Techniques de cryptographie

Azzouzi Hamza –25614

2021/2022

Introduction

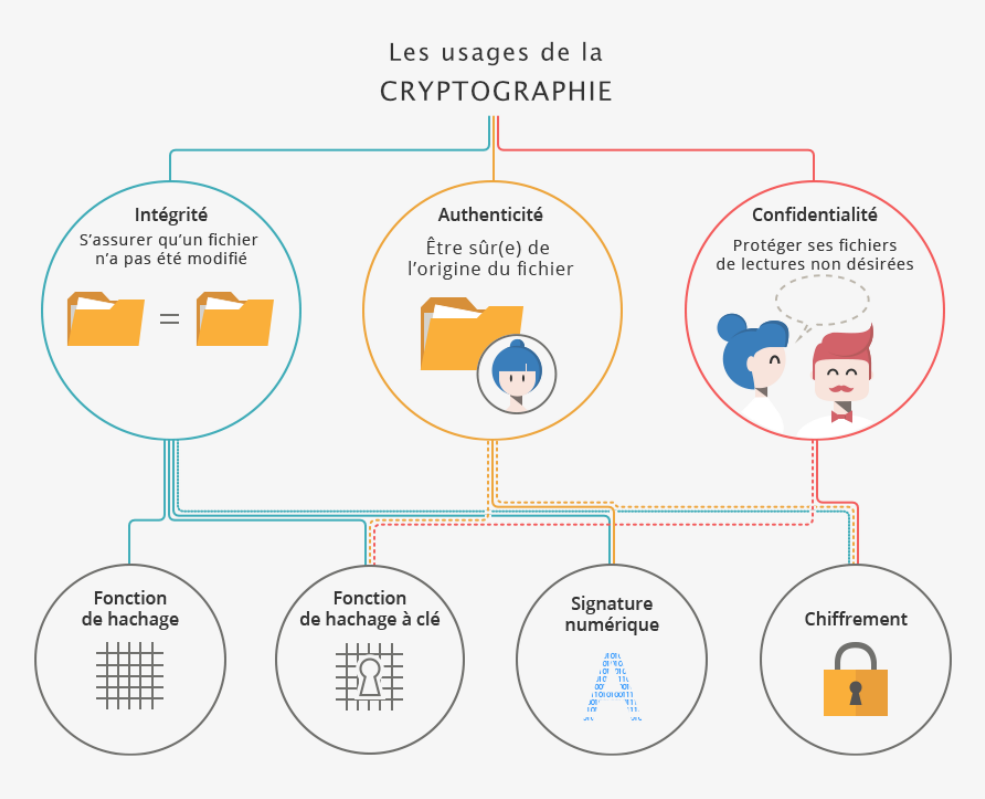
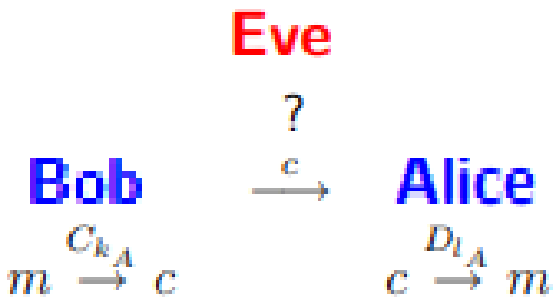


Figure –

Table de matière

Quelques notations



Protocole.

- $\mathcal{M}$ ensemble des messages clairs
- $\mathcal{C}$ ensemble des messages cryptés
- $\mathcal{K}$ ensemble des clés
- $C_k : \mathcal{M} \to \mathcal{C}$ fonction de cryptage
- $D_l : \mathcal{C} \to \mathcal{A}$ fonction de décryptage

Méthode César

- Elle consiste à créer un chiffrage de 3 lettres

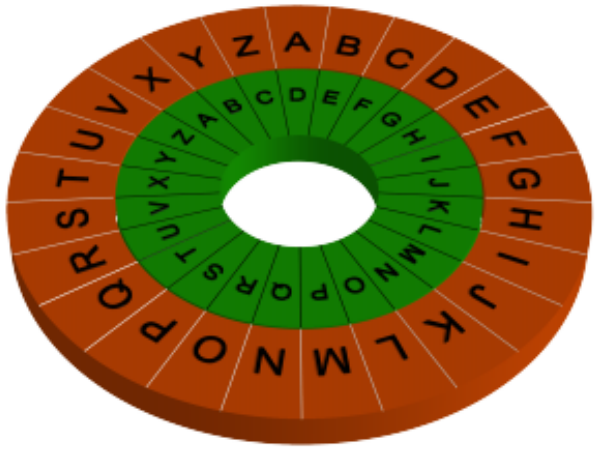


Figure – L'alphabet dans le chiffrage de César

Exemple

SALUT → VDOYX

Au delà de la méthode César

- On peut faire correspondre chaque lettre à une autre (26! possibilités)
- Technique facile à déchiffrer(fréquence des lettres)

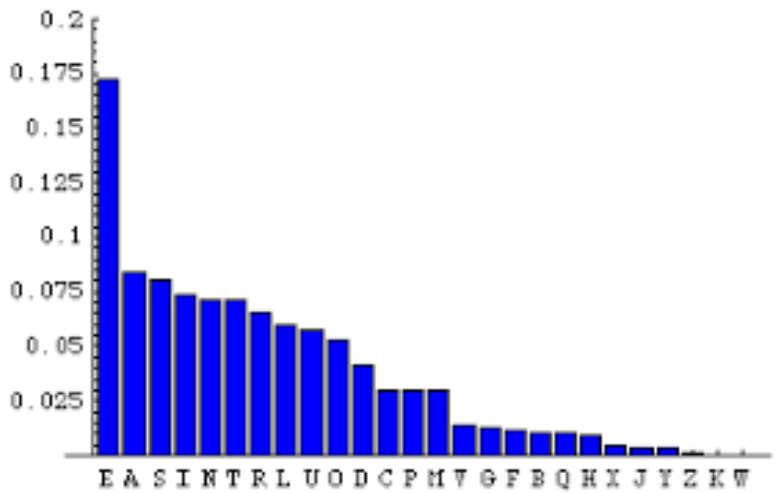


Figure – fréquence des lettres en français

La méthode XOR

Le principe du XOR (ou exclusif) est le suivant : *Pour* x,y $\in \{0,1\}^2$

$$x \oplus y = \begin{cases} 0 & \text{si x=y} \\ 1 & \text{sinon} \end{cases} \quad //$$

On peut alors crypter un message en utilisant comme clé un nombre de bits.

$$C_k(m) = k \oplus m$$

$$D_k(c) = k \oplus c$$

Exemple de la méthode XOR

- En prenant $k = 1011$ et $m = 1010$
- Message chiffré :

$$C_k(m) = 1011 \oplus 1010 = [0 \oplus 1]\,[0 \oplus 1]\,[1 \oplus 1]\,[1 \oplus 0] = 0001$$

- Message déchiffré :

$$D_k(c) = 0001 \oplus 1011 = 1010$$

Principe du cryptage asymétrique

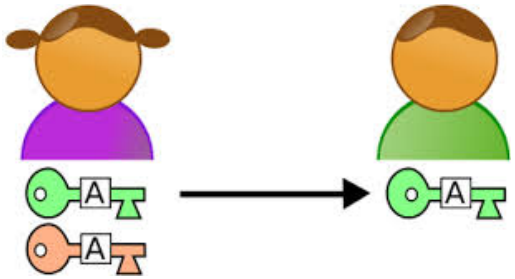


Figure – clé publique et privée

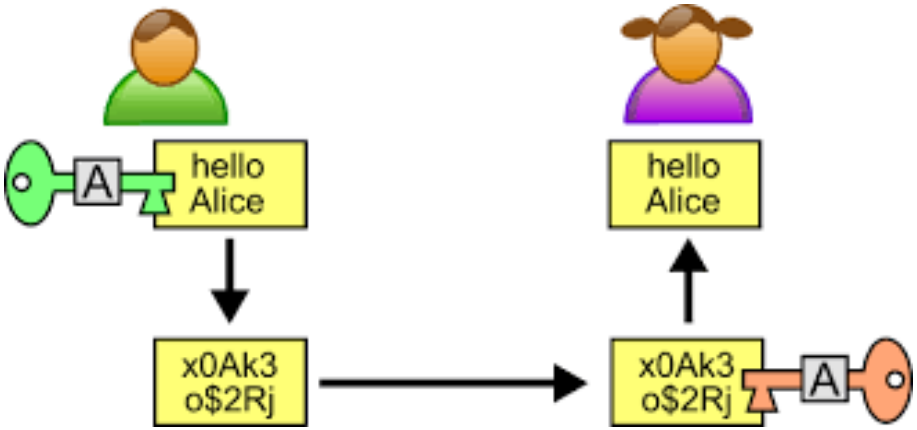


Figure – chiffrage et déchiffrage

Fonction indicatrice d'Euler

- $\forall p \in \mathrm{N}^*$, p est premier ssi $\varphi(p) = p - 1$
- si p et q sont deux entiers premiers entre eux alors : $\varphi(pq) = \varphi(p)\varphi(q)$
- on peut montrer que pour $n \in \mathrm{N}$ fixé :

$$\forall x \in (\mathrm{Z}/n\mathrm{Z})^*, x^{\varphi(n)} \equiv 1\,[n]$$

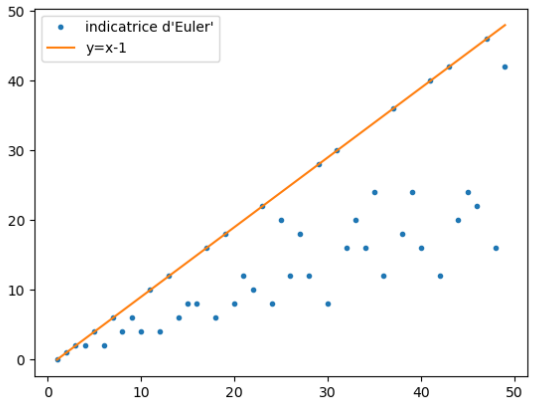


Figure – La fonction indicatrice d'Euler

Principe de La RSA

La fonction indicatrice d'Euler est la base de la cryptographie par la méthode de RSA dont le principe est le suivant :

- On choisit un grand nombre $n = pq$(où p et q sont deux grands nombres premiers), et un nombre r
- L'application de chiffrement est :

$$c = C(m) = m^r \ mod \ n$$

- L'application de déchiffrement est :

$$D(m) = c^s \ mod \ n$$

De sorte que : $rs \equiv 1 \, [\varphi(n)]$

Principe de la RSA

- Le calcul des fonctions E et D se fait rapidement si on connaît r et s grâce à l'algorithme des calcul rapides des puissances.
- Le nombre s n'est connu que par le propriétaire.
- La RSA permet de signer le message puisque le nombre s caractérise la personne qui a chiffrée le message.

Figure – Principe de La RSA

Exemple d'utilisation de RSA

- On utilise $p = 157\, q = 167\, r = 5$
- On trouve alors que $n = 26219$, $\varphi(n) = 25896\, et\, s = 20717$
- Je chiffrerai "SALUT"
- On commence par transformer notre message en code ASCII

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	Start of Header	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	Start of Text	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	End of Text	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	End of Transmission	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	Enquiry	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	Acknowledgment	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	Bell	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	Backspace	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	Horizontal Tab	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	Line feed	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	Vertical Tab	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	Form feed	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	Carriage return	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	Shift Out	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	Shift In	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	Data Link Escape	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	Device Control 1	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	Device Control 2	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	Device Control 3	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	Device Control 4	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	Negative Ack.	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	Synchronous idle	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	End of Trans. Block	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	Cancel	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	End of Medium	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	Substitute	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	Escape	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	File Separator	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	Group Separator	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	Record Separator	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	Unit Separator	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	Del

asciichars.com

Figure – code ASCII

Exemple d'utilisation de RSA

Lettre	S	A	L	U	T
$m^r \ mod \ n$	$83^5 \ modn$	$65^5 \ modn$	$76^5 \ modn$	$85^5 \ modn$	$84^5 \ modn$
Message chiffré	2959	21218	16981	11755	5391

Ainsi le message envoyé serait **"02959 21218 16981 05391"**.

Pour déchiffrer :

Message chiffré	2959	21218	16981	11755	5391
$c^r \ mod \ n$	2959^{20717}	2128^{20717}	16981^{20717}	11755^{20717}	5391^{20717}
Message déchiffré	83	65	76	85	84

On retrouve alors le message envoyé.

Sécurité du RSA

- L'algorithme le plus rapide pour factoriser un nombre se fait en $\mathcal{O}(e^{\sqrt{\ln(n)\ln(\ln(n))}})$
- On suppose que l'ordinateur prend environ une microseconde à chaque opération

Longueur	Nb d'opérations	Durée
75	$9.0.10^{12}$	74 années
200	$1.2\ .10^{23}$	$3.8\ .10^{9}$ années
500	$1.3\ .10^{39}$	$4.2\ .10^{25}$ années

Problème du Logarithme Discret

Proposition

Soit p un nombre premier.On a les propositions suivantes :

- $\mathbb{F}_p^*$ est cyclique
- il existe $g \in \mathbb{F}_p^*$ tq $ord(g) = p - 1$. g est alors appelé racine primitive de l'unité.

Définition

Soit p un nombre premier et soit g une racine primitive de l'unité.

- Pour tout $y \in \mathbb{F}_p^*$ il existe x tq $g^x = y \ mod \ p$.
- On dit alors que x est le logarithme discret de y modulo p et on note $x = \log_g(y)$

Exemple :

On prend ici $n = 11 \ et \ g = 2$

n	1	2	3	4	5	6	7	8	9	10
$\log_2(n)$	0	1	8	2	4	9	7	3	6	5

Cryptosystème d'ElGamal

Choix de la clé privée et publique

- Bob choisit un grand nombre premier p, tel que le problème du logarithme discret est difficile à résoudre dans $\mathbb{F}_p^*$. Il choisit g une racine primitive dans $\mathbb{F}_p^*$
- Bob choisit un nombre d tel que $0 < d < p - 1$ et calcule $b = g^d \ mod \ p$
- Bob alors publie le triplet (p,g,b) qui représentera la clé publique. La clé privée étant le nombre d choisi.

Cryptosystème d'ElGamal

Cryptage du message

Supposons qu'Alice veuille envoyer un message **M** à Bob.

- Elle commence par choisir un nombre k tq $0 < k < p - 1$
- Elle calcule alors $r = g^k$ et $t = b^k M \, mod \, p$
- Alice envoie le couple (r,t) à Bob

Décryptage du message

Pour décrypter le message Bob doit calculer
$tr^{-d} = (b^k M)(g^{-kd}) = (g^{kd} M)(g^{-kd}) = M \, (mod \, p)$

Sécurité des cryptosystèmes

- L'algorithme pour résoudre le problème du logarithme discret le plus rapide se fait $\mathcal{O}(e^{\left(\sqrt{1/2}+o(1)\right)\sqrt{\ln(n)\ln(\ln(n))}})$.
- On suppose que l'ordinateur effectue chaque opération en une microseconde

Longueur du nombre	Nb d'opérations	Durée
100	$3.9x10^6$	3 secondes
200	$9.9x10^9$	2.5 heures
500	$1.3x10^{17}$	3170 années

Test de Primalités

Test de Fermat

Un nombre n n'est pas premier ssi $\exists a \in (\mathbb{Z}/n\mathbb{Z})^*, a^{n-1} \not\equiv 1[n]$

Démonstration.

Le sens directe est la contraposée du théorème de Fermat.
L'autre sens découle du fait que si $a^{n-1} \equiv 1[n]$ alors a est premier avec n □

Test de primalités

Test de Miller Rabin

- Notons $n-1=2^s t$ où t est un nombre impair
- si n n'est pas premier alors il existe $a \in (\mathbb{Z}/n\mathbb{Z})^*$ tq $a^t \not\equiv 1[n]$ ou $a^{2^i t} \not\equiv -1[n]$ avec 0<i<s
- un tel entier est appelé **un témoin de Miller**

Lemme

Si n est un entier composé au moins 3 quarts des nombres entre 2 et n-2 sont des témoins de Miller.

En utilisant ce lemme on peut alors écrire le test probabiliste de Miller.

Test de primalité

Test probabiliste de Miller-Rabin

1. On choisit un nombre t qui représentera le nombre de test à effectuer
2. on choisit aléatoirement un nombre entre 2 et $n-2$
3. si a n'est pas un témoin de Miller on réitère l'opération
4. Si après t test le test n'est pas terminé on retourne *"n est premier avec une probabilité supérieure à* $1-\frac{1}{4^t}$ *"*

Comparaison entre les deux tests

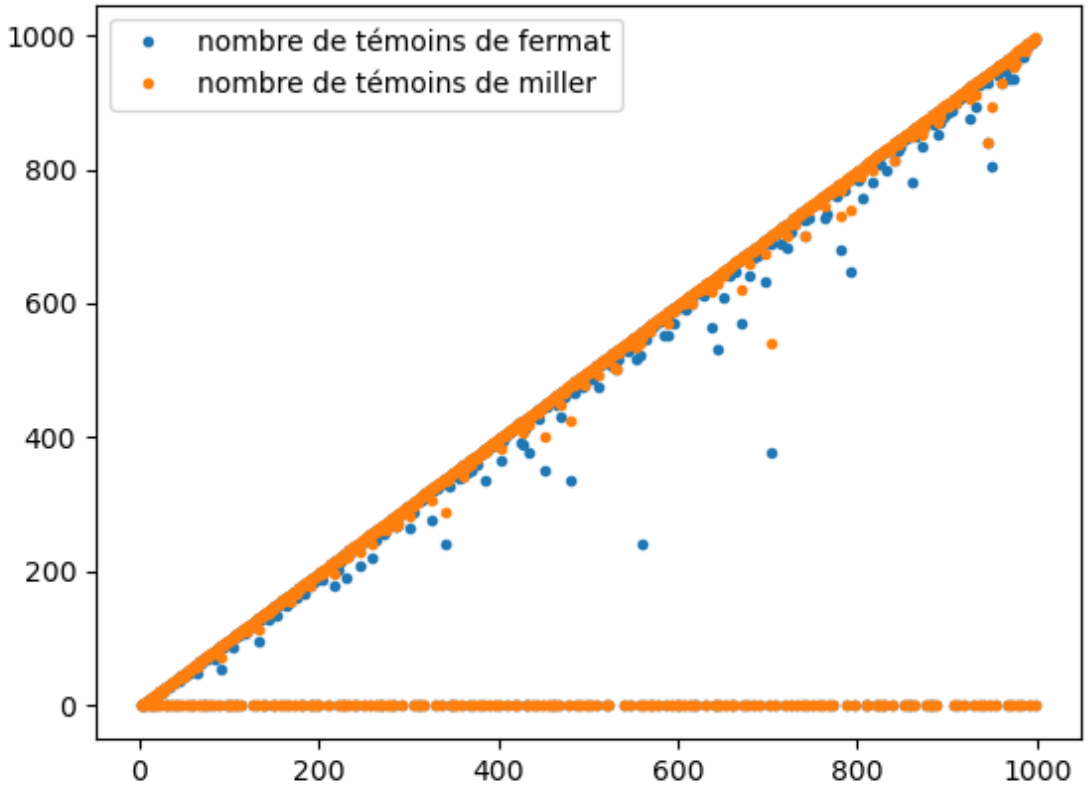


Figure – Comparaison entre les deux méthodes

Test de Lucas-Lehmer

Théorème

Un nombre p $\in$ N est premier ssi il existe a $\in \mathbb{F}_p^*$ d'ordre p-1

Définition

Un nombre de Mersenne est un terme de la suite $M_n = 2^n - 1$

Théorème

Soit la suite $(s_n)_{n\in\mathrm{N}}$ définie par récurrence : $\begin{aligned} s_0 &= 4 \\ s_{n+1} &= (s_n)^2 - 2 \end{aligned}$

Soit p un nombre premier impaire. M_p est premier ssi $M_p | s_{p-2}$

Test de Lucas-Lehmer

Description du test

1. choisir un entier naturel p qui est premier et impair
2. Calculer le nombre de Mersenne M_p
3. Calculer s_{p-2}
4. Evaluer le reste de la division euclidienne de s_{p-2} par M_p. S'il vaut 0 alors M_p est un nombre premier

Merci pour votre attention

Algorithmes

```python
def nombre_de_temoin_fermat(p) :
    acc=0
    for i in range(2,p) :
        if f(i,p-1,p)%p !=1 :
            acc+=1
    return acc

def temoin_Miller_Rabin(a,n) :
    s,t=0,n-1
    while t%2==0 :
        s,t=s+1,t/2
    if f(a,t,n)==1 :
        return False
    for i in range (s) :
        if f(a,(2**i)*t,n)== n-1 :
            return False
    return True

def test_Miller_Rabin(n,t) :
    for _ in range (t) :
        if temoin_Miller_Rabin(random.randint(2,n-2),n) :
            return "{} n'est pas premier".format(n)
    return "{} est premier avec un probabilté supérieure à {}".format(n,1-0.25**t)
```

Figure –

Algorithmes

```
def sontpremier(n,m) :
    d=min(n,m)
    for i in range (2,d+1) :
        if m%i==0 and n%i==0 :
            return False
    return True
def indEuler(n) :
    acc=1
    for i in range(1,n) :
        if sontpremier(i,n) :
            acc=acc+1
    return acc

def nbre_miller(n) :
    acc=0
    for i in range(2,n-1) :
        if temoin_Miller_Rabin(i,n) :
            acc+=1
    return acc

def nbre_fermat(n) :
    acc=0
    for i in range (2,n-1) :
        if temoin_Fermat(i,n) :
            acc+=1
    return acc

def s(n) :
    if n==0 :
        return 4
    return s(n)**2-2
def Mersenne_est_premier(n) :
    return s(n-2)%(2**n-1)==0
```

Figure –