# Cryptographie

## Concepts de la sécurité

*2022 / 2023*

François Goichot

- Présentation générale de la cryptographie (fonctions de base, apports et limites)
- Notion de complexité
- Fonctions de hachage
- Chiffrements symétrique (DES, AES) et asymétrique (RSA)

# Table des matières

# 1 Bases mathématiques, 1

- Congruences, $\mathbb{Z}/n\mathbb{Z}$
- $(\mathbb{Z}/2\mathbb{Z}, +)$ : interprétation logique
- Permutations d'un ensemble à $n$ éléments

# 2 Système de chiffrement

## 2.1 Définition

Un *système de chiffrement*, ou *cryptosystème*, ou *chiffrement*, est la donnée de $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ vérifiant :

- $\mathcal{P}$ est un ensemble, ses éléments sont appelés *messages en clair* [plaintext]
- $\mathcal{C}$ est un ensemble, ses éléments sont appelés *messages chiffrés* ou *cryptogrammes* [cyphertext]
- $\mathcal{K}$ est un ensemble, ses éléments sont appelés *clés* [keys]
- $\mathcal{E} = \{E_k\ ,\ k \in \mathcal{K}\ \}$ est une famille de fonctions $E_k : \mathcal{P} \to \mathcal{C}$ ; ses éléments sont appelés *fonctions de chiffrement* [encryption functions]
- $\mathcal{D} = \{D_k\ ,\ k \in \mathcal{K}\}$ est une famille de fonctions $D_k : \mathcal{C} \to \mathcal{P}$ ; ses éléments sont appelés *fonctions de déchiffrement* [decryption functions]
- à chaque clé $e \in \mathcal{K}$ on sait associer une clé $d \in \mathcal{K}$ telle que $D_d(E_e(p)) = p$ pour tout message $p$

Exemple (trop) facile : le *chiffrement de César*

$\mathcal{P} = \mathcal{C} = \mathcal{K} = \{A, B, C \dots Z\}$ qu'on note $\mathcal{A}$

Chaque lettre est identifiée à son rang modulo 26 ($A = 0, B = 1$, etc)

À chaque clé $e \in \mathbb{Z}_{26}$, on associe

- la fonction de chiffrement $E_e : \mathcal{A} \to \mathcal{A},\ x \mapsto x + e$ modulo 26
- la fonction de déchiffrement $D_e : \mathcal{A} \to \mathcal{A},\ x \mapsto x - e$ modulo 26

donc la clé de déchiffrement $d$ qui correspond à la clé de chiffrement $e$ est simplement $d = e$

Généralisation plus intéressante, le *chiffrement de Vigenère* (Blaise de Vigenère, 1523 - 1596) :

$\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathcal{A}^l,\ l$ entier fixé $>1$ et les fonctions de chiffrement et de déchiffrement sont les mêmes que pour César mais composante par composante. « Chiffre indéchiffrable » jusqu'en 1863

### 2.2 Chiffrement par blocs

Définition : un cryptosystème est un chiffrement *par blocs* [block cipher] si son ensemble $\mathcal{P}$ des messages en clair et son ensemble $\mathcal{C}$ des messages chiffrés sont tous deux égaux à l'ensemble $\mathcal{A}^n$ des mots de longueur $n$ sur l'alphabet $\mathcal{A}$, $\mathcal{A}$ et $n$ étant fixés tous les deux

À distinguer du chiffrement *par flot,* ou *en continu*

Théorème : les fonctions de chiffrement d'un chiffrement par blocs sont des permutations

## 3 Chiffrement symétrique (à clé secrète)

### 3.1 Le système DES

C'est un système de chiffrement par blocs, symétrique donc à clef secrète

On prend $\mathcal{P} = \mathcal{C} = \{0,1\}^{64}$ et

$$\mathcal{K} = \left\{ (b_1, b_2, \ldots, b_{64}) \in \{0,1\}^{64} \;\middle|\; \forall j = 0 \ldots 7,\ \sum_{i=1}^{8} b_{8j+i} \equiv 1 \mod 2 \right\}$$

c'est-à-dire les mots binaires de longueur 64 tels que la somme des 8 bits de chacun des 8 octets les composant, soit impaire

Exercice : combien cela fait-il de clés possibles ?

# DES : Data Encryption Standard

Développé par une équipe d'IBM dans les années 1970, adopté comme standard aux États-Unis en 1977, très largement utilisé après

Dès 1981, certains suggèrent que la National Security Agency (NSA) serait capable de décrypter DES

Mais ce n'est qu'en 1998 qu'une équipe de l'Electronic Frontier Foundation y parvient, par force brute

Il reste très résistant à la *cryptanalyse différentielle* : des attaques à *texte clair choisi* c'est-à-dire que l'attaquant a la possibilité de soumettre lui-même des messages (la clé étant fixée, inconnue bien sûr de l'attaquant). Il étudie alors l'effet d'une petite variation du message d'entrée

Le standard est abandonné en 2005. Le système reste utilisé, sous la forme du *triple DES* : trois tours de DES avec deux ou trois clés différentes

# DES, schéma général

Le cœur de DES consiste en 16 « tours » utilisant chacun une clé différente, construite à partir de la clé unique de session

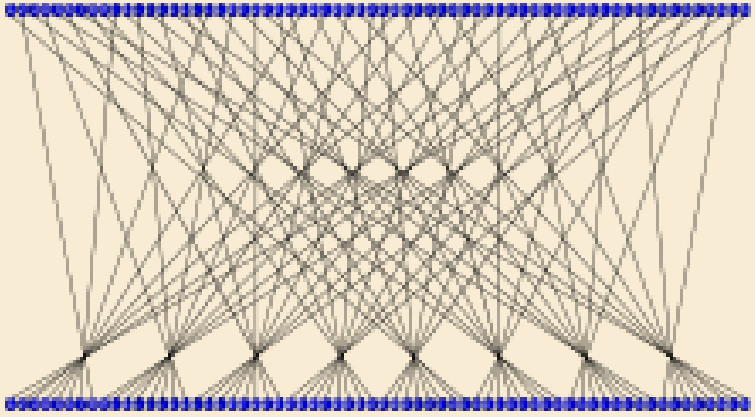
Chaque tour de chiffrement utilise des « S-boîtes » et différentes fonctions (détaillées plus loin), les mêmes à chaque tour

Ce cœur est précédé d'une permutation des 64 bits du bloc de message en clair, et suivi de la permutation inverse

*Le seul élément non public* d'une session de DES est la clé de session

### 3.1.1 La permutation initiale $\sigma_0$

Elle est appliquée à chaque bloc $p$ de message en clair, au tout début du chiffrement.



Et tout à la fin, on applique $\sigma_0^{-1}$

$\sigma_0$ est définie explicitement :

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & \dots & 62 & 63 & 64 \\ 40 & 8 & 48 & 16 & 56 & 24 & 64 & 32 & 39 & 7 & 47 & 15 & 55 & \dots & 17 & 57 & 25 \end{pmatrix}$$

voir Wikipédia (et le TP1) pour la version complète. $\sigma_0$ a deux points fixes, 22 et 43, et elle est d'ordre 6, c'est-à-dire que 6 est le plus petit entier $k \geqslant 1$ tel que $\sigma_0^k = \mathrm{Id}$

# Une subtilité mathématique

Quand on fait agir une permutation $\sigma \in S_n$ sur une liste d'objets $(x_1, x_2, \ldots, x_n)$, il faut poser

$$\sigma\,(x_1, x_2, \ldots, x_n) = (x_{\sigma^{-1}(1)}, x_{\sigma^{-1}(2)}, \ldots, x_{\sigma^{-1}(n)})$$

pour qu'il soit vrai que

$$\sigma(\sigma'(x_1, x_2, \ldots, x_n)) = (\sigma\sigma')(x_1, x_2, \ldots, x_n)$$

Si l'on posait

$$\sigma(x_1, x_2, \ldots, x_n) = (x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)})$$

alors

$$\sigma(\sigma'(x_1, x_2, \ldots, x_n)) = (\sigma'\sigma)(x_1, x_2, \ldots, x_n)$$

ce qui serait bizarre

Les sources sur DES expliquent très rarement ce point

En fait donc, si on a le message en clair $p = p_1 p_2 \dots p_{64} \in \{0,1\}^{64}$ alors

$$\sigma_0(p) = p_{58} p_{50} \dots p_7$$

car

$$\sigma_0^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & \dots & 62 & 63 & 64 \\ 58 & 50 & 42 & 34 & 26 & 18 & 10 & 2 & 60 & 52 & 44 & 36 & 28 & \dots & 23 & 15 & 7 \end{pmatrix}$$

### 3.1.2 Les clés de tours

On part d'une clé $k \in \mathcal{K}$ donc de longueur 64, et on va la « diversifier » en générant 16 clés $K_1, \dots, K_{16}$ chacune de longueur 48

On va utiliser $\varphi_1 : \{0,1\}^{64} \to \{0,1\}^{28} \times \{0,1\}^{28}$, $k \mapsto \varphi_1(k) = (C, D)$, $C$ et $D$ mots binaires qui oublient certains bits de $k$ et permutent les autres (détail p. suivante)

et $\varphi_2 : \{0,1\}^{28} \times \{0,1\}^{28} \to \{0,1\}^{48}, (C, D) \mapsto \varphi_2(C, D)$, qui concatène les deux mots $C$ et $D$, oublie 8 bits du résultat et permute les autres (détail p. suivante)

# les fonctions $\varphi_1$ et $\varphi_2$

$\varphi_1$ est définie par les tableaux :

| | | | | | | |
|---|---|---|---|---|---|---|
| 57 | 49 | 41 | 33 | 25 | 17 | 9 |
| 1 | 58 | 50 | 42 | 34 | 26 | 18 |
| 10 | 2 | 59 | 51 | 43 | 35 | 27 |
| 19 | 11 | 3 | 60 | 52 | 44 | 36 |

et

| | | | | | | |
|---|---|---|---|---|---|---|
| 63 | 55 | 47 | 39 | 31 | 23 | 15 |
| 7 | 62 | 54 | 46 | 38 | 30 | 22 |
| 14 | 6 | 61 | 53 | 45 | 37 | 29 |
| 21 | 13 | 5 | 28 | 20 | 12 | 4 |

donc si $k = k_1 k_2 \dots k_{64}$, $\varphi_1(k) = (C, D)$ avec $C = k_{57} k_{49} \dots k_{36}$ et $D = k_{63} k_{55} \dots k_4$

$\varphi_1$ est aussi notée PC1 [permuted choice 1], et $\varphi_2$ aussi notée PC2 [permuted choice 2]

$\varphi_2$ est définie par concaténation de $(C, D) = ((c_1, \dots, c_{28}), (d_1, \dots, d_{28}))$ en $(b_1, \dots b_{56})$ puis par le tableau :

| | | | | | |
|---|---|---|---|---|---|
| 14 | 17 | 11 | 24 | 1 | 5 |
| 3 | 28 | 15 | 6 | 21 | 10 |
| 23 | 19 | 12 | 4 | 26 | 8 |
| 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 |
| 30 | 40 | 51 | 45 | 33 | 48 |
| 44 | 49 | 39 | 56 | 34 | 53 |
| 46 | 42 | 50 | 36 | 29 | 32 |

donc $\varphi_2(C, D) = (b_{14}, b_{17}, \ldots, b_{32})$

# Construction des clés de tours

Pour $i = 1, \ldots, 16$, on pose $v_i = \begin{cases} 1 \text{ si } i = 1, 2, 9, \text{ ou } 16 \\ 2 \text{ sinon} \end{cases}$

Partant de la clé de session $k$, on pose $(C_0, D_0) = \varphi_1(k)$ puis, pour $i = 1, \ldots, 16$, on note $C_i$ (respectivement, $D_i$) l'image de $C_{i-1}$ (respectivement $D_{i-1}$) par permutation circulaire des bits de $v_i$ positions

Par exemple, si $C_2 = c_1 c_2 \ldots c_{28}$, alors $C_3 = c_3 c_4 \ldots c_{28} c_1 c_2$ puisque $v_3 = 2$

On pose enfin $K_i = \varphi_2(C_i, D_i)$

Les 16 clés de tours sont ainsi construites. Noter que tout part de la clé de session $k$, et qu'on ne peut pas calculer $K_{16}$ sans avoir calculé toutes les clés $K_i$ précédentes

### 3.1.3 Les fonctions de chiffrement $f_{K_i}$

Dans cette partie, on fixe $K$, qui est l'une des clés de tour $K_i$

On va construire une fonction de chiffrement $f_K$, qui opérera sur des *demi*-messages, donc sur $\{0,1\}^{32}$. On aura besoin de trois ingrédients :

- les 8 *S-boîtes* $S_i : \{0,1\}^6 \to \{0,1\}^4$
- l'*expansion* $E : \{0,1\}^{32} \to \{0,1\}^{48}$
- la *permutation de sortie* $P \in S_{32}$

# Les S-boîtes

Chacune des 8 S-boîtes est décrite par un tableau à 4 lignes et 16 colonnes d'entiers entre 0 et 15. Voici par exemple celui pour $S_1$ :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

Pour chaque mot binaire $B = b_1 b_2 \ldots b_6$,

- l'entier $(b_1 b_6)_2$ (qui est entre 0 et 3) donne l'indice de ligne dans ce tableau
- l'entier $(b_2 b_3 b_4 b_5)_2$ (qui est entre 0 et 15) donne l'indice de colonne
- on lit l'entier à l'intersection de la ligne et de la colonne, on le convertit en binaire, en ajoutant des 0 si nécessaire

  on obtient ainsi $S_i(B)$

  Avantage des S-boîtes : c'est non linéaire, donc efficace contre la cryptanalyse différentielle

Inconvénient : c'est opaque

# L'expansion $E$

L'expansion $E : \{0,1\}^{32} \to \{0,1\}^{48}$ est donnée par le tableau

| | | | | | |
|---|---|---|---|---|---|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 | 1 |

donc $E(b_1 b_2 \ldots b_{32}) = b_{32} b_1 b_2 \ldots b_{31} b_{32} b_1$

C'est une permutation circulaire avec répétitions, la moitié des bits apparaît deux fois

# La permutation de sortie $P$

Avec la même convention de notation que pour $\sigma_0$, $P$ est donnée par le tableau

| | | | |
|---|---|---|---|
| 16 | 7 | 20 | 21 |
| 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 |
| 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 |
| 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 |
| 22 | 11 | 4 | 25 |

$P$ n'a pas de point fixe, et est d'ordre 60

# La construction de $f_K$

Pour un mot binaire $R$ de longueur 32,

- on calcule $E(R) \oplus K$, somme bit-à-bit (somme dans $\mathbb{Z}/2\mathbb{Z}$, ou encore `XOR`) de deux mots binaires de longueur 48, donc de longueur 48 aussi
- on découpe ce mot en $B_1 B_2 \ldots B_8$, chaque $B_i$ étant donc de longueur 6
- avec les S-boîtes, pour chaque $i$ on calcule $C_i = S_i(B_i)$ de longueur 4
- on concatène : $C = C_1 C_2 \ldots C_8$ de longueur 32

alors $f_K(R) = P(C)$ qui est bien de longueur 32

### 3.1.4 Le schéma global de chiffrement

Partant du message en clair $p$,

- on découpe $p' = \sigma_0(p)$ en deux moitiés $(L_0, R_0)$
- pour $i = 1, \ldots, 16$, on calcule $(L_i, R_i) = (R_{i-1}, L_{i-1} \oplus f_{K_i}(R_{i-1}))$ $\qquad (\star)$
- après le dernier tour, on fait un dernier échange gauche-droite et on applique l'inverse de $\sigma_0$, soit : $\quad E_k(p) = \sigma_0^{-1}(R_{16}, L_{16})$

  qui est donc le résultat final du chiffrement DES du message clair $p$ avec la clé $k$

Remarque importante pour le déchiffrement : dans $(\star)$ on a immédiatement $R_{i-1} = L_i$ et $R_i = L_{i-1} \oplus f_{K_i}(R_{i-1})$ d'où $L_{i-1} = R_i \oplus f_{K_i}(L_i)$ car $+ = -$ dans $\mathbb{Z}/2\mathbb{Z}$

Donc le déchiffrement est la même suite d'opérations que le chiffrement, en inversant simplement l'ordre des tours, et les permutations initiale et finale