Algorithmes de chiffrement

1. Un chiffrement symétrique : Simplified DES

1.1. Description

Les algorithmes DES et AES sont hors-programme et trop complexes. On présente ici une version simplifiée de DES.

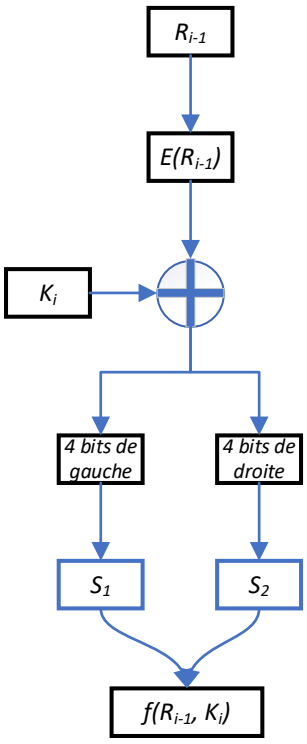
Le message à chifrer est divisé en blocs de 12 bits. L'algorithme de chiffrement est appliqué à chacun de ces blocs. La clé K est sur 9 bits.
On génère pour chaque tour i une clé K_i en prenant 8 bits consécutifs de la clé K en commençant par le i-ème bit. Ainsi, si K = 101110101, K_1 = 10111010, K_2 = 01110101, K_3 = 11101011, K_4 = 11010110 …

Le bloc de 12 bits s'écrit L_0R_0 où L_0 est constitué des 6 premiers bits et R_0 des 6 derniers.
Le i-ème tour transforme $L_{i-1}R_{i-1}$ en L_iR_i en utilisant la clé K_i de la manière suivante :

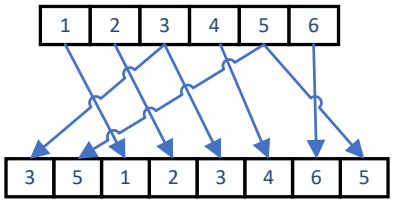
$$L_i = R_{i-1} \quad \text{et} \quad R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

L'opérateur $\oplus$ est le « ou exclusif » et f est une fonction définie ci-dessous.
Au bout d'un nombre n de tours, le message chiffré est R_nL_n.

Définition de f : la fonction f, s'appliquant à R_{i-1} et K_i, est définie par l'algorithme suivant.



E est une fonction d'expansion qui mélange les bits en dupliquant 2 bits :



Ainsi, le bloc R_{i-1} de 6 bits est étendu à un bloc de 8 bits, auquel on applique un « ou exclusif » avec la clé K_i. Le résultat est séparé en deux blocs de 4 bits.

Les boites S_1 et S_2 transforment chacune un bloc de 4 bits en un bloc de 3 bits en affectant à $b_1b_2b_3b_4$ le bloc situé ligne b_1b_2 et colonne b_3b_4.

Boite S_1

101	010	001	110
011	100	111	000
001	100	110	010
000	111	101	011

Boite S_2

100	000	110	101
111	001	011	010
101	011	000	111
110	010	001	100

La concaténation des deux blocs résultants est $f(R_{i-1}, K_i)$.

1.2. Exemple

Message d'entrée : L_0R_0 = 110010100111 ; clé : K = 100111101. Nombre de tours : 2.

<u>Chiffrement</u>
Premier tour : i = 1
K_1 = 10011110, L_0 = 110010, R_0 = 100111

$E(R_0)$ = 01100111
$E(R_0) \oplus K_1$ = 01100111 ⊕ 10011110 = 11111001 = 1111 1001

11 11 renvoie l'élément de S_1 situé ligne 3, colonne 3, soit 011.
10 01 renvoie l'élément de S_2 situé ligne 2, colonne 1, soit 011.
$f(R_0, K_1)$ = 011011
D'où, $L_1 = R_0$ = 100111
et $R_1 = L_0 \oplus f(R_0, K_1)$ = 110010 ⊕ 011011 = 101001

Deuxième tour : i = 2
K_2 = 00111101, L_1 = 100111, R_1 = 101001
$E(R_1)$ = 10101010
$E(R_1) \oplus K_2$ = 10101010 ⊕ 00111101 = 10010111 = 1001 0111

10 01 renvoie l'élément de S_1 situé ligne 2, colonne 1, soit 100.
01 11 renvoie l'élément de S_2 situé ligne 1, colonne 3, soit 010.
$f(R_1, K_2)$ = 100010
D'où, $L_2 = R_1$ = 101001
et $R_2 = L_1 \oplus f(R_1, K_2)$ = 100111 ⊕ 100010 = 000101

Le message chiffré est R_2L_2 = 000101101001.

<u>Déchiffrement</u>

Premier tour : i = 1
K_2 = 00111101, L_0 = 000101, R_0 = 101001
$E(R_0)$ = 10101010
$E(R_0) \oplus K_2$ = 10101010 ⊕ 00111101= 10010111 = 1001 0111

10 01 renvoie l'élément de S_1 situé ligne 2, colonne 1, soit 100.
01 11 renvoie l'élément de S_2 situé ligne 1, colonne 3, soit 010.
$f(R_0, K_2)$ = 100010
D'où, $L_1 = R_0$ = 101001
et $R_1 = L_0 \oplus f(R_0, K_2)$ = 000101 ⊕ 100010 = 100111

Deuxième tour : i = 2
K_1 = 10011110, L_1 = 101001, R_1 = 100111
$E(R_1)$ = 01100111
$E(R_1) \oplus K_1$ = 0110 0111 ⊕ 10011110= 11111001 = 1111 1001

11 11 renvoie l'élément de S_1 situé ligne 3, colonne 3, soit 011.
10 01 renvoie l'élément de S_2 situé ligne 2, colonne 1, soit 011.
$f(R_1, K_1)$ = 011011
D'où, $L_2 = R_1$ = 100111
et $R_2 = L_1 \oplus f(R_1, K_1)$ = 101001 ⊕ 011011= 110010

Le message déchiffré est R_2L_2 = 110010100111.

1.3. Exercice

Appliquer l'algorithme précédent pour chiffrer le message 101101101011 avec la clé *K* = 001101101.
Nombre de tours : 2.
Vérifier ensuite que le message déchiffré est bien le message de départ.

1.4. DES

Le vrai algorithme DES utilise une clé de 56 bits et travaille sur des blocs de 64 bits. Il commence par une permutation des données à l'intérieur de chaque bloc, suivi de 16 tours (rounds) de chiffrements, puis d'une autre permutation. La génération des clés utilisées à chaque tour, à partir de la clé initiale, est également plus complexe que la démarche décrite ici.
https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf
https://web.maths.unsw.edu.au/~lafaye/CCM/crypto/des.htm

2. Un chiffrement asymétrique : RSA

On décrit ici la méthode sur un exemple. La justification de cette méthode s'appuie davantage sur les mathématiques expertes que sur le programme de NSI.

2.1. Description sur un exemple

La partie la plus technique de l'algorithme, qu'on ne détaillera pas, est la génération automatique des clés, puisqu'un mauvais algorithme de génération constitue une faille importante de fiabilité.
On génère un nombre aléatoire *n*, et on effectue un test mathématique, test de Fermat ou test de Miller-Rabin, pour voir si *n* est premier. Si *n* n'est pas premier, on recommence jusqu'à obtenir un nombre premier.

Choix des clés
On va prendre des clés sur 8 bits, correspondant à des nombres entre 0 et $2^8 - 1 = 255$.
Une vraie clé RSA codée sur 2048 bits correspond à un nombre décimal à plus de 600 chiffres.

Prenons $p = 157$ et $q = 241$.
Alors $n = 37837$ et $\varphi(n) = 37440$.
On choisit *e* quelconque premier avec $\varphi(n)$. Le plus simple est de choisir e premier, ce qui donne une bonne probabilité qu'il soit premier avec $\varphi(n)$.
On prend $e = 26417$.
La clé publique est (26417, 37837).

L'algorithme d'Euclide étendu fournit $d = 10193$. C'est la clé privée.
d est choisi de manière à ce que $d*e \ \% \ \varphi(n) = 1$

Supposons que le message à chiffrer est $M = 28523$; on a bien $M < n$.
Alors le message chiffré est $C = M^e \% n = 28523^{26417} \ \% \ 37837 = 22486$.

On retrouve le message clair M avec $M = C^d \% n = 22486^{10193} \ \% \ 37837 = 28523$.

2.2. Exemple réaliste

n est codé sur 2048 bits et a 617 chiffres.

p = 954181907681679001581333113090096114454073543283378716348091222305938728513248736477145613228189524958262555181965502077686167783373151392775464128734404092146948586491162423871048974555805000619430824807456187571736245392837645101160861591063416131617167591150686635222043770407803091333524137807678234238613

q = 155994810409931125945761803406335925332006197690188014174227320661763718311298235224934356536891230822481837084788701843755050900249922391579315674226282293676311314783990254112624626500641633544049048057138728952675407716687909169615388332233675354270055739617056671809558773878891979414809919926239171939887

n = 148847425785389920105695375317033389300472188581288471249198107629246440960891693541957056818337315947018803974291531070714321274769576002439167697653191564607788841329087136337377331568136026047116218331568762084950493229050438731264968468488085105365260679263626206916313882859123220471021552873158768406770419920495097451551560457246450197629173077162023993304554885866192610954792994847097488727916698003358410808590347221096046992891587700339390806400701133383911257022078088229021237978330897818681706782811694787490627061744166571789250948448423500706770345136615709572415584554853047436487200149215136134437 07

e = 65537

d = 693532659990946579847645449239503116663750057298659498919009630371647973081128027526008313930260506241995544220664081782097519637207039200830435451048698844448699217056777819429963867245159481113682476001767508281741346299385720600257423616240012020299006683551900430962137529485052208722265916739364962093026041963806483583619537774119676720279243493290078519776193696172917169549741412961474715387952330538851926895321309516293944460778510332352897791680278084815648641119770984694178115487593314474775903892288256651072567781213110176131614799448475279064799000393496119002452827288731789140410194780911811785235 3

Le calcul de $C^d \% n$ n'est plus réalisé de manière naïve : il y a beaucoup trop de chiffres.
On utilise un algorithme d'exponentiation modulaire :
https://en.wikipedia.org/wiki/Modular_exponentiation

2.3. Exercice

On prend *p* = 5, *q* = 11 et *e* = 3.
Calculer *n* et φ(*n*).
Vérifier que *d* = 27 convient.
Chiffrer le message *M* = 42.
Déchiffrer le message et vérifier qu'on retrouve bien 42.

3. Projet

Choisir un algorithme de chiffrement puis écrire un code permettant de chiffrer et déchiffrer des messages avec ce chiffrement.

Pour chiffrer un message sous forme de chaine de caractères avec DES simplifié, il faut d'abord convertir le message en binaire, séparer ce message binaire en une liste de blocs de 12 bits, puis chiffrer chaque bloc en suivant l'algorithme décrit plus haut, et gérer le déchiffrement de manière similaire.

Le RSA est plus difficile à programmer. On doit convertir la chaine de caractères à chiffrer en liste de nombres inférieurs à la clé *n* pour pouvoir les chiffrer. Il faudra écrire des algorithmes mathématiques (test de primalité, exponentiation modulaire, algorithme d'Euclide étendu…). On peut pour cela traduire en Python les algorithmes correspondants donnés sur Wikipedia en pseudo-code.

Une option plus simple est le KID RSA que vous trouverez sur internet.