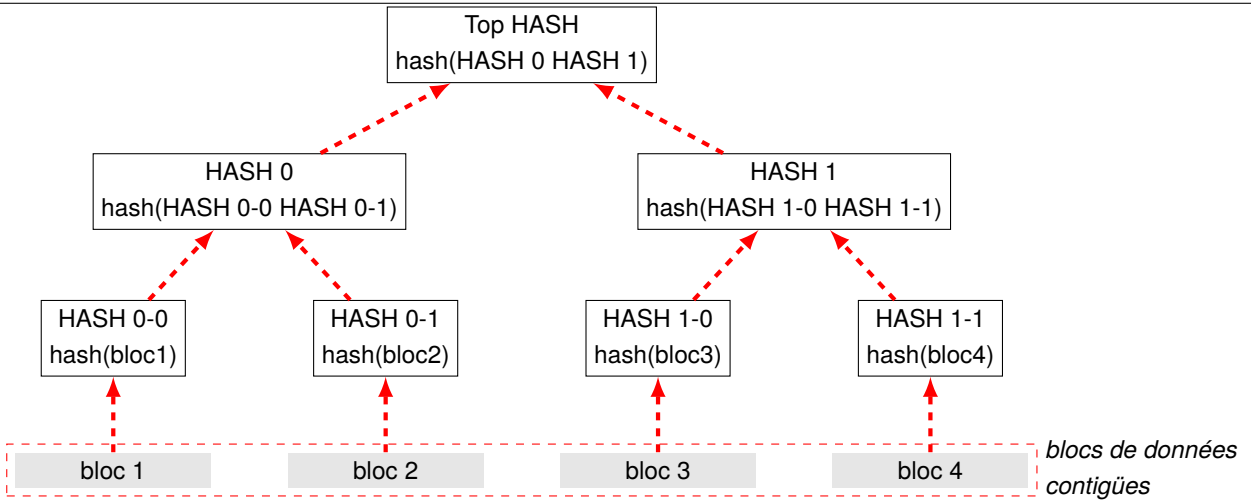


Fonction de hachage cryptographique

Tous les mots d'au plus 3 lettres à partir de l'alphabet { a,b,c }.



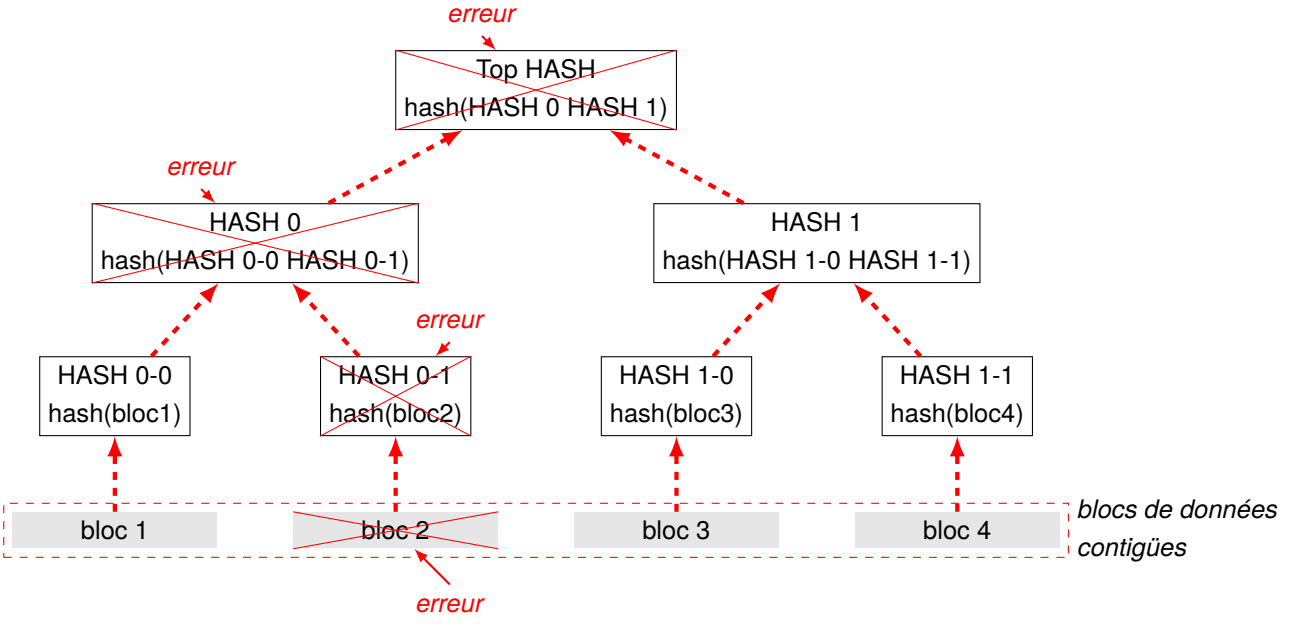
$\Longrightarrow$ ressemble à une **distribution aléatoire** !



L'utilisation de cette arbre permet **d'isoler une erreur** dans un fichier transféré d'Alice vers Bob :

1. Bob veut vérifier que le fichier reçu depuis Alice est sans erreur : il calcule l'abre de merkle sur sa copie de fichier ;
2. Alice transmets le haché Top HASH à Bob, c-à-d celui de la racine de l'arbre ;
3. Bob compare le haché qu'il a calculé avec celui qu'il a reçu d'Alice :
   - ◇ identique ? Le fichier a été transferré **sans erreur**.
   - ◇ différent ? Bob demande à Alice les hachés de la racine de chaque sous-arbre : HASH 0 et HASH 1 ;
4. pour chacun de ses hachés, HASH 0 et HASH 1, Bob peut vérifier par rapport à la valeur qu'il a lui-même calculé dans son abre de Merkle :
   - ◇ dès qu'il trouve une différence de haché, il peut identifier le sous-arbre concerné et descendre jusqu'à la vérification d'une **feuille** de l'arbre, c-à-d le haché de chaque bloc de données en comparant avec la version d'Alice ;
   - ◇ dans le cas où le haché d'un bloc de données n'est pas bon, il peut demander à Alice de lui retransmettre les données de ce bloc uniquement.

Détection d'une erreur et identification du ou des blocs concernés



Si une erreur se produit sur le bloc 2 :

- ▷ Top HASH différent de celui d'Alice ⟹ Bob demande le haché de **chaque sous-arbre** ;
- ▷ HASH 0-1 différent ⟹ Bob demande le haché de **chaque sous-arbre** ;
- ▷ HASH 0 différent ⟹ la feuille correspondant au haché du bloc bloc 2 ⟹ le bloc 2 doit être **retransmis** par Alice ou par **quiconque en possédant une copie** ;

⟹ **application au réseau «*peer-to-peer*».**

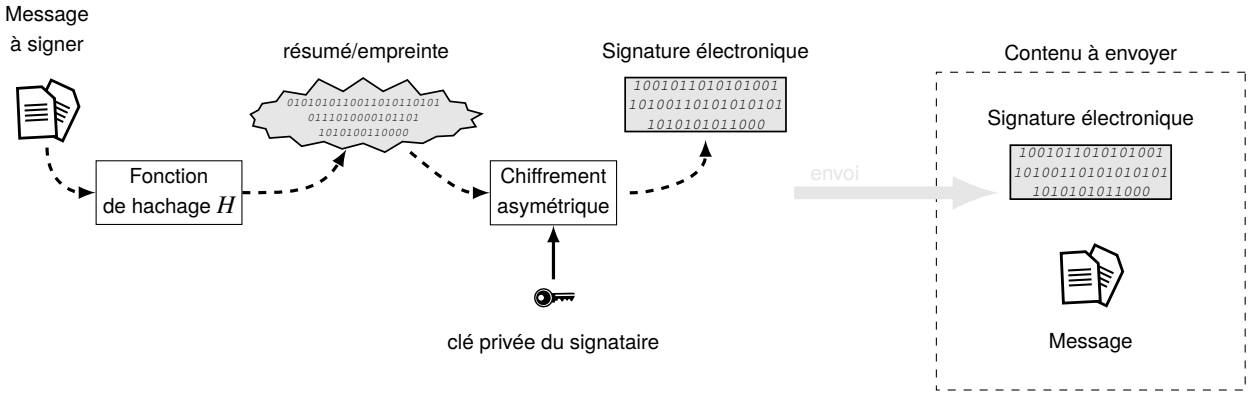
Et si on mélange chiffrement asymétrique

et

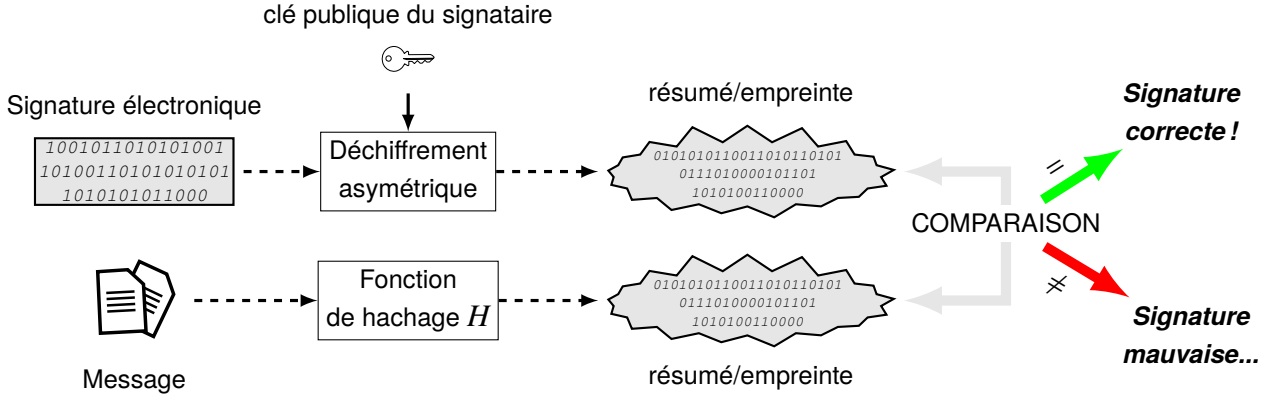
fonction de hachage ?

$\Longrightarrow$ *La signature électronique !*

Créer une signature électronique



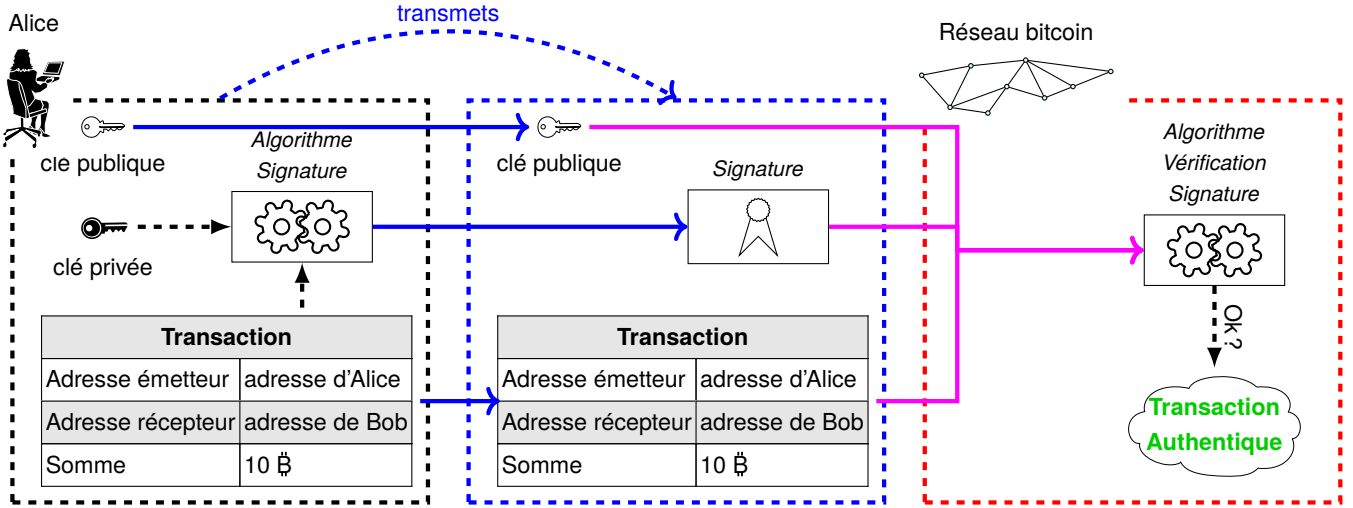
Vérifier une signature électronique



Alors ? Notre crypto-monnaie ?

Alice envoie de l'~~argent~~ *des bitcoins* à Bob

Vérification de l'authenticité d'une transaction



Alice :

- ☐ crée une **transaction** ;
- ☐ **signe** cette transaction avec sa **clé privée**.

Alice envoie au réseau :

- ☐ la **transaction** ;
- ☐ une copie de sa **clé publique** ;
- ☐ la **signature** de la transaction.

N'importe quel nœud du réseau peut **vérifier** la transaction en utilisant :

- ☐ la **clé publique** d'Alice ;
- ☐ la **transaction** ;
- ☐ la **signature**.

MAIS...

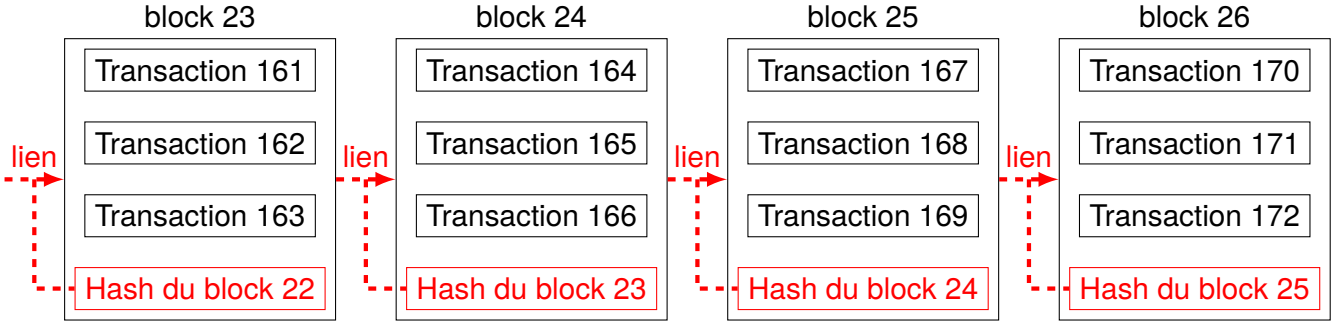
- ▷ *Qui donne 10₿ à Alice ?*
- ▷ *Comment interdire à Alice de créer autant de transactions qu'elle veut ?*
- ▷ *Qu'est-ce que Bob obtient au final ?*

⟹**Il faut mémoriser et lier toutes les transactions du système !** ⟹*Il faut une* «blockchain» *!*

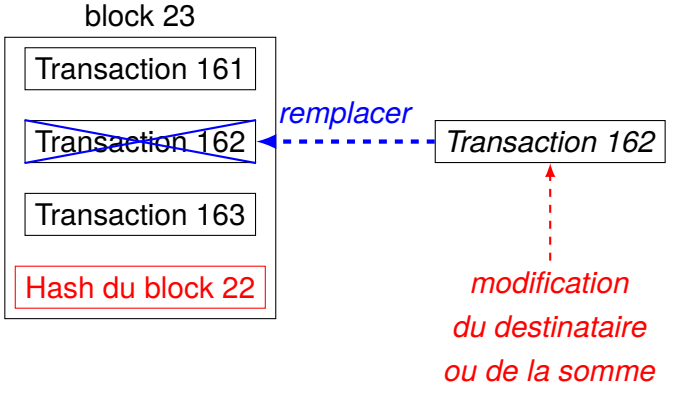
La Blockchain !

Composition de la blockchain

La blockchain est constituée de bloc liés entre eux par une fonction de hachage



Peut-on modifier un bloc ?



Si on modifie la transaction 162,
alors le Hash du block 23 va être **changé**.

⟹le bloc 24 contient une valeur de Hash du block 23 **différente** !

⟹il faut modifier le contenu de **tous les blocs** à partir du bloc modifié !

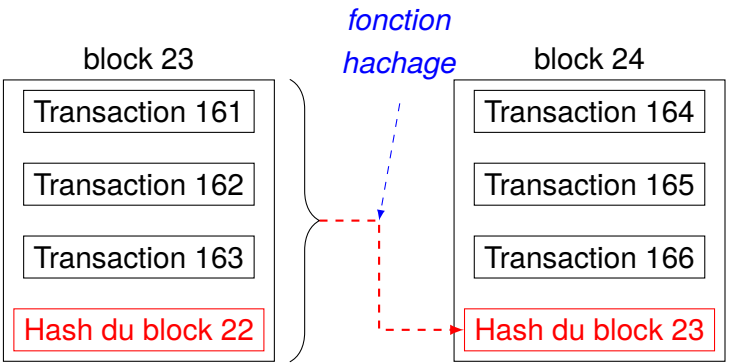
⟹on crée une **version différente** de la blockchain !

Est-ce possible d'empêcher la modification de toute la blockchain ? distribution et preuve de travail !

Preuve de travail : résolution d'un puzzle cryptographique

Utilisation d'une fonction de hachage cryptographique

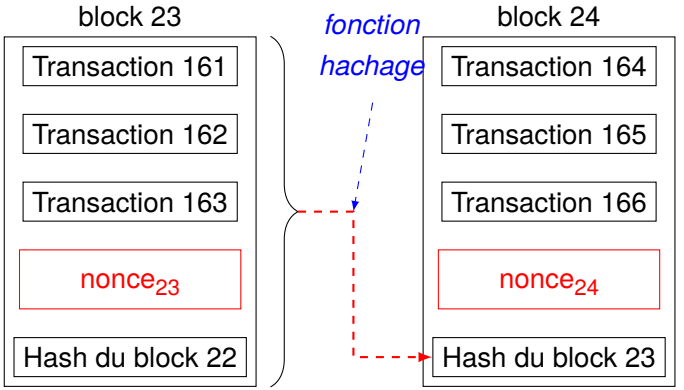
- un **hash** est une **valeur numérique de taille fixe** retournée par une fonction de hachage ;
  Dans BitCoin, on utilise la fonction SHA-256, qui donne des valeurs de hachage, ou hash, sur 32 octets.
- d'après les **propriétés** des fonctions de hachage cryptographique, il est **impossible** de trouver un document qui donnerait **hash connu** ;
- dans la blockchain, le hash est **utilisé** pour :
  - vérifier qu'un bloc **n'a pas été modifié** ;
  - **lier** ce bloc au bloc suivant ;



- si on **modifie** le bloc, le hash est **différent**.
- si on **ajoute** des données au bloc, le hash est **différent** $\Longrightarrow$ idée de Puzzle !

Définition du puzzle : sa résolution est une preuve que l'on a travaillé pour le résoudre

- ▷ on ajoute un **nouveau champ** au bloc : le nonce, «*number used once*» :

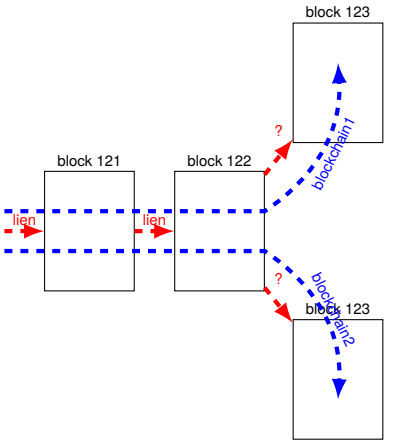


- ▷ si on en fait **varier** la valeur du nonce, le hash du bloc est **modifié** ;
- ▷ on définit le **Puzzle** :
  - ◇ on veut **trouver un hash** avec une certaine «*propriété*» : par exemple il doit commencer par un certain nombre de zéros;
  - ◇ comment faire pour obtenir cette propriété ? **en modifiant le nonce** ;
  - ◇ est-ce facile ? **Non !**
  - ◇ il faut essayer **toutes les valeurs de nonce** jusqu'à obtenir ce hash particulier !
- ▷ **Résoudre** le puzzle est **très dur** et prends beaucoup de travail, mais **vérifier** la solution est **instantanée**
  $\Longrightarrow$ *il suffit de calculer le hash du bloc et vérifier la propriété du hash obtenu !*
- ▷ C'est l'**opération de minage** : l'obtention du nonce qui donne la bonne valeur est la «***Preuve de travail***» !

Mais...

si on distribue la construction de bloc sur un réseau ?

$\Longrightarrow$ Risque de conflit !



- des **transactions** sont **transmises** dans le réseau ;
- ces transactions sont **reçues** par des mineurs différents ;
- **deux mineurs** travaillent simultanément sur le prochain bloc à ajouter ;

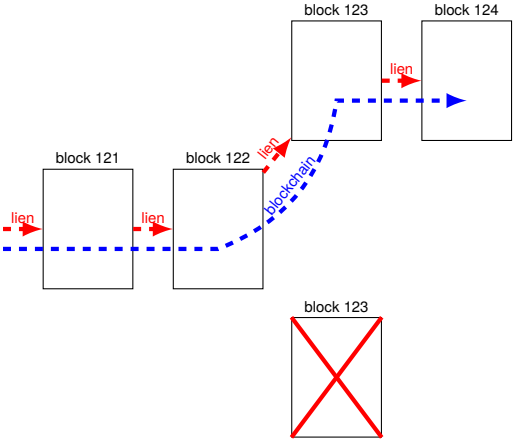
⟹ **deux versions** de la blockchain sont **créées** ;
Ici, les deux mineurs proposent chacun une version pour le `bloc 123` *et deux versions de blockchains, 1 et 2.*

⟹ situation de **conflit** !

- les deux blocs proposés peuvent ne pas contenir les **mêmes transactions**.

Résolution du conflit ? on attend le bloc suivant et c'est la blockchain la plus longue qui gagne :

- deux blockchains existent **simultanément** dans le réseau ;
- des mineurs vont travailler sur le «*blockchain 1*» et d'autres sur le «*blockchain 2*» ;
- le **premier mineur** qui construit le prochain bloc résouds le conflit :
  - si le **prochain bloc** est construit sur la «*blockchain 1*» alors c'est la «*blockchain 1*» qui est choisie ;
  - le bloc **alternatif** est détruit, les transactions non présentes dans la blockchain «*officielle*» sont remises à disposition des mineurs pour être intégrées dans de futurs blocs ;
  - les **récompenses** des mineurs du blockchain détruit sont accordées aux mineurs de la**blockchain gagnante**.



Attaque par vitesse

- Un attaquant envoie la même somme successivement vers deux adresses différentes ;

$\Longrightarrow$ il est recommandé d'attendre au moins un bloc de confirmation avant d'accepter le paiement.

Attaque de Finney

- un attaquant *pré-mine* un bloc avec une transaction et dépense la même somme dans une seconde transaction avant de distribuer le bloc qu'il a pré-miné ;

$\Longrightarrow$ la seconde transaction ne sera pas validée !

$\Longrightarrow$ il est recommandé d'attendre au moins 6 blocs de confirmation avant d'accepter le paiement.

Attaque de la majorité ou attaque des «*51%*»

- l'attaquant possède au moins 51% de la puissance de traitement du réseau ;
- l'attaquant réalise une transaction qu'il diffuse dans l'intégralité du réseau ;
- il mine ensuite une chaine privée où il dépense une seconde fois la même somme que dans la transaction précédente ;

$\Longrightarrow$ comme l'attaquant possède la majorité de la puissance de calcul du réseau, il est garanti de posséder, à un certaint moment, une chaîne plus longue que celle du reste du réseau ;

$\Longrightarrow$ il est assuré que lorsqu'il va diffuser sa chaîne privée, elle remplacera celle(s) existante(s) et annulera la transaction originale ;

$\Longrightarrow$ comment se protéger ? disposer d'une puissance supérieure à celle du réseau est très peu vraisemblable !

Ok !

Et bitcoin alors ?

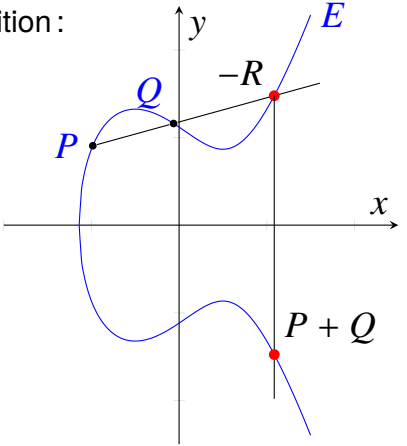
Le **bitcoin** est :

- **inventé** par un pseudonyme, Satoshi Nakamoto, en 2008 dans l'article : «*Bitcoin: A Peer-to-Peer Electronic Cash System*» ;
- **limité en quantité** : 21 millions de bitcoins au maximum ;
- **fractionné** :
  - en millibitcoin : 1 millième $1/1000$ ₿
  - en «*satoshi*» : 1 sur 100 million $1/100000000$ ₿
- **complétement décentralisé** : pas d'autorité centrale pour l'émission, la création de transaction et leur validation ;
- utilise un **système distribué** réalisant :
  - une «*élection globale*» à peu près toutes les 10 minutes ;
  - un «*consensus*» décentralisé sur l'état global des transactions ;
- interdit la **double dépense**, où l'on peut dépenser deux fois une même somme ;
- **transparent** : toutes les transactions peuvent être observées depuis l'origine du système ;
- **basé sur la confiance** :
  - toutes les transactions sont **liées entre elles** dans une «*blockchain*» accessible publiquement, dont la copie est encouragée entre tous les participants.
  - tous les mécanismes **d'identification** des acteurs, des **transactions** et des **preuves de propriété** reposent sur la **cryptographie** asymétrique ou dite «*à clé publique*» ;
  - les **évolutions de la blockchain** sont protégées contre les modifications par la cryptographie : **infalsifiable** et **immuable**.

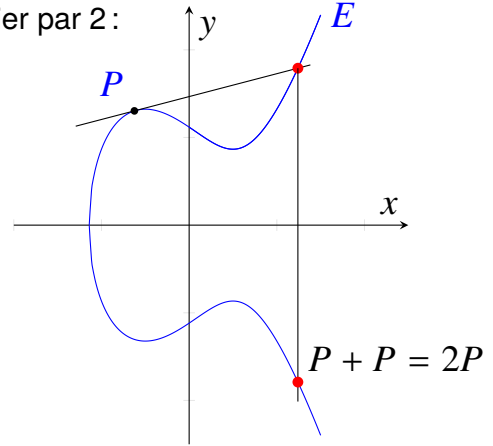
Encore de la Crypto !

Courbes elliptiques : $y^2 = x^3 + ax + b$

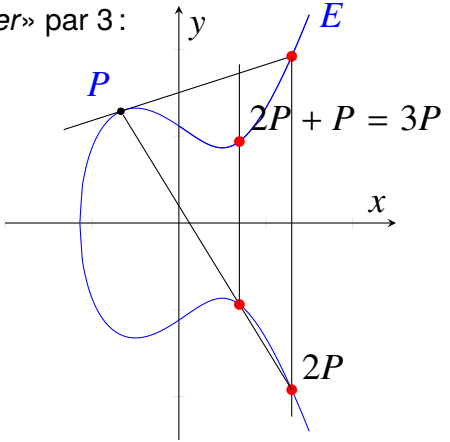
Addition :



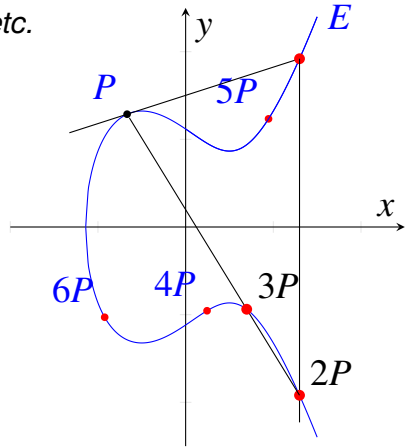
Multiplier par 2 :



«*Multiplier*» par 3 :

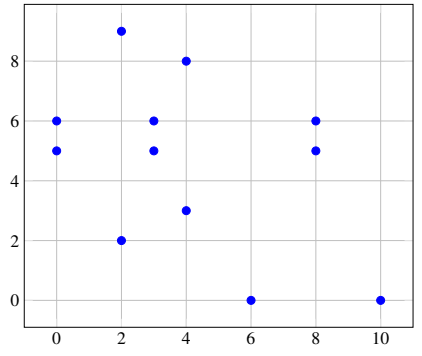


etc.

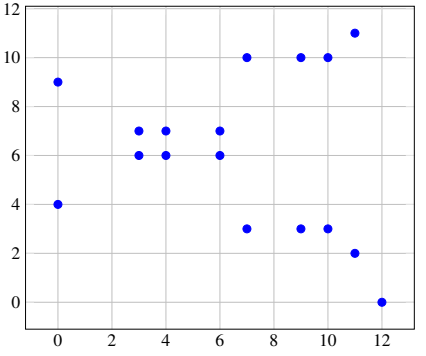


- ▷ on n'utilise que des **valeurs entières** positives pour les coordonnées des points sur la courbe ;
- ▷ on utilise le **modulo**, p, pour «*replier*» le domaine de ces points dans un **espace réduit**
  $\Longrightarrow$ on obtient un **ensemble fini** de points ;

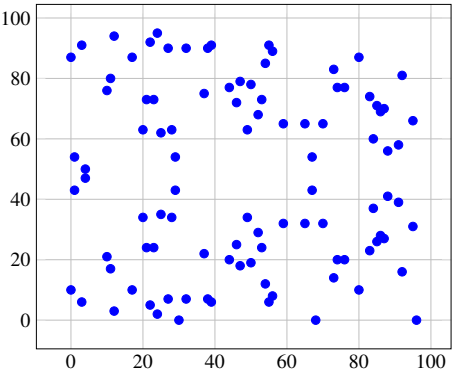
$y^2 \equiv x^3 + 2x + 3 \bmod 11$



$y^2 \equiv x^3 + 2x + 3 \bmod 13$



$y^2 \equiv x^3 + 2x + 3 \bmod 97$



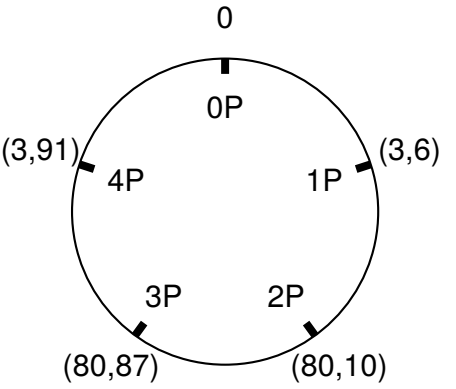
On observe que :

- ▷ plus la taille du module p est **grande**, plus le domaine $\mathbb{F}_p$ est **important** ;
- ▷ la distribution des points semble **aléatoire** et recouvre la surface de manière **régulière** ;
- ▷ Notion de **groupe** :
  - ◇ les points appartenant à la courbe elliptique définie sur $\mathbb{F}_p$ définissent un **groupe** :
    - ⋆ **l'addition** d'un point avec un autre et la **multiplication scalaire** donnent, chacune, un point **appartenant** à cet ensemble ;
  - ◇ le **nombre de points** appartenant à ce groupe définit **l'ordre** du groupe ;
    Exemple : pour $y^2 \equiv x^3 + 2x + 3 \bmod 11$ *on dénombre 12 points, et 17 points pour* $y^2 \equiv x^3 + 2x + 3 \bmod 13$.

- ▷ en utilisant un **nombre premier** comme module, on obtient des propriétés intéressantes :
  - ◇ pour un point P, on observe des **cycles** :

    Pour la courbe $y^2 \equiv x^3 + 2x + 3 \bmod 97$ définie sur le domaine fini $\mathbb{F}_p$ où $p = 97$:
    - ⋆ $0P = 0$
    - ⋆ $1P = (3, 6)$
    - ⋆ $2P = (80, 10)$
    - ⋆ $3P = (80, 87)$
    - ⋆ $4P = (3, 91)$
    - ⋆ $5P = 0$
    - ⋆ $6P = (3, 6)$

    ...



    On remarque que :
    - ⋆ on a définit un **sous groupe** de notre domaine $\mathbb{F}_p$: les autres points de la courbe elliptique n'apparaissent pas dans le cycle ;
    - ⋆ on observe que $5kP = 0$, $(5k+1)P = P$, $(5k+2)P = 2P$, ..., $(5k+4)P = 4P$ et $(5k+5)P = 0$

    $\Longrightarrow$ on déduit que $kP = (k \bmod 5)P$; P est appelé «*générateur*» du sous-groupe, 5 est **l'ordre** du sous-groupe.

$\Longrightarrow$ Et si on choisissait pour notre problème connaissant P et Q, peut on trouver k tel que $Q = kP$ en utilisant un sous-groupe comme vu plus haut ?

$\Longrightarrow$ on arrive au problème du «*logarithme discret*» car l'ensemble des points du sous-group est **fini**.

$\Longrightarrow$ dans RSA on utilisait l'**exponentiation modulaire**, en ECC on utilise la **multiplication scalaire** sur un domaine fini (sous-groupe cyclique) ;

$\Longrightarrow$ le **problème est plus dur** dans ECC que dans RSA $\Longrightarrow$ la **taille des clés peut être plus petite** pour une **difficulté similaire** !

Un ensemble d'opérations :

▷ on peut **additionner** des points ;

▷ on peut **multiplier** un point par un entier :
- ◇ Multiplier par 4 : $3P + P = 2P + 2P = 4P$
- ◇ Multiplier par 100 : $P \Longrightarrow 2P$, $2P \Longrightarrow 3P$, $3P \Longrightarrow 6P$, $6P \Longrightarrow 12P$, $12P \Longrightarrow 24P$, $24P \Longrightarrow 25P$, $25P \Longrightarrow 50P$ et $50P \Longrightarrow 100P$

▷ les points obtenus sont sur la courbe et apparaissent «*distribuer de manière aléatoire*» ;

▷ Si on donne $Q = nP$ alors il est très difficile de trouver $n \Longrightarrow$ il faut essayer les différentes possibilités : $2P$, $3P$, ...

$\Longrightarrow$ on a système qui est **rapide à calculer dans un sens** :
- ◇ connaissant n et P on cherche Q,
- ◇ **mais qui est difficile** à «*inverser*» : connaissant deux points, P et Q, trouver n tel que $Q = nP$ (problème du logarithme discret).

Avantages par rapport à RSA ?

▷ le problème posé par les courbes elliptiques est plus dur que celui posé par RSA :
$\Longrightarrow$ la taille des clés est **plus petite** pour assurer un niveau de **sécurité équivalent**.
$\Longrightarrow$ une clé de **246 bits ECC** est **équivalente** à une clé **3072 bits RSA** !

Cryptographie basée sur les courbes elliptiques

- la **clé privée** est une **valeur entière aléatoire** d, choisie entre 1 et n, où n est **l'ordre du groupe** ;
- la **clé publique** est le point $H = dG$ où G est le **générateur** définissant le groupe ;

Garantie :

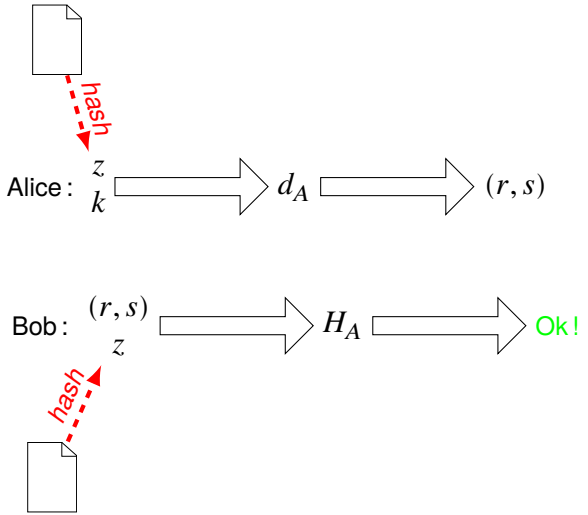
- Si l'on connait d et G, trouver H est facile ;
- Si l'on connait H et G, trouver la clé privée d est difficile $\Longrightarrow$ résoudre le *problème du logarithme discret*...

Signature ECDSA

- Alice veut **signer** un message avec sa clé privée d_A ;
- Bob veut **valider** la signature avec la clé publique d'Alice H_A ;

Fonctionnement :

- on calcule le **hash** du document à signer ;
- on tronque le **hash** de telle manière que le **nombre de bits** soient de même taille que n **l'ordre** du groupe et on appelle z ce nombre ;
- on prend une **valeur entière aléatoire** k entre 1 et $n-1$ où n est l'ordre du groupe ;
- on calcule $P = kG$ où G est le **générateur** du groupe ;
- on calcule $r = x_p \bmod n$ où x_p est la coordonnée suivant l'axe x, ou l'abscisse, de P ;
- *si* $r = 0$ *on choisit un autre* k *et on ré-essaye* ;
- on calcule $s = k^{-1}(z + r * d_A) \bmod n$ où d_A est la clé privée d'Alice et k^{-1} est l'inverse de k pour la multiplication modulo n ;
- *si* $s = 0$ *on choisit avec un nouveau* k *et on ré-essaye* ;
- (r, s) est la **signature** !



L'algorithme de signature :

- ▷ génère un **secret** k ;
- ▷ ce secret est «*caché*» dans r grâce à la **multiplication de point** (facile dans un sens, dure dans l'autre) ;
- ▷ r est «*attaché*» au haché du document par l'équation $s = k^{-1}(z + r * d_A) \bmod n$;
- ▷ pour pouvoir calculer l'inverse de k modulo n, il faut que n soit un nombre premier (sinon pas de groupe !)
- ▷ n est connu d'Alice et Bob, il fait partie du système cryptographique avec a et b de la courbe elliptique
  $\Longrightarrow$ il existe **différents systèmes standardisés** et **prouvés sûrs**.

Vérification de la signature

- ▷ on calcule $u_1 = s^{-1}z \bmod n$ et $u_2 = s^{-1}r \bmod n$;
- ▷ on calcule le point $P = u_1G + u_2H_A$ et la signature est valide seulement si $r = x_p \bmod n$.

Pourquoi ? $P = u_1G + u_2H_A = u_1G + u_2d_AG = (u_1 + u_2d_A)G = (s^{-1}z + s^{-1}r * d_A)G = s^{-1}(z + r * d_A)G$
Et comme $s = k^{-1}(z + r * d_A) \bmod n$ ou $k = s^{-1}(z + r * d_A) \bmod n$, on obtient alors $P = kG$.

C'est quoi un inverse modulaire ?

L'inverse modulaire de a est un entier i tel que

- ◦ $a * i \equiv 1 \bmod n$;
- ◦ $i \equiv a^{-1} \bmod n$.

Exemple : $i \equiv 3^{-1} mod 11$ ou $3 * i \equiv 1 \bmod 11$, ce qui donne $i = 4$ car $3 * 4 = 12 \equiv 1 \bmod 11$.

ECDSA : à quoi cela ressemble ?

xterm

```
pef@darkstar-8:…/ecc/scripts$ python3 ecdsa.py
Curve: secp256k1
Private key: 0x28ecb4b7673815c2b0e49e3ad83e8bbb20ec5fa585f15fd76df01ed4064aa796
Public key:
(0xee19ee92a0e0d1b6271ade3b5e62d65fcd367e0405fa846a3123560aa152709e,
0xc68662c15eb123634652142af365f6798e7f960ca3b21df532c558a182427cc6)

Message: 'Bonjour !'
Signature: (0x737f1035e1de3e0474f3f4f46b10c661ecd6efa4329754aef5d6cdd7a61ec86a,
0xa80ba620a96da5d52a3a6a17089ceae699dd4f17eba7dd5a04500e0a5a43cd50)
Verification: signature matches

Message: 'Coucou !'
Verification: invalid signature

Message: 'Bonjour !'
Public key:
(0xcae60f6d2c9b80fa8b4b0d4afefaca78950933f0258ca6f66cf8be197a7d4184,
0xc9d273e354da7e989f94d2046f4327db6fba06c499afdeb1015709812781f2d0)
Verification: invalid signature
```

même message et la clé publique est la bonne

le message est différent

la clé publique est différente

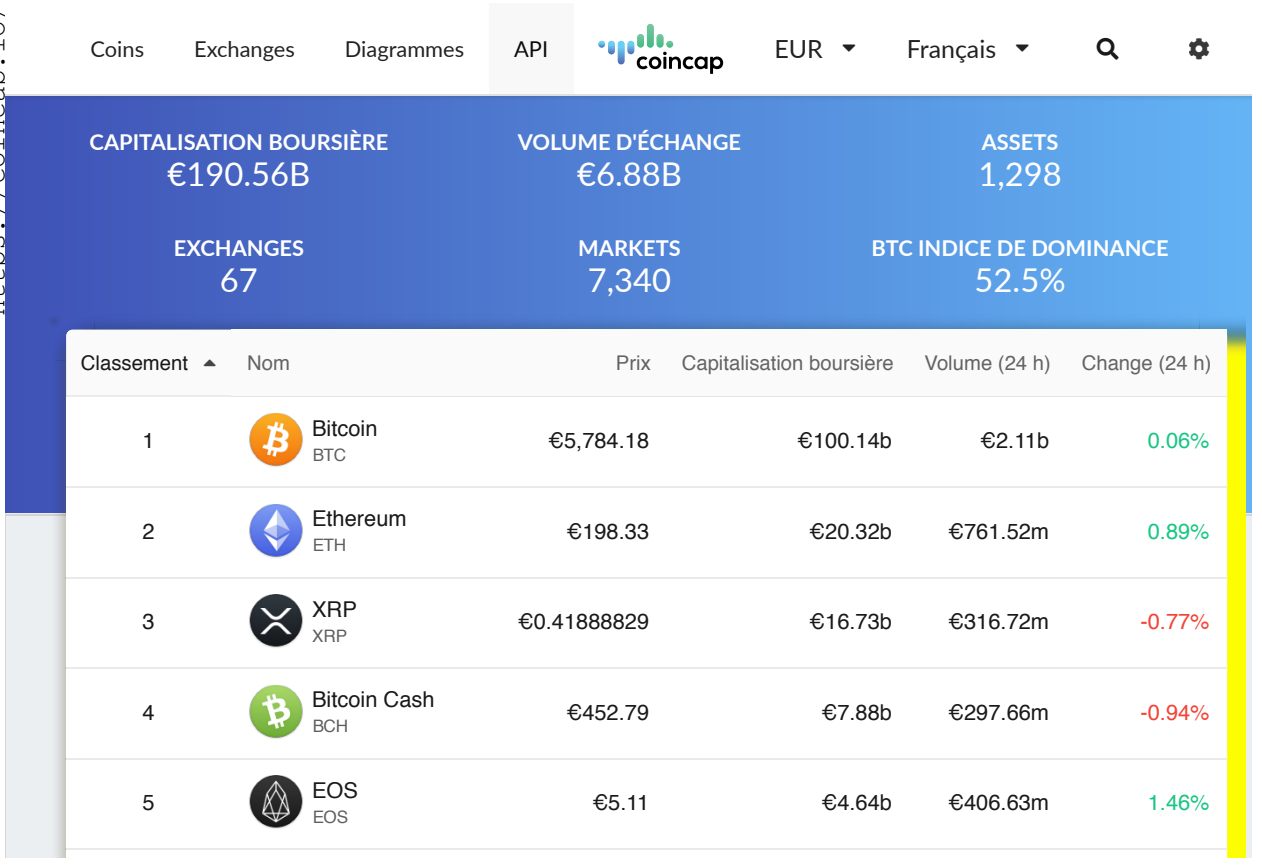
- la courbe utilisée dans Bitcoin est «*secp256k1*» ;
- La taille de la clé privée est de 32 octets ;
- la clé publique est un point dont chaque coordonnée est sur 32bits ;
- le message est le contenu du document qui va être haché pour donner la valeur z en entrée de l'algorithme.

Bitcoin ?

Oui cela repose sur les courbes elliptiques...

Bitcoin : argent cryptographique *vs* argent fiat

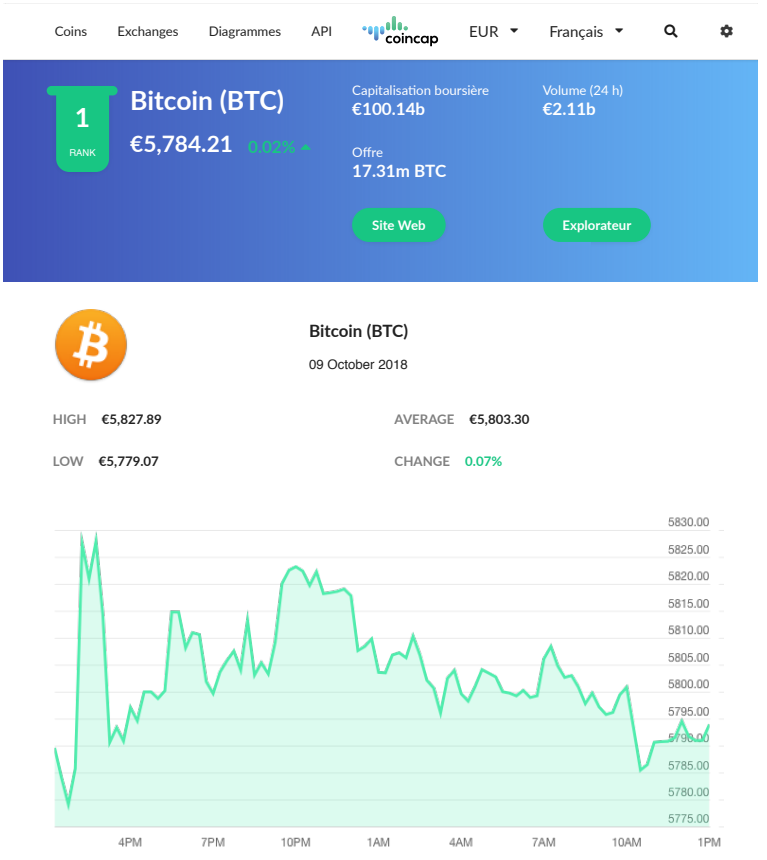
https://coincap.io/



Classement	Nom	Prix	Capitalisation boursière	Volume (24 h)	Change (24 h)
1	Bitcoin BTC	€5,784.18	€100.14b	€2.11b	0.06%
2	Ethereum ETH	€198.33	€20.32b	€761.52m	0.89%
3	XRP XRP	€0.41888829	€16.73b	€316.72m	-0.77%
4	Bitcoin Cash BCH	€452.79	€7.88b	€297.66m	-0.94%
5	EOS EOS	€5.11	€4.64b	€406.63m	1.46%

Bitcoin : argent cryptographique *vs* argent fiat

https://coincap.io/assets/bitcoin



Exemple : faire un don à un youtuber

EEVBlog

ABOUT

The EEVblog is the world's largest and most popular Youtube Engineering channel with over 450,000 subscribers, and hosts one of the most popular engineering community forums The EEVblog Forum (linked above).

The EEVblog is hosted by David L. Jones, an electronics design engineer based in Sydney Australia. In each episode he shares some of his 25+ years experience in the electronics design industry in his unique non-scripted naturally enthusiastic and passionate style.
David J. Ledger is the much younger, better looking, equally enthusiastic but less jaded sidekick.

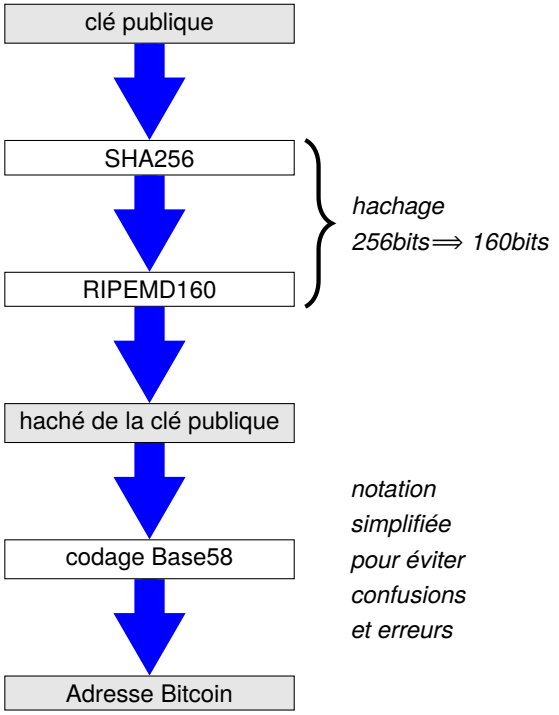
David L. Jones David J. Ledger

Adresse bitcoin :
17sH3EErEbAaqvfDFNzjmuPLrCvwdQwxQW

SUPPORT:

Producing and maintaining this blog takes a massive amount of time, and has be
like the blog and want to help support it, there are several ways you can do this:
1) Buy some of the products from my STORE
2) Buy anything from Amazon.com by clicking through here first (I make 6% on e
product, not just those in my store. Or use the Amazone search bar on the right
a real easy way to donate that costs you nothing!
3) Buy some of my Merchandise
4) You can donate money direct through Patreon. I much prefer this over PayPal
5) Donate directly through PayPal using david@alternatezone.com address. But
6) Donate Bitcoins: 17sH3EErEbAaqvfDFNzjmuPLrCvwdQwxQW

Adresse Bitcoin



Chaque utilisateur de bitcoin est identifié par une **adresse** :

- **dérivée** de la **clé publique** de l'utilisateur par une fonction de **hachage** ;
- **facile à échanger** entre individus grâce à une notation simplifiée ;
- **identifie de manière sûre** la clé à laquelle elle est liée tout en étant **plus petite** ;

L'**adresse Bitcoin** est exprimée en base58 :

- **éviter** les lettres et chiffres que l'on confond facilement : «*I*», «*L*», «*l*», ...
- contrôler que la **saisie est correcte** avec un code de contrôle constitué des 4 premiers octets du double haché de la valeur, préfixée par la valeur 1.

*Exemple de c**ode de contrôle** : le numéro de sécurité sociale est un nombre de 13 chiffres auquel on ajoute le code de contrôle calculé par la formule :* $code = 97 - (\textit{numéro ss} mod 97)$

*Une adresse bitcoin n'est pas forcément liée à une clé publique mais peut référencer un **script**, c-à-d un programme réalisant un contrôle sur la blockchain et qui peut déclencher des transactions automatiques.*

bitaddress.org

Adresse Bitcoin:

1N6d61fbTSj37EJXQSvQjvGTogVnWXgmqe

PARTAGER

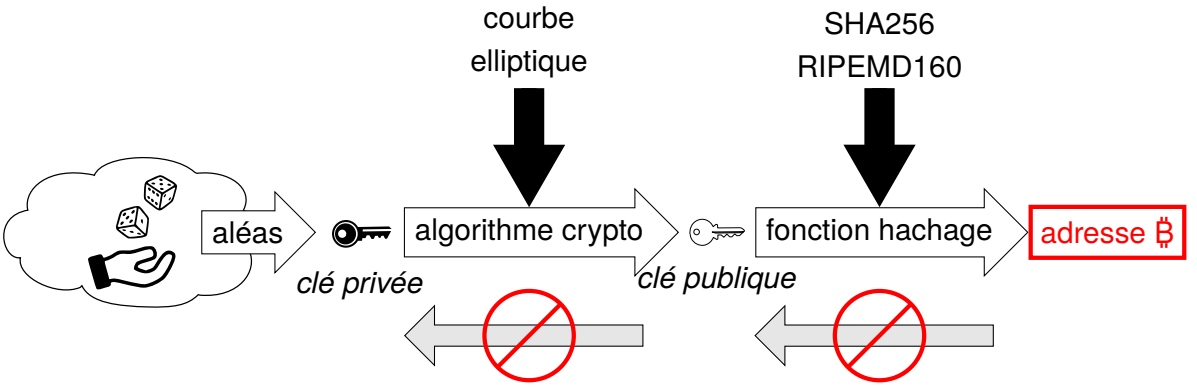
SECRET

Clé Privée (Format d'importation de porte-monnaie):

KwnKSa9VW4LCJbbSMZqAvT2VQrp63kngh3pW8YRqBJMZg7oHkFE8

- **l'adresse** est `1N6d61fbTSj37EJXQSvQjvGTogVnWXgmqe`
  On notera le 1 en préfixe qui désigne une adresse bitcoin. Pour un script le préfixe est 3.
- **Clé privée** : `KwnKSa9VW4LCJbbSMZqAvT2VQrp63kngh3pW8YRqBJMZg7oHkFE8`
- **Clé Publique** (130 caractères [0-9A-F]) :
  `04 434E97FD594C92AE550286DC2AD115914EEF1652C7317F119619630AEC202A7D`
  `538EB543D49FDB2BDBC7C9627E053B506D56009807484C60CA55F3240CDB6FE6`
  Ce sont les coordonnées (x, y) du point correspondant à la clé publique. Le préfixe est `04`.
- **Clé Publique** (**compressée**, 66 caractères [0-9A-F]) :
  `02 434E97FD594C92AE550286DC2AD115914EEF1652C7317F119619630AEC202A7D`
  Ce n'est que la coordonnées x. La coordonnée y peut être retrouvée grâce à la courbe elliptique support et le préfixe est `02` ou `3` afin d'indiquer si y est positif ou négatif (symétrie).

Adresse Bitcoin : un résumé



- l'espace des clés, c-à-d le nombre de clé différentes possibles est de 2^{256}, ce qui correspond à 10^{77} (le nombre d'atomes dans l'univers visible est estimé à 10^{80}).

Attention

- l'utilisation de **clé publique compressée**, c-à-d où il n'y a que la coordonnée x du point, permet de diminuer l'encombrement lié à la mémorisation de la clé lors d'une signature (50% d'économie) ;
- **mais** une clé publique compressée donne une **adresse Bitcoin différente** de celle non-compressée, c-à-d comportant les coordonnées (x, y) du point ;

$\Longrightarrow$ il faut en tenir compte lors de l'intégration de la signature pour rechercher **la bonne adresse** correspondant à la forme choisie pour enregistrer la clé publique !

La notion de portefeuille ou «*wallet*»

Un «*wallet*» est une structure :

- ▷ qui permet le **stockage de toutes les clés privées** de l'utilisateur ;
- ▷ il ne contient pas de bitcoins !
- ▷ les clés d'un portefeuille peuvent :
  - ◇ provenir de **valeurs aléatoires** $\Longrightarrow$ il faut les gérer une par une ;
  - ◇ être liées par un **algorithme de création de clés** utilisant une même valeur aléatoire pour générer l'ensemble des clés privées $\Longrightarrow$ on peut facilement les gérer car elles sont re-générables à la demande !

La **version papier** où sont imprimés :

- ▷ la **clé privée** ;
- ▷ la **clé publique** ;
- ▷ l'**adresse Bitcoin** ;

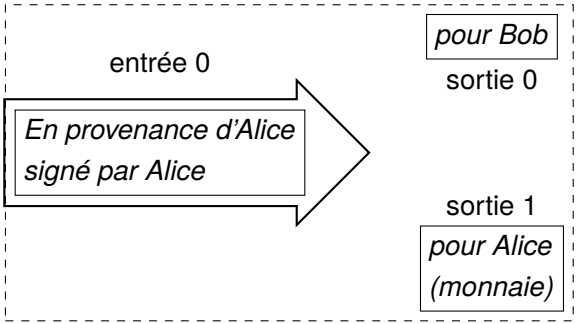
L'utilisation de QR Codes permettent de les saisir facilement sur un téléphone mobile.

La **version électronique** :

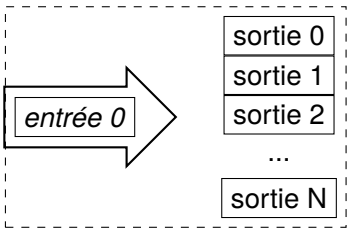
- ▷ elle permet de générer la ou les **clés privées** directement sur le terminal ;
- ▷ elle peut réaliser les **signatures**.

Ok pour les adresses

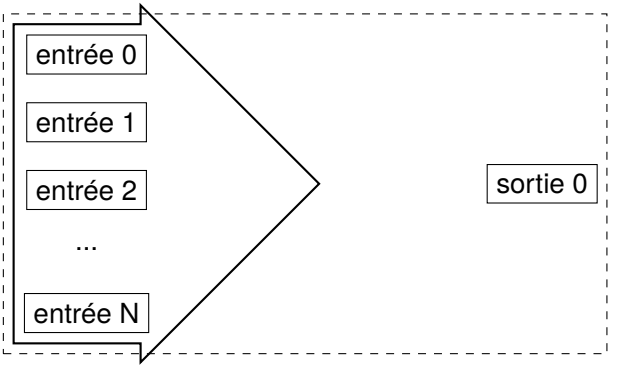
Et les transactions ?



Paiement unique : d'une adresse à une autre adresse avec le reste de l'entrée, *la monnaie*, retournée vers le propriétaire



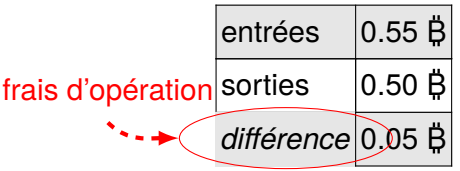
Paiement ventilé : distribuer une même entrée vers plusieurs adresses. Comme le paiment des salaires des différents employés d'une entreprise.



Paiement cumulé : on combine différentes entrées vers une seule sortie. Cela est comparable à combinées plusieurs pièces en un seul billet.

Journal des transactions

Tables des transactions			
entrées	*valeur*	*sorties*	*valeur*
entrée 1	0.10 ₿	sortie 1	0.10 ₿
entrée 2	0.20 ₿	sortie 2	0.20 ₿
entrée 3	0.10 ₿	sortie 3	0.20 ₿
entrée 4	0.15 ₿		
Total entrées :	0.55 ₿	**Total sorties :**	0.50 ₿



- les «*entrées*» sont des «*débits*» en provenance d'un compte bitcoin ;
- les «*sorties*» sont de «*crédits*» vers un compte bitcoin ;
- les «*frais d'opérations*» sont **versés au mineur** qui a ajouté la transaction à la blockchain ;
- pour chaque entrée, une preuve de propriété donnée par la **signature du propriétaire** peut être pouvée par n'importe qui ;
- «*dépenser*» correspond **à signer** la valeur d'une transaction précédente pour laquelle on possède la **clé privée** vers un nouveau propriétaire identifié par son **addresse bitcoin**.

Exemple : faire un don à un youtuber

BLOCKCHAIN

Bitcoin Address Addresses are identifiers which you use to send bitcoins to another person.

Summary	
Address	17sH3EErEbAaqvfDFNzjmuPLrCvwdQwxQW
Hash 160	4b52f0e4ae801a3e52512d527292c9eb48269d6d

Transactions	
No. Transactions	4
Total Received	€ 599.22
Final Balance	€ 0.00

Consultation de l'adresse Bitcoin sur :

`https://www.blockchain.com/btc/address/17sH3EErEbAaqvfDFNzjmuPLrCvwdQwxQW`

On peut voir la liste des transactions associées :

Transactions (Oldest First)

Filter

6c08ed9d8c4defc650bed413b63bd67cba3d3cf66d2e04df1a61ff13f08a5752 — 2017-12-24 11:42:42

17sH3EErEbAaqvfDFNzjmuPLrCvwdQwxQW → 1B54QDdqiCon5UjUv8kMZ9B5RSPYqFtpVm € 318.53
1L6aCBbrBUkvQoExGbMaEmPcJAttLy5Ta5 € 245.00

€ -576.56

99f006caa25dc3a3fb7f15b6f75a06d162edcccc1b8be288170a538a54a1592a — 2017-12-24 11:16:37

1sdMr4sFCZbgAveS91Jkvv6Y7wEQ4KiMM → 17sH3EErEbAaqvfDFNzjmuPLrCvwdQwxQW € 576.56

€ 576.56

9e6b103f2b35f60a073de6c943a5e48778ef920adbf2343a119bda43fa2908ea — 2017-07-18 16:32:15

Le détail d'une transaction

BLOCKCHAIN

Transaction View information about a bitcoin transaction

6c08ed9d8c4defc650bed413b63bd67cba3d3cf66d2e04df1a61ff13f08a5752

17sH3EErEbAaqvfDFNzjmuPLrCvwdQwxQW	1B54QDdqiCon5UjUv8kMZ9B5RSPYqFtpVm	€ 318.58
	1L6aCBbrBUkvQoExGbMaEmPcJAttLy5Ta5	€ 245.03
		€ 563.61

Summary	
Size	226 (bytes)
Weight	904
Received Time	2017-12-24 11:42:42
Included In Blocks	500822 (2017-12-24 12:04:43 + 22 minutes)
Confirmations	44235
Visualize	View Tree Chart

Inputs and Outputs	
Total Input	€ 576.64
Total Output	€ 563.61
Fees	€ 13.03
Fee per byte	1,000 sat/B
Fee per weight unit	250 sat/WU
Estimated BTC Transacted	€ 245.03
Scripts	Show scripts & coinbase

Une transaction **indique au réseau** que le propriétaire de bitcoins **a autorisé** le **transfert** de certains de ces bitcoins **vers un nouveau propriétaire**⟹ il a utilisé sa clé privée pour signer un *transfert de propriété* vers un nouveau propriétaire identifié par son **adresse bitcoin**.

Le **nouveau propriétaire** peut maintenant :

- ▷ **prendre propriété** des bitcoins en **prouvant** qu'il en est propriétaire, car il peut utiliser la **clé privée**, liée à la **clé publique**, liée à l'**adresse bitcoin** utilisée par l'ancien propriétaire pour désigner le nouveau propriétaire ;
- ▷ **dépenser** ces bitcoins en créant une **nouvelle transaction** pour transférer la propriété vers un **nouveau propriétaire**...

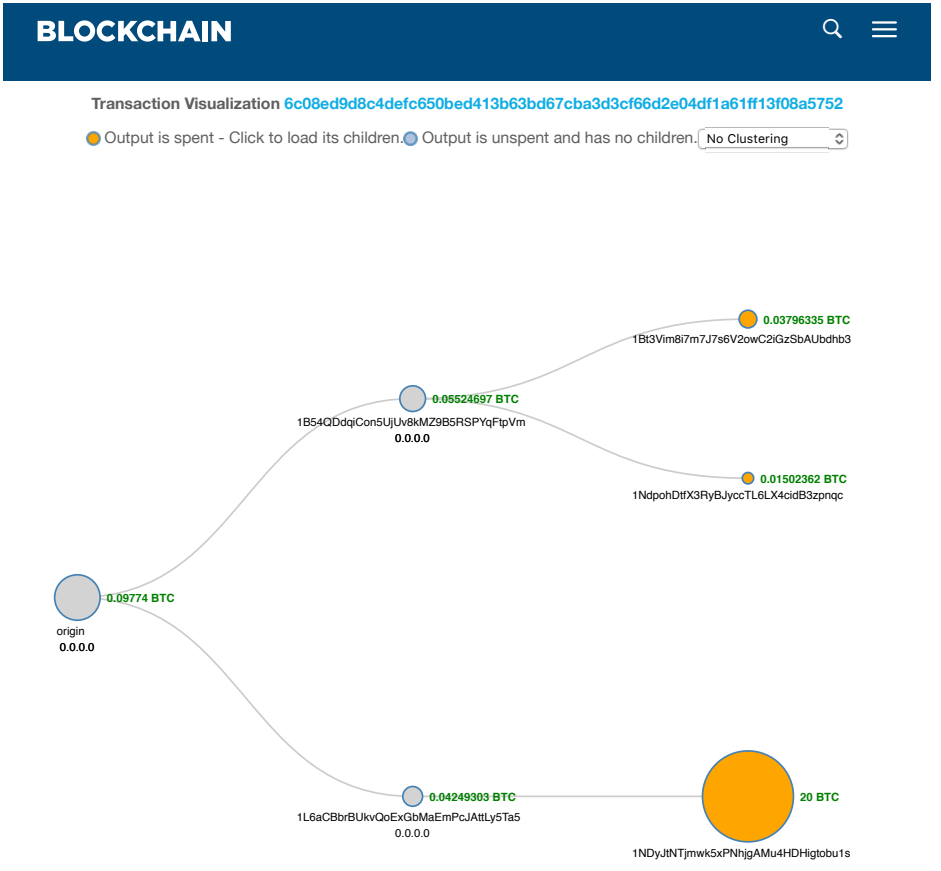
transaction 43	haché du bloc	
entrée 0	Depuis Paul, avec la signature de Paul	0,1005 ₿
sortie 0	Vers l'adresse d'Alice	0,1000 ₿

transaction 98	haché du bloc	
entrée 0	Depuis Transaction 43, numéro 0, avec la signature d'Alice	0,1000 ₿
sortie 0	Vers l'adresse de Bob	0,0150 ₿
sortie 1	Vers l'adresse d'Alice	0,0805 ₿

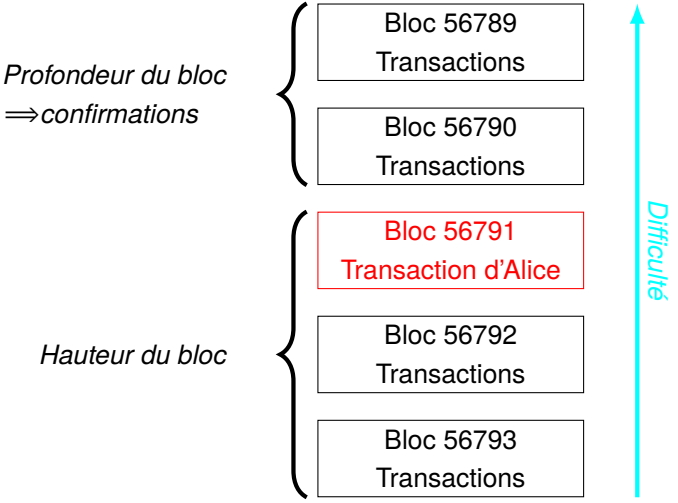
transaction 130	haché du bloc	
entrée 0	Depuis Transaction 98, numéro 0, avec la signature de Bob	0,0150 ₿
sortie 0	Vers l'adresse de Carole	0,0100 ₿
sortie 1	Vers l'adresse de Bob	0,0050 ₿

Les liens entre les transactions

https://www.blockchain.com/btc/tree/316514407



Finalisation de la transaction



- ▷ une transaction est transmise au travers du réseau :
  - ◇ elle est **ajoutée** à un «*pool*» de transaction non vérifiée présent sur chaque mineur ;
  - ◇ elle ne sera **définitive** que lorsqu'elle sera ajoutée au blockchain ;
  - ◇ les transactions sont ajoutées suivant l'**ordre défini** par la valeur des **frais de gestion** et le temps que la transaction a attendu ;
- ▷ la transaction est **vérifiée** :
  - ◇ les entrées doivent avoir été **signées** par l'ancien propriétaire vers l'adresse d'Alice ;
  - ◇ Alice a réalisé une **signature** utilisant la clé privé liée à la clé publique liée à l'adresse indiquée ;
- ▷ lorsque le bloc précédent a été **intégré dans la blockchain**, le nouveau bloc est traité et intègre le haché du bloc précédent ;
- ▷ la recherche de la «*preuve de travail*» débute aussitôt et le **mineur** intègre dans son bloc courant **son adresse bitcoin**

⟹si le mineur est le premier à réussir la «*preuve de travail*» et que son bloc est **intégré dans la blockchain**, c'est grâce à **son identité** qu'il recevra la **rémunération** associée, ainsi que le **total des frais de gestion** du bloc.

- ▷ à chaque nouveau bloc ajouté par-dessus celui contenant la transaction d'Alice, il ajoute **plus de travail** sur ce bloc ⟹ la **confiance** dans les transactions du bloc **augmente** !
- ▷ un **bloc ajouté** correspond à une **confirmation** de la transaction

  ⟹on considère qu'après 6 blocs de confirmations la transaction ne peut être défaite et devient **permanente**.

Et le premier bloc ?

https://www.blockchain.com/btc/block-height/0

Block Height 0 Blocks at depth 0 in the bitcoin blockchain

Summary	
Height	0 (Main chain)
Hash	000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
Previous Block	0000000000000000000000000000000000000000000000000000000000000000
Next Blocks	00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048
Time	2009-01-03 18:15:05
Difficulty	1
Bits	486604799
Number Of Transactions	1
Output Total	€ 288,019.00
Estimated Transaction Volume	€ 0.00
Size	0.285 KB
Version	1
Merkle Root	4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b
Nonce	2083236893
Block Reward	€ 288,019.00
Transaction Fees	€ 0.00

Bitcoin ? C’est vivant !

Sur https://insight.bitpay.com

insight Blocks Status Search for block, transaction or ad Scan BTC

Latest Blocks

Height	Age	Transactions	Mined by	Size
544945	13 minutes ago	1160		451054
544944	21 minutes ago	2364		884208
544943	37 minutes ago	1553		739158
544942	43 minutes ago	2081		957045
544941	an hour ago	2702	SlushPool	926041

See all blocks

Latest Transactions

Hash	Value Out
b67b19f04bf5edd14bc10e065ff3d4da3178f3a0c859479cd1…	0.14574995 BTC
0abc8b298a93b438ab4a6a5b4d4b631e5558fdec6989e8b36…	0.03558659 BTC
156a2024e6b0681fc5869fc3e43fecbc03c36d4fbcd2681701d…	0.000988 BTC
126146ce39e403c043f8c9014f49bc69d672f8e7231f54eddfd…	0.41424885 BTC

Mais...et la création de monnaie ?

La transaction peut être faite «*off-line*»

Le fonctionnement est similaire à celui du chèque : on utilise un logiciel client, qui s'occupe de :

- ☐ déterminer la liste **d'entrées** à utiliser : les sorties de transaction précédentes non encore dépensées et dont on est la destination ;
- ☐ récupérer quels sont les **sorties**, c-à-d les adresses des bénéficiaires auprès de l'utilisateur grâce aux adresses bitcoin ;
- ☐ réalise la **signature** de la transaction grâce à la clé privée de l'utilisateur ;

Pour réaliser ce travail, le **logiciel client** peut :

- ▷ faire une requête aux nœuds du réseau afin de connaître **l'ensemble des sorties non encore dépensées** liées à une adresse bitcoin donnée ;
- ▷ vérifier les transactions :
  - ◇ **confirmer** qu'une transaction est bien dans la **blockchain** ;
  - ◇ **vérifier** que la transaction est à une certaine «*profondeur*», c-à-d qu'il y a un certain nombre de blocs qui suivent le bloc contenant la transaction ;
- ▷ déterminer si la transaction **consomme l'intégralité** des entrées ou nécessite de transférer la «*monnaie*» **en retour** à l'utilisateur ;
- ▷ ajouter des **frais de gestion**, «*fee*», à l'intention du **mineur** pour :
  - ◇ inciter le mineur à inclure la transaction dans un bloc ;
  - ◇ empêcher d'abuser du système ;

  Elle est calculé non pas sur la valeur de la transaction, mais sur la taille de cette transaction !

  Une transaction sans fee *n'est pas prioritaire et peut même ne pas être traitée...*

Évaluation des frais de gestion

On peut demander la valeur des frais de gestion sur l'URL :

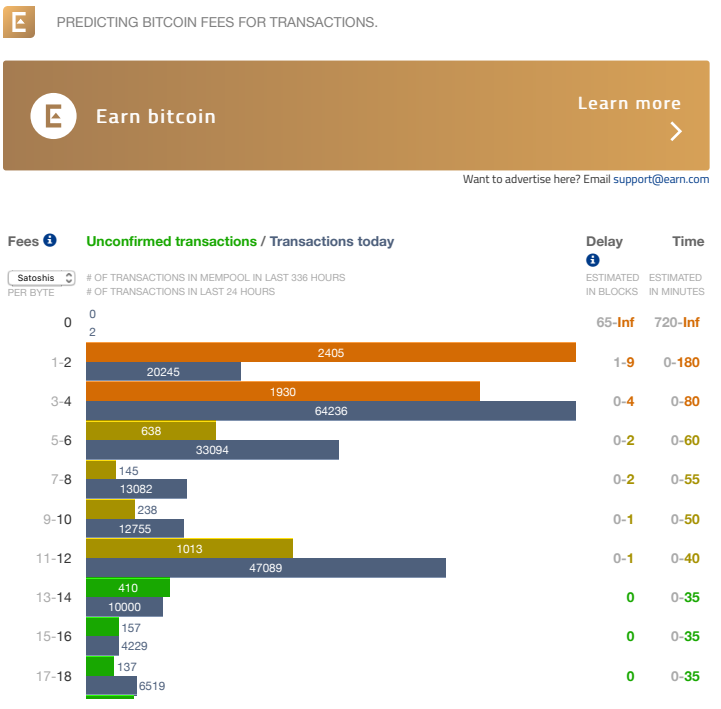
`https://bitcoinfees.earn.com/api/v1/fees/recommended`

```
{"fastestFee":14,"halfHourFee":14,"hourFee":6}
```

*Les valeurs sont indiquées en «*satoshis*» par octet et une tr*
saction moyenne est de 225 octets, soient pour 14 satoshis/
Ƀ.

- ▷ «*fastestFee*» : le prix le plus bas pour une confirmation
  pide de la transaction (attente d'un bloc
- ▷ «*halfHourFee*», «*hourFee*» : le prix le plus bas pour u
  confirmation dans la demi heure ou dans l'heure.

`https://bitcoinfees.earn.com/# fees`



La notion d'UTXO

L'ensemble des **sorties non dépensées** est appelé l'ensemble **UTXO**, «*Unspent Transaction Output*».

Chaque **transaction** représente un **changement** de cet ensemble :

- **recevoir un paiement** en bitcoins ⟹ le logiciel client détecte une UTXO qui peut être dépensée par une clé privée possédée par l'utilisateur ;
- établir le **solde du compte** de l'utilisateur correspond à :
  - établir la liste de ces UTXO qui peuvent être réparties entre des centaines de transactions et des centaines de blocs ;
  - faire la somme de toutes ces UTXO ;
- une UTXO correspond à une **valeur entière** exprimée en satoshis qui est **indivisible** ;
- si une UTXO est de valeur supérieure à la dépense, par exemple de 10 ₿ et que l'on veut dépenser 2 ₿, alors il faut construire une transaction avec **deux sorties** :
  - une de 2 ₿ vers le bénéficiaire de la dépense ;
  - une de 18 ₿ en retour vers le propriétaire initial ;

Transaction spéciale

- la **première transaction** de chaque bloc ;
- placée là par le mineur gagnant, c-à-d dont le bloc qu'il a miné est intégré dans la blockchain ;
- n'utilise pas d'UTXO :
  ⟹ appelée transaction «*coinbase*» ;
  ⟹ correspond à la **création de nouveaux bitcoins** !

La «*Preuve de Travail*»

- ▷ les transactions sont **diffusées** à tous les nœuds du réseau ;
- ▷ les mineurs récupèrent ces transactions pour **construire un bloc candidat** ;
- ▷ chaque mineur introduit un «*nonce*» dans le contenu du bloc et le modifie jusqu'à obtenir un hash plus petit que celui du bloc sans ce nonce ⟹ celui qui le fait le **plus rapidement** est **rémunéré** et son bloc est ajouté à la **blockchain**.

Jan 2009-Nov 2012	50 ₿ par bloc
Nov 2012-Jul 2016	25 ₿ par bloc
Jul 2016-Feb 2020	12.5 ₿ par bloc
Fév 2020-Sep 2023	6.25 ₿ par bloc
2140	arrêt de la création de bitcoin à 21 millions

Un mineur qui dispose d'un matériel puissant peut calculer plus rapidement des hashs⟹ probabilité qu'il reçoive une rémunération pour le calcul d'un bloc augmente.

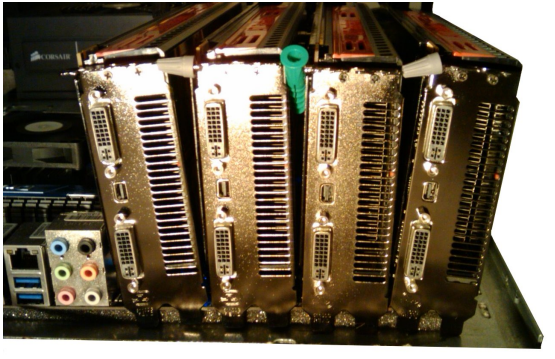
La difficulté de la* «preuve de travail» ***est ajustée au cours du temps pour viser 10 mins par bloc.***

Matériel dédié

Les mineurs peuvent utiliser :

- le **CPU** : le processeur de l'ordinateur ;
- le **GPU** : des cartes graphiques utilisées pour leur puissance de calcul ;
- des **composants dédiés** : ASICs,«*application-specific integrated circuit*», FPGA, «*Field Programmable Gate Array*»

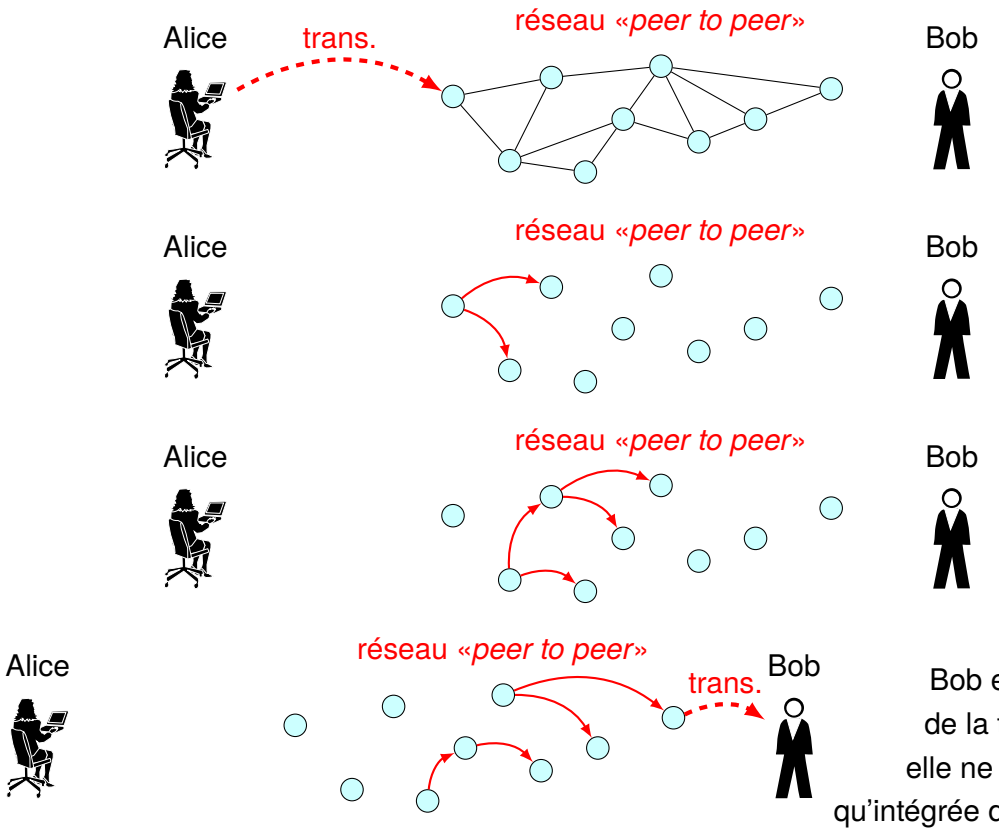
⟹Le **coût du matériel** et le prix de **l'électricité consommée** doivent être **compensés** par la rémunération en bitcoin.



Et le «*réseau Bitcoin*» ?

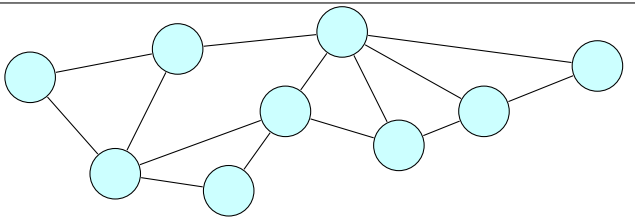
La transmission de la transaction vers les mineurs

La transaction est transmise à un nœud du réseau «*peer to peer*» qui diffuse à ses voisins :



Bob est informé
de la transaction
elle ne sera validée
qu'intégrée dans la blockchain

Comment sont trouvés les nœuds du réseau ?



Une liste d'adresses IP connues obtenues à partir de nom de domaines

- `seed.bitcoin.sipa.be`
- `dnsseed.bluematt.me`
- `dnsseed.bitcoin.dashjr.org`
- `seed.bitcoinstats.com`
- `seed.bitcoin.jonasschnelli.ch`
- `seed.btc.petertodd.org`

Des canaux IRC, «*Internet Relay Chat*» :

- `#bitcoin00`

à `#bitcoin99` sur `irc.lfnet.org`

- «*Il y a eu la séparation de l'Église et de l'État, il est temps de réaliser la séparation de la Monnaie et de l'État. Erik Voorhees, CEO de Shapeshift.io*»
  - ◇ l'argent est aussi fondamental que la religion : il influence comment votre vie va se dérouler ; les choix que vous faites influencent votre vie et la vie de ceux qui vous entourent ;

  ⟹ que l'argent soit contrôlé par une une entité centralisée et monopolistique devient **immoral** ;
  - ◇ de même qu'il y a plusieurs religions autorisées, il doit y avoir plusieurs monnaies autorisées ;

  ⟹ cela amène de la **compétition** entre la monnaie fiat et celle cryptographique.

- La force de bitcoin est de **motiver** les acteurs à vivre ensemble :
  - ◇ ceux qui possèdent des bitcoins ont intérêt à ce que le modèle perdure sinon ils déprécient la valeur qu'ils représentent ;
  - ◇ ceux qui veulent attaquer le bitcoin et en tirer profit ont le même intérêt...

- «*Crypto-monnaie*»/«*Token*» définie par une entreprise dans le cadre de son activité :

  Exemple : un service de «*mise en relation*» permet à des utilisateurs d'en contacter d'autres pour une expertise :
  - ◇ ils payent pour l'expertise avec de l'argent fiat qui est convertie en crypto-monnaie pour payer l'expert ;
  - ◇ l'expert peut choisir de payer un autre expert ou bien de convertir sa crypto-monnaie en argent fiat ;
  - ◇ si le service se développe :
    - ⋆ la crypto-monnaie peut prendre de la valeur par rapport à sa conversion en monnaie fiat ;
    - ⋆ les experts peuvent avoir envie de conserver leur crypto-monnaie en espérant que sa valeur augmente ;
    - ⋆ l'entreprise offrant le service peut disposer d'un «*fond*» au travers de la crypto-monnaie créée et non convertie.

- IPO *vs* ICO :
  - ◇ «*Initial Public Offering*» : une entreprise **augmente son capital** en devenant **publique** et en offrant à des investisseurs extérieurs d'acheter des **actions** dont la valeur va varier librement en fonction du marché ;
  - ◇ «*Initial Coin Offering*» : les prises de participation dans l'entreprise sont faites en **token** de l'entreprise qui prendront de la valeur suivant le **succès** de l'entreprise, et pourront être échangés sur des marchés les acceptant vers d'autres tokens ou de l'argent.

Merci !