UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

Corrections des exercices de cryptographie

M1 Informatique

Auteur :
Christofi Maria

Sommaire

1 Feuille d'Exercices 1

1.1 Exercice : Code d'immeuble

On cherche à trouver le code d'accès d'un immeuble. Le digicode présente k caractères ; les combinaisons valides sont composées de n caractères.

Correction L'ensemble de cette exercice est basé sur la construction des arbres. Cette construction est donc laissée aux étudiants. Ici, on donne juste les bonnes réponses pour vérification.

1. Supposons que les caractères présents sur le digicode sont A, B, C et D et que le code n'est composé que de 2 lettres.
   — Représenter à l'aide d'un arbre tous les codes possibles ;
   — En déduire le nombre de codes possibles ?

   Correction 16 codes possibles.
2. Supposons que le code est composé de 3 lettres *distinctes* parmi A, B et C.
   — Représenter à l'aide d'un arbre tous les codes possibles ;
   — En déduire le nombre de codes possibles ?

   Correction 6 codes possibles.
3. Supposons que le code est composé de 3 lettres *distinctes* parmi A, B, C et D.
   — Représenter à l'aide d'un arbre tous les codes possibles ;
   — En déduire le nombre de codes possibles ?

   Correction 24 codes possibles.
4. Supposons que le digicode présente tous les chiffres de 0 à 9. On sait que le code est composé de 3 chiffres. En appliquant du talc sur celui-ci, on peut voir que les touches les plus appuyées sont 2, 5 et 8.
   — Représenter à l'aide d'un arbre tous les codes possibles ;
   — En déduire le nombre de codes possibles ?

   Correction 6 codes possibles.
5. Supposons que le digicode présente tous les chiffres de 0 à 9. On sait que le code est composé de 3 chiffres. En appliquant du talc sur celui-ci, on peut voir que les touches les plus appuyées sont 2, 5, 8 et 9.
   — Quel est le nombre de codes possibles ?

   Correction 64 codes possibles.
   — On suppose maintenant que la porte s'ouvre si on entre les bon chiffres, quel que soit leur ordre.
   - Représenter à l'aide d'un arbre les codes possibles.
   - Quel est le nombre de codes possibles ?

   Correction 20 codes possibles.
6. Supposons que le digicode présente tous les chiffres de 0 à 9 et les lettres A et B. On sait que le code est composé de 3 chiffres et se termine par une lettre. Quel est le nombre de codes possibles ?

   Correction 2000 codes possibles.
7. Reprenez les questions précédentes dans le cas général.

Correction Cette exercice est laissé aux lecteurs.

8. On suppose qu'un attaquant met en moyenne 3 secondes pour saisir un code. Quel est, pour chacun des cas précédents, le temps d'attaque maximal nécessaire à l'ouverture de la porte si $n = 10$ et pour les cas où $k = 4, 5$ ou 6.
9. On s'identifie maintenant au syndic qui souhaite protéger l'accès à l'immeuble. Pour cela, il dispose de deux solutions concurrentes :
   (a) La première consiste en deux claviers à 10 chiffres, placés côte à côte. L'accès est autorisé une fois que la personne a saisi 4 chiffres sur le premier clavier, puis 4 chiffres sur le second.
   (b) La seconde solution propose de placer deux digicodes identiques aux précédents sur chacune des deux portes d'accès successives.

   Quelle solution vous semble la plus intéressante du point de vue sécurité ?

1.2 Problème : La machine Enigma (Sujet de contrôle continu 2011–2012)

Énoncé La machine Enigma est un système électromécanique de chiffrement symétrique qui fût utilisé par l'armée allemande durant la Deuxième Guerre mondiale.

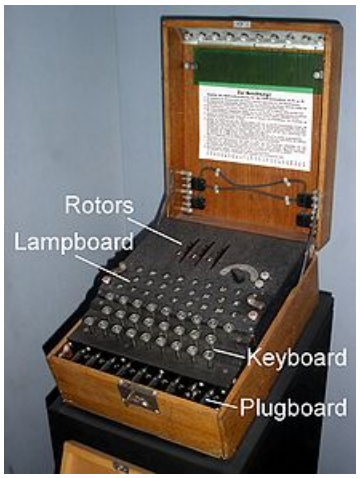


Figure 1 – Une machine Enigma militaire (Source : Wikipedia)

La clef secrète consiste à choisir
- la position de trois rotors (lesquels acceptent chacun 26 positions différentes) ;
- une connexion électrique permettant de réaliser une permutation de $\{a, b, c, \ldots, z\}$ ayant 14 points fixes et 6 échanges de deux caractères (aucun caractère ne peut être présent dans deux échanges différents). Par exemple,

$$[b \leftrightarrow t, e \leftrightarrow q, g \leftrightarrow z, h \leftrightarrow i, k \leftrightarrow p, m \leftrightarrow s]$$

  laisse $a, c, d, f, j, l, n, o, r, u, v, w, x$ et y inchangés et envoie b vers t et t vers b, e vers q et q vers e, etc.

Un exemple de machine Enigma simplifiée (limitée à 6 lettres) est représenté sur la figure 2.

(Quelques résultats de dénombrement)

Soit E un ensemble de n éléments distincts. On appelle *liste sans répétition* une suite ordonnée d'éléments distincts de E. Par exemple, si $E = \{1, 2, 3, 4\}$, $\mathcal{L}_1 = (1, 3, 4)$ et $\mathcal{L}_2 = (4, 3, 1)$ sont deux liste distinctes de tailles trois. On note A_n^k $(k \leq n)$ le nombre de listes sans répétition de taille k d'éléments d'un ensemble de taille n.

1. En énumérant pour chaque élément de la liste le nombre de choix possibles, montrer que

$$A_n^k = \frac{n!}{(n-k)!}$$

.

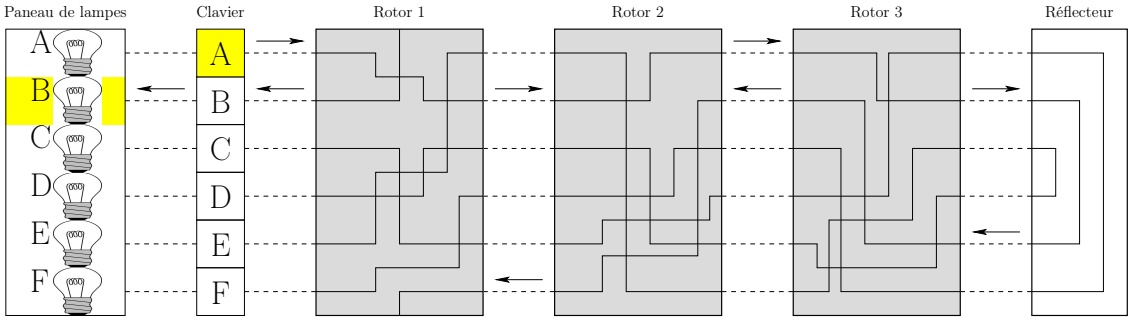


Figure 2 – Une machine Enigma simplifiée à 6 lettres

Correction Soit $E = \{x_1, ..., x_n\}$. On veut créer des listes sans répétition de taille k d'un ensemble de taille n. En énumérant pour chaque élément de la liste :
— Pour x_1 on a n choix possibles.
— Pour x_2 on a $n-1$ choix possibles.
— ...
— Pour x_k on a $n-(k-1)$ choix possibles.
Donc au total, on a $n \cdot (n-1) \cdot ... \cdot (n-(k-1)) = \frac{n!}{(n-k)!}$ listes sans répétition de taille k d'un ensemble de n éléments.

On appelle *combinaison* de k éléments de E tout sous-ensemble de E ayant k éléments. On note $\binom{n}{k}$ le nombre de combinaisons de k éléments d'un ensemble de taille n. Par exemple, pour $E = \{1,2,3,4\}$, ses combinaisons de 3 éléments sont $\{1,2,3\}, \{1,2,4\}, \{1,3,4\}$ et $\{2,3,4\}$ et $\binom{4}{3} = 4$

1. Montrer que pour toute combinaisons de taille k d'éléments de E, on peut construire $k!$ listes sans répétition de taille k (on rappelle que $0! = 1$).

   Correction On montrera ce fait par récurrence :
   — Il y a 1! façon de permuter 1 élément.
   — Supposons qu'il y a $k!$ façons de permuter k éléments.
   — Alors étant donné $k+1$ éléments, on en choisit 1 parmi $k+1$, ce qui donne $k+1$ possibilités. Il reste k éléments à ordonner, soit $k!$ possibilités. Au total, on a donc $k+1 \cdot k! = (k+1)!$ façons de permuter $k+1$ éléments.

2. En déduire que
$$A_n^k = k!\binom{n}{k}$$

   Correction Si on permute les éléments de chaque combinaison, on obtient tous les arrangements A_n^k. Il y a donc $k!$ fois plus d'arrangements que des combinaisons, i.e.
$$A_n^k = k!\binom{n}{k}$$
   .

3. En conclure que le nombre de combinaisons de k éléments d'un ensemble de taille n est
$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

   Correction $A_n^k = k!\binom{n}{k} \Rightarrow \binom{n}{k} = \frac{A_n^k}{k!} = \frac{\frac{n!}{(n-k)!}}{k!} = \frac{n!}{k! \cdot (n-k)!}$

Application à la taille des clefs

1. Nombre de clefs
   (a) Combien existe-t-il de positions initiales des rotors ?

Correction Chaque rotor a 26 éléments. Il y a donc $26 \cdot 26 \cdot 26$ positions initiales.

(b) Combien existe-t-il de choix possibles des 12 lettres permutées (i.e. le nombre de sous-ensembles de 12 lettres parmi 26) ?

Correction Il existe $\binom{26}{12}$ de choix possibles.

(c) Plaçons ces lettres dans une table de cette forme :

$$[\cdot \leftrightarrow \cdot, \cdot \leftrightarrow \cdot, \cdot \leftrightarrow \cdot, \cdot \leftrightarrow \cdot, \cdot \leftrightarrow \cdot, \cdot \leftrightarrow \cdot]$$

Combien existe-t-il de façon de placer les 12 lettres dans cette table ?

Correction Ce problème revient à construire des tables de 12 éléments sans répétitions. On a donc 12! possibilités.

(d) Parmi ces dernières, plusieurs sont équivalentes :

i. Au sein d'une même paire, $\ell_1 \leftrightarrow \ell_2$ et $\ell_2 \leftrightarrow \ell_1$ sont équivalentes. Combien faut-il éliminer de placements ?

Correction Ayant deux fois la même paire, on divise par 2 pour chaque paire. On a 6 paires, il faut diviser les placements globales par 2^6.

ii. Toutes les manières d'ordonner les différentes paires sont équivalentes. Combien faut-il éliminer de placements ?

Correction Pour les paires finalement obtenues, on veut toutes les permutations possibles. Donc pour les 6 paires, on a 6! permutations possibles.

(e) En déduire que le nombre de clefs différentes de la machine Enigma est

$$26^3 \binom{26}{12} \frac{12!}{(6!)\,2^6} = 1\ 764\ 486\ 127\ 404\ 000 \approx 1,76 \cdot 10^{15} \approx 2^{50.65}$$

2. Soit $(a)_{10} = (a_{n-1}, a_{n-2}, \ldots, a_0)_2$ un nombre de n bits (i.e. le n^e bit a_{n-1} de a est 1)

(a) En s'appuyant sur la définition de l'écriture binaire, montrer que

$$2^{n-1} \leq a$$

et que

$$a < 2^n$$

(On pourra utiliser le fait que $\sum_{i=0}^{n-1} q^i = \frac{1-q^n}{1-q}$)

Correction

$$\begin{array}{rcl} a & = & a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + ... + a_0 \cdot 2^0 \\ & \geq & a_{n-1} \cdot 2^{n-1} \text{ (car } 2^i > 0 \text{ et } a_j \geq 0) \\ & = & 2^{n-1} \text{ (car } a_{n-1} = 1) \end{array}$$

De l'autre coté :

$$\begin{array}{rcl} a & = & \sum_{i=0}^{n-1} a_i \cdot 2^i \\ & \leq & \sum_{i=0}^{n-1} 2^i \text{ (car } a_i = 0 \text{ ou } 1) \\ & = & \frac{1-2^n}{1-2} \text{ (en utilisant l'indice)} \\ & = & \frac{1-2^n}{-1} \\ & = & 2^n - 1 \\ & < & 2^n \end{array}$$

(b) En déduire que $n - 1 \leq \log_2 a < n$ et donc que $n = \lceil \log_2 a \rceil$

Correction
$$\begin{array}{lll} 2^{n-1} \leq a < 2^n & \Rightarrow & \log_2(2^{n-1}) \leq \log_2 a < \log_2 2^n \\ & \Rightarrow & n-1 \leq \log_2 a < n \end{array}$$
et donc $n = \lceil \log_2 a \rceil$

(c) Combien de bits sont donc nécessaires pour représenter une clef d'Enigma ?

Correction Le nombre de clés différentes de la machine Enigma étant $2^{50.65}$ (d'après la question 1.e), le nombre de bits nécessaires est $n = \lceil \log_2(2^{50.65}) \rceil - 1 = 51 - 1 = 50$.

3. En déduire quelle est la complexité d'une recherche exhaustive sur une telle clef.

1.3 Exercice : type BAC

1.3.1 Préliminaires

1. Déterminer le reste dans la division euclidienne de 2011 par 11.

Correction $2011 \equiv 11 \cdot 182 + 9 \equiv 0 \cdot 182 + 9 \equiv 9$ [11].

2. Déterminer le reste dans la division euclidienne de 2^{10} par 11.

Correction $2^{10} \equiv 2^{2 \cdot 5} \equiv (2^5)^2 \equiv 32^2 \equiv (-1)^2 \equiv 1$ [11].

3. Déterminer le reste dans la division euclidienne de $2^{2011} + 2011$ par 11.

Correction $2^{2011} \equiv 2^{201 \cdot 10+1} \equiv 2^{201 \cdot 10} \cdot 2^1 \equiv 2^{10} \cdot 2^{201} \cdot 2^1 \equiv 1^{201} \cdot 2^1 \equiv 2$ [11] or $2011 \equiv 9$ [11]
donc $2^{2011} + 2011 \equiv 2 + 9 \equiv 0$ [11].

1.3.2 Entrée dans le vif du sujet

$\forall p \in \mathbb{N}, \forall n \in \mathbb{N}^\star$, on définit $A_n = 2^n + p$ et d_n le pgcd de A_n et A_{n+1}.

1. Montrer que d_n divise 2^n.

Correction $d_n = pgcd(A_n, A_{n+1})$ donc $d_n | A_n = 2^n + p$ et $d_n | A_{n+1} = 2^{n+1} + p$
Par conséquent $d_n | A_{n+1} - A_n = 2^{n+1} - 2^n = 2^n.(2-1) = 2^n$

2. Déterminer la parité de A_n en fonction de celle de p. Justifier.

Correction $A_n \equiv 2^n + p \equiv 0 + p \equiv p$ [2] donc A_n et p ont la même parité.

3. Déterminer la parité de d_n en fonction de celle de p.
En déduire le pgcd de $2^{2011} + 2011$ et de $2^{2012} + 2011$.

Correction D'après la question précédente p, A_n et A_{n+1} ont même parité.
— Si p est pair, on obtient que A_n et A_{n+1} le sont aussi donc $2 | pgcd(A_n, A_{n+1}) = d_n$ et d_n est pair.
— Si p est impair, d_n ne peut pas être pair, sinon A_n le serait, ce qui contredit le résultat de la question précédente.

Finalement d_n et p ont aussi même parité.
De plus $d_{2011} = pgcd(A_{2011}, A_{2012}) = 2^{2011} + pgcd(2011, 2^{2012}) + 2011$ pour $p = 2011$.
On a vu précédemment que $d_n | 2^n$, donc d_{2011} est de la forme 2^k avec $k \in [\![0, 2011]\!]$
Or p est impair donc $d_{2011} = 2^k$ l'est aussi et nécessairement $k = 0$.
Ce qui permet d'affirmer que $pgcd(A_{2011}, A_{2012}) = d_{2011} = 1$.

1.4 Exercice : Groupes, Anneaux et Corps

1. Groupes

Énoncé

(a) Quel est le cardinal de l'ensemble des permutations de n éléments (noté $\mathcal{S}_n$).

Correction Soit σ une permutation de $[\![1,n]\!]$ dans $[\![1,n]\!]$, c'est à dire une bijection de $[\![1,n]\!]$ dans $[\![1,n]\!]$
Déterminer σ revient à déterminer un placement des entiers de 1 à n dans n cases.

— Pour la 1ère case $\sigma(1)$, on peut mettre n'importe quel entier de 1 à n. On a donc n choix.
— Pour la seconde case $\sigma(2)$, on ne peut plus prendre l'entier choisi pour $\sigma(1)$. Ainsi il nous reste $n-1$ choix.
— De proche en proche on arrive à la dernière case $\sigma(n)$ pour laquelle il ne reste plus que $n-(n-1)=1$ choix.
Finalement le nombre de permutations de $\mathcal{S}_n$ correspond à $n \cdot (n-1) \cdot ... \cdot 1 = n!$

(b) Soient σ_1 la permutation $\begin{pmatrix}1&2&3&4&5\\2&3&4&5&1\end{pmatrix}$ et σ_2 la permutation $\begin{pmatrix}1&2&3&4&5\\2&4&3&5&1\end{pmatrix}$. Déterminer $\sigma_1 \circ \sigma_2$.

Correction $\begin{pmatrix}1&2&3&4&5\\2&3&4&5&1\end{pmatrix} \circ \begin{pmatrix}1&2&3&4&5\\2&4&3&5&1\end{pmatrix} = \begin{pmatrix}1&2&3&4&5\\3&5&4&1&2\end{pmatrix} = \begin{pmatrix}1&3&4\end{pmatrix} \circ \begin{pmatrix}2&5\end{pmatrix}$.
Le support de cette permutation est $[\![1,n]\!]$ car elle ne laisse aucun élément invariant.
$\sigma(1)=3$, $\sigma(3)=4$ et $\sigma(4)=1$ donc 1, 3 et 4 ont pour orbite $\{1,3,4\}$.
De même, $\sigma(2)=5$ et $\sigma(5)=2$ donc l'orbite des éléments 2 et 5 est $\{2,5\}$.

(c) Le groupe des permutations est-il abélien ?

Correction On a :

$$\begin{pmatrix}1&2&3&4&5\\2&3&4&5&1\end{pmatrix} \circ \begin{pmatrix}1&2&3&4&5\\2&4&3&5&1\end{pmatrix} = \begin{pmatrix}1&2&3&4&5\\3&5&4&1&2\end{pmatrix}$$

et :

$$\begin{pmatrix}1&2&3&4&5\\2&4&3&5&1\end{pmatrix} \circ \begin{pmatrix}1&2&3&4&5\\2&3&4&5&1\end{pmatrix} = \begin{pmatrix}1&2&3&4&5\\4&*&*&*&*\end{pmatrix}$$

Donc le groupe des permutations (appelé aussi groupe symétrique) n'est pas abélien.

(d) Citer d'autres groupes.

Correction $(\mathbb{Z},+)$, $(\mathbb{Q},+)$, $(\mathbb{Q}^\star,\times)$, $(\mathbb{R},+)$, $(\mathbb{R}^\star,\times)$, $(\mathbb{Z}/n\mathbb{Z},+)$

2. Anneaux

(a) Montrer que l'inverse de $\begin{pmatrix}a&b\\c&d\end{pmatrix}$ est $\frac{1}{ad-bc}\begin{pmatrix}d&-b\\-c&a\end{pmatrix}$

Correction $\begin{pmatrix}a&b\\c&d\end{pmatrix} \cdot \frac{1}{ad-bc} \cdot \begin{pmatrix}d&-b\\-c&a\end{pmatrix} = \begin{pmatrix}da-bc&-ab+ba\\cd-dc&-cb+da\end{pmatrix} = \begin{pmatrix}1&0\\0&1\end{pmatrix} = I_2$
De même $\frac{1}{ad-bc} \cdot \begin{pmatrix}d&-b\\-c&a\end{pmatrix} \cdot \begin{pmatrix}a&b\\c&d\end{pmatrix} = I_2$
D'où le résultat.

(b) Citer d'autres anneaux.

Correction $(\mathcal{M}_n(\mathbb{R}), +, \times)$, les polynômes, $(\mathbb{Z}/n\mathbb{Z}, +, \times)$.

3. Corps

   (a) Montrer que l'anneau $\mathbb{Z}/n\mathbb{Z}$ est un corps si et seulement si n est premier.

   Correction Soit x un élément de $[\![1, n-1]\!]$.
   Comme n est premier et que n ne divise pas x, on peut appliquer le théorème de BEZOUT :
   $(\exists u, v \in \mathbb{Z}), xu + nv = 1$
   Ainsi comme il s'agit d'une égalité entre deux nombres, on peut affirmer que leur reste dans la division euclidienne par n est la même :
   $\overline{x \cdot u} + \overline{0} \equiv \overline{1}\ [n]$ d'où $\overline{x} \cdot \overline{u} \equiv \overline{1}\ [n]$
   Ainsi tout élément non nul de $\mathbb{Z}/n\mathbb{Z}$ a un inverse et donc l'anneau $\mathbb{Z}/n\mathbb{Z}$ est un corps.

   Le théorème de BEZOUT ayant une réciproque dans notre cas, on peut donc remonter le raisonnement précédent pour montrer l'implication inverse, et donc l'équivalence.

   (b) Exhiber un contre-exemple au fait que $\mathbb{Z}/4\mathbb{Z}$ soit un corps.

   Correction $\mathbb{Z}/4\mathbb{Z}$ ne peut être un corps puisqu'il n'est pas intègre, en effet on a $2.2 \equiv 0\ [4]$ alors que $2 \not\equiv 0\ [4]$

   (c) Donner la table de multiplication de $(\mathbb{Z}/6\mathbb{Z})^\star$ et de $(\mathbb{Z}/5\mathbb{Z})^\star$.

   Correction

$\times$	1	2	3	4	5
1	1	2	3	4	5
2	2	4	0	2	4
3	3	0	3	0	3
4	4	2	0	4	3
5	5	4	3	3	1

$\times$	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

   (d) Citer d'autres corps.

   Correction $(\mathbb{Q}, +, \times)$, les fractions rationnelles, $(\mathbb{Z}/p\mathbb{Z}, +, \times)$ avec p premier.

1.5 Exercice : Petit théorème de FERMAT

Énoncé Soit p un nombre premier.

1. Montrer que, pour tout $k \in [\![1, p-1]\!]$, p divise $\binom{p}{k}$.

   Correction
   $$\begin{aligned} k \cdot \binom{p}{k} &= k \cdot \frac{p!}{k!\cdot(n-k)!} = p \cdot \frac{(p-1)!}{(k-1)!\cdot(n-k)!} = p \cdot \frac{(p-1)!}{(k-1)!\cdot((n-1)-(k-1))!} \\ &= p \cdot \binom{p-1}{k-1} \end{aligned}$$
   Or $pgcd(p, k) = 1$ quand $k \in [\![1, p-1]\!]$ donc p divise $\binom{p}{k}$ d'après le théorème de GAUSS.

2. Montrer par récurrence sur $n \in \mathbb{N}$ que $n^p \equiv n\ [p]$.

Correction
— Initialisation ($n = 0$) : $0^p = 0$ donc $0^p \equiv 0\ [p]$.
— Hérédité : Supposons $\mathcal{H}_n$ vraie pour $n \in \mathbb{N}$ et montrons que $\mathcal{H}_{n+1}$ l'est.
$(n+1)^p = \sum_{k=0}^{p} \binom{p}{k} \cdot n^k \cdot 1^{n-k}$ d'après la formule du binôme de NEWTON
$= \binom{p}{0} \cdot 1^0 + \sum_{k=1}^{p-1} \binom{p}{k} \cdot n^k \cdot 1^{n-k} + \binom{p}{p} \cdot n^p$
Ainsi comme $p | \binom{p}{k}$ pour $k \in [\![1, p-1]\!]$, on a $(n+1)^p \equiv 1 + \sum_{k=1}^{p-1} 0 \cdot n^k \cdot 1^{n-k} + n^p\ [p]$
Or d'après $\mathcal{H}_n$, on a $n^p \equiv n\ [p]$, donc $(n+1)^p \equiv n+1\ [p]$ et $\mathcal{H}_{n+1}$ est vraie.
Ainsi par récurrence, on obtient la propriété.

3. Montrer que, si n n'est pas divisible par p, $n^{p-1} \equiv 1\ [p]$.

Correction On vient de voir que $n^p \equiv n\ [p]$.
Si n n'est pas divisible par p alors $pgcd(p, n) = 1$ et donc n est inversible dans $\mathbb{Z}/p\mathbb{Z}$, d'où : $n^p \cdot n^{-1} \equiv n \cdot n^{-1}\ [p]$ et donc $n^{p-1} \equiv 1\ [p]$.

4. Vérifier que $2011^6 \equiv 1\ [7]$.

Correction $2011^6 \equiv (287 \cdot 7 + 2)^6 \equiv 2^6 \equiv 2^{2 \cdot 3} \equiv (2^3)^2 \equiv 8^2 \equiv 1^2 \equiv 1\ [7]$

1.6 Exercice : Algorithme d'EUCLIDE - Application

1.6.1 Avec des entiers

1. Soient a et b deux entiers, montrer que $pgcd(a, b) = pgcd(b, r)$ où r est le reste de la division euclidienne de a par b.

Correction En posant la division euclidienne de a par b, on obtient deux entiers q et r tels que $a = b \cdot q + r$ avec $0 \leq r < b$.
— Montrons que $pgcd(a, b) | pgcd(b, r)$:
Par définition, $pgcd(a, b) | b$; d'autre part comme $pgcd(a, b)$ divise a et b, il divise donc aussi $a - b \cdot q = r$. Ainsi $pgcd(a, b) | pgcd(b, r)$.
— Montrons que $pgcd(b, r) | pgcd(a, b)$:
De même, $pgcd(b, r) | b$ et $pgcd(b, r) | b \cdot q + r = a$ donc $pgcd(b, r) | pgcd(a, r)$.
Finalement on obtient que $pgcd(a, b) = pgcd(b, r)$.

2. Déterminer le pgcd de 442 et 495 au moyen de l'algorithme d'EUCLIDE.

Correction

q	1	8	2	1	17
495	442	53	18	17	1
r	53	18	17	1	0

On en déduit que $pgcd(442, 495) = 1$.

3. En effectuant une remontée de l'algorithme d'EUCLIDE,
déterminer $u, v \in \mathbb{Z}$ tels que $442 \cdot u + 495 \cdot v = 1$.

Correction Les divisions euclidiennes successives calculées par l'algorithme se résument ainsi :
$495 = 442 \cdot 1 + 53 \Leftrightarrow 53 = 495 - 442 \cdot 1$
$442 = 53 \cdot 8 + 18 \Leftrightarrow 18 = 442 - 8 \cdot 53$
$53 = 18 \cdot 2 + 17 \Leftrightarrow 17 = 53 - 2 \cdot 18$
$18 = 17 \cdot 1 + 1 \Leftrightarrow 1 = 18 - 1 \cdot 17$

Ainsi en procédant par des substitutions successives :
$1 = 18 - 1 \cdot 17$
$1 = 18 - 1 \cdot (53 - 2 \cdot 18) = 3 \cdot 18 - 53$
$1 = 3 \cdot (442 - 8 \cdot 53) - 53$
$1 = 3 \cdot 442 - 25 \cdot 53$
$1 = 3 \cdot 442 - 25 \cdot (495 - 442) = 28 \cdot 442 - 25 \cdot 495 = 28 \cdot 442 + (-25) \cdot 495$
Ainsi $u = 28$ et $v = -25$ conviennent.

4. En déduire l'inverse de 442 dans $\mathbb{Z}/495\mathbb{Z}$.

Correction Ainsi ces 2 nombres étant égaux, leurs restes dans la division euclidienne par 495 le sont aussi :
$442 \cdot u + 495 \cdot v \equiv 1\ [495]$ or $495 \equiv 0\ [495]$ donc $442 \cdot u \equiv 1\ [495]$.
Par conséquent l'inverse de 442 dans $\mathbb{Z}/495\mathbb{Z}$ est $u = 28$.

5. Résoudre l'équation $442 \cdot u + 495 \cdot v = 1$ avec u et v dans $\mathbb{Z}$.

Correction On cherche u et v tels que $442 \cdot u + 495 \cdot v = 1$, et on sait que $28 \cdot 442 + (-25) \cdot 495 = 1$. Ainsi par différence : $442 \cdot (u - 28) + 495 \cdot (v + 25) = 0$ donc $442 \cdot (u - 28) = -495 \cdot (v + 25)$
Comme $pgcd(442, 495) = 1$, on obtient en appliquant le théorème de GAUSS que $442 | v + 25$
Ainsi $v + 25$ est de la forme $442 \cdot k$ avec $k \in \mathbb{Z}$, d'où $v = 442 \cdot k - 25$.
En reprenant l'équation précédente :
$442 \cdot (u - 28) + 495 \cdot (v + 25) = 0 \Leftrightarrow 442 \cdot (u - 28) + 495 \cdot (442 \cdot k) = 0 \Leftrightarrow u = -495 \cdot k + 28$
On vient de prouver que les solutions sont de la forme $u = -495 \cdot k + 28$ et $v = 442 \cdot k - 25$ mais cela ne veux pas dire que tout couple d'entiers (u, v) sous cette forme est solution de l'équation diophantienne $442 \cdot u + 495 \cdot v = 1$.

Montrons donc la réciproque :
Soit $k \in \mathbb{Z}$, posons $u = -495 \cdot k + 28$ et $v = 442 \cdot k - 25$.
Dans ce cas $442 \cdot u + 495 \cdot v = 442 \cdot (-495 \cdot k + 28) + 495 \cdot (442 \cdot k - 25) = 442 \cdot 28 - 495 \cdot 25 = 1$
Donc tous les couples $(-495 \cdot k + 28, 442 \cdot k - 25)$ sont solution de l'équation.

Finalement l'ensemble des solutions est l'ensemble des entiers $u = -495 \cdot k + 28$ et $v = 442 \cdot k - 25$ avec $k \in \mathbb{Z}$

1.6.2 Avec des polynômes

1. Effectuer la division euclidienne de $X^3 + X^2 + X + 1$ par $X - 1$.

Correction

$$\begin{array}{rl} X^3 + X^2 + X + 1 & = (X-1)(X^2+2X+3) + 4 \\ \underline{-X^3 + X^2} & \\ 2X^2 + X & \\ \underline{-2X^2 + 2X} & \\ 3X + 1 & \\ \underline{-3X + 3} & \\ 4 & \end{array}$$

2. Déterminer deux polynômes $U(X)$ et $V(X)$ vérifiant :
$(X^3 + X^2 + X + 1) \cdot U(X) + (X - 1) \cdot V(X) = 4$.

Correction Comme $X^3 + X^2 + X + 1 = (X - 1) \cdot (X^2 + 2X + 3) + 4$, on obtient :

$$(X^3 + X^2 + X + 1) - (X - 1) \cdot (X^2 + 2X + 3) = 4$$

Ainsi $U(X) = 1$ et $V(X) = -X^2 - 2X - 3$ conviennent.

1.7 Exercice : La part du butin

Énoncé Une bande de 17 pirates possède un trésor constitué de pièces d'or d'égale valeur. Ils projettent de se les partager également, et de donner le reste au cuisinier chinois. Celui-ci recevrait alors 3 pièces. Mais les pirates se querellent, et six d'entre eux sont tués. Un nouveau partage donnerait au cuisinier 4 pièces. Dans un naufrage ultérieur, seuls le trésor, six pirates et le cuisinier sont sauvés, et le partage donnerait alors 5 pièces d'or à ce dernier.

Quelle est la fortune minimale que peut espérer le cuisinier s'il décide d'empoisonner le reste des pirates ?

Correction En notant x le nombre de pièces d'or du trésor l'énoncé permet d'affirmer que :

$$\begin{cases} x & \equiv & 3 & [17] \\ x & \equiv & 4 & [11] \\ x & \equiv & 5 & [6] \end{cases}$$

Comme 6, 11 et 17 sont deux à deux premiers entre eux, on peux utiliser le théorème des restes chinois

Pour cela on a besoin de connaître i_1 l'inverse de $6 \cdot 11$ dans $\mathbb{Z}/17\mathbb{Z}$, i_2 l'inverse de $6 \cdot 17$ dans $\mathbb{Z}/11\mathbb{Z}$ et i_3 l'inverse de $11 \cdot 17$ dans $\mathbb{Z}/6\mathbb{Z}$ ce qui nous permettra d'affirmer que :

$x \equiv 3 \cdot i_1 \cdot M_1 + 4 \cdot i_2 \cdot M_2 + 5 \cdot i_3 \cdot M_3$ $[6 \cdot 11 \cdot 17]$ avec $M_1 = 6 \cdot 11$, $M_2 = 6 \cdot 17$ et $M_3 = 11 \cdot 17$.

— Calcul de i_1

Comme $pgcd(17, 66) = 1$, d'après le théorème de Bezout : $(\exists u, v \in \mathbb{Z}), 17 \cdot u + 66 \cdot v = 1$. Ainsi $66 \cdot v \equiv 1$ $[17]$ et v est l'inverse de 66 dans $\mathbb{Z}/17\mathbb{Z}$.
Déterminons donc un tel couple (u, v) (qualifié de couple de Bezout) au moyen de l'algorithme d'Euclide étendu.

Commençons par écrire les divisions euclidiennes successives :
$66 = 17 \cdot 3 + 15 \Leftrightarrow 15 = 66 - 3 \cdot 17$
$17 = 15 \cdot 1 + 2 \Leftrightarrow 2 = 17 - 15$
$15 = 2 \cdot 7 + 1 \Leftrightarrow 1 = 15 - 7 \cdot 2$
Puis effectuons la remontée d'Euclide :
$1 = 15 - 7 \cdot 2$
$1 = 15 - 7 \cdot (17 - 15) = 15 - 7 \cdot 17 + 7 \cdot 15 = 8 \cdot 15 - 7 \cdot 17$
$1 = 8 \cdot (66 - 3.17) - 7 \cdot 17 = 8 \cdot 66 - 3 \cdot 8 \cdot 17 - 7 \cdot 17 = -31 \cdot 17 + 8 \cdot 66 = (-31) \cdot 17 + 8 \cdot 66$
Ainsi un couple solution est le couple $(-31, 8)$ et donc $i_1 = 8$.

— Calcul de i_2
Commençons par écrire les divisions euclidiennes successives :
$102 = 11 \cdot 9 + 3 \Leftrightarrow 3 = 102 - 11 \cdot 9$
$11 = 3 \cdot 3 + 2 \Leftrightarrow 2 = 11 - 3 \cdot 3$
$3 = 2 \cdot 1 + 1 \Leftrightarrow 1 = 3 - 2$
Puis effectuons la remontée d'Euclide :
$1 = 3 - 2$
$1 = 3 - (11 - 3 \cdot 3) = 4 \cdot 3 - 11$
$1 = 4 \cdot (102 - 11 \cdot 9) - 11 = 4 \cdot 102 - 37 \cdot 11 = 4 \cdot 102 + (-37) \cdot 11$
On en déduit que $i_2 = 4$.

— Calcul de i_3
On obtient très vite $i_3 = 1$ puisque $187 = 6 \cdot 31 + 1$ ($\Leftrightarrow 1 = 187 + (-31) \cdot 6$).

Finalement $x \equiv 3 \cdot 8 \cdot 66 + 4 \cdot 4 \cdot 102 + 5 \cdot 1 \cdot 187 \equiv 4151 \equiv 1122 \cdot 3 + 785 \equiv 785$ $[1122]$.
On en déduit que le nombre de pièces d'or du butin est de la forme $1122 \cdot k + 785$ avec $k \in \mathbb{N}$ et donc qu'il en contient au moins 785.

2 Feuille d'Exercices 2

2.1 Exercice

On cherche à chiffrer un message $M \in \{0,1,2\}$ au moyen d'une clé aléatoire partagée $K \in \{0,1,2\}$.

2.1.1 Question 1

Énoncé Supposons qu'on procède de la façon suivante : on représente K et M en utilisant deux bits (00, 01 ou 10), puis en XORant les deux représentations. Est-ce que ce protocole vous paraît bon ? Expliquer.

Correction Afin d'analyser ce protocole, examinons la répartition des textes clairs par rapport aux textes chiffrés. Pour ce faire, on va donner tous les chiffrements pour tous les messages possibles :

K/M	$\{0\}=00$	$\{1\}=01$	$\{2\}=10$
$\{0\}=00$	$00\oplus 00=00=\{0\}$	$00\oplus 01=01=\{1\}$	$00\oplus 10=10=\{2\}$
$\{1\}=01$	$01\oplus 00=01=\{1\}$	$01\oplus 01=00=\{0\}$	$01\oplus 10=11=\{3\}$
$\{2\}=10$	$10\oplus 00=10=\{2\}$	$10\oplus 01=11=\{3\}$	$10\oplus 10=00=\{0\}$

Certains chiffrements sont impossibles ($\{3\}$ ne fait pas parti de l'ensemble des chiffrés).
De plus, dans le cas où un attaquant observe un chiffré égal à $\{0\}$, celui-ci obtient une information "très importante" : la clé et le message sont identiques.
C'est là la faiblesse de ce protocole.

2.1.2 Question 2

Énoncé Donner un bon schéma de chiffrement dans ce contexte.

Correction Afin de rendre ce protocole un peu plus robuste, il faut s'affranchir de la faiblesse précédemment évoquée. Pour cela on va changer la façon dont on représente les messages et les clés en considérant que ce sont des entiers modulo 3. L'examen du protocole devient donc :

K/M	$0 \mod 3$	$1 \mod 3$	$2 \mod 3$
$0 \mod 3$	$0+0=0 \mod 3$	$0+1=1 \mod 3$	$0+2=2 \mod 3$
$1 \mod 3$	$1+0=1 \mod 3$	$1+1=2 \mod 3$	$1+2=0 \mod 3$
$2 \mod 3$	$2+0=2 \mod 3$	$2+1=0 \mod 3$	$2+2=1 \mod 3$

Ce système de chiffrement remplace avantageusement le système précédent car l'attaquant n'est plus capable d'obtenir des informations précises sur le couple (clair,clé) en observant le message chiffré.

2.2 Chiffrement de Hill

Le schéma de chiffrement symétrique suivant a été inventé en 1929 par Lester S. Hill.

Soit m un entier strictement positif.
- L'ensemble des messages clairs est $\mathcal{P} = (\mathbb{Z}/26\mathbb{Z})^m$, et l'ensemble des messages chiffrés est $\mathcal{C} = (\mathbb{Z}/26\mathbb{Z})^m$.
- L'ensemble des clés est $\mathcal{K} = \{\text{matrices } m \times m \text{ inversibles dans } \mathbb{Z}/26\mathbb{Z}\}$.
- Pour toute clé $K \in \mathcal{K}$, on définit la fonction de chiffrement $\mathcal{E}_K$ par $\mathcal{E}_K(x) = x \cdot K$, et la fonction de déchiffrement par $\mathcal{D}_K(y) = y \cdot K^{-1}$, où toutes les opérations sont faites dans $\mathbb{Z}/26\mathbb{Z}$.

Supposons que l'on sache que le texte clair

conversation

donne le texte chiffré

HIARRTNUYTUS

par le chiffrement de Hill (où m n'est pas spécifié).

Énoncé Déterminer la clé utilisée.

Correction
On propose ici une démarche générale d'attaque de ce type de chiffrement linéaire :

1. Déterminer les tailles possibles pour m :
   Indices : on a chiffré deux fois la même lettre de manière différente donc certaines valeurs de m sont impossibles.
2. Une fois la valeur de m déterminée, il reste à trouver la valeur de la clé utilisée. Pour ceci on écrit, par bloc de taille m, l'ensemble des équations qui doivent être satisfaites. Ceci nous permet de trouver les clés potentielles ;
3. Pour finir, on regroupe les blocs pour trouver la clé qui satisfait toutes les équations.

On propose tout de même quelques pistes pour la correction :
Ici on a 12 lettres. Donc nécessairement $m|12$. Ainsi, il faut examiner successivement les valeurs suivantes de m :

— $\underline{m = 1 :}$ Impossible car sinon 'o' (et 'n') aurait toujours le même chiffré ce qui n'est pas le cas ici ;
— $\underline{m \in \{2, 3\} :}$ A priori ces cas ne sont pas à exclure et à première vue on a moins d'inconnues que d'équations ce qui permettrait de trouver une unique clé ;
— $\underline{m \in \{4, 6, 12\} :}$ A priori ces cas ne sont pas à exclure mais ne permettent pas de déterminer une clé unique puisque l'on dispose d'un nombre d'inconnues supérieur au nombre d'équations.

Maintenant que les valeurs possibles de m ont été déterminées, il faut passer à l'écriture des équations qui doivent être satisfaites dans les différents cas.

$\underline{m = 2}$ On cherche une matrice K de taille 2 qui assure la cohérence du chiffrement. On suppose donc $K = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$.
Écrivons maintenant l'ensemble des relations qui devraient être satisfaites si on a choisi la bonne valeur pour m :

$$\begin{array}{lllllllll}
(1) & \text{"c"} \cdot a + \text{"o"} \cdot c & = & 2a + 14c & = & \text{"h"} & = & 7 & \mod 26 \\
(2) & \text{"c"} \cdot b + \text{"o"} \cdot d & = & 2b + 14d & = & \text{"i"} & = & 8 & \mod 26 \\
(3) & \text{"n"} \cdot a + \text{"v"} \cdot c & = & 13a + 21c & = & \text{"a"} & = & 0 & \mod 26 \\
(4) & \text{"n"} \cdot b + \text{"v"} \cdot d & = & 13b + 21d & = & \text{"r"} & = & 17 & \mod 26 \\
(5) & \text{"e"} \cdot a + \text{"r"} \cdot c & = & 4a + 17c & = & \text{"r"} & = & 17 & \mod 26 \\
(6) & \text{"e"} \cdot b + \text{"r"} \cdot d & = & 4b + 17d & = & \text{"t"} & = & 19 & \mod 26 \\
(7) & \text{"s"} \cdot a + \text{"a"} \cdot c & = & 18a + 0c & = & \text{"n"} & = & 13 & \mod 26 \\
(8) & \text{"s"} \cdot b + \text{"a"} \cdot d & = & 18b + 0d & = & \text{"u"} & = & 20 & \mod 26 \\
(9) & \text{"t"} \cdot a + \text{"i"} \cdot c & = & 19a + 8c & = & \text{"y"} & = & 24 & \mod 26 \\
(10) & \text{"t"} \cdot b + \text{"i"} \cdot d & = & 19b + 8d & = & \text{"t"} & = & 19 & \mod 26 \\
(11) & \text{"o"} \cdot a + \text{"n"} \cdot c & = & 14a + 13c & = & \text{"u"} & = & 20 & \mod 26 \\
(12) & \text{"o"} \cdot b + \text{"n"} \cdot d & = & 14b + 13d & = & \text{"s"} & = & 18 & \mod 26
\end{array}$$

Comme $2|26$, regardons l'équation (1) modulo 2, on obtient que $0 = 1 mod 2$ ce qui est impossible. Le cas $m = 2$ est donc exclu.

$\underline{m = 3}$ Cette fois on suppose donc que la matrice K est de la forme :

$$K = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

Les équations obtenues sont alors :

$$\begin{array}{lrcrl}
(1) & 2a+14d+13g & = & 7 & \bmod 26\\
(2) & 2b+14e+13h & = & 8 & \bmod 26\\
(3) & 2c+14f+13i & = & 0 & \bmod 26\\
(4) & 21a+4d+17g & = & 17 & \bmod 26\\
(5) & 21b+4e+17h & = & 17 & \bmod 26\\
(6) & 21c+4f+17i & = & 19 & \bmod 26\\
(7) & 18a+0d+19g & = & 13 & \bmod 26\\
(8) & 18b+0e+19h & = & 20 & \bmod 26\\
(9) & 18c+0f+19i & = & 24 & \bmod 26\\
(10) & 8a+14d+13g & = & 19 & \bmod 26\\
(11) & 8b+14e+13h & = & 20 & \bmod 26\\
(12) & 8c+14f+13i & = & 18 & \bmod 26
\end{array}$$

D'après le théorème des restes chinois, il existe un isomorphisme entre $\mathbb{Z}/26\mathbb{Z}$ et $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/13\mathbb{Z}$. En d'autre termes, pour résoudre ce système d'équations, on peut le faire dans $\mathbb{Z}/2\mathbb{Z}$ et dans $\mathbb{Z}/13\mathbb{Z}$ pour retrouver les solutions dans $\mathbb{Z}/26\mathbb{Z}$. Ceci est préférable car $\mathbb{Z}/26\mathbb{Z}$ n'est pas un corps contrairement à $\mathbb{Z}/2\mathbb{Z}$ et à $\mathbb{Z}/13\mathbb{Z}$.
Résolvons déjà le système dans $\mathbb{Z}/2\mathbb{Z}$, on est ramené au système :

$$\begin{array}{lrcrl}
(1) & g & = & 1 & \bmod 2\\
(2) & h & = & 0 & \bmod 2\\
(3) & i & = & 0 & \bmod 2\\
(4) & a+g & = & 1 & \bmod 2\\
(5) & b+h & = & 1 & \bmod 2\\
(6) & c+i & = & 1 & \bmod 2\\
(7) & g & = & 1 & \bmod 2\\
(8) & h & = & 0 & \bmod 2\\
(9) & i & = & 0 & \bmod 2\\
(10) & g & = & 1 & \bmod 2\\
(11) & h & = & 0 & \bmod 2\\
(12) & i & = & 0 & \bmod 2
\end{array}$$

Les équations (7) à (12) sont redondantes (elles peuvent être obtenues à partir des équations (1) à (6)), on peut donc en faire abstraction. La résolution mène à :

$$\begin{array}{lrcrl}
(1) & g & = & 1 & \bmod 2\\
(2) & h & = & 0 & \bmod 2\\
(3) & i & = & 0 & \bmod 2\\
(4) & a & = & 0 & \bmod 2\\
(5) & b & = & 1 & \bmod 2\\
(6) & c & = & 1 & \bmod 2
\end{array}$$

On obtient donc aucune condition sur d, e et f qui pourront donc prendre plusieurs valeurs.
Résolvons maintenant le système dans $\mathbb{Z}/13\mathbb{Z}$:

$$\begin{array}{lrcrl}
(1) & 2a+d & = & 7 & \bmod 13\\
(2) & 2b+e & = & 8 & \bmod 13\\
(3) & 2c+f & = & 0 & \bmod 13\\
(4) & 8a+4d+4g & = & 4 & \bmod 13\\
(5) & 8b+4e+4h & = & 4 & \bmod 13\\
(6) & 8c+4f+4i & = & 6 & \bmod 13\\
(7) & 5a+6g & = & 0 & \bmod 13\\
(8) & 5b+6h & = & 7 & \bmod 13\\
(9) & 5c+6i & = & 11 & \bmod 13\\
(10) & 8a+d & = & 6 & \bmod 13\\
(11) & 8b+e & = & 7 & \bmod 13\\
(12) & 8c+f & = & 5 & \bmod 13
\end{array}$$

En effectuant (10) − (1), (11) − (2) et (12) − (3), on obtient :

$$\begin{aligned} 6a &= 12 \mod 13 \\ 6b &= 12 \mod 13 \\ 6c &= 5 \mod 13 \end{aligned}$$

Soit on détermine de façon intuitive ou avec l'algorithme d'Euclide étendu que l'inverse de 6 dans $\mathbb{Z}/13\mathbb{Z}$ est 11 ; ce qui permet alors de trouver a, b et c. Soit on trouve intuitivement une solution à chaque équation (qui est nécessairement unique car 6 admet un inverse). Ainsi on trouve a, b et c puis, en réinjectant leurs valeurs dans (1), (2) et (3), on détermine d, e et f :

$$\begin{aligned} a &= 2 \mod 13 \\ b &= 2 \mod 13 \\ c &= 3 \mod 13 \\ d &= 3 \mod 13 \\ e &= 4 \mod 13 \\ f &= 7 \mod 13 \end{aligned}$$

On réinjecte ensuite dans chaque équation pour trouver g, h et i, puis pour vérifier que nos solutions vérifient toutes les équations.

$$\begin{aligned} (1) \quad & 7 = 7 \mod 13 \\ (2) \quad & 8 = 8 \mod 13 \\ (3) \quad & 0 = 0 \mod 13 \\ (4) \quad & 2+4g = 4 \mod 13 \\ (5) \quad & 6+4h = 4 \mod 13 \\ (6) \quad & 0+4i = 6 \mod 13 \\ (7) \quad & 10+6g = 0 \mod 13 \\ (8) \quad & 10+6h = 7 \mod 13 \\ (9) \quad & 2+6i = 11 \mod 13 \\ (10) \quad & 6 = 6 \mod 13 \\ (11) \quad & 7 = 7 \mod 13 \\ (12) \quad & 13 = 5 \mod 13 \end{aligned}$$

On en déduit le système que doivent vérifier les valeurs de g, h et i :

$$\begin{aligned} 4g &= 2 \mod 13 \\ 4h &= 11 \mod 13 \\ 4i &= 6 \mod 13 \\ 6g &= 3 \mod 13 \\ 6h &= 10 \mod 13 \\ 6i &= 9 \mod 13 \end{aligned}$$

Ainsi les solutions sont $g = 7$, $h = 6$ et $i = 8$.
Plutôt que d'exploiter le théorème des restes chinois pour reconstruire la solution dans $\mathbb{Z}/26\mathbb{Z}$, on va procéder par énumération d'après les résultats obtenus modulo 13, puis on retirera les solutions impossibles au vu des résultats obtenus modulo 2.

$$\begin{aligned} a &= 2 \text{ ou } 15 \mod 26 \\ b &= 2 \text{ ou } 15 \mod 26 \\ c &= 3 \text{ ou } 16 \mod 26 \\ d &= 3 \text{ ou } 16 \mod 26 \\ e &= 4 \text{ ou } 17 \mod 26 \\ f &= 7 \text{ ou } 20 \mod 26 \\ g &= 7 \text{ ou } 20 \mod 26 \\ h &= 6 \text{ ou } 19 \mod 26 \\ i &= 8 \text{ ou } 21 \mod 26 \end{aligned}$$

Finalement les matrices de taille 3 permettant le chiffrement de l'énoncé sont :

$$K = \begin{pmatrix} 2 & 15 & 3 \\ 3|16 & 4|17 & 7|20 \\ 7 & 6 & 8 \end{pmatrix}$$

Malheureusement ces clés ne peuvent pas convenir pour un algorithme cryptographique car elles ne sont pas inversibles... Pour s'en convaincre il suffirait de calculer leurs déterminants et de constater qu'ils ne sont pas inversibles dans $\mathbb{Z}/26\mathbb{Z}$ (car ils ne sont pas premiers avec 26).

$\underline{m = 4}$ Dans ces conditions, la matrice est de la forme :

$$K = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix}$$

Les équations obtenues sont les suivantes :

$$\begin{array}{lrcrl}
(1) & 2a + 14e + 13i + 21m & = & 7 & \mod 26 \\
(2) & 2b + 14f + 13j + 21n & = & 8 & \mod 26 \\
(3) & 2c + 14g + 13k + 21o & = & 0 & \mod 26 \\
(4) & 2d + 14h + 13l + 21p & = & 17 & \mod 26 \\
(5) & 4a + 17e + 18i + 0m & = & 17 & \mod 26 \\
(6) & 4b + 17f + 18j + 0n & = & 19 & \mod 26 \\
(7) & 4c + 17g + 18k + 0o & = & 13 & \mod 26 \\
(8) & 4d + 17h + 18l + 0p & = & 20 & \mod 26 \\
(9) & 19a + 8e + 14i + 13m & = & 24 & \mod 26 \\
(10) & 19b + 8f + 14j + 13n & = & 19 & \mod 26 \\
(11) & 19c + 8g + 14k + 13o & = & 20 & \mod 26 \\
(12) & 19d + 8h + 14l + 13p & = & 18 & \mod 26
\end{array}$$

Pour faciliter les calculs on passe les termes constants à gauche :

$$\begin{array}{rcrl}
2a + 14e + 13i + 21m + 19 & = & 0 & \mod 26 \\
2b + 14f + 13j + 21n + 18 & = & 0 & \mod 26 \\
2c + 14g + 13k + 21o & = & 0 & \mod 26 \\
2d + 14h + 13l + 21p + 9 & = & 0 & \mod 26 \\
4a + 17e + 18i + 9 & = & 0 & \mod 26 \\
4b + 17f + 18j + 7 & = & 0 & \mod 26 \\
4c + 17g + 18k + 13 & = & 0 & \mod 26 \\
4d + 17h + 18l + 6 & = & 0 & \mod 26 \\
19a + 8e + 14i + 13m + 2 & = & 0 & \mod 26 \\
19b + 8f + 14j + 13n + 7 & = & 0 & \mod 26 \\
19c + 8g + 14k + 13o + 6 & = & 0 & \mod 26 \\
19d + 8h + 14l + 13p + 8 & = & 0 & \mod 26
\end{array}$$

Isolons d'abord les termes en m, n, o, p dans les équations (1) à (4) les termes en a, b, c, d dans les équations (9) à (12). On obtient ainsi :

$$\begin{aligned} m &= 16a + 8e + 13i + 9 \mod 26 \\ n &= 16b + 8f + 13j + 14 \mod 26 \\ o &= 16c + 8g + 13k \mod 26 \\ p &= 16d + 8h + 13l + 7 \mod 26 \\ e &= 12a + 2i + 1 \mod 26 \\ f &= 12b + 2j + 21 \mod 26 \\ g &= 12c + 2k + 13 \mod 26 \\ h &= 12d + 2l + 18 \mod 26 \\ a &= 16e + 2i + 13m + 4 \mod 26 \\ b &= 16f + 2j + 13n + 1 \mod 26 \\ c &= 16g + 2k + 13o + 12 \mod 26 \\ d &= 16h + 2l + 13p + 16 \mod 26 \end{aligned}$$

On reporte ensuite les valeurs de m, n, o, p dans les équations (9) à (12) :

$$\begin{aligned} a &= 16e + 15i + 17 \mod 26 \\ b &= 16f + 15j + 1 \mod 26 \\ c &= 16g + 15k + 12 \mod 26 \\ d &= 16h + 15l + 3 \mod 26 \end{aligned}$$

Puis en reportant les valeurs de a, b, c, d dans les équations (5) à (8) on aboutit à :

$$\begin{aligned} 3e &= 1 \mod 26 \\ 3f &= 15 \mod 26 \\ 3g &= 17 \mod 26 \\ 3h &= 3 \mod 26 \end{aligned}$$

Ce qui permet de déduire les valeurs de e, f, g, h (en utilisant le fait que l'inverse de 3 est 9 modulo 26) :

$$\begin{aligned} e &= 9 \mod 26 \\ f &= 5 \mod 26 \\ g &= 23 \mod 26 \\ h &= 20 \mod 26 \end{aligned}$$

Avec ces valeurs, le système devient :

$$\begin{aligned}
10a + 13i + m + 23 &= 0 \mod 26 \\
10b + 13j + n + 24 &= 0 \mod 26 \\
10c + 13k + o + 24 &= 0 \mod 26 \\
10d + 13l + p + 15 &= 0 \mod 26 \\
4a + 18i + 6 &= 0 \mod 26 \\
4b + 18j + 14 &= 0 \mod 26 \\
4c + 18k + 14 &= 0 \mod 26 \\
4d + 18l + 8 &= 0 \mod 26 \\
a + 24i + 13m + 8 &= 0 \mod 26 \\
b + 24j + 13n + 23 &= 0 \mod 26 \\
c + 24k + 13o + 10 &= 0 \mod 26 \\
d + 24l + 13p + 2 &= 0 \mod 26
\end{aligned}$$

En utilisant les équations (9) à (12) pour éliminer les contributions de a, b, c, d dans les équations (1) à (4), on obtient :

$$\begin{aligned}
m &= 19i + 5 \mod 26 \\
n &= 19j + 24 \mod 26 \\
o &= 19k + 24 \mod 26 \\
p &= 19l + 5 \mod 26
\end{aligned}$$

En remplaçant ces valeurs dans les équations (9) à (12), on obtient :

$$\begin{aligned}
a &= 15i + 5 \mod 26 \\
b &= 15j + 3 \mod 26 \\
c &= 15k + 16 \mod 26 \\
d &= 15l + 11 \mod 26
\end{aligned}$$

La matrice K est donc de la forme suivante :

$$K = \begin{pmatrix} 15i+5 & 15j+3 & 15k+16 & 15l+11 \\ 9 & 5 & 23 & 20 \\ i & j & k & l \\ 19i+5 & 19j+24 & 19k+24 & 19l+5 \end{pmatrix}$$

Donc l'ensemble des solutions est l'ensemble des matrices de cette forme qui sont inversibles. Le déterminant de cette matrice est :

$$3i + 16j + 17k + 5l$$

Il suffit alors de choisir les valeurs de i, j, k, l pour lesquelles ce déterminant est inversible modulo 26. Par exemple $i = j = k = l = 1$ donnent $3 + 16 + 17 + 5 = 41 = 15 \mod 26$ qui est inversible. Conclusion $m = 4$ convient. À titre d'exemple vérifier que le chiffrement est correct.

2.3 Chiffrement par transposition

$\mathcal{A}$ est l'alphabet romain et $\mathcal{B} = \mathcal{A}$. Soit σ une permutation (*i.e.* une bijection) sur $\{1, \ldots, n\}$ où n est la longueur du clair. L'opération de chiffrement d'un message $m = m_1 \ldots m_n$ est :

$$c = \mathcal{E}_\sigma(m) = m_{\sigma(1)} \ldots m_{\sigma(n)}.$$

La clé secrète est σ.

2.3.1 Question 1

Énoncé Montrer comment, connaissant cette clé, on peut déchiffrer.

Correction σ est connu donc on peut facilement calculer σ^{-1} la transposition inverse.
Calculons $\mathcal{E}_{\sigma^{-1}}(c)$:
Par définition de c on obtient :

$$c = \mathcal{E}_{\sigma(n)} = m_{\sigma(1)}...m_{\sigma(n)}$$

. Si on applique la transportation inversa à c :

$$\mathcal{E}_{\sigma^{-1}}(c) = c_{\sigma^{-1}(1)}...c_{\sigma^{-1}(n)} = m_{\sigma^{-1}(\sigma(1))}..m_{\sigma^{-1}(\sigma(n))} = m_1...m_n$$

Ainsi on obtient : $\boxed{\mathcal{E}_{\sigma^{-1}}(c) = m}$

2.3.2 Question 2

Énoncé Montrer qu'il existe une matrice A_σ telle que :

$$c = m \times A_\sigma.$$

Expliciter la matrice A_σ.

Correction Dans la suite, $\mathcal{K}$ désigne l'ensemble des valeurs possibles pour les m_i et c_i.
L'application $\mathcal{E}_\sigma$ définie par :

$$\mathcal{L} \left| \begin{array}{ccc} \mathcal{M}_{1,n}(\mathcal{K}) & \longrightarrow & \mathcal{M}_{1,n}(\mathcal{K}) \\ m & \longmapsto & c = \mathcal{E}_\sigma(m) \end{array} \right.$$

est une application linéaire. Elle admet donc une représentation matricielle.

La matrice A_σ est une matrice de permutation. Donc son terme général est $\delta_{i,\sigma(j)}$ (où $\delta_{i,\sigma(j)}$ est le symbole de Kronecker) :
Notons m le vecteur de $\S\mathcal{M}_{1,n}(\mathcal{K})$.

$$m \times A_\sigma = \left(\sum_{i=1}^{n} (A_\sigma)_{(i,j)} m_i \right)_{1 \leq j \leq n} = \left(\sum_{i=1}^{n} \delta_{i,\sigma(j)} m_i \right)_{1 \leq j \leq n} = \left(m_{\sigma(j)} \right)_{1 \leq j \leq n} = c$$

Conclusion, on a bien : $\boxed{A_\sigma = (\delta_{i,\sigma(j)})_{1 \leq i,j \leq n}}$

2.3.3 Question 3

Énoncé Supposons que l'attaquant dispose de n paires clair-chiffré. Montrer qu'il peut, avec une probabilité non négligeable, retrouver la clé.

Correction L'idée est très simple, en supposant que l'on dispose de n paires clair-chiffré ($n \geq 1$), on peut constituer un vecteur de textes clairs $M = \begin{bmatrix} m_1 & m_2 & ... & m_{n-1} & m_n \end{bmatrix}$ et un vecteur de chiffrés
$C = \begin{bmatrix} c_1 & c_2 & ... & c_{n-1} & c_n \end{bmatrix}$. Ces deux matrices sont liées par la relation suivante :

$$C = AM.$$

Si la matrice M est inversible, on peut écrire $A = C \cdot M^{-1}$.
La solution de cet exercice réside dans le fait que la probabilité que cette matrice soit inversible est non négligeable.

En fait, on peut facilement dénombrer le nombre de matrices inversibles à coefficients dans un corps fini. Ramenons nous alors à ce cas particulier. Pour cela on va rajouter quelques caractères aux 26 lettres de l'alphabet de sorte que l'on considérera les matrices à coefficients dans $\mathbb{Z}/29\mathbb{Z}$. On va compléter l'alphabet romain classique avec les caractères ",", "." et l'espace. Il nous faut maintenant dénombrer les matrices inversibles en s'appuyant sur le fait que si les colonnes d'une matrice carrée forment une famille libre, alors la matrice est inversible.
Ainsi en prenant $p = 29$,

- Pour le premier vecteur v_1, on a $(p^n - 1)$ choix : n'importe quel vecteur convient sauf le vecteur nul ;
- Pour le deuxième vecteur v_2, on a $(p^n - p)$ choix : on choisit les coefficients du second vecteur en faisant attention à ne pas prendre un vecteur qui s'exprime comme combinaison linéaire du précédent. En d'autre terme on doit exclure les vecteurs de la forme $\lambda_1 \cdot v_1$ avec λ_1 entre 0 et $p - 1$.
- ...
- Pour le i vecteur v_i, on a $(p^n - p^{i-1})$ choix : on choisit ses coefficient en faisant attention à ne pas prendre un vecteur s'exprimant comme combinaison linéaire des précédents. En d'autre terme on doit exclure les vecteurs de la forme $\sum_{k=1}^{i-1} \lambda_k \cdot v_k$ avec chaque λ_k entre 0 et $p - 1$.
- ...
- Pour le n vecteur v_n, on a $(p^n - p^{n-1})$ choix : on choisit ses coefficient en faisant attention à ne pas prendre un vecteur s'exprimant comme combinaison linéaire des précédents. En d'autre terme on doit exclure les vecteurs de la forme $\sum_{k=1}^{n-1} \lambda_k \cdot v_k$ avec chaque λ_k entre 0 et $p - 1$.

Au total, le nombre de matrices inversibles est donc : $\prod_{k=1}^{n}(p^n - p^{k-1})$.
La probabilité d'être inversible pour une matrice de $\mathcal{M}_n(\mathbb{Z}/p\mathbb{Z})$ est donc :

$$\mathcal{P}(p,n) = \frac{\prod_{k=1}^{n}(p^n - p^{k-1})}{(p^n)^n} = \frac{\prod_{k=1}^{n}(p^n - p^{k-1})}{(p^{n^2})}.$$

En simplifiant l'écriture précédente, on a :

$$\mathcal{P}(p,n) = \prod_{k=0}^{n-1}\left(1 - \frac{1}{p^{n-k}}\right)$$

Cette probabilité est non négligeable. Vous pouvez faire le calcul dans le cas qui nous intéresse pour vous en convaincre.

2.3.4 Question 4

Énoncé Calculer la probabilité de succès de l'attaque, ainsi que sa complexité.

Correction Dans la question précédente, on a déjà explicitement effectué le calcul de la probabilité de succès de l'attaque. La complexité de l'attaque est le fruit d'une inversion et d'un produit de matrices. Considérons un produit de matrices carrées de taille n : $C = A \times B$. Par définition du produit matriciel, $c_{i,j} = \sum_{k=1}^{n} a_{i,k}b_{k,j}$ ce qui fait n multiplications et n additions pour déterminer un seul des coefficients du produit. Le temps de calcul d'une addition étant négligeable devant le temps de calcul d'une multiplication, on peut considérer que le calcul d'un coefficient du produit se fait en $\mathcal{O}(n)$ multiplications. Il y a n^2 coefficients à calculer et ainsi la complexité du produit matriciel est en $\mathcal{O}(n^3)$ multiplications. De même la complexité de l'inversion matricielle par la méthode de Gauss est en $\mathcal{O}(n^2)$ multiplications.
Ainsi la complexité de l'attaque est en $\mathcal{O}(n^3)$ multiplications.
Cet exercice montre les limites d'un chiffrement linéaire : le texte clair et le texte chiffré sont "trop" dépendants l'un de l'autre.

3 Feuille d'Exercices 3

3.1 Exercice 1

3.1.1 Question 1

Énoncé : Montrer que $\text{DES}_{\overline{K}}(\overline{x}) = \overline{\text{DES}_K(x)}$ où on désigne par $\overline{z}$ le *complément* bit à bit de z, pour toute chaîne de bits z.

Correction Avant de commencer à résoudre l'exercice, rappelons le fonctionnement de l'algorithme DES :

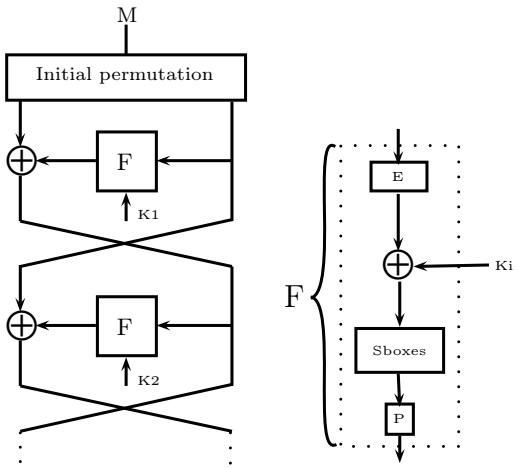


Figure 3 – L'algorithme DES

Soit $x \in \{0,1\}^{64}, K \in \{0,1\}^{56}$ Montrons que $DES_{\overline{K}}(\overline{x}) = \overline{DES_K(x)}$
Pour cela on utilisera trois propriétés :

1. $\overline{a} \oplus b = \overline{a \oplus b}$
2. $\overline{a} \oplus \overline{b} = a \oplus b$
3. $P : E \to E$ permutation $\Rightarrow (\forall x \in E, P(\overline{x}) = \overline{P(x)})$

La démonstration de ces propriétés est laissée en exercice complémentaire. Ces propriétés doivent être retenues.

Montrons par récurrence sur $i = 0..15$ que $L_i(\overline{x}, \overline{K}) = \overline{L_i(x, K)}$ et $R_i(\overline{x}, \overline{K}) = \overline{R_i(x, K)}$:

— Au rang $i = 0$: IP est une permutation donc d'après la propriété 3, $IP(\overline{x}) = \overline{IP(x)}$. Ainsi,

$$L_0(\overline{x}, \overline{K}) = \overline{L_0(x, K)} \text{ et } R_0(\overline{x}, \overline{K}) = \overline{R_0(x, K)}$$

Ce qui valide l'hypothèse au rang $i = 0$.

— Supposons les deux égalités vérifiées au rang $i \leq 15$.

— Montrons que c'est aussi vrai pour $i + 1$.
Par définition de l'algorithme DES, on a :

$$\begin{aligned} L_{i+1}(\overline{x}, \overline{K}) &= R_i(\overline{x}, \overline{K}) \\ R_{i+1}(\overline{x}, \overline{K}) &= L_i(\overline{x}, \overline{K}) \oplus f(R_i(\overline{x}, \overline{K}), K_{i+1}(\overline{K})) \end{aligned}$$

Or par hypothèse de récurrence : $L_i(\overline{x}, \overline{K}) = \overline{L_i(x, K)}$ et $R_i(\overline{x}, \overline{K}) = \overline{R_i(x, K)}$.
Donc $L_{i+1}(\overline{x}, \overline{K}) = R_i(\overline{x}, \overline{K}) = \overline{R_i(x, K)}$. Ainsi, par définition :

$$L_{i+1}(\overline{x}, \overline{K}) = \overline{L_{i+1}(x, K)}.$$

Les sous-clés du DES sont une extraction des bits de la clé, donc $K_{i+1}(\overline{K}) = \overline{K_{i+1}(K)}$. D'où :

$$f(R_i(\overline{x},\overline{K}), K_{i+1}(\overline{K})) = f(\overline{R_i(x,K)}, \overline{K_{i+1}(K)})$$

Or par définition de l'algorithme DES :

$$f(\overline{R_i(x,K)}, \overline{K_{i+1}(K)}) = P(\text{Sboxes}(E(\overline{R_i(x,K)}) \oplus \overline{K_{i+1}(K)}))$$

Par définition de E : $E(\overline{x}) = \overline{E(x)}$ donc,

$$f(\overline{R_i(x,K)}, \overline{K_{i+1}(K)}) = P(\text{Sboxes}(E(\overline{R_i(x,K)}) \oplus \overline{K_{i+1}(K)}))$$

Ainsi, d'après la propriété 2 :

$$f(\overline{R_i(x,K)}, \overline{K_{i+1}(K)}) = P(\text{Sboxes}(E(R_i(x,K)) \oplus K_{i+1}(K))) = f(R_i(x,K), K_{i+1}(K))$$

Par définition,

$$R_{i+1}(\overline{x},\overline{K}) = L_i(\overline{x},\overline{K}) \oplus f(R_i(\overline{x},\overline{K}), K_{i+1}(\overline{K}))$$

Donc en utilisant l'hypothèse de récurrence :

$$R_{i+1}(\overline{x},\overline{K}) = \overline{L_i(x,K)} \oplus f(R_i(\overline{x},\overline{K}), K_{i+1}(\overline{K}))$$

Or d'après les calculs précédents, $f(R_i(\overline{x},\overline{K}), K_{i+1}(\overline{K})) = f(R_i(x,K), K_{i+1}(K))$. Donc,

$$R_{i+1}(\overline{x},\overline{K}) = \overline{L_i(x,K)} \oplus f(R_i(x,K), K_{i+1}(K))$$

Ce qui, d'après la propriété 1, se réécrit :

$$R_{i+1}(\overline{x},\overline{K}) = \overline{L_i(x,K) \oplus f(R_i(x,K), K_{i+1}(K))}$$

Par définition, $R_{i+1}(x,K) = L_i(x,K) \oplus f(R_i(x,K), K_{i+1}(K))$. Finalement,

$$R_{i+1}(\overline{x},\overline{K}) = \overline{R_{i+1}(x,K)}$$

Ce qui achève la preuve par récurrence.

Il nous faut traiter la fin de l'algorithme DES. En effet, le rang 16 diffère des précédents : on échange R_{16} et L_{16}. Par définition :

$$L_{16}(\overline{x},\overline{K}) = R_{15}(\overline{x},\overline{K}) = \overline{R_{15}(x,K)} = \overline{R_{16}(x,K)}$$
$$R_{16}(\overline{x},\overline{K}) = L_{15}(\overline{x},\overline{K}) \oplus f(R_{15}(\overline{x},\overline{K}), K_{16}(\overline{K}))$$

En appliquant le même raisonnement que précédemment :

$$L_{16}(\overline{x},\overline{K}) = \overline{L_{16}(x,K)}$$

La dernière étape étant une permutation, on aboutit au résultat attendu : $DES_{\overline{K}}(x) = \overline{DES_K(x)}$.

3.1.2 Question 2

Énoncé : En déduire une attaque par recherche exhaustive sur la clé du DES, dont la complexité soit en moyenne de 2^{54} chiffrements DES.

Correction On rappelle ici que dans le cas de l'algorithme DES, les clés ont une taille de 56 bits. Il y a ainsi 2^{56} clés possibles.
L'attaque est une attaque à texte claire connu i.e. on dispose d'une paire texte clair x et le texte chiffré correspondant $DES_k(x)$. On suppose également que l'on dispose de la paire ($\overline{x}$,$DES_k(\overline{x})$). L'attaque est alors menée à l'aide de l'algorithme suivant :

Proposition Cet algorithme exécute en moyenne 2^{54} chiffrements DES.

Démonstration En moyenne, on effectue que la moitié de la boucle for. Mais de plus, à chaque tour de boucle, on teste deux clés : k et $\overline{k}$ à l'aide d'un seul chiffrement DES. Ainsi, il y a en moyenne $(2^{56}/2)/2 = 2^{54}$ chiffrements DES. ■

Algorithme 1 : Attaque exhaustive

Input: $C = DES_k(x),\ C' = \overline{DES_k(\overline{x})} = DES_{\overline{k}}(x)$
Output: $k/DES_k(x)$

```
for k ∈ K do
  DES_tmp = DES_k(x)
  if DES_tmp = C then
    return k
  end if
  if DES_tmp = C' then
    return k̄
  end if
end for
```

3.2 Exercice 2

3.2.1 Question 1

Énoncé On dit qu'une clé K du DES est *faible* si DES_K est une involution. Trouver quatre clés faibles pour le DES.

Correction Afin de résoudre cet exercice, il convient de revenir sur le fonctionnement de l'algorithme *key_schedule* de DES (cf cours et feuille distribuée en TD). Il faut également se rappeler que le déchiffrement d'un message chiffré $DES_K(m)$ se fait en exécutant l'algorithme DES mais en utilisant les sous-clés en sens inverse. Il est donc intéressant d'écrire l'exécution de l'algorithme *key_schedule* pour comprendre les relations qui vont lier les clés dites faibles et celles qui sont semi-faibles. On va se concentrer sur les blocs C_i c'est-à-dire les bits de poids faibles de la clé après la permutation `PC1`.
On répresente les bits du bloc C_i lors de l'exécution de l'algorithme de dérivation des clés :

valeur de décalage	sous-clé	Valeur de la sous-clé
1	K_1	$k_2, ..., k_{28}, k_1$
1	K_2	$k_3, ..., k_{28}, k_1, k_2$
2	K_3	$k_5, ..., k_{28}, k_1, ..., k_4$
2	K_4	$k_7, ..., k_{28}, k_1, ..., k_6$
2	K_5	$k_9, ..., k_{28}, k_1, ..., k_8$
2	K_6	$k_{11}, ..., k_{28}, k_1, ..., k_{10}$
2	K_7	$k_{13}, ..., k_{28}, k_1, ..., k_{12}$
2	K_8	$k_{15}, ..., k_{28}, k_1, ..., k_{14}$
1	K_9	$k_{16}, ..., k_{28}, k_1, ..., k_{15}$
2	K_{10}	$k_{18}, ..., k_{28}, k_1, ..., k_{17}$
2	K_{11}	$k_{10}, ..., k_{28}, k_1, ..., k_{19}$
2	K_{12}	$k_{22}, ..., k_{28}, k_1, ..., k_{21}$
2	K_{13}	$k_{24}, ..., k_{28}, k_1, ..., k_{23}$
2	K_{14}	$k_{26}, ..., k_{28}, k_1, ..., k_{15}$
2	K_{15}	$k_{28}, k_1, ..., k_{27}$
1	K_{16}	$k_1, ..., k_{28}$

Muni de ces informations on peut commencer à résoudre la première question. Avant de passer à la résolution concrète, analysons l'énoncé : dire que DES est une involution c'est dire que l'opération de chiffrement est la même que l'opération de déchiffrement. Ce qui équivaut concrètement à dire que les sous-clés utilisées pour le déchiffrement et celles utilisées pour le chiffrement sont les mêmes à chaque étape. Ce qui se traduit de la façon suivante :

$$K_1 = K_{16}, K_2 = K_{15}, ..K_8 = K_9$$

L'égalité $K_1 = K_{16}$ se réécrit :

$$[k_1, ..., k_{28}] = [k_2, ..., k_{28}, k_1]$$

On peut donc en déduire que

$$k_1 = k_{28} \text{ et } \forall i \in \{1..27\},\ k_j = k_{j+1}$$

Si les équations précédentes sont satisfaites, par construction, on obtiendra une clé faible.
L'analyse de ces deux équations est assez simple : si on donne une valeur à k_1, alors on détermine l'ensemble du bloc C_0. On a donc deux choix possibles 1 ou 0 (car les k_i sont des bits). Ainsi

$$C_0 = [0, ..., 0] \text{ ou } C_0 = [1, ..., 1]$$

Un raisonnement analogue permet de montrer que

$$D_0 = [0, ..., 0] \text{ ou } D_0 = [1, ..., 1]$$

Comme les deux blocs fonctionne de manière indépendante, on a donc les quatre possibilités suivantes :

$$C_0 = [0, ..., 0] \text{ et } D_0 = [0, ..., 0]$$

$$C_0 = [0, ..., 0] \text{ et } D_0 = [1, ..., 1]$$

$$C_0 = [1, ..., 1] \text{ et } D_0 = [0, ..., 0]$$

$$C_0 = [1, ..., 1] \text{ et } D_0 = [1, ..., 1]$$

Pour retrouver les "vraies" clés faibles, il faut encore inverser la permutation `PC1` sur les blocs C_0 et D_0. Cet inversion est laissée au lecteur.

3.2.2 Question 2

Énoncé On dit qu'une clé K du DES est *semi-faible* si elle n'est pas faible et s'il existe K' tel que

$$\text{DES}_K^{-1} = \text{DES}_{K'}.$$

Trouver six paires (K, K') de clés semi-faibles pour le DES.

Correction Dans le cas des clés semi-faibles, en reprenant la démarche précédente, on aboutit à une relation du type :

$$[k'_1, ..., k'_{28}] = [k_2, ..., k_{28}, k_1]$$

Ce qui se réécrit de la façon suivante :

$$k'_{28} = k_1 \text{ et } \forall i \in \{1..27\},\ k'_j = k_{j+1}$$

Ces équations ne permettent pas de conclure. Il nous faut rajouter une condition supplémentaire : $K'_{15} = K_2$:

$$[k'_{28}, k'_1, ..., k'_{27}] = [k_3, ..., k_{28}, k_1, k_2]$$

Afin de mieux comprendre ce qu'implique l'égalité précédente détaillons les premiers bits des deux sous-clés :

$$[k'_{28}, k'_1, k'_2, k'_3, k'_4, ...] = [k_3, k_4, k_5, k_6, k_7, ...]$$

Ce qui se réécrit de la façon suivante :

$$k'_{28} = k_3,\ k'_{27} = k_2,\ k'_{26} = k_1, \quad \text{et } \forall j \in \{1..25\} k'_j = k_{j+3}$$

En reprenant ce qui précède on aboutit donc à des relations du type :

$$\begin{array}{lccccc} \forall i \in \{0..12\} & k'_{2i+1} & = & k_{2i+2} & = & k_{2i+4} \\ \forall j \in \{1..12\} & k'_{2j} & = & k_{2j+1} & = & k_{2j+3} \\ & k'_{26} & = & k_1 & = & k_3 \\ & k'_{27} & = & k_2 & = & k_4 \\ & k'_{28} & = & k_3 & = & k_1 \end{array}$$

Ce se traduit sur K par le fait que les coefficients doivent avoir la configuration suivante :

$$\begin{array}{llcl} \forall i \in \{1..14\} & k_{2i} & = & k_{2i+2} \\ \forall j \in \{0..12\} & k_{2j+1} & = & k_{2j+3} \end{array}$$

La fin du raisonnement est laissée au lecteur. Les calculs ayant été achevés, il reste à expliquer comment on construit les six paires de clés semi-faibles.

4 Feuille d'Exercices 4

4.1 Exercice 1 : Chiffrement double : l'attaque "par le milieu" ou "meet-in-the-middle"

Supposons que l'on essaye d'améliorer un système de chiffrement $(\mathcal{E}_K)_{K\in\mathcal{K}}$ dont les clés trop courtes n'interdisent pas la recherche exhaustive (on peut penser au DES) en chiffrant deux fois :

$$M \mapsto C = \mathcal{E}_{K_2}(\mathcal{E}_{K_1}(M)).$$

On double donc la longueur de la clé secrète. On considère l'attaque à clair connu suivante : Charlie connaît M et $C = \mathcal{E}_{K_2}(\mathcal{E}_{K_1}(M))$ et cherche K_1 et K_2. Il crée deux listes

$$\mathcal{L}_M = (\mathcal{E}_K(M))_{K\in\mathcal{K}} \qquad \text{et} \qquad \mathcal{L}_C = (\mathcal{E}_K^{-1}(C))_{K\in\mathcal{K}}.$$

4.1.1 Question 1

Énnoncé Expliciter une attaque pour retrouver la clé (K_1, K_2). On commencera par chercher un élément commun aux deux listes.

Correction L'attaque consiste en la recherche d'éléments commun aux deux listes. En effet, soit l un tel élément. Par définition, on a :

- $l \in \mathcal{L}_M$ donc $\exists \mathcal{K}_1 \in \mathcal{K}$ telle que $l = \mathcal{E}_{\mathcal{K}_1}(M)$
- $l \in \mathcal{L}_C$ donc $\exists \mathcal{K}_2 \in \mathcal{K}$ telle que $l = \mathcal{E}_{\mathcal{K}_2}^{-1}(C)$

Il nous faut alors tester si $C = \mathcal{E}_{K_2}(\mathcal{E}_{K_1}(M))$. Si c'est le cas, alors on a réussi à casser le système de chiffrement. Sinon, on cherche un autre élément commun aux deux listes.

4.1.2 Question 2

Énnoncé Vérifier que le nombre de comparaisons nécessaire à la découverte d'un élément commun aux deux listes est de l'ordre de

$$(\#\mathcal{K}) \cdot \ln(\#\mathcal{K}).$$

Correction En fait, la résolution de cet exercice nécessite de faire appel à un algorithme de recherche rapide : la recherche dichotomique ainsi, le temps de recherche sera logarithmique. Cependant pour pouvoir utiliser cet algorithme, il faut que les listes soient triées. C'est le tri qui est le facteur limitant ici car les algorithmes de tris classiques s'exécutent en temps polynomial. Il faut donc utiliser un algorithme de tri rapide, par exemple le *quick_ sort*. La description de cet algorithme est proposée ci-après :

Algorithme 2 : quick_sort

```
quick_sort(A, p, r) :
if p < r then
  q ← partition(A, p, r)
  quick_sort(A, p, q − 1)
  quick_sort(A, q + 1, r)
end if


Partition(A, p, r) :
x ← A[r]
i ← p − 1
for j = p to r − 1 do
  if A[j] ≤ x then
    i ← i + 1
    A[i] ⟷ A[j]
  end if
  A[i + 1] ⟷ A[r]
end for
return i + 1
```

La complexité de l'algorithme *quick_ sort* est en $\mathcal{O}(n\ln(n))$. Cette affirmation doit être démontrée et d'autre part, il vous faut également vérifier que cet algorithme effectue bien un tri du tableau A en entrée.
On s'est donc muni d'un algorithme qui effectue le tri d'une liste (ou d'un tableau) en $\mathcal{O}(n\ln(n))$ opérations. En supposant que les listes soient triées, on effectue alors l'algorithme suivant pour trouver un élément commun aux deux listes :

Algorithme 3 : Liste_Elements_Communs

```
i ← 1  //Indice des éléments dans la première liste
j ← 1  //Indice des éléments dans la deuxième liste
L ← ∅ //Liste des éléments communs
while i ≤ #K and j ≤ #K  do
if L_M > L_C then
  j ← j + 1
end if
if L_M < L_C then
  i ← i + 1
end if
if L_M = L_C then
  L ← L ∪ {L_M}
  i ← i + 1
  j ← j + 1
end if
end while
return L
```

Cet algorithme constitue la liste de tous les éléments communs aux deux listes et s'exécute en temps $\mathcal{O}(\#\mathcal{K})$ en effet, à chaque étape de la boucle on avance nécessairement dans au moins une liste. Conclusion, la procédure limitante en terme d'opérations est celle qui permet le tri des deux listes.
Finalement, on a exposé une méthode pour trouver tous les éléments communs aux deux listes en un temps $\mathcal{O}(\#\mathcal{K}\ln(\#\mathcal{K}))$, vous pouvez vérifier que le nombre de comparaison nécessaires (comme demandé dans la consigne) est du même ordre.

4.1.3 Question 3

Énnoncé De combien de mémoire faut-il disposer pour pouvoir réaliser l'attaque ?

Correction Pour pouvoir faire ces comparaisons, il faut déjà avoir construit les deux listes. Donc en considérant que la taille en bits de la sortie de la fonction de chiffrement est c, il nous faut environ $2 \times c \times \#\mathcal{K}$ bits de mémoire pour mener à bien cette attaque.
Avant de conclure cet exercice, essayons de comprendre ce que cela signifie pour un algorithme de chiffrement simple comme le DES. Dans ce cas là :

— $c = 64$
— $\#\mathcal{K} = 2^{56}$

Au total il faut donc $2 \times 64 \times 2^{56} = 2^1 \times 2^6 \times 2^{56} = 2^{63} = 8.1024^6$ bits de mémoire soit 1024^6 octets $\approx 10^6$ Teraoctets pour mener à bien cette attaque.
C'est donc la "petite" taille de l'ensemble des clés possibles de DES qui le rend vulnérable à ce type d'attaque. Si on considère des systèmes de chiffrement dont la taille de clé est plus importante, on arrive très vite à des tailles qui excluent la mise en pratique de cette attaque. Vous pouvez remarquer que l'on a pas considéré le temps nécessaire au calcul des listes.
Cet exercice est instructif dans la mesure où l'on se rend compte que le double chiffrement offre une sécurité qui n'est que deux fois plus importante que le chiffrement simple. Le double chiffrement n'a donc calculatoirement aucun impact sur la sécurité d'un système cryptographique. Pour véritablement améliorer la sécurité, il faut au moins trois chiffrement. Ceci explique que dans la pratique le double DES ne soit pas utilisé au profit du triple DES.

4.2 Exercice 2 : Compromis Temps Mémoire

Soit E une primitive de chiffrement symétrique parfaite, c'est-à-dire :

$$E_{K_1}(M) = E_{K_2}(M) \Rightarrow K_1 = K_2.$$

Dans la suite, on suppose que la taille de clé utilisée par cette primitive est n bits.
On va faire une cryptanalyse générique de cette primitive via une attaque à clair connu.

4.2.1 Question 1

Énnoncé Rappeler la définition de "attaque à clair connu" .

Correction L'attaquant connaît un pair (ou plusieurs) de message - chiffré.

4.2.2 Question 2

Énnoncé Donner le coût algorithmique moyen d'une recherche exhaustive de la clé.

Correction Pour une recherche exhaustive, on essaie toutes les clés possibles une par une. Donc 2^n clés.
Statistiquement, il faut essayer la moitié de clés avant de trouver la bonne clé. Donc le coût moyen est $\frac{2^n}{2} = 2^{n-1}$.
Dans la suite, on suppose que l'on est capable de faire des "précalculs", i.e., on peut effectuer des calculs avant de mener l'attaque. L'idée étant de précalculer l'ensemble des chiffrements du message clair M connu. L'attaque de recouvrement de la clé nécessitera simplement l'identification de la clé parmis l'ensemble des valeurs précalculées.

4.2.3 Question 3

Énnoncé Donner la meilleure structure de données permettant un stockage et une recherche efficace de la clé dans ces conditions.
Donner en fonction de n l'espace de stockage nécessaire pour effectuer le stockage de tous les chiffrés possibles. Pour une valeur de $n = 256$ (cas de l'AES), donner une valeur approchée de l'espace nécessaire au stockage.

Correction L'idée pour effectuer les précalculs est de calculer le chiffré qui correspond à chaque clé possible (le message étant fixe).
Les éléments que nous avons besoin pour une recherche efficace est donc la clé et les chiffrés (le message étant fixe).
Une fois toutes les paires (clé, chiffré) stockées dans une table, à chaque fois qu'une valeur C est donnée, il suffit de chercher dans cette table une valeur de clé qui correspond à $C = E_K(M)$.
Puisque on calcule un chiffré pour chaque clé possible, le nombre des chiffrés possibles est le même que le nombre des clés possibles, i.e. 2^n. Pour $n = 256$, l'espace nécessaire est 2^256. Ce qui est approximativement est égale à 10^{64} Teraoctet.

4.2.4 Question 4

Énnoncé D'une manière générale, on néglige les précalculs à condition que le temps et l'espace nécessaires à ceux-ci soient "raisonnables". Vous paraît-il raisonnable d'envisager l'attaque décrite à la question 3 ?

Correction 10^{64} Teraoctet est bien supérieure que la mémoire d'un ordinateur commun d'aujourd'hui.
Afin d'attaquer la primitive E, on se propose de procéder comme décrit ci-dessous.
On choisit une longueur t et un nombre aléatoire noté K_1^1 de n bits.
On fixe $SP_1 = K_1^1$, puis on calcule :

$$\begin{array}{rcccccccc}
SP_1 = K_1^1 & \to & K_2^1 = E_{K_1^1}(M) & \to & K_3^1 = E_{K_2^1}(M) & \to & \cdots & \to & E_{K_{t-1}^1}(M) = EP_1 \\
SP_2 = EP_1 = K_1^2 & \to & K_2^2 = E_{K_1^2}(M) & \to & K_3^2 = E_{K_2^2}(M) & \to & \cdots & \to & E_{K_{t-1}^2}(M) = EP_2 \\
\vdots & & & & & & & & \vdots \\
SP_m = EP_{m-1} = K_1^m & \to & K_2^m = E_{K_1^m}(M) & \to & K_3^m = E_{K_2^m}(M) & \to & \cdots & \to & E_{K_{t-1}^m}(M) = EP_m
\end{array}$$

Et on stocke les couples (SP_i, EP_i).

4.2.5 Question 5

Énnoncé

1. Expliquer comment retrouver la clé à la suite de ces précalculs ;
2. Cette attaque fonctionne-t-elle toujours ?

Correction

1. Supposons qu'on obtient un chiffré Y et on veut trouver la clé qui lui correspond.
   On regarde s'il existe i tel que $Y = EP_i$.
   — Si oui, on trouve le SP_i qui correspond à EP_i et on reconstruit la chaine à partir de SP_i jusqu' à la colonne $t-1$ afin d'obtenir K_t^i (i.e. la clé qui nous a permis de calculer EP_i).
   — Si non, au départ de SP_1 on reconstruit la chaîne jusqu' à ce qu'on trouve j et l tels que $Y = E_{K_{j-1}^l}(M)$. La clé est donc K_{j-1}^l.
2. L'efficacité de cette attaque dépend du choix de m et t.

4.2.6 Question 6

Énnoncé La technique précédente n'est pas toujours réalisable. Expliquer pourquoi et proposer une modification permettant de la rendre pratique.

Correction Lors du chiffrement des conditions spéciales sont souvent demandé pour la clé (par exemple sa taille). Quand ces conditions ne sont pas vérifiées, le chiffrement n'est pas réalisable. Pour éviter ces cas, on applique une fonction de transition à la fonction du chiffrement. Donc, au lieu d'appliquer la fonction du chiffrement $E_{K_j^i}(M)$, on applique la fonction $f(K) = R(E_{K_j^i}(M))$ où R est la fonction de transition. La fonction de transition est une fonction bijective, comme une permutation ou ajouter un entier à la fonction E.

Énnoncé D'une manière générale, dans cette cryptanalyse, on utilise la dernière valeur de la chaîne précédente pour commencer la chaîne suivante. En supposant que l'on ne souhaite plus être en mesure de retrouver toutes les clés, on se propose de prendre une nouvelle valeur aléatoire pour chacune des valeurs de début de chaîne :

$$\begin{array}{rcccccccc} SP_1 =_r r_0^1 & \to & r_1^1 & \to & r_2^1 & \to & \cdots & \to & r_t^1 = EP_1 \\ SP_2 =_r r_0^2 & \to & r_1^2 & \to & r_2^2 & \to & \cdots & \to & r_t^2 = EP_2 \\ SP_3 =_r r_0^3 & \to & r_1^3 & \to & r_2^3 & \to & \cdots & \to & r_t^3 = EP_3 \\ \vdots & & & & & & & & \vdots \\ SP_m =_r r_0^m & \to & r_1^m & \to & r_2^m & \to & \cdots & \to & r_t^m = EP_m \end{array}$$

Cette attaque a été étudiée par Martin Hellman en 1980, dans l'article :

> Martin E. Hellman. A cryptanalytic time-memory trade off. *IEEE Transactions on Information Theory*, 26(4) :401–406, 1980. Available at `http://www-ee.stanford.edu/~hellman/publications/36.pdf`.

Il y montre notamment une borne inférieure sur la probabilité Pr de succès d'une telle attaque en fonction de la longueur des chaînes et de leur nombre :

$$Pr \geq \frac{1}{N} \sum_{i=1}^{m} \sum_{j=0}^{t-1} \left[1 - \frac{it}{N}\right]^{j+1}.$$

4.2.7 Question 7

Énnoncé

- Quelle est l'influence de la longueur des chaînes sur la quantité de calculs nécessaire à l'attaque ?
- Quelle est l'influence du nombre de chaînes sur la quantité de mémoire nécessaire à l'attaque ?

4.2.8 Question 8

Énnoncé Quel problème peut maintenant survenir lors de la constitution de la table ?
Proposer une solution à ce problème.

Correction L'efficacité d'une table diminue selon sa taille. Pour contourner ceci, on peut générer plusieurs tables en utilisant une fonction de réduction différente pour chaque table. Au titre indicatif, la probabilité de succès en utilisant l tables est

$$Pr \geq 1 - (1 - (\frac{1}{N} \sum_{i=1}^{m} \sum_{j=0}^{t-1} \left[1 - \frac{it}{N}\right]^{j+1})^l.$$

Dans la pratique, les tables contenant les valeurs stockées sont trop grandes pour être stockées en RAM. Ainsi la conduite d'une attaque requiert un nombre très important d'accés au disque dur. Sachant que le temps d'un accès disque est 1000 fois plus important qu'un accès en RAM, et qu'un accès registre est environ 100 fois plus rapide qu'un accès RAM, il est intéressant de limiter au maximum les accès au disque dur, i.e., de limiter le nombre de recherches dans la table. Pour atteindre cet objectif, Ronald Rivest a suggéré l'utilisation de *"points distingués"*.
Par *point distingué* on désigne des éléments de chaîne ayant une propriété particulière. Par exemple, on pourra s'intéresser aux chiffrés terminés par 10 bits à 0. L'idée est de ne plus stocker des chaînes de taille fixe, on s'arrête au moment où on tombe sur un point distingué.

$$\begin{array}{ll} SP_1 =_r r_0^1 \to r_1^1 \to r_2^1 \to \cdots & \to r_{t_1}^1 = EP_1 \in \{0,1\}^{n-10}0000000000 \\ SP_2 =_r r_0^2 \to r_1^2 \to r_2^2 & \to r_{t_2}^2 = EP_2 \in \{0,1\}^{n-10}0000000000 \\ SP_3 =_r r_0^3 \to r_1^3 \to r_2^3 \to \cdots \to \cdots & \to r_t^3 = EP_3 \in \{0,1\}^{n-10}0000000000 \\ \vdots & \vdots \\ SP_m =_r r_0^m \to r_1^m \to r_2^m \to \cdots\cdots\cdots & \to r_{t_m}^m = EP_m \in \{0,1\}^{n-10}0000000000 \end{array}$$

4.2.9 Question 9

Énnoncé En supposant que l'on choisisse comme point distingué les chiffrés terminés par 10010010011, quelle est la longueur moyenne d'une chaîne ?

Correction Le problème revient à fixer les 11 bits. Supposons donc que la fonction de transition est une fonction aléatoire, on a une chance sur 2^{11} que l'on obtienne un point distingué à chaque fois que l'on calcule une nouvelle transition.
Il est clair que certaines chaînes peuvent être très longues dans ces conditions : il faudra un très grand nombre d'itérations avant d'arriver sur un point distingué. D'autres chaînes peuvent même boucler.

4.2.10 Question 10

Énnoncé Comment contourner le problème des chaînes qui bouclent ou qui ont une longueur trop importante ?

Correction Afin de contourner ce problème, on peut ajouter un compteur dans l'algorithme qui calcule la table et forcer l'arrêt d'un calcul de chaîne dès que la taille limite est atteint.

Énnoncé

4.2.11 Question 11

Proposer une implémentation en pseudo-code de l'attaque par compromis temps mémoire générique et comparer-là avec une implémentation utilisant l'amélioration des points distingués en précisant où sont stockées les données.

Correction La réalisation de cette implémentation est laissé en exercice.

Énnoncé Actuellement, la version la plus populaire des attaques par compromis temps-mémoire repose sur l'utilisation des « *tables arc-en-ciel* »proposées par Philippe Oechslin en 2003 dans l'article

> Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In *Dan Boneh*, editor. *Advances in Cryptology - CRYPTO 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*. Springer, 2003. Pages 617–630.

De nombreuses implémentations ce cette attaque sont disponibles sur le web. Elle est aujourd'hui largement utilisée pour ~~le crack~~ la récupération de mots de passes de certains systèmes d'exploitation. Elle a également permis le crack d'A5/1, le standard de chiffrement de la voix dans le protocole GSM.

5 Feuille d'Exercices 5

5.1 Exercice 1

Énoncé Combien y a-t-il :

1. de permutations de 128 bits sur 128 bits ?
2. de fonctions de 128 bits sur 128 bits ?

Correction Une permutation est une fonction bijective.
Donc il y a autant d'éléments dans l'ensemble d'arrivée que dans l'ensemble d'arrivée. Or, l'ensemble de départ contient toutes les chaînes possibles de 128 bits. Donc au total, il contient 2^{128} éléments.
Ainsi :

- Pour le premier élément, il y a 2^{128} choix possibles pour son image.
- Pour le deuxième élément, il y a $2^{128} - 1$ choix possibles pour son image.
- Pour le troisième élément, il y a $2^{128} - 2$ choix possibles pour son image.
- ...
- Pour le ième élément, il y a $2^{128} - i + 1$ choix possibles pour son image.
- ...
- Pour l'avant dernier élément, il y a deux choix possible pour son image.
- Pour le dernier élément (le 2^{128me}), il n'y a qu'un seul choix possible pour son image.

En tout, il y a donc $2^{128} * (2^{128} - 1) * ... * 1 = 2^{128}!$ permutations de 128 bits sur 128 bits.

Une fonction est une correspondance entre un élément de son ensemble de définition et un élément de son ensemble d'arrivée. Ainsi :

- Pour le premier élément, il y a 2^{128} choix possibles pour son image.
- Pour le deuxième élément, il y a 2^{128} choix possibles pour son image.
- Pour le troisième élément, il y a 2^{128} choix possibles pour son image.
- ...
- Pour le ième élément, il y a 2^{128} choix possibles pour son image.
- ...
- Pour l'avant dernier élément, il y a 2^{128} choix possibles pour son image.
- Pour le dernier élément, il y a 2^{128} choix possibles pour son image.

En tout, il y a donc $(2^{128})^{2^{128}}$ fonctions de 128 bits sur 128 bits.

5.2 Exercice 2 : Représentation matricielle vs. Représentation polynomiale

5.2.1 Question 1

Énoncé On considère, comme dans l'AES, le corps à 2^8 éléments, défini au moyen du polynôme irréductible $m(X) = X^8 + X^4 + X^3 + X + 1$. Calculer l'octet obtenu en calculant le produit suivant :

$$\{e1\} \times \{05\}.$$

Correction Traduisons le produit sous une forme polynomiale :

$$\{e1\} \times \{05\} = (X^7 + X^6 + X^5 + 1) \times (X^2 + 1)$$

Calculons maintenant le produit des polynômes modulo $m(X)$:

$$\begin{aligned}(X^7 + X^6 + X^5 + 1) \times (X^2 + 1) &= X^9 + X^8 + X^7 + X^2 \\ &+ X^7 + X^6 + X^5 + 1\end{aligned}$$

Il nous faut exprimer $X^9 \mod m(X)$ et $X^8 \mod m(X)$:

$$\begin{aligned}X^9 &= X.X^8 \mod m(X) \\ &= X(X^4 + X^3 + X + 1) \mod m(X) \\ &= X^5 + X^4 + X^2 + X \mod m(X) \\ &\text{et} \\ X^8 &= X^4 + X^3 + X + 1 \mod m(X)\end{aligned}$$

Donc :

$$\begin{aligned}(X^7 + X^6 + X^5 + 1) \times (X^2 + 1) &= X^5 + X^4 + X^2 + X \\ &+ X^4 + X^3 + X + 1 \\ &+ X^7 + X^2 + X^7 + X^6 + X^5 + 1 \mod m(X) \\ &= 2X^7 + X^6 + 2X^5 + 2X^4 + X^3 + 2X^2 + 2X + 2 \mod m(X)\end{aligned}$$

En réduisant les coefficients modulo 2, on obtient le polynôme :

$$X^6 + X^3 \mod m(X).$$

Le polynôme $X^6 + X^3$ représente l'octet $\{48\}$

5.2.2 Question 2

Énoncé On considère maintenant l'opération

$$\text{MixColumn} : \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} \mapsto \begin{pmatrix} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \end{pmatrix}$$

qui agit sur chaque colonne de la représentation "carrée" de l'AES.
Dans le cours, on a défini cette transformation par

$$\begin{pmatrix} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}.$$

Montrer que l'opération MixColumn peut se définir de manière équivalente par la formule

$$a'_3X^3 + a'_2X^2 + a'_1X + a'_0 =$$
$$(a_3X^3 + a_2X^2 + a_1X + a_0) \cdot (\{03\}X^3 + \{01\}X^2 + \{01\}X + \{02\}) \mod (X^4 + 1).$$

Correction Avant de corriger cette question, revenons sur les ensembles que l'on utilise dans les calculs :
Les octets sont mis en bijection avec l'ensemble des polynômes à coefficients dans $\mathbb{Z}/2\mathbb{Z}$
quotienté par le polynôme $m(X)$.
Et l'égalité à démontrer est en fait une égalité entre polynôme à coefficients dans $\mathbb{Z}/2\mathbb{Z}$
quotienté par le polynôme X^4+1.
Calculons le produit matriciel (les coefficients de la matrice sont des éléments de $(\mathbb{Z}/2\mathbb{Z}[X])/m(X)(\mathbb{Z}/2\mathbb{Z}[X])$: par définition on obtient les coefficients suivants :

$$\begin{aligned} a'_0 &= a_0\{02\} + a_1\{03\} + a_2\{01\} + a_3\{01\} \\ a'_1 &= a_0\{01\} + a_1\{02\} + a_2\{03\} + a_3\{01\} \\ a'_2 &= a_0\{01\} + a_1\{01\} + a_2\{02\} + a_3\{03\} \\ a'_3 &= a_0\{03\} + a_1\{01\} + a_2\{01\} + a_3\{02\} \end{aligned}$$

Maintenant calculons le produit des polynômes proposés dans la formule :

$$\begin{aligned} (a_3X^3+a_2X^2+a_1X+a_0)(\{03\}X^3+\{01\}X^2+\{01\}X+\{02\}) &= a_3\{03\}X^6 + a_2\{03\}X^5 + a_1\{03\}X^4 + a_0\{03\}X^3 \\ &+ a_3\{01\}X^5 + a_2\{01\}X^4 + a_1\{01\}X^3 + a_0\{01\}X^2 \\ &+ a_3\{01\}X^4 + a_2\{01\}X^3 + a_1\{01\}X^2 + a_0\{01\}X \\ &+ a_3\{02\}X^3 + a_2\{02\}X^2 + a_1\{02\}X + a_0\{02\} \end{aligned}$$

Il reste maintenant à réduire les termes dont la puissance est supérieure où égale à X^4 :
- $a_3\{03\}X^6 : X^6 = X^2X^4 = X^2 \mod X^4+1$ donc $a_3\{03\}X^6 = a_3\{03\}X^2 \mod X^4+1$
- $a_2\{03\}X^5 : X^5 = XX^4 = X \mod X^4+1$ donc $a_2\{03\}X^5 = a_2\{03\}X \mod X^4+1$
- $a_1\{03\}X^4 : X^4 = 1 \mod X^4+1$ donc $a_1\{03\}X^4 = a_1\{03\} \mod X^4+1$
- $a_3\{01\}X^5 : X^5 = XX^4 = X \mod X^4+1$ donc $a_3\{01\}X^5 = a_3\{01\}X \mod X^4+1$
- $a_2\{01\}X^4 : X^4 = 1 \mod X^4+1$ donc $a_2\{01\}X^4 = a_2\{01\} \mod X^4+1$
- $a_3\{01\}X^4 : X^4 = 1 \mod X^4+1$ donc $a_3\{01\}X^4 = a_3\{01\} \mod X^4+1$

Ainsi en réduisant modulo X^4+1, le produit est égale au polynôme suivant :

$$\begin{aligned} & (a_0\{03\} + a_1\{01\} + a_2\{01\} + a_3\{02\}) \quad X^3 \\ + & (a_3\{03\} + a_0\{01\} + a_1\{01\} + a_2\{02\}) \quad X^2 \\ + & (a_2\{03\} + a_3\{01\} + a_0\{01\} + a_1\{02\}) \quad X \\ + & (a_1\{03\} + a_2\{01\} + a_3\{01\} + a_0\{02\}) \end{aligned}$$

En réordonnant les coefficients du polynôme précédent, on aboutit à :

$$\begin{aligned} & (a_0\{03\} + a_1\{01\} + a_2\{01\} + a_3\{02\}) \quad X^3 \\ + & (a_0\{01\} + a_1\{01\} + a_2\{02\} + a_3\{03\}) \quad X^2 \\ + & (a_0\{01\} + a_1\{02\} + a_2\{03\} + a_3\{01\}) \quad X \\ + & (a_0\{02\} + a_1\{03\} + a_2\{01\} + a_3\{01\}) \end{aligned}$$

En reprenant le calcul matriciel précédent, on conclut que :

$$(a_3X^3+a_2X^2+a_1X+a_0)(\{03\}X^3+\{01\}X^2+\{01\}X+\{02\}) = (a'_3X^3+a'_2X^2+a'_1X+a'_0)$$

Ce qui prouve donc la formule proposée.

5.2.3 Question 3

Énoncé Vérifier que la matrice utilisée pour MixColumn a pour inverse la matrice

$$\begin{pmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{pmatrix}.$$

Expliquer pourquoi tous les coefficients de cette matrice commencent par un 0.

Correction Calculons maintenant le produit matriciel suivant :

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix}$$

$$= \begin{bmatrix} 02.0e + 03.09 + 01.0d + 01.0b & 02.0b + 03.0e + 01.09 + 01.0d & 02.0d + 03.0b + 01.0e + 01.09 & 02.09 + 03.0d + 01.0b + 01.0e \\ 01.0e + 02.09 + 03.0d + 01.0b & 01.0b + 02.0e + 03.09 + 01.0d & 01.0d + 02.0b + 03.0e + 01.09 & 01.09 + 02.0d + 03.0b + 01.0e \\ 01.0e + 01.09 + 02.0d + 03.0b & 01.0b + 01.0e + 02.09 + 03.0d & 01.0d + 01.0b + 02.0e + 03.09 & 01.09 + 01.0d + 02.0b + 03.0e \\ 03.0e + 01.09 + 01.0d + 02.0b & 03.0b + 01.0e + 01.09 + 02.0d & 03.0d + 01.0b + 01.0e + 02.09 & 03.09 + 01.0d + 01.0b + 02.0e \end{bmatrix}$$

Il reste à faire les calculs suivant :

- $01.09 = 09$
- $01.0b = 0b$
- $01.0d = 0d$
- $01.0e = 0e$
- $02.09 \iff X.(X^3+1) = X^4 + X$ donc $02.09 = 12$
- $02.0b \iff X.(X^3+X+1) = X^4 + X^2 + X$ donc $02.0b = 16$
- $02.0d \iff X.(X^3+X^2+1) = X^4 + X^3 + X$ donc $02.0d = 1a$
- $02.0e \iff X.(X^3+X^2+X) = X^4 + X^3 + X^2$ donc $02.0e = 1c$
- $03.09 \iff (X+1).(X^3+1) = X^4 + X^3 + X + 1$ donc $03.09 = 1b$
- $03.0b \iff (X+1).(X^3+X+1) = X^4 + X^3 + X^2 + 1$ donc $03.0b = 1d$
- $03.0d \iff (X+1).(X^3+X^2+1) = X^4 + X^2 + X + 1$ donc $03.0d = 17$
- $03.0e \iff (X+1).(X^3+X^2+X) = X^4 + X$ donc $03.0e = 12$

Le résultat du produit matriciel se réécrit de la manière suivante :

$$\begin{bmatrix} 1c + 1b + 0d + 0b & 16 + 12 + 09 + 0d & 1a + 1d + 0e + 09 & 12 + 17 + 0b + 0e \\ 0e + 12 + 17 + 0b & 0b + 1c + 1b + 0d & 0d + 16 + 12 + 09 & 09 + 1a + 1d + 0e \\ 0e + 09 + 1a + 1d & 0b + 0e + 12 + 17 & 0d + 0b + 1c + 1b & 09 + 0d + 16 + 12 \\ 12 + 09 + 0d + 16 & 1d + 0e + 09 + 1a & 17 + 0b + 0e + 12 & 1b + 0d + 0b + 1c \end{bmatrix}$$

On notera par commodité les coefficients de la matrices précédente $M(i,j)$. Calculons les seize coefficients de cette matrice :

$$\begin{aligned} M(1,1) = 1c + 1b + 0d + 0b &= (X^4 + X^3 + X^2) + (X^4 + X^3 + X + 1) + (X^3 + X^2 + 1) + (X^3 + X + 1) \\ &= 2X^4 + 4X^3 + 2X^2 + 2X + 3 \\ &= 1 \\ M(1,2) = 16 + 12 + 09 + 0d &= (X^4 + X^2 + X) + (X^4 + X) + (X^3 + 1) + (X^3 + X^2 + 1) \\ &= 2X^4 + 2X^3 + 2X^2 + 2X + 2 \\ &= 0 \\ M(1,3) = 1a + 1d + 0e + 09 &= (X^4 + X^3 + X) + (X^4 + X^3 + X^2 + 1) + (X^3 + X^2 + X) + (X^3 + 1) \\ &= 2X^4 + 4X^3 + 2X^2 + 2X + 2 \\ &= 0 \\ M(1,4) = 12 + 17 + 0b + 0e &= (X^4 + X) + (X^4 + X^2 + X + 1) + (X^3 + X + 1) + (X^3 + X^2 + X) \\ &= 2X^4 + 2X^3 + 2X^2 + 4X + 2 \\ &= 0 \end{aligned}$$

Puis :

$$\begin{array}{lcllcll}
M(2,1) & = & 0e+12+17+0b & = & M(1,4) & = & 0 \\
M(2,2) & = & 0b+1c+1b+0d & = & M(1,1) & = & 1 \\
M(2,3) & = & 0d+16+12+09 & = & M(1,2) & = & 0 \\
M(2,4) & = & 09+1a+1d+0e & = & M(1,3) & = & 0 \\
M(3,1) & = & 0e+09+1a+1d & = & M(1,3) & = & 0 \\
M(3,2) & = & 0b+0e+12+17 & = & M(1,4) & = & 0 \\
M(3,3) & = & 0d+0b+1c+1b & = & M(1,1) & = & 1 \\
M(3,4) & = & 09+0d+16+12 & = & M(1,2) & = & 0 \\
M(4,1) & = & 12+09+0d+16 & = & M(1,2) & = & 0 \\
M(4,2) & = & 1d+0e+09+1a & = & M(1,3) & = & 0 \\
M(4,3) & = & 17+0b+0e+12 & = & M(1,4) & = & 0 \\
M(4,4) & = & 1b+0d+0b+1c & = & M(1,1) & = & 1
\end{array}$$

D'après les calculs, on obtient :

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} = \begin{bmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \\ 00 & 00 & 00 & 01 \end{bmatrix}$$

On a donc bien vérifié que la matrice utilisée pour MixColumn a pour inverse la matrice

$$\begin{pmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{pmatrix}.$$

5.3 Exercice 3 : Fonction de hachage

Énoncé Soit $f : \{0,1\}^{2m} \longrightarrow \{0,1\}^m$ une fonction de hachage. Soit maintenant une deuxième fonction de hachage définie par

$$h : \begin{array}{ccc} \{0,1\}^{4m} & \longrightarrow & \{0,1\}^m \\ x_1||x_2 & \mapsto & f(f(x_1)||f(x_2)) \end{array}$$

où $||$ désigne l'opération de concaténation. Montrer que si f est à collisions fortes difficiles [1], alors h est aussi à collisions fortes difficiles.

Correction Supposons que h ne soit pas à collisions fortes difficiles ; par définition,

$$\exists y, y' \text{ tels que } y \neq y' \text{ et } h(y) = h(y')$$

On peut écrire $y = x_1||x_2$ et $y' = x'_1||x'_2$. Ainsi, par définition de h, on obtient :

$$f(f(x_1)||f(x_2)) = f(f(x'_1)||f(x'_2))$$

Ainsi, toute collision sur h permet de construire une collision sur f. Or, par hypothèse, on construit facilement des collisions sur h ainsi, f n'est pas à collisions fortes difficiles. Ce qui est absurde.
Conclusion, h est à collisions fortes difficiles.

5.4 Exercice 4 : Fonction de hachage basée sur AES

Soit $m = m_1 m_2 \dots m_n$ une chaîne de bits dans laquelle pour chaque $i = 1 \dots n$, m_i est un bloc de 128 bits. On définit une fonction de hachage H qui opère sur les mots binaires de cette forme en posant

— h_0 est un bloc de 128 bits tous nuls ;

1. C'est-à-dire qu'il est calculatoirement difficile d'obtenir deux messages différents x et x' tels que $f(x) = f(x')$.

— pour chaque $i = 1 \dots n$, $h_i = \text{AES}_{m_i}(h_{i-1})$, où $\text{AES}_K(m)$ est le résultat du chiffrement du bloc m avec la clé K ;
— $H(m) = h_n$.

Avant de passer à la résolution de cette question, examinons en détail le fonctionnement de cette fonction de hashage. Pour commencer, schématisons la fonction H :

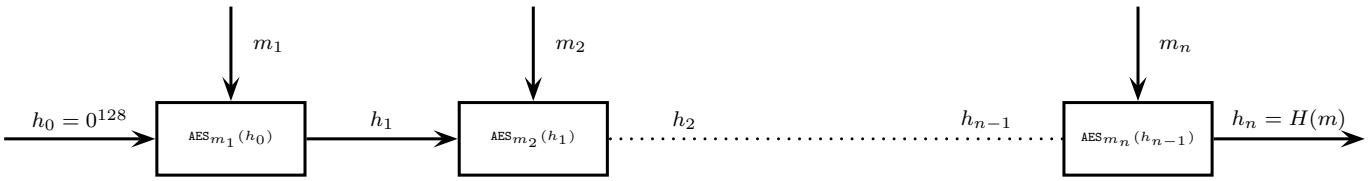


Figure 4 – Fonction H

Ce schéma permet de comprendre le fonctionnement de cette fonction de hashage : on sépare le message à hasher en n blocs de 128 bits, chacun de ces blocs sera ensuite utilisé comme une clé pour des chiffrements successives de $h_0 = 0^{128}$.

5.4.1 Question 1

Énoncé Montrer comment on peut trouver des collisions pour H en appliquant approximativement $c.2^{64}$ fois l'AES, où c est une constante.

Correction Ici on cherche simplement une collision sur la fonction de hashage H. La taille de la sortie de H est de 128 bits. En utilisant le paradoxe des anniversaires, on sait que l'on peut trouver une collision sur H avec une probabilité supérieure à 1/2, en choisissant aléatoirement $\text{E}(\sqrt{2\ln(2)}\sqrt{2^{128}}) + 1$ éléments et en calculant ensuite leur image par H. Cette recherche de collisions aboutit en effectuant de l'ordre de $n\sqrt{2\ln(2)}2^{64}$ chiffrements AES.

5.4.2 Question 2

Énoncé Étant donnée une chaîne m, montrer comment trouver une chaîne différente m' telle que $H(m) = H(m')$, en appliquant approximativement 2^{64} fois l'AES. [Indication : s'inspirer de l'attaque sur le double DES - voir Feuille 4, Exercice 1.]

Correction Dans cette question on cherche à montrer que cette fonction de hashage est vulnérable à des attaques de types seconde pré-image. L'énoncé suggère d'utiliser une attaque de type meet-in-the-middle.
Pour cette attaque, on va construire deux messages m et m' tels que $h'_2 = h_2$ et $m \neq m'$.
Pour faire le parallèle avec l'exercice de la Feuille 1, on a :

$$E = AES,\ K_1 = m_1, K_2 = m_2,\ M = h_0,\ C = h_2.$$

$|h_2| = 128$ bits. Donc en utilisant le paradoxe des anniversaires, si on tire au hasard suffisamment d'éléments de type $AES^{-1}_{m_2}(h_2)$ et $AES_{m_1}(h_1)$, on aura une bonne probabilité d'obtenir une collision.
Cette collision sera nécessairement une collision entre des éléments de types différents (car AES est une bijection).
Dès lors, deux cas peuvent se produire :

1. On obtient une collision $(m'_1, m'_2) \neq (m_1, m_2)$ Il suffit alors de considérer le message $m' = m'_1 m'_2 m_3 ... m_n$ qui, par construction, vérifie $m' \neq m$ et $H(m') = H(m)$.
2. On obtient la "collision' (m_1, m_2). Il nous suffit alors de recommencer la démarche précédente.

La construction de la collision repose sur le paradoxe des anniversaires. C'est donc une attaque probabiliste. En considérant qu'une bonne probabilité est de l'ordre de 1/2, il nous faudra considérer de l'ordre de $\sqrt{2^{128}}$ exécutions de l'algorithme AES. Ce qui nous permet de conclure que l'on peut construire une seconde pré-image en appliquant $c'2^{64}$.

6 Feuille d'Exercices 6

6.1 Exercice 1

Énoncé On considère un LFSR défini pour $n \geq 4$ sur $\mathbb{Z}/2\mathbb{Z}$ par la relation de récurrence

$$u_{i+4} = u_{i+1} + u_i.$$

1. Montrer que quelque soit le choix des conditions initiales $(u_0, u_1, u_2, u_3) \in (\mathbb{Z}/2\mathbb{Z})^4$, la période de la suite récurrente linéaire $(u_n)_{n\in\mathbb{N}}$ est majorée par 15.

   Correction Pour ceci il suffit de calculer les 15 premières éléments de la suite u_n en fonction de u_0, u_1, u_2 et u_3 :
   $u_4 = u_1 + u_0$
   $u_5 = u_{1+4}u_2 + u_1$
   $u_6 = u_3 + u_2$
   $u_7 = u_4 + u_3 = u_0 + u_1 + u_3$
   $u_8 = u_5 + u_4 = u_2 + u_1 + u_1 + u_0 = u_0 + u_2$
   $u_9 = u_6 + u_5 = u_3 + u_2 + u_2 + u_1 = u_1 + u_3$
   $u_{10} = u_7 + u_6 = u_0 + u_1 + u_2$
   $u_{11} = u_8 + u_7 = u_1 + u_2 + u_3$
   $u_{12} = u_9 + u_8 = u_0 + u_1 + u_2 + u_3$
   $u_{13} = u_{10} + u_9 = u_0 + u_2 + u_3$
   $u_{14} = u_{11} + u_{10} = u_0 + u_3$
   $u_{15} = u_{12} + u_{11} = u_0$
   $u_{16} = u_{13} + u_{12} = u_1$

2. Dessiner le LFSR correspondant.
3. On considère le chiffrement à flot dont le générateur pseudo-aléatoire est le LFSR précédent. On encode chaque lettre de l'alphabet par le quintuplet $(a_0, a_1, a_2, a_3, a_4)$ tel que $a_0 + 2a_1 + 4a_2 + 8a_3 + 16a_4$ soit le rang de la lettre dans l'alphabet. Par exemple, la lettre A (de rang 1) est encodée par le quintuplet $(1, 0, 0, 0, 0)$; la lettre K (de rang 11) est encodée par le quintuplet $(1, 1, 0, 1, 0)$.
   On reçoit le message suivant :

   10000 10101 10111 00000 10001 11101

   — D'après la question précédente, et sans déchiffrer le message, que peut on dire sur le message ?
   — Déchiffrer le message, sachant que l'état initial du LFSR est $(u_0, u_1, u_2, u_3) = (1, 0, 1, 0)$.

   Correction Le message donné contient 30 bits. On a donc besoin une clé de 30 bits également : $u_0 = 1 = u_{15}$
   $u_1 = 0 = u_{16}$
   $u_2 = 1 = u_{17}$
   $u_3 = 0 = u_{18}$
   $u_4 = 1 = u_{19}$
   $u_5 = 1 = u_{20}$
   $u_6 = 1 = u_{21}$
   $u_7 = 1 = u_{22}$
   $u_8 = 0 = u_{23}$
   $u_9 = 0 = u_{24}$
   $u_{10} = 0 = u_{25}$
   $u_{11} = 1 = u_{26}$
   $u_{12} = 0 = u_{27}$
   $u_{13} = 0 = u_{28}$
   $u_{14} = 1 = u_{29}$

Pour trouver le message en clair, il suffir de xorer le message chiffré et la clé calculée :

$$10000\ 10101\ 10111\ 00000\ 10001\ 11101\oplus$$

$$10101\ 11100\ 01001\ 10101\ 11100\ 01001 =$$

$$00101\ 01001\ 11110\ 10101\ 01101\ 10100$$

En faisant la correspondance en décimal, on obtient la suite : 20 18 15 21 22 5 qui correspond au mot : "TROUVE".

6.2 Exercice 2

Énoncé On considère l'anneau $\mathbb{Z}/2\mathbb{Z}[X]$ des polynômes à coefficients dans $\mathbb{Z}/2\mathbb{Z}$ munis de l'addition et de la multiplication usuelle. Dans tout cet exercice, on considère un polynôme *irréductible* $P(X) \in \mathbb{Z}/2\mathbb{Z}[X]$ de degré d. On défini le corps fini $\mathbb{K} = \mathbb{Z}/2\mathbb{Z}[X]/(P(X))$ des polynômes de degré au plus (d-1) à coefficients dans $\mathbb{Z}/2\mathbb{Z}[X]$, muni de l'addition et de la multiplication usuelle entre $a(X), b(X) \in \mathbb{Z}/2\mathbb{Z}[X]$ définie par

$$a(X) * b(X) = a(X) \times b(X) \mod P(X).$$

On construit une séquence $s_0(X), s_1(X), \ldots$ dans $\mathbb{K}$ définie par $s_0(X) = 1$ et pour tout $t \geq 0$, $s_{t+1}(X) = X * s_t(X)$. On a

$$s_t(X) = X^t \mod P(X) \text{ pour tout } t \geq 0$$

.

1. Calculer les huit premiers éléments de la séquence lorsque $P(X) = X^3 + X + 1$. Quelle est la période de la séquence ?

   Correction

   $$\begin{aligned}
   s_0(X) &= X^0 = 1 \mod P(X)\\
   s_1(X) &= X^1 = X \mod P(X)\\
   s_2(X) &= X^2 \mod P(X)\\
   s_3(X) &= X^3 = X + 1 \mod P(X)\\
   s_4(X) &= X^4 = X^2 + X \mod P(X)\\
   s_5(X) &= X^5 = X^3 + X^2 = X^2 + X + 1 \mod P(X)\\
   s_6(X) &= X^6 = X^3 + X^2 + X = X^2 + 1 \mod P(X)\\
   s_7(X) &= X^7 = X^3 + X = 1 \mod P(X)\\
   s_8(X) &= X^8 = X \mod P(X)
   \end{aligned}$$

   On remarque que $s_0(X) = s_7(X)$ donc la période est 7.
2. À chaque élément $q(X) = q_0 + q_1 X + \cdots + q_{d-1} X^{d-1}$ de $\mathbb{K}$, on associe un entier $\tilde{q}$ défini par

   $$\tilde{q} = q_0 + q_1 2 + \cdots + q_{d-1} 2^{d-1}.$$

   Comment est-il possible d'implémenter le calcul de $\tilde{s}_{t+1}$ à partir de $\tilde{s}_t$ à l'aide des instructions usuelles contenues dans un microprocesseur ?

Correction Le calcul de s_{t+1} s'effectue de la façon suivante :

(a) On multiplie s_t par X ;

(b) Si besoin on réduit modulo $P(X)$.

Il faut traduire ces opérations sur les polynômes en des opérations sur les entiers associés.
La multiplication par X revient à faire un décalage vers la droite (ou une multiplication par 2). La réduction doit s'effectuer si le bit en position d vaut 1. On va donc supposer que l'on est capable de tester cette position de la façon suivante : nos données sont stockées sur d bits ; ainsi après le décalage, le bit en position d se trouve dans le flag que l'on est capable d'interroger.
Si le flag contient 0, la réduction n'est pas nécessaire ; en revanche si le flag vaut 1, il faut effectuer la réduction modulaire. Comme on travaille avec des polynômes à coefficients dans $\mathbb{Z}/2\mathbb{Z}$, la réduction modulaire correspond simplement à une addition entre l'entier après décalage et l'entier $\tilde{P}$.

3. On définit $c_{t,j}$ comme le coefficient de X^j dans $s_t(X)$ et la matrice M_t de taille $d \times d$ à coefficients dans $\mathbb{Z}/2\mathbb{Z}$ comme la matrice

$$(M_t)_{i,j} = c_{i+t-1,j-1}$$

pour $1 \leq i, j \leq d$ et $t \geq 0$.

(a) Montrer qu'il existe une relation $M_{t+1} = B \times M_t$ et calculer la matrice B ;

Correction Soient $P(X) = P_0 + P_1 \cdot X + ... + P_{d-1} \cdot X^{d-1} + X^d$ et $Q(X) = Q_0 + Q_1 \cdot X + ... + Q_{d-1} \cdot X^{d-1}$.
Ces deux polynômes peuvent être représentés par des vecteurs-lignes. Notons $\bar{Q}$ la représentation de $Q(X)$: $\bar{Q} = (Q_0, Q_1, ..., Q_{d-1})$.
La multiplication par X peut alors être représentée par une multiplication de matrice. Notons $R(X)$ cette multiplication. Donc :

$$\begin{array}{rcll} R(X) & = & R_0 + ... + R_{d-1}X^{d-1} & \\ & = & X \times Q(X) & \mod P(X) \\ & = & XQ_0 + ... + X^d Q_{d-1} & \\ & = & XQ_0 + ... + (P_0 + P_1X + ... + P_{d-1}X^{d-1})Q_{d-1}) & \\ & = & P_0Q_{d-1} + X(Q_0 + Q_{d-1}P_1) + ... + X^{d-1}(Q_{d-2} + Q_{d-1}P_{d-1}) & \end{array}$$

ou alors $\bar{Q} = (Q_{d-1}P_0, Q_0 + Q_{d-1}P_1, ..., Q_{d-2} + Q_{d-1}P_{d-1})$. Donc la multiplication par X peut être représentée par $\bar{R} = \bar{Q} \cdot B$, avec

$$B = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ P_0 & P_1 & P_2 & \cdots & P_{d-1} \end{bmatrix}$$

Par définition de la séquence : $\bar{s_{i+t}} = \bar{s_{i+t-1}} \cdot B$ pour $i \geq 1$ et $t \geq 0$.
Notons que la i-ème ligne de M_t correspond à $\bar{s_{i+t-1}}$ et que la i-ème ligne de M_{t+1} correspond à $\bar{s_{i+t}}$.
Alors $M_{t+1} = M_t \cdot B$.
M_0 étant la matrice identité et $M_t = B^t$ pour tout $t \geq 0$, M_t et B permutent et donc $M_{t+1} = B \cdot M_t$

(b) Monter que pour un $0 \leq j \leq d-1$ donné, il est possible de calculer $c_{t+d,j}$ linéairement à partir de $c_{t,j}, c_{t+1,j}, \ldots, c_{t+d-1,j}$;

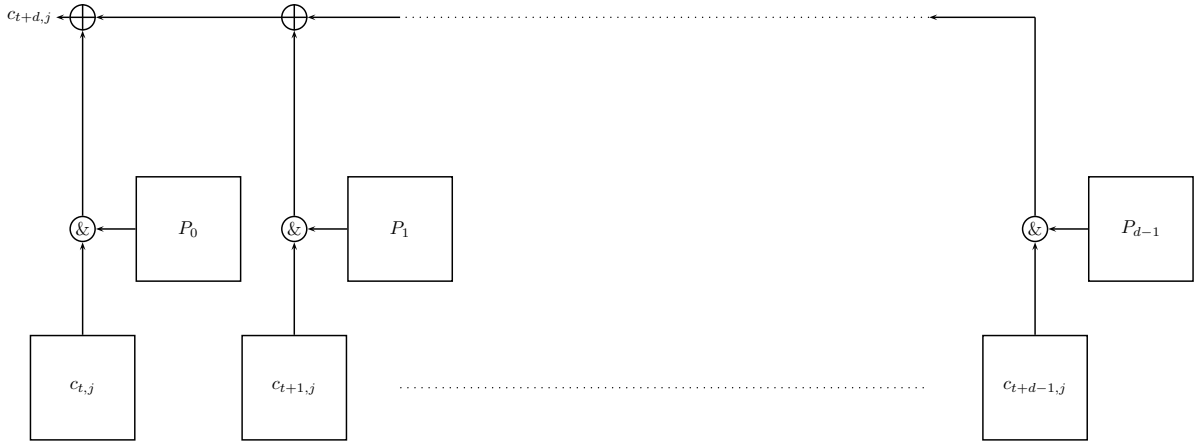
Correction On va maintenant montrer que $c_{t+d,j}$ s'exprime linéairement en fonction des $c_{t+i,j}$:
par définition : $c_{t+d,j} = (M_{t+1})_{d,j+1} = (B \times M_t)_{d,j+1} = P_0c_{t,j} + P_1c_{t+1,j} + ... + P_{d-1}c_{t+d-1,j}$ ce qui achève la preuve.

(c) Comment est-il possible de construire un circuit électronique qui calcule la séquence définie à la première question à partir de registres de 1 bit et d'additionneurs 1 bit ?

Correction Pour finir, on doit montrer qu'il est possible de calculer la séquence définie à la première question à l'aide de registres et d'additionneurs 1 bit. Pour cela il suffit de remarquer que le calcul de la séquence est équivalent au calcul des $c_{i,j}$. En s'appuyant sur la question le fait que

$$c_{t+d,j} = P_0 c_{t,j} + P_1 c_{t+1,j} + ... + P_{d-1} c_{t+d-1,j},$$

on obtient directement le résultat. En effet, il suffit de construire le circuit suivant :



4. Quelles sont les valeurs possibles de la période de la séquence $s_i(X)$ pour $i \geq 0$? Quand est-elle maximale ?

Correction L'ensemble $\mathcal{E}$ des éléments de la séquence $\{X^t \mod P(X) : t \in \mathbb{N}\}$ muni de la multiplication est un sous-groupe des éléments inversibles du corps $\mathbb{K}$ noté $\mathbb{K}^*$ dans la suite.
En effet : $1 \in \mathcal{E}$ car $1 = X^0 \mod P(X)$.
De plus en considérant a et b deux éléments de $\mathcal{E}$, il existe par définition α et β deux entiers tels que $a = X^\alpha \mod P(X)$ et $b = X^\beta \mod P(X)$, on a alors $a \times b = X^{\alpha+\beta} \mod P(X) \in \mathcal{E}$. Ce qui achève la preuve.
$\mathbb{K}^*$ est un ensemble contenant $2^d - 1$, comme $\mathcal{E}$ est un sous-groupe de $\mathbb{K}^*$, d'après le théorème de Lagrange, le nombre d'éléments de $\mathcal{E}$ divise $2^d - 1$.
La question de la période maximale se comprend de la façon suivante : $\mathcal{E}$ étant inclus dans $\mathbb{K}^*$, la période est maximale lorsque le cardinal de $\mathcal{E}$ est maximal. D'après le théorème de Lagrange, cela revient à dire que la période est maximale lorsque $2^d - 1$ est un nombre premier. En effet, il n'admet alors que deux diviseurs : lui-même et 1. Mais si l'on considère que $\mathbb{K}$ est un quotient par un polynôme $P(X)$ de degré supérieur à 1, on est sûr du fait que $\mathcal{E}$ contient au moins $1 \mod P(X)$ et $X \mod P(X)$ ce qui permet de garantir que le nombre d'éléments de $\mathcal{E}$ est exactement $2^d - 1$ et les deux ensembles $\mathcal{E}$ et $\mathbb{K}^*$ sont alors égaux.
Lorsque n'est pas premier, on ne peut pas garantir que le nombre d'éléments de $\mathcal{E}$ est maximal.

6.3 Exercice 3

Énoncé Dans un chiffrement à flot, le chiffrement est réalisé par un *ou exclusif* (ou *xor*) entre un clair de m bits et un flux chiffrant m (*keystream* en anglais), lequel est la sortie d'un générateur de flux (*keystream generator* en anglais) alimenté par une clef de ℓ bits où ℓ est largement plus petit que m. Une hypothèse idéale pour un bon chiffrement à flot est que toute fenêtre de ℓ bits de la séquence de m bits est possiblement modifiée lorsque la clef de ℓ bits est modifiée. Cet exercice vise à faire un petit test de cette hypothèse, en utilisant le schéma A5/1.
A5/1 est composé de trois LFSRs désignés par R_1, R_2 et R_3 de longueurs respectives 19, 22 et 23. Le total des positions contenues dans ces trois LFSR est donc 64 bits. On désigne l'état initial de 64 bits comme étant la clef de A5/1 et on désigne par $R_i[n]$ le contenu de la n^{me} cellule de R_i pour $i = 1, 2, 3$ et $n \geq 0$. Pour chaque LFSR, on désigne une cellule d'horloge : $R_1[8], R_2[10]$ et $R_3[10]$. Á chaque cycle d'horloge, un bit du flux chiffrant est généré en suivant la procédure suivante :

— Les trois LFSR réalisent un "vote" désignant le bit de majorité (le bit le plus représenté) contenu dans les trois cellules d'horloge ;

— Chaque R_i compare le résultat du vote avec sa propre cellule d'horloge. Si ces deux bits sont égaux, R_i est décalé :
  — Un bit de rétroaction est calculé par ou exclusif entre les contenus d'un sous-ensemble des cellules de R_i : $R_1[18] \oplus R_1[17] \oplus R_1[13], R_2[21] \oplus R_2[20]$ et $R_3[22] \oplus R_3[21] \oplus R_3[20] \oplus R_3[7]$ respectivement ;
  — Les contenus de chaque cellule sont décalés d'une position vers la gauche ;
  — $R_i[0]$ est remplacé par le bit de rétroaction précédemment calculé ;
— Renvoyer le bit $R_1[18] \oplus R_2[21] \oplus R_3[22]$.

Le fonctionnement du générateur de flux chiffrant d'A5/1 est représenté sur la figure 5.

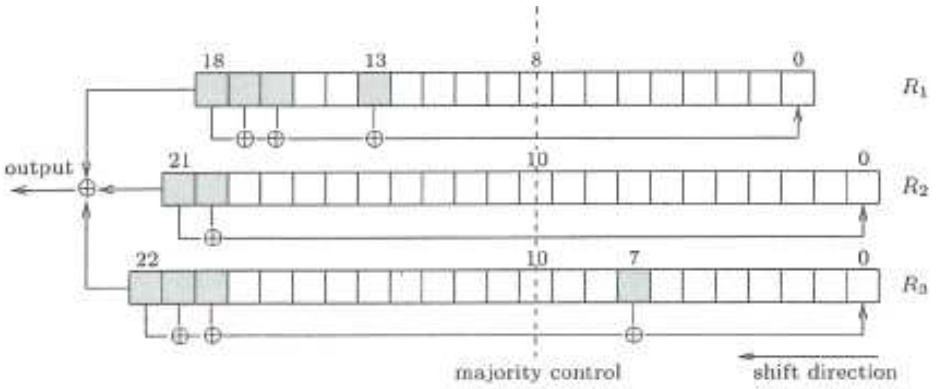


Figure 5 – Générateur de flux chiffrant A5/1

1. Montrer que lorsque R_1 est chargé avec un état initial précis, alors son état interne ne changera jamais. Est-il possible d'étendre ce résultat à R_2 et R_3 ?

   Correction Si R_1 est chargé avec uniquement des 0, la rétroaction sera toujours 0. Donc l'état suivant restera inchangeable.
   Évidement ce résultat est aussi valide pour R_2 etR_3.

2. Utiliser la réponse précédente pour contredire l'hypothèse mentionnée plus tôt dans le cas suivant : montrer que le flux de 64 bits tous à zéro peut être générée par différentes clefs.

   Correction Quand deux LFSR sont initialisés à 0, A5/1 est équivalent au LFSR qui reste. Ce LFSR sortira le bit le plus à gauche à chaque coup d'horloge et il est décalé si et seulement si sa cellule d'horloge est 0. Mais la cellule d'horloge d'un LFSR non-nul ne peut pas être toujours 0. Donc, après un nombre limité de coups d'horloge, la cellule d'horloge du LFSR sera 1 et donc le LFSR s'arrêtera pour toujours et il sortira le même bit. Donc, tant que le LFSR non nul sort 0 avant que (et quand) sa cellule d'horloge retournera 1, A5/1 générera le flux nul (y compris le cas de 3 registres initialement mis à 0).

3. Calculer une borne inférieure sur le nombre de clefs différentes qui génèrent ce flux.

   Correction On peut distinguer 4 cas :
   — $R_1 = R_2 = R_3 = 0$: on a 1 seule possibilité (tous équivaut à 0)
   — $R_1 \neq 0, R_2 = R_3 = 0$:
     — Si $R_1[8] = 1$, R_1 n'est jamais décalé. Dans ce cas, il suffit d'avoir aussi $R_1[18] = 0$ pour obtenir un flux avec uniquement de 0. Ceci laisse $2^{19-2} = 2^{17}$ états d'initialisation différents.
     — Si $R_1[8] = 0$ et $R_1[7] = 1$, alors R_1 sera décalé une seule fois. Dans ce cas, il suffit d'avoir aussi $R_1[18] = R_1[17] = 0$ pour obtenir un flux avec uniquement de 0. Ceci laisse $2^{19-4} = 2^{15}$ états d'initialisation différents.
     — En continuant ce raisonnement on obtient la borne inférieure suivante :
     $2^{17} + 2^{15} + 2^{13} + 2^{11} + 2^9 + 2^7 + 2^5 + 2^3 + 2 = \frac{2^{19}-2}{3}$
   — $R_2 \neq 0, R_1 = R_3 = 0$: Idem, on obtient la borne inférieure suivante :
   $2^{20} + 2^{18} + ... + 1 = \frac{2^{22}-1}{3}$

— $R_3 \neq 0, R_1 = R_2 = 0$: Idem, on obtient la borne inférieure suivante :
$2^{21} + 2^{19} + ... + 2 = \frac{2^{23}-2}{3}$

En ajoutant toutes ces bornes, on obtient la borne inférieure globale.

klae

7 Feuille d'Exercices 7

7.1 Exercice 1 : Un mauvais MAC

Considérons le schéma de MAC suivant. Soit E un algorithme de chiffrement par bloc, la taille des blocs étant de n bits, et soit h une fonction de hachage (à collisions fortes difficiles) dont la sortie fait n bits. Alors, pour tout message M de longueur $N > n$, on obtient le MAC en calculant $E_k(h(M))$. Par ailleurs, pour tout message m de taille n, le MAC est $E_k(m)$. (Pour simplifier, on suppose que tous les messages font au moins n bits).

7.1.1 Question 1

Énoncé Montrer que ce MAC n'est pas sûr.

Correction Montrer que ce MAC n'est pas sûr, c'est, pour un message donné m, construire un autre message m' qui a le même MAC.
Il est facile de voir que si l'on considère un message m de taille $|m| > n$, alors en calculant $h(m)$, on obtient un message m' tel que MAC(m) = MAC (m').
Conclusion ce MAC n'est pas sûr.

7.1.2 Question 2

Énoncé Comment modifier la construction du MAC pour le rendre sûr ?

Correction Pour rendre ce MAC sûr, il faut analyser l'endroit où il pose problème : les messages de même taille que la fonction de hashage ne sont pas hashés avant d'être encrypté. Afin de s'affranchir de cette difficulté, on peut proposer une solution de padding en construisant le MAC suivant :

$$\text{MAC} : \begin{pmatrix} \{0,1\}^n.\{0,1\}^* & \longrightarrow & \{0,1\}^n \\ m & \longmapsto & E_k(h(m)) \end{pmatrix}$$

En fait on impose le hashage de n'importe quel message.

7.2 Exercice 2 : CFB-MAC

Dans cet exercice, on étudie un schéma de MAC basé sur le mode de chiffrement CFB. On considère un algorithme de chiffrement par blocs

$$E : \{0,1\}^{64} \times \{0,1\}^{64},$$

où $E_k(x) = E(k,x)$ désigne le résultat du chiffrement du message x avec la clé k. Le CFB-MAC d'un message donné $m \in \{0,1\}^*$ avec la clé k est obtenu en chiffrant tout d'abord m par E_k en mode CFB, puis en calculant le XOR de tous les blocs obtenus en sortie.
Plus précisément, pour un message $m = x_1 x_2 \dots x_n$,

$$\text{CFB-MAC}_k(m) = y_1 \oplus y_2 \oplus \dots \oplus y_n,$$

où $y_i = E_k(y_{i-1}) \oplus x_i$ pour $i = 2, \dots, n$ et $y_1 = E_k(IV) \oplus x_1$, IV étant une "valeur d'initialisation".
Pour simplifier, on supposera que tous les messages ont une longueur multiple de 64 bits. On suppose également dans toutes les questions de l'exercice, que IV est constante et connue.

7.2.1 Question 1

Énoncé Supposons qu'on ait accès à un oracle $\mathcal{O}$ qui calcule le CFB-MAC décrit ci-dessus, pour une clé secrète k donnée et une valeur IV fixée et connue. Montrer que l'on peut retrouver $E_k(IV)$ en faisant un seul appel à l'oracle.

Correction On souhaite récupérer en un seul appel à l'oracle la valeur $E_k(IV)$. L'idée est donc de construire un message m dont le hashé sera $E_k(IV)$. Ainsi, en soumettant ce message à l'oracle, on récupérera $E_k(IV)$.
Le message m est connu (on en est maître).
Choisissons un message qui ne comprend qu'un seul bloc. Par définition de $CFB-MAC$,

$$CFB-MAC_k(m) = E_k(IV) \oplus m$$

On peut donc récupérer $E_k(IV)$ en calculant $CFB-MAC_k(m) \oplus m$.

7.2.2 Question 2

Énoncé Supposons qu'un attaquant ait accès à un oracle $\mathcal{O}$ qui calcule le CFB-MAC décrit ci-dessus, pour une clé secrète k donné et une valeur IV fixée et connue. L'attaquant voudrait trouver une collision pour CFB-MAC, pour 2 messages différents ayant 192 bits chacun. Combien de messages de 192 bits l'attaquant doit-il envoyer à $\mathcal{O}$ pour obtenir une collision avec une probabilité proche de 0.9996 ($\simeq 1 - e^{-8}$) ?

Correction Pour répondre à cette question, on va utiliser le paradoxe des anniversaires :

$$n = f(p) = \mathrm{E}(\sqrt{2\ln(\frac{1}{1-(1-e^{-8})})}\sqrt{d}) + 1$$

En faisant l'application numérique, on obtient : $n \approx 2^{34} = 17179869184$.

7.2.3 Question 3

Énoncé Étant donné un message m de n blocs et h =CFB-MAC$_k(m)$, montrer comment on peut construire un nouveau message m' de n blocs, et $h' \in \{0,1\}^{64}$, tels que $m' \neq m$ et CFB-MAC$_k(m') = h'$.

Correction Pour cette question, on suppose que $m = x_1x_2...x_n$. Par définition de la fonction $CFB-MAC$, on a :

$$CFB-MAC_k(m) = \underbrace{x_1 \oplus E_k(IV)}_{y_1} \oplus \underbrace{E_k(y_1) \oplus x_2}_{y_2} \oplus ... \oplus \underbrace{E_k(y_{n-1}) \oplus x_n}_{y_n} = h$$

Remarquons que $CFB-MAC_k(m) \oplus h \oplus h' = h'$.
Ainsi, en choisissant $m' = x_1x_2...x'_n$ où $x'_n = x_n \oplus h \oplus h'$, on a bien construit un message m' différent de m tel que $CFB-MAC_k(m') = h'$.

7.2.4 Question 4

Énoncé Supposons qu'on connaisse IV, $E_k(IV)$, et $h \in \{0,1\}^{64}$. Montrer comment il est possible de construire un message m de deux blocs, tel que CFB-MAC$_k(m) = h$.

Correction Cette attaque se base sur la façon dont le chiffrement est fait :

$$CFB-MAC_k(m = x_1x_2) = E_k(IV) \oplus x_1 \oplus x_2 \oplus E_k(E_k(IV) \oplus x_1)$$

En choisissant :

$$\begin{cases} x_1 &= E_k(IV) \oplus IV \\ x_2 &= E_k(IV) \oplus IV \oplus h \end{cases}$$

Le hash de $m = x_1x_2$ est h :

$$\begin{array}{rcl} CFB-MAC_k(m) & = & \underbrace{E_k(IV) \oplus E_k(IV) \oplus IV}_{y_1} \oplus E_k(y_1) \oplus x_2 \\ \text{Par définition} \quad y_1 = & & IV \\ CFB-MAC_k(m) & = & IV \oplus E_k(IV) \oplus x_2 \\ \text{Or on a choisi} \quad x_2 = & & E_k(IV) \oplus IV \oplus h \\ CFB-MAC_k(m) & = & \underbrace{IV \oplus E_k(IV) \oplus E_k(IV) \oplus IV}_{=0} \oplus h \\ & = & h \end{array}$$

7.2.5 Question 5

Énoncé Peut-on étendre l'attaque de la question précédente à des messages m de plus que deux blocs ? Expliquer votre réponse.

Correction On se place dans le cas général d'un message de n blocs.
En s'insiprant du raisonnement précédent, on voit que l'on peut faire en sorte que les $y_i, i < n$ soient tous égaux à IV.
Pour cela il suffit de choisir $x_i = IV \oplus E_k(IV)$. La preuve se fait ensuite par récurrence :
Initialisation : par définition, $y_1 = x_1 \oplus E_k(IV) = IV \oplus \underbrace{E_k(IV) \oplus E_k(IV)}_{=0} = IV$.
Hérédité : Supposons que $\forall j < k < n-1, y_j = IV$; montrons qu'alors, $y_k = IV$:
Par définition $y_k = x_k \oplus E_k(y_{k-1})$.
D'après l'hypothèse de récurrence : $y_{k-1} = IV$ et par construction, $x_k = E_k(IV) \oplus IV$.
Ainsi $y_k = x_k \oplus E_k(IV) = IV \oplus \underbrace{E_k(IV) \oplus E_k(IV)}_{=0} = IV$.
Ceci nous permet de conclure.
En s'appuyant sur le raisonnement précédent, examinons les différents couples de la forme
$y_{2i} \oplus y_{2i+1},\ 2i+1 < n$:

$$y_{2i} \oplus y_{2i+1} = IV \oplus IV = 0$$

Ainsi les "blocs" (ici "bloc" désigne un couple de la forme $y_{2i} \oplus y_{2i+1},\ 2i+1 < n$) précédents le dernier "blocs" s'éliminent tous deux à deux.
Il reste à examiner le cas du dernier "bloc". Il faut distinguer deux sous-cas :

1. Cas d'un nombre pair de "blocs" :
   Le dernier "bloc" est de la forme $y_{n-1} \oplus y_n$ où, d'après la récurrence précédente, $y_{n-1} = IV$.
   Par construction,

   $$CFB-MAC(m) = y_{n-1} \oplus y_n = IV \oplus E_k(y_{n-1}) \oplus x_n = IV \oplus E_k(IV) \oplus x_n.$$

   Comme on souhaite $CFB-MAC(m) = h$, on obtient : $x_n = IV \oplus E_k(IV) \oplus h$.
2. Cas d'un nombre impair de "blocs" :
   Par construction et d'après la récurrence précédente,

   $$CFB-MAC(m) = y_n = E_k(y_{n-1}) \oplus x_n = E_k(IV) \oplus x_n$$

   Comme on souhaite $CFB-MAC(m) = h$, on obtient : $x_n = E_k(IV) \oplus h$.

Conclusion :

1. Dans le cas où $|m| = 0 \mod 2$, on choisit :

   $$\begin{cases} x_i = E_k(IV) \oplus IV & \forall i,\ i < |m| \\ x_{|m|} = E_k(IV) \oplus IV \oplus h \end{cases}$$

2. Dans le cas où $|m| = 1 \mod 2$, on choisit :

$$\begin{cases} x_i & = & E_k(IV) \oplus IV \quad \forall i,\ i < |m| \\ x_{|m|} & = & E_k(IV) \oplus h \end{cases}$$

8 Feuille d'Exercices 8

8.1 Exercice 1 : Malléabilité et Indistinguabilité

De manière générale, on souhaite que les messages chiffrés d'un cryptosystème aient une répartition aléatoire afin d'empêcher tout attaquant qui se contenterait d'observer les messages chiffrés de récupérer des informations. Cette propriété est capturée par la notion d'intistinguabilité de deux messages chiffrés : si étant donnés deux messages m_1 et m_2 choisis par un adversaire, celui-ci ne peut décider de quel message provient un chiffré c, on dit que le cryptosystème est *indistinguable*. De plus, il est souhaitable qu'un attaquant ne puisse pas obtenir un message chiffré valide en modifiant un message chiffré qu'il aurait intercepté. On dit alors que le cryptosystème n'est pas *malléable*.

Énoncé

1. Rappelez le fonctionnement du cryptosystème RSA ;

   Correction Soient p et q deux entiers premiers et $n = p \cdot q$.
   On choisit un nombre entier e tel que $1 < e < \phi(n)$ et $gcd(e, \phi(n)) = 1$. On calcule alors d tel que $1 < d < \phi(n)$ et $e \cdot d = 1 \ mod \ \phi(n)$.
   Pour le cryptosystème RSA,
   — les clés privées sont : $\phi(n)$ et d
   — les clés publiques sont : n et e
   — le chiffrement est calculé à l'aide de : $c = m^e \ mod \ n$
   — le déchiffrement est calculé à l'aide de : $m = c^d \ mod \ n$
2. Montrer que ce cryptosystème est à la fois malléable et distinguable.

   Énnoncé
   — malléable :
   Il suffit de trouver un exemple, pour lequel étant donné un chiffré du message m, il est possible de générer un autre chiffré qui déchiffre $f(m)$.
   Supposons qu'on veut calculer le chiffré de $m \cdot t$:
   $E(m \cdot t) = (m \cdot t)^e \ mod \ n = m^e \cdot t^e \ mod \ n = t^e \ mod \ n \cdot E(m)$
   — distinguable :
   Un attaquant peut choisir deux messages : m_0 et m_1. Il peut ensuite calculer les deux chiffrés et comparer les résultats.

On propose alors une variante dite RSA - OAEP (Optimal Asymetric Encryption Padding) qui fonctionne de la manière suivante :
— On se donne deux tailles : n et k_r ;
— G est une fonction de hachage cryptographique dont la sortie est sur $n - k_r$ bits ;
— H est une fonction de hachage cryptographique dont la sortie est sur k_r bits ;
— On suppose que l'on souhaite chiffrer des messages de taille comprise entre 1 et $n - k_r$ bits (lorsque le message a une taille strictement inférieure à $n - k_r$, on le complète avec des 0 afin d'atteindre un bloc de taille $n - k_r$) ;
— On génère une valeur aléatoire r sur k_r bits ;
— Enfin, on chiffre enfin le message suivant :

$$[(m||000) \oplus G(r)] \, ||[H \, ((m||000) \oplus G(r)) \oplus r].$$

3. Montrer comment le destinataire peut déchiffrer un message ;

   Correction Supposons que le destinataire reçoit un chiffré c. Pour le dechiffrer, il coupe le résultat en deux parties c_1 et c_2 où c_1 et de longueur $n - k_r$.
   Il calcule ensuite : $r = H(c_1) \oplus c_2$. Le message est alors : $m = c_1 \oplus G(r)$
4. Quel intérêt y a-t-il selon vous à utiliser RSA - OAEP ?

8.2 Exercice 2 : Factorisation

On suppose $n = p \times q$, où p et q sont deux nombres premiers distincts. On rappelle que $\varphi(n) = (p-1)(q-1)$.

Énoncé On suppose que n et $\varphi(n)$ sont connus. Montrer comment cela permet de retrouver p et q.

Correction On va utiliser les formules de Newton pour résoudre cette question :

$$\begin{aligned} P &= n = p \times q \\ S &= (n+1) - \varphi(n) = p + q \end{aligned}$$

De là, on résout l'équation suivante :

$$X^2 - SX + P$$

Les solutions sont p et q.

Énoncé Trouver la factorisation de n dans les deux cas suivants :
- $n = 667$, $\varphi(n) = 616$.
- $n = 15049$, $\varphi(n) = 14800$.

Correction Les solutions sont :
- $n = 667$: $p = 23$ et $q = 29$
- $n = 15049$: $p = 101$ et $q = 149$

8.3 Exercice 3 : RSA avec un modulo commun

On suppose que deux entités Alice et Bob utilisent un schéma de chiffrement RSA avec le même modulo n et des exposants publics différents e_1 et e_2.

Énoncé Montrer qu'Alice peut déchiffrer les messages adressés à Bob.

Correction Alice connaît n et $\varphi(n)$. De plus la clé publique de Bob est (e_2, n).
Alice est donc capable de calculer $d_2 = e_2^{-1} \mod \varphi(n)$, i.e., Alice peut retrouver la clé privée de Bob et ainsi déchiffrer les messages qui lui sont destinés.

Énoncé Montrer qu'un attaquant Charlie peut déchiffrer un message envoyé à la fois à Alice et Bob, à condition d'avoir $\text{pgcd}(e_1, e_2) = 1$.

Correction L'attaquant connaît les clés publiques de Alice et Bob : (e_1, n) et (e_2, n).
En utilisant l'algorithme d'Euclide étendu, il est capable de calculer u, v tels que $e_1 u + e_2 v = 1$.
Imaginons maintenant que Robert envoie un message à Alice et à Bob :

- Robert envoie $C_A = M^{e_1} \mod n$ à Alice
- Robert envoie $C_B = M^{e_2} \mod n$ à Bob

L'attaquant intercepte ces messages et effectue le calcul suivant :

$$\begin{aligned} C_A^u \times C_B^v &= (M^{e_1})^u \times (M^{e_2})^v \mod n \\ &= M^{e_1 u} \times M^{e_2 v} \mod n \\ &= M^{e_1 u + e_2 v} \mod n \\ &= M \mod n \end{aligned}$$

L'attaquant est donc capable de retrouver le message envoyé à Alice et Bob.

8.4 Exercice 4 : Génération de clés RSA

Dans cet exercice, on montre que lors de la génération des clés RSA, on doit prendre garde au fait que $q-p$ ne soit pas trop petit, où $n = p \times q$ et $q > p$.

Énoncé Supposons que $q - p = 2d > 0$ et $n = p \times q$. Montrer que $n + d^2$ est un carré parfait.

Correction Le calcul permettant de montrer que $n + d^2$ est un carré parfait est le suivant :

$$\begin{aligned} n + d^2 &= pq + \frac{(q-p)^2}{4} \\ &= pq + \frac{q^2}{4} - \frac{2pq}{4} + \frac{p^2}{4} \\ &= \frac{p^2}{4} + \frac{2pq}{4} + \frac{q^2}{4} \\ &= \frac{1}{4}(p+q)^2 \end{aligned}$$

Il nous reste à montrer que $\frac{1}{4}(p+q)^2$ est un carré parfait, i.e., que c'est un entier.
Par hypothèse, p et q sont deux entiers premiers (donc impairs) ainsi $\exists k_p, k_q \in \mathbb{N}$ tels que

$$\begin{aligned} p &= 2k_p + 1 \\ q &= 2k_q + 1 \end{aligned}$$

Donc $\frac{1}{4}(p+q)^2 = \frac{1}{4}(2k_p + 1 + 2k_q + 1)^2 = \frac{1}{4}(2(k_p + k_q + 1))^2 = \frac{4}{4}(k_p + k_q + 1)^2 = (k_p + k_q + 1)^2 \in \mathbb{N}$.

Énoncé Étant donné un entier n, qui est le produit de deux nombres premiers impairs, et étant donné un petit entier d tel que $n + d^2$ soit un carré parfait, montrer comment cette information peut être utilisée pour factoriser n.

Correction Soit $\Delta = n + d^2 \Rightarrow \Delta - d^2 = n$ et $\delta = \sqrt{\Delta}$.
Mais de plus $\Delta - d^2 = \delta^2 - d^2 = (\delta - d)(\delta + d)$.
En réduisant cette égalité modulo n, on obtient $(\delta - d)(\delta + d) = 0 \mod n$.
Ainsi, on obtient une décomposition de n. Par unicité de la décomposition en facteurs premiers, on a la décomposition de n.

Énoncé Utiliser cette technique pour factoriser $n = 2189284635403183$.

Correction

$2189284635403183 + 0^2$	=	2189284635403183	ce n'est pas un carré parfait
$2189284635403183 + 1^2$	=	2189284635403184	ce n'est pas un carré parfait
$2189284635403183 + 2^2$	=	2189284635403187	ce n'est pas un carré parfait
$2189284635403183 + 3^2$	=	2189284635403192	ce n'est pas un carré parfait
$2189284635403183 + 4^2$	=	2189284635403199	ce n'est pas un carré parfait
$2189284635403183 + 5^2$	=	2189284635403208	ce n'est pas un carré parfait
$2189284635403183 + 6^2$	=	2189284635403219	ce n'est pas un carré parfait
$2189284635403183 + 7^2$	=	2189284635403232	ce n'est pas un carré parfait
$2189284635403183 + 8^2$	=	2189284635403247	ce n'est pas un carré parfait
$2189284635403183 + 9^2$	=	2189284635403264	la racine carrée est 46789792

On obtient alors $n = 46789783 \times 46789801$

8.5 Exercice 5 :Chiffrement RSA itéré

Un des outils du cryptanalyste consiste à rechiffrer plusieurs fois le message chiffré. Il arrive que, pour un cryptosystème apparemment solide, on retrouve le message clair après un petit nombre d'applications de la fonction de chiffrement. Ceci constitue évidemment une grave faiblesse du schéma.

Énoncé Soit $n = 35$ un modulo RSA, m le message clair et c le message chiffré. Vérifier que $E(c) = m^{e^2} = m$ pour tout exposant e légitime (c'est à dire tout entier e tel que $0 < e < \varphi(35)$ et $\text{pgcd}(e, \varphi(35)) = 1$), de sorte que le système n'est pas sûr du tout.

Correction Afin de savoir quels sont les e possibles, il faut examiner le groupe $(\mathbb{Z}/\varphi(35)\mathbb{Z})^*$:

$$(\mathbb{Z}/\varphi(35)\mathbb{Z})^* = \{5, 7, 11, 13, 17, 19, 23\}$$

Intéressons-nous aux carrés de ces éléments :

$$\begin{array}{rcrcl}
5^2 & = & 25 & = & 24 \times 1 + 1 \\
 & & & & \text{donc } 5^{-1} = 5 \mod \varphi(35) \\
7^2 & = & 49 & = & 24 \times 2 + 1 \\
 & & & & \text{donc } 7^{-1} = 7 \mod \varphi(35) \\
11^2 & = & 121 & = & 24 \times 5 + 1 \\
 & & & & \text{donc } 11^{-1} = 11 \mod \varphi(35) \\
13^2 & = & 169 & = & 24 \times 7 + 1 \\
 & & & & \text{donc } 13^{-1} = 13 \mod \varphi(35) \\
17^2 & = & 289 & = & 24 \times 12 + 1 \\
 & & & & \text{donc } 17^{-1} = 17 \mod \varphi(35) \\
19^2 & = & 361 & = & 24 \times 15 + 1 \\
 & & & & \text{donc } 19^{-1} = 19 \mod \varphi(35) \\
23^2 & = & 529 & = & 24 \times 22 + 1 \\
 & & & & \text{donc } 23^{-1} = 23 \mod \varphi(35)
\end{array}$$

Énoncé Expliquer comment mettre en place une "attaque par cycle" de façon à déchiffrer un message chiffré c, étant donnée la clé publique correspondante (n, e).

Correction En considérant les $\mathcal{C} = \{C^{e^i} \mod n, i \in \mathbb{Z}\}$, il existe un certain k tel que $C^{e^k} = C \mod n$. Puisque le chiffrement RSA définit une bijection, nécessairement, $C^{e^{k-1}} = M \mod n$.
Pour que cette attaque marche, il faut prouver que ce qui précède arrive effectivement, i.e.,
que l'on a effectivement un cycle dans l'ensemble des $\mathcal{C}$.
Pour montrer cela, on va considérer l'ensemble $\mathcal{E} = \{e^i \mod \varphi(n), i \in \mathbb{Z}\}$.
L'idée est que si on a un cycle dans $\mathcal{E}$, on aura nécessairement un cycle dans $\mathcal{C}$.
En effet, si on a un cycle dans $\mathcal{E}$, on aura deux éléments $e_1, e_2 \in \mathcal{E}$ tels que $e_1 = e_2$ ainsi,

$$C^{e_1} = C^{e_2}.$$

Or, $\mathcal{E} \subseteq (\mathbb{Z}/\varphi(n)\mathbb{Z})^*$, donc $|\mathcal{E}| \leq \varphi(\varphi(n))$. Conclusion, $\mathcal{E}$ est un ensemble fini.
D'où, nécessairement, $\exists n\mathbb{N}$ tel que $\exists i \in \mathbb{N}$ avec $i < n$ et $e^n = e^i \mod \varphi(n)$.
Ceci prouve que $e^{n-i} = 1 \mod \varphi(n)$.
On conclut donc que $\mathcal{E}$ contient un cycle et que de plus ce cylce reboucle sur la valeur 1.
On a donc $C^{e^{n-i}} = C^1 = C \mod n$. Or le chiffrement RSA définit une bijection, donc,
nécessairement, $C^{e^{n-i-1}} = M \mod n$.

Ceci permet de conclure que l'on peut donc déchiffrer le message sans connaître d! Il suffit, pour ce faire, de parcourir l'ensemble des valeurs de $\mathcal{E}$ et de les compter, dès que l'on a un cycle, on a la valeur du cardinal de $\mathcal{E}$, et d'après ce qui précède, $C^{e^{|\mathcal{E}|}} = C^1 \mod n$.
Comme le chiffrement RSA définit une bijection, $C^{|\mathcal{E}|-1} = M \mod n$.

8.5.1 Question 3

Énoncé Essayer de trouver les conditions qui rendent l'attaque possible. Proposer une méthode de génération des paramètres RSA, qui permette d'empêcher cette attaque. [Remarque : on peut montrer que la probabilité de succès d'une telle "attaque par cycle" est négligeable si p et q sont choisis aléatoirement et suffisamment grands.]

Correction Le principe de l'attaque a été expliqué. Il nous faut maintenant donner des conditions pour que cette attaque ne soit pas réalisable.
Analysons l'ensemble $\mathcal{E}$:

- $1 = e^0 \in \mathcal{E}$
- Soient $x, y \in \mathcal{E}$. Par définition, $\exists n_x, n_y \in \mathbb{Z}$ tels que $x = e^{n_x}$ et $y = e^{n_y}$.
  Ainsi, $x.y = e^{n_x}.e^{n_y} = e^{n_x+n_y}$. Or, $n_x + n_y \in \mathbb{Z}$; ceci prouve que par définition, $x.y \in \mathcal{E}$.

Conclusion, $\mathcal{E}$ est un sous-groupe de $(\mathbb{Z}/\varphi(n)\mathbb{Z})^*$.
L'attaque repose sur le fait qu'il est calculatoirement possible de parcourir $\mathcal{E}$, i.e., $|\mathcal{E}|$ n'est pas trop grand. Or, d'après le théorème de Lagrange, $|\mathcal{E}|$ divise $|(\mathbb{Z}/\varphi(n)\mathbb{Z})^*| = \varphi(\varphi(n))$.
Ainsi, pour que cette attaque soit impossible, il faut que $\varphi(\varphi(n))$ n'admette pas de petits
diviseurs.
Dans la pratique, il est impossible que $\varphi(\varphi(n))$ n'admette pas de petits diviseurs. En effet, même en utilisant des nombres premiers dits "nombres premiers sûrs", i.e., de la forme $2p + 1$ où $p \in \mathcal{P}$, $\varphi(\varphi(n))$ admettra de petits diviseurs.
En effet, en construisant $n = (2p + 1)(2q + 1)$ où $2p + 1, 2q + 1, p, q \in \mathcal{P}$, on aura $\varphi(n) = 4pq$.
Et donc, $\varphi(\varphi(n)) = \varphi(4pq) = \varphi(2^2pq) = (2^2 - 2^1)(p - 1)(q - 1) = 2(p - 1)(q - 1)$.
Puisque $p, q \in \mathcal{P}$, $\exists k_p, k_q \in \mathbb{Z}$ tels que $p = 2k_p + 1$ et $q = 2k_q + 1$; d'où,

$$\varphi(\varphi(n)) = 8k_pk_q.$$

Conclusion, même avec des nombres premiers sûrs, $\varphi(\varphi(n))$ admet de petits diviseurs (2 par exemple...). La solution est de prendre un e résistant, i.e., un e qui n'engendre pas un $\mathcal{E}$ de trop petit cardinal. Pour ceux qui souhaitent en savoir d'avantage sur la question, intéressez-vous à la théorie des nombres, mais sinon, vous pouvez considérer que prendre e aléatoire suffit.

9 Feuille d'Exercices 9

9.1 Exercice 1 : L'âge du capitaine

Énoncé On cherche à connaître l'âge du capitaine. On a uniquement les indices suivants :
- Il y a un an, son âge était un multiple de 3.
- Dans 2 ans, son âge sera un multiple de 5.
- Dans 4 ans, ce sera un multiple de 7.

Énoncé Quel est l'âge du capitaine ?

Correction Modélisons ce que nous donne l'énoncé :

$$\begin{aligned} x-1 &= 0 \mod 3 \\ x-2 &= 0 \mod 5 \\ x-4 &= 0 \mod 7 \end{aligned}$$

Ce qui peut être reformulé de la manière suivante :

$$x = 1 \mod 3 \quad (1)$$
$$x = 3 \mod 5 \quad (2)$$
$$x = 3 \mod 7 \quad (3)$$

On va utiliser le théorème des restes chinois afin de résoudre cet exercice.
Pour les équations (1) et (2), calculons les coefficients de Bezout :

$$5 \times 2 - 3 \times 3 = 1.$$

Ainsi en appliquant le théorème des restes chinois, on obtient une quatrième équation :

$$x = 5 \times 2 \times 1 + 3 \times (-3) \times 3 \mod (3 \times 5) = 13 \mod 15$$

On considère alors le système suivant :

$$\begin{aligned} x &= 13 \mod 15 \\ x &= 3 \mod 7 \end{aligned}$$

On va refaire le raisonnement précédent. Les coefficients de Bezout sont :

$$15 \times 1 - 7 \times 2 = 1$$

Ainsi, en utilisant le théorème des restes chinois, on obtient :

$$x = 7 \times (-2) \times 13 + 15 \times 1 \times 3 \mod (15 \times 7) = -137 \mod 105 = 73 \mod 105$$

Conclusion : le capitaine a 73 ans.

9.2 Exercice 2 : Attaque passive contre le cryptosystème RSA

Soit $n = p \times q$ un entier, produit de deux nombres premiers p et q. On considère d un exposant secret RSA de k bits, dont l'écriture binaire est $d = d_{k-1}2^{k-1} + d_{k-2}2^{k-2} + ... + d_1 2 + d_0$ (avec $d_{k-1} = 1$).
La fonction $f : Z/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$, définie par $f(x) = x^d \bmod n$, est implémentée dans une carte à puce de la manière suivante (dite "square-and-multiply-always") :

Input: x, d, n
Output: $y_0 = x^d \bmod n$
$y_0 := x$

for $i = k - 2$ down to 0 **do**
  $y_0 := y_0^2 \bmod n$
  $y_1 := y_0 \times x \bmod n$
  $y_0 := y_{d_i}$
end for
Return y_0

1. Expliquer l'avantage de la méthode "square-and-multiply-always" sur la méthode "naïve" suivante :

   Input: x, d, n
   Output: $y = x^d \bmod n$
     $y := x$
     for $i = k - 2$ down to 0 **do**
       $y := y^2 \bmod n$
       if $d_i = 1$ **then** $y := y \times x \bmod n$
     end for
     Return y

   Correction Si un attaquant examine les courbes de consommation électrique, alors, dans le cas de la méthode naïve, l'attaquant sera capable de détecter si la multiplication a été effectuée. Autrement dit, il est capable de connaître la valeur du bit d_i a chacune des étapes de la boucle.
   Dans le cas de la méthode square-and-mutliply-always, l'attaquant n'est pas capable de distinguer la valeur de d_i par simple observation.

2. On se place maintenant dans l'hypothèse suivante : l'attaquant est capable de détecter (par exemple par observation des courbes de consommation électrique) que la carte effectue deux fois le même calcul. Plus précisément, si la carte calcule $u^2 \bmod n$ (à un instant t_1), puis $v^2 \bmod n$ (à un instant $t_2 > t_1$), l'attaquant n'est pas capable d'en déduire la valeur de u ni celle de v, mais est capable de dire si $u = v$.
   On suppose en outre que l'attaquant peut effectuer le calcul $x^d \bmod n$ avec les valeurs x de son choix. Montrer qu'il peut alors retrouver la valeur de l'exposant secret d, dans le cas de l'algorithme "square-and-multiply-always" (pour cette attaque de type SPA, on pourra considérer la suite des valeurs intermédiaires, successivement pour les valeurs d'entrée x et $x' = x^2 \bmod n$)

Correction De proche en proche, il est possible de découvrir quel calcul a été effectué par la carte. Ce qui permet par une simple observation de retrouver le paramètre d.

9.3 Exercice 3 : Attaque par injection de faute

Dans la suite, on s'intéresse au cryptosystème RSA. On note :
- $n = p \times q$ le module RSA ;
- e l'exposant public ;
- d l'exposant privé.

1. Rappeler le théorème des restes chinois ;

   Correction Soient $m_1, ..., m_k$ des entiers deux à deux premiers entre eux (*i.e.* $gcd(m_i, m_j) = 1$ pour $i \neq j$ et $i, j \in \{1, k\}$). Pour tout entier $r_1, ..., r_k$, il existe un entier x (unique modulo $m = \prod_{i=1}^{k} m_i$) tel que :

$$\begin{aligned} x &\equiv r_1 \mod m_1 \\ &... \\ x &\equiv r_k \mod m_k \end{aligned}$$

Alors $x = \sum_{i=1}^{k} u_i \cdot M_i \cdot r_i$, où $M_i = \frac{\prod_{j=1}^{k} m_j}{m_i}$ et $u_i = (M_i)^{-1} \mod m_i$.

2. Expliquer comment celui-ci peut être utilisé dans le cadre du déchiffrement ;

   Correction Afin que quelqu'un dechiffre un message, il doit connaitre la clé privée. Il connait donc aussi p et q. Il peut donc calculer :
   — $d_p = d \ mod \ (p-1) = e^{-1} \ mod \ (p-1) \Rightarrow d_p \cdot e = 1 \ mod \ (p-1)$
   — $d_q = d \ mod \ (q-1) = e^{-1} \ mod \ (q-1) \Rightarrow d_q \cdot e = 1 \ mod \ (q-1)$
   et donc :
   — $m_p = c^{d_p} \ mod \ p = m^{e \cdot d_p} \ mod \ p = m^{k_p \cdot (p-1)+1} \ mod \ p = m \ mod \ p$, avec
   — $m_q = c^{d_q} \ mod \ q = m^{e \cdot d_q} \ mod \ q = m^{k_q \cdot (p-1)+1} \ mod \ q = m \ mod \ q$
   En utilisant le théorème de reste chinois, on obtient l'unique message $m \in \mathbb{Z}_n$.

3. Donner le facteur d'accélération résultant de l'utilisation du théorème des restes chinois par rapport à une implémentation "classique".

   Correction Si l'on conserve les valeurs de p et q, on peut ramener le calcul $c^d \ mod \ n$ à deux exponentiation d'exposant de taille divisée par 2 et avec un module de taille également divisée par 2. Ceci nous permet d'obtenir un facteur d'accélération de 4 par rapport à une implémentation classique.

Énoncé On suppose maintenant que la carte à puce implémente le déchiffrement à l'aide du théorème des restes chinois. Sous l'action d'un laser, il est possible de changer la valeur d'un registre en une valeur complètement aléatoire. Supposons que l'attaquant soit capable de changer la valeur du registre lors de la fin du déchiffrement modulo q ; le résultat du déchiffrement est alors "faux".

5. Montrer comment il peut factoriser le module RSA à l'aide de ce déchiffrement "faux".

Correction Si une faute est introduite pendant le déchiffrement modulo q, alors une valeur erronée m'_q est utilisée lors de l'étape de recombinaison CRT. Ceci produit un message erroné m'. Comme $m \equiv m_p \mod p$ et $m \equiv m_q \mod q$, on a $m' \equiv m \mod p$, et $m' \not\equiv m \mod q$. Donc, si $q \nmid (m - m')$, on obtient le paramètre secret q en calculant $gcd(m - m', n)$.