Cryptographie

Laurent Signac –

Dans ce document, nous explicitons les notions de cryptographie symétrique, cryptographie asymétrique et fonctions de hachage.

1 Chiffrement par flot/bloc et modes opératoires

Un algorithme donné peut chiffrer des blocs de données (de taille fixe) ou un flux de données. Dans le cas du chiffrement de blocs de données, le mode opératoire décrit de quelles manière et à quel endroit on applique la fonction de chiffrement.

Sans entrer dans les détails, voici deux modes opératoires différents (il en existe d'autres)

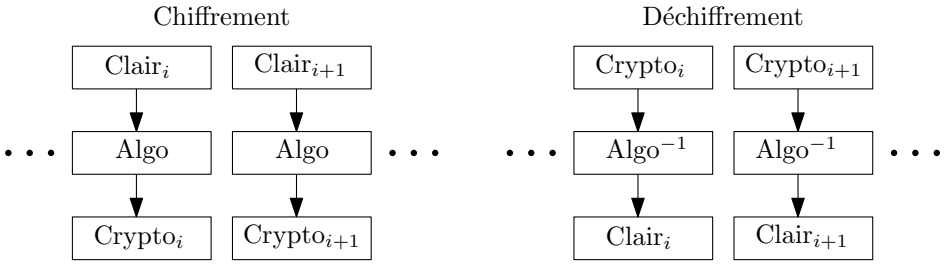


Figure 1: Chiffrement par bloc, mode Ecb (Electronic CodeBook

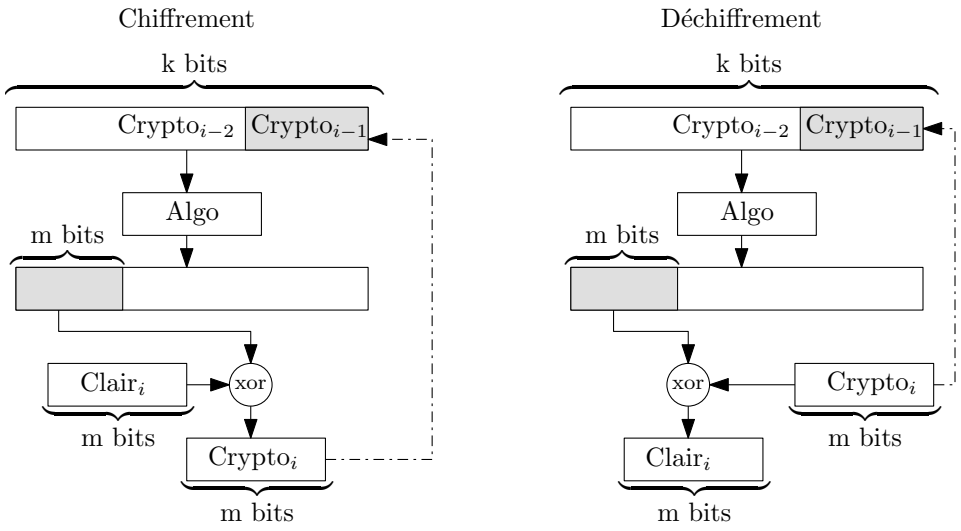


Figure 2: Chiffrement par bloc, mode Cfb (Cipher FeedBack)

2 Cryptographie Symétrique

Le principe des algorithmes symétriques est que la même clé est utilisée pour le chiffrement et le déchiffrement.

2.1 Masque jetable

Nous citons ici cette méthode comme exemple de chiffrement par flot, et en raison de sa fiabilité, puisqu'un tel chiffre ne peut pas être cassé en l'absence d'information supplémentaire (preuve donnée par Shannon en 1949). La méthode est très simple et consiste à appliquer par exemple un ou-exclusif, bit par bit, entre le message en clair et la clé. Cet algorithme est involutif et sa sécurité repose sur la longueur et la nature de la clé qui doit être aléatoire et avoir la même longueur que le texte en clair. Chaque clé ne peut servir qu'une fois (sinon la propriété sur la longueur de la clé n'est plus vérifiée), d'où le nom du principe de chiffrement. Une telle méthode, bien que parfaitement sûre pose des problèmes pratiques (contraintes fortes sur la clé).

2.2 Data Encryption Standard

La norme DES est basée sur l'algorithme Lucifer, adopté comme standard en 1976 par le NIST (*National Institute of Standards and Technology*).

Il est composé de substitutions, de permulations, de ou exclusifs… Les opérations sont réalisées plusieurs fois de suite, on les appelle des *rondes*. L'algorithme chiffre des blocs de 56 bits, et la clé utilisée fait 64 bits (56 bits utiles + une somme de contrôle).

Une attaque classique contre ce code consiste à essayer d'appliquer toutes les clés possibles sur le texte chiffré. Or, il y a plus de 10^{16} clés possibles. L'exploration de l'espace des clés était impossible il y a quelques années… Mais il ne l'est plus maintenant.

2.3 Advanced Encryption Standard

AES a été adopté comme standard en 2001 en remplacement de DES et Triple DES. Il est basé sur l'algorithme Rijndael (inventé par Joan Daemen et Vincent Rijmen). La longueur de clé peut varier de 128 à 256 bits, et le texte est chiffré par blocs de 128 à 256 bits.

AES est basé sur les opérations suivantes (Fips 197[1]), qui sont réalisées plusieurs fois (de 10 à 14 rondes pour les implémentations standard) :

- ou exclusif bit à bit
- substitutions d'octets
- permutations d'octets
- multiplications par des polynômes

Actuellement (2019), utiliser une implantation correcte d'AES avec une clé de 256 bits est un moyen extrêmement sûr de chiffrer des données.

2.4 Problème des algorithmes à clé secrète

Les algorithmes à clé secrète actuellement utilisés sont très sûrs et très rapides. Cependant, il font apparaître un problème récurrent : celui de l'échange des clés. Pour communiquer, les deux entités doivent impérativement connaître une donnée commune : la clé. Si l'échange de cette clé se fait en clair, c'est la sécurité des communications futures qui n'est plus assurée.

C'est aux trois chercheurs américains Whitfield Diffie, Martin Hellman et Ralph Merkle qu'a été attribuée la découverte d'un protocole d'échange des clés (1976). Le système est basé sur l'arithmétique modulo n et sur une fonction à sens unique de type $x \mapsto Y^x \pmod n$.

Voici comment Alice et Bob peuvent convenir d'une clé commune (un nombre) sans chiffrer leurs communications (fig. 3).

1. Alice et Bob s'entendent publiquement sur les valeurs de Y et n (n très grand premier et $Y < n$).
2. Chacun d'eux choisit de son côté un nombre que nous noterons x_A et x_B.
3. Chacun d'eux calcule l'image par la fonction à sens unique ($Y^x \pmod n$) du nombre choisi. Ils disposent maintenant des nombres y_A et y_B.
4. Alice et Bob échangent publiquement les nombres y_A et y_B.
5. Alice calcule $z_A = y_B^{x_A} \pmod n$ et Bob calcule $z_B = y_A^{x_B} \pmod n$
6. Les nombres z_A et z_B sont identiques et constituent la clé qu'utiliseront Alice et Bob.

On peut vérifier facilement que z_A et z_B sont égaux. En revanche, il est difficile de croire qu'une personne ayant écouté leur conversation ne peut pas trouver la clé. Une telle personne connaîtrait en effet : Y, n, y_A et y_B. Si elle essaie de calculer la clé de la même façon qu'Alice ou Bob, il lui faut x_A ou x_B, ce qu'elle n'a pas. Elle peut aussi essayer de refaire le calcul à l'envers, et connaissant Y, n et y_A, calculer le nombre x_A tel que $y_A = Y^{x_A} \pmod n$ (calcul de logarithme discret modulo n). Toute la subtilité de la méthode repose sur le fait que ce calcul ne peut pas être fait efficacement (fonction à sens unique). En choisissant de grandes valeurs pour n, x_A, et x_B le calcul est actuellement impossible en pratique (voir figure 4 pour une analogie intéressante)

[1] https://csrc.nist.gov/publications/detail/fips/197/final

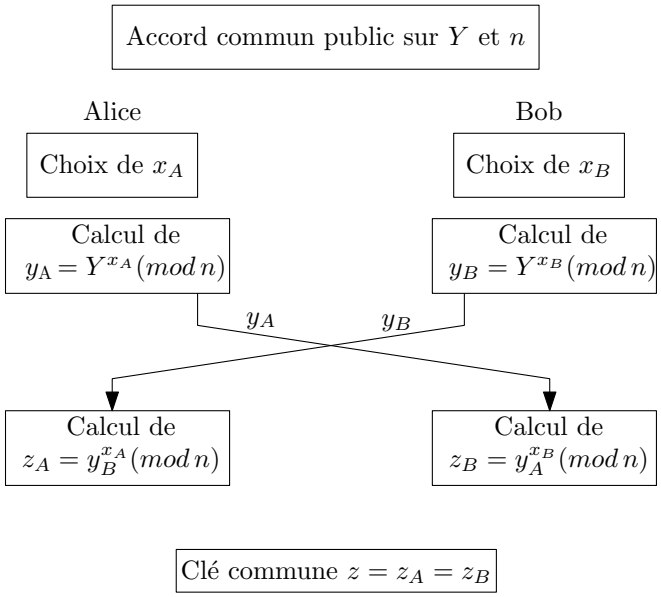


Figure 3: Protocole d'échange de clés de Diffie, Hellman et Merkle

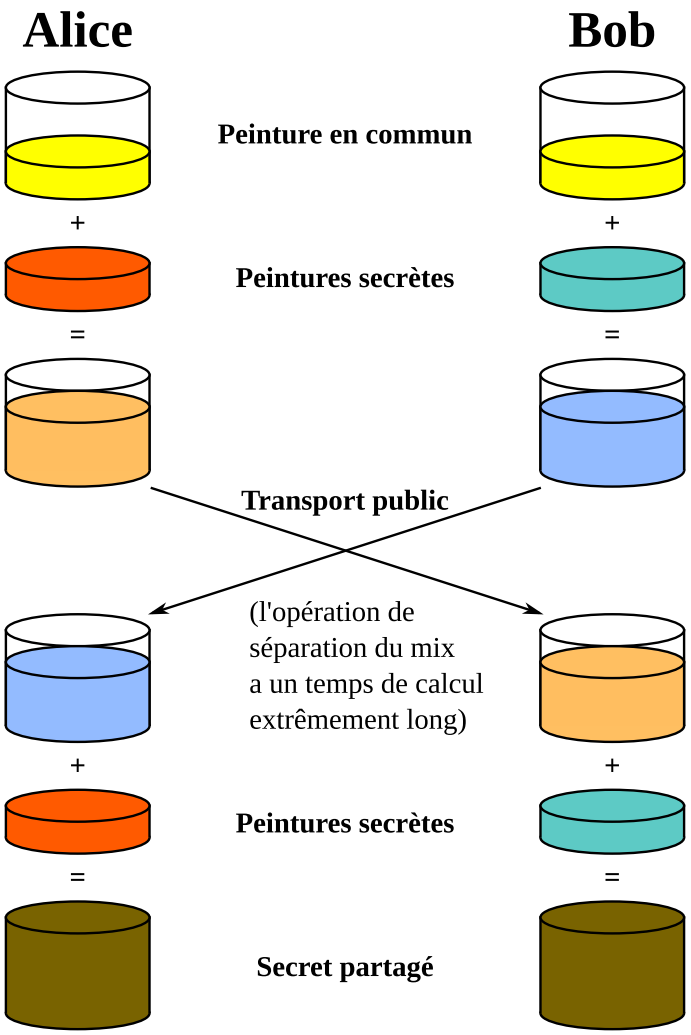


Figure 4: Idée originale : A.J. Han Vinck / Version vectorielle : Flugaal / Traduction : Dereckson [Public domain]

3 Cryptographie asymétrique

L'idée du chiffrement asymétrique a été proposée par Diffie et Hellman en 1976 (bien que probablement, d'autres aient eu l'idée avant, sans la publier). Dans ce type d'algorithme, chaque intervenant possède deux clés. La *clé publique*, utilisée généralement pour le chiffrement, et que tout le monde connaît, et la *clé privée* utilisée pour le déchiffrement, que seul son propriétaire connaît (voir figure 5)

Bien entendu, les deux clés sont liées, mais il doit être très difficile d'obtenir la clé privée à partir de la clé publique.

Si le principe était établi, il restait à trouver un système utilisable en pratique.

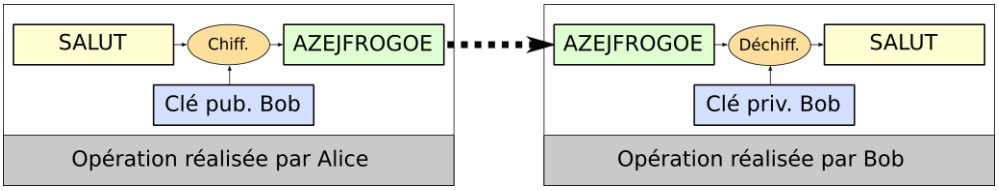


Figure 5: Pour qu'Alice envoie un message à Bob, seul Bob a besoin d'une paire de clés

3.1 RSA

L'algorithme à clé publique le plus utilisé est Rsa (du nom de ses auteurs, Ronald Rivest, Adi Shamir et Leonard Adleman qui l'ont découvert en 1977). Sa mise en œuvre sur des exemples jouets (avec de petits nombres) est très simple :

- choisir deux nombres premiers (normalement très grands) : p et q
- rechercher un nombre e qui soit premier avec $(p-1)$ et $(q-1)$ et donc avec $z = (p-1)(q-1)$.
- calculer d tel que $ed = 1 \mod z$) (possible puisque e est premier avec z)

Les nombres e et $n = pq$ constituent la clé publique, et les nombres d et n la clé privée. Nous admettrons que, connaissant e et n, on ne peut pas retrouver d autrement qu'en retrouvant les entiers p et q (pour calculer z). Or, il n'existe pas de méthodes efficace pour retrouver la décomposition en facteurs premiers de n et le calcul est impraticable si n est très grand (quelques milliers de chiffres binaires).

Pour chiffrer un nombre P (avec $P < n$), on calcule simplement $C = P^e \mod n$, ce qui est facile connaissant e et n, mais très difficile à inverser… sauf dans certains cas particuliers (on dit que la fonction utilisée est une fonction à sens unique et à brèche secrète). En effet, la fonction réciproque est simplement $P = C^d \mod n$, mais son calcul implique de connaître d, qu'on ne sait pas calculer efficacement à partir de e et n…

Malgré les nombreux avantages du système Rsa (plus de problème d'échange de clés secrètes), le système est trop lent pour être utilisé de façon complètement transparente. Il est donc essentiellement utilisé pour deux choses : la signature numérique, et l'échange de clé secrètes pour pouvoir utiliser un chiffrement symétrique (comme Aes, bien plus rapide)

3.2 Certificats - Authentification des clés publiques

Pour recevoir des message chiffrés, le destinataire doit communiquer sa clé publique (peu importe par quel biais). L'expéditeur doit donc disposer d'un moyen d'authentifier la clé publique et de vérifier que c'est bien celle du destinataire et non celle d'une tierce personne (voir par exemple : Attaque de l'homme du milieu[2])

Les *certificats* permettent cette authentification : un tiers de confiance certifie la provenance de la clé privée et *signe* (voir plus loin pour la signature numérique) le certificat.

[2] https://fr.wikipedia.org/wiki/Attaque_de_l%27homme_du_milieu

4 Condensés

4.1 Condensés de messages

Il est souvent nécessaire d'obtenir, à partir d'un texte de longueur n, un texte plus court, appelé condensé de message (ou hachage ou hash), qui a les propriétés suivantes (dans le cas d'un hachage cryptographique) :

- un condensé de message doit être facile (i.e. rapide) à calculer (même si le message est très long) ;
- quel que soit le condensé, on ne peut pas l'utiliser pour retrouver le message original ;
- pour un message donné, il est impossible (comprendre hors de portée d'un calcul actuel et pour les années à venir) de trouver un autre message ayant le même condensé (alors qu'il y en a, par définition, puisque le condensé est plus court) ;
- une modification mineure dans le message modifie grandement le condensé.

Les deux méthodes les plus connues (dont nous ne détaillerons pas le fonctionnement) sont MD5 et SHA-1 qui permettent respectivement d'obtenir des condensés de 128 et 160 bits.

Bien qu'encore très utilisées, elles sont maintenant dépassées et devraient être remplacées par des versions plus modernes comme SHA-256 ou SHA-512 (famille SHA-2).

Les condensés sont utilisés dans différents contextes :

- vérification de l'intégrité d'un fichier (on peut vérifier visuellement le condensé d'un énorme fichier de plusieurs Go)
- hachage des mots de passe (pour contrer un éventuel vol de bases de données d'authentification)
- signature numérique

Voici un exemple de calcul d'un condensé en ligne de commandes :

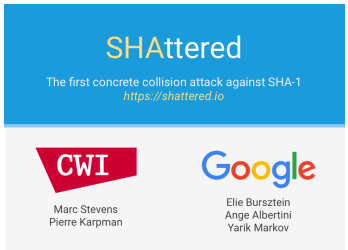
```
$ echo "Utilisez Linux!" | sha256sum
832107cfab70836be4611b1faae024cd773b3e69b77a0091ded717b72666eae1  -

$ echo "utilisez Linux!" | sha256sum
6715f0f776edf18ded91985c286af30e7ee88a7cd7ef0eea3c5846bcbe8afcb2  -
```

4.2 Collisions

Une fonction de hachage est non-injective par construction (l'ensemble d'arrivée est plus petit que l'ensemble de départ). Il existe donc forcément deux entrées ayant le même condensé (il y en a même énormément). La difficulté est d'exhiber une telle collision. Un peu de recul est nécessaire pour comprendre la gravité de la découverte de collisions.

En 2017, une groupe de recherche a produit deux documents PDF différents ayant le même condensé SHA-1 (voir le site shattered.io[3]).



C'est une attaque de collision avec préfixe choisi : il est possible, à partir d'un fichier quelconque (ou presque) de produire deux autres fichiers différents, ayant le même condensé, et semblables au fichier modèle.

C'est plus fort que simplement exhiber une collision. C'est moins fort que produire une version modifiée d'un fichier de départ en gardant le même condensé.

[3] https://shattered.io