Cryptographie

Une introduction

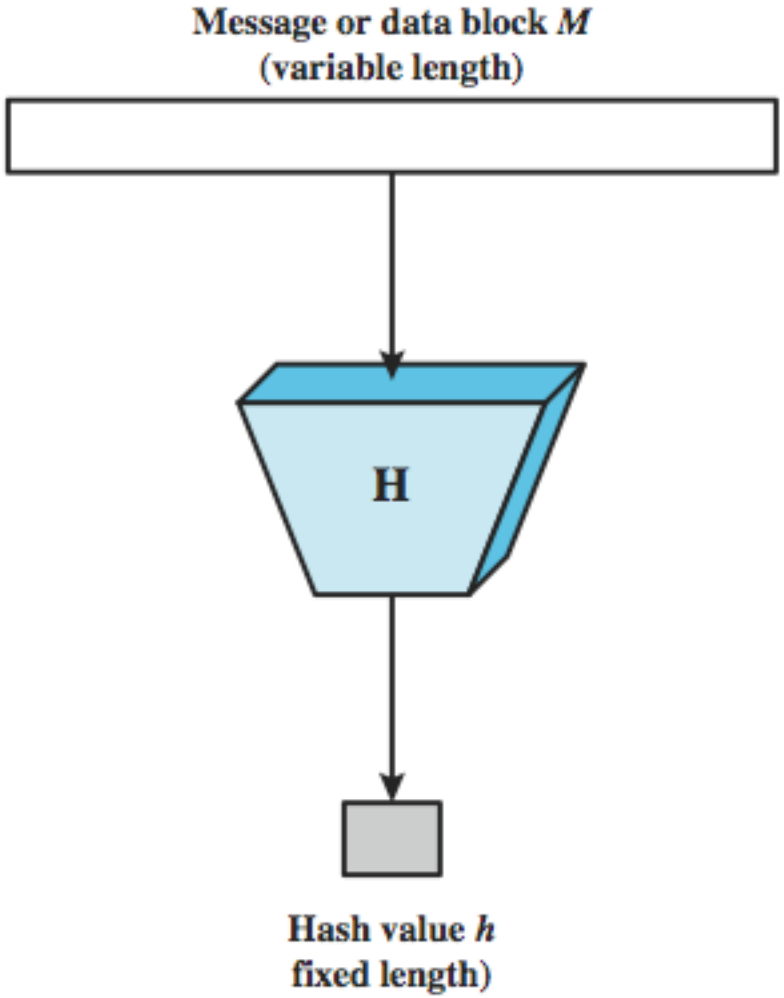
A. Bonnecaze

Polytech/IML

Cours 2

- Fonctions de hachage
- Applications des fonctions de hachage

Fonctions de hachage



Fct de hachage cryptographique

- S'applique à n'importe quelle taille de donnée
- H produit un output (l'empreinte) de longueur fixe
- H(x) est facile à calculer pour un x donné
- Sans unique (one-way property)
  - Impossible calculatoirement de trouver x tel que H(x) = h
- weak collision resistance (2eme préimage résistante)
  - Impossible calculatoirement de trouver $y \neq x$ tel que H(y) = H(x)
- strong collision resistance
  - Impossible calculatoirement de trouver (x, y) tel que H(x) = H(y)

Le paradoxe des anniversaires

- Dans une classe, quelle est la probabilité pour que 2 élèves fêtent leurs anniversaires le même jour?
- L'année compte 365 jours.
- La proba est-elle grande lorsque le nombre d'élèves est d'environ 50?

Le paradoxe des anniversaires

- probabilité pour que, dans un groupe de k personnes, ces personnes aient toutes un jour d'anniversaire différent :
- Pk=(1-1/365)(1-2/365)...(1-(k-1)/365)
- P1=1 ; P2=0.99 ; P23=0.49 ; P50=0.03
- Donc dans une classe de 23 élèves, plus d'une chance sur deux que deux élèves aient la même date d'anniversaire

Conséquence crypto

- Combien l'attaquant doit-il essayer de textes avant de trouver la même empreinte avec une probabilité d'au moins un demi?

Taille du haché en bits	Nbre de textes à essayer
8	13
32	54562
128	1.5×10^{19}
256	2.8×10^{38}

Sécurité

- Deux approches d'attaques
  - cryptanalyse
    - exploite les points faibles de l'algo
  - Attaque par force brute
    - Essayer des inputs
    - Difficulté proportionnelle à la taille de la fct
- L'algo le plus utilisé : SHA (Secure Hash Algo)
  - SHA-1, SHA-2, SHA-3
  - Allez sur la page :

http://csrc.nist.gov/groups/ST/hash/index.html

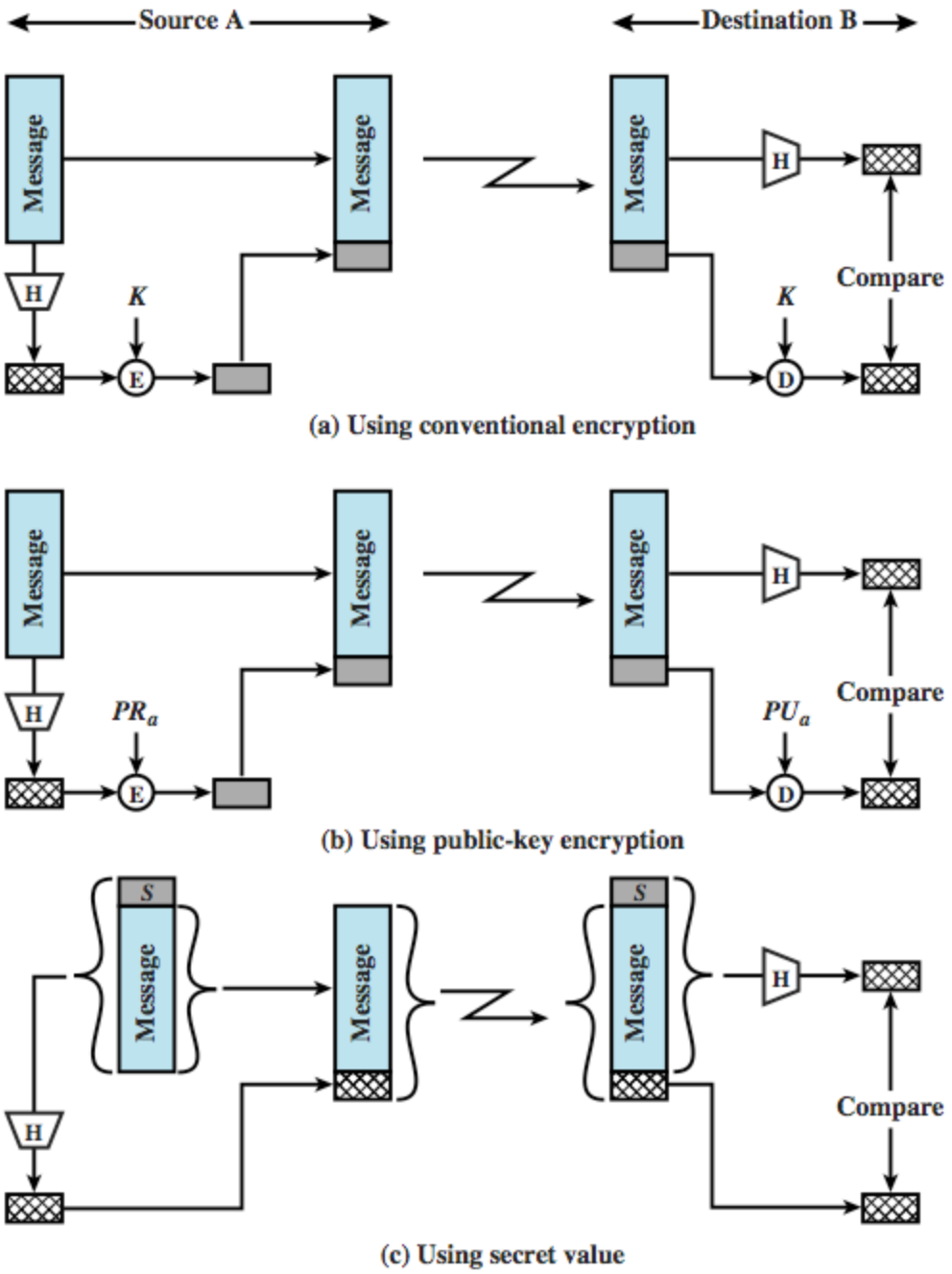
Applications des fonctions de hachage

- MAC / Signatures
- Clés hierarchiques
- S/Key one time password (voir TD)
- Merkle tree
- Commitment
- Blockchain
- ...

Authentification de message

- Protège contre les attaques actives
- Vérifie l'authenticité du message reçu
  - Contenu non altéré
  - Vient d'une source authentique
  - Date et séquencement correct
- Message Authentication Code (MAC)
  - Seuls sender & receiver ont la clé (secrete)
- Signature digitale (clé publique/privée)
- Modif.Detect.Code
  - Appose un *authentication tag* (hash) au message (tag envoyé sur canal sûr)

Message Auth



Signature digitale

RSA : sécurité basée sur le problème RSA

ECDSA : sécurité basée sur le DLP (voir FIPS 186-3) sur courbe elliptique

BLS : signature courte basée sur les pairings

Lamport (One time signature)

One time signature (Lamport)

Génération de clés :

Alice génère 256 paires de nombre aléatoires de 256 bits : c'est la clé privée de taille 16 KiB

Clé publique : elle hache chacun des 512 nombres aléatoires et obtient 256 paires de hachés

Signature :

Alice hache le message m (empreinte de 256 bits)

Pour chaque bit du haché elle choisit dans l'ordre un élément de la paire de la clé privée : si le bit est 0, elle choisit le premier de la paire ; si le bit est 1, elle choisit le second.

La longueur de la signature est 8 KiB

One time signature

La clé privée ne peut être utilisée qu'une seule fois !

Vérification :

Bob hache le message m : obtient 256 bits

Bob hache les nombres de la signature

Pour chaque bit de H(m), il fait la même chose qu'Alice mais sur la clé publique

Il obtient 256 nombres qu'il compare aux hachés des éléments de la signature

One time signature

H : one way function, output of 256 bits

Private key : $y_{i,j}$ i in [1..256], j in {0,1}

public key : $H(y_{i,j}) = z_{i,j}$

M = H(m) = m1 … m256 (256 bits)

- $S = y_{1,m1} y_{2,m2} \dots y_{256,m256} = (s_1, \dots, s_{256})$

Vérification

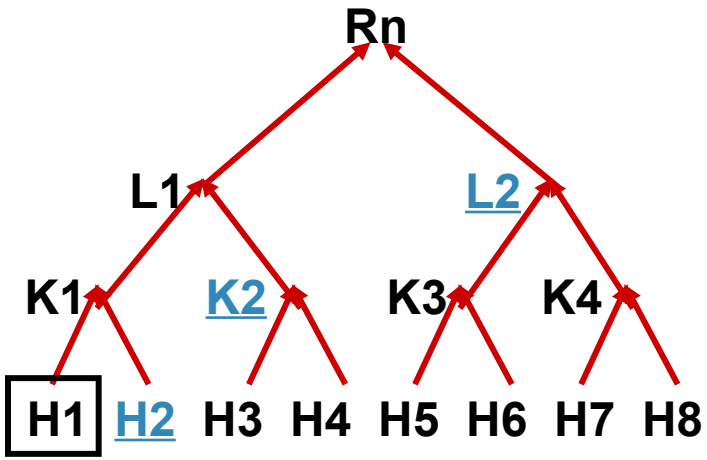
- $H(s_i) = ? \ z_{i,mi}$

Sécurité = sécurité de la fonction de hachage H

La sécurité ne repose pas sur un problème difficile (postquantum)

Signature de Merkle

Une clé publiée pour plusieurs messages



Exemple avec 8 feuilles
Hi = H(yi), yi étant la i^{ème} clé publique

La clé publique publiée est la racine de l'arbre

Sign = (s||yi||H2||K2||L2)

Où s est une one time signature

MAC (assure intégrité mais pas confidentialité)

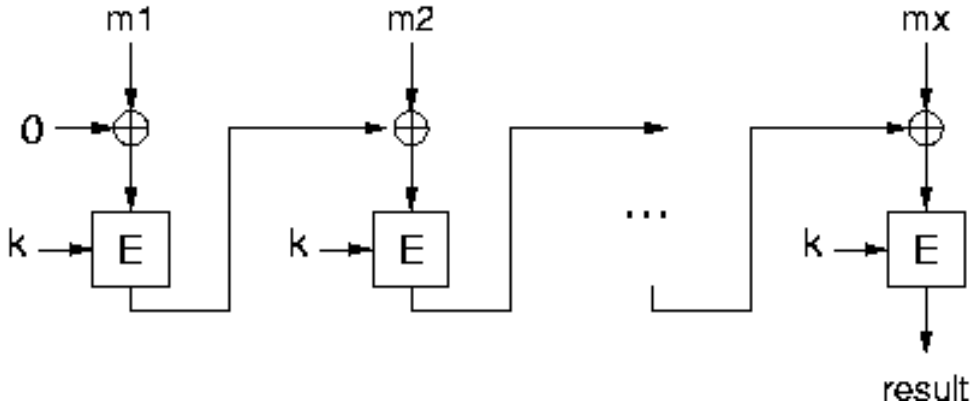
Un MAC est sûr si il n'est pas possible calculatoirement d'obtenir une paire (m',t') valide à partir d'un ensemble de paires valides (m_i,t_i)

Il utilise une clé secrète

Il existe de nombreuses constructions de MAC (ECBC-MAC, CMAC, NMAC, HMAC, PMAC, ...)

Comment construire un MAC ?

rawCBC-MAC



E est une PRF comme
Par exemple AES

rawCBC-MAC n'est pas sûr :

soit m = message arbitraire de taille 1 bloc

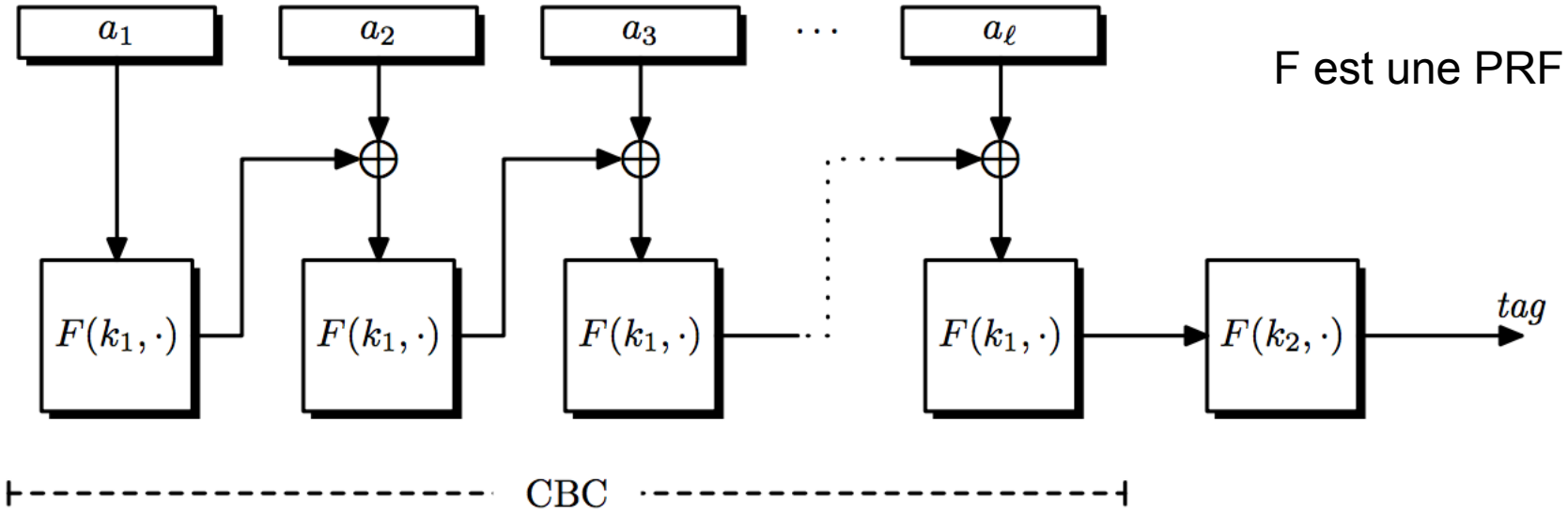
rawCBC(m) = E(k,m) = t

On va construire un autre message ayant le même MAC

m' = m||t⊕m (message de 2 blocs)

rawCBC(m') = E(k, E(k, m)⊕t⊕m) = E(k, m) = t

ECBC-MAC



F est une PRF

Le second chiffrement empêche l'attaque précédente.

ECBC

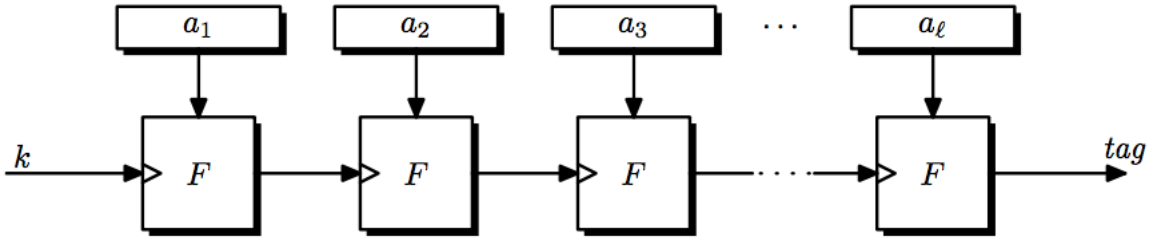
ECBC-MAC est généralement utilisé avec AES

Utilisé dans mode de chiffrement CCM : *Counter with CBC-MAC* qui permet d'obtenir confidentialité et authentification (utilisé dans 802.11i, IPSEC, TLS, Bluetooth Low Energy, etc)

Base pour CMAC en tant que standard du NIST

NMAC

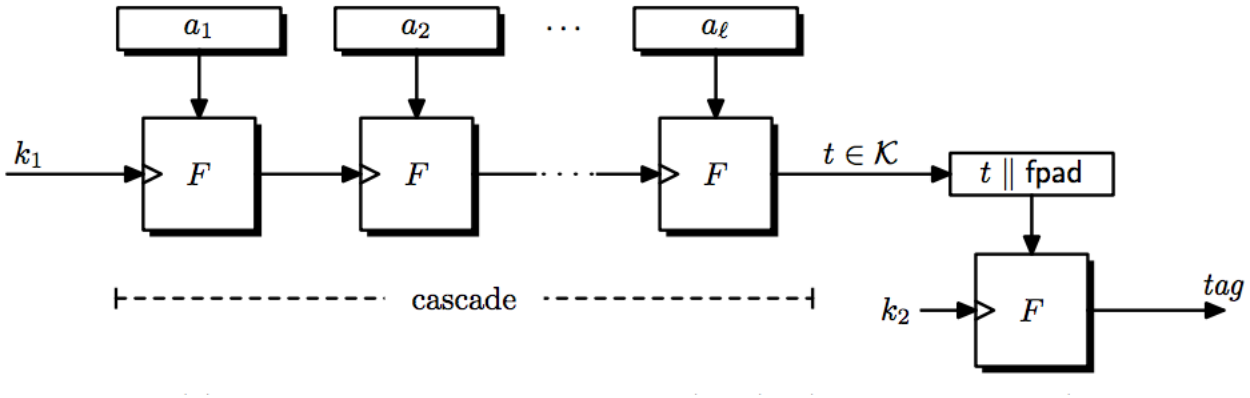
Construire un MAC à partir d'une PRF :



Cette construction n'est pas sûre :

Soit m un message arbitraire, à partir de la donnée de MAC(k,m), on peut calculer sans la clé MAC(k, m || m1)

NMAC



Pour éviter l'attaque, il faut rechiffrer avec une autre clé

fpad est un padding pour obtenir la taille d'un bloc

NMAC

Généralement pas utilisé avec AES ou 3DES car la clé change à chaque bloc

AES n'est pas efficace lorsque la clé change à chaque bloc

NMAC est à la base de HMAC

HMAC

H est une fonction de hachage

K la clé secrète complétée par des zéros

M le message à authentifier

ipad = 0x363636...3636 (inner)

opad = 0x5c5c5c...5c5c (outer)

HMACk(m) = H((k⊕opad)||H(k⊕ipad)||m)

Utilisation de HMAC : voir FIPS PUB 198

KDF (Key Derivation Function)

- A partir d'une suite d'octets en entrée, on récupère une suite d'octets de taille donnée (ici la taille peut être variable)
- H(x,n) = z
- A une suite d'octets x et un entier n, fait correspondre une suite d'octets de longueur n

Clés hierarchiques

- Dans la société d'Oscar, le personnel a un niveau d'habilitation (1,2,3)
- Oscar a le niveau 1 : il a accès a tous les documents de sa société (niveaux 1,2,3)
- Maryline (secrétaire personnelle d'Oscar) a le niveau 2 : elle a accès à tous les documents de niveaux 2 et 3
- Carla a le niveau 3 : elle n'a pas accès aux niveaux 1 et 2

- Avec sa clé, Oscar veut pouvoir déchiffrer tous les documents, Maryline doit pouvoir déchiffrer aux niveaux 2 et 3 et Carla uniquement au niveau 3.

Clés hierarchiques

h est une fonction à sens unique publique

Clé K1
Il peut calculer h(K1) et h(h(K1))
Il peut déchiffrer tous les documents

Clé K2 = h(k1)
Elle peut calculer h(K2)
Elle peut déchiffrer les documents de niveaux 2 et 3

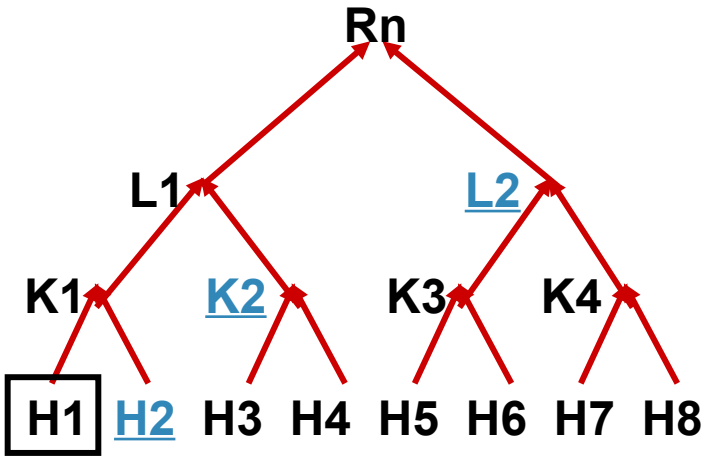
Clé K3 = h(K2)
Elle ne peut déchiffrer que les documents de niveau 3

Arbre de Merkle
(datation d'un document)

- Soit E = {D1,...,Dn} l'ensemble des fichiers créés au temps T
- Iwan prétend que le fichier X a été créé au temps T (donc appartient à E)
- Alice veut le vérifier
- Mais n est grand et les fichiers sont gros
- Alice doit comparer X avec les Di
- Si X appartient à E il lui faut faire n/2 comparaisons en moyenne
- Sinon, elle aura fait n comparaisons (complexité linéaire !)

Arbre de Merkle

S = (H1, Rn, H1 to Rn)



Complexité : Log(n) opérations

Rn est publié

Et si le nombre de documents n'est pas une puissance de 2?

Commitment

Alice veut jouer à Pile ou Face avec Bob en communiquant par internet

Bob choisit Pile et le dit à Alice

Alice tire au sort : c'est Face

Comment Bob est-il sûr qu'Alice n'a pas triché?

Bob devrait choisir Pile ou Face et donner à Alice une preuve de son choix sans le divulguer

Pile ou face

- Bob choisit Pile et calcule b = h(Pile)
- Bob envoie b à Alice
- Alice tire au sort : Face, calcule h(Face) et compare avec b
- Comment modifier le protocole pour qu'il devienne sûr?
- Rajouter un peu de sel?

Blockchain

But : stockage et transmission d'informations

- Transparente
- Sécurisée
- Sans organe central de contrôle

c'est une BD distribuée qui contient l'historique de tous les échanges effectués

Chacun peut vérifier la validité de la chaîne

Blockchains publiques ou privées

Utilise des fonctions de hachage et signatures

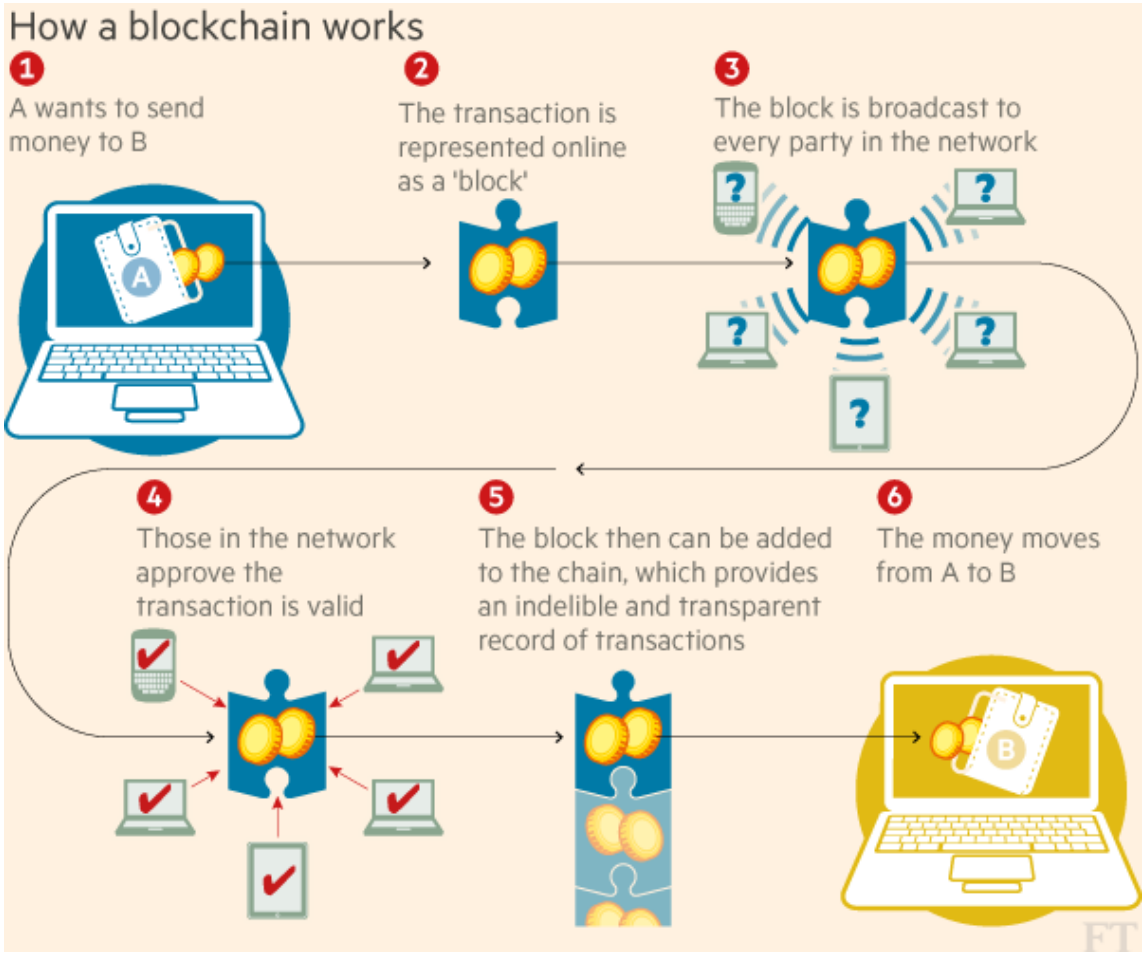
Blockchain

Les transactions effectuées sont regroupées par blocs. Chaque bloc est validé par des noeuds du réseau appelés les “mineurs”, selon des techniques qui dépendent du type de blockchain.

Dans la blockchain du bitcoin : “Proof-of-Work”, preuve de travail, consiste en la résolution de problèmes algorithmiques.

Une fois le bloc validé, il est horodaté et ajouté à la chaîne de blocs. La transaction est alors visible pour le récepteur ainsi que l’ensemble du réseau.

Exemple de paiement par blockchain (image Financial Times)



Applications de la blockchain

Monnaie électronique (bitcoin, Ethereum, etc)

Contrats intelligents (échanges de biens ou services)

Moyens de réduire les couts de transactions

Moyens d'améliorer les systèmes prédictifs des assurances

Cadastre virtuel

Utilisation à l'intérieur des entreprises (blockchain privée)

...

Point négatif : le minage est très énergivore : en 2014 la consommation de la blockchain = consommation de l'Irlande (3GW). Les nouveau protocoles sont moins énergivores

Résumé

- Les fonctions de hachage sont omniprésentes
- Elles doivent être rapides et sûres
- La sécurité de beaucoup de protocoles repose sur la sécurité des fonctions de hachage
- Elles sont une brique importante pour assurer l'authentification