CRYPTOLOGIE – Banque d'exercices

Version de 07/2025

Table des matières

1 Chiffrements symétriques anciens

1.1 César, Hill, ...

ChifSymAnc-01

Exercice 1. Chiffrement de César

Mathématiquement, on peut décrire ce système de la manière suivante : les lettres de l'alphabet étant codées par les entiers modulo 26 : `'A'` $= 0, \ldots,$ `'Z'` $= 25$. Soient $M = m[0]m[1]m[2]\ldots m[n-1]$ le texte clair (de longueur n), $C = c[0]c[1]c[2]\ldots c[n-1]$ le message chiffré, et un entier k pour la clé. Les symboles $m[i]$ (resp. $c[i]$) désignent les codes numériques des lettres du texte clair (resp. chiffré). On a :

$$c[i] = m[i] + k \bmod 26$$

Voici un texte chiffré obtenu avec la clé $k = 7$:

```
QLZBP  ZHSVU  KYLZK  HUZBU  LKLZY
BLZSL  ZWSBZ  TPZLY  HISLZ  KLSHC
PSSLQ  LTHYJ  OLLUT  LKLTH  UKHUA
JVTTL  UAZLK  PABYP  UVPYL  UHUNS
HPZ
```

Retrouver le texte clair.

La correspondance entre les caractères clairs minuscules et chiffrés majuscules est la suivante, puisque la clé $k = 7$ donne la valeur du décalage (le chiffrement est une permutation circulaire)

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
```

Il suffit de substituer caractère par caractère

```
jesui  salon  dresd  ansun  edesr  uesle  splus  miser  ables  delav
illej  emarc  heenm  edema  ndant  comme  ntsed  ituri  noire  nangl
ais
```

Ce qui donne, une fois reconstitué :

« Je suis à Londres dans une des ruelles les plus misérables de la ville. Je marche en me demandant comment se dit urinoir en anglais »

ChifSymAnc-02

Exercice 2. Chiffrement affine

Le chiffrement affine est une technique de chiffrement par substitution simple qui consiste à substituer à chaque symbole $m[i]$ du message clair, le symbole chiffré $c[i]$ calculé par

$$c[i] = (a \times m[i] + b) \bmod 26$$

où a et b sont deux entiers compris entre 0 et 25, a devant être premier avec 26. Le déchiffrement s'effectue en calculant

$$m[i] = a^{-1} \times (c[i] - b) \bmod 26$$

où a^{-1} désigne l'inverse de a modulo 26, c'est-à-dire l'unique entier compris entre 0 et 25 tel que $a \times a^{-1} = 1 \bmod 26$. Ce cryptogramme a été obtenu par chiffrement affine avec les clés $a = 15$, $b = 7$.

```
MPRVX  RHQJU  ACPRA  HURVU  PAPRC  VPRQP  RYQVR  FXRPC  HWQPR  APQHK
XQQPM  PFHCL  IPPUF  PAPFH  UAHUG  LJFFP  UGRPA  XGVCX  UJXCP  UHUTQ
HXR
```

Retrouver le texte clair.

Il s'agit d'une substitution mono alphabétique plus compliquée. On a

$$c[i] = (15 \times m[i] + 7) \bmod 26$$

Ce qui donne la correspondance suivante :

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
7	22	11	0	15	4	19	8	23	12	1	16	5	20	9	24	13	2	17	6	21	10	25	14	3	18
H	W	L	A	P	E	T	I	X	M	B	Q	F	U	J	Y	N	C	R	G	V	K	Z	O	D	S

et le déchiffré

```
jesui  salon  dresd  ansun  edesr  uesle  splus  miser  ables  delav
illej  emarc  heenm  edema  ndant  comme  ntsed  ituri  noire  nangl
ais
```

ChifSymAnc-03

Exercice 3. **Chiffrement de Hill**

Les lettres de l'alphabet sont codées modulo 26, et les calculs sont faits modulo 26. L'algorithme est un chiffrement par blocs de n lettres. La clé est une matrice carrée K de taille n inversible modulo 26. À noter que la valeur de n fait partie de la clé, c'est-à-dire est inconnue de l'attaquant. Le chiffrement consiste en un produit matrice-vecteur :

$$C = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} = KM = \begin{pmatrix} k_{1,1} & \dots & k_{1,n} \\ \vdots & & \vdots \\ k_{n,1} & \dots & k_{n,n} \end{pmatrix} \begin{pmatrix} m_1 \\ \vdots \\ m_n \end{pmatrix}$$

1. Quelle est l'opération de déchiffrement ?

A titre d'exemple, on pose $n = 2$ et $K = \begin{pmatrix} 5 & 12 \\ 1 & 3 \end{pmatrix}$.

2. Calculer le chiffré du message $\begin{pmatrix} 10 \\ 21 \end{pmatrix}$
3. Calculer la matrice de déchiffrement et vérifier le résultat de la question 2.

On étudie maintenant la cryptanalyse dans le cas général. On suppose que l'attaquant connaît la valeur de n (qui en principe fait partie de la clé) et connait n couples (clair, chiffré) $(M_1, C_1), \dots, (M_n, C_n)$, et que la matrice X formée des vecteurs colonnes $M_1, \dots, M_n$ est inversible.

4. Montrer que l'attaquant peut calculer la matrice K.
5. Application numérique avec $n = 2$, $M_1 = \begin{pmatrix} 2 \\ 9 \end{pmatrix}$, $C_1 = \begin{pmatrix} 11 \\ 11 \end{pmatrix}$, $M_2 = \begin{pmatrix} 7 \\ 3 \end{pmatrix}$, $C_2 = \begin{pmatrix} 11 \\ 23 \end{pmatrix}$.

1. Puisque $C = KM$, et que K est inversible, on a $M = K^{-1}C$. Le déchiffrement consiste à multiplier par la matrice inverse de celle du chiffrement.
2. On effectue un classique produit matrice-vecteur :

$$\begin{pmatrix} 5 & 12 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} 10 \\ 21 \end{pmatrix} = \begin{pmatrix} 5 \cdot 10 + 12 \cdot 21 \\ 1 \cdot 10 + 3 \cdot 21 \end{pmatrix} = \begin{pmatrix} 16 \\ 21 \end{pmatrix}$$

3. Pour une matrice carrée d'ordre 2, la formule la plus simple pour calculer l'inverse est celle de la comatrice.

Sans entrer les détails, on rappelle que :

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \underbrace{\frac{1}{ad-bc}}_{\det} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

expression valide lorsque le déterminant $ad - bc$ est inversible modulo 26 (ce qui est plus exigeant que d'être non nul modulo 26).

Dans l'exemple numérique, le déterminant vaut $3 \bmod 26$, qui est bien inversible d'inverse 9 puisque $3 \times 9 = 1 \bmod 26$. Il vient

$$K^{-1} = \frac{1}{5 \times 3 - 1 \times 12} \begin{pmatrix} 3 & -12 \\ -1 & 5 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 3 & -12 \\ -1 & 5 \end{pmatrix} = 9 \begin{pmatrix} 3 & 14 \\ 25 & 5 \end{pmatrix} = \begin{pmatrix} 1 & 22 \\ 17 & 19 \end{pmatrix}$$

et on déchiffre le résultat de la question 2 :

$$\begin{pmatrix} 1 & 22 \\ 17 & 19 \end{pmatrix} \begin{pmatrix} 16 \\ 21 \end{pmatrix} = \begin{pmatrix} 10 \\ 21 \end{pmatrix}$$

4. Comme $C_i = KM_i$, pour $i = 1, \ldots, n$, on écrit sous forme matricielle

$$K \underbrace{(M_1 | \ldots | M_n)}_{X} = \underbrace{(C_1 | \ldots | C_n)}_{Y}$$

La matrice X est supposée inversible. On a donc $K = YX^{-1}$, ce qui permet le recouvrement de clé.

5. La matrice $\begin{pmatrix} 2 & 7 \\ 9 & 3 \end{pmatrix}$ est inversible car son déterminant est égal à $2 \times 3 - 7 \times 9 = 21 \bmod 26$ qui est inversible (d'inverse 5). Ainsi

$$\begin{pmatrix} 2 & 7 \\ 9 & 3 \end{pmatrix}^{-1} = \frac{1}{21} \begin{pmatrix} 3 & -7 \\ -9 & 2 \end{pmatrix} = 5 \begin{pmatrix} 3 & 19 \\ 17 & 2 \end{pmatrix} = \begin{pmatrix} 15 & 17 \\ 7 & 10 \end{pmatrix}$$

Et finalement

$$K = \begin{pmatrix} 11 & 11 \\ 11 & 23 \end{pmatrix} \begin{pmatrix} 2 & 7 \\ 9 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 11 & 11 \\ 11 & 23 \end{pmatrix} \begin{pmatrix} 15 & 17 \\ 7 & 10 \end{pmatrix} = \begin{pmatrix} 8 & 11 \\ 14 & 1 \end{pmatrix}$$

1.2 Vigenère

ChifSymAnc-04

Exercice 4. Chiffrement de Vigenère

1. Chiffrer le message :
   `‘‘ As tu un animal de compagnie ?’’`
   avec la clé $K =$`‘‘Miaou’’`.
2. Déchiffrer le message suivant :
   `‘‘ Aci ih pqpzipwcim viuby.’’`
   avec la même clé.

1. `‘‘Matiozinwgmtdswaupoazqe ’’`
2. `‘‘Oui un diplodocus jaune.’’`

ChifSymAnc-05

Exercice 5. Cryptanalyse du chiffrement de Vigenère à chiffré connu

On considère un texte chiffré $C = c[0]c[1]\dots c[n-1]$ de longueur n produit par un chiffrement de Vigenère à partir du texte clair $M = m[0]m[1]\dots m[n-1]$ et d'un mot clé $K = k[0]k[1]\dots k[\ell-1]$ (on rappelle que ℓ fait partie du secret). On suppose que n est significativement plus grand que ℓ.

1. Montrer qu'à clair connu, il est immédiat de retrouver la clé.

2. Montrer que si le clair M est aléatoire, il est impossible de retrouver la clé ni le clair.

On suppose dorénavant que le clair possède une "structure statistique", par exemple que c'est un texte en français (fréquence d'appartition des lettres non équidistribuée).

Dans un premier temps, on suppose déterminée la taille ℓ de la clé. On construit les textes suivants à partir du chiffré :

$$\begin{aligned} Y_0 &= c[0]c[\ell]c[2\ell]\dots \\ Y_1 &= c[1]c[\ell+1]c[2\ell+1]\dots \\ &\dots \\ Y_{\ell-1} &= c[\ell-1]c[2\ell-1]c[3\ell-1]\dots \end{aligned}$$

et on note

$$\begin{aligned} X_0 &= m[0]m[\ell]m[2\ell]\dots \\ X_1 &= m[1]m[\ell+1]m[2\ell+1]\dots \\ &\dots \\ X_{\ell-1} &= m[\ell-1]m[2\ell-1]m[3\ell-1]\dots \end{aligned}$$

les "sous-textes" clairs correspondants.

3. Montrer que pour i donné, la transformation $X_i \mapsto Y_i$ est un chiffrement de César. Si δ est le décalage, on notera $Y_i = X_i^{(+\delta)}$. En déduire une attaque consistant en ℓ attaques d'un chiffrement de César.

Afin d'améliorer cette complexité, on introduit un outil statistique fondamental : l'*indice de coïncidence*. Nous admettrons le résultat suivant.

> **Théorème** *Soient x et y deux chaînes de caractères dont les fréquences d'apparition des lettres sont $(f_0,\dots,f_{25})$ et $(f'_0,\dots,f'_{25})$. On définit :*
>
> $$IC(x) = \sum_{i=0}^{25} f_i^2 \ , \quad MIC(x,y) = \sum_{i=0}^{25} f_i f'_i$$
>
> *Alors :*
>
> *(1) Si x est un texte aléatoire (ou au moins à lettres équidistribuées), $IC(x) \approx 0,0385$.*
>
> *(2) Si x est un texte français, $IC(x) \approx 0,0778$.*
>
> *(3) Si x est un décalé de César d'un texte français, $IC(x) \approx 0,0778$.*
>
> *(4) Si x et y sont deux textes français, alors $MIC(x,y^{+\delta}) \approx 0.0778$ si $\delta = 0$, et $MIC(x,y) \leq 0,0450$ si $\delta \neq 0$.*

4. Soient $1 \leq i \leq \ell-1$. Montrer que $MIC(Y_0, Y_i^{+\delta}) \approx 0,0778$ si $\delta = k[0]-k[i]$, $\leq 0,0450$ sinon. En déduire une attaque dont la complexité est l'attaque d'un César et le calcul de $26(\ell-1)$ indices de coïncidence mutuel.

On montre maintenant comment l'on détermine (au préalable) la taille de clé.

5. On suppose que deux trigrammes de chiffré $c[i]c[i+1]c[i+2]$ et $c[j]c[j+1]c[j+2]$ sont égaux. On suppose que $j-i$ est multiple de ℓ. Montrer que les trigrammes de clair correspondants sont égaux, *ie* que $m[i]m[i+1]m[i+2] = m[j]m[j+1]m[j+2]$.

Cette propriété est mise à profit pour deviner la taille de la clé dans le test de Kasiski :

---

1. Identifier des trigrammes fréquents dans le chiffré.

\\ On fait le pari qu'ils chiffrent la même séquence de clair, donc les distances mutuelles sont multiples de ℓ.

2. calculer le pgcd des distances mutuelles.

Il existe des variantes où l'on repère des répétitions de polygrammes de taille 4, 5, etc.

Voici une autre méthode pour deviner la taille dela clé basée sur l'indice de coïncidence. On découpe le texte comme à la question 3 mais en faisant un "pari" ℓ^* sur la valeur de ℓ. Ainsi :

$$\begin{aligned} Y_0 &= c[0]c[\ell^*]c[2\ell^*]\dots \\ Y_1 &= c[1]c[\ell^*+1]c[2\ell^*+1]\dots \\ &\dots \\ Y_{\ell^*-1} &= c[\ell^*-1]c[2\ell^*-1]c[3\ell^*-1]\dots \end{aligned}$$

6. Montrer que si $\ell^* = \ell$ (c'est-à-dire le pari est bon), alors $IC(Y_i) \approx 0,0778$.

7. Montrer que si le pari est mauvais, $IC(Y_i) \approx 0,0385$.

1. Comme le chiffrement est une simple addition modulo 26, on a $k[i] = c[i] - m[i] \bmod 26$. La connaissance d'un couple (clair, chiffré) dont la taille est plus grande que celle de la clé suffit à retrouver cette dernière.

2. Si le clair est aléatoire, la seule connaissance du chiffré rend tous les motifs de clé équiprobables. On est dans un analogue du One Time Pad, mais modulo 26 au lieu de modulo 2. La sécurité du clair et de la clé sont donc parfaites.

3. Par définition $c[i] = m[i] + k[i \bmod \ell] \bmod 26$. Ainsi pour $0 \leq i \leq \ell - 1$, on a

$$c[i] = m[i] + k[i], c[i+\ell] = m[i+\ell] + k[i], \dots$$

et $X_i \mapsto Y_i$ est un chiffrement de César dont le décalcge est $k[i]$. Si on effectue ℓ attaques de chiffrement de César sur les "sous chiffrés" Y_i, on retrouve le clair. Cela peut se faire par attaque fréquentielle par exemple.

4. On a

$$MIC(Y_0, Y_i^{+\delta}) = MIC(X_0^{+k[0]}, X_i^{+k[i]+\delta}).$$

Par définition de MIC, on vérifie que pour tout d, $MIC(x^{+d}, y^{+d}) = MIC(x, y)$. Ainsi,

$$MIC(Y_0, Y_i^{+\delta}) = MIC(X_0, X_i^{+\delta+k[i]-k[0]})$$

L'application de (4) du théorème donne le résultat demandé.

On peut monter l'attaque suivante : avec $26(\ell - 1)$ calcul de MIC, on détermine les $k[0] - k[i]$ pour $i = 1, \dots, \ell - 1$. Puis on détermine $k[0]$ par un attaque de César sur Y_0, et on déduit les autres $k[i]$.

5. Par définition du chiffrement et du fait que $i \equiv j \bmod \ell$, on a

$$m[i] = c[i] - k[i \bmod \ell] = c[j] - k[j \bmod \ell] = m[j].$$

De même pour $m[i+1] = m[j+1]$ et $m[i+2] = m[j+2]$.

6. Si le pari est bon, Y_i est le décalé de César d'un texte français (de décalage $k[i]$), donc d'après (3) du théorème, $IC(Y_i) \approx 0,0778$.

7. Si le pari est mauvais, les Y_i ne sont pas des décalés de César, et pour un i donné, plusieurs symboles de clés interviennent. Il est vraisemblable que la chaîne obtenue ait une statistique ressemblant à celle d'une suite aléatoire. D'où le résultat.

ChifSymAnc-06

Exercice 6. **Cryptanalyse de Vigenère en pratique.**

Décrypter le texte suivant et retrouver la clé.

```
aejifurocdjabmwrncgxvrjifjpolwvrrywceqixderoyvbwyyriaypuubytefwxpceihpwizfpsishgzvohevqicdv
rsppzlzyluegfpedsglzngkfzcvuykebndfugfpjozytccsmevlzybleziydahllhuskfvlchafufmfzteopcobncrq
iynvshywceeotjegifcejybligifwffypkssgpkebacvvswpjtsfwvqiiyvmdltjobhphucheiavcehuchlsabxzens
kfzncoduobhpvnjcpuejcgieeotuobhpvnjcpuezudlijlpaugkfrupifkjimblaivzltxygfurllzsguyjlohzdmsl
wlifyyursbzdmoapaozcpwlsocuuaitjdsglzoizclihmllvoaplnsjwrnhymzebjwrnhypjufmpjdsoiaaavpjehkf
ztfuteeshwzbslevoivzelicdvmpfptegnpclskfvlchxrtfubleeopcobjzlrgotkqiywfnhllhuswpjtsfwvqicdv
scowvvskfzscoqwrsyejeayevnulpmeqydkezfphuchpdpfcdfnbyblobncrhwnblobumrnriyeeeoteoimofnbypev
wyovvwpcvqicofnbypevwyovlomfzvfyulseollbcoeaugkfrupifkjspzldfutjsohdcabixdefpzlsduccefxpcls
vtvnocxvecoxrlocxvesfwvegnqzdsfpvtgcgfugpzllstblexygfugflgrsmpetsiycadjpclslpmoZoeZobjpimoh
petswpjtsfwvqiywfnaueiaeophusfzepcocjuwnbleziykrokfvcsmevlzybligydfuzygvqicdfutzcvehmpdehyy
xrspptegnpclskffnsgaiigiyeeeozetfuszteozeapuyuobhphuwhzlsriyeeshgzerygzvfybliriyeeshgzerywr
sicgiexodhuoomfuhdfjqiufsoin
```

En appliquant la méthode exposée à l'exercice précédent, on obtient $K =$ raoul, et les paroles de la chanson de Georges Moustaki, "Sans la nommer" :

```
‘‘ Je voudrais, sans la nommer,
Vous parler d'elle
Comme d'une bien-aimée,
D'une infidèle,
Une fille bien vivante
Qui se réveille
A des lendemains qui chantent
Sous le soleil.
C'est elle que l'on matraque,
Que l'on poursuit que l'on traque.
C'est elle qui se soulève,
Qui souffre et se met en grève.
C'est elle qu'on emprisonne,
Qu'on trahit qu'on abandonne,
Qui nous donne envie de vivre,
Qui donne envie de la suivre
Jusqu'au bout, jusqu'au bout.
Je voudrais, sans la nommer,
Lui rendre hommage,
Jolie fleur du mois de mai
Ou fruit sauvage,
Une plante bien plantée
Sur ses deux jambes
```

Et qui traîne en liberté
Où bon lui semble.
C'est elle que l'on matraque,
Que l'on poursuit que l'on traque.
C'est elle qui se soulève,
Qui souffre et se met en grève.
C'est elle qu'on emprisonne,
Qu'on trahit qu'on abandonne,
Qui nous donne envie de vivre,
Qui donne envie de la suivre
Jusqu'au bout, jusqu'au bout.
Je voudrais, sans la nommer,
Vous parler d'elle.
Bien-aimée ou mal aimée,
Elle est fidèle
Et si vous voulez
Que je vous la présente,
On l'appelle
RÉVOLUTION PERMANENTE.
C'est elle que l'on matraque,
Que l'on poursuit que l'on traque.
C'est elle qui se soulève,
Qui souffre et se met en grève.
C'est elle qu'on emprisonne,
Qu'on trahit qu'on abandonne,
Qui nous donne envie de vivre,
Qui donne envie de la suivre
Jusqu'au bout, jusqu'au bout. ''

---

2 Génération de pseudo aléa

2.1 Tutoriels

ChifSymFlot-05

Exercice 7. Tuto LFSR

On considère un LFSR de taille 4 et de polynôme de rétroaction $T^4 + T + 1$. Ainsi, les coefficients de rétroaction sont $(a_0, a_1, a_2, a_3) = (1, 1, 0, 0)$. On désigne par $X_{\bullet,t} = (X_{0,t}, \ldots, X_{3,t})$ l'état interne à l'instant t.

1. Écrire les équations d'évolution du LFSR, donnant $X_{0,t+1}, \ldots, X_{3,t+1}$ en fonction de $X_{0,t}, \ldots, X_{3,t}$.
2. On prend l'état interne initial $(X_{0,0}, \ldots, X_{3,0}) = (1, 0, 1, 0)$. Calculer les 5 premiers états internes.
3. On note dorénavant l'état interne initial $(x_0, \ldots, x_3)$, en omettant le deuxième indice égal à 0. Donner l'expression algébrique des 5 premiers états internes en fonction de $(x_0, \ldots, x_3)$. Retrouver les résultats de la question 2.
4. Écrire les équations d'évolution sous forme matricielle : de la forme $X_{\bullet,t+1} = AX_{\bullet,t}$ où A est une matrice carrée binaire 4×4 que l'on précisera, et les vecteurs $X_{\bullet,t}$ et $X_{\bullet,t+1}$ sont des colonnes.
5. Vérifier que $X_{\bullet,t} = A^t X_{\bullet,0}$ pour $t = 0, 1, 2, \ldots$. Calculer les matrices A^2, A^3, A^4, A^5, et retrouver les résultats de la question 3.

1. On a
$$\begin{cases} X_{0,t+1} &= X_{1,t} \\ X_{1,t+1} &= X_{2,t} \\ X_{2,t+1} &= X_{3,t} \\ X_{3,t+1} &= X_{0,t} \oplus X_{1,t} \end{cases}$$

2. On part des équations de récurrence de la Q1. Simple application numérique

t	$X_{\bullet,t}$	bit de rétroaction à venir
0	1 0 1 0	$1 \oplus 0 = 1$
1	0 1 0 1	$0 \oplus 1 = 1$
2	1 0 1 1	$1 \oplus 0 = 1$
3	0 1 1 1	$0 \oplus 1 = 1$
4	1 1 1 1	$1 \oplus 1 = 0$
5	1 1 1 0	...

3. On part encore des équations de récurrence de la Q1, mais on fait cette fois les calculs littéraux.

t	$X_{\bullet,t}$				bit de rétroaction à venir
0	$(x_0$	x_1	x_2	$x_3)$	$x_0 \oplus x_1$
1	$(x_1$	x_2	x_3	$x_0 \oplus x_1)$	$x_1 \oplus x_2$
2	$(x_2$	x_3	$x_0 \oplus x_1$	$x_1 \oplus x_2)$	$x_2 \oplus x_3$
3	$(x_3$	$x_0 \oplus x_1$	$x_1 \oplus x_2$	$x_2 \oplus x_3)$	$x_3 \oplus (x_0 \oplus x_1) = x_0 \oplus x_1 \oplus x_3$
4	$(x_0 \oplus x_1$	$x_1 \oplus x_2$	$x_2 \oplus x_3$	$x_0 \oplus x_1 \oplus x_3)$	$(x_0 \oplus x_1) \oplus (x_1 \oplus x_2) = x_0 \oplus x_2$
5	$(x_1 \oplus x_2$	$x_2 \oplus x_3$	$x_0 \oplus x_1 \oplus x_3$	$x_0 \oplus x_2)$	...

4. L'écriture matricielle est un grand classique de la transcription d'un système d'équations linéaires. Ici
$$\underbrace{\begin{bmatrix} X_{0,t+1} \\ X_{1,t+1} \\ X_{2,t+1} \\ X_{3,t+1} \end{bmatrix}}_{X_{\bullet,t+1}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}}_{=A} \underbrace{\begin{bmatrix} X_{0,t} \\ X_{1,t} \\ X_{2,t} \\ X_{3,t} \end{bmatrix}}_{X_{\bullet,t}}$$
On rappelle qu'il est commode et traditionnel de noter les vecteurs en colonne.

5. C'est une récurrence immédiate sur t. Pour calculer les puissances de A, pas d'astuce particulière, mais des calculs un peu pédestres, il faut l'avouer.
$$A^2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix},\ A^3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix},\ A^4 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix},\ A^5 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$
En effectuant le produit de chacune de ces matrices contre le vecteur colonne
$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$
on retrouve les expressions algébriques obtenues à la Q3 (avec la même convention vecteurs colonnes).

ChifSymFlot-06

Exercice 8. Tuto registre filtré

On considère un LFSR R de taille 8 de polynôme de rétroaction $T^8 \oplus T^4 \oplus T^3 \oplus T \oplus 1$. Les coefficients de rétroaction sont donc $(a_0, \ldots, a_7) = (1, 1, 0, 1, 1, 0, 0, 0)$. Le registre est filtré par la fonction

$$g(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_2x_3 \oplus x_1x_2x_3$$

prélevant ses entrées sur les cellules n. 2, 5 et 7. Ainsi, à chaque instant,

$$z_t = g(R_{2,t}, R_{5,t}, R_{7,t})$$

On considère une clé jouet de 6 bits $K = (k_0, \ldots, k_5)$ et un marquant jouet de 2 bits $IV = (v_0, v_1)$. L'état interne initial ($t = 0$) est initialisé par une copie de la clé et du marquant juxtaposés :

$$X_{\bullet,0} = (k_0, \ldots, k_5, v_0, v_1)$$

Exprimer en fonction des k_i et v_i l'état du registre et le bit de sortie aux instants $1, 2, 3$

Pour chaque t, on donne l'état interne puis le bit de sortie.

$$\begin{array}{lcl}
t = 0 & : & [k_0, k_1, k_2, k_3, k_4, k_5, v_0, v_1] \\
 & & k_2k_5v_1 \oplus k_2 \oplus k_5v_1 \oplus k_5 \\
t = 1 & : & [k_1, k_2, k_3, k_4, k_5, v_0, v_1, k_0 \oplus k_1 \oplus k_3 \oplus k_4] \\
 & & k_0k_3v_0 \oplus k_0v_0 \oplus k_1k_3v_0 \oplus k_1v_0 \oplus k_3k_4v_0 \oplus k_3 \oplus k_4v_0 \oplus v_0 \\
t = 2 & : & [k_2, k_3, k_4, k_5, v_0, v_1, k_0 \oplus k_1 \oplus k_3 \oplus k_4, k_1 \oplus k_2 \oplus k_4 \oplus k_5] \\
 & & k_1k_4v_1 \oplus k_1v_1 \oplus k_2k_4v_1 \oplus k_2v_1 \oplus k_4k_5v_1 \oplus k_4 \oplus k_5v_1 \oplus v_1 \\
t = 3 & : & [k_3, k_4, k_5, v_0, v_1, k_0 \oplus k_1 \oplus k_3 \oplus k_4, k_1 \oplus k_2 \oplus k_4 \oplus k_5, k_2 \oplus k_3 \oplus k_5 \oplus v_0] \\
 & & k_0k_2k_5 \oplus k_0k_2 \oplus k_0k_3k_5 \oplus k_0k_3 \oplus k_0k_5v_0 \oplus k_0v_0 \oplus k_0 \oplus k_1k_2k_5 \oplus k_1k_2 \oplus k_1k_3k_5 \oplus k_1k_3 \\
 & & \oplus k_1k_5v_0 \oplus k_1v_0 \oplus k_1 \oplus k_2k_3k_5 \oplus k_2k_3 \oplus k_2k_4k_5 \oplus k_2k_4 \oplus k_3k_4k_5 \oplus k_3k_4 \oplus k_3k_5v_0 \oplus k_3k_5 \\
 & & \oplus k_3v_0 \oplus k_4k_5v_0 \oplus k_4v_0 \oplus k_4 \oplus k_5
\end{array}$$

2.2 Exemples d'attaques

ChifSymFlot-01

Exercice 9. Distinguer une suite binaire biaisée d'une suite aléatoire

Soit $x_1 \ldots x_n$ une suite binaire. Un attaquant veut distinguer s'il s'agit d'une suite biaisée de biais ε ("real") ou d'un suite parfaitement aléatoire ("random"). On modélise le problème comme la réalisation d'une suite de variables aléatoires de Bernoulli indépendantes identiquement distribuées et on teste si le paramètre est $1/2 + \varepsilon$ ou $1/2$.

1. Quelle est l'espérance du nombre de "1" dans le cas "real" ? Dans le cas "random" ?

L'attaquant procède par vote majoritaire : si le nombre de "1" est supérieur ou égal à $n(1/2+\varepsilon/2)$, il déclare "real", sinon il déclare "random". Son avantage est défini par

$$\begin{aligned}\mathbf{Adv}^{rr}(A) = &\, Pr(A \text{ déclare "real" } | \text{ le cas est "real"}) \\ - &\, Pr(A \text{ déclare "real' } | \text{ le cas est "random"})\end{aligned}$$

On rappelle les inégalités de Chernoff :

Théorème *Soient $X_1, \ldots, X_n$ une suite de v.a. i.i.d. selon une loi de Bernoulli de paramètre p. Soit $Y = \dfrac{X_1 + \cdots + X_n}{n}$. Pour tout $\alpha > 0$, on a*

$$Pr(Y \geq p + \alpha) \leq e^{-2n\alpha^2}, \quad Pr(Y \leq p - \alpha) \leq e^{-2n\alpha^2}, \quad Pr(|Y - p| \leq \alpha) \leq 2e^{-2n\alpha^2}$$

2. Montrer que $\mathbf{Adv}^{rr}(A) \geq 1 - 2e^{-n\varepsilon^2/2}$.

3. Calculer n minimal pour que $\mathbf{Adv}^{rr}(A) \geq 1/2$.

1. Le nombre de "1" dans le cas réel (resp. aléatoire) est une variable aléatoire qui suit une loi binomiale de paramètres n et $1/2+\varepsilon$ (resp. n et $1/2$). Ainsi, son espérance est égale à $n(1/2+\varepsilon)$ (resp. $n/2$)

2. On note Y la proportion du nombre de "1" obtenus. Le vote majoritaire consiste à comparer Y à $1/2+\varepsilon/2$. On peut appliquer facilement les inégalités de Chernoff.

Si le monde est "real", les X_i suivent une loi de Bernoulli de paramètre $1/2+\varepsilon$. Ainsi :

$$\Pr(Y \geq 1/2 + \varepsilon/2) = 1 - \Pr(Y \leq (1/2+\varepsilon) - \varepsilon/2) \geq 1 - e^{-2n(\varepsilon/2)^2} = 1 - e^{-n\varepsilon^2/2}.$$

Si le monde est "random", les X_i suivent une loi binomiale de paramètre $1/2$. Ainsi :

$$\Pr(Y \geq 1/2 + \varepsilon/2) \leq e^{-2n(\varepsilon/2)^2} = e^{-n\varepsilon^2/2}.$$

Finalement,

$$\begin{aligned}\mathbf{Adv}^{rr}(A) &= \Pr(Y \geq 1/2+\varepsilon/2 \text{ si "real"}) - \Pr(Y \geq 1/2+\varepsilon/2 \text{ si "random"}) \\ &\geq (1 - e^{-n\varepsilon^2/2}) - (e^{-n\varepsilon^2/2}) \\ &\geq 1 - 2e^{-n\varepsilon^2/2}\end{aligned}$$

3. Pour que $\mathbf{Adv}^{rr}(A) \geq 1/2$, il suffit que $2e^{-n\varepsilon^2/2} \leq 1/2$, ie $e^{-n\varepsilon^2/2} \leq \ln(1/4)$, soit finalement

$$n \geq \frac{4\ln 2}{\varepsilon^2} \approx \frac{2.7}{\varepsilon^2}.$$

En pratique on retient l'ordre de grandeur $O(1/\varepsilon^2)$.

ChifSymFlot-07

Exercice 10. Attaque d'un GPA par compromis temps-mémoire

On considère un GPA de mémoire de taille $192 = 128 + 64$. Il est utilisé dans un chiffrement par flot avec une clé K *(resp. un marquant IV) de taille 128 (resp. 64). On note, pour tout $t \geq 0$, X_t l'état interne et $z_t = \mathcal{F}(X_t) = \mathcal{F}(\mathcal{A}^{\circ t}(X_0))$ le bit de pseudo aléa.

On note enfin G la fonction (parfois dite *augmentée*), qui à une valeur quelconque d'état initial associe

les 192 premiers bits de pseudo aléa.

$$\begin{array}{rccl} G: & \{0,1\}^{192} & \to & \{0,1\}^{192} \\ & Y & \mapsto & (\mathcal{F}(Y), \mathcal{F}(\mathcal{A}(Y)), \ldots, \mathcal{F}(\mathcal{A}^{\circ 191}(Y))) \end{array}$$

//de sorte que, par exemple, $G(X_0) = (z_0, \ldots, z_{191})$.

L'attaque décrite est un recouvrement de clé à séquence pseudo aléatoire connue. En fait, on va voir que l'attaque permet de retrouver X_0 tout entier. En contexte cryptographique, c'est un peu du luxe puisqu'une partie de X_0 (celle qui est égale à IV) est connue dès le début par l'attaquant. En d'autres termes, l'attaque retrouve X_0 sans tirer parti de la connaissance *a priori* de l'IV.

---

1 : A collecte $z_0, z_1, \ldots, z_{2^{96}+190}$

$$\begin{array}{rcl} G(X_0) & = & (z_0, z_1, \ldots, z_{191} \\ \textit{// De sorte que} \quad G(X_1) & = & (z_1, z_2, \ldots, z_{192} \\ \ldots G(X_{2^{96}-1}) & = & (z_{2^{96}-1}, z_{2^{96}}, \ldots, z_{2^{96}+190} \end{array}$$

2 : A génère aléatoirement hors ligne 2^{96} éléments (indépendants) de $\{0,1\}^{192}$: $Y_0, \ldots Y_{2^{96}-1}$, et calcule $G(Y_i)$ pour $i = 0, 1, \ldots, 2^{96} - 1$.

3 : A cherche une rencontre entre les deux listes $\mathcal{L}_X = [G(X_0), G(X_1), \ldots G(X_{2^{96}-1})]$ et $\mathcal{L}_Y = [G(Y_0), G(Y_1), \ldots G(Y_{2^{96}-1})]$.

4 : Si pas de rencontre retourner en 2.

5 : Si rencontre $G(X_i) = G(Y_j)$
// *cela implique probablement* $X_i = Y_j$
retourner $\mathcal{A}^{-i}(Y_j)$ // censé être égal à X_0

---

1. Vérifier que, pour tout $t \geq 0$, $G(X_t) = (z_t, \ldots, z_{t+191})$ (c'est une conséquence directe des définitions et notations)
2. Soit i fixé. Quelle est la probabilité $Y_i \in \mathcal{L}_X$?
3. En déduire la probabilité que $\mathcal{L}_Y \cap \mathcal{L}_X \neq \emptyset$, c'est-à-dire que l'un au moins des Y_i soit dans $mathcal L_X$.
4. Quelle est la complexité de l'attaque, À supposer qu'il y ait eu rencontre entre $\mathcal{L}_Y$ et $\mathcal{L}_X$ et en négligeant le temps de trouver cette rencontre, déterminer la complexité de l'attaque. Vérifier qu'elle est plus efficace que la brute force.
5. Que proposer (Indication : sur la taille de l'état interne du GPA) pour que cette attaque soit alignée à la complexité de la brute force.

1. Cette propriété tient au fait que les fonctions d'avance et de calcul du bit de sortie sont indépendantes du temps t. Ainsi, il y a un phénomène d'invariance par translation temporelle.
2. Le tirage de chaque Y_i étant uniforme dans $\{0,1\}^{192}$, la probabilité cherchée est $2^{96}/2^{192} = 2^{-96}$.
3. En passant par les événements contraires, et ceux-ci étant indépendants, la probabilité qu'au moins l'un des Y_i rencontre $\mathcal{L}_X$ est $1 - (1 - 2^{-96})^{2^{96}} \approx 1 - e^{-1} \approx 0,63$
4. La complexité réside dans le nombre d'étapes pour construire les deux listes, donc de l'ordre de $2 * 2^{96} = 2^{97}$. Elle est significativement inférieure à 2^{128} (brute force de la clé).
5. Il faut que la taille de l'état interne soit au moins égale à deux fois la taille de la clé.

ChifSymFlot-08

Exercice 11. Attaque de Siegenthaler diviser pour régner

On considère un GPA dit à combinaison. Il est composé de 4 automates A_0, A_1, A_2, A_3 indépendants fonctionnant en parallèle et d'une fonction booléenne $f(x_0, \ldots x_3)$ à 4 entrées. À chaque instant $t \geq 0$, les

automates $A_0, \ldots, A_3$ génèrent chacun un bit $a_{0,t}, \ldots, a_{3,t}$, et le bit de pseudo aléa est calculé comme $z_t := f(a_{0,t}, \ldots, a_{3,t})$, f étant définie par sa table de ses $2^4 = 16$ valeurs :

		$x = [x_0, \ldots x_3]$	$f(x)$
Dec	Hex	Bin	
0	`0x0`	$[0,0,0,0]$	0
1	`0x1`	$[1,0,0,0]$	0
2	`0x2`	$[0,1,0,0]$	0
3	`0x3`	$[1,1,0,0]$	1
4	`0x4`	$[0,0,1,0]$	0
5	`0x5`	$[1,0,1,0]$	1
6	`0x6`	$[0,1,1,0]$	0
7	`0x7`	$[1,1,1,0]$	1
8	`0x8`	$[0,0,0,1]$	0
9	`0x9`	$[1,0,0,1]$	1
10	`0xa`	$[0,1,0,1]$	1
11	`0xb`	$[1,1,0,1]$	1
12	`0xc`	$[0,0,1,1]$	0
13	`0xd`	$[1,0,1,1]$	1
14	`0xe`	$[0,1,1,1]$	0
15	`0xf`	$[1,1,1,1]$	1

On suppose que chaque automate A_i a une mémoire de taille 32 bits. La valeur initiale $A_{0,t=0}, A_{1,t=0}, A_{2,t=0}, A_{3,t=0}$ est la cible de l'attaque. On suppose que l'attaquant connait 640 (valeur arbitraire, juste pour fixer les idées) bits de pseudo aléa $z_0, \ldots, z_{639}$.

1. Quelle est la complexité de la brute force ?
2. Montrer que f présente une corrélation d'ordre 1 par rapport à sa première variable x_0, càd montrer que l'égalité $f(x_0, \ldots x_3) = x_0$ a lieu pour une proportion des vecteurs d'entrées $x_0, \ldots x_3$ différente de $1/2$. Si on note $1/2 + \varepsilon$ cette proportion, que vaut ε ?

La propriété ci-dessus permet à l'attaquant de brute forcer $A_{0,t=0}$ malgré son ignorance des valeurs $A_{1,t=0}, A_{2,t=0}, A_{3,t=0}$. L'attaquant :

---

1 : crée une copie B_0 de A_0 ;

2 : pour chaque valeur de $B_{0,t=0} \in \mathbb{F}_2^8$

3 : calcule $b_0, \ldots, b_{639}$ la suite générée par B_0,

4 : et calcule $N = \#\{t \ : \ t = 0, 1, \ldots 639 \ : \ b_t = z_t\}$, nombre d'instants de coïncidence entre $b_0, \ldots, b_{639}$ et $z_0, \ldots, z_{639}$

5 : retourne la valeur de $B_{0,t=0}$ pour lequel N est maximal

---

3. Montrer que lorsque le choix de $B_{0,t=0}$ est égal à $A_{0,t=0}$, alors N est de l'ordre de $640 \times 14/16$.
4. Lorsque le choix de $B_{0,t=0}$ est différent de $A_{0,t=0}$, on suppose (heuristique) que $[b_0, \ldots, b_{639}]$ est indépendante de $z_0, \ldots, z_{639}]$, la situation est aléatoire uniforme. Sous cette hypothèse, quelle est l'ordre de grandeur de N ?
5. Justifier *a posteriori* la pertinence de l'étape 5 de l'algorithme.
6. Quelle est la complexité de l'attaque retrouvant $A_{0,t=0}$ seul ?
7. Sous une hypothèse naturelle que l'on formulera, donner la complexité globale de l'attaque qui retrouve $A_{0,t=0}, A_{1,t=0}, A_{2,t=0}, A_{3,t=0}$.

1. La recherche exhaustive coûte $2^{128} = (2^{32})^4$
2. Il suffit d'examiner la table de valeurs de la fonction f. On voit, par simple comptage, que l'égalité

$f(x_0, \dots x_3) = x_0$ a lieu dans 14 cas sur 16. Ainsi $\varepsilon = 14/16 - 1/2 = 3/8$

3. Si $B_{0,t=0}$ est égal à $A_{0,t=0}$, alors, pour tout $t = 0, 1, \dots, 639$, $b_t = a_{0,t}$. Or, la Q2 implique "qu'en moyenne", le nombre de coïncidences entre $[z_0, \dots, z_{639}]$ et $[a_{0,t=0}, \dots, a_{0,t=639}]$ est de $640 \times 14/16 (= 560)$.
4. Comme b_t et z_t sont indépendantes, le nombre moyen de coincidences est $640 \times 1/2 = 320$
5. Le bon candidat pour $A_{0,t=0}$ génère "en moyenne" significativement plus de coïncidences que les mauvais candidats. La stratégie de l'attaquant est donc pertinente et le conduira à la bonne valeur de $A_{0,t=0}$ avec forte probabilité.
6. 2^{32}.
7. Si la fonction f présente également une corrélation significative sur chacune de ses trois autres entrées, alors on peut déterminer $A_{0,t=0}, A_{1,t=0}, A_{2,t=0}, A_{3,t=0}$ par quatre boucles **séparées** (et non imbriquées, c'est le cas point crucial), chacune de complexité 2^{32}. De sorte que la complexité gloable est de 4×2^{32} qui est évidemment $\ll 2^{128}$.

3 Primitives blocs

3.1 Tutoriel jouet

ChifSymBloc-11

Exercice 12. Une primitive bloc en taille jouet

On définit une fonction bloc 32 bits, clé 64 bits, au niveau octet

---

Entrées : $X = (x_0, \dots, x_3) \in \{0,1\}^{32}$, $K = (k_0, \dots, k_7) \in \{0,1\}^{64}$

1. $(y_0, \dots, y_3) \leftarrow (x_0, \dots, x_3) \oplus (k_0, \dots, k_3)$
2. $(z_0, \dots, z_3) \leftarrow (\text{SBox}(y_0), \dots, \text{SBox}(y_3))$
3. $\begin{bmatrix} v_0 \\ \vdots \\ v_3 \end{bmatrix} \leftarrow [\text{MixColumns}] \begin{bmatrix} z_0 \\ \vdots \\ z_3 \end{bmatrix}$
4. $(w_0, \dots, w_3) \leftarrow (v_0, \dots, v_3) \oplus (k_4, \dots, k_7)$
5. **Retourner** $(w_0, \dots, w_3)$

---

où les fonctions SBox et MixColumns sont celles de l'AES. Calculer le chiffré du message clair `32 43 f6 a8` avec la clé `2b 7e 15 16 28 ae d2 a6`.

```
This a Toy Block Cipher with input
  Key (64-bit):          2b 7e 15 16 28 ae d2 a6
  Input block (32-bit): 32 43 f6 a8
After first add key:    19 3d e3 be
After Sbox:             d4 27 11 ae
After MixColumns:       65 07 38 16
After second add key:   4d a9 ea b0   (output block)
```

3.2 Exemples d'attaque en distingueur

ChifSymBloc-02

Exercice 13. Famille d'applications linéaires : distinguer et recouvrement de clé

On considère la famille des applications linéaires $\mathbb{F}_2^n \to \mathbb{F}_2^m$, c'est-à-dire des matrices binaires m lignes n colonnes (notation française). On considère un oracle $\mathcal{O}$ real-or-random, le monde réel (resp. random) signifiant que l'oracle implémente une application linéaire inconnue (resp. une application aléatoire inconnue). On considère un attaquant cherchant à distinguer, accédant à $\mathcal{O}$, et agissant comme suit

---

1. A effectue la requête $0^n = \underbrace{(0,\ldots,0)}_{n}$ à l'oracle $\mathcal{O}$, qui retourne Y.
2. Si $Y = 0^m = \underbrace{(0,\ldots,0)}_{m}$, A parie "real"
3. sinon, A parie "random"

---

1. Montrer que l'avantage de A est égal à $1 - 2^{-m}$.

2. On suppose à présent que l'attaquant sait qu'il est dans le monde "real", donc que la fonction est une fonction linéaire de matrice inconnue. Construire une attaque qui retrouve la matrice avec n requêtes à $\mathcal{O}$.

Rappelons tout d'abord qu'une fonction aléatoire est une fonction dont la table de valeurs a été tirée aléatoirement au moment de sa construction. Il s'agit d'aléatoire "statique", par opposition à une fonction "randomisée", où le calcul dynamique de l'image fait intervenir une quantité auxiliaire aléatoire non prédictible et non reproductible d'un appel à l'autre.

1. La probabilité que l'attaquant retourne "real" est celle que l'image de 0^n soit 0^m. Si le monde est "real", l'image de 0^n est 0^m de manière certaine. Donc la probabilité que l'attaquant retourne "real" est 1. Si le monde est "random", l'image de 0^n est une valeur au hasard dans $\mathbb{F}_2^m$, donc égale à 0^m avec probabilité 2^{-m}. La formule de l'avantage suit.

2. B demande l'image des n vecteurs de la base canonique. La réponse de l'oracle à la requête i est exactement la i-ième colonne de la matrice.

ChifSymBloc-03

Exercice 14. Une concaténation distinguable

On suppose que E_K une bonne prf sur $\{0,1\}^n$. On définit

$$G_K : x \mapsto E_K(x)||E_K(\bar{x})$$

Soit A un attaquant et $\mathcal{O}$ un oracle real or random pour G_K, et A l'attaquant suivant :

---

1. $(y_1||y_2) \leftarrow \mathcal{O}(1^n)$
2. $(z_1||z_2) \leftarrow \mathcal{O}(0^n)$
3. si $y_1 = z_2$ parier "real" sinon "random"

Montrer que l'avantage de A est égal à $1 - 2^{-n}$

La probabilité que A retourne "real" est celle que $y_1 = z_2$.

— Si le monde est "real", $y_1 = z_2 = E_K(1^n)$ de manière certaine.

— Si le monde est "random", $y_1 = z_2$ est une égalité de deux valeurs sur n bits prises au hasard, donc se produit avec probabilité 2^{-n}.

Le résultat demandé suit.

ChifSymBloc-04

Exercice 15. Distinguer un Feistel à 2 tours d'une prf

On considère $F2_K$ un Feistel à deux tours. On note $(L_i, R_i) \in \{0,1\}^n$ la variable de tour ((L_0, R_0) est le clair), f la fonction de tour, et SK_i la sous-clé du tour i.

1. Vérifier que

$$\begin{aligned} L_2 &= L_0 \oplus f_{SK_0}(R_0) \\ R_2 &= R_0 \oplus f_{SK_1}(L_0 \oplus f_{SK_0}(R_0)) \end{aligned}$$

2. On pose, pour $L_0 \neq L'_0$:

$$\begin{aligned} (L_2, R_2) &= F2_K(L_0, R_0) \\ (L'_2, R'_2) &= F2_K(L'_0, R_0) \end{aligned}$$

Montrer que $L_2 \oplus L'_2 = L_0 \oplus L'_0$.

3. Quelle serait la probabilité que cette égalité soit vraie si $F2_K$ était une prf ?

4. En déduire un attaquant dont l'avantage est $1 - 2^{-n/2}$.

1. On a $L_1 = R_0, \quad R_1 = L_0 \oplus f_{SK_0}(R_0)$. Puis

$$\begin{aligned} L_2 &= R_1 &&= L_0 \oplus f_{SK_0}(R_0) \\ R_2 &= L_1 \oplus f_{SK_1}(R_1) &&= R_0 \oplus f_{SK_1}(L_0 \oplus f_{SK_0}(R_0)) \end{aligned}$$

2. $L_2 \oplus L'_2 = (L_0 \oplus f_{SK_0}(R_0)) \oplus (L'_0 \oplus f_{SK_0}(R_0)) = L_0 \oplus L'_0$, cqfd.

3. Si $F2_K$ était une fonction aléatoire, la probabilité de l'égalité de la question 2 serait de $2^{-n/2}$, car c'est l'égalité de deux valeurs aléatoires indépendantes de taille $n/2$.

4. L'attaquant A effectue deux requêtes :

---

1. $(L_2, R_2) \leftarrow \mathcal{O}((0^n))$
2. $(L'_2, R'_2) \leftarrow \mathcal{O}((1^{n/2}, 0^{n/2}))$
3. Si $L_2 \oplus L'_2 = (1^{n/2})$ retourner "real"
4. sinon retourner "random"

— Si le monde est "real", on a de manière certaine l'égalité $L_2 \oplus L'_2 = (1^{n/2})$. L'attaquant retourne "real" avec probabilité 1.

— Si le monde est " random", l'égalité $L_2 \oplus L'_2 = (1^{n/2})$ est aléatoire. L'attaquant retourne "real" avec probabilité 2^{-n}.

Le résultat suit.

3.3 DES

ChifSymBloc-05

Exercice 16. Un calcul de DES

Calculer le premier tour du DES pour le message :

$$T = \texttt{0001 0001 0010 0010 0011 0011 0100 0100 0101 0101 0110 0110 0111 0111 1000 1000}$$

avec la clé de tour :

$$K_0 = \texttt{0011 1000 1010 1100 1110 1111 0100 0110 0101 0110 0100 1010}$$

À partir des valeurs du message et de la clé de tour, on a :
$L_0 = \texttt{0111 1000 0101 0101 0111 1000 0101 0101}$.
$R_0 = \texttt{1000 0000 0110 0110 1000 0000 0110 0110}$.
$E(R_1) = \texttt{0100 0000 0000 0011 0000 1101 0100 0000 0000 0011 0000 1101}$.
$E(R_1) \oplus K_0 = \texttt{011110 001010 111111 100010 000001 100101 010101 000111}$.
$S_1 \to \texttt{0111}$ $S_2 \to \texttt{1011}$ $S_3 \to \texttt{1100}$ $S_4 \to \texttt{0110}$
$S_5 \to \texttt{1110}$ $S_6 \to \texttt{0010}$ $S_7 \to \texttt{0101}$ $S_8 \to \texttt{1000}$.
$P(S_1||\cdots||S_8) = \texttt{0100 1011 0111 1101 1101 0011 1000 0010}$.
$L_1 = \texttt{0011 0011 0010 1000 1010 1011 1101 0111}$.
$R_1 = \texttt{0101 0011 1001 1010 0010 1001}$.

ChifSymBloc-06

Exercice 17. Deux variantes du DES

Pour $K_1 \in \{0,1\}^{56}$, $K_2 \in \{0,1\}^{64}$, on définit deux chiffrements par blocs 64 bits à partir du DES :

$$\text{DESY}_{K_1,K_2} : M \mapsto \text{DES}_{K_1}(K_2 \oplus M)$$
$$\text{DESW}_{K_1,K_2} : M \mapsto K_2 \oplus \text{DES}_{K_1}(M)$$

On s'intéresse à une attaque qui retrouve la clé K_1, K_2 à clair choisi.

1. Soit (M, C) un couple (clair, chiffré). Montrer que

$$\forall K_1,\ \exists K_2\ C = \text{DESY}_{K_1,K_2}(M)$$

En déduire qu'il est (presque) impossible de retrouver la clé avec un seul couple (clair, chiffré).

2. Montrer que $C = \text{DESY}_{K_1,K_2}(M) \Leftrightarrow \text{DES}^{-1}_{K_1}(C) \oplus M = K_2$. En déduire une attaque retrouvant la clé de DESY avec deux couples (clair, chiffré).

3. Reprendre la question 1 pour DESW.

4. Montrer que $C = \text{DESW}_{K_1,K_2}(M) \Leftrightarrow \text{DES}_{K_1}(M) \oplus C = K_2$. En déduire une attaque retrouvant la clé de DESW avec deux couples (clair, chiffré).

1. On a

$$\text{DESY}_{K_1,K_2}(M) = C \quad \Leftrightarrow \quad \text{DES}_{K_1}(K_2 \oplus M) = C \quad \Leftrightarrow \quad \text{DES}^{-1}_{K_1}(C) = K_2 \oplus M$$

Ainsi M, C étant fixés, pour tout K_1, $K_2 = \text{DES}^{-1}_{K_1}(C) \oplus M$ répond à la question. Cela signifie qu'il existe au moins 2^{56} clés de DESY compatibles d'un couple clair, chiffré). Il est donc impossible de trouver la bonne à moins de la deviner par hasard, soit avec probabilité 2^{-56}.

2. L'égalité a été montrée à la question 1. Prenons deux couples (M_1, C_1) et (M_2, C_2). On a

$$\mathrm{DES}^{-1}_{K_1}(C_1) \oplus M_1 = \mathrm{DES}^{-1}_{K_1}(C_2) \oplus M_2 \ (= K_2)$$

Si K_1 n'est pas la bonne clé, l'égalité ci-dessus a lieu avec probabilité 2^{-64}. Une telle égalité est donc "révélatrice" de la vraie clé. D'où l'attaque :

---

1. Pour $K_1 \in \{0,1\}^{56}$
2. si $\mathrm{DES}^{-1}_{K_1}(C_1) \oplus M_1 = \mathrm{DES}^{-1}_{K_1}(C_2) \oplus M_2$
3. poser $K_2 =$ cette valeur commune
4. retourner (K_1, K_2)

---

3. Le principe est le même que pour DESY.

$$C = \mathrm{DESW}_{K_1,K_2}(M) \Leftrightarrow \mathrm{DES}_{K_1}(M) \oplus C = K_2$$

donc pour tout K_1, $K_2 = \mathrm{DES}_{K_1}(M) \oplus C$ est compatible du couple (M, C).

4. Attaque à deux couples (clair, chiffré) :

---

1. Pour $K_1 \in \{0,1\}^{56}$
2. si $\mathrm{DES}_{K_1}(M_1) \oplus C_1 = \mathrm{DES}_{K_1}(M_2) \oplus C_2$
3. poser $K_2 =$ cette valeur commune
4. retourner (K_1, K_2)

ChifSymBloc-07

Exercice 18. Double DES, attaque par le milieu

On note $E_{(.)}$ le simple DES. Pour accroître la sécurité du DES, en particulier la (relativement faible) taille de clé (56 bits), une idée naturelle consiste à enchaîner plusieurs chiffrements du message clair m avec des clés différentes. D'où la définition du double DES :

$$\begin{array}{rcccl} c & = & \mathrm{2DES}_{K_1,K_2}(m) & = & E_{K_2} \circ E_{K_1}(m) \\ m & = & \mathrm{2DES}^{-1}_{K_1,K_2}(c) & = & E^{-1}_{K_1} \circ E^{-1}_{K_2}(c) \end{array}$$

1. Quelle est la complexité de la recherche exhaustive des clés du 2DES ?

On considère l'algorithme dit "attaque par le milieu". On suppose connu un couple (clair, chiffre) (m, c), et on cherche à retrouver la clé (K_1, K_2).

---

1 : Construire $\mathcal{L}_1 = \{(E_i(m), i) :\ i \in \{0,1\}^{56}\}$
2 : Construire $\mathcal{L}_2 = \{(E_j^{-1}(c), j) :\ j \in \{0,1\}^{56}\}$
3 : initialiser l'ensemble `ListeCandidats` $= \emptyset$
4 : pour $y \in \mathcal{L}_2$ d'abscisse j
5 : si $y \in \mathcal{L}_1$ à l'abscisse i
6 : ajouter (i, j) à `ListeCandidats`
7 : retourner `ListeCandidats`

L'algorithme est supposé effectuer la boucle "pour" en entier et retourne donc potentiellement plusieurs couples (i, j).

2. Montrer que, pour toute couple (i,j) retourné, on a $c = 2\text{DES}_{i,j}(m)$.
   Ainsi, dans le cas où l'algorithme ne retourne qu'une seule valeur, celle-ci est la clé cherchée (K_1, K_2).
3. Quelle est la complexité de l'algorithme (on négligera le temps du test d'appartenance d'un élément à une liste) ?

1. La recherche exhaustive porte sur les deux clés K_1 et K_2 donc sa complexité est 2^{112} calculs de DES
2. Puisque $y \in \mathcal{L}_1$ à l'abscisse i, on a $y = \text{DES}_i(m)$. De même, $y \in \mathcal{L}_2$ à l'abscisse j, donc on a $y = \text{DES}_j^{-1}(c)$. La conclusion suit aisément.
3. La construction des deux listes a pour complexité 2^{56} calculs de DES (pour chacune des listes). La complexité du test d'appartenance étant négligée, la complexité de l'attaque est donc 2^{57} calculs de DES et 2^{57} en mémoire pour stocker les listes. (NB : ce stockage n'est pas nécessaire dans la recherche exhaustive).

Une variante de l'algorithme consiste à ne construire que la liste $\mathcal{L}_1$ et à ne pas stocker les éléments de $\mathcal{L}_2$ mais au contraire à les tester en direct et à les jeter tout de suite s'ils ne sont pas dans $\mathcal{L}_1$. La complexité en temps ne change pas mais le stockage est en 2^{56}.

ChifSymBloc-12

Exercice 19. **Triple DES, attaque par le milieu**

On note encore $E_{(.)}$ le simple DES. On étudie maintenant le triple DES : 3DES3 avec trois clés K_1, K_2, K_3. Soit la formule de chiffrement

$$c = \text{3DES3}_{K_1,K_2,K_3} = E_{K_3} \circ E_{K_2}^{-1} \circ E_{K_1}(m)$$

NB : la fonction centrale est l'inverse du DES, par souci de compatibilité arrière avec le simple DES. En effet, on vérifie immédiatement que

$$\text{DES}_{K_1} = \text{3DES3}_{K_1,K_1,K_1}$$

1. Quelle est la complexité de la recherche exhaustive ?
2. Montrer que l'on peut adapter l'algorithme d'attaque du double DES et retrouver la clé (K_1, K_2, K_3) avec une complexité en temps 2^{112} (Indication : : on peut couper l'algorithme en deux parties "inégales")

On considère maintenant trois versions de triple DES à deux clés :

$$\begin{array}{lrcl} (1) & c & = & E_{K_2} \circ E_{K_1} \circ E_{K_1}(m) \\ (2) & c & = & E_{K_1} \circ E_{K_1} \circ E_{K_2}(m) \\ (3) & c & = & E_{K_1} \circ E_{K_2}^{-1} \circ E_{K_1}(m) \end{array}$$

3. Quelle est la complexité de la recherche exhaustive ?
4. Montrer que l'on peut encore adapter l'attaque du double DES aux versions (1) et (2)
5. Quelle est alors la complexité de l'attaque ?

On s'intéresse désormais à la version (3), et on applique l'attaque à clair choisi suivante :

---

1 : On construit $\mathcal{L}_1 = \{(P_i = E_i^{-1}(0^{64}), i) : \ i \in \{0,1\}^{56}\}$.

2 : On requiert à l'oracle $C_i = \underbrace{\text{Oracle-Encryption}}_{E_{K_1} \circ E_{K_2}^{-1} \circ E_{K_1}}(P_i)$, pour $i \in \{0,1\}^{56}$.

3 : On construit $\mathcal{L}_2 = \{(Q_j = E_j^{-1}(C_j), j) : \ j \in \{0,1\}^{56}\}$.

4 : Trier $\mathcal{L}_1$ et $\mathcal{L}_2$ selon la première composante.

6. Montrer que s'il existe i, j tels que $P_j = Q_i$ alors (i, j) est un candidat pour la clé (K_1, K_2).
7. Justifier que ce type de collision est rapide à trouver.
8. Combien y a-t-il en moyenne de telles collisions, c'est-à-dire de couples candidats ?
9. Comment trouver le bon candidat avec peu de complexité supplémentaire ?
10. Quelle est la complexité globale de cette stratégie ?

Remarque. L'attaque à clair choisi du triple DES option (3) nécessite grand nombre d'appels à l'oracle. Le triple DES option (3) reste en ce sens plus robuste que le triple DES options (1), (2) ou le double DES. En effet, les attaques à clair connu contre ces derniers nécessitent peu de couples (clair, chiffré) et pas d'oracle.

1. La recherche exhaustive a pour complexité 2^{168} calculs de 3DES.
2. On adapte l'algorithme d'attaque du double DES comme suit. On indexe la liste $\mathcal{L}_1$ par $i \in \{0,1\}^{56}$:
$$\mathcal{L}_1 = \{(E_i(m), i),\ i \in \{0,1\}^{56}\}$$
et la liste $\mathcal{L}_2$ par $j = (j_1, j_2) \in \{0,1\}^{112}$, c'est-à-dire
$$\mathcal{L}_2 = \{(E_{j_1}^{-1} \circ E_{j_2}^{-1}(c), j),\ j \in \{0,1\}^{112}\}$$
L'attaque fonctionne comme précédemment, bien que les deux listes soient maintenant de tailles différentes. En utilisant la variante où $\mathcal{L}_2$ n'est pas stockée, on a donc une complexité en temps 2^{112} (parcours des éléments de $\mathcal{L}_2$) et stockage 2^{56} (la liste $\mathcal{L}_1$).
3. La recherche exhaustive sur les versions (1), (2) et (3) du triple DES a pour complexité 2^{112}.
4. L'attaque par le milieu du double DES fonctionne en fait pour tout chiffrement de type $G_{K_2} \circ F_{K_1}$, la complexité étant celle de la construction des deux listes. On voit que les options (1) et (2) entrent dans ce cas, mais pas l'option (3).
5. La complexité est $\approx 2^{|K_1|} + 2^{|K_2|}$. Ici $\approx 2^{57}$.
6. S'il existe (i, j) satisfaisant $P_j = Q_i$, on a
$$C_i = E_i(Q_i) = E_i(P_j) = E_i \circ E_j^{-1}(0) = E_i \circ E_j^{-1} \circ E_i(P_i)$$
Mais par construction, $C_i = E_{K_1} \circ E_{K_2}^{-1} \circ E_{K_1}(P_i)$. Le couple (i, j) est donc candidat pour la clé (K_1, K_2).
7. Pour trouver de telles collisions, on fusionne les listes $\mathcal{L}_1$ et $\mathcal{L}_2$ et on trie sur la première composante. Pour conserver la trace du fait qu'un élément vient de la liste 1 ou 2, il faut ajouter un label en troisième composante. On fusionne donc les éléments $(P_i, i, 1)$ et $(Q_i, i, 2)$. Les collisions cherchées sont des éléments consécutifs dans la liste fusionnée pour lesquels les premières composantes sont identiques et les labels sont différents.
8. Il y a 2^{112} couples (i, j), et les P_i et Q_i sont de taille 64 bits. Il y a donc en moyenne $2^{112}/2^{64} = 2^{48}$ collisions $P_j = Q_i$.
9. Pour trouver le bon candidat, il faut identifier (et éliminer) les $2^{48} - 1$ fausses alarmes. Mais un couple candidat peut être testé en $O(1)$, en vérifiant la cohérence avec les réponses de l'oracle sur un petit nombre de couples (clair, chiffré). Globalement, l'identification du bon candidat coûte au pire $O(2^{48})$.
10. La complexité de la stratégie est :
    — 2^{57} calculs hors ligne pour la construction des deux listes ;
    — 2^{57} blocs de mémoire pour le stockage des deux listes ;
    — 2^{56} appels à l'oracle ;
    — $57 \cdot 2^{57}$ calculs pour le tri de la liste résultant des deux listes fusionnées ;
    — $O(2^{48})$ calculs pour l'identification du bon candidat.

    On voit que l'ordre de grandeur est encore celui de la recherche exhaustive sur le simple DES.

3.4 AES

ChifSymBloc-08

Exercice 20. Chiffrement AES tutorial

1. À quelle catégorie de méthodes blocs l'AES appartient-il ?
2. Montrer que la S-box de l'AES est une application non linéaire.
3. Voici l'état à l'entrée d'un tour et la clé de tour correspondante :

$$\text{state} = \begin{pmatrix} 42 & 00 & 3a & 8e \\ 28 & 6b & 0a & 6c \\ 03 & aa & 88 & bc \\ 4b & 27 & 11 & 60 \end{pmatrix}, \qquad \text{round key} = \begin{pmatrix} ac & 19 & 28 & 57 \\ 77 & fa & d1 & 5c \\ 66 & dc & 29 & 00 \\ f3 & 21 & 41 & 6e \end{pmatrix},$$

   Calculer le state en sortie du SubByte.
4. Appliquer le ShiftRow.
5. Calculez la sortie du MixColumn.
6. Appliquer AddRoundKey pour obtenir la sortie du tour

1. Il s'agit d'un schéma de Substitution-Permutation (SPN, *Substitution-Permutation Network*)
2. Il suffit de constater que $SB(00) \neq 00$. Ici, on en rajoute une couche en exhibant deux octets x et y pour lesquels $SB(x \oplus y) \neq SB(x) \oplus SB(y)$.

$$\begin{aligned} SB(00) &= 63 \\ SB(01) &= 7c \\ SB(00) \oplus SB(01) &= 1f \\ SB(00 \oplus 01) &= SB(01) = 7c \\ SB(00 \oplus 01) &\neq SB(00) + SB(01) \end{aligned}$$

3. Sortie du SubByte :

$$\begin{pmatrix} 2c & 63 & 80 & 19 \\ 34 & 7f & 67 & 50 \\ 7b & ac & c4 & 65 \\ b3 & cc & 82 & d0 \end{pmatrix}$$

4. Sortie du ShiftRows :

$$\begin{pmatrix} 2c & 63 & 80 & 19 \\ 7f & 67 & 50 & 34 \\ c4 & 65 & 7b & ac \\ d0 & b3 & cc & 82 \end{pmatrix}$$

5. MixColumns :

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} 2c \\ 7f \\ c4 \\ d0 \end{pmatrix} = \begin{pmatrix} 02 \bullet 2c \oplus 03 \bullet 7f \oplus 01 \bullet c4 \oplus 01 \bullet d0 \\ 01 \bullet 2c \oplus 02 \bullet 7f \oplus 03 \bullet c4 \oplus 01 \bullet d0 \\ 01 \bullet 2c \oplus 01 \bullet 7f \oplus 02 \bullet c4 \oplus 03 \bullet d0 \\ 03 \bullet 2c \oplus 01 \bullet 7f \oplus 01 \bullet c4 \oplus 02 \bullet d0 \end{pmatrix} = \begin{pmatrix} cd \\ 55 \\ ab \\ 74 \end{pmatrix}$$

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} 63 \\ 67 \\ 65 \\ b3 \end{pmatrix} = \begin{pmatrix} 02 \bullet 63 \oplus 03 \bullet 67 \oplus 01 \bullet 65 \oplus 01 \bullet b3 \\ 01 \bullet 63 \oplus 02 \bullet 67 \oplus 03 \bullet 65 \oplus 01 \bullet b3 \\ 01 \bullet 63 \oplus 01 \bullet 67 \oplus 02 \bullet 65 \oplus 03 \bullet b3 \\ 03 \bullet 63 \oplus 01 \bullet 67 \oplus 01 \bullet 65 \oplus 02 \bullet b3 \end{pmatrix} = \begin{pmatrix} b9 \\ b1 \\ 00 \\ da \end{pmatrix}$$

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} 80 \\ 50 \\ 7b \\ cc \end{pmatrix} = \begin{pmatrix} 02 \bullet 80 \oplus 03 \bullet 50 \oplus 01 \bullet 7b \oplus 01 \bullet cc \\ 01 \bullet 80 \oplus 02 \bullet 50 \oplus 03 \bullet 7b \oplus 01 \bullet cc \\ 01 \bullet 80 \oplus 01 \bullet 50 \oplus 02 \bullet 7b \oplus 03 \bullet cc \\ 03 \bullet 80 \oplus 01 \bullet 50 \oplus 01 \bullet 7b \oplus 02 \bullet cc \end{pmatrix} = \begin{pmatrix} 5c \\ 61 \\ 69 \\ 33 \end{pmatrix}$$

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} 19 \\ 34 \\ ac \\ 82 \end{pmatrix} = \begin{pmatrix} 02 \bullet 19 \oplus 03 \bullet 34 \oplus 01 \bullet ac \oplus 01 \bullet 82 \\ 01 \bullet 19 \oplus 02 \bullet 34 \oplus 03 \bullet ac \oplus 01 \bullet 82 \\ 01 \bullet 19 \oplus 01 \bullet 34 \oplus 02 \bullet ac \oplus 03 \bullet 82 \\ 03 \bullet 19 \oplus 01 \bullet 34 \oplus 01 \bullet ac \oplus 02 \bullet 82 \end{pmatrix} = \begin{pmatrix} 40 \\ 1c \\ f3 \\ ac \end{pmatrix}$$

d'où la sortie de MixColumns

$$\begin{pmatrix} cd & b9 & 5c & 40 \\ 55 & b1 & 61 & 1c \\ ab & 00 & 69 & f3 \\ 74 & da & 33 & ac \end{pmatrix}$$

6. Après AddRoundKey

$$\begin{pmatrix} cd & b9 & 5c & 40 \\ 55 & b1 & 61 & 1c \\ ab & 00 & 69 & f3 \\ 74 & da & 33 & ac \end{pmatrix} \oplus \begin{pmatrix} ac & 19 & 28 & 57 \\ 77 & fa & d1 & 5c \\ 66 & dc & 29 & 00 \\ f3 & 21 & 41 & 6e \end{pmatrix} = \begin{pmatrix} 61 & a0 & 74 & 17 \\ 22 & 4b & b0 & 40 \\ cd & dc & 40 & f3 \\ 87 & fb & 72 & c2 \end{pmatrix}$$

ChifSymBloc-09

Exercice 21. Un AES randomisé à distinguer d'une prf

On considère l'algorithme de chiffrement symétrique $\mathcal{E}$ suivant, avec espace d'entrée les messages de 64 bits, et clé de longueur 128 bits. Pour éviter une attaque par dictionnaire, on utilise une primitive bloc 128 (l'AES) et on randomise le chiffrement.

$\mathcal{E}_K$: pour toute entrée $M \in \{0,1\}^{64}$

---

1. choisir R aléatoire de taille 64 bits
2. retourner $C = \mathrm{AES}_K(R||M)$ $(\in \{0,1\}^{128})$

On étudie l'indistinguabilité left-right de $\mathcal{E}$. On va montrer que le schéma est distinguable avec un nombre de requêtes de l'ordre de 2^{64}. Soit l'attaquant A suivant ayant accès à un oracle $\mathcal{O}$ left-right :

---

1. pour $i = 1, \ldots, q$
2. demander $C_i \leftarrow \mathcal{O}(\underbrace{\texttt{Vect}(64)(i)}_{\text{left}}, \underbrace{0^{64}}_{\text{right}})$
3. si $\exists i < j$ tel que $C_i = C_j$ parier "right"
4. sinon parier "left"

---

où $\texttt{Vect}(64)(i)$ désigne l'écriture binaire de l'entier i sur 64 bits.

1. Montrer que la probabilité que A parie "right" dans le cas du monde "right" est égale à $\mathsf{PrCol}(2^{64}, q)$.

2. Montrer que la probabilité que A parie "right" dans le cas du monde "left" est égale à 0.

3. En déduire l'avantage de l'attaquant, puis que le schéma est distinguable left-right avec un nombre de requêtes de l'ordre de 2^{64}.

1. Si le monde est "right", la valeur renvoyée par l'oracle est $\mathrm{AES}_K(R||0^{64})$. La probabilité qu'il y ait une collision parmi les C_i est donc celle qu'il y ait une collision parmi les R qui ont été tirés : c'est $\mathsf{PrCol}(2^{64}, q)$.

2. Si le monde est "left", la valeur renvoyée par l'oracle est $\mathrm{AES}_K(R||\mathtt{Vect}(64)(i))$. Les messages traités par AES sont deux à deux différents : il est donc impossible qu'il y ait collision sur les C_i (même s'il y a collision sur les R). La probabilité que A retourne "right" est nulle.

3. L'avantage de l'attaquant est $\mathsf{PrCol}(2^{64}, q)$. Cet avantage est $\geq 1/2$ dès que $q \geq 1,17 \times 2^{32}$.

ChifSymBloc-10

Exercice 22. **Attaque par saturation de l'AES 4 tours (avec programmation)**

Dans notre contexte, une *collection* de vecteurs est un 256-uplet $(x_0, \ldots, x_{255})$, où chaque $x_i \in \mathbb{F}_2^8$, cad est un "octet". Les x_i ne sont pas forcément 2 à 2 distincts. Une collection est dite :

- *saturante* S lorsque toutes les valeurs de $\mathbb{F}_2^8$ sont prises exactement une fois ; cela équivaut à ce que les x_i soient 2 à 2 distincts ;
- *équilibrée* B lorsque $\bigoplus_{i=0}^{255} x_i = 0$;
- *constante* C lorsque tous les x_i sont égaux

1. Montrer que (x_i) est $S \Rightarrow (x_i)$ est B et que (x_i) est $C \Rightarrow (x_i)$ est B

2. Montrer que :

(1) (x_i) et (y_i) sont S, C ou $B \Rightarrow (x_i \oplus y_i)$ est B

(2) (x_i) et (y_i) sont $C \Rightarrow (x_i \oplus y_i)$ est C

(3) (x_i) est S et (y_i) est $C \Rightarrow (x_i \oplus y_i)$ est S

(4) (x_i) et (y_i) sont $S \not\Rightarrow (x_i \oplus y_i)$ est C ou S

(5) (x_i) est S et (y_i) est $B \not\Rightarrow (x_i \oplus y_i)$ est C ou S

3. Montrer que si f est une bijection et (x_i) est S, alors $(f(x_i))$ est S.

4. Montrer que si L est une application linéaire (resp. g une application affine) et (x_i) est B, alors $(L(x_i))$ (resp. $(g(x_i))$) est B.

5. Montrer par un contre exemple que si f est une application bijective et (x_i) une collection B, alors $(f(x_i))$ peut ne pas être B.

6. Quelle est la probabilité qu'une collection aléatoire soit B ? (sous l'hypothèse que les 256 vecteurs sont choisis indépendants et selon une loi uniforme)

Notation. Pour toute variable X de taille 16 octets, $X[i]$ désigne l'octet d'indice i pour $i = 0, \ldots, 15$. Lorsque X est représenté sous forme d'un tableau 4×4, les indices se lisent de haut en bas puis de gauche à droite. Ainsi,

$$X = (X[0], \ldots, X[15]) = \begin{array}{|c|c|c|c|} \hline X[0] & X[4] & X[8] & X[12] \\ \hline X[1] & X[5] & X[9] & X[13] \\ \hline X[2] & X[6] & X[10] & X[14] \\ \hline X[3] & X[7] & X[11] & X[15] \\ \hline \end{array}$$

On considère une collection de 256 clairs d'AES-4T : $P_i = (x_i, 0, \ldots, 0)$, où les x_i saturent $\mathbb{F}_2^8$. On note

symboliquement la collection

S	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

et C_i les chiffrés correspondants.

7. Montrer que l'état de la collection au fur et à mesure des 4 premiers tours d'AES est

	après SubBytes	après ShiftRows	après MixColumns	après AddRoundKey	
Tour 0				S C C C C C C C C C C C C C C C	
Tour 1	S C C C C C C C C C C C C C C C	S C C C C C C C C C C C C C C C	S C C C S C C C S C C C S C C C	S C C C S C C C S C C C S C C C	
Tour 2	S C C C S C C C S C C C S C C C	S C C C C C C S C C S C C S C C	S S S S S S S S S S S S S S S S	S S S S S S S S S S S S S S S S	
Tour 3	S S S S S S S S S S S S S S S S	S S S S S S S S S S S S S S S S	B B B B B B B B B B B B B B B B	B B B B B B B B B B B B B B B B	$= Z$
Tour 4	? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?	? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?		? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?	$= C$

On note Z_i la valeur des chiffrés intermédiaires à la fin du 3me tour. Pour $k_{\text{test}} \in \mathbb{F}_2^8$, on calcule la collection d'octets $y_i = Sbox^{-1}(C_i[0] \oplus k_{\text{test}})$.

8. Montrer que, si k_{test} est le bon octet de sous-clé, c'est-à-dire si $k_{\text{test}} = sk_4[0]$, on a $y_i = Z_i[0]$. En déduire la nature de la collection (y_i).

9. Si $k_{\text{test}} \neq sk_4[0]$, exprimer y_i à l'aide des octets $Z_i[0], k_{\text{test}}$ et $sk_4[0]$. On admet que, dans ce cas, la collection (y_i) se comporte comme une collection aléatoire. Quelle est la probabilité que la collection (y_i) soit B ? (On parle alors de faux candidat).

10. En déduire une stratégie pour trouver des candidats à la valeur de $sk_4[0]$.

11. Montrer que le nombre de faux candidats suit une loi binomiale $\mathcal{B}\left(255, \frac{1}{256}\right)$.

12. En déduire le nombre moyen de faux candidats.

On recommence l'opération en prenant comme collection de clairs $P_i^* = (x_i, 1, \ldots, 1)$, où (x_i) est encore saturante :

S	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

13. Justifier que la suite de l'attaque fonctionne comme précédemment.

14. Fort de cette observation, proposer une stratégie pour éliminer les faux candidats.

15. Montrer comment on peut trouver la valeur des autres octets de sk_4.

16. Quelle est alors la complexité globale de l'attaque ?

1. $S \Rightarrow B$.
Soit (x_i) une collection S : comme $x_0, \ldots, x_{255}$ sont (tous) les éléments de $\mathbb{F}_2^8$, il suffit de voir que ceux-ci se somment à 0 composante à composante. Or il y a 2^7 vecteurs dont la première composante vaut 1, et 2^7 vecteurs dont la première composante vaut 0, donc la première composante de la somme vaut 0. Le même raisonnement vaut pour chaque composante, et la conclusion suit.

$C \Rightarrow B$.
Comme on somme un nombre pair de fois une même quantité, on obtient 0.

2. (1) : d'après la question précédente il suffit de montrer que

$$(x_i) \text{ et } (y_i) \text{ sont } B \Rightarrow (x_i \oplus y_i) \text{ est } B$$

ce qui est facile en revenant à la définition
(2) est évident
(3) tient au fait que, pour y fixé, l'application $\mathbb{F}_2^8 \to \mathbb{F}_2^8$, $x \mapsto x \oplus y$ est un permutation de $\mathbb{F}_2^8$, égale à sa propre inverse d'ailleurs
(4) et (5) : Prenons $(x_i) = (\{\texttt{00}\}, \{\texttt{01}\}, \ldots, \{\texttt{ff}\})$ et $(y_i) = (\{\texttt{01}\}, \{\texttt{00}\}, \ldots, \{\texttt{ff}\})$. Alors
- (x_i) est S,
- (y_i) est S et donc B
- $(x_i \oplus y_i) = (\{\texttt{01}\}, \{\texttt{01}\}, \{\texttt{00}\}, \ldots, \{\texttt{00}\})$ n'est ni S ni C

3. C'est une conséquence directe de la définition d'une bijection : c'est une permutation des éléments de $\mathbb{F}_2^8$. Si tous les éléments de $\mathbb{F}_2^8$ sont présents une et une seule fois dans la collection (x_i), il en est de même dans la collection $(f(x_i))$.

4. Pour L linéaire, il suffit d'écrire

$$\bigoplus_{i=0}^{255} L(x_i) = L\left(\bigoplus_{i=0}^{255} x_i\right) = L(0) = 0$$

la première et la troisième égalités étant valides du fait que f est linéaire.

Pour g affine, on écrit $g(x) = L(x) + c$ où c est une constante et L linéaire. Ainsi

$$\bigoplus_{i=0}^{255} g(x_i) = \bigoplus_{i=0}^{255} (L(x_i) + c) = \bigoplus_{i=0}^{255} L(x_i) + \bigoplus_{i=0}^{255} c = 0$$

Dans l'avant-denier membre de l'égalité, le premier terme est nul car L est linéaire et on vient de le démontrer. Le deuxième est nul car on somme une constante un nombre pair (256) fois.

5. La Q3 nous dit qu'il faut chercher un contre exemple avec pour (x_i) une collection B qui n'est pas S, et pour f une application qui ne soit pas affine (en particulier pas linéaire). On prend :

— la collection $(x_i) = (\{\texttt{01}\}, \{\texttt{02}\}, \{\texttt{03}\}, \{\texttt{00}\}, \ldots, \{\texttt{00}\})$. Elle en effet B mais pas S, la vérification est presque immédiate.

— la bijection f de $\mathbb{F}_2^8$ qui échange les éléments $\{\texttt{03}\}$ et $\{\texttt{07}\}$ et laisse les autres invariants. On a en particulier

$$f(\{\texttt{00}\}) = \{\texttt{00}\},\ f(\{\texttt{01}\}) = \{\texttt{01}\},\ f(\{\texttt{02}\}) = \{\texttt{02}\},\ f(\{\texttt{03}\}) = \{\texttt{07}\}$$

Alors

$$(f(\{\texttt{01}\}), f(\{\texttt{02}\}), f(\{\texttt{03}\}), f(\{\texttt{00}\}), \ldots, f(\{\texttt{00}\})) = (\{\texttt{01}\}, \{\texttt{02}\}, \{\texttt{07}\}, \{\texttt{00}\}, \ldots, \{\texttt{00}\})$$

est une collection qui n'est B – vérification là encore presque immédiate.

6. La somme d'une collection aléatoire de vecteurs indépendants et uniformes est un élément aléatoire uniforme de $\mathbb{F}_2^n$ (résultats facile mais non trivial de probabilité admis ici). La probabilité recherchée est exactement la probabilité que cette somme soit nulle. Elle est donc égale à 2^{-8}.

7. Cette évolution est une application des résultats des questions 1,2,3 aux équations algébriques qui décrivent le fonctionnement de l'AES.

— L'étape ShiftRows ne pose pas de problème : comme elle ne fait aucun calcul mais ne fait que déplacer les octets, les collections restent de même nature et changent de position dans le tableau.

— Les étapes SubBytes et AddRoundKey ne mélangent pas les octets du tableau entre eux. A chaque position dans le tableau, elles appliquent une bijection à la collection : en vertu de Q2 (3) pour AddRoundKey et en vertu de Q3 pour SubBytes, elles opèrent $S \to S$. Par ailleurs, elles vérifient évidemment $C \to C$. Enfin, AddRoundKey est affine donc opère $B \to B$ en vertu de Q4, et SubBytes opère $B \to ?$ en vertu de Q5.

— L'étape MixColumns demande un examen des équations. On va faire la preuve pour l'octet de position 0, les 15 autres positions se démontrent de manière analogue.

$$X_{\text{out}}[0] = \{02\} \bullet X_{\text{in}}[0] \oplus \{03\} \bullet X_{\text{in}}[1] \oplus \{01\} \bullet X_{\text{in}}[2] \oplus \{01\} \bullet X_{\text{in}}[3]$$

Les applications $x \mapsto \{01\} \bullet x$, $x \mapsto \{02\} \bullet x$, $x \mapsto \{03\} \bullet x$, sont $\mathbb{F}_2$-linéaires bijectives sur $\mathbb{F}_2^8$ donc opèrent $S \to S,\ C \to C,\ B \to B$. Dans les 3 premiers tours, l'entrée de MixColumns ne fait intervenir que des collections S ou C. On peut écrire symboliquement

$$\begin{array}{rcl} \text{Collection des } X_{\text{out}}[0] & = & \{02\} \bullet (S \text{ ou } C) \ \oplus\ \{03\}(S \text{ ou } C) \ \oplus\ \{01\}(S \text{ ou } C) \ \oplus\ \{01\}(S \text{ ou } C) \\ & = & (S \text{ ou } C) \oplus (S \text{ ou } C) \oplus (S \text{ ou } C) \oplus (S \text{ ou } C) \end{array}$$

D'après Q2 (2) et (3), la somme symbolique de droite est une collection

— C si les 4 termes sont C

— S si l'un exactement des termes est S et les 3 autres sont C (cas des tours 1 et 2)

— B si au moins 2 termes sont S (cas du tour 3)

8. Que $y_i = Z_i[0]$ est évident par définition ; dans ce cas la collection y_i est B d'après la question 7.

9. $y_i = Sbox^{-1}(C_i[0] \oplus k_{\text{test}}) = Sbox^{-1}(Sbox(Z_i[0] \oplus \underbrace{sk_4[0] \oplus k_{\text{test}}}_{\neq 0})$. La probabilité que la collection (y_i) soit B est $2^{-8} = 1/256$, d'après la question 5.

10. On effectue le calcul des (y_i) pour tout $k_{\text{test}} \in \mathbb{F}_2^8$ et on conserve la ou les valeurs de k_{test} pour lesquelles la collection (y_i) est B. Il y en a au moins une : $k_{\text{test}} = sk_4[0]$).

11. La probabilité d'un faux candidat est celle calculée à la Q9. Donc elle vaut$1/256$. Comme il y a 255 mauvaises valeurs, le nombre de faux candidats suit une loi binomiale $\mathcal{B}(255, 1/256)$.

12. Le nombre moyen de faux candidats est de $255/256 \approx 1$.

13. L'évolution de l'état sur les 4 tours suit le même schéma que précédemment.

14. Quand on recommence tout l'algorithme en remplaçant 0 par la valeur 1, des candidats sont à nouveau proposés pour la valeur $sk_4[0]$. Parmi eux, il y a la bonne valeur plus des faux candidats, très probablement différents des faux candidates précédents. La bonne valeur est donc celle que l'on retrouve dans les deux listes de candidats.

15. On peut dérouler toute l'attaque par un procédé analogue en ciblant l'octet $sk_4[1], sk_4[2], \ldots sk_4[15]$ au lieu de l'octet $sk_4[0]$. Le critère de sélection des candidats (le vrai et les éventuels faux) sera le même.

16. En supposant qu'il faille deux familles de paires (clairs, chiffres) pour éliminer les faux candidats, la complexité en temps est $16 \times 2 \times 2^8 = 2^{13}$ opérations, la complexité mémoire étant le stockage de 2^9 couples (clair, chiffre), soit $2^9 \times 2 \times 16 = 2^{14}$ octets.

4 Modes de chiffrement blocs

4.1 Tutoriel jouet

ChifSymMode-04

Exercice 23. **ECB, CTR, CBC jouet**

Dans le cadre de cet exercice, on considère une primitive bloc de 8 bits avec clé 8 bits définie par :

$$E_K(X) = \texttt{AES\_Sbox}(X \oplus K)$$

Soit P le texte clair constitué de 5 octets (donc ici 5 blocs) : `3243f6a888`, et la clé K égale à `27`. Calculer le texte chiffré

1. en mode ECB
2. en mode CTR avec $IV =$ `a7`
3. en mode CBC avec $IV =$ `b9`

Dans le trois cas, il faut commencer par partitionner le clair en blocs de taille adéquate :

Clair : `32 43 f6 a8 88`

Pour cet exemple jouet il n'y a pas de padding à adopter. Il suffit maintenant d'appliquer les équations et de faire les applications numériques pour les trois modes.

1. `ECB Mode`

```
ciphertext block 1 = Enc_{key}(plaintext block 1)
                   = Enc_{27}(32)
                   = AES_Sbox(27 xor 32)
                   = AES_Sbox(15)
                   = 59
```

```
ciphertext block 2 = Enc_{key}(plaintext block 2)
                   = Enc_{27}(43)
                   = AES_Sbox(27 xor 43)
                   = AES_Sbox(64)
                   = 43
ciphertext block 3 = Enc_{key}(plaintext block 3)
                   = Enc_{27}(f6)
                   = AES_Sbox(27 xor f6)
                   = AES_Sbox(d1)
                   = 3e
ciphertext block 4 = Enc_{key}(plaintext block 4)
                   = Enc_{27}(a8)
                   = AES_Sbox(27 xor a8)
                   = AES_Sbox(8f)
                   = 73
ciphertext block 5 = Enc_{key}(plaintext block 5)
                   = Enc_{27}(88)
                   = AES_Sbox(27 xor 88)
                   = AES_Sbox(af)
                   = 79
Ciphertext:  59 43 3e 73 79
```

2. CTR Mode

```
Block cipher input 1 = IV + 1
                     = a7 + 1
                     = a8
Block cipher output 1 = Enc_{key}(IV + 1)
                      = Enc_{27}(a8)
                      = AES_Sbox(27 xor a8)
                      = AES_Sbox(15)
                      = 73
Ciphertext block 1 = (Plaintext block 1) xor (block cipher output 1)
                   = 32 xor 73
                   = 41
Block cipher input 2 = IV + 2
                     = a7 + 2
                     = a9
Block cipher output 2 = Enc_{key}(IV + 2)
                      = Enc_{27}(a9)
                      = AES_Sbox(27 xor a9)
                      = AES_Sbox(64)
                      = 19
Ciphertext block 2 = (Plaintext block 2) xor (block cipher output 2)
                   = 43 xor 19
                   = 5a
Block cipher input 3 = IV + 3
                     = a7 + 3
                     = aa
Block cipher output 3 = Enc_{key}(IV + 3)
                      = Enc_{27}(aa)
                      = AES_Sbox(27 xor aa)
                      = AES_Sbox(d1)
                      = 5d
Ciphertext block 3 = (Plaintext block 3) xor (block cipher output 3)
                   = f6 xor 5d
                   = ab
Block cipher input 4 = IV + 4
                     = a7 + 4
                     = ab
Block cipher output 4 = Enc_{key}(IV + 4)
                      = Enc_{27}(ab)
                      = AES_Sbox(27 xor ab)
                      = AES_Sbox(8f)
                      = 64
Ciphertext block 4 = (Plaintext block 4) xor (block cipher output 4)
                   = a8 xor 64
                   = cc
Block cipher input 5 = IV + 5
                     = a7 + 5
```

```
                      = ac
Block cipher output 5 = Enc_{key}(IV + 5)
                       = Enc_{27}(ac)
                       = AES_Sbox(27 xor ac)
                       = AES_Sbox(af)
                       = 3d
Ciphertext block 5 = (Plaintext block 5) xor (block cipher output 5)
                   = 88 xor 3d
                   = b5
Ciphertext: 41 5a ab cc b5
```

3. CBC Mode

```
Ciphertext block 1 = Enc_{key}( (plaintext block 1) xor IV )
                   = Enc_{27}(32 xor b9)
                   = Enc_{27}(8b)
                   = AES_Sbox(27 xor 8b)
                   = AES_Sbox(ac)
                   = 91
Ciphertext block 2 = Enc_{key}( (plaintext block 2) xor (ciphertext bolck 1) )
                   = Enc_{27}(43 xor 91)
                   = Enc_{27}(d2)
                   = AES_Sbox(27 xor d2)
                   = AES_Sbox(f5)
                   = e6
Ciphertext block 3 = Enc_{key}( (plaintext block 3) xor (ciphertext bolck 2) )
                   = Enc_{27}(f6 xor e6)
                   = Enc_{27}(10)
                   = AES_Sbox(27 xor 10)
                   = AES_Sbox(37)
                   = 9a
Ciphertext block 4 = Enc_{key}( (plaintext block 4) xor (ciphertext bolck 3) )
                   = Enc_{27}(a8 xor 9a)
                   = Enc_{27}(32)
                   = AES_Sbox(27 xor 32)
                   = AES_Sbox(15)
                   = 59
Ciphertext block 5 = Enc_{key}( (plaintext block 5) xor (ciphertext bolck 4) )
                   = Enc_{27}(88 xor 59)
                   = Enc_{27}(d1)
                   = AES_Sbox(27 xor d1)
                   = AES_Sbox(f6)
                   = 42
Ciphertext: 91 e6 9a 59 42
```

4.2 Exemples d'attaques

ChifSymMode-02

Exercice 24. Une attaque sur le mode CBC en sécurité sémantique

Une fonction de chiffrement sur des blocs de n bits $E_K : x \mapsto y$ est utilisée en mode CBC pour chiffrer un message $M = (M[1], M[2], \ldots, M[q])$. On obtient un cryptogramme $C = (C[0] = IV, C[1], C[2], \ldots, C[q])$ où IV est le vecteur d'initialisation.

1. Montrer que si deux blocs du chiffré $C[i]$ et $C[j]$ sont identiques, avec $1 \leq i < j \leq q$, alors on dispose d'une information sur les blocs de clairs $M[i]$ et $M[j]$ (montrer que $M[i] \oplus M[j]$ est connu).

2. On suppose que les blocs de chiffré sont répartis aléatoirement. Quelle est la probabilité $p(n, q)$ pour qu'il existe deux blocs de n bits égaux dans un cryptogramme de q blocs ?

3. On suppose $n = 64$. Calculer q pour que $p(n, q) \geq 1/2$.

1. Rappelons que $C[i] = E_K(C[i-1] \oplus M[i])$ et $C[j] = E_K(C[j-1] \oplus M[j])$. Comme E_K est une bijection l'égalité $C[i] = C[j]$ est équivalente à $C[i-1] \oplus M[i] = C[j-1] \oplus M[j]$, et donc $M[i] \oplus M[j] = C[i-1] \oplus C[j-1]$. Comme $C[i-1]$ et $C[j-1]$ sont connus, $M[i] \oplus M[j]$ est connu. Ceci contrevient à la sécurité sémantique selon laquelle $M[i] \oplus M[j]$ doit se comporter comme une variable aléatoire uniformément distribuée sur $\{0,1\}^n$.

2. Les blocs étant supposés indépendants, calculer les $C[i]$ équivaut à effectuer un tirage aléatoire successif avec remise dans $\{0,1\}^n$. La probabilité que deux blocs soient identiques est donc la probabilité de collision caractérisée par l'exercice "paradoxe des anniversaires" : $p(n,q) = \mathsf{PrCol}(2^n, q)$.

3. C'est une application directe du même exercice. L'application numérique donne, pour $n = 64$, $q \geq 1,17 \times 2^{32} = 2^{32,2\ldots}$.

ChifSymMode-03

Exercice 25. Attaque en distingueur left-right du mode CBC

Soit l'attaquant A utilisant un oracle $\mathcal{O}$ left-right du mode CBC avec une primitive blocs E_K, n la taille du bloc.

---

1. pour $i = 1, \ldots, q$
2. demander (requêtes d'un seul bloc)

$$(C_i[0] = IV_i, C_i[1]) \leftarrow \mathcal{O}(\underbrace{(i_{n-1}i_{n-2}\ldots i_1 i_0)_2}_{\text{left}}, \underbrace{0^n}_{\text{right}})$$

3. Déterminer S l'ensemble des couples (i,j) tels que $IV_i = IV_j$
4. si S est non vide
5. choisir (i,j) au hasard dans S
6. si $C_i[1] = C_j[1]$ parier "right"
7. sinon parier "left"

---

1. Montrer que la probabilité que A parie "right" dans le cas du monde "right" est égale à $\mathsf{PrCol}(2^n, q)$.
2. Montrer que la probabilité que A parie "right" dans le cas du monde "left" est égale à 0.
3. En déduire l'avantage de l'attaquant, puis que le schéma est distinguable left-right avec un nombre de requêtes de l'ordre de $2^{n/2}$.

1. Si le monde est "right", le message à chiffrer est toujours le même $M_i[1] = 0^n$. Donc $IV_i = IV_j$ si, et seulement si $C_i[1] = C_j[1]$. Ainsi, l'attaquant retourne "right" dès qu'il y a collision sur les IV puisque l'autre condition est alors vérifiée *de facto*. Cet événement intervient avec probabilité $\mathsf{PrCol}(2^n, q)$.

2. Si le monde est "left", la condition $IV_i = IV_j$ équivaut à $IV_i \oplus (i_{n-1}i_{n-2}\ldots i_1 i_0)_2 \neq IV_j \oplus (j_{n-1}j_{n-2}\ldots j_1 j_0)_2$ et donc à $C_i[1] \neq C_j[1]$. Il est clair que l'attaquant n'est jamais en situation de répondre "right".

3. L'avantage de l'attaquant est $\mathsf{PrCol}(2^n, q)$, donc non négligeable pour q de l'ordre de $2^{n/2}$.

5 Chiffrement asymétrique

5.1 Tutoriels RSA et Elgamal

ChifAsym-03

Exercice 26. **Un calcul RSA**

On considère le chiffrement RSA avec les paramètres $p = 47$, $q = 59$, $n = pq = 2773$ et $e = 3$.

1. Calculer $\phi(n)$ et vérifier que $\gcd(e, \phi(n)) = 1$.
2. Trouver l'entier $d \in \{0, \ldots, \phi(n) - 1\}$ tel que $ed = 1 \bmod \phi(n)$.
3. Calculer le chiffré c du message $m = 1190$.
4. Déchiffrer c and vérifier le résultat

. On explicitera les exponentiations modulaires par la méthode binaire.

1. On peut calculer facilement $\phi(n)$ par la formule explicite puisqu'on connait la factorisation de n : $\phi(n) = (47 - 1)(59 - 1) = 2668$.
   En toute généricité, (et efficacité), on utilise l'algorithme d'Euclide pour calculer PGCD(2668,3) et vérifier qu'il est égal à 1. Ici comme $e = 3$ est premier, vérifier la condition du pgcd revient à vérifier que 3 ne divise pas 2668, ce qui est immédiat.
2. On peut procéder
   — par algorithme d'Euclide étendu qui fournit la relation de Bezout :

   $$1 = 1 \times 2668 - 3 \times 889$$

   de sorte que $d = -889 = 1779 \bmod 2688$.

```
This is Extended Euclid Algorithm for (a,b) = (2668,3)

Step -1 (initialization)
r[-1]: a = 2668
u[-1]: 1
v[-1]: 0
Relation r[-1] = a*(u[-1])+b*(v[-1]) :  2668 = 2668*(1)+3*(0)

Step 0 (initialization)
r[0]: b = 3
u[0]: 0
v[0]: 1
Relation r[0] = a*(u[0])+b*(v[0]) :  3 = 2668*(0)+3*(1)

Step 1 (while loop)
Euclidean division r[-1] div r[0] = 2668 div 3   quotient: q=889   remainder:
r[1]=1
==> (*) r[1] = r[-1] - q*r[0] i.e. 1 = 2668 - 889*3
Computing u[1] and v[1] following the order 2 recurrence (*)
u[1] = u[-1] - q*u[0] = 1 - 889*(0) = 1
v[1] = v[-1] - q*v[0] = 0 - 889*(1) = -889
Relation r[1] = a*(u[1])+b*(v[1]) :  1 = 2668*(1)+3*(-889)

Step 2 (while loop)
Euclidean division r[0] div r[1] = 3 div 1   quotient: q=3   remainder: r[2]=0
==> (*) r[2] = r[0] - q*r[1] i.e. 0 = 3 - 3*1
Computing u[2] and v[2] following the order 2 recurrence (*)
u[2] = u[0] - q*u[1] = 0 - 3*(1) = -3
v[2] = v[0] - q*v[1] = 1 - 3*(-889) = 2668
Relation r[2] = a*(u[2])+b*(v[2]) :  0 = 2668*(-3)+3*(2668)

At penultimate step 1, get GCD and Bezout coefficients
```

```
Gcd: r[1] = 1    Bezout coefficients: (u[1], v[1]) = (1, -889)

At last step 2, get quotients of initial integers by their GCD
(a/gcd(a,b) , b/gcd(a,b)) = (2668/1, 3/1) = (|v[2]|, |u[2]|) = 2668, 3
```

— par recherche exhaustive en calculant tous les multiples de 3 modulo 2668

— utiliser une ruse *ad'hoc* permettant un calcul très rapide au papier-crayon : constatant que $2667 = 3 \times 889 = -1 \bmod 2668$, on a $1 = 3 \times (-889) \bmod 2668$.

3. On applique la formule et on calcule $c = 1190^3 \bmod 2773 = 1354$.

```
Computing 1190^3 mod 2773 by binary exponentiation
Binary sequence of exponent is [ 1, 1 ] (leftmost significant bit)
Step 1: bit of sequence is 1
  Start: y = 1
  After squaring : y = 1^2 mod 2773 = 1
  After multiplying by 1190: y = 1*1190 mod 2773 = 1190
Step 2: bit of sequence is 1
  Start: y = 1190
  After squaring : y = 1190^2 mod 2773 = 1870
  After multiplying by 1190: y = 1870*1190 mod 2773 = 1354
Result is: 1354
```

4. Le déchiffrement opère ainsi : $m = 1354^{1779} \bmod 2773 = 1190$. On retrouve le clair initial.

```
Computing 1354^1779 mod 2773 by binary exponentiation
Binary sequence of exponent is [ 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1 ] (leftmost significant bit)
Step 1: bit of sequence is 1
  Start: y = 1
  After squaring : y = 1^2 mod 2773 = 1
  After multiplying by 1354: y = 1*1354 mod 2773 = 1354
Step 2: bit of sequence is 1
  Start: y = 1354
  After squaring : y = 1354^2 mod 2773 = 363
  After multiplying by 1354: y = 363*1354 mod 2773 = 681
Step 3: bit of sequence is 0
  Start: y = 681
  After squaring : y = 681^2 mod 2773 = 670
Step 4: bit of sequence is 1
  Start: y = 670
  After squaring : y = 670^2 mod 2773 = 2447
  After multiplying by 1354: y = 2447*1354 mod 2773 = 2276
Step 5: bit of sequence is 1
  Start: y = 2276
  After squaring : y = 2276^2 mod 2773 = 212
  After multiplying by 1354: y = 212*1354 mod 2773 = 1429
Step 6: bit of sequence is 1
  Start: y = 1429
  After squaring : y = 1429^2 mod 2773 = 1113
  After multiplying by 1354: y = 1113*1354 mod 2773 = 1263
Step 7: bit of sequence is 1
  Start: y = 1263
  After squaring : y = 1263^2 mod 2773 = 694
  After multiplying by 1354: y = 694*1354 mod 2773 = 2402
Step 8: bit of sequence is 0
  Start: y = 2402
  After squaring : y = 2402^2 mod 2773 = 1764
Step 9: bit of sequence is 0
  Start: y = 1764
  After squaring : y = 1764^2 mod 2773 = 390
Step 10: bit of sequence is 1
  Start: y = 390
  After squaring : y = 390^2 mod 2773 = 2358
  After multiplying by 1354: y = 2358*1354 mod 2773 = 1009
Step 11: bit of sequence is 1
  Start: y = 1009
  After squaring : y = 1009^2 mod 2773 = 390
  After multiplying by 1354: y = 390*1354 mod 2773 = 1190
Result is: 1190
```

ChifAsym-09

Exercice 27. **Un calcul Elgamal**

On considère le chiffrement Elgamal avec les paramètres $p = 107$, $g = 2$, $\ell = k_{\text{pr}} = 71$

1. Montrer que 107 est un nombre premier sûr
2. Vérifier que 2 est un générateur de $\mathbb{F}_{107}^*$.
3. Calculer la valeur du h de la clé publique.
4. Soit $m = 62$ le texte clair, Calculer le chiffré en prenant $r = 72$ pour la variable auxiliaire.
5. Déchiffrer le message $(37, 85)$

1. On a $107 = 2 \times 53 + 1$ et 53 est premier. Donc c'est bon.
2. On peut revenir à la définition, et montrer par brute force que $\mathbb{F}_{107}^* = \{1, 2, 2^2, \ldots, 2^{105}\}$. Ceci est possible car 107 est un nombre premier artificiellement tout petit pour les besoins du tutoriel.

   On peut aussi raisonner sur l'ordre de 2 mod 107. On sait que cet ordre divise 106, donc s'il n'est pas ≤ 53 il est égal à 106 (ce que nous voulons). On vérifie par le calcul exhaustif que $2^i mod 107 \neq 1$ pour $i = 0, \ldots, 53$. En fait cela ne change pas fondamentalement de l'idée précédente mais on a divisé le nombre de calculs par 2.

   On peut enfin appliquer le critère spécial d'un générateur dans un groupe abélien fini. Comme $106 = 2 \times 53$ il suffit de vérifier

   $$2^{\frac{106}{2}} \neq 1, \quad 2^{\frac{106}{53}} \neq 1 \mod 107$$

   Or $2^{53} = 52 \neq 1 \mod 107$ et $2^2 = 4 \neq 1$, comme souhaité.
3. $h = 2^{71} \mod 107 = 6$.
4. C'est une application numérique basique :

   $$(c_1, c_2) = (2^{72}, 62 \times 6^{72}) = (12, 53)$$

5. Idem :

   $$m = 85 \times 37^{-71} = 85 \times 37^{106-71} = 85 \times 37^{35} = 62 \mod 107$$

ChifAsym-08

Exercice 28. **Tuto RSA-OAEP**

On considère le chiffrement RSA avec les paramètres $p = 239$, $q = 269$, $n = pq = 64291$. Comme $2^{15} < 64291 < 2^{16}$, il s'agit d'un "RSA 16 bits". On pose $e = 257$ l'exposant public (de chiffrement). On réalise un construction OAEP jouet avec :

— espace des clairs : $\{0, \ldots, 1023\}$, donc le clair est "sur 10 bits"
— les champ fixe '000' (sur 3 bits)
— le champ aléatoire r sur 3 bits.
— la fonction $G : \{0,1\}^3 \to \{0,1\}^3,\ x \mapsto (1962(x+1)+5)^2 \mod 2^{13}$
— la fonction $H : \{0,1\}^{13} \to \{0,1\}^3, x \mapsto ((x + 1971)^2 \div 11) \mod 2^3$

1. Calculer le chiffré c_1 du message $m_1 = 985$ avec l'aléa $r_1 = 5$

2. Calculer $\phi(n)$ et vérifier que $\gcd(e, \phi(n)) = 1$.

3. Trouver l'entier $d \in \{0, \ldots, \phi(n) - 1\}$ exposant privé (de déchiffrement).

4. Déchiffrer le résultat obtenu pour vérifier le texte clair et sa validité.

5. Déchiffrer $c_2 = 25314$ et vérifier la validité du clair m_2 trouvé.

1. On applique les formules. À noter que l'on n'a pas besoin de la factorisation de n.

```
This is RSA OAEP encryption with:
Parameters: modulus n=64291, public exponent e=257
Plaintext (10-bit) m: 985, [ 1, 1, 1, 1, 0, 1, 1, 0, 0, 1 ]
Random value (3-bit) r:  5, [ 1, 0, 1 ]
m|000:  7880 [ 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0 ]
G(r):  7169 [ 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 ]
X = m|000 + G(r):  713 [ 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1 ]
H(X):  0 [ 0, 0, 0 ]
Y = H(X) + r:  5 [ 1, 0, 1 ]
X|Y:  5709 [ 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1 ]
C = (X|Y)^e mod n :  5377 [ 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1 ]
```

2. On peut calculer facilement $\phi(n)$ par la formule explicite puisqu'on connait la factorisation de n : $\phi(n) = (239 - 1)(269 - 1) = 63784$.

L'algorithme d'Euclide étendu donne l'identité de Bezout : $1 = 257 \cdot 22585 + 63784 \cdot (-91)$, ce qui prouve que le pgcd recherché est bien égal à 1.

3. La même identité de Bezout donne $257 \cdot 22585 = 1 \bmod 63784$, de sorte que $d = 22585$.

4. On applique les formules de déchiffrement

```
This is RSA OAEP decryption with:
Parameters:modulus n=64291, public exponent e=257, private exponent d=22585
Ciphertext (16-bit) c: 5377, [ 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1 ]
After exponentiation with private exponent: 5709, [ 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1 ]
After parsing this output: (X (13-bit) | Y (3-bit)) : (713 | 5), [ 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1 ], [ 1, 0, 1 ]
Retrieving the (auxiliary and useless) random value r: 5, [ 1, 0, 1 ]
Retrieving the (expected) padded plaintext m|000 : 7880, [ 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0 ]
Recovered padded plaintext (13-bit) is valid
Actual plaintext (10-bit) m: 985, [ 1, 1, 1, 1, 0, 1, 1, 0, 0, 1 ]
```

5. Simple application des formules là encore...

```
This is RSA OAEP decryption with:
Parameters:modulus n=64291, public exponent e=257, private exponent d=22585
Ciphertext (16-bit) c: 25314, [ 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0 ]
After exponentiation with private exponent: 37902, [ 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0 ]
After parsing this output: (X (13-bit) | Y (3-bit)) : (4737 | 6), [ 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1 ], [ 1, 1, 0 ]
Retrieving the (auxiliary and useless) random value r: 2, [ 0, 1, 0 ]
Retrieving the (expected) padded plaintext m|000 : 6280, [ 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0 ]
Recovered padded plaintext (13-bit) is valid
Actual plaintext (10-bit) m: 785, [ 1, 1, 0, 0, 0, 1, 0, 0, 0, 1 ]
```

5.2 Sécurité du RSA

Exercice 29. Sécurité des paramètres RSA : protéger $\phi(n)$ et d aussi bien que les nombres p, q

Soit $n = pq$ produit de deux nombres premiers. Eve connaît n.

1. Montrer que les trois propositions suivantes sont équivalentes :
   (1) Eve connaît p ou q ;
   (2) Eve connaît p et q ;
   (3) Eve connaît $\phi(n)$.

À partir de $n, p, q, \phi(n)$, on fabrique les exposants RSA e public et d privé. Eve connaît donc e.

2. Montrer que se valent :
   (1) Eve connaît p et q ;
   (4) Eve connaît d.

Pour (4) $\Rightarrow$ (1), on pourra utiliser le lemme (admis) suivant

> **Lemme** *L'entier 1 a quatre racines carrées modulo $n = pq$: $1, -1, b, -b$. De plus, la connaissance de b permet de factoriser n.*

1. (1) $\Rightarrow$ (2). Si Eve connaît l'un des deux nombres p ou q, il est évident qu'elle connaît l'autre par simple division puisqu'elle connaît $n = pq$.

(2) $\Rightarrow$ (3). Eve connaît p et q donc peut calculer $\phi(n) = (p-1)(q-1)$.

(3) $\Rightarrow$ (2). Nous avons $\phi(n) = (p-1)(q-1) = pq - (p+q) + 1 = n - (p+q) + 1$. Si Eve connaît $\phi(n)$, elle connaît donc la somme $(p+q)$: elle vaut $n - \phi(n) + 1$. Par ailleurs, Eve connaît le produit pq (qui vaut n). Donc p et q sont solutions de l'équation du second degré $x^2 - (n - \phi(n) + 1)x + n = 0$.

2. (1) $\Rightarrow$ (4). Si Eve connaît p et q, d'après la question précédente, Eve peut calculer $\phi(n)$. L'algorithme d'Euclide étendu lui permet de trouver u, v tels que $eu + \phi(n)v = 1$. Alors d n'est autre que $u \bmod \phi(n)$.

(4) $\Rightarrow$ (1). Supposons que Eve connaît d. Eve choisit $m \in \{1, \ldots, n-1\}$ au hasard, puis calcule $\text{pgcd}(m, n)$: c'est un diviseur de n, et c'est même un diviseur strict de n puisqu'il est $\leq m < n$. S'il est $\neq 1$, il est donc égal à p ou à q et c'est gagné. S'il est $= 1$, m est inversible modulo n. Comme on a $m^{ed} = m \bmod n$, on a aussi, en divisant par m, $m^{ed-1} = 1$. Maintenant, remarquons que, comme$p \neq q$, l'un au moins de ces deux nombres est impair, donc l'un au moins des nombres $p-1$ et $q-1$ est pair et finalement $\phi(n) = (p-1)(q-1)$ est pair. Comme e et d) sont premiers avec $\phi(n)$, ils sont tous deux impairs. Il suit $ed-1$ pair et on a l'égalité

$$(m^{\frac{ed-1}{2}})^2 = 1 \bmod n.$$

Autrement dit, $m^{\frac{ed-1}{2}}$ est une racine carrée de 1 modulo n. Pour continuer, nous utilisons le lemme : avec probabilité $1/2$, $m^{\frac{ed-1}{2}} = \pm b$. Et c'est gagné.

ChifAsym-05

Exercice 30. Attaque RSA du "commom modulus"

On suppose que deux utilisateurs de RSA ont le même modulus n, seul diffère leur exposant public e_1 et e_2 que l'on suppose premiers entre eux. On chiffre un même message $M \in \mathbb{Z}/n\mathbb{Z}$ à ces deux utilisateurs. Les messages chiffrés correspondants sont C_1 et C_2. On suppose que :

- C_2 est premier avec n ;
- C_1 est premier avec n.

(hypothèses réalistes : en pratique, la probabilité de ces deux événements est très élevée). Eve intercepte C_1 et C_2 et, connaissant n, e_1, e_2, fait les calculs suivants :

— Calcul de a et b tels que $ae_1 + be_2 = 1$
— $X = C_1^a C_2^b \bmod n$
À quoi mènent ces calculs ?

Comme e_1 et e_2 sont premiers entre eux, le théorème de Bezout assure l'existence de a et b vérifiant la relation demandée, et l'algorithme d'Euclide étendu permet de calculer efficacement a et b. Observons qu'alors l'un des deux nombres a ou b est positif et l'autre négatif. Supposons par exemple $a > 0$ et $b < 0$. Comme C_2 est premier avec n, il est légitime de parler de $C_2^{-1} \bmod n$, et il est facile de le calculer (encore l'algorithme d'Euclide étendu). Alors :

$$C_1^a \left(C_2^{-1}\right)^{-b} = C_1^a C_2^b = (M^{e^1})^a (M^{e_2})^b = M^{ae_1+be_2} = M \bmod n$$

Eve est donc capable de retrouver le texte clair M à partir de seules connaissances publiques C_1, C_2, n, e_1, e_2.

ChifAsym-06

Exercice 31. **Attaque "broadcast" du RSA**
Un petit exposant de chiffrement permet d'accélérer les calculs. Il est tentant de prendre autant que possible $e = 3$. À l'attention de trois utilisateurs de modules respectifs $n_1 < n_2 < n_3$ premiers entre eux 2 à 2, on chiffre un message $m \in [0, n_1 - 1]$, obtenant c_1, c_2, c_3. On pose $x = m^3$.

1. Montrer que $x < n_1 n_2 n_3$.

2. Écrire un système d'équations modulaires vérifié par x.

3. En déduire que l'on peut calculer m facilement par théorème chinois.

1. Comme $m < n_1 < n_2 < n_3$, on a $x = m^3 < n_1 n_2 n_3$.

2. Par définition, on a $c_1 = m^3 \bmod n_1$, $c_2 = m^3 \bmod n_2$, $c_3 = m^3 \bmod n_3$. Ainsi, x veifie le système de congruences suivant :

$$\begin{aligned} x &= c_1 \bmod n_1 \\ x &= c_2 \bmod n_2 \\ x &= c_3 \bmod n_3 \end{aligned}$$

3. Par le théorème des restes chinois, le système d'équations de la question 2 permet de retrouver $x \bmod n_1 n_2 n_3$. Mais, d'après la question 1, $x < n_1 n_2 n_3$, donc en fait on retrouve x exactement, c'est-à-dire m^3. Dès lors il suffit de prendre une racine cubique "ordinaire" (et non modulaire), ce qui est facile, pour retrouver m.

ChifAsym-07

Exercice 32. **RSA et clairs reliés**
Soit n un module RSA et $e = 3$ l'exposant public. Soit $m \in \{0, \ldots, n-1\}$ un message, $r \in \{0, \ldots, n-1\}$ un nombre aléatoire. On pose les deux chiffrés suivants :

$$c = m^e \bmod n, \qquad c' = (m + r)^e \bmod n$$

On suppose qu'une adversaire Eve connait c, c', r, ne connait pas m mais connait la lien entre les deux clairs sous-jacents, précisément qu'ils sont de la forme m and $m + r$.

1. Calculer les valeurs $c' - c + 2r^3$ et $c' + 2c - r^3$.

2. En déduire que, si $c' - c + 2r^3$ est inversible modulo n, Eve peut retrouver la valeur de m.

1. On a $c = m^3 \bmod n$ et $c' = (m+r)^3 = m^3 + 3m^2r + 3mr^2 + r^3 \bmod n$. Ainsi

$$\begin{aligned} c' - c + 2r^3 &= r(3m^2 + 3mr = 3r^2) \bmod n \\ c' + 2c - r^3 &= m(3m^2 + 3mr = 3r^2) \bmod n \end{aligned}$$

2. On a trivialement

$$m = r\frac{c' + 2c - r^3}{c' - c + 2r^3}$$

5.3 Diffie-Hellman

NegoCle-01

Exercice 33. Négociation de clé par Diffie-Hellman

1. On pose $n = 23$, $g = 5$ $a = 4$ et $b = 3$. Prouver que g est un générateur.
2. Quel est le secret calculé par Alice ?
3. Vérifier que le secret calculé par Bob est le même.

1. On peut vérifier que g génère bien tous les entiers modulo 23.

$5 = 5 \bmod 23$	$5^2 = 25 = 2 \bmod 23$	$5^3 = 5 \cdot 2 = 10 \bmod 23$
$5^4 = 10 \cdot 5 = 50 = 4 \bmod 23$	$5^5 = 5 \cdot 4 = 20 \bmod 23$	$5^6 = 5 \cdot 20 = 8 \bmod 23$
$5^7 = 5 \cdot 8 = 40 = 17 \bmod 23$	$5^8 = 5 \cdot 17 = 58 = 16 \bmod 23$	$5^9 = 5 \cdot 16 = 80 = 11 \bmod 23$
$5^{10} = 5 \cdot 11 = 55 = 9 \bmod 23$	$5^{11} = 5 \cdot 9 = 45 = 22 \bmod 23$	$5^{12} = 5 \cdot 12 = 60 = 18 \bmod 23$
$5^{13} = 5 \cdot 18 = 21 \bmod 23$	$5^{14} = 5 \cdot 21 = 13 \bmod 23$	$5^{15} = 5 \cdot 13 = 65 = 19 \bmod 23$
$5^{16} = 5 \cdot 19 = 95 = 3 \bmod 23$	$5^{17} = 5 \cdot 3 = 15 \bmod 23$	$5^{18} = 5 \cdot 15 = 6 \bmod 23$
$5^{19} = 5 \cdot 6 = 30 = 7 \bmod 23$	$5^{20} = 5 \cdot 7 = 35 = 12 \bmod 23$	$5^{21} = 5 \cdot 12 = 60 = 14 \bmod 23$
$5^{22} = 5 \cdot 14 = 70 = 1 \cdot \bmod 23$	$5^{23} = 5 \cdot 1 = 5 \bmod 23$	

2. — Alice a envoyé : $g^a \bmod n = 5^4 \bmod 23 = 4 \bmod 23$.
   — Alice a reçu : $10 = g^b \bmod n = 5^3 \bmod 23$.
   — Elle calcule le secret :$(g^b)^a \bmod n = 10^4 \bmod 23 = 18 \bmod 23$.
3. — Bob a envoyé : $g^b \bmod n = 5^3 \bmod 23 = 10 \bmod 23$.
   — Bob a reçu : $4 = g^a \bmod n = 5^4 \bmod 23$.
   — Il calcule le secret : $(g^a)^b \bmod n = 4^3 \bmod 23 = 18 \bmod 23$

NegoCle-02

Exercice 34. Échange de clé Diffie-Hellmann

1. On pose $n = 11$, $g = 2$ $a = 7$ et $b = 8$. Prouver que g est un générateur.
2. Quel est le secret calculé par Alice ?

3. Vérifier que le secret calculé par Bob est le même.

1. Calculons les puissances successives de 2 mod 11

i	0	1	2	3	4	5	6	7	8	9
$2^i \bmod 11$	1	2	4	8	5	10	9	7	3	6

En effet

$2^0 = 1 \bmod 11, 2^1 = 2 \bmod 11, 2^2 = 4 \bmod 11, 2^3 = 8 \bmod 11,$
$2^4 = 16 = 5 \bmod 11, 2^5 = 2 \times 24 = 2 \times 5 = 10 \bmod 11,$
$2^6 = 2 \times 10 = 9 \bmod 11, 2^7 = 2 \times 9 = 7 \bmod 11,$
$2^8 = 2 \times 7 = 3 \bmod 11, 2^9 = 2 \times 3 = 6 \bmod 11, 2^{10} = 2 \times 6 = 12 = 1 \bmod 11.$

NB : La dernière puissance était prévisible puisque le groupe $(\mathbb{Z}/11\mathbb{Z})^*$ est d'ordre 10. On voit que l'ensemble des puissances de 2 couvre l'ensemble des nombres non nuls modulo 11, CQFD.

2. — Alice a envoyé : $g^a \bmod n = 2^7 \bmod 11 = 7 \bmod 23$.
   — Alice a reçu : $3 (= g^b \bmod n = 2^8 \bmod 11)$.
   — Elle calcule le secret :$(g^b)^a \bmod n = 3^7 \bmod 11 = 9 \bmod 11$.
3. — Bob a envoyé : $g^b \bmod n = 2^8 \bmod 11 = 3 \bmod 23$.
   — Bob a reçu : $7 = g^a \bmod n = 2^7 \bmod 11$.
   — Il calcule le secret : $(g^a)^b \bmod n = 7^8 \bmod 11 = 9 \bmod 11$

NegoCle-03

Exercice 35. Diffie-Hellman sur courbes elliptiques

Soit la courbe elliptique définie sur le corps fini $\mathbb{F}_5$ par : $y^2 = x^3 + 1$. On utilise les notations et résultats de l'exercice CryMath-07 "Tutoriel courbe elliptique". On considère le protocole de Diffie-Hellman sur la courbe elliptique, avec $a = 2$ et $b = 4$ les valeurs aléatoires tirées par Alice et Bob.

1. Quel est le secret calculé par Alice ?
2. Vérifier que le secret calculé par Bob est le même.

1. — Alice a envoyé $a \cdot P = 2 \cdot P = (0; 1)$ à Bob.
   — Alice a reçu $(0; 4) = b \cdot P = 4 \cdot P = (0; 4)$ de Bob.
   — Alice calcule le secret $2 \cdot (0; 4) = (0; 1)$.
2. — Bob a envoyé $b \cdot P = 4 \cdot P = (0; 4)$ à Bob.
   — Bob a reçu $(0; 1) = 2 \cdot P$ d'Alice.
   — Bob calcule le secret $4 \cdot (0; 1) = (0; 1)$.

6 Hachage et intégrité

6.1 Fonctions de hachage

Integ-01

Exercice 36. Tuto hachage éponge

On définit une bijection f sur 32 bits,au niveau octet

Entrées : $X = (x_0, \ldots, x_3) \in \{0,1\}^{32}$

1. $(y_0, \ldots, y_3) \leftarrow (\text{SBox}(x_0), \ldots, \text{SBox}(x_3))$
2. $\begin{bmatrix} z_0 \\ \vdots \\ z_3 \end{bmatrix} \leftarrow [\text{MixColumns}] \begin{bmatrix} y_0 \\ \vdots \\ y_3 \end{bmatrix}$
3. **Retourner** $(z_0, \ldots, z_3)$

---

où les fonctions SBox et MixColumns sont celles de l'AES. On définit la fonction de hachage de type éponge avec :

— taille d'état interne 32 bits

— taille de bloc 8 bits

— capacité 24 bits

— bijection f ci-dessus

— taille de haché 8 bits

Soit M le message de 24 bits – donc 3 blocs – égal à `3243f6`. Calculer le haché de M.

```
This is toy Hash_sponge function
Input message: 32 43 f6   is 3 blocks
Hence running has 3 absorbing steps, and one squeezing step
At absorbing step 1
Starting state: 00 00 00 00     Block message: 32
Input f_sponge bijection: 32 00 00 00
This is toy f_sponge bijection
Input (32-bit): 32 00 00 00
After Sbox: 23 63 63 63
After MixColumns: Output: e3 23 23 a3
Output f_sponge bijection: e3 23 23 a3
At absorbing step 2
Starting state: e3 23 23 a3     Block message: 43
Input f_sponge bijection: a0 23 23 a3
This is toy f_sponge bijection
Input (32-bit): a0 23 23 a3
After Sbox: e0 26 26 0a
After MixColumns: Output: 9d cc 94 2f
Output f_sponge bijection: 9d cc 94 2f
At absorbing step 3
Starting state: 9d cc 94 2f     Block message: f6
Input f_sponge bijection: 6b cc 94 2f
This is toy f_sponge bijection
Input (32-bit): 6b cc 94 2f
After Sbox: 7f 4b 22 15
After MixColumns: Output: 14 9a 4f c2
Output f_sponge bijection: 14 9a 4f c2
Squeezing step. Hash value: 14
```

Integ-02

Exercice 37. Collision hachage par collision compression

On considère une fonction de hachage H définie par l'itération d'une fonction de compression $f : \{0,1\}^h \times \{0,1\}^n \to \{0,1\}^n$.

1. Montrer que si f est à sens unique, alors h est à sens unique. Plus précisément, si un adversaire est capable de trouver une pré-image pour h avec complexité C, alors il trouve une pré-image pour f avec complexité $C + \varepsilon$

2. Montrer que si f est à sens unique et sans collision, alors h est sans collision. Plus précisément, si un adversaire est capable de trouver une collision pour h avec complexité C, alors il trouve ou bien une pré-image ou bien une collision pour f avec complexité $C + \varepsilon$.

Notons, pour $x = x[1]\dots x[n]$, $H[0] = IV$, et pour $i = 1, \dots, n$, $H[i] = f(H_{i-1}, x[i])$.

1. Par contraposée. Supposons h n'est pas à sens unique, alors pour un certain y, on peut trouver $x = x[1]\dots x[n]$ tel que $h(x) = y$ avec complexité C. Comme $h(x) = f(H[n-1], x_n)$, $H[n-1], x_n$ est une pré image de y par f. Cette pré image est de plus facilement calculable à partir de x. D'où la complexité annoncée.

2. Supposons qu'on ait trouvé une collision avec complexité C pour h

$$h(x[1]\dots x[n]) = h(x'[1]\dots x'[n'])$$

avec $x \neq x'$, et par exemple $n \geq n'$. Cela s'écrit :

$$f(H[n-1], x[n]) = f(H'[n'-1], x'[n'])$$

Si $H[n-1] \neq H'[n'-1]$ ou $x[n] \neq x[n']$, c'est gagné car on a trouvé une collision sur f. Sinon, on a en particulier $H[n-1] = H'[n'-1]$ c'est-à-dire

$$f(H[n-2], x[n-2]) = f(H'[n'-2], x'[n'-1])$$

On peut ainsi "redescendre" tant que $x[n-k] = x'[n'-k]$ et $H[n-k-1] = H'[n'-k-1]$. À la k-ième étape, on a

$$f(H[n-k-1], x[n-k]) = f(H'[n'-k-1], x'[n'-k]).$$

Deux cas se présentent :

— pour tout k, on a $x[n-k] = x'[n'-k]$ et $H[n-k-1] = H'[n'-k-1]$. La "redescente" s'arrête lorsque $k = n'$. Dans ce cas, on a forcément $n > n'$, sinon on aurait $x[n-k] = x'[n-k]$ pour tout k et donc $x = x'$. En fait, c'est le cas où x' est un suffixe de x, , c'est-à-dire $x = x[1]\dots x[n-n']x[n-n'+1]\dots x[n] = x[1]\dots x[n-n']\dots x'[1]\dots x'[n']$, et où les hachés intermédiaires sont égaux, c'est-à-dire $H[n-k-1] = H'[n'-k-1]$. La dernière égalité obtenue dans le processus de "redescente" est alors

$$f(H[n-n'-1], x[n-n']) = H'[0],$$

qui donne une pré image de $H'[0] = IV$ par f.

— il existe $k < n'$ tel que $x[n-k] \neq x'[n'-k]$ ou $H[n-k-1] \neq H'[n'-k-1]$, et k_0 minimal tel que cela se produise. Alors, la "redescente" s'arrête à k_0 et on a

$$f(H[n-k_0-1], x[n-k_0]) = f(H'[n'-k_0-1], x'[n'-k_0]),$$

qui donne une collision sur f.

Récapitulons : on vient de montrer que si h a une collision, alors ou bien f a une collision ou bien f a une pré image calculable au point IV. De plus, la complexité nécessaire à exhiber la collision/la pré image sur f est négligeable une fois connue la collision sur h. C'est ce que nous voulions démontrer.

Integ-03

Exercice 38. Hachage concaténé et multicollision

On considère $H : x \mapsto H_1(x)||H_2(x)$ une fonction de hachage sur $2n$ bits, concaténation de deux fonctions de hachage H_1 et H_2 sur n bits, l'une et l'autre sans collision.

1. Quelle est la complexité d'une collision sur H par simple application du paradoxe des anniversaires ?

On suppose dorénavant que H_1 est une construction de type Merkle-Damgard : itération d'une fonction de compression f. Le but de ce qui suit est de montrer qu'alors on peut obtenir une collision avec une complexité $(n/2) \cdot 2^{n/2}$.

2. Avec quelle complexité trouve-t-on $M[1] \neq M'[1]$ tel que $H_1(M[1]) = H_1(M'[1])$?

3. En notant $y_1 = H_1(M[1]) = H_1(M'[1])$, avec quelle complexité trouve-t-on $M[2] \neq M'[2]$ tel que $f(y_1, M[2]) = f(y_1, M'[2])$?

4. Montrer que l'on peut construire une collection $\mathcal{M}$ de $2^{n/2}$ messages 2 à 2 distincts ayant tous la même image par H_1 avec une complexité de $(n/2) \cdot 2^{n/2}$ (on parle de *multicollision*).

5. Montrer que la construction de la collection $\mathcal{M}$ permet de "récupérer" une collision sur H_2.

6. En déduire qu'on peut trouver une collision sur H avec une complexité de $(n/2) \cdot 2^{n/2}$.

1. La sortie de H est de taille $2n$ bits, une collision par paradoxe des anniversaires s'obtient avec une complexité de 2^n.

2. La complexité pour trouver $M[1] \neq M'[1]$ tel que $H_1(M[1]) = H_1(M'[1])$ est $2^{n/2}$: c'est le paradoxe des anniversaires.

3. Comme la fonction de compression est supposée idéale, c'est la même complexité. Tout se passe comme si l'on avait modifié H_1 en changeant sa valeur initiale de IV à y_1, ce qui est sans incidence sur la collision demandée.

4. Par récurrence, pour $i = 1, 2, \ldots, n/2$, on peut trouver $M[i] \neq M'[i]$ tel que $y_i = f(y_{i-1}, M[i]) = f(y_{i-1}, M'[i])$ avec complexité totale $\frac{n}{2}2^{n/2}$ (chaque étape de la récurrence coûte $2^{n/2}$ et il y a $n/2$ étapes). Soit $\mathcal{M}$ la collection de messages :

$$\{x = (x[1], \ldots, x[n/2]);\ \forall i = 1, \ldots, n/2,\ x[i] = M[i] \text{ ou } M'[i]\}.$$

il est clair que l'on définit ainsi une collection de $2^{n/2}$ messages deux à deux distincts. L'hypothèse que $y_i = f(y_{i-1}, M[i]) = f(y_{i-1}, M'[i])$ à chaque étape assure que tous les messages de $\mathcal{M}$ ont pour haché y_n par H_1.

5. $\mathcal{M}$ est une collection de $2^{n/2}$ messages deux à deux distincts, et H_2 est une fonction aléatoire. Par le paradoxe des anniversaires, la probabilité d'une collision sur H_2 est non négligeable.

6. Considérons deux messages de $\mathcal{M}$ réalisant la collisison sur H_2 à la question 4. Par construction, ils réalisent aussi une collisison sur H_1. Donc ils réalisent une collisison sur H. On vient donc de réaliser une collisison sur H avec une complexité $\frac{n}{2}2^{n/2}$.

Integ-04

Exercice 39. Construction fonction de hachage par extension de domaine

Soit h_1 une fonction de hachage résistante aux collisions, de $\{0,1\}^{2m}$ dans $\{0,1\}^m$. On définit h_2 de

$\{0,1\}^{4m}$ dans $\{0,1\}^m$ par

$$h_2(x) = h_2(x_1|x_2) = h_1\left(h_1(x_1)|h_1(x_2)\right)$$

Montrer que h_2 est résistante aux collisions.

On procède par contraposée. Si h_2 n'est pas collision-résistante aux collisions, alors on peut, avec complexité $\ll O(2^{m/2})$ trouver $y \neq x$ tel que $h_2(y) = h_2(x)$. En notant $x = (x_1|x_2)$ et $y = (y_1|y_2)$, nous avons $h_2(y_1|y_2) = h_2(x_1|x_2)$ et finalement, par définition de h_2

$$h_1\left(h_1(x_1)|h_1(x_2)\right) = h_1\left(h_1(y_1)|h_1(y_2)\right) \quad .$$

Deux cas peuvent se présenter :

— ou bien $h_1\left(h_1(x_1)|h_1(x_2)\right) \neq h_1\left(h_1(y_1)|h_1(y_2)\right)$, ce qui donne immédiatement une collision sur h_1.

— ou bien $h_1(x_1)|h_1(x_2) = h_1(y_1)|h_1(y_2)$, ce qui implique :

$$\begin{cases} h_1(x_1) = & h_1(y_1) \\ h_1(x_2) = & h_1(y_2) \end{cases}$$

avec $x_1 \neq y_1$ ou $x_2 \neq y_2$ (puisque $x \neq y$), ce qui donne là encore une (voire deux) collision(s) sur h_1.

Dans les deux cas une collision sur h_2 implique une collision sur h_1 avec une sur-complexité negligeable, donc avec complexité globale $\ll O(2^{m/2})$. Ce qui prouve que h_1 n'est pas collision-résistante.

6.2 MAC

Integ-06

Exercice 40. Tuto HMAC

On considère la fonction de hachage H jouet de l'exercice Integ-01. Pour tout message M, on considère le mode HMAC avec une clé secrète K de 16 bits :

$$\text{HMAC\_}H_K(M) = H(K \oplus 0x5c5c||H(K \oplus 0x3636||M))$$

Soit M le message de 24 bits, donc 3 blocs, égal à `3243f6` et $K =$ `2b7e`. Calculer le HMAC de M.

```
This is HMAC
Secret key (16-bit): 2b 7e     Input message: 32 43 f6
Inner call to Hash function
     This is toy Hash_sponge function
     Input message: 1d 48 32 43 f6   is 5 blocks
     Hence running has 5 absorbing steps, and one squeezing step
     At absorbing step 1
     Starting state: 00 00 00 00     Block message: 1d
     Input f_sponge bijection: 1d 00 00 00
          This is toy f_sponge bijection
          Input (32-bit): 1d 00 00 00
          After Sbox: a4 63 63 63
          After MixColumns: Output: f6 a4 a4 31
     Output f_sponge bijection: f6 a4 a4 31
     At absorbing step 2
     Starting state: f6 a4 a4 31     Block message: 48
     Input f_sponge bijection: be a4 a4 31
          This is toy f_sponge bijection
          Input (32-bit): be a4 a4 31
```

```
          After Sbox: ae 49 49 c7
          After MixColumns: Output: 12 20 27 7c
     Output f_sponge bijection: 12 20 27 7c
     At absorbing step 3
     Starting state: 12 20 27 7c     Block message: 32
     Input f_sponge bijection: 20 20 27 7c
          This is toy f_sponge bijection
          Input (32-bit): 20 20 27 7c
          After Sbox: b7 b7 cc 10
          After MixColumns: Output: 6b 9d b3 99
     Output f_sponge bijection: 6b 9d b3 99
     At absorbing step 4
     Starting state: 6b 9d b3 99     Block message: 43
     Input f_sponge bijection: 28 9d b3 99
          This is toy f_sponge bijection
          Input (32-bit): 28 9d b3 99
          After Sbox: 34 5e 6d ee
          After MixColumns: Output: 09 d1 99 a8
     Output f_sponge bijection: 09 d1 99 a8
     At absorbing step 5
     Starting state: 09 d1 99 a8     Block message: f6
     Input f_sponge bijection: ff d1 99 a8
          This is toy f_sponge bijection
          Input (32-bit): ff d1 99 a8
          After Sbox: 16 3e ee c2
          After MixColumns: Output: 42 81 b2 75
     Output f_sponge bijection: 42 81 b2 75
     Squeezing step. Hash value: 42
Outer call to Hash function
     This is toy Hash_sponge function
     Input message: 77 22 42   is 3 blocks
     Hence running has 3 absorbing steps, and one squeezing step
     At absorbing step 1
     Starting state: 00 00 00 00     Block message: 77
     Input f_sponge bijection: 77 00 00 00
          This is toy f_sponge bijection
          Input (32-bit): 77 00 00 00
          After Sbox: f5 63 63 63
          After MixColumns: Output: 54 f5 f5 c2
          Output f_sponge bijection: 54 f5 f5 c2
     At absorbing step 2
     Starting state: 54 f5 f5 c2     Block message: 22
     Input f_sponge bijection: 76 f5 f5 c2
          This is toy f_sponge bijection
          Input (32-bit): 76 f5 f5 c2
          After Sbox: 38 e6 e6 25
          After MixColumns: Output: 82 fb 66 02
     Output f_sponge bijection: 82 fb 66 02
     At absorbing step 3
     Starting state: 82 fb 66 02     Block message: 42
     Input f_sponge bijection: c0 fb 66 02
          This is toy f_sponge bijection
          Input (32-bit): c0 fb 66 02
          After Sbox: ba 0f 33 77
          After MixColumns: Output: 3a 86 4a 07
     Output f_sponge bijection: 3a 86 4a 07
     Squeezing step. Hash value: 3a
HMAC value: 3a
```

Integ-05

Exercice 41. **EMAC mal utilisé**

On considère EMAC avec une primitive bloc E_K. On prend $K_1 = K_2 = K$, et on note n la taille du bloc.

1. Montrer que $\mathrm{EMAC}_{K,K}(M)$ est égal à $\mathrm{CBCMAC}_K(M')$ pour un certain message M' que l'on explicitera.

2. On note $M = M[1] \,||\, \ldots \,||\, M[m]$, $\text{EMAC}_{K,K}(M) = t$, et

$$M^* = M[1] \,||\, \ldots \,||\, M[m] \,||\, 0^n \,||\, M[1] \oplus t \,||\, M[2] \,||\, \ldots \,||\, M[m]$$

Montrer que $\text{EMAC}_{K,K}(M^*) = t$.

3. En déduire que EMAC utilisé avec une seule clé n'est pas sûr.

1. Notons $M = M[1] \,||\, M2 \,||\, \ldots \,||\, M[m]$. En écrivant les équations de EMAC, il est direct de vérifier que

$$\text{EMAC}_{K,K}(M[1] \,||\, M[2] \,||\, \ldots \,||\, M[m]) = \text{CBCMAC}_K(M[1] \,||\, M[2] \,||\, \ldots \,||\, M[m] \,||\, 0^n).$$

2. On doit calculer EMAC du message $M[1] \,||\, M[2] \,||\, \ldots \,||\, M[m] \,||\, 0^n \,||\, M[1] \oplus t \,||\, M[2] \,||\, \ldots \,||\, M[m]$. Jusqu'au bloc 0^n inclus, le calcul est identique à celui de la question 1, et donc $C[m+1] = t$. Puis,

$$C[m+2] = E_K(C[m+1] \oplus M^*[m+2]) = E_K(M[1]) = C[1]$$

et, par récurrence immédiate,

$$\begin{array}{lcllcll} C[m+3] & = & E_K(C[m+2] \oplus M[m+3]) & = & E_K(C[1] \oplus M[2]) & = & C[2] \\ \ldots & & & & & & \\ C[2m+1] & = & E_K(C[2m] \oplus M[2m+1]) & = & E_K(C[m-1] \oplus M[m]) & = & C[m] \\ t' & = & E_K(C[2m+1]) & = & E_K(C[m]) & = & t \end{array}$$

3. L'attaquant effectue une requête à l'oracle sur un message M. Si t est la réponse, l'attaquant sait que $t = \text{EMAC}_{K,K}(M^*)$ (avec les notations précédentes), ce qui est une contrefaçon puisque l'on a évidemment $M^* \neq M$.

Integ-08

Exercice 42. Attaque préfixe-MAC pour des messages de taille variable

Soit H une fonction de hachage de type Merkle-Damgaard, f la fonction de compression sous-jacente, K une clé secrète et M un message. On rappelle que

$$\text{PrefixMAC\_}H_K(M) = H(K||M)$$

1. Soit x un bloc quelconque. Montrer que

$$\text{PrefixMAC\_}H_K(M||x) = f(\text{PrefixMAC\_}H_K(M), x)$$

2. En déduire une attaque par contrefaçon "presque" universelle avec un appel à l'oracle de MAC : plus précisément, motrer que l'on peut contrefaire le MAC de n'importe quel message non vide.

1. L'égalité est une conséquence directe de l'application de la procédure de Merkle-Damgaard. Attention, il est crucial que la sortie de la fonction de hachage soit égal à la totalité de la sortie de la fonction de compression. En cas de troncature, l'attaque telle quelle n'est pas applicable.
2. Soit M un message quelconque non vide. On l'écrit $M = (M^*||x)$ avec M^* potentiellement vide et x un bloc. L'attaque consiste à :

1. Requérir l'oracle de MAC sur le message M^*. Soit T^* la réponse de l'oracle.

2. Calculer $y := f(T^*, x)$
3. Retourner (M, y)

D'après la question 1, y est effectivement le MAC valide de M.

7 Signatures

7.1 Signature RSA

Sign-01

Exercice 43. Signature RSA tutorial

1. Quelle est la grande différence entre un chiffrement et une signature ?
2. On pose $p = 3$, $q = 11$, calculer n et $\Phi(n)$.
3. Peut-on choisir $e = 5$?
4. On pose $e = 7$, calculer d.
5. Le haché du document d'Alice est $T = 2$, Bob reçoit $S = 1$, qu'en pensez vous ?

1. Le rôle des clés est "inversé" :
   — en chiffrement, Bob chiffre avec la clé publique d'Alice et Alice déchiffre avec sa clé privée ;
   — en signature, Alice signe avec sa clé privée et Bob vérifie avec la clé publique d'Alice.
2. $n = 3 \cdot 11 = 33$ et $\Phi(n) = 2 \cdot 10 = 20$.
3. Non car $PGCD(\Phi(n), e) = PGCD(20, 5) = 5 \neq 1$.
4. $d = 3$ car $7 \cdot 3 = 21 = 1 \bmod 20$
5. On a : $T^3 = 2^3 \bmod 33 = 8 \neq 1 = S$ et
   $S^7 = 1^7 \bmod 33 = 1 \neq 2 = T$.
   Ce n'est pas Alice qui a signé le document reçu par Bob. Soit c'est un autre signataire, soit c'était Alice et le document a été modifié.

Sign-02

Exercice 44. Vérifications de signatures RSA

Deux documents accompagnés d'une signature électronique produite à l'aide d'une clé RSA, dont la partie publique est $n = 1\,833,\ e = 3$.

— Haché message1 = 12, signature = 363
— Haché message2 = 13, signature = 227

Les signatures de ces deux documents sont-elles valides ? ?

Il suffit d'appliquer l'algorithme de vérification de signature RSA :

— Message 1 : $363^3 = 12 \bmod 1\,833$ donc la signature est valide.
— Message 2 : $227^3 = 710 \neq 12 \bmod 1\,833$ donc la signature est invalide.

Sign-09

Exercice 45. Contrefaçon universelle pour la signature RSA native

On considère la signature RSA avec clé publique (n, e) et clé privée (p, q, d). On suppose que le schéma de signature ne réalise que la partie arithmétique à savoir :

— signature d'un message $m \in \{0, \ldots, p-2\} \mapsto m^d \bmod n$

— vérification de (m, σ) : la signature est valide $\Leftrightarrow \sigma^e = m \bmod n$

1. Montrer que pour tout message m, $\mathrm{Sig}(2^e m) = 2\mathrm{Sig}(m)$
2. En déduire qu'il est possible de contrefaire la signature de tout message m^* avec une requête à l'oracle de signature.

1. C'est une conséquence facile de la propriété multiplicative du RSA natif :

$$\mathrm{Sig}(m) = (2^e m)^d = 2^{ed} m^d = 2m^d \bmod n = 2\mathrm{Sig}(m)$$

2. On demande à l'oracle de signature de retourner la signature σ de $m = 2^e m^*$. Alors $\sigma := 2^{-1}\sigma$ est la signature valide de m^*

Sign-03

Exercice 46. Sécurité de H-RSA et propriétés de H

On considère le schéma de signature RSA avec une fonction de hachage H. Prouver les propriétés suivantes

1. Si H n'est pas préimage résistante, alors une contrefaçon existentielle est possible sans requête à l'oracle de signature. (Indication : choisir d'abord la valeur de la signature et déduire ce que doit être la valeur de $H(m)$)

2. Si H n'est pas seconde préimage résistante, alors une contrefaçon universelle à message choisi est possible avec une requête à l'oracle de signature. (Indication : construire la requête de sorte que la réponse soit précisément la signature à contrefaire)

3. Si H n'est pas collision résistante, alors une contrefaçon existentielle à message choisi est possible avec une requête à l'oracle de signature.

1. z est une signature valide de m si et seulement si $z^e = H(m)$. L'adversaire choisit z, calcule z^e, puis lui trouve un antécédent m par H, ce qui est facile puisque H n'est pas préimage résistante. Et c'est fini.

2. Soit m le message dont l'adversaire souhaite obtenir la signature. Comme H n'est pas seconde préimage résistante, l'adversaire trouve aisément $m' \neq m$ tel que $H(m) = H(m')$. Ainsi les signatures de m et de m' sont égales. Alors, en requérant à l'oracle la signature z de m', l'adversaire exhibe (m, z) comme un couple valide. Comme cette procédure est valable pour tout m, il s'agit bien d'une contrefaçon universelle.

3. H n'étant pas collision résistante, l'adversaire trouve aisément $m_1 \neq m_2$ tels que $H(m_1) = H(m_2)$. La suite est comme précédemment : en requérant à l'oracle la signature z de m_1, l'adversaire (m_2, z) comme un couple valide. Comme l'adversaire ne peut pas cibler ce que sont les messages m_1 et m_2 qui collisionnent, il s'agit bien d'une forge existentielle.

7.2 Signature Elgamal

Sign-05

Exercice 47. Signature El Gamal

Soient $p = 17$ et $g = 5$.

1. La clé privée d'Alice est $K_s = 2$. Calculez sa clé publique.
2. Calculer la signature du message M dont le haché vaut $\mathcal{H}(M) = 15$ avec $k = 3$ dans la position d'Alice.
3. Vérifier la signature de ce message dans la position de Bob.

1. La clé publique K_p vaut :
$$K_p = g^{K_s} \bmod p = 5^2 \bmod 17 = 8 \quad .$$
2. On a :
$$K = g^k \bmod p = 5^3 \bmod 17 = 6$$
et
$$k^{-1} \bmod p - 1 = 3^{-1} \bmod 16 = 11$$
car $3 \times 11 = 33 = 1 \bmod 16$. Il suit :
$$\begin{aligned} \sigma &= (\mathcal{H}(M) - K_s \cdot K) \cdot k^{-1} \bmod p - 1 \\ \sigma &= (15 - 2 \cdot 6) \cdot 11 \bmod 16 \\ \sigma &= 1 \quad . \end{aligned}$$
Le signé est $(M, 6, 1)$.
3. Bob vérifie que :
   - $1 \leq K = 6 \leq 16$;
   - $1 \leq \sigma = 1 \leq 15$;
   - $K_p^K \cdot K^\sigma = g^{\mathcal{H}(M)} \bmod p$:
$$\left\{ \begin{array}{lcll} K_p^K \cdot K^\sigma \bmod p & = & 8^6 \cdot 6^1 \bmod 17 & = & 7 \bmod 17 \quad . \\ g^{\mathcal{H}(M)} \bmod p & = & 5^{15} \bmod 17 & = & 7 \bmod 17 \quad . \end{array} \right.$$
Par conséquent, la signature est valide.

Sign-06

Exercice 48. Signature et vérification avec EC Elgamal

Soit la courbe elliptique définie sur le corps fini $\mathbb{F}_5$ par : $y^2 = x^3 + 2x + 1$. On pose $P = (0; 1)$ un point de la courbe.

1. Donner une encadrement du nombre de points de la courbe
2. Calculer les multiples de P, et vérifier qu'ils sont sur la courbe. En déduire que la courbe est le groupe cyclique engendré par P.
3. Soit $d = 2$ la clé privée d'Alice. Calculer sa clé publique.
4. Signer comme Alice le message M dont le haché vaut $\mathcal{H}(M) = 4$ avec $r = 3$.
5. Vérifier comme Bob la signature de la question précédente.

1. Le nombre de points de la courbe est compris entre $5+1-2\sqrt{5}=1,52\ldots$ et $5+1-2\sqrt{5}=10,47\ldots$, donc en fait entre 2 et 10 (c'est un entier !!!)
2. On calcule les multiples de P
   - $P=(0;1)$, $y^2=1 \mod 5$ et $x^3+2x+1=1 \mod 5$, donc P vérifie l'équation de la courbe.
   - $2\cdot P=(1;3)$, $y^2=4$ et $x^3+2x+1=4$, donc $2\cdot P$ vérifie l'équation de la courbe.
   - $3\cdot P=P+2\cdot P=(3;3)$, $y^2=4$ et $x^3+2x+1=4$, donc $3\cdot P$ vérifie l'équation de la courbe.
   - $4\cdot P=2\cdot 2\cdot P=(3;2)$, $y^2=4$ et $x^3+2x+1=4$, donc $4\cdot P$ vérifie l'équation de la courbe.
   - $5\cdot P=P+4\cdot P=(1;2)$, $y^2=4$ et $x^3+2x+1=4$, donc $5\cdot P$ vérifie l'équation de la courbe.
   - $6\cdot P=2\cdot P+4\cdot P=(0;4)$, $y^2=1$ et $x^3+2x+1=1$, donc $6\cdot P$ vérifie l'équation de la courbe.
   - $7\cdot P=O_\infty$ car $6\cdot P=(0;4)=(0;-1)=-P$

   Ceci termine le calcul des multiples de P : l'ordre de P est donc égal à 7, et c'est par définition le cardinal du sous-groupe engendré par P. Ainsi, 7 divise le cardinal de la courbe. L'encadrement de la question 1 montre que ce cardinal est en fait égal à 7. Ainsi la courbe est égale au sous-groupe engendré par P. Les points de la courbe sont les suivants : $(0;1)(1;3)(3;3)(3;2)(1;2)(0;4)$ et O_∞.
3. La clé publique est $Q=d\cdot P=2\cdot P=(1;3)$.
4. Le point $r\cdot P=3\cdot P=(3;3)$.

$$\begin{aligned}\sigma &= (\mathcal{H}(M)+x_{rP}\cdot d)\cdot r^{-1} \mod |E|\\ \sigma &= (4+3\cdot 2)\cdot 3^{-1} \mod 7\\ \sigma &= 1 \quad .\end{aligned}$$

   Le signé est $(M, x_{rP}, \sigma)=(M,3,1)$
5. Bob reçoit $(M,3,1)$ et il possède la clé publique $Q=(1;3)$.

$$\begin{aligned}V &= \mathcal{H}(M)\cdot\sigma^{-1}\cdot P+x_{rP}\cdot\sigma^{-1}\cdot Q \quad ;\\ V &= 4\cdot 1\cdot P+3\cdot 1\cdot K_p \quad ;\\ V &= 4\cdot P+3\cdot K_p \quad ;\\ V &= (3;2)+(0;4) \quad ;\\ V &= (3;3) \quad .\end{aligned}$$

   On vérifie que $V\neq O_E$ et $x_V=3=x_K$.

Sign-07

Exercice 49. Contrefaçon existentielle pour la signature El Gamal naïf

On considère la signature El Gamal avec clé publique (p,g,h) et clé privée k. On suppose que le schéma de signature ne réalise que la partie arithmétique à savoir :

— signature d'un message $m\in\{0,\ldots,p-2\}$:

---

1. Choisir au hasard $r\in\{0,\ldots,p-2\}$ inversible modulo $p-1$
2. calculer $(s_1,s_2)=(g^r \bmod p, (m-ks_1)r^{-1} \bmod p-1)\in\{0,\ldots,p-1\}\times\{0,\ldots,p-2\}$

— vérification de $(m,(s_1,s_2))$: la signature est valide $\Leftrightarrow h^{s_1}s_1^{s_2}=g^m \bmod p$

Étant donné $a\in\{0,\ldots,p-2\}, b\in\{1,\ldots,p-2\}$, on pose $s_1=g^a h^b \bmod p$ et $s_2=-s_1 b^{-1} \bmod p-1$. Montrer que (s_1,s_2) est la signature valide d'un message m que l'on explicitera.

On a

$$\begin{aligned} h^{s_1} s_1^{s_2} &= h^{s_1}(g^a h^b)^{s_2} \\ &= h^{s_1} g^{as_2} h^{b\cdot(-s_1 b^{-1})} \\ &= h^{s_1} g^{as_2} h^{-s_1} \\ &= g^{as_2} \end{aligned}$$

ce qui prouve que (s_1, s_2) est la signature valide du message as_2

Sign-08

Exercice 50. Contrefaçon universelle pour la signature El Gamal naïf

On considère la signature El Gamal avec clé publique (p, g, h) et clé privée k. On suppose que $p \equiv 1 \bmod 4$ et que $g = 2$.

1. Montrer que $h^{\frac{p-1}{2}} = \pm 1 \bmod p$, $2^{\frac{p-1}{2}} = -1 \bmod p$.

2. Montrer que le logarithme discret de $\frac{p-1}{2}$ en base 2 et égal à $\frac{p-3}{2}$.

Soient

$$z = \log_{-1}(h^{\frac{p-1}{2}}), \text{ c'est-à-dire } z = \begin{cases} 0 & \text{if } h^{\frac{p-1}{2}} = 1 \bmod p \\ 1 & \text{if } h^{\frac{p-1}{2}} = -1 \bmod p \end{cases}$$

$s_1 = \frac{p-1}{2}$, et $s_2 \in \{0, \ldots, p-2\}$.

3. Montrer que $h^{s_1} s_1^{s_2} = 2^{z(\frac{p-1}{2}) + s_2(\frac{p-3}{2})}$.

4. Montrer que $\frac{p-3}{2}$ est inversible modulo $p-1$. (Indication : ils sont premiers entre eux - pourquoi ?)

On choisit $m \in \{0, \ldots, p-2\}$, et on pose

$$s_2 = \left(m - z\frac{p-1}{2}\right) \cdot \left(\frac{p-3}{2}\right)^{-1} \bmod p-1.$$

5. Montrer que (s_1, s_2) est la signature valide de m.

1. Pour tout $x \in (\mathbb{Z}/p\mathbb{Z})^\times$, on a $x^{p-1} = 1$ (petit théorème de Fermat). Ainsi, $x^{\frac{p-1}{2}} = \pm 1$ puisque son carré est égal à 1 et que -1 et 1 sont les deux racines carrées de 1 dans $\mathbb{Z}/p\mathbb{Z}$. Cela prouve déjà $h^{\frac{p-1}{2}} = \pm 1$.

Pour $g = 2$, on a aussi $2^{\frac{p-1}{2}} = \pm 1$. Mais $2^{\frac{p-1}{2}}$ ne peut pas être égal à 1, sinon l'ordre de 2 diviserait $\frac{p-1}{2}$, donc ne serait pas égal à $p-1$, et 2 ne serait pas générateur. On déduit $2^{\frac{p-1}{2}} = -1$.

2. On a

$$2^{\frac{p-3}{2}} = 2^{\frac{p-1}{2}} 2^{-1} = \frac{-1}{2} = \frac{p-1}{2} \bmod$$

CQFD

3. On a

$$\begin{aligned} h^{s_1} &= h^{\frac{p-1}{2}} \\ &= (-1)^z \text{ par définition de } z \\ &= (2^{\frac{p-1}{2}})^z \text{ d'après Q1} \end{aligned}$$

et

$$\begin{aligned} s_1^{s_2} &= \left(\frac{p-1}{2}\right)^{s_2} \\ &= \left(2^{\frac{p-3}{2}}\right)^{s_2} \text{ d'après Q2} \\ &= 2^{s_2 \frac{p-3}{2}} \end{aligned}$$

ce qui prouve le résultat demandé.

4. Les entiers $\frac{p-3}{2}$ et $\frac{p-1}{2}$ sont consécutifs donc premiers entre eux. En particulier, $\frac{p-3}{2}$ et $p-1$ n'ont aucun facteur impair en commun. Par ailleurs 2 ne divise pas $\frac{p-3}{2}$, puisque $p \equiv 1 \bmod 4$. Il suit que $\frac{-3}{2}$ et $p-1$ sont premiers entre eux, d'où le résultat.

5. D'après Q3,

$$h^{s_1} s_1^{s_2} = 2^{z(\frac{p-1}{2})+s_2(\frac{p-3}{2})}.$$

On le réécrit

$$h^{s_1} s_1^{s_2} = 2^{z(\frac{p-1}{2})+(m-z(\frac{p-1}{2}))} = 2^m$$

ce qui prouve le résultat.

8 Mathématiques de la cryptographie

8.1 Complexité

CryMath-01

Exercice 51. **La force brute**

Le facteur de travail d'un algorithme est le nombre d'instructions élémentaires nécessaires à son exécution. La puissance d'une machine est le nombre d'instructions qu'elle exécute par unité de temps. Nous allons approcher la puissance d'un PC actuel à environ 2000 Mips (millions d'instructions par seconde). Le facteur de travail d'un algorithme optimisé pour tester une clé de 128 bits de l'algorithme AES est d'environ 1200 instructions élémentaires. On dispose d'un couple clair/chiffré (P, C) connu et on désire retrouver la clé utilisée par force brute, c'est-à-dire en testant toutes les clés les unes après les autres. Une clé est constituée d'un mot de 128 bits : On suppose que toutes les clés sont équiprobables.

1. En combien de temps une machine de 2000 Mips teste-t-elle une clé ?
2. Combien y a-t-il de clés possibles ?
3. Quel est le nombre moyen de clés à tester avant de trouver la bonne ?
4. À quel temps moyen de calcul cela correspond-il si on suppose qu'un seul PC effectue la recherche ? Si les 1 milliard de PC de l'Internet sont mobilisés à cette tâche ?

1. $$t = \frac{\text{facteur de travail}}{\text{puissance}} = \frac{1200}{2000} = 0.6\mu s$$
2. Nombre de clés possibles $= 2^{128}$.
3. On considère les clés possibles comme étant les entiers de 0 à $2^{128} - 1$. La clé secrète est notée k. On note $n = 2^{128}$. On a deux scénarios d'attaque par force brute possible.
   — Essayer tous les entiers les uns après les autres. La probabilité, pour un entier i donné, d'avoir $k = i$ (et donc d'avoir exactement $i + 1$ tirages à effectuer si on part de 0), est égale à $\frac{1}{n}$.
   L'espérance du nombre d'essais est donc :
   $$\sum_{i=0}^{n-1}(i+1)\cdot\frac{1}{n} = \frac{n\cdot(n+1)}{2\cdot n} \approx \frac{n}{2} \quad .$$
   — Effectuer un grand nombre de tirages aléatoires parmi 2^{128} et s'arrêter quand on a trouvé la clé. On a une loi géométrique de paramètre $p = 1/n$.
   Chaque tirage a une probabilité de succès $\frac{1}{n} = p$.
   La probabilité qu'on trouve la clé au bout de i tirages est : $(1-p)^{i-1}\cdot p$.
   On a donc l'espérance du nombre de tirages nécessaires :
   $$\sum_{i=1}^{+\infty} i\cdot(1-p)^{i-1}\cdot p = f'(1-p) = \frac{p}{(1-(1-p))^2} = \frac{1}{p} = n \quad ;$$
   avec $f(x) = \frac{1}{1-x}$.
4. En abusant des approximations :
   — $10^3 = 1000 \approx 2^{10}$,
   — 1 jour $= 2^{16}$ secondes,
   — 1 an $= 2^9$ jour $= 2^{25}$ secondes.

   On calcule d'abord le nombre d'instructions calculées en un an à la fréquence de 2000 Mips.
   $$\text{2000 Mips.années} \approx 2000\cdot 2^{20}\cdot 2^9\cdot 2^{16} \approx 2^{25}\ \text{instructions} \approx 2^{11+20+9+16} \approx 2^{56}.$$
   Le nombre d'instructions à effectuer pour trouver la clé est : $1200\cdot 2^{127} \approx 2^{138}$.
   Soit un temps de : $\approx 2^{138-56} \approx 2^{81}$ années ($\approx 2\cdot(2^{10})8 \approx 2\cdot 10^{24}$).
   Les un milliard ($\approx 2^{30}$) de PC d'Internet permettent de gagner un facteur 2^{30}, ou 10^9.
   Soit quelque chose comme $2\cdot 10^{15}$ années, soit un petit million de fois l'âge de l'univers.

8.2 Paradoxe des anniversaires

CryMath-08

Exercice 52. Le paradoxe des anniversaires

On considère N boules numérotées de 1 à N. On effectue un tirage de q boules, successif avec remise. On note $\mathsf{PrCol}(N, q)$ la probabilité dite de **collision**, c'est-à-dire qu'au moins une boule soit apparue plusieurs fois.

1. Que vaut $\mathsf{PrCol}(N, q)$ si $q > N$?

Dans la suite, on suppose $q \leq N$. On remarque que l'on peut modéliser le résultat d'un tirage de q boules successivement avec remise par un q-uplet ordonné d'éléments de $\{1, \ldots, N\}$.

2. Par un raisonnement de dénombrement, montrer que

$$\mathsf{PrCol}(N,q) = 1 - \left(1 - \frac{1}{N}\right) \dots \left(1 - \frac{q-1}{N}\right) = 1 - \prod_{i=0}^{q-1} \left(1 - \frac{i}{N}\right).$$

3. En utilisant (sans preuve) l'inégalité $\forall 0 \leq x \leq 1, \quad 0 \leq 1 - x \leq e^{-x}$, vérifier que $0 \leq 1 - i/N \leq e^{-i/N}$. En déduire que

$$\mathsf{PrCol}(N,q) \geq 1 - \exp\left(-\frac{q(q-1)}{2N}\right) \geq 1 - \exp\left(-\frac{(q-1)^2}{2N}\right)$$

Soit $p \in [0,1]$.

4. Montrer que, pour que $\mathsf{PrCol}(N,q) \geq 1 - \varepsilon$, il suffit que $q \geq 1 + \sqrt{2N \ln\left(\frac{1}{\varepsilon}\right)}$.
5. Application numérique pour $N \gg q \gg 1$ et $\varepsilon = 1/2$.

On va pour terminer donner une majoration de $\mathsf{PrCol}(N,q)$. On considère, pour $i = 1, \dots, q$, l'événement

$C_i =$ {le i-ième tirage fait apparaître une boule déjà apparue lors de l'un des $i-1$ premiers tirages}

6. Montrer que $\Pr(C_i) \leq \dfrac{i-1}{N}$
7. En observant que l'événement "collision" est la réunion (non disjointe) des événements $C_1, \dots, C_q$, en déduire que

$$\mathsf{PrCol}(N,q) \leq \frac{q(q-1)}{2N}.$$

Remarque. Pour $\sqrt{N} \geq q \gg 1$, l'encadrement de $\mathsf{PrCol}(N,q)$ se réduit en pratique à l'approximation numérique de l'ordre de grandeur $\mathsf{PrCol}(N,q) \approx O\left(\frac{q^2}{2N}\right)$.

1. Si $q > N$, il est évident que $\mathsf{PrCol}(N,q) = 1$.
2. Le nombre de cas favorables à la non-collision est le nombre de q-uplets dont les éléments sont 2 à 2 distincts. Le nombre de cas total est le nombre de q-uplets tout court. On déduit

$$\Pr(\text{non-collision}) = \frac{N(N-1)\dots(N-(q-1))}{N^q}$$

ce qui conduit à la formule demandée par simplification.

3. Comme on a supposé $q \leq N$, on a $0 \leq i/N \leq 1$, et donc $0 \leq 1 - i/N \leq e^{-i/N}$ par le rappel. En remplaçant dans l'expression obtenue à la question 2, on obtient

$$\mathsf{PrCol}(N,q) \geq 1 - \prod_{i=0}^{q-1} \exp\left(-\frac{i}{N}\right) = 1 - \exp\left(-\sum_{i=0}^{q-1} \frac{i}{N}\right) = 1 - \exp\left(-\frac{q(q-1)}{2N}\right).$$

La deuxième inégalité est facile : $q(q-1) \geq (q-1)^2$ et $x \mapsto 1 - exp(-x)$ est croissante.

4. D'après la borne inférieure de la question précédente $\mathsf{PrCol}(N,q) \geq 1 - \varepsilon$ dès que $1 - \exp\left(-\frac{(q-1)^2}{2N}\right) \geq 1 - \varepsilon$. Un calcul élémentaire donne le résultat escompté.
5. On trouve le "fameux" $q \geq 1,17\sqrt{N} \Rightarrow \mathsf{PrCol}(N,q) \geq 1/2$.
6. Au cours des $i-1$ premiers tirages, le nombre n_{i-1} de boules apparues est $\leq i-1$, car il a pu y avoir des répétitions. Ainsi,

$$\Pr(C_i) = \frac{n_{i-1}}{N} \leq \frac{i-1}{N}.$$

7. La probabilité d'une réunion quelconque d'événements est inférieure à la somme des probabilités individuelles. Il s'ensuit immédiatement

$$\mathsf{PrCol}(N,q) \leq \sum_{i=1}^{q} \Pr(C_i) \leq \sum_{i=1}^{q} \frac{i-1}{N} = \frac{q(q-1)}{2N}.$$

CryMath-03

Exercice 53. **Une variante des anniversaires**

On considère N boules numérotées de 1 à N. On effectue un tirage de q boules, successif avec remise. On définit un ensemble cible $A \subset \{1,\ldots,N\}$ de cardinal a. On étudie la probabilité de la "rencontre", $\mathsf{PrMatch}(N,q,a)$ qu'au moins un tirage soit dans A.

1. Montrer que $\mathsf{PrMatch}(N,q,a) = 1-(1-a/N)^q$ (on suppose les tirages indépendants).

On définit, pour $i = 1,\ldots,q$, l'événement $C_i = \{$le i-ième tirage appartient à la cible A $\}$.

2. Justifier que l'événement "rencontre" est la réunion des événements C_i.
3. En déduire que $\mathsf{PrMatch}(N,q,a) \leq qa/N$.
4. En utilisant $\forall 0 \leq x \leq 1,\ \ 0 \leq 1-x \leq e^{-x}$, montrer que

$$\mathsf{PrMatch}(N,q,a) \geq 1-e^{-qa/N}$$

5. Donner la valeur de q (en fonction de a et N) pour que $\mathsf{PrMatch}(N,q,a) \geq 1/2$.
6. En utilisant l'approximation $1-x \approx e^{-x}$ pour $x \ll 1$, déduire des Q3 et 4 que, si $qa/N \ll 1$, on a $\mathsf{PrMatch}(N,q,a) \approx qa/N$.
7. Variante numérique : montrer que si $qa \approx N$, $\mathsf{PrMatch}(N,q,a) \geq \approx 0,63$.

On s'intéresse maintenant au nombre X de tirages parmi les N qui rencontrent l'ensemble A.

8. Quelle est la loi de probabilité suivie par X ?
9. En déduire le nombre moyen $\mathsf{NrMatches}(N,q,a)$ de rencontres entre les tirages et A

1. Chaque tirage a une probabilité $1-a/N$ de "rater" l'ensemble cible A. Comme tous les tirages sont indépendants et que la probabilité d'une intersection d'événements indépendants est le produit des probabilités, la probabilité qu'il n'y ait pas de rencontre est $(1-a/N)^q$. Le résultat suit par simple passage à l'événement contraire.
2. C'est une simple conséquence de la définition. À noter que la réunion des événements C_i n'est pas exclusive, il peut y avoir plusieurs tirages "tombant" dans l'ensemble cible A.
3. La probabilité d'une réunion d'événements est inférieure à la somme des probabilités individuelles. D'où

$$\mathsf{PrMatch}(N,q,a) \leq \sum_{i=1}^{q} a/N = qa/N.$$

4. Comme $0 \leq a/N \leq 1$, on a $0 \leq 1-a/N \leq e^{-a/N}$. On reporte dans la formule du 1 :

$$\mathsf{PrMatch}(N,q,a) = 1-(1-a/N)^q \geq 1-(e^{-a/N})^q$$

   et c'est gagné.
5. Pour que $\mathsf{PrMatch}(N,q,a) \geq 1/2$, il suffit que $e^{-qa/N} \leq 1/2$, c'est-à-dire $qa/N > \ln 2$, soit finalement

$$q \geq \frac{N\ln(2)}{a}.$$

6. Si $qa/N \ll 1$ les Q3 et 4 montrent que

$$\approx qa/N \leq \mathsf{PrMatch}(N,q,a) \leq qa/N$$

ce qui fournit l'approximation demandée.

7. Si $qa = N$, $\mathsf{PrMatch}(N,q,a) \geq 1 - 1/e \approx 0,63\ldots$

En pratique (et en crypto), on retient l'ordre de grandeur et l'on formule :

$\mathsf{PrMatch}(N,q,a)$ *est non négligeable dès que* aq *est de l'ordre de* N*.*

Par exemple, c'est le cas lorsque $a = q = \sqrt{N}$, mais aussi (situation "déséquilibrée") $a = N^{1/3},\ q = N^{2/3}$.

8. Chaque tirage est une expérience de Bernoulli de paramètre a/N, et il y a q tirages indépendants. La variable X suit par définition une loi binomiale $\mathcal{B}(q; a/N)$.
9. C'est l'espérance de X, égale (loi binomiale) à qa/N

On retiendra que la quantité qa/N est interprétée de manières différentes (bien que liées) selon sa valeur :

— si elle est $\ll 1$, elle est une bonne approximation numérique de la probabilité de rencontre ;

— si elle est ≥ 1, il est plus pertinent de la voir comme le nombre moyen de rencontres.

Voici enfin un résultat dont la formulation diffère quelque peu. La technique de démonstration est la même.

> **Proposition** *Soient E de cardinal N, et A et B deux parties aléatoires de cardinaux a et b. Alors le cardinal moyen de l'intersection $A \cap B$ est de l'ordre de ab/N.*

8.3 Fonctions booléennes

CryMath-06

Exercice 54. Tuto Fonctions booléennes

Pour alléger la graphie, l'addition binaire - i.e. *modulo 2 - est exceptionnellement notée "+" et non "⊕"*

Soit f la fonction booléenne de 3 variables définie par

$$f(x_1,x_2,x_3) = x_1 + x_2 + x_2x_3 + x_1x_2x_3$$

1. Écrire la table de valeurs de la fonction f.

Dans la suite de l'exercice on montre comment retrouver la forme algébrique à partir de la table de valeurs. On prend la table de valeurs obtenue à la question ci-dessus

```
[x1,x2,x3]     | f(x)
---------------|-----
[ 0, 0, 0 ]    |  0
[ 1, 0, 0 ]    |  1
[ 0, 1, 0 ]    |  1
[ 1, 1, 0 ]    |  0
[ 0, 0, 1 ]    |  0
[ 1, 0, 1 ]    |  1
[ 0, 1, 1 ]    |  0
[ 1, 1, 1 ]    |  0
```

2. Soit (a_1,a_2,a_3) un 3-uplet fixé de bits. Montrer que l'expression

$$(x_1 + a_1 + 1)(x_2 + a_2 + 1)(x_3 + a_3 + 1)$$

vaut 1 si $(x_1,x_2,x_3) = (a_1,a_2,a_3)$ et 0 sinon. On la notera $\mathbf{1}_{(a_1,a_2,a_3)}(x_1,x_2,x_3)$

3. À titre d'exemple, calculer $\mathbf{1}_{(0,0,0)}(x_1,x_2,x_3)$.

4. Montrer que l'expression

$$f(0,0,0)\mathbf{1}_{(0,0,0)}(x_1,x_2,x_3) + f(1,0,0)\mathbf{1}_{(1,0,0)}(x_1,x_2,x_3) + \dots f(1,1,1)\mathbf{1}_{(1,1,1)}(x_1,x_2,x_3)$$

coïncide avec f en tout point de $\{0,1\}^3$.

5. Développer réduire l'expression de la question précédente et retrouver l'expression algébrique de f du début de l'énoncé

1. Il suffit de substituer les variables x_1, x_2, x_3 dans l'expression algébrique de g par tous les triplets de bits. Un simple calcul donne

$$\begin{array}{rcl} f(0,0,0) & = & 0+0+0\cdot 0+0\cdot 0\cdot 0 = 0 \\ f(1,0,0) & = & 1+0+0\cdot 0+1\cdot 0\cdot 0 = 1 \\ f(0,1,0) & = & 0+1+1\cdot 0+0\cdot 1\cdot 0 = 1 \\ f(1,1,0) & = & 1+1+1\cdot 0+1\cdot 1\cdot 0 = 0 \\ f(0,0,1) & = & 0+0+0\cdot 1+0\cdot 0\cdot 1 = 0 \\ f(1,0,1) & = & 1+0+0\cdot 1+1\cdot 0\cdot 1 = 1 \\ f(0,1,1) & = & 0+1+1\cdot 1+0\cdot 1\cdot 1 = 0 \\ f(1,1,1) & = & 1+1+1\cdot 1+1\cdot 1\cdot 1 = 0 \end{array}$$

2. Le produit $(x_1+a_1+1)(x_2+a_2+1)(x_3+a_3+1)$ est composé de facteurs valant 0 ou 1. Ainsi, il est nul sauf lorsque TOUS les facteurs sont égaux à 1. Cela se produit si et seulement si $x_1 = a_1, x_2 = a_2, x_3 = a_3$.
3. On a

$$\begin{array}{rcl} \mathbf{1}_{(0,0,0)}(x_1,x_2,x_3) & = & (1+x_1)(1+x_2)(1+x_3) \\ & = & 1+x_1+x_2+x_3+x_1x_2+x_1x_3+x_2x_3+x_1x_2x_3 \end{array}$$

4. Fixons un point $(a_1,a_2,a_3) \in \{0,1\}^3$ et substituons les variables x_1, x_2, x_3 par ces valeurs. D'après ce qui précède, seul survit le terme $f(a_1,a_2,a_3)\mathbf{1}_{(a_1,a_2,a_3)}(a_1,a_2,a_3) = f(a_1,a_2,a_3)\times 1 = f(a_1,a_2,a_3)$, ce que nous voulions démontrer.
5. On calcule l'expression algébrique de la question 4. Seuls survivent les termes pour lesquels $f(\cdot,\cdot,\cdot)$ est égal à 1. Ainsi cela donne

$$\begin{array}{rl} & \mathbf{1}_{(1,0,0)}(x_1,x_2,x_3) + \mathbf{1}_{(0,1,0)}(x_1,x_2,x_3) + \mathbf{1}_{(1,0,1)}(x_1,x_2,x_3) \\ = & x_1(1+x_2)(1+x_3) + (1+x_1)x_2(1+x_3) + x_1(1+x_2)x_3 \end{array}$$

Un développement classique (mais nécessitant soin et concentration...) redonne l'expression du début d'énoncé.

8.4 Générateur et groupe cyclique

Alg-groupes-01

Exercice 55. Caractérisations d'un générateur dans un groupe fini

1. Soit G un groupe fini de cardinal n dont on suppose connue la factorisation. Montrer que se valent :
   (1) g est un générateur de G
   (2) pour tout $i < n$, on a $g^i \neq 1$
   (3) pour tout d diviseur strict de n, on a $g^d \neq 1$
   (4) pour tout facteur premier p de n, on a $g^{n/p} \neq 1$

   On illustre ce résultat dans le cas du groupe multiplicatif $\mathbb{F}^*_{56701}$. On admet que 56701 est premier et que

$$56700 = 2^2 \cdot 3^4 \cdot 5^2 \cdot 7^1$$

On utilisera préférentiellement le critère (4) qui est le plus rapide à mettre en œuvre.

2. Montrer que 2 est générateur de $\mathbb{F}^*_{56701}$.
3. Montrer que 3 n'est pas générateur de $\mathbb{F}^*_{56701}$.

1. On montre le "cercle d'implications" (1) $\Rightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (4) $\Rightarrow$ (1).

   (1) $\Rightarrow$ (2)
   Supposons par l'absurde qu'il existe $i < n$ tel que $g^i = 1$, et prenons i minimal (en fait i est l'ordre de g). Alors le sous-groupe engendré par g est $\{1, g, g^2, \ldots, g^{i-1}\}$: il est de cardinal i et ne peut être égal à G, contradiction.

   (2) $\Rightarrow$ (3)
   C'est évident puisque tout diviseur d strict de n est en particulier $< n$.

   (3) $\Rightarrow$ (4)
   C'est aussi évident puisque, pour tout p facteur premier de n, n/p est un diviseur strict de n

   (4) $\Rightarrow$ (1)
   On va montrer que l'ordre $\text{ord}(g)$ de g est égal à n. Par une propriété classique de l'ordre, pour tout entier i on a
   $$g^i = 1 \Longleftrightarrow \text{ord}(g) \text{ divise } i$$
   Or $\text{ord}(g)$ divise n (toujours vrai dans un groupe) et par l'hypothèse (4), $\text{ord}(g)$ ne divise aucun n/p, pour p diviseur premier de n. Il suit que $\text{ord}(g) = n$, ce que nous voulions.
2. Les diviseurs de 56700 du critère 4 sont :
   $$56700/2 = 28350,\ 56700/3 = 18900,\ 56700/5 = 11340,\ 56700/7 = 8100$$
   Maintenant, comme
   $$2^{28350} = 56700,\ 2^{18900} = 36160,\ 2^{11340} = 10173,\ 2^{8100} = 1279$$
   le critère (4) implique que 2 est générateur.
3. De même,
   $$3^{28350} = 1,\ 3^{18900} = 36160,\ 3^{11340} = 6733,\ 3^{8100} = 11737$$
   et donc 3 n'est pas générateur.

Arith-01

Exercice 56. **Autour du petit théorème de Fermat**

Le (petit) théorème de Fermat stipule que si n est un nombre premier alors pour tout entier x premier avec n, $x^{n-1} = 1 \bmod n$.

1. Énoncer la réciproque du petit théorème de Fermat.
2. Montrer que $4^{14} = 1 \bmod 15$. Cette égalité invalide-t-elle la réciproque du théorème de Fermat ?

Nombres de Carmichael

3. Vérifier que 561 est composé.
4. Montrer que pour tout x premier avec 561, on a $x^{560} = 1 \bmod 561$. On pourra utiliser un programme ou suivre le cheminement des questions suivantes

5. Montrer que 561 vérifie la propriété :
   (Carm) *n est sans facteur carré et pour tout nombre premier p divisant n, on a $p-1$ divise $n-1$*
6. Soit x premier avec 561. Montrer que
$$x^2 = 1 \bmod 3,\ x^{10} = 1 \bmod 11,\ x^{16} = 1 \bmod 17$$
   On pourra utiliser le petit théorème de Fermat.
7. En déduire que
$$x^{560} = 1 \bmod 3,\ x^{560} = 1 \bmod 11,\ x^{560} = 1 \bmod 17$$
   On pourra utiliser la propriété (Carm) démontrée plus haut
8. En déduire finalement $x^{560} = 1 \bmod 561$.
9. La réciproque du petit théorème de Fermat est-elle vraie ?

1. La réciproque du petit théorème de Fermat s'énonce comme suit :
$$\left(\forall\ x \text{ premier avec } n,\ \ x^{n-1} = 1 \bmod n\right) \Rightarrow (n \text{ est premier})$$
2. Calculons les puissances successives de 4 mod 15. Comme $4^2 = 16 = 1 \bmod 15$, c'est élémentaire même sans calculatrice. On a $4^i = 1$ si i est pair, $= 4$ si i est impair. Donc $4^{15-1} = 1 \bmod 15$, et 4 et 15 sont premiers entre eux. Pourtant 15 n'est pas premier.
3. Ici, nous n'avons pas invalidé cette réciproque car nous avons seulement montré qu'il existe x ($= 4$ en l'occurrence) premier à $n(= 15)$ tels que $x^{n-1} = 1 \bmod n$. Mais on pourrait trouver des x premiers à 15 tels que $x^{14} \bmod 15 \neq 1$: $x = 2$ par exemple.
4. $561 = 3 \times 11 \times 17$
5. 561 est sans facteur carré comme le montre sa décomposition en produit de facteurs premiers. Par ailleurs, ses facteurs premiers sont 3, 11 et 17, et on a bien 2, 10 et 16 divisent 560, ce qui prouve la deuxième partie de la propriété.
6. Puisque x est premier avec 561, il est premier avec 3 et avec 11 et avec 17. On peut donc appliquer trois fois le petit théorème de Fermat pour récupérer les trois égalités souhaitées.
7. Comme 2 divise 560, il est très facile et archi classique de voir que $(x^2 = 1 \bmod 3) \Rightarrow (x^{560} = 1 \bmod 3)$. Le raisonnement est analogue pour les deux autres égalités.
8. Plutot que d'invoquer le puissant théorème des restes chinois, on va résoudre la question d'une manière méthodique mais plus élémentaire. Posons $y := x^{560} - 1$. D'après la question précédentes, nous savons que 3, 11 et 17 divisent y. Il suit, par définition du PPCM, que y est multiple de $\text{PPCM}(3, 11, 17)$. Enfin $\text{PPCM}(3, 11, 17) = 3 \times 11 \times 17$ (3, 11 et 17 sont premiers entre eux deux à deux !). Nous venons de montrer que y est multiple de 561, c'est-à-dire $y = 0 \bmod 561$. Mais par définition de y c'est exactement ce que demandait la question.
9. Non, elle est fausse, puisque 561 n'est pas premier et vérifie
$$\forall\ x \text{ premier avec } 561,\ \ x^{560} = 1 \bmod 561$$
   Il existe d'autres nombres tels que 561, qui invalident la réciproque du petit théorème de Fermat. Ces nombres sont appelés *nombres de Carmichael*.

8.5 Algorithmes arithmétiques élémentaires

Exercice 57. Racines carrées modulaires.
Trouvez les quatre racines carrées de 1 modulo 77, puis celles de 53 modulo 77.

Comme $77 = 7 \times 11$, il suffit de déterminer les racines carrées de 1 modulo 7 et modulo 11, puis d'appliquer le théorème des restes chinois. Comme 7 (resp. 11) est premier, 1 a deux racines carrées modulo 7 (resp. modulo 11) : 1 et -1. Le quatre racines carrées de 1 mod 77 sont données par l'unique solution de chacun des systèmes suivants :

$$(1) \quad \begin{cases} x = 1 \bmod 7 \\ x = 1 \bmod 11 \end{cases}$$

$$(2) \quad \begin{cases} x = -1 \bmod 7 \\ x = -1 \bmod 11 \end{cases}$$

$$(3) \quad \begin{cases} x = 1 \bmod 7 \\ x = -1 \bmod 11 \end{cases}$$

$$(4) \quad \begin{cases} x = -1 \bmod 7 \\ x = 1 \bmod 11 \end{cases}$$

(1) et (2) donnent évidemment 1 et -1 mod 77. (3) et (4) donnent deux autres solutions, opposées l'une de l'autre. Il suffit donc de résoudre (3).

La méthode est de chercher sous la forme $x = 7u + 11v$, en déterminant séparément u et v.

$$7u + 11v = 1 \bmod 7 \Rightarrow 11v = 1 \bmod 7 \Rightarrow v = 11^{-1} \bmod 7 = 4^{-1} \bmod 7 = 2 \bmod 7$$

et de même

$$7u + 11v = -1 \bmod 11 \Rightarrow 7u = -1 \bmod 11 \Rightarrow u = -7^{-1} \bmod 11 = 3 \bmod 11$$

En regroupant, il vient $x = 7 \times 3 + 2 \times 11 = 43 \bmod 77$ Les quatre racines carrées de 1 mod 77 sont donc : $1, 43, 34(= -43), 76(= -1)$

Pour trouver les racines de 53, on procède de manière analogue. On détermine d'abord les racines carrées de 53 modulo 7 et modulo 11.

— $53 = 4 = 2^2 \bmod 7$. Les racines carrées de 53 mod 7 sont donc 2 et -2.

— $53 = 9 = 3^2 \bmod 11$ Les racines carrées de 53 mod 11 sont donc 3 et -3.

Les quatre racines carrées de 53 modulo 77 sont les solutions des quatre systèmes

$$(1) \quad \begin{cases} x = 2 \bmod 7 \\ x = 3 \bmod 11 \end{cases}$$

$$(2) \quad \begin{cases} x = -2 \bmod 7 \\ x = -3 \bmod 11 \end{cases}$$

$$(3) \quad \begin{cases} x = 2 \bmod 7 \\ x = -3 \bmod 11 \end{cases}$$

$$(4) \quad \begin{cases} x = -2 \bmod 7 \\ x = 3 \bmod 11 \end{cases}$$

On cherche, pour chaque système, la solution sous la forme $x = 7u + 11v$. Pour (1),

$$7u + 11v = 2 \bmod 7 \Rightarrow 11v = 2 \bmod 7 \Rightarrow v = 2 \times 11^{-1} = 2 \times 4^{-1} = 2 \times 2 = 4 \bmod 7$$

et

$$7u + 11v = 3 \bmod 11 \Rightarrow 7u = 3 \bmod 11 \Rightarrow u = 3 \times 7^{-1} = 3 \times (-3) = 2 \bmod 11$$

Et finalement $x = 7 \times 2 + 11 \times 4 = 58 \bmod 77$ La solution de (2) est l'opposé, soit $-58 = 19 \bmod 77$ Pour (3),

$$7u + 11v = 2 \bmod 7 \Rightarrow 11v = 2 \bmod 7 \Rightarrow v = 2 \times 11^{-1} = 2 \times 4^{-1} = 2 \times 2 = 4 \bmod 7$$

et

$$7u + 11v = -3 \bmod 11 \Rightarrow 7u = -3 \bmod 11 \Rightarrow u = -3 \times 7^{-1} \bmod 11 = -3 \times (-3) = 9 \bmod 11$$

Et finalement $x = 7.9 + 11.4 = 30 \bmod 77$ Pour (4), c'est l'opposé de la solution de (3), soit $-30 = 47 \bmod 77$. Au bilan les quatre racines de 53 modulo 77 sont 19, 30, 47 et 58.

AlgoArith-02

Exercice 58. **Exponentiation modulaire.**
Calculer sans calculatrice $2^7 \bmod 55$, puis $18^{23} \bmod 55$.

Le premier calcul est facile : $2^7 = 128 = 18 \bmod 55$.

Le deuxième calcul est un peu plus difficile sans calculatrice, mais on va ruser un peu. Comme 2 est inversible modulo 55, on peut appliquer le théorème d'Euler, $2^{\phi(55)} = 1 \bmod 55$. Mais $\phi(55) = \phi(5 \times 11) = (5-1)(11-1) = 40$. Donc $2^{40} = 1 \bmod 55$. Par ailleurs on vient de voir que $18 = 2^7 \bmod 55$. Ainsi

$$18^{23} = 2^{7 \times 23} = 2^{161} = 2^{4 \times 40 + 1} = (2^{40})^4 \times 2 = 2 \bmod 55$$

AlgoArith-03

Exercice 59. **Exponentition rapide**
Soit $(G, \cdot)$ un groupe multiplicatif, x un élément de G et n un entier naturel. On veut calculer le plus efficacement possible l'élément x^n. La complexité sera mesurée en nombre de multiplications dans le groupe. Par exemple le calcul de $x^3 = x \cdot x \cdot x$ requiert en évidence 2 multiplications

1. Vérifier que la méthode naïve requiert $n - 1$ multiplications.

On va maintenant introduire une amélioration significative de complexité par la méthode dite "binaire". Nous présentons cette méthode à l'aide de quelques exemples de plus en plus élaborés.

2. Montrer que l'on peut calculer x^{32} à l'aide de 5 multiplications "particulières" : des élévations au carré (on mettra à profit le fait que $32 = 2^5$).

3. En déduire comment calculer x^{33} avec 6 multiplications (5 élévations au carré et une multiplication)

4. Montrer que l'on peut calculer x^{35} avec 7 multiplications (6 élévations au carré et deux multiplications).

On explique maintenant la méthode dans le cas général pour l'exposant n. On note

$$n = (n_r n_{r-1} \dots n_2 n_1 n_0)_2 = (n_r 2^r + n_{r-1} 2^{r-1} + \cdots n_2 2^2 + n_1 2 + n_0$$

la représentation binaire (dite aussi "en base 2") de n.

5. Montrer que

$$x^n = \left(x^{2^r}\right)^{n_r} \cdot \left(x^{2^{r-1}}\right)^{n_{r-1}} \cdots \left(x^{2^2}\right)^{n_2} \cdot \left(x^2\right)^{n_1} \cdot (x)^{n_0} \quad (1)$$

6. En déduire que le calcul de x^n est réalisable avec r élévations au carré et un nombre de multiplication que l'on précisera.

7. En s'inspirant de l'écriture "à la Hörner"

$$n = 2(2(2(\dots 2(2n_r + n_{r-1}) + \dots) + n_2) + n_1) + n_0$$

proposer un algorithme de calcul de x^n.

8. Quelle est la complexité de l'algorithme ?

9. Refaire les calculs de l'exercice "Exponentiation modulaire" avec la présente méthode

1. C'est une conséquence directe de la définition de x^n.

2. On réalise l'algorithme

---

1. $y_1 \leftarrow x^2$
2. $y_2 \leftarrow y_1^2$ // $= x^{2^2}$
3. $y_3 \leftarrow y_2^2$ // $= x^{2^3}$
4. $y_4 \leftarrow y_3^2$ // $= x^{2^4}$
5. $y_5 \leftarrow y_4^2$ // $= x^{2^5}$

---

3. On constate que $x^{33} = x^{32+1} = x^{32} \cdot x$. Ainsi, une solution est :

---

1. $y_1 \leftarrow x^2$
2. $y_2 \leftarrow y_1^2$ // $= x^{2^2}$
3. $y_3 \leftarrow y_2^2$ // $= x^{2^3}$
4. $y_4 \leftarrow y_3^2$ // $= x^{2^4}$
5. $y_5 \leftarrow y_4^2$ // $= x^{2^5}$
6. $y_6 \leftarrow y_5 \cdot x$ // $= x^{33}$

---

4. On écrit $35 = 32 + 2 + 1$, de sorte que $x^{35} = x^{32} \cdot x^2 \cdot x$. Ce calcul nécessite deux multiplications, et au préalable le calcul de x^{32} et x^2. En reprenant les questions précédentes, le calcul de x^{32} nécessite 5 élévations au carré, et le calcul de x^2 ne coute...rien du tout car il a été effectué au cours du calcul de x^{32}.

5. C'est une conséquence directe des règles sur les puissances.

6. On calcule $x^2, x^{2^2}, \ldots, x^{2^{r-1}}, x^{2^r}$ par r élévations au carré, exactement comme à la question 2. Attention il faut penser à stocker les valeurs ainsi obtenues. La suite du calcul n'est faite que de multiplications, et à ce titre on peut faire deux remarques :

— contrairement aux apparences, il n'y a plus d'élévations à une puissance. En effet, dans l'égalité (1), les n_i valent) ou 1, car ce sont les chiffres binaires de n ;

— par ailleurs, chaque facteur du produit correspondant à un n_i nul vaut 1 donc ne nécessite aucun traitement. Le nombre de facteurs "effectif" dans le produit est donc égal au nombre de n_i non nuls (autrement égaux à 1) : c'est le *poids binaire* de l'entier n, parfois noté $w_2(n)$.

Le nombre de multiplications à réaliser est clairement égal à $w_2 - 1$ et les facteurs intervenant dans le produit ont tous été (pré-)calculés lors des élévations au carré.

7. Voici l'algorithme en question. Il est essentiellement identique à la méthode de la question 6, mais l'astuce de l'écriture "à la Hörner" permet d'éviter le stockage mémoire des calculs intermédiaires.

Entrées : x, n

1. Écrire la représentation binaire de n : $(n_r n_{r-1} \dots n_2 n_1 n_0)_2$
2. $y \leftarrow 1$
3. **Pour** i décroissant de r à 0
4. **Si** $n_i = 1$
5. $y \leftarrow y \cdot x$
6. fin si
7. $y \leftarrow y^2$
8. fin pour
9. **Retourner** y

---

8. L'algorithme requiert r élévations au carré (qui sont des multiplications particulières), et $w_2 - 1 \geq r$ multiplications. Par ailleurs, le nombre de bits d'un entier n est égal à $\lfloor \log_2(n) \rfloor + 1$, qui est $\equiv \log_2(n)$ quand $n \to \oplus\infty$. On retient donc synthétiquement que la complexité de l'algorithme est de $O(\log_2(n))$ multiplications.

9. Calcul de 2^7 mod 55 et de 18^{23} mod 55

```
Computing 2^7 mod 55 by binary exponentiation
Binary sequence of exponent is [ 1, 1, 1 ] (leftmost significant bit)
Step 1: bit of sequence is 1
  Start: y = 1
  After squaring : y = 1^2 mod 55 = 1
  After multiplying by 2: y = 1*2 mod 55 = 2
Step 2: bit of sequence is 1
  Start: y = 2
  After squaring : y = 2^2 mod 55 = 4
  After multiplying by 2: y = 4*2 mod 55 = 8
Step 3: bit of sequence is 1
  Start: y = 8
  After squaring : y = 8^2 mod 55 = 9
  After multiplying by 2: y = 9*2 mod 55 = 18
Result is: 18

Computing 18^23 mod 55 by binary exponentiation
Binary sequence of exponent is [ 1, 0, 1, 1, 1 ] (leftmost significant bit)
Step 1: bit of sequence is 1
  Start: y = 1
  After squaring : y = 1^2 mod 55 = 1
  After multiplying by 18: y = 1*18 mod 55 = 18
Step 2: bit of sequence is 0
  Start: y = 18
  After squaring : y = 18^2 mod 55 = 49
Step 3: bit of sequence is 1
  Start: y = 49
  After squaring : y = 49^2 mod 55 = 36
  After multiplying by 18: y = 36*18 mod 55 = 43
Step 4: bit of sequence is 1
  Start: y = 43
  After squaring : y = 43^2 mod 55 = 34
  After multiplying by 18: y = 34*18 mod 55 = 7
Step 5: bit of sequence is 1
```

```
  Start: y = 7
  After squaring : y = 7^2 mod 55 = 49
  After multiplying by 18: y = 49*18 mod 55 = 2
Result is: 2
```

8.6 Factorisation des entiers

IntFact-01

Exercice 60. Algorithme de factorisation de Fermat

Soit $n = pq$ un entier produit de deux nombres premiers impairs $p < q$ (cible standard : un entier RSA). On supposera si nécessaire pour éviter des subtilités inutiles que n est "grand" devant $\sqrt{n}$ et que p et q sont de l'ordre de $\sqrt{n}$. La méthode de factorisation de Fermat présentée ici réside sur le fait que l'on peut retrouver p et q dès lors que l'on a trouvé un "bon" couple d'entiers (x, y) tels que $n = x^2 - y^2$.

1. Vérifier que si (x, y) est un couple d'entiers tels que $n = x^2 - y^2$, on a $x > \sqrt{n}$ et $x^2 - n$ est un carré parfait.
2. Montrer plus précisément qu'il existe exactement deux couples (x, y) tels que $n = x^2 - y^2$. On pourra raisonner par condition nécessaire en observant que pour un tel couple, $(x+y)(x-y)$ est une factorisation de n en produit de deux entiers.
3. Montrer que parmi ces deux couples, l'un des deux est pertinent au sens où il permet de retrouver p et q, et pas l'autre.

On considère l'algorithme suivant :

---

1 : pour $x = \lceil\sqrt{n}\rceil, \ldots, \dfrac{n+1}{2}$
2 : si $x^2 - n$ est un carré parfait
3 : retourner $(x, \sqrt{x^2 - n}$
4 : fin si
5 : fin pour

---

4. Montrer qu'il retourne exactement les deux couples (x, y) tels que $n = x^2 - y^2$.
5. Montrer que le premier des deux couples retourné est celui qui est pertinent.
6. Quel est le nombre d'itérations réalisées par l'algorithme jusqu'à l'atteinte de la solution pertinente ?

Cette méthode est un algorithme de factorisation dont la complexité est $O(\sqrt{q} - \sqrt{p})$. Elle implique que dans un nombre RSA $n = pq$, s'il est bien de choisir p et q de même taille, il ne faut pas les choisir "trop proches".

1. Ce sont deux conséquences élémentaires de l'identité $n = x^2 - y^2$. D'une part, $x = \sqrt{n + y^2} \geq \sqrt{n}$, et comme $\sqrt{n}$ n'est pas un entier (immédiat vu la forme de n), on a $x > \sqrt{n}$. D'autre part $x^2 - n$ est un carre parfait puisqu'il est égal à y^2 !
2. On raisonne par analyse-synthèse. Si un tel couple (x, y) existe, on a $n = (x + y)(x - y)$. Or les deux factorisations possibles de n sont $n = p \times q$ et $n = n \times 1$. On a donc

$$\left\{\begin{array}{rcl} x+y &=& q \\ x-y &=& p \end{array}\right. \text{ ou } \left\{\begin{array}{rcl} x+y &=& n \\ x-y &=& 1 \end{array}\right.$$

c'est-à-dire (calcul facile) :

$$(x, y) = \left(\frac{q+p}{2}, \frac{q-p}{2}\right) \text{ ou } \left(\frac{n+1}{2}, \frac{n-1}{2}\right)$$

3. Il est clair que trouver le premier couple permet de récupérer p et q comme $x-y$ et $x+y$, c'est-à-dire conduit à la factorisation non triviale de n. L'autre couple retrouvera les entiers 1 et n et donc conduit à la factorisation triviale de n.
4. Au cours de l'exécution de l'algorithme, la variable x va prendre entre autre les deux valeurs $\frac{q+p}{2}$ et $\frac{n+1}{2}$ (cette dernière est même la dernière itération). Et donc l'algorithme va passer par les
5.
6.
7.

1. On raisonne par condition nécessaire. Si de tels a et b existent on a

$$n = pq = (a+b)(a-b)$$

Mais comme un entier admet une unique factorisation cela implique que $a+b$ et $a-b$ sont égaux à p et q. Puisque $p > q$, on a $a+b = p$ et $a-b = q$, et en conclusion, nécessairement

$$a = \frac{p+q}{2},\ b = \frac{p-q}{2}$$

Réciproquement, ces valeurs de a et b satisfont l'identité $n = a^2 - b^2$. De plus $b \neq 0$ sinon n serait un carré, et donc $a^2 > n$ ce qui prouve $a > \sqrt{n}$. Enfin comme $a - b = q > 1$, la condition $b < a - 1$ est aussi satisfaite.

Remarque : ce raisonnement montre au passage l'unicité de a et b

2. Pour $t = \lceil\sqrt{n}\rceil, \lceil\sqrt{n}\rceil + 1, \lceil\sqrt{n}\rceil + 2, \ldots$ on calcule $t^2 - n$ jusqu'à ce soit le carré d'un nombre entier. Lorsque c'est le cas, nécessairement $t = a$ et $\sqrt{t^2-n} = b$. La factorisation de n est alors trouvée.

3. $\sqrt{23\,360\,947\,609} = 152\,842,8\ldots$. D'où les calculs :

t	$t^2 - n$	$\sqrt{t^2-n}$
152 843	35 040	$187,19\ldots$
152 844	340 727	$583,72\ldots$
152 845	646 416	804

Donc $23\,360\,947\,609 = 152\,845^2 - 804^2 = (152\,845 + 804)(152\,845 - 804) = 153\,649 \times 152\,041$.

4. La complexité est essentiellement égale au nombre d'étapes de la boucle "for", donc de l'ordre de

$$a - \sqrt{n} = \frac{p+q}{2} - \sqrt{pq} = \frac{1}{2}(\sqrt{p} - \sqrt{q})^2$$

5. Si p et q sont "trop proches", cette quantité peut être en dessous du seuil atteignable même si p et q sont assez grands.

IntFact-02

Exercice 61. **Algorithme $p-1$ de Pollard.**

Le but est de trouver un facteur non trivial de $n = 19\,048\,567$. On prend $B = 19$ (borne de lissité) et $a = 3$ une base première avec n.

1. Vérifier que $\text{pgcd}(a, n) = 1$.

2. Déterminer, pour chaque nombre premier ≤ 19, sa plus grande puissance qui soit $\leq n$.

Soit Q le ppcm de toutes ces puissances, et p un - hypothétique - facteur premier de n tel que $p-1$ soit

19-friable, c'est-à-dire dont les facteurs premiers sont tous inférieurs à 19. (À noter que dans cette signification "friable" est un synonyme de "lisse")

3. Montrer que $p-1$ divise Q.

4. En déduire que p divise $a^Q - 1$ (on pourra utiliser le petit théorème de Fermat).

5. En déduire que $\text{pgcd}(a^Q - 1, n) (= \text{pgcd}((a^Q - 1) \bmod n, n))$ est différent de 1.

6. On admet le calcul intermédiaire $a^Q \bmod n = 554\,506$. En déduire numériquement un facteur non trivial de n.

1. Le calcul du pgcd par l'algorithme d'Euclide est direct. Ici on peut faire encore plus rapide : 19 048 567 n'est pas multiple de 3 (critère de CM2 avec la somme des chiffres) et comme 3 est premier, le pgcd vaut forcément 1.

2. Les puissances demandées sont $2^{24}, 3^{15}, 5^{10}, 7^8, 11^6, 13^6, 17^5, 19^5$.

3. $p-1$ est supposé 19-friable, donc s'écrit $2^{e_2}3^{e_3}\dots 19^{e_{19}}$. Par ailleurs, $p-1 < p|n$ donc $2^{e_2} < n$ ce qui implique $e_2 \leq 24$. De même, $e_3 \leq 15, \dots, e_{19} \leq 5$, et au bilan $p-1 = 2^{e_2}3^{e_3}\dots 19^{e_{19}}$ divise $Q = 2^{24}3^{15}5^{10}7^8 11^6 13^6 17^5 19^5$.

4. D'après le petit théorème de Fermat, on a $a^{p-1} = 1 \bmod p$. Comme $p-1$ divise Q, on en déduit $a^Q = 1 \bmod p$, c'est-à-dire ce qui est demandé.

5. p divise n par hypothèse et p divise $a^Q - 1$ d'après la question 4. Donc $\text{pgcd}(a^Q - 1, n)$ est divisible par p, en particulier il est $\neq 1$.

NB : À cette étape, on n'a peut-être pas encore un facteur non trivial de n : en effet le pgcd est certes $\neq 1$ mais peut très bien être égal à n. Aller plus loin et trouver un facteur de n nécessite donc d'avoir la "chance" que le pgcd soit $< n$. Ne pas avoir cette "chance" signifie que l'hypothèse "$p-1$ est 19-friable" était fausse. Il faut alors recommencer l'algorithme avec une borne de friabilité plus grande (c'est-à-dire moins contraignante)

6. On calcule $\text{pgcd}(554\,505, 19\,048\,567) = 5\,281$. Ce dernier est un facteur non trivial de 19 048 567.

IntFact-03

Exercice 62. **Un critère de factorisation par congruence de carrés et PGCD**

Soit n un entier RSA, cad un entier composé de deux facteurs premiers inconnus. Démontrer que :

Si deux entiers x et y vérifient $x^2 = y^2 \bmod n$ (une telle relation s'appelle une *congruence quadratique*) et $x \neq \pm y \bmod n$, alors $\text{pgcd}(x-y, n)$ et/ou $\text{pgcd}(x+y, n)$ fournit un facteur non trivial de n.

Par hypothèse, $x^2 = y^2 \bmod n \leftrightarrow (x+y)(x-y) = 0 \bmod n$ et ni $x+y$ ni $x-y$ ne sont nuls modulo n. $x+y$ ni $x-y$ sont donc des diviseurs de 0 dans $\mathbb{Z}/n\mathbb{Z}$. Ils sont multiples de l'un facteur premier de n et pas de l'autre. Le calcul des deux pgcd indiqués fournit donc l'un de ces facteurs premiers (et même les deux).

Remarques.

1. si $x = \pm y \bmod n$, les deux pgcd retournés sont 1 et n, c qui fournit la factorisation triviale $n = 1 \times n$
2. ce résultat s'applique, mais pas systématiquement, à des nombres n composés de plus de facteurs premiers. De plus, quand elle fonctionne, le facteur non trivial de n trouvé n'est pas forcément premier.
3. autrement dit, dans tous les cas, la méthode PEUT retourner la factorisation triviale $n = 1 \times n$. Il ne faut surtout pas en conclure pour autant que n est premier.

IntFact-05

Exercice 63. Factorisation par combinaison de congruences

Soit n un entier RSA, cad un entier composé de deux facteurs premiers inconnus. Le but est de la factoriser par l'obtention d'une congruence quadratique $x^2 = y^2 \bmod n$.

On considère des entiers $x_1, \ldots, x_N$ et on calcule $b_i = x_i^2 \bmod n$ pour $i = 1, \ldots, N$.

1. Démontrer que si l'un des b_i (disons b_1 quitte à renuméroter) est un carré parfait alors on a récupéré une congruence quadratique.

On suppose qu'aucun des b_i n'est un carré parfait, mais qu'il existe $m \leq N$ indices $1 \leq i_1 < i_2 < \cdots < i_m \leq N$ tels que $b_{i_1} b_{i_2} \ldots b_{i_m}$ soit un carré parfait

2. Montrer qu'en combinant les identités $b_i = x_i^2 \bmod n$ (comment précisément ?), on obtient une congruence quadratique

On suppose que les b_i se factorisent tous selon une même base de nombres premiers $\mathcal{B} = (p_1, p_2, \ldots, p_s)$. On a donc :

$$\begin{cases} b_1 &= p_1^{e_{1,1}} p_2^{e_{1,2}} \cdots p_s^{e_{1,s}} \\ b_2 &= p_1^{e_{2,1}} p_2^{e_{2,2}} \cdots p_s^{e_{2,s}} \\ \cdots & \\ b_N &= p_1^{e_{N,1}} p_2^{e_{N,2}} \cdots p_s^{e_{N,s}} \end{cases}$$

Pour $i = 1, \ldots, N$, on note :

— $\mathbf{e}_i = (e_{1,1}, e_{1,2}, \ldots, e_{1,s})$ le vecteur d'exposants de b_i selon la base $\mathcal{B}$.

— $\bar{\mathbf{e}}_i = (\bar{e}_{1,1}, \bar{e}_{1,2}, \ldots, \bar{e}_{1,s})$ le vecteur d'exposants modulo 2, c'est-à-dire $\mathbf{e}_i \bmod 2 = (e_{1,1} \bmod 2, e_{1,2} \bmod 2, \ldots, e_{1,s} \bmod 2)$.

On fixe $m \leq N$ indices $1 \leq i_1 < i_2 < \cdots < i_m \leq N$.

3. Montrer que se valent
   (1) $b_{i_1} b_{i_2} \ldots b_{i_m}$ est un carré parfait
   (2) $\mathbf{e}_1 + \mathbf{e}_2 + \cdots + \mathbf{e}_m$ est un vecteur à composantes paires
   (3) $\bar{\mathbf{e}}_i \oplus \bar{\mathbf{e}}_i \oplus \ldots \oplus \bar{\mathbf{e}}_i = 0$ dans $\mathbb{F}_2^N$.

Cette dernière équivalence est à la base de l'algorithme de factorisation du crible quadratique. Le point crucial est que la complexité pour trouver une combinaison des b_i se ramène à la recherche d'une combinaison linéaire nulle de vecteurs dans $\mathbb{F}_2^N$, ce qui la rend beaucoup plus efficace que ne le laisserait penser l'approche naïve : $O(N^3)$ contre $O(2^N)$ (NB : une application numérique montre que $N^3/2^N \leq 10^{-3}$ (resp. $\leq 10^{-9}$) pour $N \geq 24$ (resp. ≥ 47).

1. Il suffit de noter $y_1^2 = b^1$, et on a la congruence quadratique $x_1^2 = y_1^2$.
2. On multiplie entre elles les identités $x_i^2 = b_i$ pour $i = i_1, \ldots, i_m$. Comme la multiplication est compatible du modulo on récupère
$$x_{i_1}^2 x_{i_2}^2 \ldots x_{i_m}^2 = b_{i_1} b_{i_2} \ldots b_{i_m} \bmod n$$
Le membre de gauche est égal à $(x_{i_1} x_{i_2} \ldots x_{i_m})^2$ donc est un carré, disons x^2. Le membre de droite est un carré par hypothèse, disons y^2. Ainsi on a bien $x^2 = y^2 \bmod n$, congruence quadratique comme on le voulait.
3. Le fait de multiplier des identités $b_i = p_1^{e_{i,1}} p_2^{e_{i,2}} \cdots p_s^{e_{i,s}}$ entre elles a pour effet d'additionner les exposants sur chacun des nombres premiers de la décomposition : c'est classique (et visuel !). Par ailleurs, un entier est pair si et seulement si les exposants apparaissant dans sa décomposition en produits de facteurs premiers sont tous pairs. Ainsi, on établit que (1) $\Leftrightarrow$ (2). L'équivalence (2) $\Leftrightarrow$ (3) est une conséquence immédiate de la définition du "modulo 2".

Exercice 64. **Crible quadratique.**

Le but de cet exercice est de proposer une méthode efficace pour trouver des entiers X et Y tels que $X^2 = Y^2 \bmod n$. C'est donc un algorithme de factorisation. il explicite la méthode dans le cas d'un exemple numérique $n = 24\,961$.

Soit $\mathcal{S} = \{-1, 2, 3, 5, 7, 11, 13, 19, 23\}$ la famille d'entiers composée par définition de -1 et des nombres premiers inférieurs à une certaine *borne de lissité* B qui est un paramètre de l'algorithme. Nous prenons ici $B = 23$. On pose $m == 157$, et

$$q(x) = (x + \lfloor\sqrt{n}\rfloor)^2 - n = (x + \lfloor\sqrt{24\,961}\rfloor)^2 - 24\,961 = (x + 157)^2 - 24\,961.$$

Pour $x = 0, 1, -1, 2, -2, \ldots$, on va tester (et retenir) les $q(x)$ qui seront B-lisse, c'est-à-dire se décomposant en facteurs premiers dans $\mathcal{S}$. Le fait que $-1 \in \mathcal{S}$ permet d'accepter le cas où $q(x)$ est négatif.

1. Montrer que si $q(x)$ est B-lisse, ses facteurs premiers p_i (qui sont dans $\mathcal{S}$) vérifient que n est un carré modulo p_i.

2. Montrer que n est un carré modulo les premiers suivants $2, 3, 5, 13, 23$, et n'est pas un carré modulo les premiers $7, 11, 19$.

3. En déduire dans notre cas particulier que, si $q(x)$ est B-lisse, les seuls éléments de $\mathcal{S}$ pouvant apparaitre dans la décomposition de $q(x)$ sont $-1, 2, 3, 5, 13, 23$.

On appelle *base de facteurs* la famille $\mathcal{B} = \{-1, 2, 3, 5, 13, 23\}$. C'est donc la famille des facteurs potentiels de $q(x)$ que l'on a "nettoyée" sur la base du résultat de la question précédente. Lorsque $q(x)$ est $B - lisse$, on pose :

- $a_x = x + m$,
- $b_x = q(x) = (-1)^{e_{x,0}} 2^{e_{x,1}} 3^{e_{x,2}} 5^{e_{x,3}} 13^{e_{x,4}} 23^{e_{x,5}}$, (NB : $e_{x,0} \in \{0, 1\}$)
- $e_x = (e_{x,0}, e_{x,1}, e_{x,2}, e_{x,3}, e_{x,4}, e_{x,5}) \in \{0, 1\} \times \mathbb{N}^5$ est le vecteur d'exposants
- $v_x = (v_{x,0}, \ldots v_{x,5}) \in \{0, 1\}^6$ le vecteur binaire d'exposants modulo 2 : $v_{x,i} = e_{x,i} \bmod 2$

4. Pour $x = 0, 1, -1, 2, -2, \ldots$, chercher (par tests successifs) 7 valeurs de x telles que $q(x)$ soit $B - lisse$. Conserver les x, a_x, b_x, e_x, v_x dans un tableau.

5. Montrer que multiplier des b_x entre eux, disons $b_{x_1} \times \ldots b_{x_r}$, correspond à une certaine opération (laquelle ?) sur les e_x.

6. En déduire que se valent :

- (1) $b_{x_1} \times \ldots b_{x_r}$ est un carré ;
- (2) $e_{x_1} + \cdots + e_{x_r}$ est pair ;
- (3) $v_{x_1} + \ldots v_{x_r} = 0$

7. Montrer qu'il y a au moins une combinaison linéaire binaire nulle des vecteurs v_x collectionnés.

8. Vérifier dans le cas particulier de l'exercice que $v_{-1} + v_4 + v_{-6} = 0$.

9. En déduire que $b_{-1} b_4 b_{-6}$ est un carré B-lisse, et qu'il est égal à $a_{-1}^2 a_4^2 a_{-6}^2$ modulo n (on observera que par construction $a_x^2 = b_x \bmod n$)

10. En déduire une écriture $X^2 = Y^2 \bmod n$

11. Calculer les deux facteurs non triviaux de $n = 24\,961$.

1. Si $q(x)$ est B-lisse, on a, tout $p_i \in \mathcal{S}$ divise $q(x0 = (x + \lfloor\sqrt{n}\rfloor)^2 - n$. Ainsi $n = (x + \lfloor\sqrt{n}\rfloor)^2 \bmod p_i$, ce qui signifie que n est un carré modulo p_i.

2. On calcule : $24\,961 \bmod 2 = 1 = 1^2, 24\,961 \bmod 3 = 1 = 1^2, 24\,961 \bmod 5 = 1 = 1^2, 24\,961 \bmod 13 = 1 = 1^2, 24\,961 \bmod 23 = 6 = 11^2$.

3. C'est une conséquence immédiate des questions 1 et 2.

4. On itère tel que demandé, et on conserve les données sur x lorsque$q(x)$ est S-lisse. Cela donne le tableau suivant

x	a_x	b_x	e_x	v_x
0	157	$-312 = -2^3 \cdot 3 \cdot 13$	$(1,3,1,0,1,0)$	$(1,1,1,0,1,0)$
1	158	$3 = 3$	$(0,0,1,0,0,0)$	$(0,0,1,0,0,0)$
-1	156	$-625 = -5^4$	$(1,0,0,4,0,0)$	$(1,0,0,0,0,0)$
2	159	$320 = 2^6 \cdot 5$	$(0,6,0,1,0,0)$	$(0,0,0,1,0,0)$
-2	155	$-936 = -2^3 \cdot 3^2 \cdot 13$	$(1,3,2,0,1,0)$	$(1,1,0,0,1,0)$
4	161	$960 = 2^6 \cdot 3 \cdot 5$	$(0,6,1,1,0,0)$	$(0,0,1,1,0,0)$
-6	151	$-2160 = -2^4 \cdot 3^3 \cdot 5$	$(1,4,3,1,0,0)$	$(1,0,1,1,0,0)$

5. Quand on multiplie entre eux des b_x du tableau, chaque nombre premier p_i de la base de facteurs voit son exposant égale à la somme des exposants qu'il a dans chacun des b_x. L'opération correspond de combinaison multiplicative de b_x correspond donc à additionner les vecteurs d'exposants correspondants.$e_{x_1}, \ldots e_{x_r}$.

6. Comme les b_x sont décomposés sur la base $\mathcal{B}$, il en est de même de tout produit d'entre eux. Or il est immédiat qu'un tel nombre entier est un carré si et seulement si le vecteur d'exposants n'a que des composantes paires, ce qui établit (1) $\Leftrightarrow$ (2). Quant à (2) $\Leftrightarrow$ (3), c'est une conséquence directe de la définition des v_x.

7. On a collectionné 7 vecteurs v_x de dimension 6 sur $\mathbb{F}_2$. Il y a un vecteur de plus que la dimension de l'espace ambiant, la famille ne peut pas être libre dans $\mathbb{F}_2^6$: c'est un résultat classique et fondamental d'algèbre linéaire.

8. Un calcul immédiat permet de vérifier $v_{-1} + v_4 + v_{-6} = 0$.

9. On invoque l'équivalence de la question 6 : $v_{-1} + v_4 + v_{-6} = 0 \Leftrightarrow e_{-1} + e_4 + e_{-6}$ est un vecteur de coordonnées paires $\Leftrightarrow b_{-1}b_4b_{-6}$ est un carré. On vérifie en effet que $b_{-1}b_4b_{-6} = (-1)^2 \cdot 2^{10} \cdot 3^4 \cdot 5^6 = (2^5 \times 3^2 \times 5^3)^2$.

10. Comme $a_x^2 = b_x \bmod n$ pour tout x, on a par produit (modulo n) $(a_{-1}a_4a_{-6})^2 = b_{-1}b_4b_{-6}$, ce qui numériquement donne

$$(156 \cdot 161 \cdot 151)^2 = (2^5 \times 3^2 \times 5^3)^2 \bmod 24\,961$$

On peut réduire chacun des intérieurs de parenthèse modulo 24 961, par principe-même des calculs modulaires. Il vient finalement

$$23\,405^2 = 13\,922^2 \bmod 24\,961$$

11. C'est le calcul de deux pgcd, comme expliqué dans l'exercice IntFact-03, qui permet de conclure. On a $\text{pgcd}(23\,405 - 13\,922, 24\,961) = 109$ et $\text{pgcd}(23\,405 + 13\,922, 24\,961) = 229$.

8.7 Calcul de logarithme discret

DiscLog-01

Exercice 65. **Algorithme de Shanks : pas de bébé - pas de géant**

On se place dans le groupe multiplicatif $\mathbb{F}_{113}^{\times}$.

1. Montrer que 3 est un générateur.

On va déterminer le logarithme x de 57 en base 3 dans ce groupe, *ie*, $3^x = 57 \bmod 113$. On pose $m = \lceil\sqrt{112}\rceil = 11$. Alors tout entier entre 0 et 111 s'écrit $11i + j$ avec $0 \leq i, j \leq 10$. On va chercher x sous la forme $11i + j$ en exploitant le jeu d'écriture suivant :

$$3^{11i+j} = 57 \bmod 113 \Leftrightarrow 3^j = 57 \cdot (3^{-11})^i \bmod 113$$

2. Construire une table $(j, 3^j \bmod 113)$ pour $j = 0, \ldots, 10$. Trier la table par ordre croissant sur le deuxième élément.

3. Pour $i = 0, \ldots, 10$, calculer $57 \cdot (3^{-11})^i \bmod 113$ et regarder s'il appartient à la table de la question 2. Si oui, sortir et noter le couple (i, j) correspondant.

Remarque : on est certain à l'avance que l'on va trouver une solution car 3 est un générateur. La même méthode permet, pour tous α, β, de dire si oui ou non il existe x t.q. $\alpha^x = \beta$ et, si oui, le détermine.

4. En déduire le logarithme recherché.

1. Le cardinal du groupe est $112 = 2^4 \times 7$. Pour vérifier que 3 est générateur, il suffit de vérifier que $3^{112/2}$ et $3^{112/7}$ sont $\neq 1 \bmod 113$. Or $3^{112/2} = 112 \bmod 113$ et $3^{112/7} = 49 \bmod 113$.

2. On a

j	0	1	2	3	4	5	6	7	8	9	10
$3^j \bmod 113$	1	3	9	27	81	17	51	40	7	21	63

puis la table triée selon la deuxième composante

j	0	1	8	2	5	9	3	7	6	10	4
$3^j \bmod 113$	1	3	7	9	17	21	27	40	51	63	81

3. On a $3^{-11} = 3^{101} = 58 \bmod 113$.

i	0	1	2	3	4	5	6	7	8	9
$57 \cdot 58^i$	57	29	100	37	112	55	26	39	2	**3**

La valeur 3 en gras dans la deuxième ligne est aussi dans la liste de la deuxième ligne à la question 2. Elle correspond à $i = 9$ dans la présente liste et $j = 1$ dans la liste de la question 2. Il est inutile d'aller plus loin et de calculer pour $i = 10$. On note le couple $(i, j) = (9, 1)$.

4. On a par construction la relation $57 \times (3^{-11})^9 = 3^1 \bmod 113$, donc $57 = 3^{9\times 11+1} = 3^{100} \bmod 113$, et finalement $\log_3 57 = 100$.

DiscLog-02

Exercice 66. **Algorithme de Pohlig-Hellman.**

Cette méthode exploite efficacement la connaissance du produit en facteurs premiers de l'ordre du groupe, en particulier lorsque ce nombre est friable.

On se place dans le groupe multiplicatif $\mathbb{F}_{251}^{\times}$. Son ordre est $(2, 5)$-friable $250 = 2.5^3$.

1. Vérifier que $\alpha = 71$ est un générateur du groupe.

Soit $\beta = 210$. On va déterminer le logarithme de β en base α. On note x ce logarithme recherché,

$0 \leq x \leq 249$. On a donc $\alpha^x = \beta \bmod 251$. L'idée de la méthode est de calculer séparément des petits logarithmes, puis de recomposer par restes chinois. Plus précisément :

— calcul de $y = x \bmod 2$;

— calcul de $z = x \bmod 5^3$;

— calcul de x par restes chinois.

2. Donner l'ordre de l'élément α^{125}. Calculer cet élément (un peu d'astuce ne nuira pas).

3. Montrer que $\alpha^{125x} = \alpha^{125y}$.

4. En déduire que $y = \log_{\alpha^{125}}(\beta^{125})$. Donner la valeur numérique de y.

5. Montrer que $\alpha^{2x} = \alpha^{2z}$.

On pose $z = \ell_0 + 5\ell_1 + 5^2\ell_2$. La méthode consiste à déterminer les ℓ_i de proche en proche. On a, d'après ce qui précède,

$$(\alpha^2)^{\ell_0+5\ell_1+5^2\ell_2} = \beta^2 \tag{2}$$

6. Montrer qu'en élevant (2) à la puissance 5^2, on obtient une équation ne faisant intervenir que ℓ_0.

7. En déduire $\ell_0 = \log_{\alpha^{50}}(\beta^{50})$. Calculer.
On réécrit (2) en

$$(\alpha^2)^{5\ell_1+5^2\ell_2} = \beta^2\alpha^{-2\ell_0} \tag{3}$$

8. Montrer qu'en élevant (3) à une certaine puissance (laquelle ?), on obtient une équation ne faisant intervenir que ℓ_1.

9. En déduire ℓ_1, littéralement puis numériquement.

10. Calculer ℓ_2 de manière analogue.

11. En déduire z.

12. En déduire finalement x.

1. Le cardinal du groupe est $250 = 2 \times 5^3$. Il suffit de vérifier que $71^{250/2}$ et $71^{250/5}$ sont $\neq 1 \bmod 251$. Or on a $71^{250/2} = 250 \bmod 251$ et $71^{250/5} = 20 \bmod 251$.

2. Le calcul a été fait à la question 1. En fait, on peut aussi ruser : $(\alpha^{125})^2 = \alpha^{250} = 1 \bmod 251$. Donc α^{125} est une racine carrée de 1, mais n'est pas 1 lui-même (sinon α ne serait pas générateur). Donc c'est -1 (251 est premier donc il n'y a que deux racines carrées de 1), c'est-à-dire 250. Son ordre est évidemment 2.

3. Par hypothèse $y = x \bmod 2$, ie $x = y + 2k$. Ainsi $\alpha^{125x} = \alpha^{125(y+2k)} = \alpha^{125y+250k)} = \alpha^{125y} \bmod 251$. Rappelons en effet que $\alpha^{250} = 1 \bmod 251$.

4. Puisque $\alpha^x = \beta \bmod 251$, on a $\alpha^{125x} = \beta^{125} \bmod 251$, et donc $(\alpha^{125})^y = \beta^{125} \bmod 251$. Finalement $y = \log_{\alpha^{125}} \beta^{125}$.
Numériquement, $\beta^{125} = 250 \bmod 251$, donc $y = 1$.

5. Par hypothèse $z = x \bmod 125$, *i.e.* $x = z + 125k$. Ainsi $\alpha^{2x} = \alpha^{2(z+125k)} = \alpha^{2z}\alpha^{250k} = \alpha^{2z}$.

6. Élevons l'équation

$$(\alpha^2)^{\ell_0+5\ell_1+5^2\ell_2} = \beta^2$$

à la puissance 5^2. Il vient :

$$\alpha^{50\ell_0+250\ell_1+1250\ell_2} = \beta^{50}$$

soit, toujours en utilisant $\alpha^{250} = 1$,

$$(\alpha^{50})^{\ell_0} = \beta^{50}$$

7. La question précédente donne immédiatement $\ell_0 = \log_{\alpha^{50}}(\beta^{50})$. On a encore un problème de logarithme discret, mais celui-ci est plus simple car l'ordre de la base α^{50} est plus faible ($= 5$). Il est très rapide de déterminer ℓ_0, qui $\in [0, 4]$. Numériquement, on a $149 = 20^{\ell_0} \bmod 251$. Un calcul donne $\ell_0 = 2$.

8. Il faut élever l'équation

$$(\alpha^2)^{5\ell_1+5^2\ell_2} = \beta^2\alpha^{-2\ell_0}$$

à la puissance 5. On récupère

$$\alpha^{50\ell_1+250\ell_2} = \beta^{10}\alpha^{-10\ell_0}$$

soit, toujours avec $\alpha^{250} = 1$,

$$(\alpha^{50})^{\ell_1} = \beta^{10}\alpha^{-10\ell_0}$$

Notons que cette équation fait intervenir ℓ_1 et ℓ_0 que l'on a déterminé juste avant. Il s'agit bien d'un méthode itérative.

9. On a $\ell_1 = \log_{\alpha^{50}}(\beta^{10}\alpha^{-10\ell_0})$. Remarquons que c'est encore un logarithme en base α^{50} qu'il faut calculer. Numériquement, $\ell_1 = \log_{20}(113)$. De même que ℓ_0, ℓ_1 est à chercher dans $[0, 4]$. Un calcul donne $\ell_1 = 4$.

10. Pour calculer ℓ_2, on écrit

$$(\alpha^2)^{5^2\ell_2} = \beta^2\alpha^{-2(\ell_0+5\ell_1)}$$

c'est-à-dire

$$(\alpha^{50})^{\ell_2} = \beta^2\alpha^{-2(\ell_0+5\ell_1)}$$

qui donne

$$\ell_2 = \log_{\alpha^{50}}(\beta^2\alpha^{-2(\ell_0+5\ell_1)})$$

Numériquement, $\ell_2 = \log_{20}(149) = 2$.

11. Le calcul itératif de ℓ_0, ℓ_1, ℓ_2 permet de reconstituer $z = \ell_0 + 5\ell_1 + 5^2\ell_2$. Numériquement $z = 2 + 5 \times 4 + 5^2 \times 2 = 72$.

12. On a $x = 1 \bmod 2$ et $x = 72 \bmod 125$. Le théroème chinois permet de reconstituer x. La technique est classique, on écrit $x = 2u + 125v$ et on détermine séparément u et v :

— $x = 1 \bmod 2$ donne $125v = 1 \bmod 2$, soit $v = 1$;

— $x = 72 \bmod 125$ donne $2u = 72 \bmod 125$, soit $u = 36$;

finalement $x = 72 + 125 = 197$, le logarithme recherché. On vérifie numériquement que $\alpha^{197} = 71^{197} = 210 = \beta \bmod 251$.

DiscLog-03

Exercice 67. **Algorithme de calcul d'index.**

On se place dans le groupe multiplicatif $\mathbb{F}_{229}^{\times}$.

1. Vérifier que 6 est un générateur du groupe.

(Pour ce faire, on supposera **momentanément** connue la décomposition en facteurs premiers de l'ordre du groupe, *ie* $228 = 2^2.3.19$. On observera que, dans ce qui suit, c'est-à-dire le calcul du log discret proprement

dit, la connaissance de cette factorisation est inutile.)

On va calculer le logarithme en base 6 de 13. On considère la base des premiers $\mathcal{B} = \{2, 3, 5, 7, 11\}$.

2. Montrer que les nombres $6^i \bmod 229$ se décomposent dans la base $\mathcal{B}$, pour $i = 12, 18, 62, 100, 143, 206$.
3. En déduire un système linéaire dont les inconnues sont les $\log_6(b),\ b \in \mathcal{B}$.
4. Résoudre ce système pour obtenir le logarithme des éléments de $\mathcal{B}$.
5. Montrer que $13 \cdot 6^{77} \bmod 229$ se décompose sur la base $\mathcal{B}$.
6. En "passant aux log" (en base 6), en déduire le logarithme recherché.

1. Pour vérifier que 6 est un générateur de $\mathbb{F}_{229}^{\times}$, il suffit de vérifier que $6^{228/2}, 6^{228/3}$ et $6^{228/19}$ sont $\neq 1 \bmod 229$. Un calcul numérique donne :

$$6^{228/2} = 228,\ 6^{228/3} = 134,\ 6^{228/19} = 165 \bmod 229$$

2. Un calcul établit

i	$6^i \bmod 229$	Décomposition dans $\mathcal{B}$
12	165	$3 \cdot 5 \cdot 11$
18	176	$2^4 \cdot 11$
62	154	$2 \cdot 7 \cdot 11$
100	180	$2^2 \cdot 3^2 \cdot 5$
143	198	$2 \cdot 3^2 \cdot 11$
206	210	$2 \cdot 3 \cdot 5 \cdot 7$

3. En passant au logarithme en base 6 dans le tableau précédent, il vient un système d'équations modulo 228 :

$$\begin{aligned}
12 &= (\log_6 3) + (\log_6 5) + (\log_6 11) \\
18 &= 4(\log_6 2) + (\log_6 11) \\
62 &= (\log_6 2)(\log_6 7) + (\log_6 11) \\
100 &= 2(\log_6 2) + 2(\log_6 3) + (\log_6 5) \\
143 &= (\log_6 2) + 2(\log_6 3) + (\log_6 11) \\
206 &= (\log_6 2) + (\log_6 3) + (\log_6 5) + (\log_6 7)
\end{aligned}$$

4. Une résolution, par pivot de Gauss par exemple, donne :

$$\begin{aligned}
\log_6 2 &= 21 \\
\log_6 3 &= 208 \\
\log_6 5 &= 98 \\
\log_6 7 &= 107 \\
\log_6 11 &= 162
\end{aligned}$$

5. Là encore, on calcule $13 \times 6^{77} \bmod 229 = 147 = 3 \times 7^2$.
6. En passant aux logarithmes en base 6, il vient

$$(\log_6 13) + 77 = (\log_6 3) + 2(\log_6 7) \bmod 228$$

soit finalement $\log_6 13 = 117$.

8.8 Calculs sur courbes elliptiques

Exercice 68. Tuto courbes elliptiques

Soit E la courbe elliptique définie sur le corps fini $\mathbb{F}_5$ par l'équation affine : $y^2 = x^3 + 1$.

1. Calculer de manière exhaustive le nombre de points de la courbe (NB : possible ici car taille jouet).
2. Retrouver la liste des points par utilisation des racines carrées.
3. Vérifier la borne de Hasse-Weil.
4. Vérifier que $P = (2; 3)$ est générateur de la courbe en calculant ses multiples successifs.

1. Il y a $5^2 = 25$ points rationnels $(x, y) \in \mathbb{F}_5^2$ candidats, et le point à l'infini. pour les 25 points rationnels, on teste directement sur l'équation l'appartenance à E.

```
(0,0) in E ? <=> ? 0^2 = 0^3+1 (mod 5) <=> 0 = 1 (mod 5) : false
(0,1) in E ? <=> ? 1^2 = 0^3+1 (mod 5) <=> 1 = 1 (mod 5) : true
(0,2) in E ? <=> ? 2^2 = 0^3+1 (mod 5) <=> 4 = 1 (mod 5) : false
(0,3) in E ? <=> ? 3^2 = 0^3+1 (mod 5) <=> 4 = 1 (mod 5) : false
(0,4) in E ? <=> ? 4^2 = 0^3+1 (mod 5) <=> 1 = 1 (mod 5) : true
(1,0) in E ? <=> ? 0^2 = 1^3+1 (mod 5) <=> 0 = 2 (mod 5) : false
(1,1) in E ? <=> ? 1^2 = 1^3+1 (mod 5) <=> 1 = 2 (mod 5) : false
(1,2) in E ? <=> ? 2^2 = 1^3+1 (mod 5) <=> 4 = 2 (mod 5) : false
(1,3) in E ? <=> ? 3^2 = 1^3+1 (mod 5) <=> 4 = 2 (mod 5) : false
(1,4) in E ? <=> ? 4^2 = 1^3+1 (mod 5) <=> 1 = 2 (mod 5) : false
(2,0) in E ? <=> ? 0^2 = 2^3+1 (mod 5) <=> 0 = 4 (mod 5) : false
(2,1) in E ? <=> ? 1^2 = 2^3+1 (mod 5) <=> 1 = 4 (mod 5) : false
(2,2) in E ? <=> ? 2^2 = 2^3+1 (mod 5) <=> 4 = 4 (mod 5) : true
(2,3) in E ? <=> ? 3^2 = 2^3+1 (mod 5) <=> 4 = 4 (mod 5) : true
(2,4) in E ? <=> ? 4^2 = 2^3+1 (mod 5) <=> 1 = 4 (mod 5) : false
(3,0) in E ? <=> ? 0^2 = 3^3+1 (mod 5) <=> 0 = 3 (mod 5) : false
(3,1) in E ? <=> ? 1^2 = 3^3+1 (mod 5) <=> 1 = 3 (mod 5) : false
(3,2) in E ? <=> ? 2^2 = 3^3+1 (mod 5) <=> 4 = 3 (mod 5) : false
(3,3) in E ? <=> ? 3^2 = 3^3+1 (mod 5) <=> 4 = 3 (mod 5) : false
(3,4) in E ? <=> ? 4^2 = 3^3+1 (mod 5) <=> 1 = 3 (mod 5) : false
(4,0) in E ? <=> ? 0^2 = 4^3+1 (mod 5) <=> 0 = 0 (mod 5) : true
(4,1) in E ? <=> ? 1^2 = 4^3+1 (mod 5) <=> 1 = 0 (mod 5) : false
(4,2) in E ? <=> ? 2^2 = 4^3+1 (mod 5) <=> 4 = 0 (mod 5) : false
(4,3) in E ? <=> ? 3^2 = 4^3+1 (mod 5) <=> 4 = 0 (mod 5) : false
(4,4) in E ? <=> ? 4^2 = 4^3+1 (mod 5) <=> 1 = 0 (mod 5) : false
There are 5 rational points in E
```

En comptant le point à l'infini, il y a 6 points sur la courbe elliptique E.

2. Le principe est de faire une seule boucle sur x, de calculer $x^3 + 1$ et d'en extraire les éventuelles racines carrées.

```
List rational points of the Elliptic Curve of equation: y^2 = x^3 + 0x + 1 in
GF(5)
x:0   0^3 + 0*0 + 1 (mod 5) = 1 has square root: [ 1, 4 ]
  (0,1) is a point of the curve
  (0,4) is a point of the curve
x:1   1^3 + 0*1 + 1 (mod 5) = 2 has square root: []
x:2   2^3 + 0*2 + 1 (mod 5) = 4 has square root: [ 2, 3 ]
  (2,2) is a point of the curve
  (2,3) is a point of the curve
x:3   3^3 + 0*3 + 1 (mod 5) = 3 has square root: []
x:4   4^3 + 0*4 + 1 (mod 5) = 0 has square root: [ 0 ]
  (4,0) is a point of the curve
There are 5 rational points in E
Cardinality is 6 including the point at infinity
```

3. D'après la borne de Hasse-Weil, on a

$$5 + 1 - 2\sqrt{5} \leq \#E \leq 5 + 1 + 2\sqrt{5}$$

ce qui est de toute évidence correct.

4. On a $0 \cdot P = O_{\mathbb{E}} = (\infty; \infty)$ et $1 \cdot P = P = (2; 3)$. On calcule les multiples suivants de $P = (2; 3)$ par simple application des formules d'addition

$$\begin{array}{rcl} 0 \cdot P & = & (\infty; \infty) \\ 1 \cdot P & = & (2; 3) \\ 2 \cdot P & = & (0; 1) \\ 3 \cdot P & = & (4; 0) \\ 4 \cdot P & = & (0; 4) \\ 5 \cdot P & = & (2; 2) \\ 6 \cdot P & = & (\infty; \infty) \end{array}$$

Ces multiples successifs couvrent bien l'ensemble des points de la courbe, et ça "boucle" comme attendu au bout de 6 itérations.

```
This is Elliptic Curve add points procedure with
  Equation of the Elliptic Curve: y^2 = x^3 + 0x + 1 in GF(5)
  input points P and Q with  (xP,yP)=(2,3)  (xQ,yQ)=(2,3)
Intermediate values s=2 t=4
Result is point R with  (xR,yR)=(0,1)
[2]P = (0,1)
This is Elliptic Curve add points procedure with
  Equation of the Elliptic Curve: y^2 = x^3 + 0x + 1 in GF(5)
  input points P and Q with  (xP,yP)=(2,3)  (xQ,yQ)=(0,1)
Intermediate values s=1 t=1
Result is point R with  (xR,yR)=(4,0)
[3]P = (4,0)
This is Elliptic Curve add points procedure with
  Equation of the Elliptic Curve: y^2 = x^3 + 0x + 1 in GF(5)
  input points P and Q with  (xP,yP)=(2,3)  (xQ,yQ)=(4,0)
Intermediate values s=1 t=1
Result is point R with  (xR,yR)=(0,4)
[4]P = (0,4)
This is Elliptic Curve add points procedure with
  Equation of the Elliptic Curve: y^2 = x^3 + 0x + 1 in GF(5)
  input points P and Q with  (xP,yP)=(2,3)  (xQ,yQ)=(0,4)
Intermediate values s=2 t=4
Result is point R with  (xR,yR)=(2,2)
[5]P = (2,2)
This is Elliptic Curve add points procedure with
  Equation of the Elliptic Curve: y^2 = x^3 + 0x + 1 in GF(5)
  input points P and Q with  (xP,yP)=(2,3)  (xQ,yQ)=(2,2)
Result is point R with  (xR,yR)=(1000000,1000000)
[6]P = (1000000,1000000)
```

8.9 Tests de primalité probable

Ces tests sont appelés ainsi car ils ont la propriété de présenter des probabilités de "fausse alarme". Autrement dit, lorsqu'ils retournent "`premier`", il y a une probabilité que le nombre soit en fait composite. À l'inverse, il n'y a pas de "non détection" : s'ils répondent "`composite`", le nombre est en effet composite de manière certaine. Ceci tient au fait qu'ils sont fondés sur des réciproques de propriétés vérifiées par les nombres premiers.

PrimProb-01

Exercice 69. **Test de Fermat.**

Soit n un entier et $\phi(n)$ son indicateur d'Euler.

1. Soit a un entier. Montrer que si $\text{pgcd}(a, n) = 1$, alors $a^{\phi(n)} = 1 \bmod n$ (théorème d'Euler). En déduire que, si n est premier, alors pour tout $a \in [1, n-1]$, on a $a^{n-1} = 1 \bmod n$ (petit théorème de Fermat).

À propos de la réciproque

2. Vérifier que $3^{340} \neq 1 \bmod 341$. En déduire que 341 n'est pas premier. 3 s'appelle un *témoin de Fermat*

du caractère composé de 341.

3. Vérifier que $2^{340} = 1 \bmod 341$, et que cette égalité fournit un contre-exemple à la réciproque du petit théorème de Fermat. 2 s'appelle un *menteur de Fermat* pour 341, et 341 et dit *pseudo premier de Fermat* en base 2.

On pourrait croire que, pour un n donné, les a menteurs sont rares. La suite de l'exercice montre au contraire que pour certains nombres n particuliers, tous les a sont des menteurs !

4. On suppose que n est composé, sans facteur carré, $n = p_1 \dots p_s$. On suppose aussi que pour tout p diviseur premier de n, $p-1$ divise $n-1$. Un tel n s'appelle un *nombre de Carmichael*. Montrer que pour tout a tel que $\text{pgcd}(a, n) = 1$, on a $a^{n-1} = 1 \bmod n$. Autrement dit, tout a est un menteur pour la primalité de n.

5. Montrer que 561 est un nombre de Carmichael.
Voici maintenant le test de Fermat :

---

1. pour $i = 1, \dots, t$,
   (a) choisir une base a telle que $\text{pgcd}(a, n) = 1$.
   (b) Calculer $r = a^{n-1} \bmod n$.
   (c) Si $r \neq 1$, retourner "`composite`"
2. Retourner "`premier`"

---

6. Démontrer que si n est premier, le test retourne "`premier`".

7. Réciproquement, si le test retourne "`premier`", n peut-il être composé ? En particulier, que se passe-t-il si n est un nombre de Carmichael ?

1. C'est le théorème de Lagrange, car $\phi(n)$ est l'ordre du groupe multiplicatif $(\mathbb{Z}/n\mathbb{Z})^\times$. En particulier si n est premier, $\phi(n) = n - 1$.

2. On a $\text{pgcd}(3, 341) = 1$ et $3^{340} = 56 \bmod 341$. Ainsi 341 n'est pas premier, par contraposée du petit théorème de Fermat.

3. On a $\text{pgcd}(2, 341) = 1$ et $2^{340} = 1 \bmod 341$, pourtant 341 n'est pas premier. Ainsi la réciproque du théorème de Fermat est fausse.

4. Soit p_i un diviseur premier de n, et a premier avec n. a est en particulier premier avec p_i, donc, toujours par Fermat, on a $a^{p_i - 1} = 1 \bmod p_i$. Comme $p_i - 1$ divise $n - 1$, on a $a^{n-1} = 1 \bmod p_i$, c'est-à-dire p_i divise $a^{n-1} - 1$. Mais ceci est vrai pour tout i, donc $n = \prod_i p_i$ divise $a^{n-1} - 1$, cqfd.

5. On a $561 = 3 \times 11 \times 17$. Comme $2, 10$ et 16 divisent 560, 561 est par définition un nombre de Carmichael.

6. Si n est premier, r est toujours égal à 1 (théorème de Fermat toujours). L'algorithme ne peut que retourner "`premier`".

7. Si l'algorithme retourne "`premier`", c'est que l'on a choisit des bases a qui sont des menteurs. On pourrait croire que cette "malchance"est rare, mais en fait non. Si n est un nombre de Carmichael, la question 4 montre

que tout a est menteur et le test retourne forcément "`premier`" (donc le test se trompe de manière certaine).

PrimProb-02

Exercice 70. **Test de Miller-Rabin.**

Soit n un entier premier impair. On écrit $n-1 = 2^s r$, r impair. Soit a un entier $\in [1, n-1]$. Noter qu'alors $\text{pgcd}(a, n) = 1$.

1. Montrer que ou bien $a^r = 1 \bmod n$, ou bien l'un des nombres $a^r, a^{2r}, a^{4r}, \ldots, a^{2^{s-1}r}$ est $= -1 \bmod n$.

À propos de la réciproque.

2. Vérifier que $3^{90} = 1 \bmod 91$, alors que $3^{45} \neq 1 \bmod 91$ et $3^{45} \neq -1 \bmod 91$. On a donc, en posant $\beta = 3^{45} \bmod 91$, $\beta^2 = 1$ et $\beta \neq \pm 1$. Pourquoi ceci n'est-il pas contradictoire ?

3. Que peut-on déduire sur l'entier 91 des deux inégalités $3^{45} \neq 1 \bmod 91$ et $3^{45} \neq -1 \bmod 91$. On dit que 3 est un *témoin fort* de la compositude de 91.

Remarque : on note au passage que 3 est un menteur de Fermat pour 91.

4. Montrer que $9^{45} = 1 \bmod 91$. En déduire que cela fournit un contre exemple à la question 1. 9 est appelé *menteur fort* pour 91, et 91 est dit *pseudo premier fort* en base 9.

5. Montrer que si a est un menteur fort pour n, alors c'est aussi un menteur de Fermat.

Le résultat de la question 1 suggère le test suivant, dit de Miller-Rabin :

---

1. pour $i = 1, \ldots, t$,
   (a) choisir $a \in [2, n-2]$,
   (b) si $a^r \neq 1 \bmod n$ et $a^r, a^{2r}, a^{4r}, \ldots, a^{2^{s-1}r}$ sont tous $\neq -1 \bmod n$, retourner "`composite`"
2. Retourner "`premier`".

---

6. Démontrer que si n est premier, le test retourne "`premier`".

On va étudier la réciproque, et supposer que le test retourne "premier" sur un certain entier n impair. On admet le résultat important suivant :

> **Théorème** *Si n est un nombre impair composite, le nombre de menteurs forts de n est $\leq n/4$, et si $n \neq 9$, le nombre de menteurs forts de n est $\leq \phi(n)/4$.*

7. En supposant indépendantes les diverses itérations de la boucle sur i dans le test, montrer que la probabilité que n soit composite est $\leq (1/4)^t$.

Ainsi, contrairement au test de Fermat, l'augmentation de t permet de rendre résiduelle la probabilité de fausse alarme pour **toute** entrée. Autrement dit :

- le test de Miller-Rabin marche presque sûrement pour toute entrée :
- le test de Fermat marche presque sûrement pour presque toute entrée, mais échoue sûrement pour certaines entrées (les nombres de Carmichael)

1. On écrit

$$\begin{aligned} a^{n-1}-1 &= a^{2^s r}-1 = (a^{2^{s-1}r}+1)(a^{2^{s-1}r}-1) \\ &= (a^{2^{s-1}r}+1)(a^{2^{s-2}r}+1)(a^{2^{s-2}r}-1) \\ &= \cdots = (a^{2^{s-1}r}+1)(a^{2^{s-2}r}+1)\ldots(a^r+1)(a^r-1) \end{aligned}$$

Cette égalité est en particulier vraie modulo n et, comme n est premier, on peut appliquer "un produit de facteurs est nul si et seulement si l'un des facteurs est nul". Ceci donne exactement la réponse.

2. On a $3^{45} = 27 \bmod 91$. Ainsi $3^{45} \bmod 91$ est une racine carrée de 1 modulo 91, différente de 1 et -1. Ce n'est pas contradictoire car 91 est composé, et 1 peut avoir plus de deux racines carrées modulo 91. Plus précisément comme $91 = 7 \times 13$, 1 a quatre racines carrées modulo 91.

3. Par la contraposée de la question 1 (qu'on a en fait évoquée à la question 2), 91 est forcément composé.

4. $9^{45} = 3^{90} = 1 \bmod 91$. Comme $\text{pgcd}(9, 91) = 1$, cela fournit un contre exemple à la réciproque de la question 1 (avec $n = 91, r = 45, s = 1$).

5. Supposons que a est un menteur fort pour n. Si $a^r = 1 \bmod n$, comme r divise $n-1$ on a tout de suite $a^{n-1} = 1 \bmod n$. Sinon l'un des nombres $a^{2^i r}$ est $= -1 \bmod n$, pour $i = 0, \ldots s-1$. Mais alors $a^{2^{i+1}r} = 1 \bmod n$, puis comme $i+1 \leq s$, $2^{i+1}r$ divise $2^s r = n-1$, et on a encore $a^{n-1} = 1 \bmod n$.

6. D'après la question 1, si n est premier le test ne peut jamais retourner "`composite`".

7. Puisque le test ne retourne pas "`composite`", c'est que l'on a choisi pour les nombres a uniquement des menteurs forts pour n. D'après le théorème, la probabilité de choisir un mentuer fort est $\leq 1/4$ à chaque étape. les étapes étant supposées indépendantes, le résultat suit.

8.10 Tests de primalité prouvée

Au contraire des précédents, ces tests offrent une certitude de primalité, par l'intermédiaire d'un certificat vérifiable aisément.

PrimProuv-01

Exercice 71. Test $n-1$, cas totalement factorisé.
Soit $n \geq 3$ un entier.

1. (Version forte) On suppose qu'il existe un entier a vérifiant :
— $a^{n-1} = 1 \bmod n$;
— pour tout diviseur premier p de $n-1$, $a^{(n-1)/p} \neq 1 \bmod n$.
Montrer que n est premier. (On pourra raisonner sur l'ordre de a).

2. (Version faible) On suppose que pour tout diviseur premier p de $n-1$, il existe a_p tel que
— $a_p^{n-1} = 1 \bmod n$;
— $a_p^{(n-1)/p} \neq 1 \bmod n$.
Montrer que n est premier. (On pourra raisonner sur l'ordre de a_p).
Remarque : on observera que les réciproques sont vraies.

Préliminaire : dans les deux cas, nous montrons que les hypothèses impliquent que $n-1$ divise $\phi(n)$. Cela

permet de conclure car ceci se produit si seulement si n est premier (vérification facile).

1. Soit d l'ordre de a modulo n. Par hypothèse d divise $n-1$, Supposons par l'absurde que $d \neq n-1$. Alors il existe un facteur premier p de $n-1$ tel que d divise $(n-1)/p$. Mais alors $a^d = 1 \bmod n$ implique $a^{(n-1)/p} = 1 \bmod n$, contradiction. Ainsi, l'élément a est d'ordre $d = n-1$. Par ailleurs d divise $\phi(n)$ (Fermat), d'où la conclusion d'après notre préliminaire.

2. Notons $n = p_1^{e_1} \dots p_s^{e_s}$. Soit d_i l'ordre de a_{p_i} modulo n. Par hypothèse, d_i divise $n-1$ mais ne divise pas $(n-1)/p_i$. En conséquence, dans la décomposition en facteurs premiers de d_i, on trouve le facteur $p_i^{e_i}$, autrement dit $p_i^{e_i}$ divise d_i. Par ailleurs d_i divise $\phi(n)$, donc $p_i^{e_i}$ divise $\phi(n)$, et ce pour tout i. Il suit $n-1 = \prod_i p_i^{e_i}$ divise $\phi(n)$, d'où la conclusion encore d'après notre préliminaire.

PrimProuv-02

Exercice 72. **Test $n-1$, cas partiellement factorisé : théorème de Pocklington**

Soit n entier impair. On suppose connue une factorisation partielle de $n-1$:

$$n-1 = RF = Rp_1^{e_1} \dots p_s^{e_s}, \quad \text{pgcd}(R,F) = 1.$$

On suppose qu'il existe un entier a vérifiant :

— $a^{n-1} = 1 \bmod n$;

— $\forall i = 1, \dots, s, \quad \text{pgcd}(a^{(n-1)/p_i} - 1, n) = 1$.

Soit p un facteur premier de n.

1. Montrer que l'ordre multiplicatif d de a dans $\mathbb{F}_p$ divise $n-1$.

Dans la suite, on écrit donc $d = R' p_1^{a_1} \dots p_s^{a_s}$, avec R' divise R et $a_i \leq e_i$ pour $i = 1, \dots, s$.

2. On fixe $i \in [1, s]$. Montrer que p ne divise pas $a^{(n-1)/p_i} - 1$. En déduire que d ne divise pas $(n-1)/p_i$, puis que $p_i^{e_i}$ divise d.

3. Vérifier que F divise d, et que d divise $p-1$. Conclure que F divise $p-1$, *ie* $p \equiv 1 \bmod F$.

On suppose maintenant $R < F$. Autrement dit, $n-1$ est factorisé à "plus de la moitié de sa taille".

4. En utilisant le résultat de la question 3, montrer que tout facteur premier p de n vérifie $p > \sqrt{n}$. Conclure.

1. Comme $a^{n-1} = 1 \bmod n$, a fortiori $a^{n-1} = 1 \bmod p$, puisque p divise n. Il suit par définition de d que d divise $n-1$.

2. Par l'absurde, si p divisait $a^{(n-1)/p_i} - 1$, il diviserait aussi $\text{pgcd}(a^{(n-1)/p_i} - 1, n) = 1$, contradiction. Ainsi $a^{(n-1)/p_i} \neq 1 \bmod p$, et donc d ne divise pas $(n-1)/p_i$. Examinons les exposants de p_i dans les décompositions de $n-1$ et $(n-1)/p_i$ et d : d'après la question 1, d divise $n-1$ donc $a_i \leq e_i$ et d'après la présente question, d ne divise pas $(n-1)/p_i$, donc $a_i > e_i - 1$. Il suit $a_i = e_i$, et donc $p_i^{e_i}$ divise d.

3. D'après la question 2, $p_i^{e_i}$ divise d pour tout i. Ainsi $F = \prod_i p_i^{e_i}$ divise d. Par ailleurs, Fermat implique $a^{p-1} = 1 \bmod p$, donc d divise $p-1$. Par transitivité, F divise $p-1$.

4. Comme $RF = n$, l'hypothèse $R < F$ implique $F > \sqrt{n}$. D'après la question 3, $p \equiv 1 \bmod F$, en particulier $p \geq F + 1 > \sqrt{n}$. Ce raisonnement étant valable pour tout facteur premier de n, on en déduit que tout facteur

premier de n est $> \sqrt{n}$. Évidemment, cela implique n premier.

PrimProuv-03

Exercice 73. L'algorithme AKS (Agrawal, Kayal, Saxena)

Cet algorithme est une preuve de primalité dont la complexité est polynomiale. La version originelle date de 2002. Si l'algorithme en lui-même n'est pas le plus efficace en pratique, en revanche le résultat théorique résout une question ouverte : il existe un algorithme déterministe polynomial prouvant la primalité d'un entier. Bien entendu, un très grand nombre de mathématiciens ont réagi à la première publication et, forts de moult suggestion, les auteurs ont proposé dernièrement une version à la fois plus simple à appréhender et de complexité plus faible.
En outre, l'une de caractéristiques principales de l'article (en particulier dans sa version la plus récente) est l'extraordinaire simplicité des mathématiques (niveau licence) mis en oeuvre par rapport à l'importance du résultat.
Le présent exercice est construit autour de la dernière version.

Tout d'abord voici l'algorithme.

Entrée : un entier $n \geq 2$.

1. Si n est la puissance non triviale d'un entier, *ie* $n = a^b$ pour $a \in \mathbb{N}$ et $b \geq 2$, retourner "`composite`".
2. Trouver le plus petit entier r tel que $o_r(n) > \log^2 n$.
3. S'il existe $a \leq \min(r, n)$ tel que $1 < \text{pgcd}(a, n) < n$, retourner "`composite`".
4. Si $n \leq r$, retourner "`premier`".
5. Pour $a = 1$ à $\ell := \lfloor \sqrt{\phi(r)} \log n \rfloor$
   si $(X + a)^n \neq X^n + a \pmod{X^r - 1, n}$, retourner "`composite`".
6. **Retourner** "`premier`".

Notations :

— log désigne le logarithme en base 2.

— $\phi(a)$ est l'indicatrice d'Euler de l'entier a, nombre d'entiers inférieurs à a et premiers avec a.

— $o_r(n)$ est l'ordre de n modulo r, c'est-à-dire le plus petit entier k tel que $n^k = 1 \pmod r$. On a $\text{pgcd}(r, n) = 1$ (Bezout), et $o_r(n) | \phi(r)$ (l'ordre de n dans $(\mathbb{Z}/r\mathbb{Z})^\times$ divise le cardinal de ce dernier (théorème de Lagrange).

Au travers des questions qui suivent, nous allons étudier à la fois le caractère exact de l'algorithme ainsi que sa complexité.

1. Montrer que pour tout $1 \leq i \leq n$,
$$i\binom{n}{i} = n\binom{n-1}{i-1}.$$

2. Soit n premier, montrer que pour tout $1 \leq i \leq n-1$,
$$\binom{n}{i} = 0 \pmod n.$$

3. Soit n composite et q un facteur premier de n. On note $n = q^k m$ avec m non multiple de q. On suppose (par l'absurde) que $q^k | \binom{n}{q}$, et on note r le quotient.

   a. Montrer que $q | \binom{n-1}{q-1}$. (On pourra utiliser que $q\binom{n}{q} = n\binom{n-1}{q-1}$, et montrer que $qr = m\binom{n-1}{q-1}$)
   b. Montrer qu'il n'y a aucun multiple de q dans la séquence d'entiers $n - q + 1, \ldots, n - 1$
   c. En déduire une contradiction, puis que n ne divise pas $\binom{n}{q}$

4. Soient $n \in \mathbb{N}, a \in \mathbb{Z}$ et $\text{pgcd}(a, n) = 1$. Montrer que n est premier si et seulement si

$$(X + a)^n = X^n + a \pmod{n}.$$

On utilisera la question 2 pour la partie directe et la question 3 pour la réciproque.

5. Soit n premier. Montrer que l'algorithme ne peut pas retourner "`composite`".

6. On suppose que l'algorithme retourne "`premier`" à l'étape 4. Montrer que n est premier (indication : raisonner par l'absurde et montrer qu'alors, l'algorithme sort à l'étape 3).

On suppose désormais que l'algorithme retourne "`premier`" à l'étape 6. On admettra le lemme suivant :

Lemme. *Pour tout entier $n \geq 2$, il existe $r \leq max\{3, \lceil \log^5 n \rceil\}$ tel que $o_r(n) > \log^2 n$.*

7. Montrer que $\text{pgcd}(r, n) = 1$ (indication : par l'absurde, montrer que sinon l'algorithme sort à l'étape 3 ou 4). En déduire que $\text{pgcd}(r, p) = 1$ pour tout diviseur premier p de n.

8. Montrer qu'il existe un diviseur premier p de n tel que $o_r(p) > 1$. (Raisonner par l'absurde et montrer qu'alors $o_r(n) = 1$).

9. Montrer que $p > r$ (indication : par l'absurde, montrer que sinon l'algorithme sort à l'étape 3 ou à l'étape 4).

10. Justifier que $o_r(n) \leq \phi(r)$, puis $\log^2 n \leq \phi(r)$. En déduire $\ell < \phi(r) < r$.

On considère à présent l'ensemble :

$$\mathcal{G} = \{\Pi_{a=0}^{\ell}(X + a)^{e_a} \pmod{h(X), p} : e_a \geq 0\},$$

où $h(X)$ est un diviseur irréductible de $X^r - 1$ dans $\mathbb{F}_p[X]$, de degré $o_r(p)$, et dont les racines sont des racines primitives de l'unité dans l'extension $K = \mathbb{F}_p[X]/(h(X)) = \mathbb{F}_{p^{o_r(p)}}$. Autrement dit, $\mathcal{G}$ est un sous groupe multiplicatif de $K^{\times}$.

Le résultat suivant est une des parties cruciales de l'article, et la démonstration y est réalisée en détails. Par manque de temps, on se contentera d'admettre le résultat dans le cadre de cet exercice.

Lemme. *Il existe un entier t, $\log^2 n < t < \phi(r)$, tel que*
(1) $|\mathcal{G}| \geq \binom{t+\ell}{t-1}$.
(2) Si n n'est pas une puissance de p, $|\mathcal{G}| \leq n^{\sqrt{t}}$.

11. Montrer que

$$\binom{t+\ell}{t-1} \geq \binom{\ell + 1 + \lfloor \sqrt{t} \log n \rfloor}{\lfloor \sqrt{t} \log n \rfloor} \geq \binom{2\lfloor \sqrt{t} \log n \rfloor + 1}{\lfloor \sqrt{t} \log n \rfloor}.$$

On observera que $\log^2 n < t < \phi(r)$ implique en particulier $t > \lfloor \sqrt{t} \log n \rfloor$.

12. Justifier que $\lfloor \sqrt{t} \log n \rfloor > 1$. (On raisonnera par l'absurde et on déduira que $n \leq 3$).

13. En déduire $\binom{t+\ell}{t-1} > 2^{\lfloor \sqrt{t} \log n \rfloor + 1}$ (on pourra montrer que $\binom{2u+1}{u} > 2^{u+1}$ pour $u \geq 2$), et enfin $\binom{t+\ell}{t-1} > n^{\sqrt{t}}$.

14. Montrer que n est une puissance de p (utiliser le lemme), puis que $n = p$ (raisonner par l'absurde et observer l'étape 1 de l'algorithme).

On va sommairement établir la complexité de l'algorithme. On admettra les résultats généraux suivants :

- les calculs arithmétiques sur les entiers $\leq N$ sont polynomiaux en $\log N$.
- les calculs arithmétiques sur les polynômes à coefficients entiers $\leq N$ et de degré $\leq d$ sont polynomiaux en $d \log N$.

15. Établir que pour toute entrée n, le nombre d'étapes de l'algorithme est polynomial en $\log n$.

16. En déduire que la complexité de l'algorithme est polynomiale en $\log n$.

1. C'est une conséquence élémentaire de la définition du coefficient binomial :

$$i\binom{n}{i} = \frac{i\ n!}{i!\ (n-i)!} = \frac{n\ (n-1)!}{(i-1)!\ ((n-1)-(i-1))!} = n\binom{n-1}{i-1}.$$

2. C'est une classique conséquence de la question 1. n divise $i\binom{n}{i}$. Si n est premier et si $i \neq 0$ et n, alors n est premier avec i et le lemme de Gauss implique que n divise $\binom{n}{i}$.

3.a. Par hypothèse $\binom{n}{q} = q^k r$. En multipliant par q et en appliquant l'indication de l'énoncé, il vient

$$q^{k+1} r = q\binom{n}{q} = n\binom{n-1}{q-1} = q^k m\binom{n-1}{q-1},$$

d'où $qr = m\binom{n-1}{q-1}$. Ainsi q divise $m\binom{n-1}{q-1}$, et comme q ne divise pas m, à nouveau le lemme de Gauss implique q divise $\binom{n-1}{q-1}$.

3.b. $n-q+1, \ldots, n$ est une séquence de q entiers consécutifs. Elle contient donc exactement un multiple de q. Mais par hypothèse q divise n, donc n est ce multiple de q. La conclusion suit.

3.c. D'après le a, q divise $\binom{n-1}{q-1} = \frac{(n-1)\ldots(n-q+1)}{(q-1)!}$. Donc q divise au moins l'un des facteurs du numérateur (rappelons que q est premier), ce qui contredit b.

Au final, on vient de montrer par l'absurde que q^k ne divise pas $\binom{n}{q}$. Il suit que n, mulitple de q^k, ne divise pas $\binom{n}{q}$ non plus.

4. On développe par la formule du binôme :

$$(X+a)^n = \sum_{i=0}^{n} \binom{n}{i} X^i a^{n-i} = X^n + a^n + \sum_{i=1}^{n-1} \binom{n}{i} X^i a^{n-i}.$$

Si n est premier, on a $a^n = a \bmod n$ et la question 2 montre que $\binom{n}{i}$ est nul mod$n$ pour $i = 1, \ldots, n-1$, ce qui donne le résultat.

Si n n'est pas premier, d'une part il n'est pas sûr que $a^n = a \bmod n$, et d'autre part la question 3 implique que les $\binom{n}{i}$ sont non nuls mod$n$ au moins lorsque i est un facteur premier de n.

5. Supposons que l'algorithme retourne "`composite`". Si c'est à l'étape 1, il est notoire que n est composite. À l'étape 3, cela signifie que l'on a trouvé un facteur non trivial de n. Dans ce cas encore n est composite. Enfin, le test de l'étape 5 est vrai si et seulement si n est composite, d'après la contraposée de la question 4. Dans tous

les cas n est composite. Ainsi, si n est premier, l'algorithme ne peut pas retourner "`composite`".

6. Supposons que l'algorithme retourne "`premier`" à l'étape 4. Cela signifie, d'une part que $n \leq r$, et que l'algorithme a franchi l'étape 3 sans jamais rencontrer de $a \leq \min(n, r) = n$ tel que $1 < \text{pgcd}(a, n) < n$. Il est clair que n ne peut pas être composite, sinon l'étape 3 rencontrerait le plus petit facteur premier de n et provoquerait une sortie de l'algorithme.

7. Par l'absurde, si $\text{pgcd}(n, r) \neq 1$. Alors, ou bien $\text{pgcd}(n, r) < n$, n est composé et l'algorithme le détecte à l'étape 3. Ou bien $\text{pgcd}(n, r) = n$, cela signifie que r est multiple de n, en particulier $r \geq n$ et l'algorithme sort à l'étape 4.

Si p est un diviseur premier de n, il est évident que $\text{pgcd}(p, r) = 1$, sinon p diviserait r et donc diviserait $\text{pgcd}(n, r) = 1$.

8. Écrivons $n = p_1^{e_1} \dots p_s^{e_s}$. Supposons par l'absurde, que $o_r(p_i) = 1$ pour tout i. Cela signifie que $p_i = 1 \bmod r$, puis par multiplication, $n = 1^{e_1} \dots 1^{e_s} = 1 \bmod r$, soit enfin $o_r(n) = 1$. Contradiction avec le lemme.

9. Par l'absurde, supposons que $p \leq r$. Si $n > r$, alors $p \leq r < n$ implique $p < n$, et l'algorithme rencontre p à l'étape 3 et sort. Si $n \leq r$, ou bien $p < n$ et là encore l'algorithme rencontre le facteur p à l'étape 3, ou bien $p = n$ et l'algorithme sort à l'étape 4.

10. $o_r(n)$ divise $|(\mathbb{Z}/r\mathbb{Z})^\times| = \phi(r)$. En particulier $o_r(n) \leq \phi(r)$. Le lemme de l'énoncé établit que $o_r(n) > \log^2 n$, d'où $\log^2 n < \phi(r)$. Enfin

$$\ell = \lfloor \sqrt{\phi(r)} \log n \rfloor < \lfloor \sqrt{\phi(r)} \sqrt{\phi(r)} \rfloor \leq \phi(r).$$

. La dernière inégalité $\phi(r) < r$ est toujours vraie.

11. On a

$$\binom{t+\ell}{t-1} = \binom{t+\ell}{\ell+1} \geq \binom{\lfloor \sqrt{t} \log n \rfloor + 1 + \ell}{\ell+1} = \binom{\lfloor \sqrt{t} \log n \rfloor 1 + \ell}{\lfloor \sqrt{t} \log n \rfloor}.$$

L'inégalité a lieu car $t \geq \lfloor \sqrt{t} \log n \rfloor + 1$ (c'est dit dans l'énoncé), et que $\binom{N}{K}$ est une fonction croissante de N à K fixé.

Le même argument implique

$$\binom{\lfloor \sqrt{t} \log n \rfloor 1 + \ell}{\lfloor \sqrt{t} \log n \rfloor} \geq \binom{2\lfloor \sqrt{t} \log n \rfloor + 1}{\lfloor \sqrt{t} \log n \rfloor},$$

puisque $\phi(r) > t$ et que par conséquent $\ell = \lfloor \sqrt{\phi(r)} \log n \rfloor \geq \lfloor \sqrt{t} \log n \rfloor$.

12. Comme $t > \log^2 n$, $\sqrt{t} \log n > \log^2 n$. Par l'absurde, si $\log^2 n < 2$, on a $n \leq 3$. On peut alors vérifier à la main que pour $n = 2$ ou 3, l'algorithme retourne "`premier`" à l'étape 4. Ceci est en contradiction avec l'hypothèse que l'algorithme retourne "`premier`" à l'étape 6. Ainsi $\sqrt{t} \log n > \log^2 n \geq 2$, et donc $\lfloor \sqrt{t} \log n \rfloor \geq 2$.

13. La propriété entre parenthèses dans l'énoncé est immédiate par récurrence sur u. En remplaçant u par $\lfloor \sqrt{t} \log n \rfloor$ dans la question 11, il vient $\binom{t+\ell}{t-1} > 2^{\lfloor \sqrt{t} \log n \rfloor + 1}$. Enfin, $\lfloor \sqrt{t} \log n \rfloor + 1 \geq \sqrt{t} \log n$ implique $2^{\lfloor \sqrt{t} \log n \rfloor + 1} \geq 2^{\sqrt{t} \log n} = n^{\sqrt{t}}$, ce qui établit la conclusion.

14. On vient de voir que $\binom{t+\ell}{t-1} > n^{\sqrt{t}}$. D'après le lemme, n est une puissance du nombre premier p. Mais si n était une puissance composite de p, l'algorithme le détecterait dès l'étape 1. Ainsi $n = p$ et n est premier.

15. Le nombre d'opérations de l'algorithme est :

— à l'étape 1, au plus $\log n$ calculs sur des entiers ;

— à l'étape 2, au plus $r \leq \lceil \log^5 n \rceil$ opérations sur des entiers ;

— à l'étape 3, idem étape 2 ;
— à l'étape 4, un test sur les entiers ;
— à l'étape 5, $\ell \leq r \leq \lceil \log^5 n \rceil$ opérations sur des polynômes de degré $\leq r - 1 < \lceil \log^5 n \rceil$ à coefficients $< n$.

16. L'algorithme effectue un nombre polynomial en $\log n$ d'opérations, chaque opération étant elle-même dans l'une des deux catégories décrites par l'énoncé, donc polynomiale en $\log n$. La complexité totale de l'algorithme est donc polynomiale $\log n$.