Introduction à la cryptographie

CRI

CENTRE DES RESSOURCES INFORMATIQUES

-- Cyril zarak Duval, root CRI/ACU 2020

Présentation

Quelques généralités

Pourquoi écouter cette présentation ?

- La crypto est partout
  - HTTPS, SSH, LUKS, GPG, ...
- Affaire avec à un moment ou un autre
  - JWT, git, connexion SSH, dm-crypt, PBKDF2, ...
- Vie privée

Qu’est-ce que la cryptographie ?

- ➔ Discipline de la cryptologie
- ➔ Protection des messages
- ➔ Pas de dissimulation des messages (stéganographie)

Qu’est-ce que la cryptographie ?

- ➔ ~Codage
- ➔ Hachage
- ➔ Chiffrement symétrique
- ➔ Chiffrement asymétrique

Qu'est-ce que la cryptographie ?

- ~~Codage~~
- Hachage
- Chiffrement symétrique
- Chiffrement asymétrique

- *Signature numérique*
- *Echange de clés*
- *Chiffrement par bloc, par flot, …*
- *Certificats, identités, …*

Fonctions de hachage

Commençons simple

Fonctions de hachage

➔ Calcul de l'empreinte numérique d'une donnée *(hash)*

Fonctions de hachage

- ➔ Calcul de l'empreinte numérique d'une donnée *(hash)*
- ➔ Entrée taille variable, sortie taille fixe

Fonctions de hachage

- ➔ Calcul de l'empreinte numérique d'une donnée *(hash)*
- ➔ Entrée taille variable, sortie taille fixe
- ➔ Déterministe

Fonctions de hachage

```
$ printf "fonction de hachage" | sha256sum
2db100af5d9d4e41b37bdec521f22ff06536e068e2a365285ac5f9b0e8542a14  -
$ printf "Fonction de hachage" | sha256sum
8b59f93d7714b00c370d386703eecc4aa5da829e248190ed27fe217bb440c17a  -
$ cat /boot/vmlinuz-4.19.0-16-amd64 | sha256sum
d758470d7d1b4148309533e73de20ad2276fa861ce4dabaf0dae360f782fa1fa  -
```

Fonctions de hachage cryptographique

Ajoutons des contraintes

Fonctions de hachage cryptographique

➔ Le hash doit être unique

Fonctions de hachage cryptographique

- ➔ Le hash doit être unique
  - En pratique impossible
  - Ensemble S vers N avec `card(S) > card(N)`

Fonctions de hachage cryptographique

- Le hash doit être unique
  - En pratique impossible
  - Ensemble S vers N avec `card(S) > card(N)`
- Fonction mathématique à sens unique
  - Si `f(x) = y`, impossible* de calculer `x` à partir de `y`

Fonctions de hachage cryptographique

- Le hash doit être unique
  - En pratique impossible
  - Ensemble S vers N avec `card(S) > card(N)`
- Fonction mathématique à sens unique
  - Si `f(x) = y`, impossible* de calculer `x` à partir de `y`
- Impossibilité* de calculer une collision

Fonctions de hachage cryptographique

- Le hash doit être unique
  - En pratique impossible
  - Ensemble S vers N avec `card(S) > card(N)`
- Fonction mathématique à sens unique
  - Si `f(x) = y`, impossible* de calculer `x` à partir de `y`
- Impossibilité* de calculer une collision
- Impossibilité* de calculer un message d'origine

Fonctions de hachage cryptographique

- Le hash doit être unique
  - En pratique impossible
  - Ensemble S vers N avec `card(S) > card(N)`
- Fonction mathématique à sens unique
  - Si `f(x) = y`, impossible* de calculer `x` à partir de `y`
- Impossibilité* de calculer une collision
- Impossibilité* de calculer un message d'origine
- Impossibilité de déterminer le message d'origine

Fonctions de hachage cryptographique

- Le hash doit être unique
  - En pratique impossible
  - Ensemble S vers N avec `card(S) > card(N)`
- Fonction mathématique à sens unique
  - Si `f(x) = y`, impossible* de calculer `x` à partir de `y`
- Impossibilité* de calculer une collision
- Impossibilité* de calculer un message d'origine
- Impossibilité de déterminer le message d'origine
- 2 messages très proches donnent hash très différents

Fonctions de hachage

```
$ printf "fonction de hachage" | sha256sum
2db100af5d9d4e41b37bdec521f22ff06536e068e2a365285ac5f9b0e8542a14  -
$ printf "Fonction de hachage" | sha256sum
8b59f93d7714b00c370d386703eecc4aa5da829e248190ed27fe217bb440c17a  -
$ cat /boot/vmlinuz-4.19.0-16-amd64 | sha256sum
d758470d7d1b4148309533e73de20ad2276fa861ce4dabaf0dae360f782fa1fa  -
```

Fonctions de hachage – utilité

➔ Identification rapide de données diverses

Fonctions de hachage - utilité

- ➔ Identification rapide de données diverses
- ➔ Sécurisation du stockage de mots de passe

Fonctions de hachage - utilité

- ➔ Identification rapide de données diverses
- ➔ Sécurisation du stockage de mots de passe
- ➔ Structures de données *(hash tables)*

Fonctions de hachage - utilité

- ➔ Identification rapide de données diverses
- ➔ Sécurisation du stockage de mots de passe
- ➔ Structures de données *(hash tables)*
- ➔ Vérification de l'intégrité

Fonctions de hachage – utilité

- ➔ Identification rapide de données diverses
- ➔ Sécurisation du stockage de mots de passe
- ➔ Structures de données *(hash tables)*
- ➔ Vérification de l'intégrité
- ➔ ...

Impossible ?

Concept clé en crypto

Impossibilité et difficulté

➔ Aspect théorique != pratique

Impossibilité et difficulté

- ➔ Aspect théorique != pratique
- ➔ En crypto on parle de difficulté

Impossibilité et difficulté

- ➔ Aspect théorique != pratique
- ➔ En crypto on parle de difficulté
- ➔ Problème difficile = qui prend trop de temps à l'échelle humaine avec les moyens actuels

Impossibilité et difficulté

- Aspect théorique != pratique
- En crypto on parle de difficulté
- Problème difficile = qui prend trop de temps à l'échelle humaine avec les moyens actuels
- Attention à :
  - Maths cassées
  - La progression de la puissance de calcul dispo

Exemple de difficulté

Crackons un hash en live

Fonctions de hachage cryptographique en pratique

- MD5 (1991)
- SHA-1 (1995)
- SHA-2 (2002) : sha256, sha512 *(sha384, sha224)*
- Outil générique
- *Pour aller plus loin pour les matheux :*
  - *Merkle–Damgård construction*
  - *Poly1305-AES*

Fonctions de hachage cryptographique en pratique

```
$ printf "fonction de hachage" | md5sum
c80b2ae9edb840bb0a9b0838a1c9eecd  -
$ printf "fonction de hachage" | sha1sum
ea6e33286d070761dbd617e67e429d9c3a82da0a  -
$ printf "fonction de hachage" | sha256sum
2db100af5d9d4e41b37bdec521f22ff06536e068e2a365285ac5f9b0e8542a14  -
$ printf "fonction de hachage" | sha512sum
f360f704d3f547795dff130436f5f2a26c256e53d6f432603e4d307c49c143f2ed158599ec78733
e27e1936780510c432dad863fad4b6b158bc1195861a63116  -
```

Sel cryptographique

"Salt is a way of life" – youtu.be/3KquFZYi6L0

Sel cryptographique

➔ Données en plus avant hachage

Sel cryptographique

- ➔ Données en plus avant hachage
- ➔ Lutte contre les Rainbow Tables
- ➔ Lutte potentiellement contre la brute-force

Sel cryptographique

```
$ printf "soleil" | sha1sum
45c8586a626ddabd233951066138d0efa7f4eb9d  -
$ printf "soleilCECIESTUNSELSTATICUNPEUBASIQUE" | sha1sum
49306899374 49fc4ba0bf2578e3bbb130d39341d  -
```

Sel cryptographique

- Données en plus avant hachage
- Lutte contre les Rainbow Tables
- Lutte potentiellement contre la brute-force
- Sel statique ou dynamique
  - Si dynamique, stocké à côté

Attention !

➔ Fonction de hachage != hash de mot de passe

Attention !

- Fonction de hachage != hash de mot de passe
- Fonction de dérivation de clés
  - Utilise des fonctions de hachage
- PBKDF2, Argon2, Bcrypt, ...

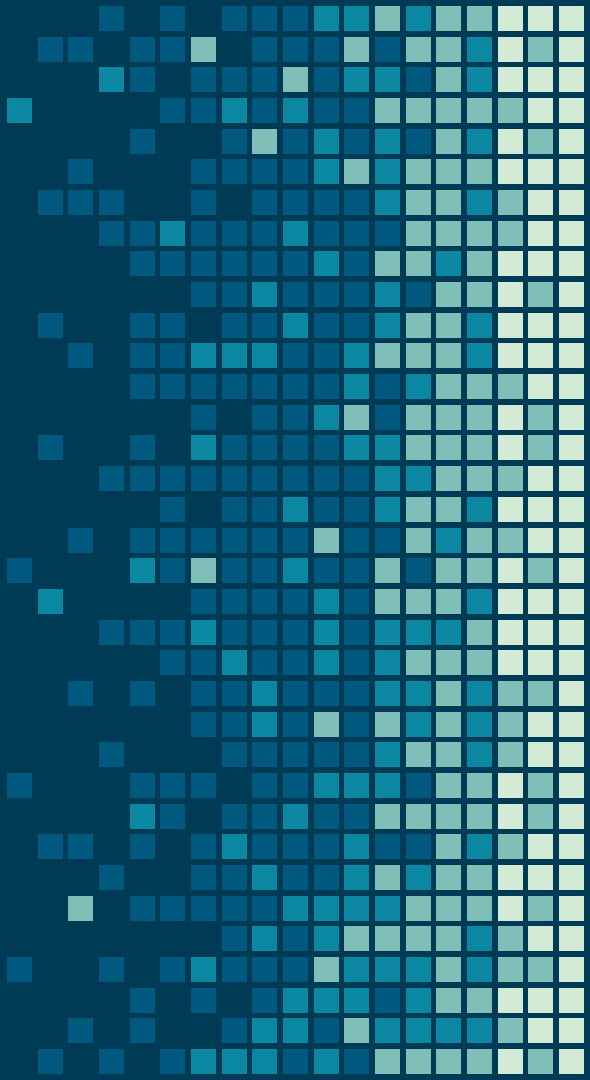
Attention !

- Fonction de hachage != hash de mot de passe
- Fonction de dérivation de clés
  - Utilise des fonctions de hachage
- PBKDF2, Argon2, Bcrypt, …
- Ne **jamais** faire sa tambouille perso en crypto

Fonctions de hachage

Des questions ?

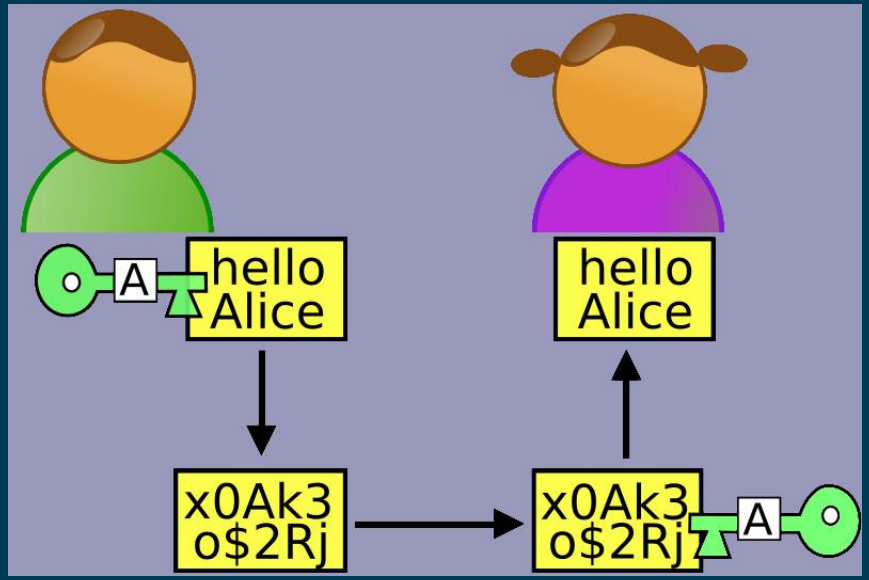
Cryptographie symétrique



Chiffrement symétrique

Crypto moderne

Pourquoi “symétrique” ?



Pourquoi “symétrique” ?

Types de chiffrement symétrique

- ➔ Chiffrement par blocs
- ➔ Chiffrement par flot

Chiffrement par blocs

Segmentons la donnée

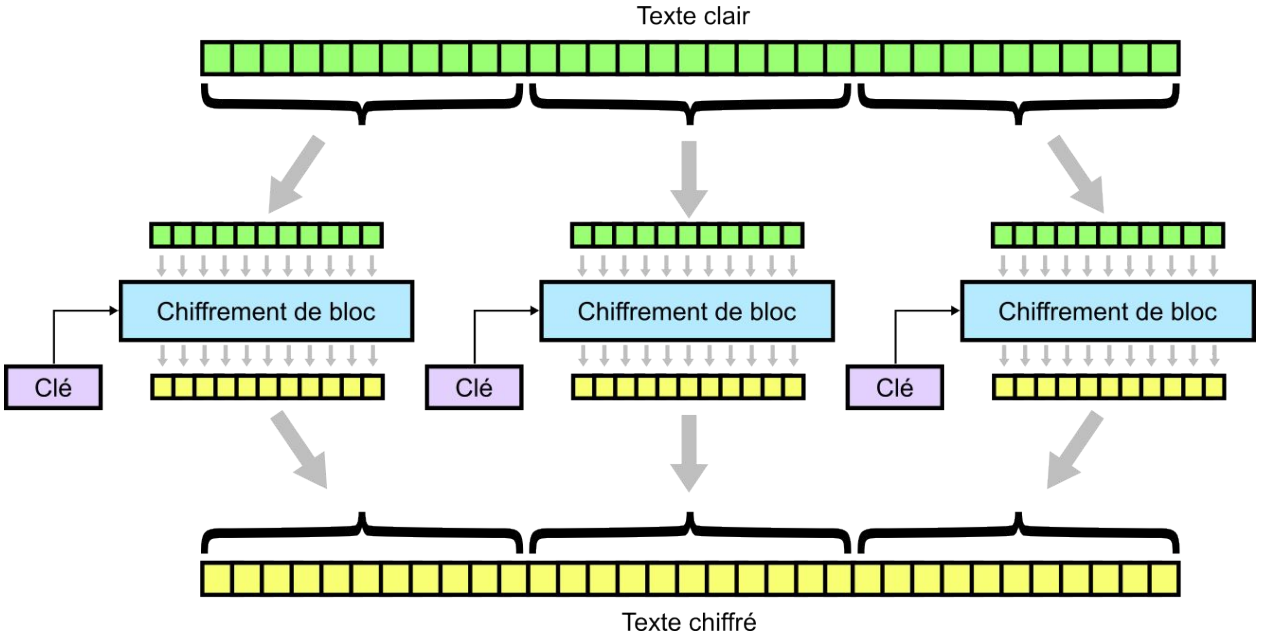
Chiffrement par blocs

- ➔ Segmentations de la données à chiffrer en blocs
- ➔ Taille du bloc qui dépend de l'algorithme

Chiffrement par blocs

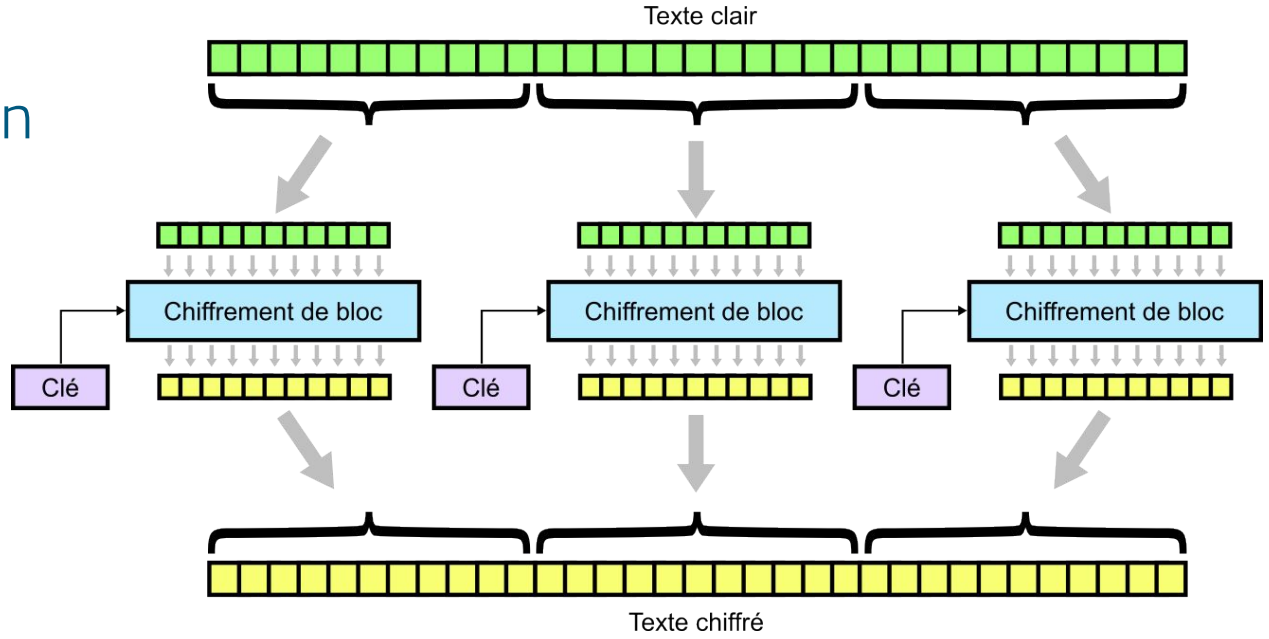
- ➔ Segmentations de la données à chiffrer en blocs
- ➔ Taille du bloc qui dépend de l'algorithme
- ➔ Application de l'algorithme successivement sur les blocs
- ➔ Utilisation d'un "mode d'opération"

Chiffrement par blocs – mode d'opération



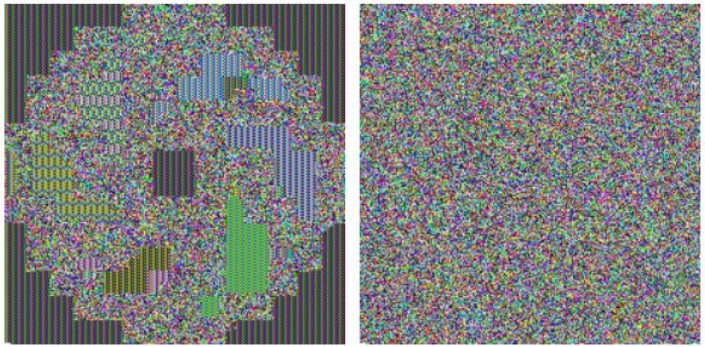
Chiffrement par blocs – mode d'opération

Mode d'opération = **ECB** *Electronic Codebook*



Chiffrement par blocs – mode d'opérations

Mode d'opération = **ECB** *Electronic Codebook*



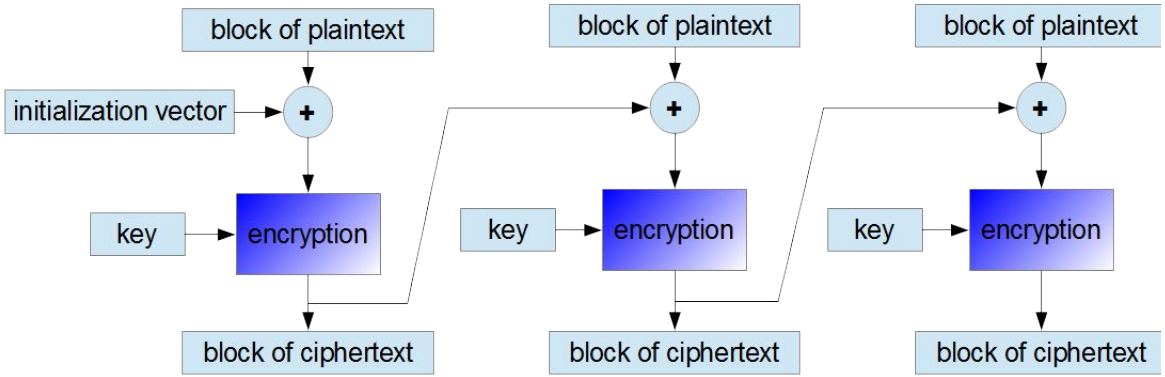
À gauche : ECB
À droite : CBC

Chiffrement par blocs – mode d'opérations

➔ Problème d'ECB : pas de relations entre les blocs

Chiffrement par blocs – mode d'opérations

- ➔ Problème d'ECB : pas de relations entre les blocs
- ➔ Exemple avec relation + Vecteur d'initialisation : CBC



Chiffrement par blocs – IV

➔ Données aléatoires* de la taille d'un bloc *(Le plus souvent)*

Chiffrement par blocs – IV

- Données aléatoires* de la taille d'un bloc *(Le plus souvent)*
- Public
- Fourni avec le message chiffré pour le déchiffrement

Chiffrement par blocs – IV

- Données aléatoires* de la taille d'un bloc *(Le plus souvent)*
- Public
- Fourni avec le message chiffré pour le déchiffrement
- Nonce

Chiffrement par blocs – notations

- ➔ Algorithme de chiffrement par bloc *(e.g. : AES)*
- ➔ Mode d'opération *(e.g. : CBC)*

Chiffrement par blocs – notations

- Algorithme de chiffrement par bloc *(e.g. : AES)*
- Mode d'opération *(e.g. : CBC)*
- Padding (si applicable) *(e.g. : PKCS7)*

Chiffrement par blocs – clé de chiffrement

- ➔ Clé de la taille d'un bloc
- ➔ Si on veut un mot de passe ?

Chiffrement par blocs – clé de chiffrement

- ➔ Clé de la taille d'un bloc
- ➔ Si on veut un mot de passe ?
- ➔ Fonction de hachage

Chiffrement symétrique – exemple

```
$ alias gpg="gpg --pinentry-mode=loopback"  # Use CLI provided password
$ cat message
Ceci est un message secret
$ gpg --passphrase toto -c message
$ cat message.gpg
▒       zL▒F▒Om▒▒R▒P    Uh▒▒▒▒1X9▒▒π$n▒▒{<▒pч‘íI▒▒
▒▒▒▒\*Z▒◦▒pY▒[▒▒¶▒^▒▒n¬▒G▒R▒▒▒▒f`%

$ rm message
$ gpg --passphrase toto -d message.gpg
gpg: AES256 encrypted data
gpg: encrypted with 1 passphrase
Ceci est un message secret
$ gpg --passphrase tata -d message.gpg
gpg: AES256 encrypted data
gpg: encrypted with 1 passphrase
gpg: decryption failed: Bad session key
```

Chiffrement symétrique par bloc

Des questions ?

Aléatoire et pseudo-aléatoire

Transformons du déterminisme en non-déterminisme

Aléatoire et pseudo-aléatoire

➔ Ordinateur intrinsèquement déterministe

Aléatoire et pseudo-aléatoire

- ➔ Ordinateur intrinsèquement déterministe
- ➔ Aléatoire cryptographique

Aléatoire et pseudo-aléatoire

- ➔ Ordinateur intrinsèquement déterministe
- ➔ Aléatoire cryptographique
- ➔ Pseudo-aléatoire

Générateur de pseudo-aléatoire

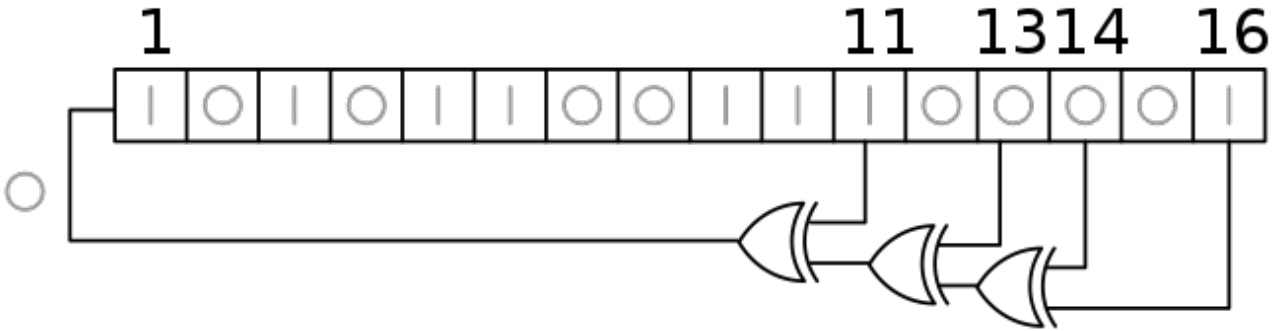
➔ PRNG *(PseudoRandom Number Generator)*

Générateur de pseudo-aléatoire

- PRNG *(PseudoRandom Number Generator)*
- Suite mathématiques déterministe
- Dépend d'une seed

Générateur de pseudo-aléatoire

- PRNG *(PseudoRandom Number Generator)*
- Suite mathématiques déterministe
- Dépend d'une seed



Générateur de pseudo-aléatoire - entropie

- ➔ Entropie de Shannon
- ➔ “Quantité d’information”

Générateur de pseudo-aléatoire - entropie

- ➔ Entropie de Shannon
- ➔ “Quantité d’information”
- ➔ ACABACABACABACAB … -> H(x)= 1.5

Générateur de pseudo-aléatoire - entropie

- ➔ Entropie de Shannon
- ➔ “Quantité d’information”
- ➔ `ACABACABACABACAB … -> H(x)= 1.5`
- ➔ `miHdnDFbgMIuqFkP … -> H(x)= 4`

Générateur de pseudo-aléatoire - entropie

- Entropie de Shannon
- “Quantité d’information”
- ACABACABACABACAB … -> H(x)= 1.5
- miHdnDFbgMIuqFkP … -> H(x)= 4
- “Nombre de questions à poser en moyenne pour connaitre un symbole”

Générateur de pseudo-aléatoire - 2 approches pour la crypto

➔ CSPRNG *(cryptographically secure pseudorandom number generator)*

Générateur de pseudo-aléatoire - 2 approches pour la crypto

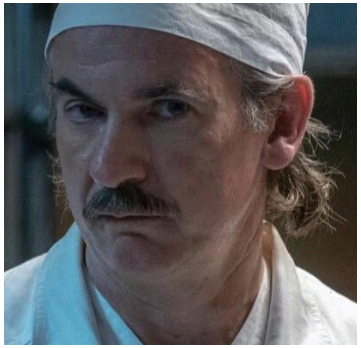
- ➔ CSPRNG *(cryptographically secure pseudorandom number generator)*
- ➔ PRNG avec ajout d'entropie
  - ◆ Evénements imprédictibles

Générateur de pseudo-aléatoire - 2 approches pour la crypto

- ➔ CSPRNG *(cryptographically secure pseudorandom number generator)*
- ➔ PRNG avec ajout d'entropie
  - ◆ Evénements imprédictibles
  - ◆ TRNG *(true random number generator)*

Générateur de pseudo-aléatoire - 2 approches pour la crypto

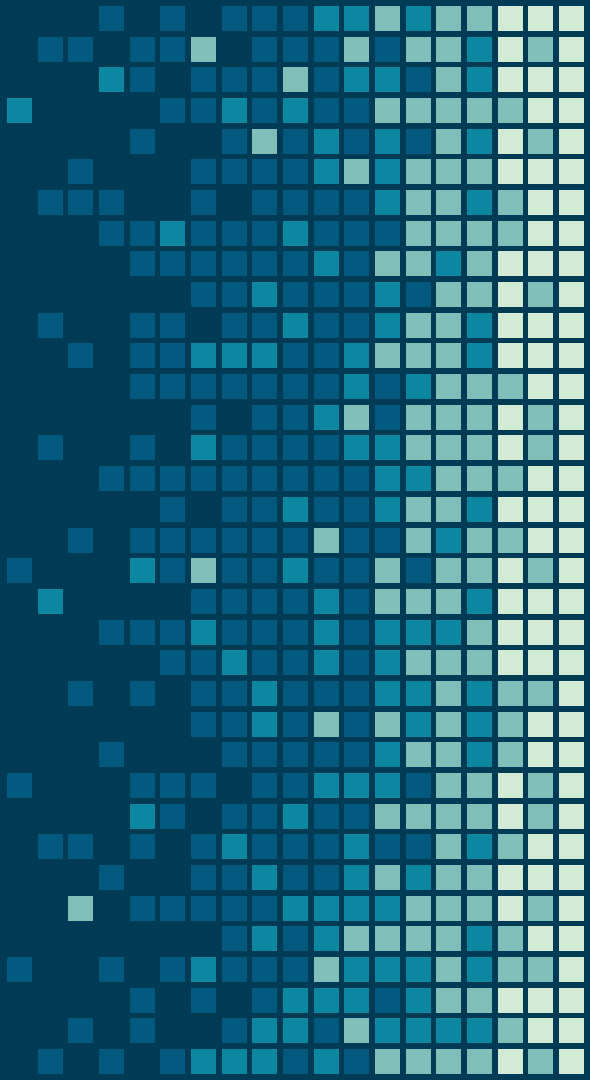
- ➔ CSPRNG *(cryptographically secure pseudorandom number generator)*
- ➔ PRNG avec ajout d'entropie
  - ◆ Evénements imprédictibles



Lavarand @Cloudflare
10% internet traffic

Nuclear decay

Extraction d'entropie de désintégrations nucléaires



Générateur de pseudo-aléatoire - Exemple concret : /dev/random

- Interface de linux avec un CSPRNG
  - PRNG + entropie

Générateur de pseudo-aléatoire - Exemple concret : /dev/random

- Interface de linux avec un CSPRNG
  - PRNG + entropie
- Sources d'entropie :
  - Latences du disque
  - Mouvements souris + clavier
  - Cycles du CPU
  - ...

Aléatoire et cryptographie - Quoi retenir

- ➔ Notion d'entropie
- ➔ Aléatoire et pseudo-aléatoire
- ➔ Vrai aléatoire vient de sources externes
- ➔ PRNG, CSPRNG

Chiffrement symétrique en pratique

- **DES** *(1977)*
- **3DES** *(1999)*
- **Blowfish** *(1993)*
  - blocs 64 bits
  - clé 32-448
- **AES** *(2000)*
  - blocs de 128 bits
  - clé 128/192/256

Aléatoire et entropie

Des questions ?

Chiffrement symétrique par flot

Chiffrement (presque) parfait

Chiffrement par flot

- ➔ Pas de blocs avec obligation de taille
- ➔ Pas de padding

Chiffrement par flot

- ➔ Pas de blocs avec obligation de taille
- ➔ Pas de padding
- ➔ Grosso modo un PRNG pour masque jetable

Chiffrement par flot - chiffre de Vernam

- ➔ Chiffrement impossible à casser
- ➔ Pas de vecteur d'attaque pour la cryptanalyse

Chiffrement par flot - chiffre de Vernam

- ➔ Chiffrement impossible à casser
- ➔ Pas de vecteur d'attaque pour la cryptanalyse
- ➔ Clé de la taille de la donnée à chiffrer

Chiffrement par flot - chiffre de Vernam

- ➔ Chiffrement impossible à casser
- ➔ Pas de vecteur d'attaque pour la cryptanalyse
- ➔ Clé de la taille de la donnée à chiffrer
- ➔ XOR entre donnée et clé

Opération XOR

A	B	$C = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

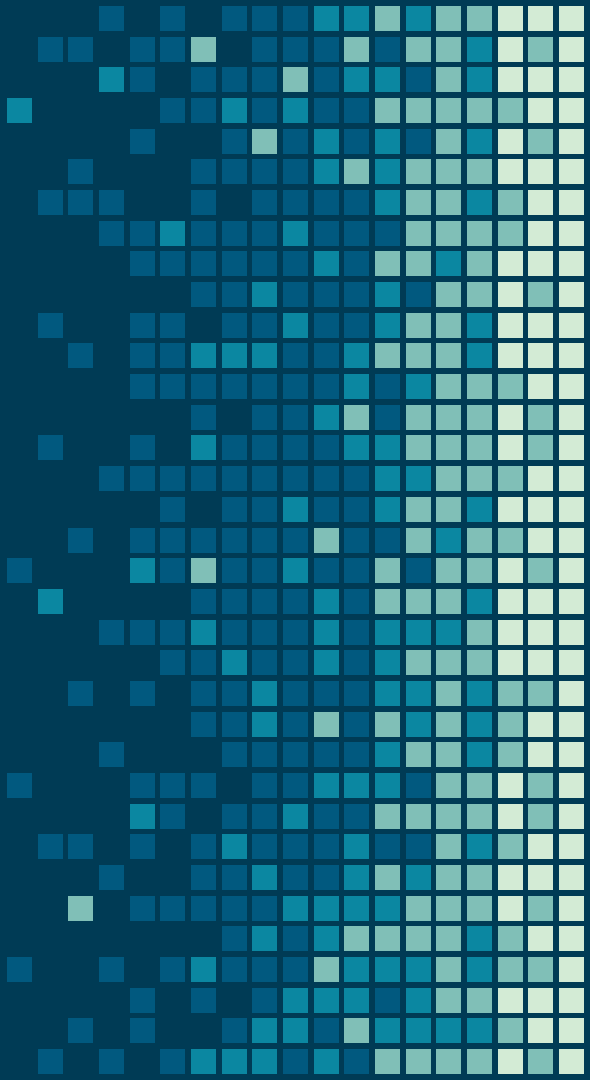
Chiffrement par flot – en pratique

- ➔ RCA (1987)
- ➔ E0 (fin 1990) (bluetooth)
- ➔ Chacha20 (2008)

Chiffrement symétrique par flot

Des questions ?

Cryptographie asymétrique



Chiffrement asymétrique

Généralités

Chiffrement asymétrique - Généralités

➔ Communication sécurisés sur canaux non sécurisés

Chiffrement asymétrique - Généralités

- ➔ Communication sécurisés sur canaux non sécurisés
- ➔ Authentification
- ➔ Signature numérique

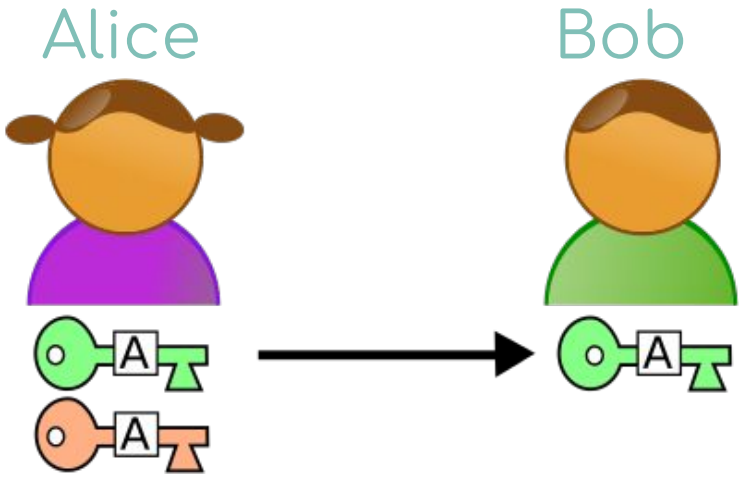
Clé publique, clé privée

- ➔ Base de la crypto asymétrique
- ➔ Clé privée et publique liées

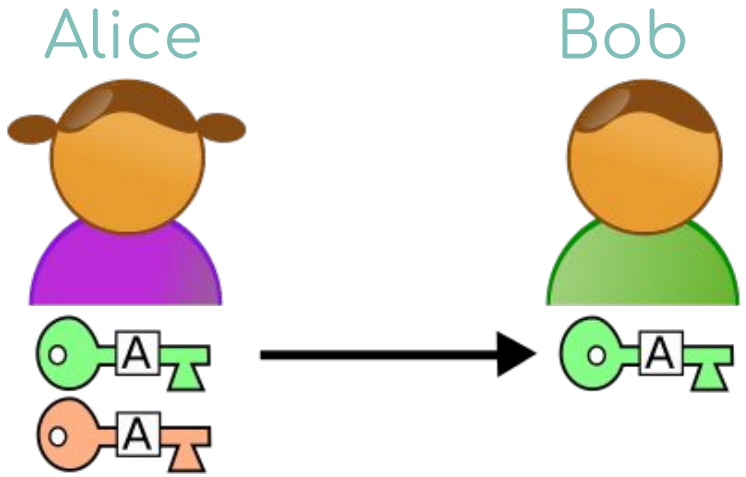
Clé publique, clé privée

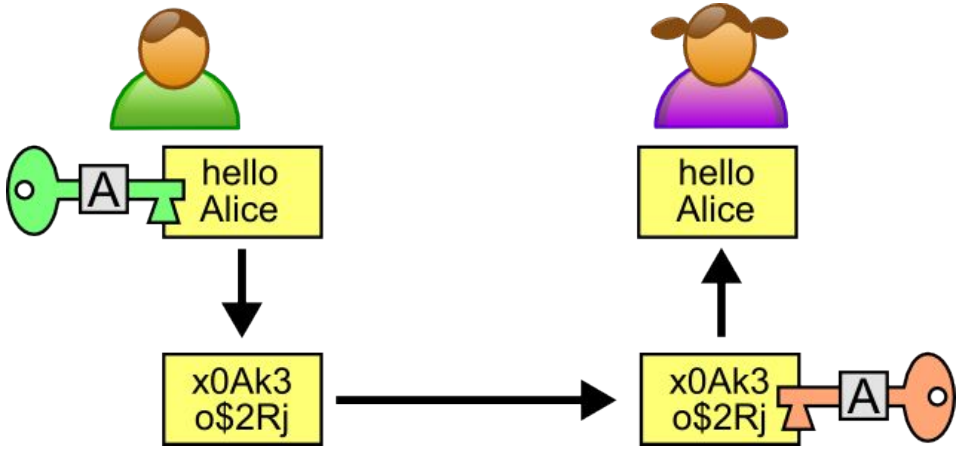
- Base de la crypto asymétrique
- Clé privée et publique liées
- Clé publique dérivable de la clé privée
  - Inverse évidemment difficile *au sens cryptographique*

Clé publique, Clé privée



Clé publique, Clé privée





Mise en pratique

2 exemples volontairement simplifiés

Exemple : authentification pour git*

- ➔ Valable pour git ACU, github, gitlab, …
- ➔ On donne sa clé publique au site

Exemple : authentification pour git*

- Valable pour git ACU, github, gitlab, …
- On donne sa clé publique au site
  - Rattachée à un compte

Exemple : authentification pour git*

- Valable pour git ACU, github, gitlab, …
- On donne sa clé publique au site
  - Rattachée à un compte
- À la connexion : fourni une preuve de son identitée grâce à sa clé privée

Exemple : authentification pour git*

- Valable pour git ACU, github, gitlab, …
- On donne sa clé publique au site
  - Rattachée à un compte
- À la connexion : fourni une preuve de son identitée grâce à sa clé privée
- Le serveur peut vérifier le processus d'auth avec la clé publique

Exemple : HTTPS

➔ Le site diffuse sa clé publique

Exemple : HTTPS

- ➔ Le site diffuse sa clé publique
- ➔ Le client l'utilise pour chiffrer un message

Exemple : HTTPS

- ➔ Le site diffuse sa clé publique
- ➔ Le client l'utilise pour chiffrer un message
- ➔ Le message chiffré n'est déchiffrable que par la privée du site

Exemple : HTTPS

- ➔ Le site diffuse sa clé publique
- ➔ Le client l'utilise pour chiffrer un message
- ➔ Le message chiffré n'est déchiffrable que par la privée du site
- ➔ Le site peut recevoir les données du client en toute sécurité

Exemple : HTTPS

- ➔ Le site diffuse sa clé publique
- ➔ Le client l'utilise pour chiffrer un message
- ➔ Le message chiffré n'est déchiffrable que par la privée du site
- ➔ Le site peut recevoir les données du client en toute sécurité … vraiment ?

Exemple : HTTPS

➔ Comment le site envoie de la donnée ?

Exemple : HTTPS

➔ ~~Comment le site envoie de la donnée ?~~

Exemple : HTTPS

- ~~Comment le site envoie de la donnée ?~~
- Crypto asymétrique beaucoup plus lente
  - Performante sur petites données

Exemple : HTTPS

- ~~Comment le site envoie de la donnée ?~~
- Crypto asymétrique beaucoup plus lente
  - Performante sur petites données
- Asymétrie pour l'échange de clés symétriques

Exemple : HTTPS

- ~~Comment le site envoie de la donnée ?~~
- Crypto asymétrique beaucoup plus lente
  - Performante sur petites données
- Asymétrie pour l'échange de clés symétriques
  - Protocole d'échange de clés *(Diffie-Hellman)*

Exemple : HTTPS

- ~~Comment le site envoie de la donnée ?~~
- Crypto asymétrique beaucoup plus lente
  - Performante sur petites données
- Asymétrie pour l'échange de clés symétriques
  - Protocole d'échange de clés *(Diffie-Hellman)*
- Nécessité de s'assurer de l'identité du destinataire

Chiffrement asymétrique en pratique

- RSA (1977)
- Clé de 2048/4096 bits

```
$ time openssl genrsa 2048 > /dev/urandom
Generating RSA private key, 2048 bit long modulus (2 primes)
openssl genrsa 2048 > /dev/urandom  0.04s user 0.01s system 97% cpu 0.046 total
$ time openssl genrsa 4096 > /dev/urandom
Generating RSA private key, 4096 bit long modulus (2 primes)
openssl genrsa 4096 > /dev/urandom  0.98s user 0.01s system 99% cpu 0.996 total
$ time openssl genrsa 8192 > /dev/urandom
Generating RSA private key, 8192 bit long modulus (2 primes)
openssl genrsa 8192 > /dev/urandom  9.36s user 0.01s system 99% cpu 9.442 total
```

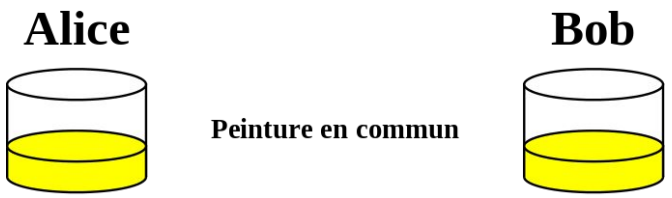
Clé publique/privée

Des questions ?

Échange de clés de Diffie-Hellman

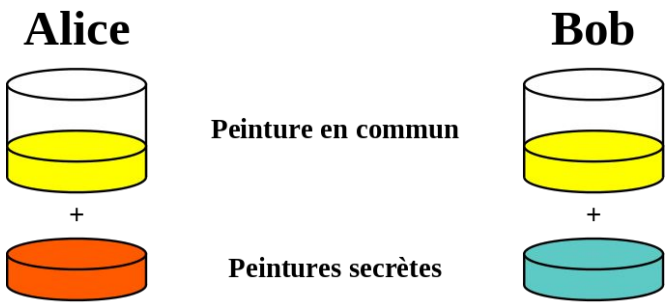
Mélangeons de la peinture

Échange de clés de Diffie-Hellman



➔ Peinture jaune : donnée publique

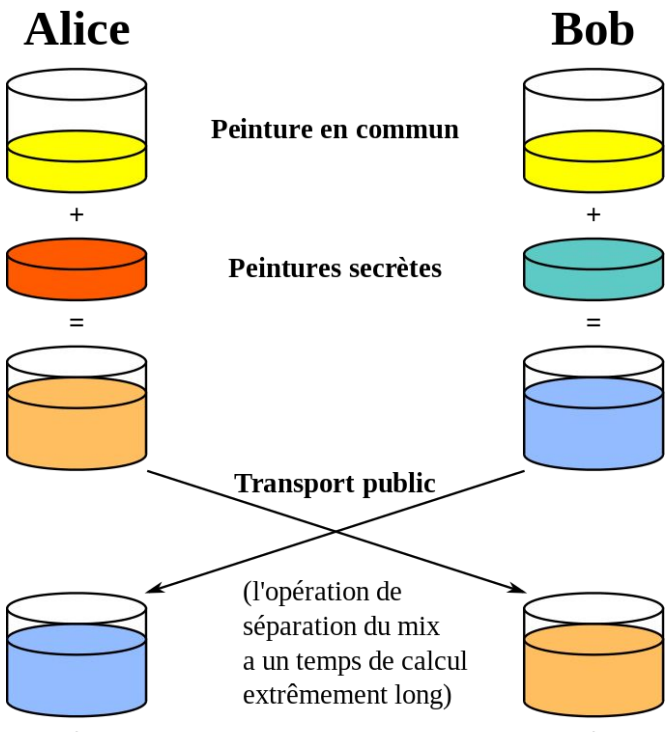
Échange de clés de Diffie-Hellman



- ➔ Peinture jaune : donnée publique
- ➔ Peinture rouge+bleu : données privées

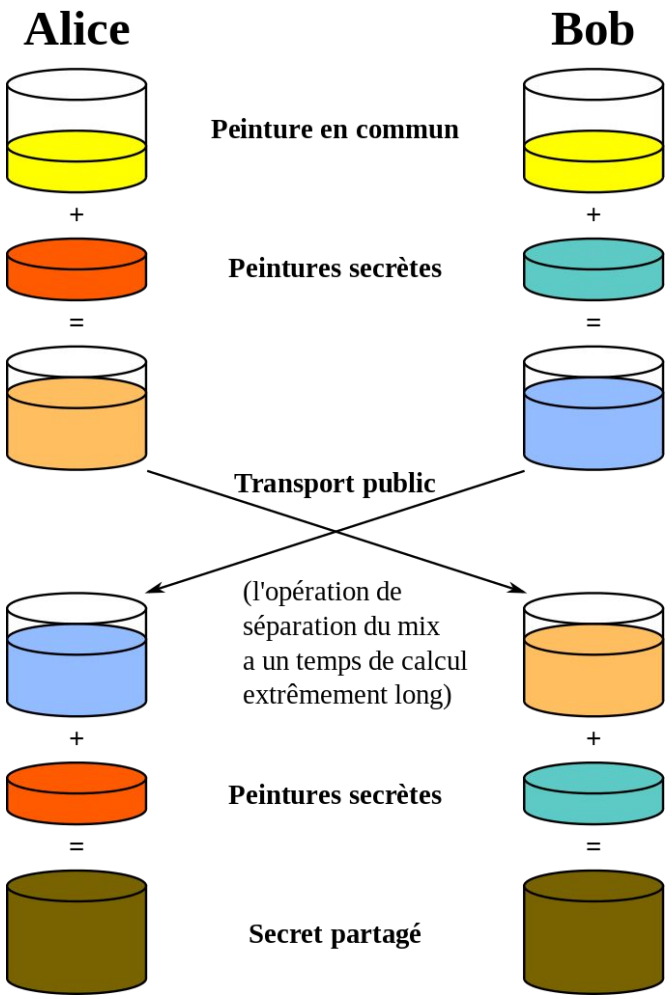
Échange de clés de Diffie-Hellman

- Peinture jaune : donnée publique
- Peinture rouge+bleu : données privées
- Peintures orange+cyan : résultat non réversible



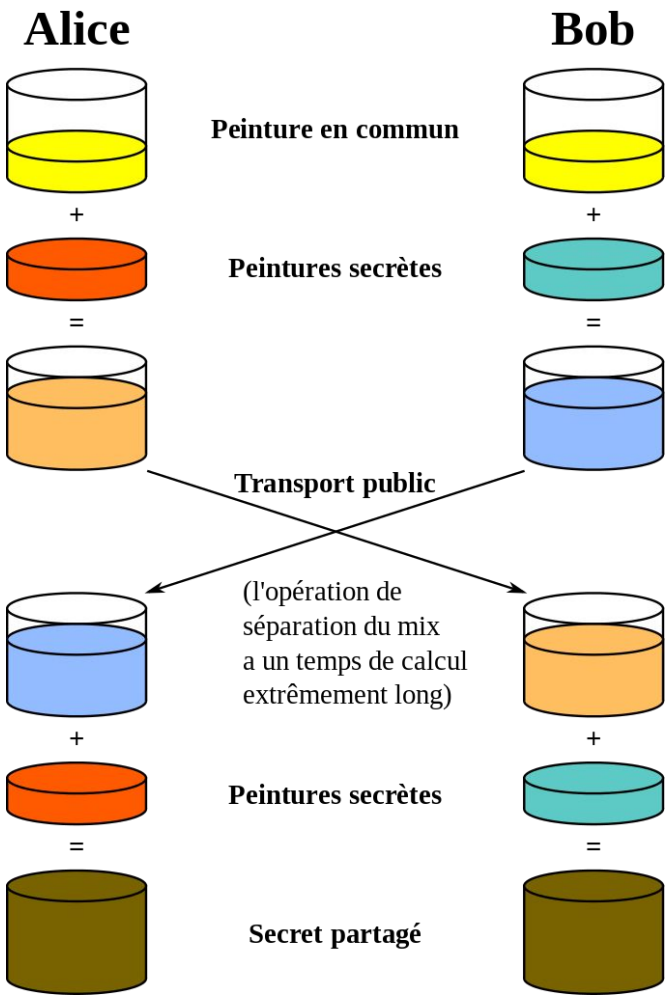
Échange de clés de Diffie-Hellman

- Peinture jaune : donnée publique
- Peinture rouge+bleu : données privées
- Peintures orange+cyan : résultat non réversible
- Ajout de la donnée privée : secret identique



Échange de clés de Diffie-Hellman

- Problème du logarithme discret



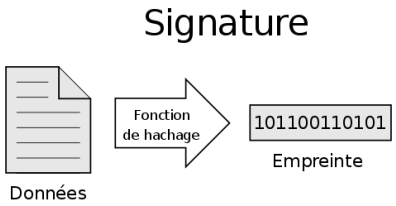
Signature numérique (DSA)

S’assurer de communiquer avec la bonne personne

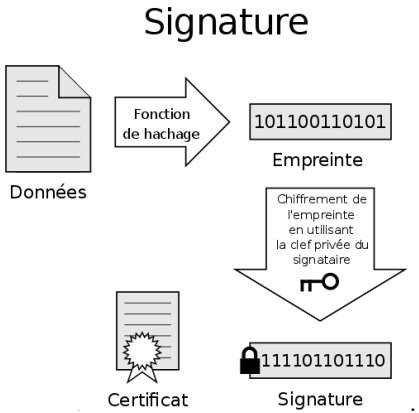
Signature numérique – postulats de base

- ➔ Le signataire possède une identité publique
- ➔ Cette identité est connue du destinataire

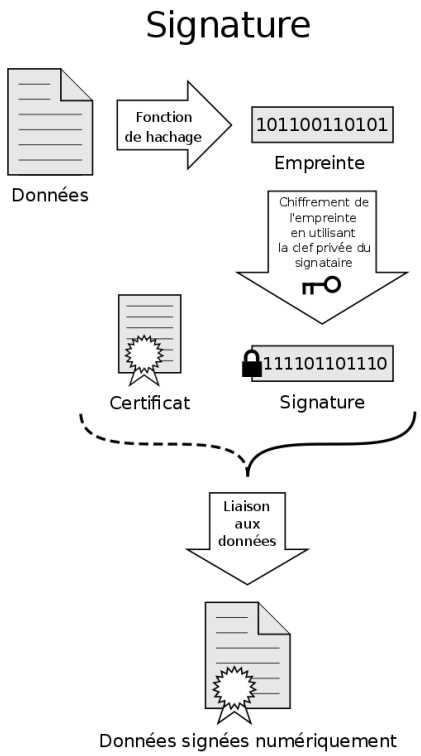
Signature numérique – principe



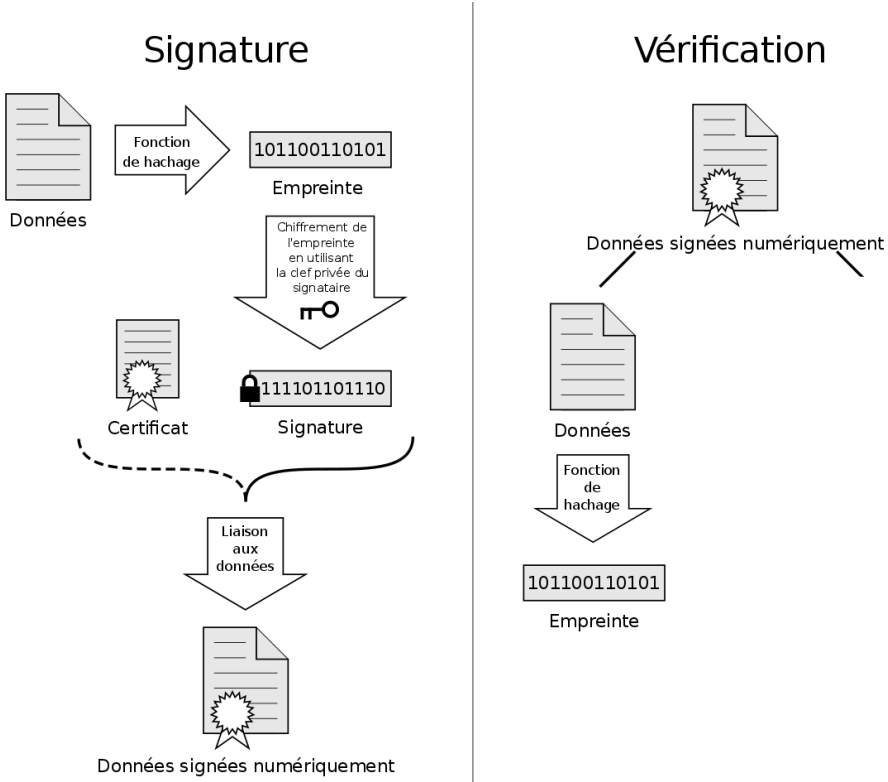
Signature numérique – principe



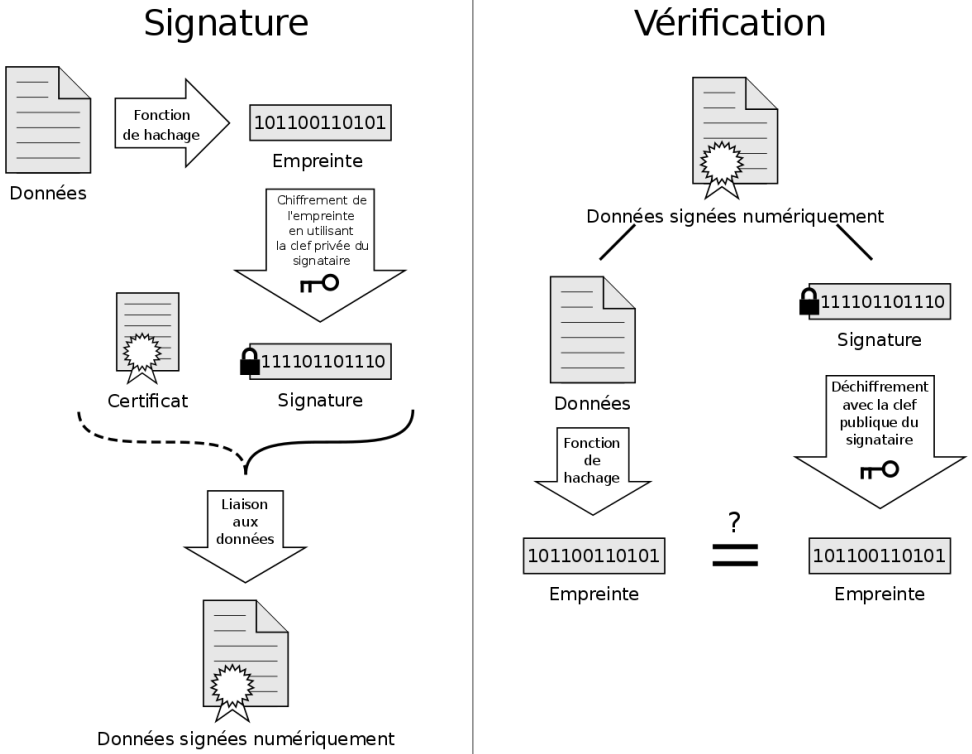
Signature numérique – principe



Signature numérique – principe



Signature numérique – principe



Echange de clés et signature

Des questions ?

Certificats X.509

Identité numérique publique

Certificats X.509

➔ Spécification d'un certificat numérique

Certificats X.509

- Spécification d'un certificat numérique
- Lier une clé publique à une identité
  - Distinguished Name (DN)
  - Alternative Name (AN)

Certificats X.509

- Spécification d'un certificat numérique
- Lier une clé publique à une identité
  - Distinguished Name (DN)
  - Alternative Name (AN)
- Dates de validité, infos crypto, ...

Certificats X.509

- Spécification d'un certificat numérique
- Lier une clé publique à une identité
  - Distinguished Name (DN)
  - Alternative Name (AN)
- Dates de validité, infos crypto, ...
- Signature du certificat par une Autorité de Certification

Certificat X.509 - Exemple

```
$ openssl x509 -in /etc/letsencrypt/live/cyrilduval.fr/chain.pem -text -noout
Certificate:
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
        Validity
            Not Before: Nov 18 21:14:43 2019 GMT
            Not After : Feb 16 21:14:43 2020 GMT
        Subject: CN = cyrilduval.fr
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:b4:09:19:d4:0e:3c:82:3b:0f:ae:66:b7:c9:3d:
                    ....
                    3d:0a:2e:f8:02:f6:b8:3f:49:6c:51:f4:ac:48:90:
                    b3:6a:1b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:cyrilduval.fr, DNS:www.cyrilduval.fr
    Signature Algorithm: sha256WithRSAEncryption
         71:e3:26:c5:51:db:30:55:07:d4:ac:7a:8b:05:67:a7:81:ba:
         ....
         78:04:19:da:b7:c1:25:d8:12:ab:62:bf:0e:3d:0a:b9:55:82:
         d6:22:fd:6e
```

Autorité de certification

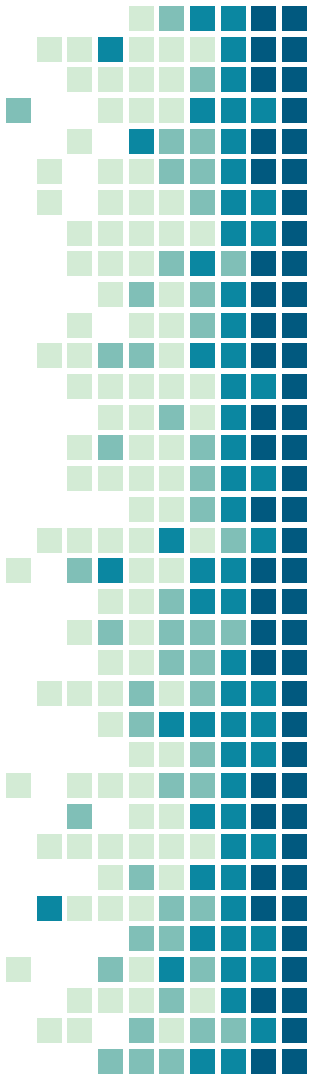
Ne pas faire confiance à n'importe qui

Autorité de certification

➔ Tiers de confiance

Autorité de certification

- ➔ Tiers de confiance
- ➔ Chaine les certificats



Autorité de certification

- ➔ Tiers de confiance
- ➔ Chaine les certificats
- ➔ Sommet de la chaine : certificat racine

Autorité de certification

- Tiers de confiance
- Chaine les certificats
- Sommet de la chaine : certificat racine
  - Présents sur les machines des clients
  - `/etc/ssl/certs` pour linux

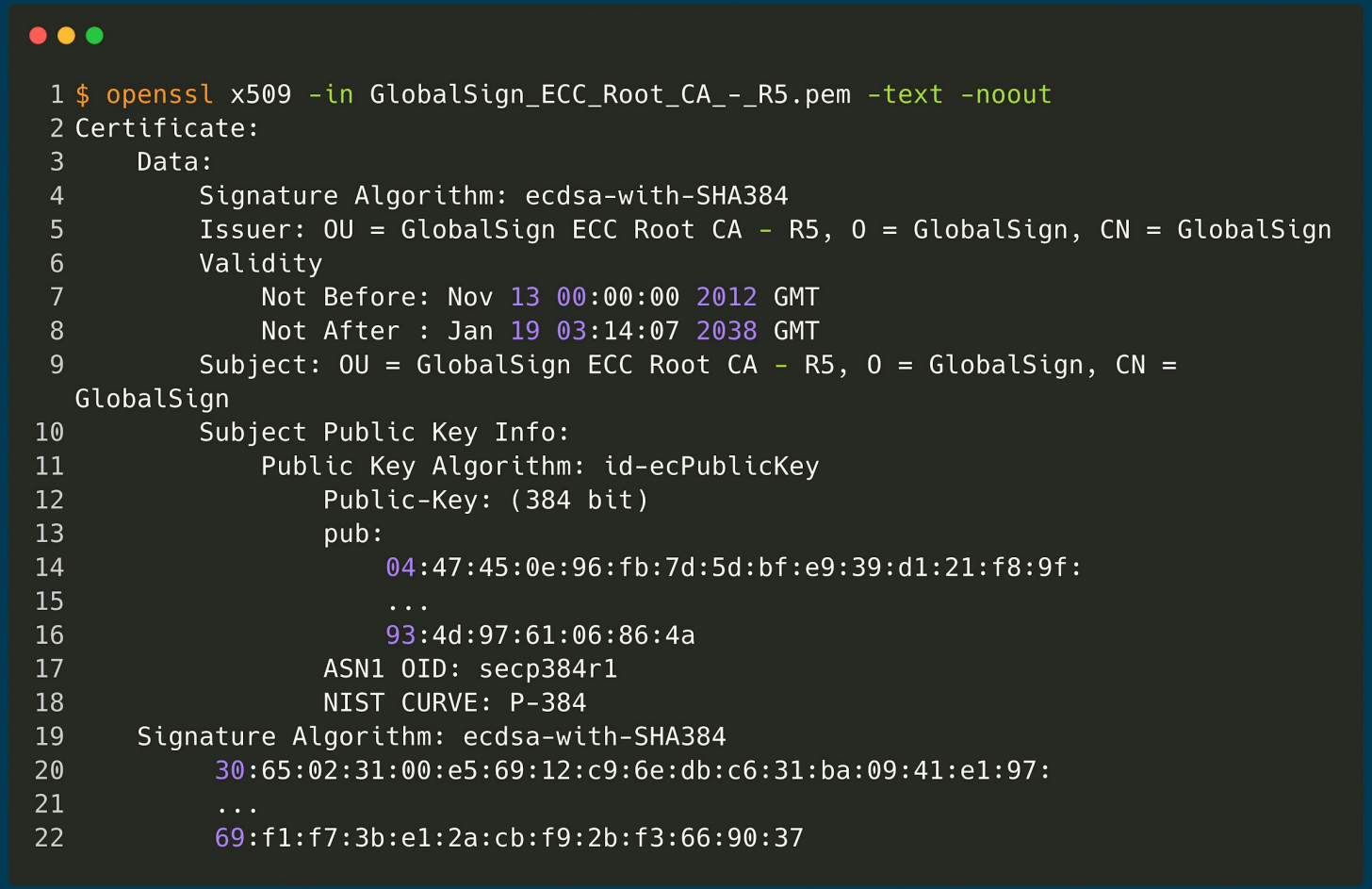
Autorité de certification - exemple

- ➔ Regardons par exemple le certificat fourni par wikipédia en HTTPS
- ➔ On utilise `openssl connect` pour vérifier

```
$ openssl s_client -connect fr.wikipedia.org:443
CONNECTED(00000003)
depth=2 OU = GlobalSign ECC Root CA - R5, O = GlobalSign, CN = GlobalSign
verify return:1
depth=1 C = BE, O = GlobalSign nv-sa, CN = GlobalSign ECC OV SSL CA 2018
verify return:1
depth=0 C = US, ST = California, L = San Francisco, O = "Wikimedia Foundation,
 Inc.", CN = *.wikipedia.org
verify return:1
---
Certificate chain
 0 s:C = US, ST = California, L = San Francisco, O = "Wikimedia Foundation,
 Inc.", CN = *.wikipedia.org
   i:C = BE, O = GlobalSign nv-sa, CN = GlobalSign ECC OV SSL CA 2018
 1 s:C = BE, O = GlobalSign nv-sa, CN = GlobalSign ECC OV SSL CA 2018
   i:OU = GlobalSign ECC Root CA - R5, O = GlobalSign, CN = GlobalSign
 2 s:OU = GlobalSign ECC Root CA - R5, O = GlobalSign, CN = GlobalSign
   i:OU = GlobalSign Root CA - R3, O = GlobalSign, CN = GlobalSign
---
Server certificate
...
subject=C = US, ST = California, L = San Francisco, O = "Wikimedia Foundation,
 Inc.", CN = *.wikipedia.org

issuer=C = BE, O = GlobalSign nv-sa, CN = GlobalSign ECC OV SSL CA 2018
```

```
$ openssl s_client -connect fr.wikipedia.org:443
...
 2 s:OU = GlobalSign ECC Root CA - R5, O = GlobalSign, CN = GlobalSign
   i:OU = GlobalSign Root CA - R3, O = GlobalSign, CN = GlobalSign
...
$ ls /etc/ssl/certs/GlobalSign*
GlobalSign_ECC_Root_CA_-_R4.pem
GlobalSign_ECC_Root_CA_-_R5.pem
GlobalSign_Root_CA.pem
GlobalSign_Root_CA_-_R2.pem
GlobalSign_Root_CA_-_R3.pem
GlobalSign_Root_CA_-_R6.pem
```



```
$ openssl x509 -in GlobalSign_ECC_Root_CA_-_R5.pem -text -noout
Certificate:
    Data:
        Signature Algorithm: ecdsa-with-SHA384
        Issuer: OU = GlobalSign ECC Root CA - R5, O = GlobalSign, CN = GlobalSign
        Validity
            Not Before: Nov 13 00:00:00 2012 GMT
            Not After : Jan 19 03:14:07 2038 GMT
        Subject: OU = GlobalSign ECC Root CA - R5, O = GlobalSign, CN =
GlobalSign
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    04:47:45:0e:96:fb:7d:5d:bf:e9:39:d1:21:f8:9f:
                    ...
                    93:4d:97:61:06:86:4a
                ASN1 OID: secp384r1
                NIST CURVE: P-384
    Signature Algorithm: ecdsa-with-SHA384
         30:65:02:31:00:e5:69:12:c9:6e:db:c6:31:ba:09:41:e1:97:
         ...
         69:f1:f7:3b:e1:2a:cb:f9:2b:f3:66:90:37
```

Cryptographie et courbes elliptiques

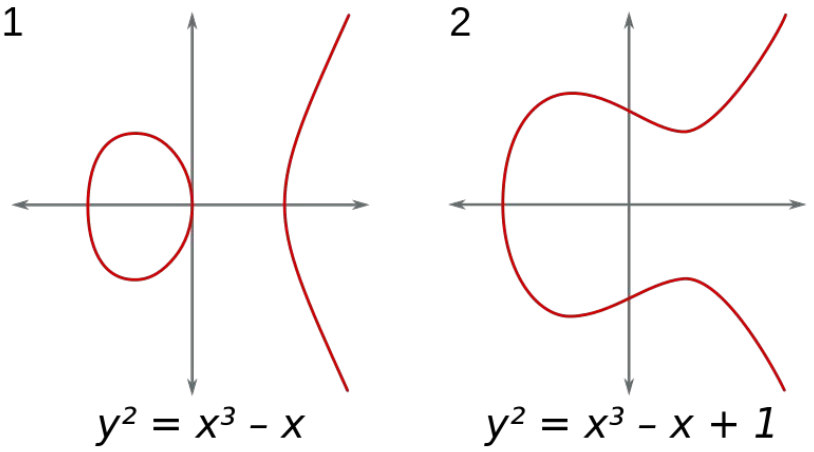
Car la crypto était simple jusqu'à présent

Courbes elliptiques

➔ Problème du logarithme discret

Courbes elliptiques

- Problème du logarithme discret
- Autre approche avec des problèmes sur courbes elliptiques

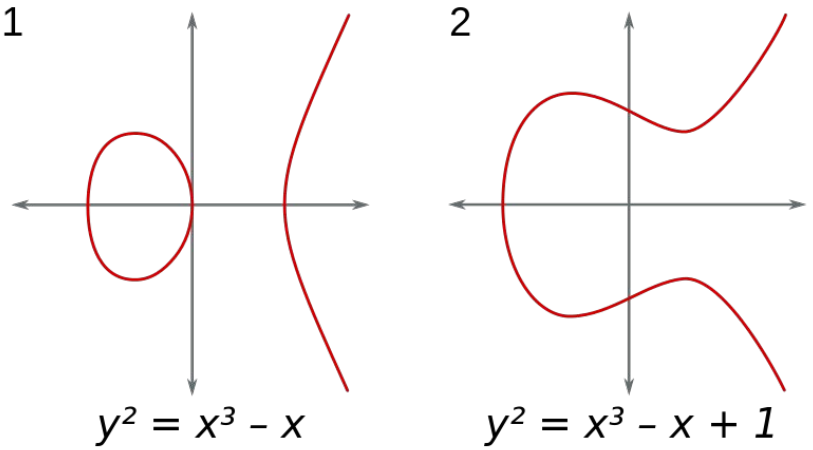


Courbes elliptiques

- Problème du logarithme discret
- Autre approche avec des problèmes sur courbes elliptiques
- Courbes précises

  e.g. : Curve25519

  $y^2 = x^3+486662x^2+x$



Courbes elliptiques

- ➔ Pas directement transposable
  - ◆ Diffie-Hellman *(DH)* ->

    Elliptic Curves Diffie-Hellman *(ECDH)*

Courbes elliptiques

- ➔ Pas directement transposable
  - ◆ Diffie-Hellman *(DH)* ->

    Elliptic Curves Diffie-Hellman *(ECDH)*
- ➔ Implémentation souvent plus difficile

Courbes elliptiques

- ➔ Pas directement transposable
  - ◆ Diffie-Hellman *(DH)* ->

    Elliptic Curves Diffie-Hellman *(ECDH)*
- ➔ Implémentation souvent plus difficile
- ➔ Utilisation (utilisateur) identique

Courbes elliptiques

- Pas directement transposable
  - Diffie-Hellman *(DH)* ->

    Elliptic Curves Diffie-Hellman *(ECDH)*
- Implémentation souvent plus difficile
- Utilisation (utilisateur) identique
- Clés beaucoup plus petite pour sécurité égale

Courbes elliptiques

Le plus important

Soyez attentifs

Le plus important

On dit :

-> Chiffrer

-> Déchiffrer

-> Chiffrement

-> Décrypter

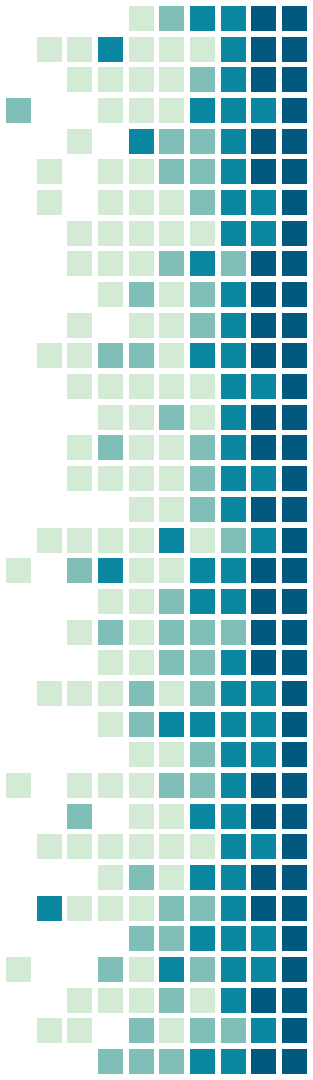
On ne dit pas :

-> Crypter

-> Cryptage

-> Encrypter

-> Chiffrage



Merci !

Des questions ?

Merci !

Des questions ?

Disponible sur
zarak.fr/crypto/introduction