Chapitre 3

Théorie des nombres et cryptographie

3.1 Cryptographie : Généralités

3.1.1 But de la Cryptographie

Le besoin de protection de l'information est aussi ancien que la civilisation elle-même. D'abord exclusivement utilisée par les militaires et les diplomates, elle a acquis avec l'apparition des réseaux informatiques quantité d'applications commerciales. Par exemple, l'intégrité des échanges dans une transaction financière doit être garantie. Clairement, l'utilisateur d'un tel systeme demandera à ce que toute information échangée soit confidentielle et la personne qui fournit un tel système voudra exclure l'utilisation frauduleuse du système.

Une autre inquiétude qui doit être addressée est le problème de l'authentification (comment puis-je être sur que le message a vraiment été écrit par la personne qui prétend en être l'auteur). Dans ce cas, nous chercherons à créer l'équivalent d'une signature.

Le principe de la cryptographie est représenté à la figure 3.1. Lors de l'opération de chiffrage (ou cryptage), le *texte clair* ("plaintext") P est transformé par une fonction E paramétrisée par une *clé* K, pour ainsi obtenir un *cryptogramme* ("ciphertext") $C = E_K(P)$. Ce cryptogramme est alors transmis au receveur, qui applique l'algorithme de décryptage D, inverse de celui de cryptage : $P = D(C) = D(E_K(P))$. On suppose que "l'intrus" écoute et peut reproduire fidèlement le cryptogramme complet. Il ne connaît cependant pas la clé de chiffrage, et ne peut retrouver aisément le message en clair bien que l'on suppose qu'il connaisse l'algorithme utilisé. Cette hypothèse est connue sous le nome de *thèse de Kerckhoffs*. Parfois l'intrus ne se contente pas d'écouter le canal de communication (intrus passif), mais peut altérer les messages ou injecter ses propres messages dans le canal de communication (intrus actif). L'art de composer des cryptogrammes est la *cryptographie*, l'art de les briser est la *cryptanalyse*.

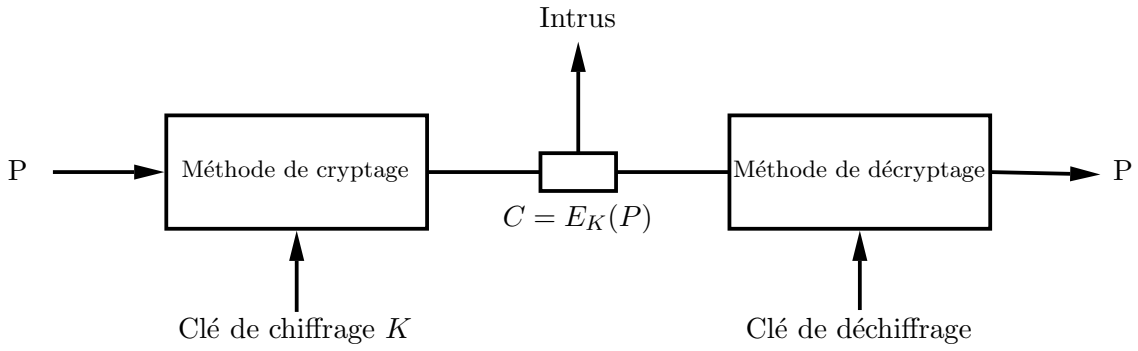

FIG. 3.1 – Modèle de cryptage.

Le problème du cryptanalyste peut être divisé en plusieurs catégories, en fonction de l'information dont il dispose. S'il ne dispose que d'une certaine quantité de cryptogrammes, mais pas du texte clair correspondant, il essaie de compromettre le système cryptographique par une attaque *à texte crypté seul* ("ciphertext-only attack"). Clairement, un chiffrage doit être protégé contre

une telle attaque, puisque nous faisons l'hypothèse qu'il a accès au canal de communication. Si le cryptananlyste peut analyser des paires composées de texte clair et du cryptogramme correspondant, l'attaque devient *à texte clair connu* ("known plaintext attack"). Une telle attaque est possible lorsque l'intrus, à un moment donné, a eu accès à une base de données contenant de telles paires. Comme nous ne pouvons être sûrs que cela ne se produira jamais, notre système doit aussi pouvoir résister à ce genre d'attaque. Enfin, si les circonstances sont tellement favorables (du point de vue du cryptanalyste) que celui-ci peut obtenir le cryptogramme correspondant au text clair de son choix (si, par exemple, il parvient à accéder au mécanisme d'encodage, mais ne peut pas voir la clé) ; ceci s'appelle une attaque *à texte clair choisi* ("chosen plaintext attack").

La justification la plus importante pour l'hypothèse de Kerckhoffs, *c.-à-d.* que nous supposons que l'algorithme de chiffrage est connu, est la suivante. L'histoire a montré qu'un secret ne le reste pas longtemps une fois qu'il est partagé par plusieurs personnes, nous ne pouvons supposer que l'algorithme restera inconnu. Alors que changer d'algorithme chaque fois que le système est compromis serait long et couteux, il n'y a pas de problème à changer fréquemment la clé. Le modèle de base de cryptage comporte donc une méthode générale de chiffrage constante et connue, paramétrisée par une clé secrète et facilement modifiable. Etant donné un cryptosystème, idéalement nous aimerions pouvoir garantir qu'il ne peut être (facilement) brisé. Cette garantie sous sa forme la plus forte est un *chiffrage inconditionnellement sécurisé*. Ce sont des chiffrages qui ne peuvent être brisés, même si l'attaquant à une puissance informatique infinie à sa disposition. Bien que de tels chiffrages existent, ils ne sont pas souvent utilisés. Ensuite viennent les chiffrages inviolables en pratique en raison du fait qu'avec les ordinateurs actuels et les algorithmes connus, il n'est pas possible de les briser en un temps raisonnable.

3.2 Algorithmes de Chiffrage Symétriques

Nous allons analyser une première catégorie d'algorithmes de chiffrage, pour laquelle la même clé secrète est utilisée au cryptage et au décryptage, d'où le nom d'algorithmes symétriques. Ces algorithmes demandent à ce que la clé soit transmise au receveur sur un canal sécurisé (voir figure 3.2). Nous étudierons plus tard les algorithmes asymétriques, qui n'utilisent pas la même clé au cryptage et au décryptage.

3.2.1 Les Cryptages de Substitution

Jules César utilisait un procédé de chiffrage qui consistait en une rotation de toutes les lettres de l'alphabet de trois positions. Ainsi, a devenait d, b devenait e, ...et z devenait c. On peut généraliser le cryptage de César pour permettre une rotation de k lettres, au lieu de prendre toujours 3 lettres. Dans ce cas k est la clé de la méthode générale de chiffrage qui consiste en une rotation de l'alphabet.

Exemple 1. *Dans le film* Odyssée dans l'Espace 2001*, le nom de l'ordinateur est HAL, et c'est en fait un cryptogramme. Quel est le text clair correspondant ? Et quelle est la clé k ?*

Evidemment, le cryptage de rotation n'est pas très sécurisé. Puisque, par hypothèse (de Kerck-

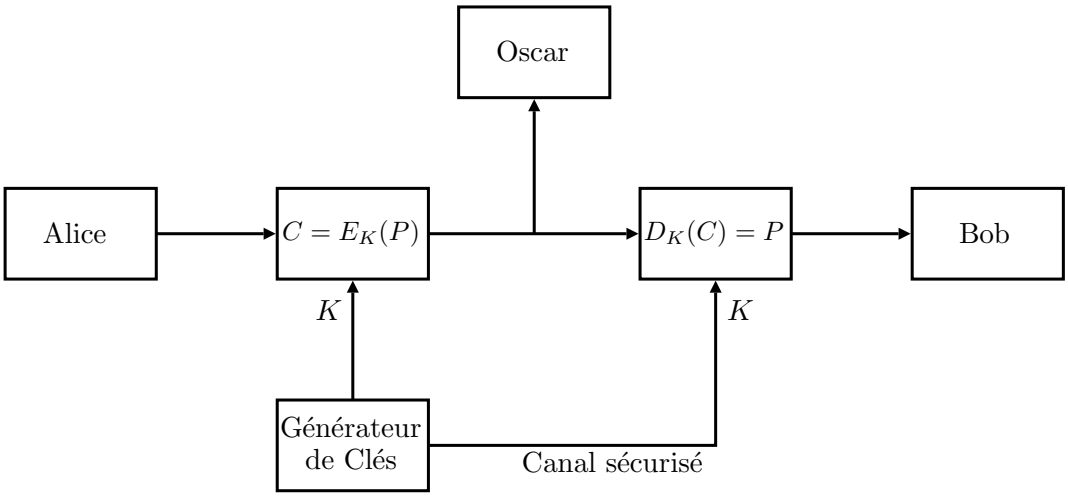

FIG. 3.2 – Algorithme de chiffrage symétrique.

hoffs), le cryptanalyste sait que nous utilisons un cryptage de rotation, il n'y a que 26 clés possibles et il est trivial de toutes les essayer.

Une généralisation plus sécurisée est de remplacer chaque lettre de l'alphabet du texte clair par une autre lettre, sans respecter une relation de rotation entre les deux alphabets. Ce système général est appelé *substitution monoalphabétique*, la clé étant le tableau de correspondance entre les alphabets du texte en clair et du texte crypté. Par exemple, on pourrait prendre la clé

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

Pour un alphabet de D caractères, il y a $D!$ clés possibles. Puisque $26! \sim 10^{26}$, un très grand nombre, ce qui semble indiquer que cet algorithme soit relativement sécurisé. Malheureusement, ce n'est pas tout à fait le cas.

Néanmoins, le cryptanalyste se facilite grandement la tâche en se basant sur la distribution des fréquences des lettres dans le cryptogramme. En anglais par exemple, la répartition des lettres

est montrée dans le tableau suivant.

lettre	frequence[%]	lettre	frequence[%]
E	13	M	2
T	9	W	2
A	8	F	2
O	8	G	2
I	7	Y	2
N	7	P	2
S	6	B	2
H	6	V	1
R	6	K	1
D	4	J	0
L	4	X	0
C	3	Q	0
U	3	Z	0

Nous voyons que les lettres les plus fréquentes sont donc e et t, que le cryptanalyste peut alors essayer d'assigner aux deux lettres les plus fréquentes dans le cryptogramme. Il trouvera vraisemblalement alors beaucoup de triplets de la forme tXe, suggérant fortement que X correspond à h. En procédant de la sorte, il peut retrouver, lettre par lettre, la clé de cryptage assez rapidement (du moins avec un ordinateur).

De plus, il peut parfois deviner certains mots selon le contexte. Par exemple, s'il s'agit d'une transaction boursière, le message risque de comporter les mots "action" ou "cours"...

Pour compliquer la tâche du cryptanalyste, il faut donc "cacher" la distribution de fréquences des lettres, de telle sorte que des lettres comme e, a, t ne soient pas si faciles à repérer. Une manière de procéder est d'introduire plusieurs alphabets cycliques, pour obtenir un cryptage de Vigenère, qui est un exemple de cryptage utilisant la *substitution polyalphabétique* :

alphabet d'origine	*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*
rangée A	*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*
rangée B	*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*	*A*
rangée C	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*
rangée D	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*
rangée E	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*
rangée F	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*
rangée G	*G*	*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*
rangée H	*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*
rangée I	*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*	*H*
rangée J	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
rangée K	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*
rangée L	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*
rangée M	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*
rangée N	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*
rangée O	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*
rangée P	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*
rangée Q	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*
rangée R	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*
rangée S	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*
rangée T	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*
rangée U	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*
rangée V	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*
rangée W	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*
rangée X	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*
rangée Y	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*
rangée Z	*Z*	*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*

Comme un cryptage monoalphabétique, ce cryptage polyalphabétique a aussi une clé, qui est d'habitude un mot court, facile à mémoriser, comme BONJOUR (au lieu du tableau complet des 26 lettres dans le cas monoalphabétique). La clé est répétée constamment sous le texte en clair, et indique quelle rangée de la table précédente doit être utilisée pour le cryptage. Dans l'exemple suivant, *l* est crypté avec l'alphabet de la rangée B et devient donc *M*, et ainsi de suite :

texte clair :	l	a	c	r	y	p	t	o	g	r	a	p	h	i	e
clé :	B	O	N	J	O	U	R	B	O	N	J	O	U	R	B
texte crypté :	M	O	P	A	M	J	K	P	U	E	J	D	B	Z	F

Bien qu'il soit nettement plus sûr qu'un cryptage de substitution monoalphabétique, un cryptage de substitution polyalphabétique peut encore être brisé par une attaque à texte crypté seul. L'astuce consiste à deviner la longueur de la clé.

Un exemple célèbre de méthode de cryptage basée sur la substitution polyalphabétique est la machine ENIGMA utilisée par les forces de l'axe (Allemagne, Japon et leurs alliés) pendant la seconde guerre mondiale. La substitution s'opérait grâce à trois rotors ayant chacun 26 positions. Les positions initiales des rotors étaient encodées sur l'en-tête du message, ce qui était une faiblesse exploitée par les cryptanalystes britanniques, français et polonais. (Voir *http ://www.bletchleypark.org.uk/* pour de plus amples details.)

3.2.2 Les Cryptages de Transposition

Un *cryptage de transposition* réordonne les lettres du message mais ne les "déguise" pas.

Exemple 2. *Un premier exemple d'un tel cryptage est le suivant :*

texte en clair : *pleasetransferonemilliondollarsto*
myswissbankaccountsixtwotwo

clé :

M	*E*	*G*	*A*	*B*	*U*	*C*	*K*
p	*l*	*e*	*a*	*s*	*e*	*t*	*r*
a	*n*	*s*	*f*	*e*	*r*	*o*	*n*
e	*m*	*i*	*l*	*l*	*i*	*o*	*n*
d	*o*	*l*	*l*	*a*	*r*	*s*	*t*
o	*m*	*y*	*s*	*w*	*i*	*s*	*s*
b	*a*	*n*	*k*	*a*	*c*	*c*	*o*
u	*n*	*t*	*s*	*i*	*x*	*t*	*w*
o	*t*	*w*	*o*	*a*	*b*	*c*	*d*

texte crypté : *AFLLSKSOSELAWAIATOOSSCTCLNMOMANT*
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

Ce cryptage est paramétrisé par une clé qui est un mot ou une phrase dans laquelle chaque lettre n'apparaît qu'une fois. Le texte clair est écrit sous forme de rangées successives, de même longueur que la clé (le message est complété par des lettres quelconques si nécessaire). Le texte crypté est lu colonne par colonne, dans l'ordre alphabétique des lettres formant la clé.

Même si le cyptanalyste ne sait pas qu'il est en présence d'un cryptogramme de transposition, ce n'est pas très dur de s'en rendre compte : la freéquence de répartition des lettres est identique dans le texte clair et dans le cryptogramme. Ensuite, il doit deviner la longueur de la clé utilisée. Enfin, dans de nombreux cas, certains mots peuvent être devinés à partir du contexte (cf. l'exemple ci-dessus).

Un autre exemple de cryptage de transposition est le celui de *permutation* qui découpe le texte en clair en blocs de longueur d, où d est la longueur de la clé. Les d caractères sont alors permutés suivant l'ordre alphabétique des lettres formant la clé.

Exemple 3. *Dans le cas de l'exemple précédent ($d = 8$), on a donc*

clé :	*MEGABUCK*			
texte en clair :	*pleasetr* *omyswiss*	*ansferon* *bankacco*	*emillion* *untsixtw*	*dollarst* *otwo*
texte crypté :	*ASTLERPE* *SWSMYSOI*	*FEONSNAR* *KACANOBC*	*LLOMINEI* *SITNTWUX*	*LASOLTDR* *OTWO*

Même si la longueur d de la clé est connue du cryptanalyste, celui-ci doit essayer les $d!$ permutations possibles des d caractères formant la clé.

3.2.3 Entropie d'une Langue

Evaluer la sécurité des cryptosystèmes nécessite l'introduction d'une nouvelle notion, celle "d'entropie" d'une langue. Considèrons une page de texte écrite en langue anglaise. Combien de pages grammaticalement correctes peut-on écrire ? Appelons $\mathcal{E}$, l'ensemble contenant toutes ces pages et $|\mathcal{E}|$, la cardinalité de cet ensemble. L'ensemble contenant toues les pages possibles (qu'elles soient grammaticalement correctes ou pas) est appelé $\mathcal{P}$ et a pour cardinalité $|\mathcal{P}|$.

Si l'on considère à présent deux pages. Combien de telles paires différentes avons-nous ? La réponse est clairement $|\mathcal{P}|^2$. Combien parmis elles seront grammaticalement correctes ? Nous nous attendons à en avoir à peu près $|\mathcal{E}|^2$. En fait, nous en aurons un peu plus à cause de ce qu'on pourrait appeler des effets de bords (il nous est maintenant possible d'avoir des phrases qui commencent sur la première page et finissent sur la deuxième).

Dans le cas général de n pages, il y a $|\mathcal{P}|^n$ pages possibles et à peu près $|\mathcal{E}|^n$ d'entre elles sont grammaticalement correctes.

Soit $N(n)$ le nombre de documents de n pages qui soient grammaticalement corrects. Definisson $M(n) = \log_2 N(n)$.

Définition 1. *Soit H_L le nombre réel*

$$H_L \doteq \lim_{n\to\infty} \frac{1}{n} M(n).$$

Nous appelons H_L l'entropie de la langue (qui est ici mesurée par page). L'entropie est tout simplement une facon de décrire le nombre de pages grammaticalement correctes.

Cela signifie que le nombre de documents de n pages grammaticalement corrects est à peu près 2^{nH_L} parmis les $2^{n \log_2(|\mathcal{P}|)}$ possibles.

Supposons maintenant que nous parlions de caractères au lieu de pages. L'alphabet latin comprend 26 caractères. Le nombre de chaînes de n caractères est donc $2^{n \log_2(26)}$. Rappelons nous que d'après ce qui a été vu au chapitre 2, l'entropie par caractère représente la quantité d'information contenue dans un caracteère. Si l'on considère une langue qui contiendrait toutes les

chaines de caractères possibles, il y aurait $2^{n \log_2(26)}$ chaines de longueur n qui seraient grammaticalement correctes. Ce qui confère a cette langue l'entropie par caractère maximale qu'une langue de 26 caractères peut avoir et qui est de $\log_2(26) = 4,7$.

Prenons maintenant une langue qui n'autorise que les chaînes composées du caractere a. Pour chaque longueur n, il n'y a qu'une unique chaîne possible, ce qui nous donne une entropie de 0. Aucune information n'étant contenue dans un caractère de telles chaînes car on sait à l'avance que ce sera un a. L'entropie d'une langue utilisant l'alphabet $\mathcal{C}$ contenant $|\mathcal{C}|$ caractères est donc comprise entre 0 et $\log_2(|\mathcal{C}|)$. Plus celle-ci sera élevée, plus le nombre de chaînes grammaticalement correctes sera élevé.

Exemple 4. *La langue anglaise a une entropie par caractère d'à peu près* $H_{English} = 1.5$. *Ce qui signifie qu'il y a aux alentours de* $2^{1.5n}$ *chaînes de* n *caractères correctes. D'un autre point de vue, si l'on considère les chaînes de* 100 *caratères. Il y en a* $26^{100} = 10^{141}$, *parmis lesquelles seules* $2^{150} = 10^{45}$ *sont grammaticalement correctes.*

Supposons maintenant que nous ayons à disposition un cryptogramme C et que la clé du cryptosystème appartienne à un ensemble $\mathcal{K}$. Le nombre de clés possibles est $2^{H(\mathcal{K})}$ avec $H(\mathcal{K}) = \log_2(|\mathcal{K}|)$. Une facon de briser le code consiste à essayer toutes les clés possibles jusqu'à ce que le texte décrypté ait l'air correct (ait un sens). Le message étant écrit dans l'alphabet $\mathcal{C}$ et dans la langue L, combien de textes décryptés auront un sens ?

Supposons que pour chaque clé incorrecte le texte décrypté est uniformément distribué parmis les $|\mathcal{C}|^n$ possibles. Dans ce cas la, la probabilité de choisir une clé résultant en un texte grammaticalement correct est

$$\frac{\#\text{ de textes corrects}}{\#\text{ de textes possibles}} = \frac{2^{nH_L}}{2^{n \log_2(|\mathcal{C}|)}}.$$

Cela nous donnes que le nombre moyen de décryptage résultant en un texte grammaticalement correct est

$$\frac{2^{H(\mathcal{K})}\ 2^{nH_L}}{2^{n \log_2(|\mathcal{C}|)}} = 2^{n(\frac{1}{n}H(\mathcal{K}) + H_L - \log_2(|\mathcal{C}|))}$$

Si ce nombre est proche de 1, nous nous attendons à ne trouver qu'un seul décryptage donnant un texte correct. Cela signifie qu'en principe, il serait possible de briser le code. Dans le cas contraire, s'il y a plusieurs décryptages donnant des textes grammaticalement corrects, il n'est pas possible de reconnaitre le bon.

Le nombre de décryptage résultant en des textes grammaticalement corrects est proche de 1 si nous avons

$$\frac{1}{n}H(\mathcal{K}) + H_L - \log_2(|\mathcal{C}|) = 0.$$

Nous pouvons maintenant définir le concept de *distance d'unicité*. C'est la longueur du texte clair, pour laquelle le nombre de décryptages résultant en un texte correct, est égal à un.

$$n = \frac{H(\mathcal{K})}{\log_2(|\mathcal{C}|) - H_L}.$$

Exemple 5. *Dans le cas d'un cryptage par substitution et d'un texte clair en anglais, nous avons*

$$\begin{aligned} \log_2 |\mathcal{K}| &= \log_2(26!) = 88 \\ H_{English} &= 1.5 \\ \log_2(|\mathcal{C}|) &= \log_2(26) = 4.7 \\ n &= \frac{H(\mathcal{K})}{\log_2(|\mathcal{C}|) - H_L} = \frac{88}{4.7-1.5} = 28. \end{aligned}$$

Cela signifie qu'un texte clair de 28 caractères crypté par substitution ne pourra être décrypté correctement que par une unique clé.

3.3 Eléments de Théorie des Nombres

Nous allons passer à la cryptographie à clé publique dans la prochaine section, mais avant cela, nous voyons quelques notions de théorie des nombres qui seront néssaires par la suite.

3.3.1 Opérations sur les Entiers

On désigne par $\mathbb{Z}$ l'ensemble des entiers (positifs et négatifs). Tous les cours de calcul à l'école primaire ont débuté par les opérations fondamentales qui peuvent être effectuées sur les entiers : l'addition et la multiplication. Rappelons les propriétés basiques de ces opérations. Si a, b et c sont des éléments de $\mathbb{Z}$, l'addition a les propriétés suivantes :
– Fermeture : $a + b \in \mathbb{Z}$
– Associativité : $a + (b + c) = (a + b) + c$
– 0 est l'élément neutre : $a + 0 = a$
– $(-a)$ est l'opposé de a : $a + (-a) = 0$
– Commutativité : $a + b = b + a$
Un ensemble (dans ce cas $\mathbb{Z}$), muni d'une opération binaire (ici $+$) qui à les propriétés précédentes (fermeture, associativité, élement neutre, inverse, commutativité) est appelé un *groupe abélien* ou *groupe commutatif*. La multiplication, quant à elle, présente les propriétés suivantes :
– Fermeture : $a \cdot b \in \mathbb{Z}$
– Associativité : $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
– 1 est l'élément neutre : $a \cdot 1 = a$
– Commutativité : $a \cdot b = b \cdot a$
Malheureusement, seuls 1 et -1 ont un inverse pour la multiplication dans $\mathbb{Z}$. Pour cette raison, $\mathbb{Z}$ muni de la multiplication n'est pas un groupe abélien, mais seulement un *semi-groupe commutatif*

De plus, comme la multiplication est distributive par rapport à l'addition, à savoir

$$\begin{aligned} a \cdot (b + c) &= a \cdot b + a \cdot c \\ (a + b) \cdot c &= a \cdot c + b \cdot c \end{aligned}$$

Un ensemble (dans ce cas $\mathbb{Z}$), muni de deux opérations binaires (ici $+$ et $\cdot$) qui forment un groupe abélien et un semi-groupe abélien et qui ont la propriété de distributivité est appelé un *anneau commutatif* ("commutative ring").

L'opération de soustraction $a-b$ est simplement l'opération d'addition $a+(-b)$, où b est remplacé par son opposé. L'opération de division par contre n'est pas toujours définie, car l'inverse de a n'est pas un entier (sauf si $a = \pm 1$). Néanmoins, si $a = bc$ pour deux entiers b et c, on dit que b (ou c) divise a, ou encore que b (ou c) est un diviseur de a, et on écrit $b|a$ (ou $c|a$). Un *nombre premier* est un entier qui n'admet que 1 et lui-même comme diviseurs.

Si b ne divise pas a, on peut néanmoins définir la division comme suit.

3.3.2 Division Euclidienne

Soient a et b des entiers. Si b est non nul, il existe deux entiers, le *quotient* q et le *reste* r, tel que

$$a = bq + r \quad \text{et} \quad 0 \leq r < |b|. \tag{3.1}$$

Le calcul de q et r s'appelle la *division euclidienne* de a par b.

3.3.3 Plus Grand Diviseur Commun (PGDC)

Soient a et b deux entiers. Les entiers positifs qui divisent à la fois a et b forment donc un ensemble fini non vide, puisque 1 divise tous les entiers. Par conséquent, cet ensemble possède un élément plus grand ou égal à tous les autres : le *plus grand commun diviseur* de a et b, noté $\text{pgdc}(a, b)$. Si $\text{pgcd}(a, b) = 1$, a et b sont *relativement premiers entre eux*.

Les propriétés basiques du PGDC sont :

1. $\text{pgdc}(a, b) = \text{pgdc}(b, a)$, par symétrie de la définition,
2. $\text{pgdc}(a, 0) = a$, puisque a est le plus grand diviseur de lui-même et tous les entiers divisent 0.
3. $\text{pgdc}(a, b) = \text{pgdc}(a, b + ca)$ pour tout $c \in \mathbb{Z}$.

Cela nous donne un moyen efficace de calculer $\text{pgdc}(a, b)$ par un choix judicieux de c dans la 3ème propriété. Supposons, sans perte de généralité (à cause de la première propriété), que $|b| \geq |a|$. De la section 3.3.2, nous savons que nous pouvons écrire $b = aq + r$, où $0 \leq r < |b|$ ce qui donne $r = b - qa$, nous permettant d'ecrire $\text{pgdc}(a, b) = \text{pgdc}(a, b - qa) = \text{pgdc}(a, r)$. En remplaçant successivement le plus grand terme par le reste de la division euclidienne, nous réduirons finalement notre expression à $\text{pgdc}(a, b) = \text{pgdc}(r_n, r_{n-1}) = \text{pgdc}(r_n, 0) = r_n$:

$$\begin{array}{rcll}
b & = & aq_1 + r_1 & \text{pgdc}(a, b) = \text{pgdc}(a, b - aq_1) \quad |a| < |b| \\
 & & & = \text{pgdc}(a, r_1) \quad r_1 < |a| \\
a & = & r_1 q_2 + r_2 & = \text{pgdc}(a - r_1 q_2, r_1) \\
 & & & = \text{pgdc}(r_2, r_1) \quad r_2 < r_1 \\
r_1 & = & r_2 q_3 + r_3 & = \text{pgdc}(r_2, r_3) \quad r_3 < r_2 \\
 & \vdots & & \vdots \quad \vdots \\
r_{n-2} & = & r_{n-1} q_n + r_n & = \text{pgdc}(r_n, r_{n-1}) \quad r_n < r_{n-1} \\
r_{n-1} & = & r_n q_{n+1} + 0 & = \text{pgdc}(r_n, 0) \quad 0 = r_{n+1} < r_n \\
 & & & = r_n.
\end{array}$$

Nous savons que cet algorithme terminera car r_n est une série positive décroissante : $|b| > |a| > r_1 > r_2 > \ldots > r_n \geq 0$.

3.3.4 Algorithme d'Euclide Etendu

Théorème 1. *Quelque soient les entiers a and b, il existe deux entiers α et β tels qu*

$$\alpha a + \beta b = pgdc(a, b). \tag{3.2}$$

.

Ceci est connu sous le nom *d'identité de Bézout* et est utile, en particulier, pour résoudre les *equations Diophantines* (equations aux coefficients entiers dont les solutions sont des entiers).

Afin de trouver ces valeurs, nous pouvons étendre l'algorithme précédent, d'où le nome *algorithme d'Euclide étendu*. Remontant en arrière, nous pouvons commencer avec $\text{pgdc}(a,b) = r_n$ et substituter chaque r_N avec $r_{N-2} - r_{N-1}q_N$, pour finalement r_1 par $b - aq_1$ afin qu'il ne reste plus qu'une *combinaison linéaire* de a et b.

$\text{pgdc}(a,b)$	$=$	r_n	
	$=$	$r_{n-2} - r_{n-1}q_n$	r_n eliminé
	$=$	$r_{n-2} - (r_{n-3} - r_{n-2}q_{n-1})q_n$	r_{n-1} eliminé
	$=$	$(-q_n)r_{n-3} + (1 + q_{n-1}q_n)r_{n-2}$	groupement de termes semblables
	$=$	$r_{n-3}\rho_a + r_{n-2}\rho_b$	remplacé les termes complexes par des coefficients
	$=$	$r_{n-4}\pi_a + r_{n-3}\pi_b$	r_{n-2} eliminé
	$\vdots$		
	$=$	$\delta_a r_1 + \delta_b r_2$	
	$=$	$\gamma_a a + \gamma_b r_1$	r_2 eliminé
	$=$	$\alpha a + \beta b$	r_1 eliminé.

Comme nous pouvons voir, chaque reste r_N peut être exprimé comme une combinaison linéaire de a and b. Ceci est vrai en particulier pour $r_n = \text{pgdc}(a,b)$.

Exemple 6. *Utilisons cette méthode pour calculer les valeurs manquantes de l'identité de Bézout suivante :* $\alpha 120 + \beta 23 = pgdc(120, 23)$.

a	b	q_n	r_n
120	23	5	$5 = \quad 1 \cdot 120 - 5 \quad \cdot 23$
5	23	4	$3 = -4 \cdot 120 + 21 \cdot 23$
5	3	1	$2 = \quad 5 \cdot 120 - 26 \cdot 23$
2	3	1	$1 = -9 \cdot 120 + 47 \cdot 23$

Ce qui montre que $-9 \cdot 120 + 47 \cdot 23 = pgdc(120, 23) = 1$.

3.3.5 Congruences

Soient a et b deux entiers. On dit que a est *congru à* b *modulo* m, et on écrit

$$a \equiv b \pmod{m} \tag{3.3}$$

si et seulement si m divise $a - b$. L'expression (3.3) s'appelle *congruence* et m est son *module*.

La congruence (3.3) implique donc que

$$a = b + xm \tag{3.4}$$

pour un certain entier x.

Enfin, si r est le reste de la division euclidienne de a par m, on a

$$a \equiv r \pmod{m}$$

et r s'appelle alors le *résidu* de a modulo m. On peut montrer aisément que $a \equiv b \pmod{m}$ si et seulement si leurs résidus modulo m sont les mêmes.

De même, le module m étant fixé, on vérifie que la relation entre a et b définie par (3.3) est une relation d'équivalence sur $\mathbb{Z}$, car elle est
– réflexive : $a \equiv a \pmod{m}$,
– symétrique : si $a \equiv b \pmod{m}$ alors $b \equiv a \pmod{m}$,
– et transitive : si $a \equiv b \pmod{m}$ et $b \equiv c \pmod{m}$ alors $a \equiv c \pmod{m}$.
Par conséquent, cette relation définit des *classes d'équivalence*, obtenues à partir d'un entier quelconque a en lui ajoutant tous les multiples de m. On note ces classes d'équivalence $|a|_m$ et on appelle a un représentant de $|a|_m$. On vérifie facilement que chaque classe d'équivalence contient un unique entier r compris entre 0 et m : $0 \leq r < m$, il est le résidu de tous les éléments de la classe $|r|_m$. Il y a donc m classes d'équivalence, ce sont $|0|_m, |1|_m, \ldots, |m-1|_m$. L'ensemble de ces m classes d'équivalence, $\mathbb{Z}_m$, est l'ensemble des *entiers modulo* m.

3.3.6 Opérations modulo m

Considérons deux classes modulo m ; a est un représentant quelconque de la première et b est un représentant quelconque de la seconde. La somme des deux classes est la classe de $a+b$ et le produit est la classe de ab :

$$|a|_m + |b|_m = |a+b|_m \qquad \text{et} \qquad |a|_m \cdot |b|_m = |ab|_m.$$

La propriété suivante nous assure que les classes obtenues ne dépendent pas des représentants a et b choisis.

Si $a \equiv a' \pmod{m}$ et $b \equiv b' \pmod{m}$, alors

$$a + b \equiv a' + b' \pmod{m} \tag{3.5}$$
$$ab \equiv a'b' \pmod{m} \tag{3.6}$$

En effet, il existe deux entiers x_1 et x_2 tels que $a = a' + x_1 m$ et $b = b' + x_2 m$, d'où $(a+b) = (a'+b') + (x_1+x_2)m$ et $(ab) = (a'b') + (a'x_2 + b'x_1 + x_1x_2m)m$, ce qui prouve la propriété.

Par la suite, on prendra comme représentants des classes leurs résidus modulo m.

Exemple 7. *Les tables d'addition et de multiplication modulo 3 sont donc*

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\times$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

tandis que les tables d'addition et de multiplication modulo 4 sont

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

×	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

On vérifie aisément que l'addition modulo m est interne, associative, possède un élément neutre ($|0|_m$), un opposé pour chaque classe $|a|_m$ ($|-a|_m$) et est commutative. D'autre part, la multiplication modulo m est interne, associative, possède un élément neutre ($|1|_m$) et est commutative. Par conséquent, $\mathbb{Z}_m$ est un anneau commutatif.

La multiplication n'admet pas toujours un inverse. En effet, si par exemple $m = 4$, $|2|_4$ n'a pas d'inverse : aucune des classes $|1|_4$, $|2|_4$ ou $|3|_4$, multipliée par $|2|_4$ ne donne $|1|_4$ comme résultat. Par contre, si $m = 3$, on voit que $|1|_3 \cdot |1|_3 = |1|_3$ tandis que $|2|_3 \cdot |2|_3 = |1|_3$. Par conséquent, chaque élément non nul a un inverse pour la multiplication, et $\mathbb{Z}_3$ est un champ.

Nous prétendons que a possède un inverse pour la multiplication modulo m si et seulement si $\text{pgdc}(a, m) = 1$. Montrons d'abord que a possède un inverse pour la multiplication si $\text{pgdc}(a, m) = 1$. Rapellons que d'après l'identité de Bézout's (3.2) nous avons :

$$\begin{aligned} \alpha a + \mu m &= \text{pgdc}(a, m) = 1 \\ \alpha a &\equiv 1 \pmod{m} \\ \alpha &= a^{-1}. \end{aligned}$$

Supposons maintenant, réciproquement que a possède un inverse modulo m, *c.à-d.*

$$\begin{aligned} a \cdot a^{-1} &\equiv 1 \pmod{m} \\ a \cdot a^{-1} + bm &= 1 \qquad \text{for some } b \in \mathbb{Z}. \end{aligned} \tag{3.7}$$

puisque $\text{pgdc}(a, m)$ divise à la fois a and m, il divise aussi n'importe quelle combinaison linéaire de a and m. 1 doit donc être divisible par $\text{pgdc}(a, m)$ (3.7), impliquant que $\text{pgdc}(a, m) = 1$, cqfd. Lorsque m est premier, à savoir que $\text{pgdc}(a, m) = 1$ pour tout a tel que $0 < a < m$, tous les élements non-nuls de $\mathbb{Z}_m$ ont un inverse. $\mathbb{Z}_m$ est donc un champ si et seulement si m est premier. On appelle alors $\mathbb{Z}_m$ un *champ de Galois*, que l'on note $\mathcal{CG}(m)$.

Une opération particulièrement facile à réaliser est l'*exponentiation modulo* m, à savoir

$$x \equiv a^n \pmod{m}.$$

En effet, calculons par exemple le résidu de 3^{12} modulo 7. On réduit successivement la base modulo 7, de la façon suivante :

$$|3^{12}|_7 = (|3^2|_7)^6 = (|2|_7)^6 = |64|_7 = |1|_7$$

et donc

$$3^{12} \equiv 1 \pmod{7}.$$

Par contre, le calcul du *logarithme discret* d'un nombre x modulo m, à savoir la détermination de x tel que

$$a^x \equiv b \pmod{m},$$

est nettement plus difficile et aucun algorithme efficace n'est connu. Par exemple, pour trouver $2^x \equiv 3 \pmod{13}$, on essaie successivement

$$\begin{aligned} 2^1 &\equiv 2 \pmod{13} \\ 2^2 &\equiv 4 \pmod{13} \\ 2^3 &\equiv 8 \pmod{13} \\ 2^4 &\equiv 3 \pmod{13} \\ 2^5 &\equiv 6 \pmod{13} \\ &\vdots \end{aligned}$$

d'où $x = 4$. Cette procédure devient très longue si m est grand, et de plus une congruence de ce type n'admet pas toujours de solution. Comme la fonction exponentielle est facile à calculer, mais sa réciproque, le logarithme, est difficile à calculer, nous appellons cette fonction une *fonction à sens unique*. De la même façon, un botin téléphonique est une fonction à sens unique car il est facile de trouver le numéro de quelqu'un en connaissant son nom, mais beaucoup moins évident de trouver le nom de la personne au numéro de téléphone donné.

3.3.7 Fonction Indicatrice d'Euler

L'ensemble *complet* des résidus modulo m est $\{0, 1, 2, \ldots, m-1\}$. L'ensemble *réduit* des résidus modulo m est l'ensemble des résidus modulo m qui sont premiers relativement à m.

Exemple 8. *Par exemple, l'ensemble complet des résidus modulo 10 est* $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ *mais l'ensemble réduit des résidus modulo 10 est* $\{1, 3, 7, 9\}$.

La *fonction indicatrice d'Euler* $\phi(m)$ est le nombre d'éléments de l'ensemble réduit des résidus modulo m. Pour notre exemple, on a donc $\phi(10) = 4$. Remarquons que $\phi(1) = 1$.

Si m est un nombre premier, chaque résidu non nul $r = 1, 2, \ldots, m-1$ est relativement premier avec m. Par conséquent, $\phi(m) = m - 1$ dans ce cas.

De plus, si m est premier, l'ensemble réduit des résidus modulo m^2 est $\{1, 2, \ldots, m-1, m+1, \ldots, 2m-1, 2m+1, \ldots, m^2-1\}$ et comporte donc $m^2 - 1 - (m-1) = m(m-1)$ éléments, d'où $\phi(m^2) = m(m-1)$. D'une façon générale, on peut montrer que $\phi(m^n) = m^{n-1}(m-1)$ si m est premier.

La fonction d'Euler a enfin la propriété intéressante d'être multiplicative, à savoir

$$\phi(mn) = \phi(m)\phi(n)$$

si m et n sont relativement premiers entre eux.

Par conséquent, si un nombre entier m est décomposé en facteurs premiers p_i :

$$m = \prod_i p_i^{e_i}$$

avec $e_i > 0$, sa fonction indicatrice d'Euler sera

$$\phi(m) = \prod_i p_i^{e_i - 1}(p_i - 1) = m \prod_i (1 - \frac{1}{p_i}). \tag{3.8}$$

3.3.8 Théorème d'Euler

Théorème 2. *Si a et m sont relativement premiers entre eux,*

$$a^{\phi(m)} \equiv 1 \pmod{m}. \tag{3.9}$$

Démontrons ce théorème. Soit $\{r_1, r_2, \ldots, r_{\phi(m)}\}$ l'ensemble réduit des résidus modulo m. Multiplions chacun d'eux par a, et examinons l'ensemble des résidus $\left\{|ar_1|_m, |ar_2|_m, \ldots, |ar_{\phi(m)}|_m\right\}$. Comme d'une part a et m sont relativement premiers entre eux, et d'autre part r_i et m sont également relativement premiers entre eux pour tout $1 \leq i \leq \phi(m)$, ar_i et m sont relativement premiers entre eux. Par conséquent, $|ar_i|_m = |r_j|_m$ pour un certain $1 \leq j \leq \phi(m)$ (cfr exercice 15). D'autre part, si $i \neq k$, alors $|r_i|_m \neq |r_k|_m$, et (cfr exercice 16)

$$|ar_i|_m = |a|_m \cdot |r_i|_m \neq |a|_m \cdot |r_k|_m = |ar_k|_m,$$

ce qui montre que chaque résidu de ar_i modulo m est un membre différent de l'ensemble réduit des résidus modulo m, et donc

$$\left\{|ar_1|_m, |ar_2|_m, \ldots, |ar_{\phi(m)}|_m\right\} = \left\{|r_1|_m, |r_2|_m, \ldots, |r_{\phi(m)}|_m\right\}.$$

Il s'ensuit que

$$\begin{aligned} \left|a^{\phi(m)} r_1 r_2 \ldots r_{\phi(m)}\right|_m &= |ar_1|_m \cdot |ar_2|_m \cdot \ldots \cdot |ar_{\phi(m)}|_m \\ &= |r_1|_m \cdot |r_2|_m \cdot \ldots \cdot |r_{\phi(m)}|_m \\ &= \left|r_1 r_2 \ldots r_{\phi(m)}\right|_m \end{aligned}$$

et donc, comme r_i et m sont relativement premiers entre eux pour tout $1 \leq i \leq \phi(m)$, que

$$\left|a^{\phi(m)}\right|_m = |1|_m.$$

CQFD

3.3.9 Théorème de Fermat

Théorème 3. *Un corollaire direct du théorème d'Euler est celui de Fermat : if m est premier, et si a n'est pas divisible par m, alors*

$$a^{m-1} \equiv 1 \pmod{m}. \tag{3.10}$$

3.3.10 Calcul de l'Inverse

Nous avons montré que a possède un inverse modulo m, en d'autres termes il existe un élément (a^{-1}) tel que

$$a.a^{-1} \equiv 1 \pmod{m} \tag{3.11}$$

si et seulement si a et m sont relativement premiers entre eux.

On dispose de trois méthodes de calcul de l'inverse :

1. Chercher parmi les nombres $1, 2, \ldots, m-1$ juqu'à ce qu'on trouve un nombre a^{-1} satisfaisant (3.11).
2. Si la fonction d'Euler $\phi(m)$ est connue, on calcule

$$a^{-1} \equiv a^{\phi(m)-1} \pmod{m}. \tag{3.12}$$

   Le théorème d'Euler garantit alors que $a.a^{-1} \equiv 1 \pmod{m}$.
3. Utilisant l'indentité de Bezout 3.2, nous avons que $\alpha a + \mu m + \gcd(a, m) = 1$. Ce qui nous donne $a^{-1} = \alpha$ comme $\alpha \equiv 1 \pmod{m}$. Nous pouvons utiliser l'algorithme d'Euclide étendu pour calculer α.

3.3.11 Théorème des Restes Chinois

Théorème 4. *Soient m_1 et m_2 deux entiers relativement premiers entre eux. Alors quels que soient les entiers r_1 et r_2, il existe un entier r tel que tout entier a vérifiant le système de congruences*

$$a \equiv r_1 \pmod{m_1} \tag{3.13}$$

$$a \equiv r_2 \pmod{m_2} \tag{3.14}$$

vérifie aussi

$$a \equiv r \pmod{m_1 m_2}. \tag{3.15}$$

Pour démontrer ce résultat, considérons deux entiers u et v vérifiant l'identité de Bezout

$$um_1 + vm_2 = 1 \tag{3.16}$$

et posons

$$r = r_2 u m_1 + r_1 v m_2. \tag{3.17}$$

Alors $r \equiv r_1 v m_2 \pmod{m_1}$, mais l'identité de Bezout (3.16) entraîne que $vm_2 \equiv 1 \pmod{m_1}$, et donc $r \equiv r_1 \pmod{m_1}$. Le même raisonnement donne $r \equiv r_2 \pmod{m_2}$, et a vérifie les deux premières congruences (3.13) et (3.14).

Si a est un entier qui vérifie ces congruences nous avons $a - r \equiv 0 \pmod{m_1}$ et $a - r \equiv 0 \pmod{m_2}$. Il s'ensuit que m_1 et m_2 divisent $a - r$. Comme ces deux nombres sont premiers entre eux, leur produit $m_1 m_2$ divise également $a - r$, et a vérifie la troisième congruence (3.15). CQFD.

Ce théorème s'étend par récurrence à un nombre quelconque de congruences.

3.4 Algorithmes de Chiffrage Asymétriques

3.4.1 Conditions de Mise en Place d'un Cryptosystème à Clé Publique

Rappelons qu'un algorithme de cryptage E, paramétrisé par une clé K, transforme le texte clair P en cryptogramme $C = E_K(P)$. L'algorithme de décryptage D, paramétrisé par une clé k, effectue la transformation inverse $P = D_k(C)$.

A la section 3.2, on n'avait considéré que les algorithmes de cryptage symétriques, pour lesquels la même clé était utilisée au cryptage et au décryptage : $K = k$. Un tel cryptosystème requiert donc la communication à l'avance de la clé secrète entre l'expéditeur et le récepteur, ce qui est une opération délicate, surtout si la clé doit être distribuée entre un grand nombre d'usagers, comme c'est le cas pour des réseaux informatiques. Si la clé secrète tombe entre les mains de l'ennemi, le système entier est compromis.

Un cryptosystème à clé publique élimine ce problème en permettant l'utilisation d'une clé différente au cryptage et au décryptage : $K \neq k$. La clé K est publique tandis que la clé k est secrète. De cette manière, la clé de cryptage K peut être transmise à travers le canal de communication non sûr, puisque l'ennemi la connaît.

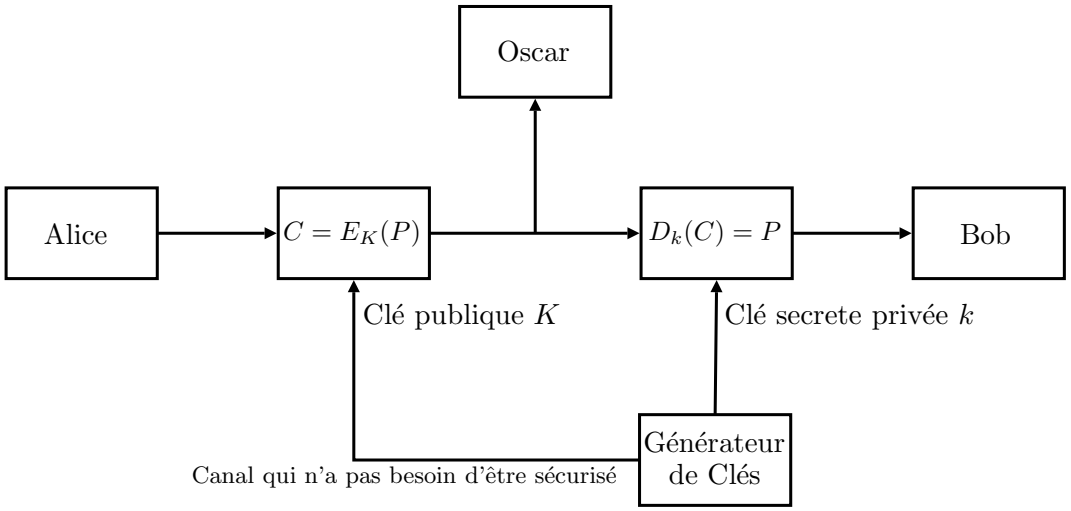

FIG. 3.3 – Chiffrage asymétrique.

Pour qu'un cryptosystème à clé publique fonctionne correctement, plusieurs conditions doivent être remplies :

1. Le cryptage $E_K(P)$ d'un message clair P est une opération aisée et rapide.
2. Le décryptage $D_k(C)$ d'un cryptogramme C est une opération aisée si le récepteur connaît la clé k.
3. Par contre, le calcul de $E_K^{-1}(C)$ sans la connaissance de k est extrêmement difficile, et impossible à effectuer en pratique, en un temps raisonnable.

4. Enfin, comme dans n'importe quel cryptosystème, l'algorithme de décryptage doit être l'inverse de celui de cryptage : $D_k(E_K(P)) = P$.

Comme nous pouvons voir, E_K est une foncition à sens unique. Si en plus, $f^{-1}(\cdot)$ est facilement calculable à condition de disposer d'une information secrète associée à la fonction (c'est-à-dire la clé k), la fonction n'est plus tout à fait à sens unique mais devient alors une *fonction à "trappe"* ("trap-door function"). La fonction $E_K(\cdot)$ est donc en fait une telle fonction.

Un usager qui désire envoyer et recevoir des cryptogrammes se munit donc deux clés, l'une (k) qu'il garde secrète et l'autre (K) qu'il publie dans un annuaire que tous les autres usagers peuvent consulter. La distribution des clés est donc extrêmement facile.

3.4.2 L'Algorithme de Rivest-Shamir-Adleman (RSA)

Dans le cryptosystème RSA, les messages clairs P, cryptogrammes C et clés K, k sont codés comme des nombres entiers modulo m, et sont tous compris entre 1 et $m-1$. Par conséquent, P, C, K et $k \in \mathbb{Z}_m$. Le module m est rendu public. Les fonctions de cryptage et de décryptage sont alors respectivement

$$E_K(P) = C \equiv P^K \pmod{m} \tag{3.18}$$

avec $1 \leq C \leq m-1$ et

$$D_k(C) = P \equiv C^k \pmod{m} \tag{3.19}$$

avec $1 \leq P \leq m-1$, tandis que les clés sont choisies de telle manière que

$$\begin{aligned} \text{pgcd}(K, \phi(m)) &= 1 && (3.20) \\ Kk &\equiv 1 \pmod{\phi(m)} && (3.21) \end{aligned}$$

avec $1 \leq K, k \leq \phi(m) - 1$. Remarquons que cette dernière congruence indique que k est l'inverse de K modulo $\phi(m)$, et que cet inverse existe toujours puisque K et $\phi(m)$ sont relativement premiers entre eux par (3.20).

Exemple 9. *Considérons l'exemple trivial suivant : chaque lettre du message clair est codée par sa position dans l'alphabet, et on prend $m = 33$, d'où $\phi(m) = \phi(11)\phi(3) = 20$. Prenons $K = 7$, qui est relativement premier avec $\phi(m)$. On peut alors calculer que $k = 3$ car $7 \cdot 3 \equiv 1 \pmod{20}$. Par conséquent le cryptage et le décryptage du mot "bonjour" deviennent :*

	Texte clair codé P	*Texte crypté* $C = \lvert P^7 \rvert_{33}$	*Texte décrypté* $\lvert C^3 \rvert_{33}$	
B	02	29	02	B
O	15	27	15	O
N	14	20	14	N
J	10	10	10	J
O	15	27	15	O
U	21	21	21	U
R	18	06	18	R

Il faut maintenant vérifier que cet algorithme satisfait aux quatre conditions établies en 3.4.1, et qui définissent un chiffrage à clé publique :

1. Tout d'abord, le **cryptage** est une opération d'exponentiation qui est **rapide**, comme nous l'avons vu sur un exemple en 3.3.6, et est de plus **facile** à mettre en oeuvre en hardware.

2. Le **décryptage** est également une opération d'exponentiation, elle aussi **rapide et aisée dès que k est connu**.

3. Pour que la troisième condition soit satisfaite, il doit être **extrêmement compliqué de découvrir C et k si k n'est pas connu**.

Supposons tout d'abord que m soit un nombre premier. A ce moment-là, $\phi(m) = m - 1$ est connu puisque m est public, et la congruence (3.21) devient

$$Kk \equiv 1 \pmod{m-1}.$$

Par conséquent, k est égal à K^{-1} modulo $m - 1$, et peut être calculé en utilisant une des méthodes vues en 3.3.10. Nous savons que ce calcul doit être faisable, sinon l'utilisateur ne pourrait calculer k lors de l'installation de son cryptosystème. Ceci permettrait au cryptanalyste de compromettre le système, ce qui est inacceptable. Clairement, nous devons choisir m tel que $\phi(m)$ soit difficile à calculer.

On doit donc choisir un nombre m non premier. On prend

$$m = pq \tag{3.22}$$

où p et q sont deux nombres premiers, appelés *conditions initiales* de l'algorithme. Maintenant, $\phi(m) = (p-1)(q-1)$ n'est connu que si l'on dispose des deux facteurs premiers p et q. Par conséquent, l'usager régulier va publier K et m, mais pas k ni p et q, qu'il est donc le seul à connaître. Lui seul pourra dès lors calculer relativement facilement la clé k à partir de K par une des méthodes vues en 3.3.10, car il connaît $\phi(m)$. Par contre, le cryptanalyste ne disposant pas de $\phi(m)$, aura beaucoup plus de difficultés à retrouver k. Nous verrons à la section suivante qu'il sera en fait incapable de compromettre le cryptosystème dans des délais raisonnables, ce qui montre que cette troisième condition est maintenant satisfaite.

4. Enfin, il faut vérifier que **les fonctions E et D sont inverses l'une de l'autre**, c'est-à-dire que $D_k(E_K(P)) = P$ ou encore que

$$(P^K)^k \equiv P \pmod{m}. \tag{3.23}$$

Comme (3.21) est vérifiée, il existe un nombre entier x tel que $Kk = 1 + x\phi(m)$ et donc

$$P^{Kk} = P^{1+x\phi(m)} = P \cdot (P^{\phi(m)})^x.$$

Si P et m sont relativement premiers entre eux, le théorème d'Euler indique que $P^{\phi(m)} \equiv 1 \pmod{m}$. Dès lors

$$|P \cdot (P^{\phi(m)})^x|_m = |P|_m \cdot (|P^{\phi(m)}|_m)^x = |P|_m \cdot |1^x|_m = |P|_m \cdot |1|_m = |P|_m,$$

et par conséquent (3.23) est satisfaite.

Supposons maintenant que P et m ne soient pas relativement premiers entre eux, ce qui est assez rare si m est grand. Comme $m = pq$ avec p et q premiers, il faut que $P = ip$ ou que $P = jq$ avec $1 \leq j \leq p-1$. Supposons sans perte de généralité que $P = ip$. A ce moment-là, $|P|_p = |i|_p.|p|_p = |0|_p$ et donc

$$P^{x\phi(m)} \equiv 0 \pmod{p}. \tag{3.24}$$

D'autre part, comme q est premier et ne divise pas P, le théorème de Fermat entraîne que

$$P^{q-1} \equiv 1 \pmod{q},$$

et donc que

$$P^{x(p-1)(q-1)} = P^{x\phi(m)} \equiv 1 \pmod{q}. \tag{3.25}$$

Par conséquent, les congruences (3.24) et (3.25) et le théorème des restes chinois impliquent qu'il existe un entier r tel que

$$P^{x\phi(m)} \equiv r \pmod{pq},$$

et donc

$$P^{Kk} = P^{1+x\phi(m)} \equiv Pr \pmod{pq}. \tag{3.26}$$

Si u et v sont deux entiers satisfaisant l'identité de Bezout $up+vq = 1$, la relation (3.17) permet de déterminer cet entier r : $r = up \cdot 1 + vq \cdot 0 = up$, ou encore $r = 1 - vq$. Dès lors

$$|Pr|_{pq} = |P|_{pq} \cdot |1 - vq|_{pq} = |P|_{pq} \cdot (|1|_{pq} - |vq|_{pq}) = |P|_{pq} - |Pvq|_{pq} = |P|_{pq} - |ivpq|_{pq} = |P|_{pq}$$

et (3.26) devient

$$P^{Kk} \equiv P \pmod{pq},$$

ce qui prouve que la dernière condition mentionnée à la section précédente est toujours satisfaite.

3.4.3 Sécurité de l'Algorithme RSA

Comment un cryptanlyste pourrait-il essayer de briser un cryptosystème RSA ? Il a trois angles d'attaque possibles :

1. Il peut tout d'abord essayer de déterminer les *conditions initiales* p et q. En effet, une fois celles-ci découvertes, il connaît $\phi(m) = (p-1)(q-1)$ et peut retrouver la clé secrète k à partir de (3.21). Mais si on prend m très grand, la **factorisation** de m en deux nombres premiers est extrêmement difficile. Une méthode serait par exemple d'essayer tous les entiers jusqu'à $\sqrt{m}$ pour retrouver p et q. D'autres algorithmes sont plus rapides, mais restent totalement inefficaces si m est très grand. Rivest, Shamir et Adleman ont calculé qu'il faudrait 4 million d'années pour factoriser un nombre m de 200 chiffres avec un processeur de 1Ghz.

2. Il peut ensuite essayer de résoudre

$$C \equiv P^K \pmod{m}$$

connaissant C, K et m, pour ne retrouver cette fois que le texte clair P (la clé k restant alors toujours inconnue). Il est alors confronté au problème du calcul du **logarithme discret**, qui

est très compliqué, comme nous l'avons vu en 3.3.6. Ce calcul est de complexité équivalente à la factorisation, nous laissons la même attente de 4 million d'annés.

3. Une troisième attaque possible est le **cryptage successif du cryptogramme**. En effet, le cryptanalyste disposant de K, m et C peut décider de crypter lui-même le cryptogramme C pour obtenir un nouveau cryptogramme C_1 :

$$C_1 = E_K(C) \equiv C^K \pmod{m}.$$

Il peut répéter la même opération avec C_1 pour obtenir C_2, et ainsi de suite pour n'importe quel i :

$$C_{i+1} = E_K(C_i) \equiv C_i^K \pmod{m}.$$

Comme $1 \leq C \leq m-1$, il y a un nombre fini de C_i et il se peut qu'à un certain moment $i = n$, $C_n = C$. A ce moment-là,

$$C_n = C = E_K(C_{n-1}) = E_K(P) \equiv P^K \pmod{m}$$

et donc le cryptage du cryptogramme précédent, C_{n-1}, doit être identique au texte clair P. Il faut donc que cet entier n, appelé *ordre* de $E(\cdot)$, soit aussi élevé que possible, pour qu'une telle attaque soit impossible à réaliser en un temps relativement court. Rivest a montré que si les entiers $p-1$ et $q-1$ contiennent de grands facteurs premiers (par exemple $p-1 = 2p'$ et $q-1 = 2q'$, où p' et q' sont des nombres premiers), alors la probabilité qu'une telle attaque réussisse est proche de zéro.

Par conséquent, le cryptanalyste ne peut pas, en pratique, compromettre un cryptosystème RSA.

3.4.4 Remarques sur le Choix des Paramètres du Cryptosystème RSA

Si le cryptanalyste est heureusement confronté à des problèmes insurmontables, la détermination des clés par le concepteur du cryptosystème n'est cependant pas triviale pour autant. Deux problèmes doivent retenir son attention :

1. Pour résister au troisième type d'attaque mentionné à la section précédente, il faut que $p-1$ et $q-1$ contiennent de grands facteurs, par exemple que $p-1 = 2p'$ et $q-1 = 2q'$, où p' et q' sont des nombres premiers. Une fois que p' et q' sont choisis, il faut encore s'assurer que p et q sont eux aussi des nombres premiers. Typiquement, p et q sont des nombres de 100 chiffres, de sorte que m est un nombre de 200 chiffres. Il est nettement plus facile de tester si un nombre de 100 chiffres est premier, que de trouver les diviseurs d'un nombre de 200 chiffres (ce que tente la première attaque du cryptanalyste), mais ce test prend néanmoins du temps.
2. Lors du cryptage présenté à la section 3.4.2, on remarque que les nombres 10 et 21 du texte clair sont cryptés respectivement en 10 et 21 ! En effet, $10 \equiv 10^7 \pmod{33}$ et $21 \equiv 21^7 \pmod{33}$, et donc le cryptogramme et le texte clair sont identiques pour ces valeurs, ce qui est évidemment tout bénéfice pour le cryptanalyste (Le troisième type d'attaque

mentionné à la section précédente est alors immédiat). L'algorithme RSA ne "cache" pas tous les messages P tels que $C = P$, c'est-à-dire

$$P^K \equiv P \pmod{m}.$$

On peut montrer que le nombre σ de messages $1 \leq P \leq m-1$ que le cryptosystème ne peut cacher est égal à

$$\sigma = (1 + \text{pgcd}(K-1, p-1))(1 + \text{pgcd}(K-1, q-1)) - 1.$$

Il faut donc choisir la clé K de manière à minimiser σ. Si cette clé est choisie sans faire attention à cette remarque, plus de la moitié du texte clair risque d'être à découvert !

Enfin, signalons que l'algorithme RSA peut être utilisé pour d'autres tâches que la protection d'informations, comme l'*authentification*.

3.5 Références Bibliographiques

D. R. Stinson, *Cryptography : Theory and Practice*. CRC Press, 1995.

Simulateur d'Enigma, site web de Bletchley Park, `http ://www.bletchleypark.org.uk`

C. E. Shannon, *Prediction and Entropy of Printed English*. Bell System Technical Journal, Vol 30 (1951), pp.50-64 Reprinted in D. Slepian, editor, Key Papers in the Development of Information Theory, IEEE Press, NY, 1974.

3.6 Exercices

Exercice 1. Briser les cryptogrammes suivants, cryptés par substitution monoalphabétique effectuant une rotation de lettres de l'alphabet de k positions :

a) TU JEKI BUI FUKFBUI TU BQ WQKBU BUI RUBWUI IEDJ BUI FBKI RHQLUI

b) JK ZUAY RKY GAZXOINOKTY RK VRAY IKRKHXK KYZ IGROLUXTOKT

La première phrase (a) est une traduction en français d'une phrase de Jules César, provenant de "La Guerre des Gaules". La seconde (b) est un point de vue particuiler.

Exercice 2. Quelle est la distance d'unicité d'un cryptage monoalphabétique effectuant une rotation de lettres de l'alphabet de k positions, dans le cas de l'anglais ? Quelle est la longueur minimale de texte crypté dont le cryptanaliste doit disposer pour briser un tel cryptogramme (Il connaît la méthode de cryptage qui consiste en une rotation des lettres de k positions) ? Montrez que si la longueur du texte crypté dont il dispose est inférieure à la distance d'unicité, le cryptanalyste est incapable de déterminer le texte clair correspondant.

Exercice 3. Répéter l'exercice 2 pour un cryptage de César, où la valeur de k est fixée à 3.

Exercice 4. Le texte clair "thisisasample" est crypté en "KVCFLTADLDDFR".

a) La méthode de cryptage était-elle la substitution monoalphabétique ou polyalphabétique (cryptage de Vigenère) ?

b) Quelle était la clé utilisée ?

Exercice 5. L'alphabet d'une source est formé de cinq symboles émis indépendamment les uns des autres et ayant chacun la même probabilité $p_i = 0.2$ $(1 \leq i \leq 5)$ d'être émis. Les symboles sont cryptés par substitution monoalphabétique. Cette méthode vous semble-t-elle suffisante pour résister à l'attaque d'un cryptanalyste ne disposant que du cryptogramme, et pas du texte clair (Attaque à texte crypté seul) ? Pourquoi ?

Exercice 6. Décryptez le cryptogramme "EOBABNOKTOTEORTX" obtenu par transposition avec la clé SURF.

Exercice 7. Calculer x tel que

a) $x \equiv 4^8 \pmod{15}$

b) $x \equiv 1234^{567} \pmod{99}$

c) $x \equiv 2^{123}3^{456}5^{789} \pmod{4}$.

Exercice 8. Calculer la fonction indicatrice d'Euler $\phi(m)$ pour $m = 29$, 30, 31 et 32.

Exercice 9. *Démontrer les propriétés suivantes.*
- $pgdc(a, 1) = pgdc(a - a \cdot 1, 1) = pgdc(0, 1) = 1$,
- $pgdc(a, p) = 1$ *avec* $a \neq 0$ *et* p *premier,*
- $pgdc(a, p) = p$ *avec* $a = 0$ *et* p *premier.*

Exercice 10. Calculer le PGCD de

a) 234 et 325

b) 97 et 109.

Exercice 11. Déterminer deux entiers u et v qui vérifient l'identité de Bezout

$$234u + 325v = \text{pgcd}(234, 325).$$

Exercice 12. Calculer, s'ils existent, les inverses x suivants :

a) $x \equiv 3^{-1} \pmod{10}$

b) $x \equiv 3^{-1} \pmod{15}$

c) $x \equiv 3^{-1} \pmod{23}$.

Exercice 13. Déterminer un entier x qui vérifie les congruences

$$\begin{aligned} x &\equiv 7 \pmod{11} \\ x &\equiv 6 \pmod{13}. \end{aligned}$$

Exercice 14. Démontrer que la relation (3.3) est symétrique et transitive.

Exercice 15. Si $a \equiv b \pmod{m}$ et que b et m sont relativement premiers entre eux, montrer que a et m sont alors eux aussi relativement premiers entre eux.

Exercice 16. a) Si $a \equiv a' \pmod{m}$, montrer que pour tout entier t

$$at \equiv a't \pmod{m}.$$

b) Si $ad \equiv a'd \pmod{m}$ et que d et m sont relativement premiers entre eux, montrer que

$$a \equiv a' \pmod{m}.$$

Cette propriété est-elle toujours vraie si d et m ne sont plus relativement premiers entre eux ?

Exercice 17. Un cryptosystème à clé publique doit-il pouvoir résister à une attaque à texte crypté seul ? à texte clair connu ? à texte clair choisi ?

Exercice 18. Les conditions initiales d'un cryptosystème RSA sont $p = 23$ et $q = 47$, tandis que la clé publique est $K = 13$. Quelle est la clé secrète ?

Exercice 19. Vous êtes un cryptanalyste confronté à un cryptosystème RSA où $m = 187$ et $K = 7$.

a) Vous disposez du cryptogramme $C = 183$, qui a été envoyé par l'usager régulier. Sans essayer de déterminer la clé secrète, brisez ce cryptogramme.

b) Compromettez le système complet en déterminant la clé secrète k.

Exercice 20. Les conditions initiales d'un cryptosystème RSA sont $p = 97$ et $q = 109$. Quelles sont, parmi les clés publiques suivantes, celles qui peuvent effectivement être choisies ?

a) $K = 123$

b) $K = 865$

c) $K = 169$

Exercice 21. Un cryptanalyste dispose, en plus de $m = 10001$, de $\phi(m) = 9792$. Quelles sont les conditions initiales p et q ? (Indication : on peut écrire $\phi(m)$ sous la forme $\phi(m) = m - p - q + 1$.)

Exercice 22. Comme signalé à la section 3.4.3, il est intéressant de disposer, lors de la conception d'un cryptosystème RSA, d'un algorithme permettant de tester rapidement si un entier p donné est premier.

Le théorème de Wilson, qui précise que

$$(p-1)! \equiv -1 \pmod{p}$$

si et seulement si p est premier, procure un tel test : la fonction

$$f(p) = \sin(\pi \, \frac{(p-1)! + 1}{p})$$

s'annule si et seulement si p est premier. Ce test est-il utilisable en pratique ?

Exercice 23. Au lieu d'encrypter le texte clair avec la clé publique K et de le décrypter avec la clé secrète k, on effectue l'opération inverse : le clé de *cryptage* est la clé secrète k tandis que la clé de *décryptage* est la clé publique K. A quoi pourrait servir ce système ?