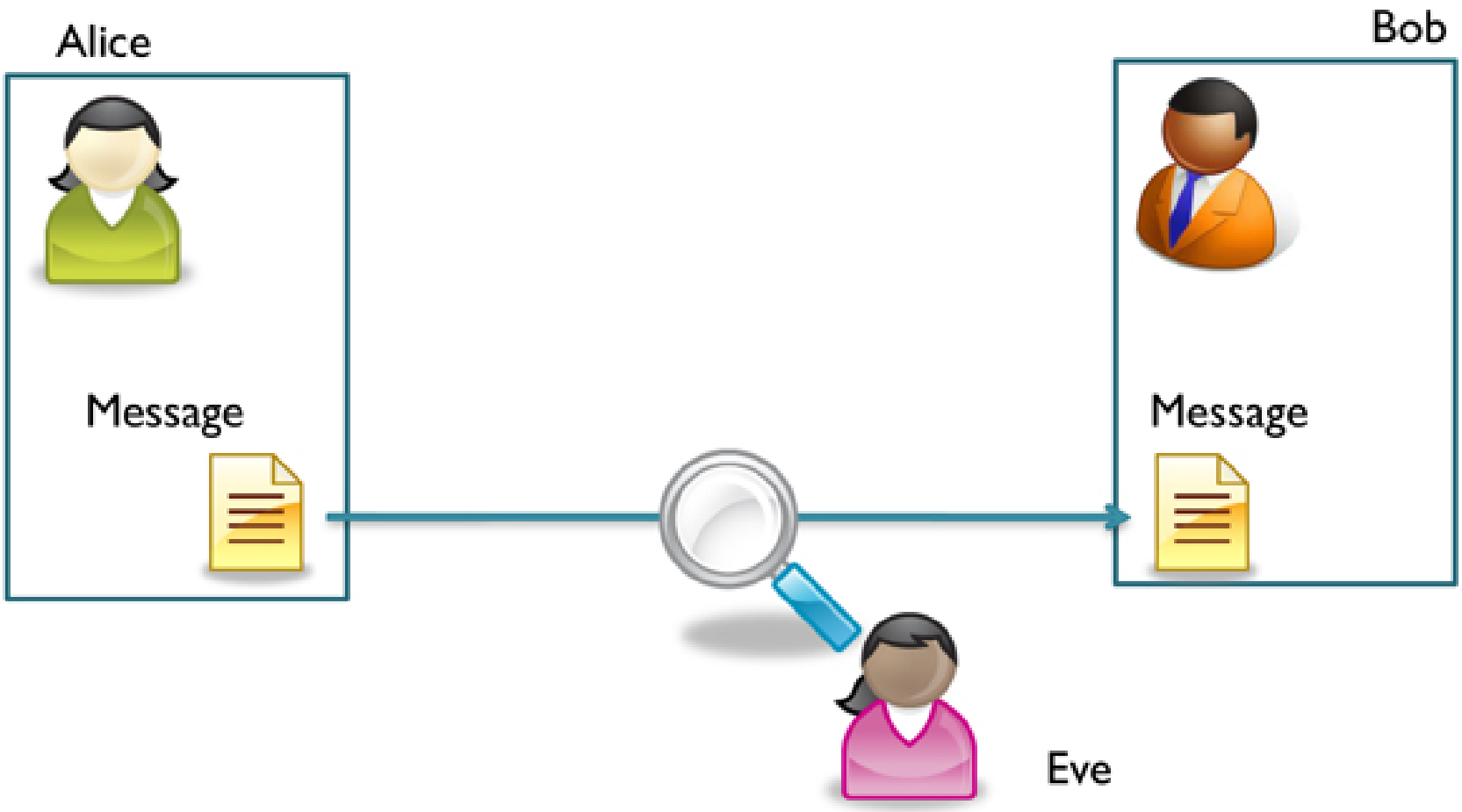
Cryptographie

Paul Zimmermann
INRIA Nancy - Grand Est

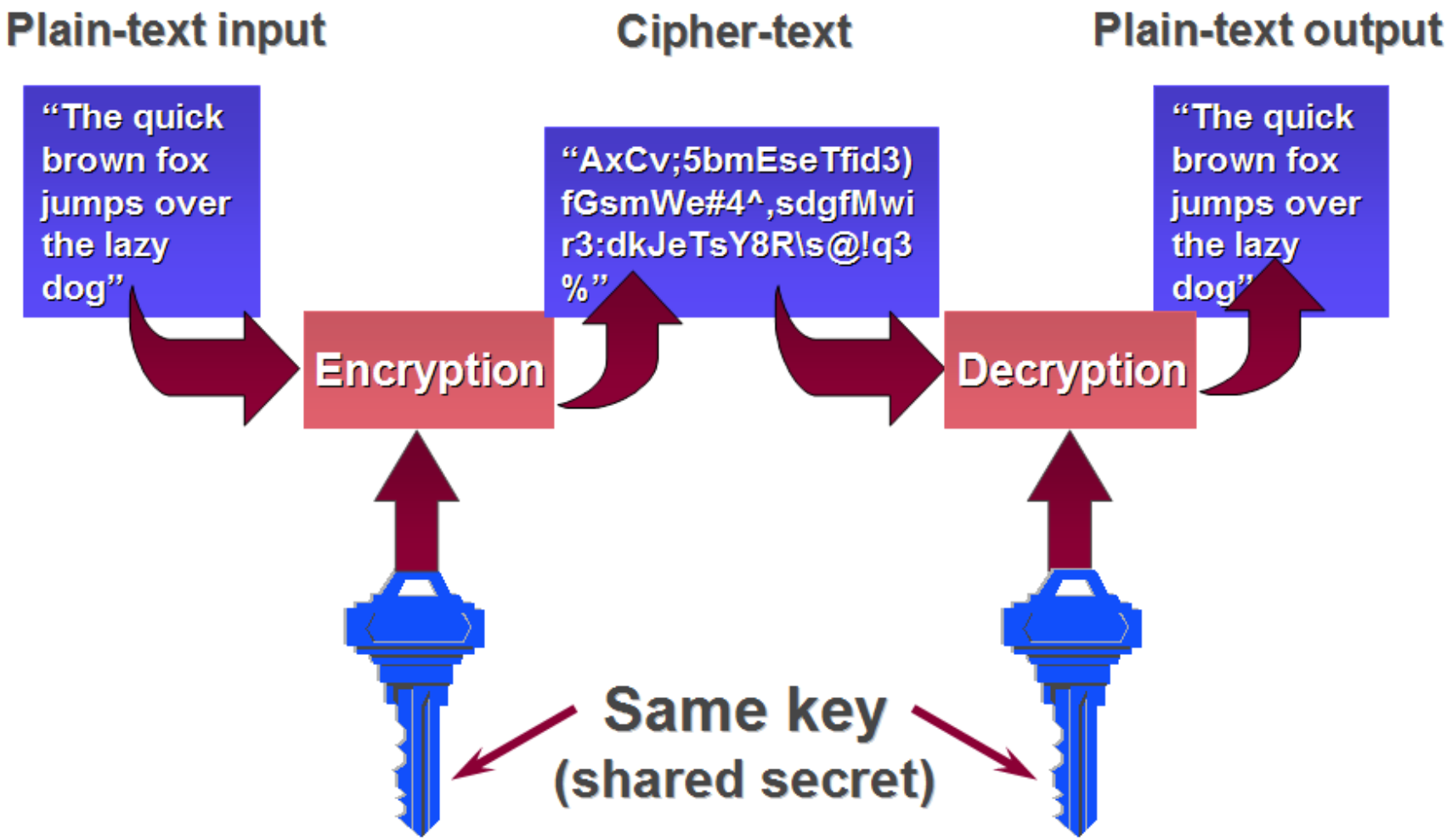
Lycée Jean de Pange
Sarreguemines
10 février 2012

- cryptographie : construction de codes secrets
- cryptanalyse : « cassage » de codes secrets

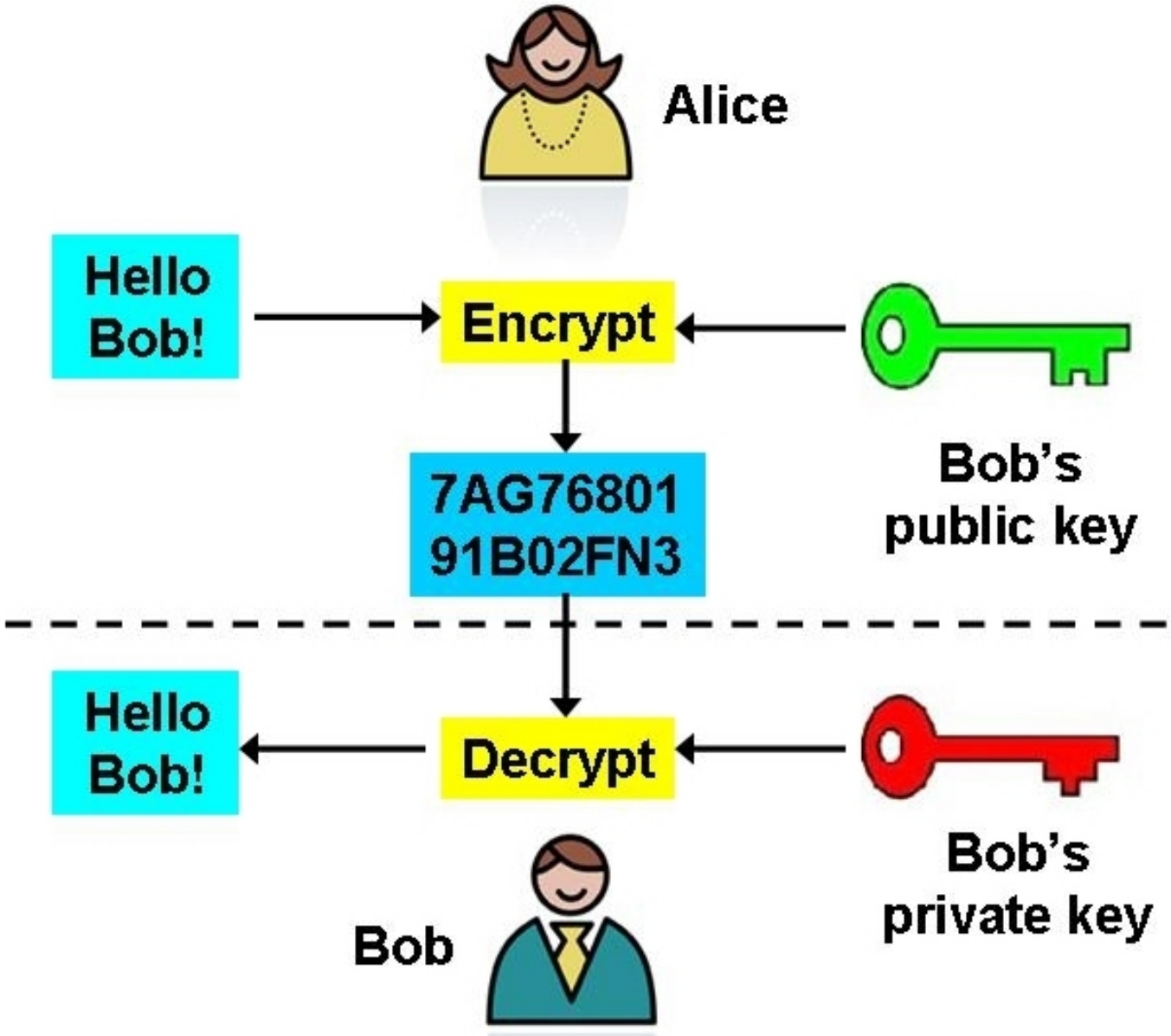
$$\text{cryptologie} = \text{cryptographie} + \text{cryptanalyse}$$


Alice
Bob
Message
Message
Eve

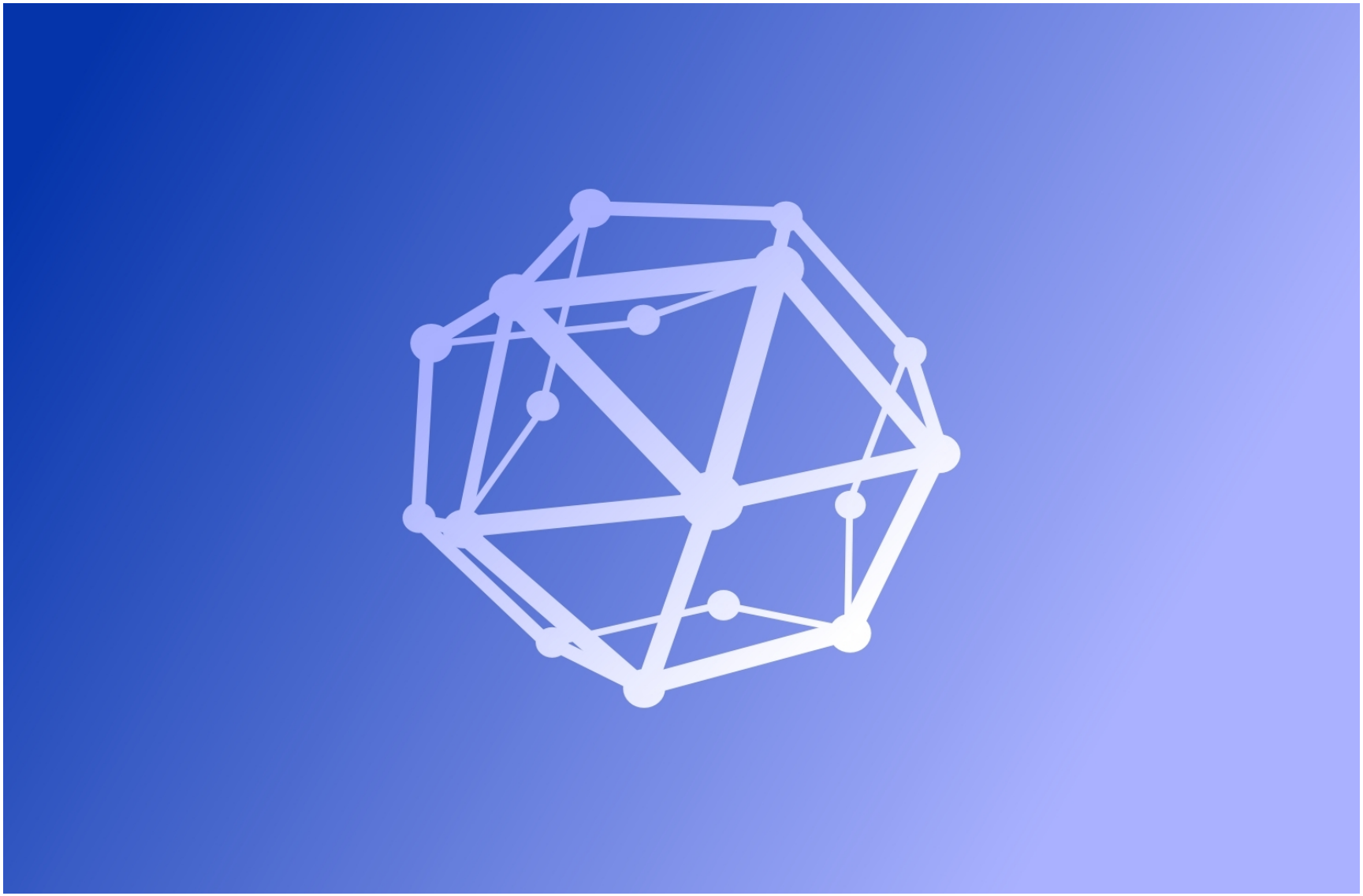
Cryptographie symétrique (à clef privée)



Cryptographie asymétrique (à clef publique)



Logiciel Sage : sagemath.org



Chiffrement par décalage (César)

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
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

(substitution mono-alphabétique)

```
sage: alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
sage: message = 'MATHEMATIQUES'

sage: def encode(m, cle):
          return join([alphabet[(alphabet.find(x)+cle)
              % 26] for x in m],'')

sage: encode(message, 4)
'QEXLIQEXMUYIW'
```

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
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

```
sage: def decode(c, cle):
          return join([alphabet[(alphabet.find(x)-cle)
             % 26] for x in c],'')

sage: decode('QEXLIQEXMUYIW', 4)
'MATHEMATIQUES'
```

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
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

```
sage: def decode(c, cle):
          return join([alphabet[(alphabet.find(x)-cle)
             % 26] for x in c],'')

sage: decode('QEXLIQEXMUYIW', 4)
'MATHEMATIQUES'

sage: encode('QEXLIQEXMUYIW', -4)
'MATHEMATIQUES'
```

```
sage: def casse(c):
          for cle in range(26):
              print cle, decode(c, cle)
sage: casse('QEXLIQEXMUYIW')
0 QEXLIQEXMUYIW
1 PDWKHPDWLTXHV
2 OCVJGOCVKSWGU
3 NBUIFNBUJRVFT
4 MATHEMATIQUES
5 LZSGDLZSHPTDR
6 KYRFCKYRGOSCQ
7 JXQEBJXQFNRBP
...
```

Attaque sur la taille de l'espace des clés

Employé pourtant par officiers sudistes pendant guerre de Sécession (1861-1865) et armée russe en 1915.

Quelques primitives cryptographiques

- confidentialité des données
- intégrité des données
- authentification
- signature
- dater un document (*timestamping*)
- connaissance d'une donnée sans la révéler (*zero-knowledge proof*)
- préserver anonymat
- non-révocation

Un exemple : le vote électronique

On veut garantir :

- secret du scrutin
- vérification de bonne prise en compte d'un vote
- possibilité de dépouillement par tout-un-chacun
- impossibilité de « vendre » un vote

- machines à voter : États-Unis depuis 1990, Belgique depuis 1991 (44% des électeurs), élections municipales de 6 novembre 2005 au Québec (95% des électeurs)
- vote électronique à domicile

Le tatouage numérique (*watermarking*)

Permet d'identifier la source d'une image (copyright).

Tatouage invisible : pour détecter les copies illicites.

Peut s'appliquer aussi à des données (introduction de mots faux dans un dictionnaire), du son, de la vidéo, ...

Chiffrement de Blaise de Vigenère (1586)

TRAICTE'

DES CHIFFRES,

OV SECRETES

MANIERES

D'ESCRIRE:

PAR

BLAISE DE VIGENERE,

BOVRBONNOIS.

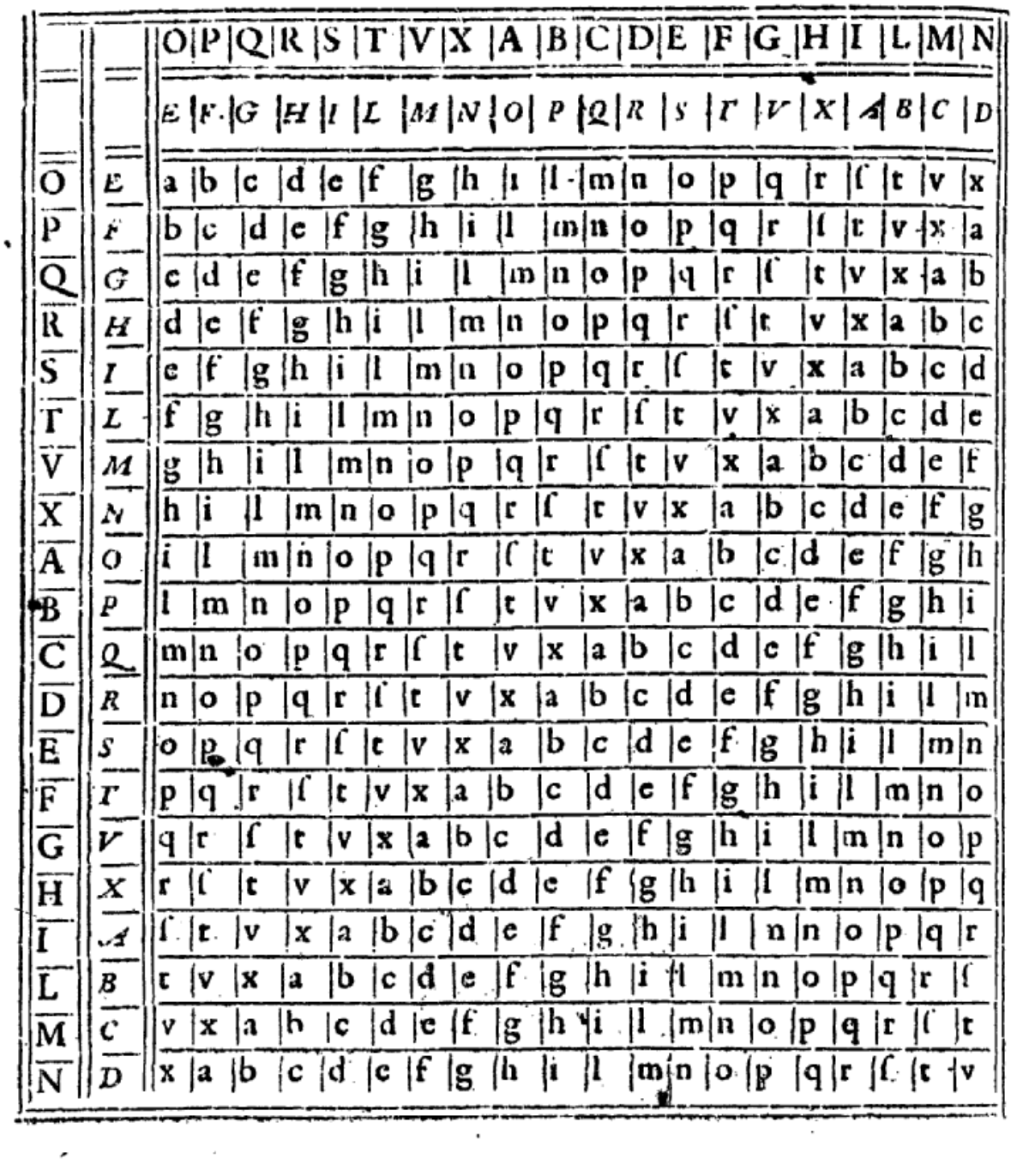
antes muerto que mudado

A PARIS,

Chez ABEL L'ANGELIER, au premier pillier de la grand' Salle du Palais.

M. D. LXXXVI.

AVEC PRIVILEGE DV ROY.



		O	P	Q	R	S	T	V	X	A	B	C	D	E	F	G	H	I	L	M	N
		E	F	G	H	I	L	M	N	O	P	Q	R	S	T	V	X	A	B	C	D
O	E	a	b	c	d	e	f	g	h	i	l	m	n	o	p	q	r	s	t	v	x
P	F	b	c	d	e	f	g	h	i	l	m	n	o	p	q	r	s	t	v	x	a
Q	G	c	d	e	f	g	h	i	l	m	n	o	p	q	r	s	t	v	x	a	b
R	H	d	e	f	g	h	i	l	m	n	o	p	q	r	s	t	v	x	a	b	c
S	I	e	f	g	h	i	l	m	n	o	p	q	r	s	t	v	x	a	b	c	d
T	L	f	g	h	i	l	m	n	o	p	q	r	s	t	v	x	a	b	c	d	e
V	M	g	h	i	l	m	n	o	p	q	r	s	t	v	x	a	b	c	d	e	f
X	N	h	i	l	m	n	o	p	q	r	s	t	v	x	a	b	c	d	e	f	g
A	O	i	l	m	n	o	p	q	r	s	t	v	x	a	b	c	d	e	f	g	h
B	P	l	m	n	o	p	q	r	s	t	v	x	a	b	c	d	e	f	g	h	i
C	Q	m	n	o	p	q	r	s	t	v	x	a	b	c	d	e	f	g	h	i	l
D	R	n	o	p	q	r	s	t	v	x	a	b	c	d	e	f	g	h	i	l	m
E	S	o	p	q	r	s	t	v	x	a	b	c	d	e	f	g	h	i	l	m	n
F	T	p	q	r	s	t	v	x	a	b	c	d	e	f	g	h	i	l	m	n	o
G	V	q	r	s	t	v	x	a	b	c	d	e	f	g	h	i	l	m	n	o	p
H	X	r	s	t	v	x	a	b	c	d	e	f	g	h	i	l	m	n	o	p	q
I	A	s	t	v	x	a	b	c	d	e	f	g	h	i	l	m	n	o	p	q	r
L	B	t	v	x	a	b	c	d	e	f	g	h	i	l	m	n	o	p	q	r	s
M	C	v	x	a	b	c	d	e	f	g	h	i	l	m	n	o	p	q	r	s	t
N	D	x	a	b	c	d	e	f	g	h	i	l	m	n	o	p	q	r	s	t	v

Substitution poly-alphabétique : une même lettre peut être chiffrée de plusieurs manières.

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
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	26

M	A	T	H	E	M	A	T	I	Q	U	E	S
1	7	4	2	1	7	4	2	1	7	4	2	1
N	H	X	J	F	T	E	V	J	X	Y	G	T

```
sage: alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
sage: message = 'MATHEMATIQUES'

sage: def encode(m, cle):
          return join([alphabet[(alphabet.find(m[i])
   +cle[i % len(cle)]) % 26] for i in range(len(m))],'')

sage: encode(message, [1,7,4,2])
'NHXJFTEVJXYGT'
```

```
sage: def decode(m, cle):
          return join([alphabet[(alphabet.find(m[i])
    -cle[i % len(cle)]) % 26] for i in range(len(m))],'')

sage: decode('NHXJFTEVJXYGT', [1,7,4,2])
'MATHEMATIQUES'

sage: encode('NHXJFTEVJXYGT', [-1,-7,-4,-2])
'MATHEMATIQUES'
```

Cryptanalyse du chiffrement de Vigenère

Charles Babbage (1854) et Friedrich Wilhelm Kasiski (1863).

Déterminer la longueur de la clé :

```
sage: message = 'RIRILOULOUJEANJEAN'
sage: encode(message, [1,7,4,2])
'SPVKMVYNPBNGBUNGBU'

sage: message = 'LYCEEJEANDEPANGESARREGUEMINES'
sage: encode(message[0::4],[1]), encode(message[1::4],[7]),
      encode(message[2::4],[4]), encode(message[3::4],[2])
('MFOBTFNT', 'FQKUHNP', 'GIIKVYR', 'GCRGTGG')

sage: c = encode(message, [1,7,4,2])
sage: c[0::4], c[1::4], c[2::4], c[3::4]
('MFOBTFNT', 'FQKUHNP', 'GIIKVYR', 'GCRGTGG')
```

Analyse de fréquence

Une fois la longueur de la clé trouvée (ici 4), les sous-messages espacés de 4 lettres sont chiffrés par substitution mono-alphabétique (César)

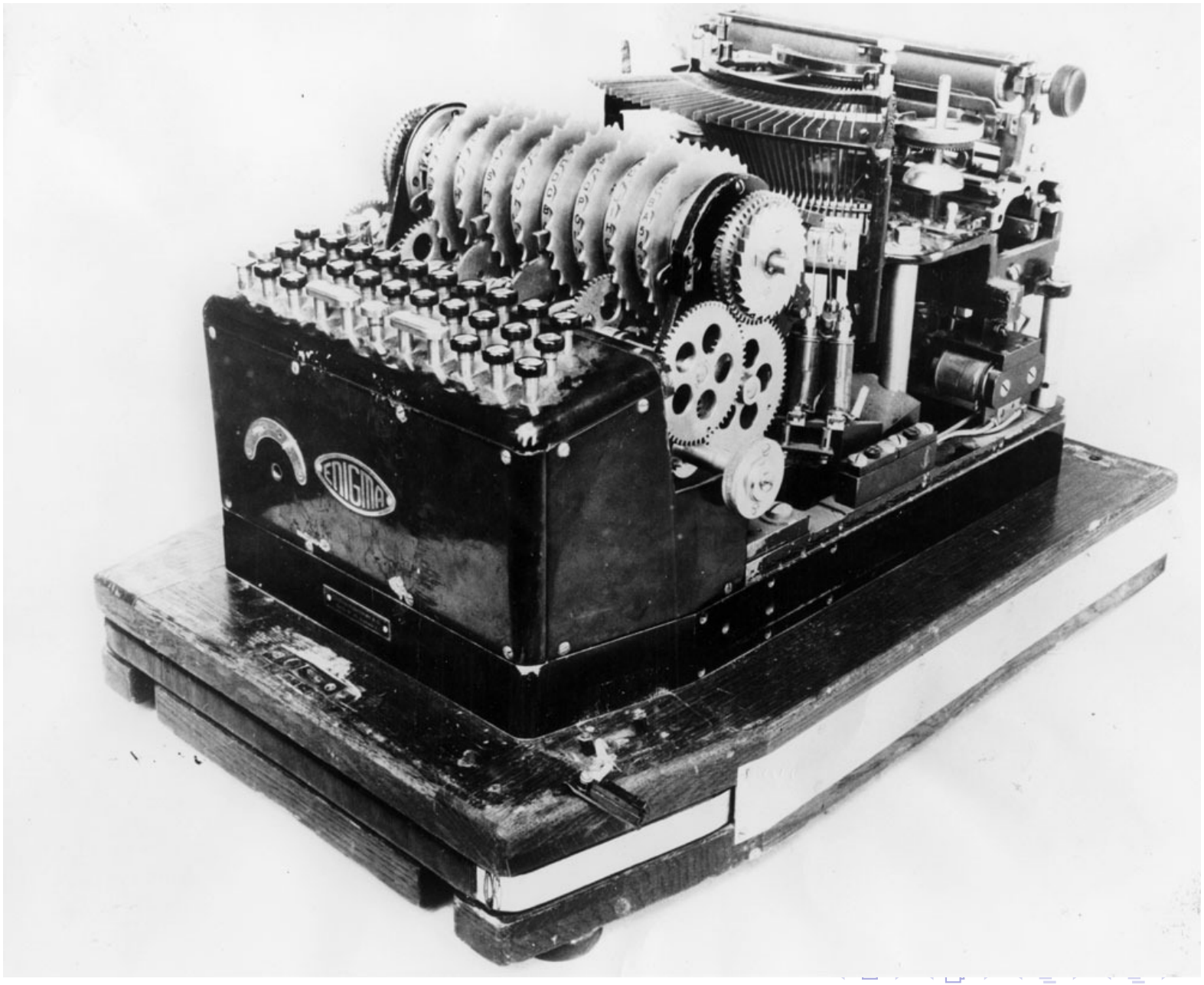
lettre	A	F	H	J	K	Q	U	Z
français	9.42	0.95	0,77	0.89	0.00	1.06	6.24	0.32
anglais	8.08	2.17	5.27	0.14	0.63	0.09	2.79	0.07

Cryptanalyse de Vigenère

Cryptanalyse à clair choisi :

```
sage: encode('AAAAAAAAAAAAAAAAAAAA', [1,7,4,2])
'BHECBHECBHECBHECBHE'
```

La machine Enigma (1919-1942)



Fonctionnement d'Enigma

L'entrée et la sortie sont les 26 lettres de l'alphabet.

- un tableau de connexion permet d'échanger au plus 6 paires de lettres
- des rotors (au plus 3) codent des permutations
- le premier rotor tourne d'un cran à chaque lettre tapée
- le second rotor tourne d'un cran après 26 lettres
- le troisième rotor tourne d'un cran après 26^2 lettres
- un réflecteur permute les lettres deux par deux, puis les fait traverser les rotors en sens inverse, puis le tableau de connexion

En tout plus de 10^{16} possibilités !

Principe de Kerckhoffs

La sécurité d'un protocole cryptographique ne doit pas reposer sur la confidentialité de l'algorithme, mais sur celle de la clé utilisée.

Illustration avec Enigma : la position des lettres sur les 3 rotors peut être connue (toutes les machines utilisent les mêmes rotors).

Seule la *position initiale* des rotors est secrète.

La cryptographie moderne

1976 : invention du concept de cryptographie à clé publique par Diffie et Hellman

1977 : adoption du standard DES (Data Encryption Standard), 2^{56} clés

1978 : invention du protocole RSA par Rivest, Shamir et Adleman (factorisation d'entier)

1985 : invention du protocole ElGamal (logarithme discret)

2001 : nouveau standard AES (*Advanced Encryption Standard*), avec des clés de 128 bits et plus

Le côté « obscur » de la force

1999 : distributed.net a cassé une clé DES en 22 heures et 15 minutes

2004 : collisions complètes trouvées dans MD5 (moins d'une minute)

2005 : faiblesses découvertes dans SHA-1 (1993)

L'algorithme RSA

Inventé par Rivest, Shamir et Adleman en 1978.

Premier protocole à clé publique connu.

Sécurité repose sur la factorisation d'entier :

$$15 \Longrightarrow 3 \times 5$$

RSA : fabrication de la clé

Alice choisit $n = p \cdot q$ où p et q sont premiers (seulement divisibles par 1 et eux-mêmes)

Exemple : $n = 77 = 7 \times 11$ avec $p = 7$ et $q = 11$

RSA : fabrication de la clé

Alice choisit $n = p \cdot q$ où p et q sont premiers (seulement divisibles par 1 et eux-mêmes)
Exemple : $n = 77 = 7 \times 11$ avec $p = 7$ et $q = 11$

Alice choisit e sans facteur commun avec $(p-1)(q-1)$
Ex. : $e = 13$, sans facteur commun avec $(p-1)(q-1) = 60$

RSA : fabrication de la clé

Alice choisit $n = p \cdot q$ où p et q sont premiers (seulement divisibles par 1 et eux-mêmes)
Exemple : $n = 77 = 7 \times 11$ avec $p = 7$ et $q = 11$

Alice choisit e sans facteur commun avec $(p-1)(q-1)$
Ex. : $e = 13$, sans facteur commun avec $(p-1)(q-1) = 60$

Alice calcule $d = 1/e \bmod (p-1)(q-1)$ [cf plus loin]
Ex. : $d = 1/13 \bmod 60 = 37$ $[13 \times 37 = 481 = 8 \times 60 + 1]$

RSA : fabrication de la clé

Alice choisit $n = p \cdot q$ où p et q sont premiers (seulement divisibles par 1 et eux-mêmes)
Exemple : $n = 77 = 7 \times 11$ avec $p = 7$ et $q = 11$

Alice choisit e sans facteur commun avec $(p-1)(q-1)$
Ex. : $e = 13$, sans facteur commun avec $(p-1)(q-1) = 60$

Alice calcule $d = 1/e \bmod (p-1)(q-1)$ [cf plus loin]
Ex. : $d = 1/13 \bmod 60 = 37$ $[13 \times 37 = 481 = 8 \times 60 + 1]$

Partie publique : n, e.
Ex. : $n = 77, e = 13$

RSA : fabrication de la clé

Alice choisit $n = p \cdot q$ où p et q sont premiers (seulement divisibles par 1 et eux-mêmes)
Exemple : $n = 77 = 7 \times 11$ avec $p = 7$ et $q = 11$

Alice choisit e sans facteur commun avec $(p-1)(q-1)$
Ex. : $e = 13$, sans facteur commun avec $(p-1)(q-1) = 60$

Alice calcule $d = 1/e \bmod (p-1)(q-1)$ [cf plus loin]
Ex. : $d = 1/13 \bmod 60 = 37$ $[13 \times 37 = 481 = 8 \times 60 + 1]$

Partie publique : n, e.
Ex. : $n = 77, e = 13$

Partie privée : p, q, d.
Ex. : $p = 7, q = 11, d = 37$.

Calcul de $d = 1/e \bmod (p-1)(q-1)$

Soit $k = (p-1)(q-1)$

On suppose que e et k sont premiers entre eux.

L'identité de Bézout affirme qu'il existe u et v tels que :

$$ue + vk = \mathrm{pgcd}(e, k) = 1$$

On peut calculer u et v par un calcul de pgcd étendu.

On a alors :

$$ue \equiv 1 \bmod k \quad \Longrightarrow \quad u = 1/e \bmod k$$

donc la clé secrète est $d = u$.

RSA : échange de message

Bob veut envoyer le message $0 < m < n$ à Alice.
Ex. : Bob veut envoyer la lettre « Z », qu'il code par $m = 26$.

Bob calcule :

$$c = m^e \bmod n$$

Ex. : $c = 26^{13} = 24811528732037365 76 \equiv 75 \bmod 77$.

Bob envoie c à Alice (ici $c = 75$)

Alice calcule :

$$m' = c^d \bmod n$$

Ex. : $m' = 75^{37} =$
$2383783149425184419351300062032972260084306981298141181468963623046875 \equiv$
$26 \bmod 77$

Pourquoi ça marche ?

$$m' = m^{de} \bmod n = m^{1+\lambda(p-1)(q-1)} \bmod n = m \bmod n$$

Comment calcule-t-on m^e mod n ?

Si m, e, n font 1024 bits, m^e a de l'ordre de $1024 \cdot 2^{1024}$ bits, soit environ $5.5 \cdot 10^{310}$ chiffres !

1. On fait tous les calculs modulo n :

$$
\begin{aligned}
26^2 &= 26 \times 26 = 60 \bmod 77 \\
26^3 &= 60 \times 26 = 20 \bmod 77 \\
26^4 &= 20 \times 26 = 58 \bmod 77 \\
26^5 &= 58 \times 26 = 45 \bmod 77 \\
26^6 &= 45 \times 26 = 15 \bmod 77 \\
26^7 &= 15 \times 26 = 05 \bmod 77 \\
26^8 &= 05 \times 26 = 53 \bmod 77 \\
26^9 &= 53 \times 26 = 69 \bmod 77 \\
26^{10} &= 69 \times 26 = 23 \bmod 77 \\
26^{11} &= 23 \times 26 = 59 \bmod 77 \\
26^{12} &= 59 \times 26 = 71 \bmod 77 \\
26^{13} &= 71 \times 26 = 75 \bmod 77
\end{aligned}
$$

Les calculs ne dépassent pas 77^2 : 12 multiplications

2. On utilise un algorithme d'exponentiation binaire :

$$m^{13} = (m^6)^2 \times m = ((m^3)^2)^2 \times m = ((m^2 \times m)^2)^2 \times m$$

$$\begin{array}{rcrcl} 26^2 & = & 26 \times 26 & = & 60 \bmod 77 \\ 26^3 & = & 60 \times 26 & = & 20 \bmod 77 \\ 26^6 & = & 20 \times 20 & = & 15 \bmod 77 \\ 26^{12} & = & 15 \times 15 & = & 71 \bmod 77 \\ 26^{13} & = & 71 \times 26 & = & 75 \bmod 77 \end{array}$$

Au lieu de 12, on ne fait plus que 5 multiplications (dont 3 carrés) !

Exponentiation binaire

La représentation de 13 en binaire est $(1101)_2$

Pour calculer m^{13}, on part de $x = m$ et on lit les bits de gauche à droite, sauf le « 1 » de tête :

- si on lit un « 0 » on effectue $x \leftarrow x^2 \bmod n$
- si on lit un « 1 » on effectue $x \leftarrow x^2 \bmod n$ puis $x \leftarrow xm \bmod n$

bit	opération	x
		m
1	$x \leftarrow x^2$	m^2
	$x \leftarrow xm$	m^3
0	$x \leftarrow x^2$	m^6
1	$x \leftarrow x^2$	m^{12}
	$x \leftarrow xm$	m^{13}

« Preuve » de l'exponentiation binaire

Soit e dont la représentation binaire est $(1 b_\ell \ldots b_1 b_0)$.

Supposons que l'algorithme calcule correctement m^e.

Alors $2e$ a comme représentation binaire $(1 b_\ell \ldots b_1 b_0 0)$

L'algorithme va calculer (correctement) $x = m^e$, puis la dernière étape sera $x \leftarrow x^2$ qui donne $(m^e)^2 = m^{2e}$.

De même $2e + 1$ a comme représentation binaire $(1 b_\ell \ldots b_1 b_0 1)$

L'algorithme va calculer (correctement) $x = m^e$, puis les dernières étapes sont $x \leftarrow x^2$ et $x \leftarrow xm$ qui donnent m^{2e} et m^{2e+1}.

Équivalence entre RSA et la factorisation

Étant donnés $n = p \cdot q$ et e, trouver d tel que

$$d = 1/e \bmod (p-1)(q-1)$$

est appelé le problème RSA.

Il est facile de voir que la factorisation de n permet de résoudre le problème RSA, car on connaît alors p et q, donc $(p-1)(q-1)$, et on en déduit d par un calcul de pgcd étendu.

Réciproquement, supposons qu'on connaisse d vérifiant :

$$d = 1/e \bmod (p-1)(q-1).$$

Pour tout entier $0 < a < n$, on a (petit théorème de Fermat) :

$$a^{ed-1} = 1 \bmod n$$

Soit $ed - 1 = 2^s t$ avec t impair.
On peut montrer qu'il existe i, $1 \leq i \leq s$, tel que

$$a^{2^{i-1}t} \not\equiv \pm 1 \bmod n \quad \text{et} \quad a^{2^i t} \equiv 1 \bmod n$$

pour au moins la moitié des valeurs de a.

Alors

$$\gcd(a^{2^{i-1}t} - 1, n)$$

est un facteur non trivial de n, à savoir p ou q.

Donc le problème RSA est équivalent à la factorisation.

Illustration

Supposons qu'on connaisse $e = 13$, $d = 37$ pour $n = 77$. On a $ed - 1 = 480$.

Soit $a = 2$: on vérifie que $2^{480} \equiv 1 \bmod 77$.

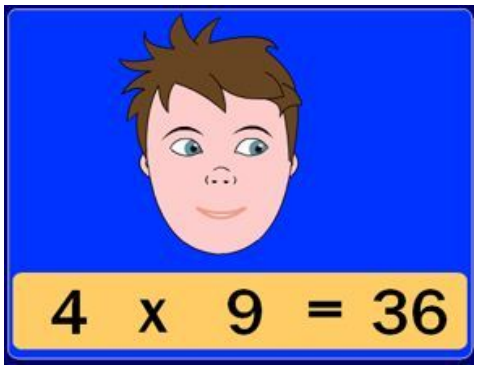
On a $2^{240} \equiv 1 \bmod 77$, $2^{120} \equiv 1 \bmod 77$, $2^{60} \equiv 1 \bmod 77$, $2^{30} \equiv 1 \bmod 77$, mais $2^{15} \equiv 43 \bmod 77$.

On calcule $\gcd(2^{15} - 1, 77) = \gcd(42, 77) = 7$.

Comment les ordinateurs calculent-ils ?

Comme nous :

$$
\begin{array}{r}
374 \\
\times\ 292 \\
\hline
748 \\
+3366\,. \\
+748\,.\,. \\
\hline
109208
\end{array}
$$



Comment les ordinateurs calculent (suite)

... sauf que les ordinateurs ont $2^{64} = 18446744073709551616$ doigts dans leur main !

Ils connaissent donc par cœur la table de multiplication $x \times y$ pour tout $0 \leq x, y < 2^{64}$.

Les « grands nombres » sont représentés en base 2^{64}, par exemple

$$(14945441618107492229 \quad 15730929240803800450)$$

représente

$$14945441618107492229 \cdot 2^{64} + 15730929240803800450$$

$$= 275694736597796474019892063930598192514$$

Comment les ordinateurs multiplient ?

Pour multiplier deux nombres de n chiffres chacun, la multiplication « classique » calcule n^2 produits intermédiaires :

$$
\begin{array}{cccccccccc}
 & & & & & 7 & 4 & 8 & 1 \\
 & & & & \times & 2 & 7 & 0 & 9 \\
\hline
 & & & & 6 & 7 & 3 & 2 & 9 \\
 & & & + & 0 & 0 & 0 & 0 & \\
 & & + & 5 & 2 & 3 & 6 & 7 & \\
 & + & 1 & 4 & 9 & 6 & 2 & & \\
\hline
 & 2 & 0 & 2 & 6 & 6 & 0 & 2 & 9
\end{array}
$$

Algorithme de Karatsuba (1962)

Groupons les chiffres par deux : $(74 \cdot 100 + 21)(27 \cdot 100 + 09) =$

$$(74 \times 27) \cdot 10^4 + (74 \times 09 + 21 \times 27) \cdot 100 + (21 \times 09)$$

$$(a_1 \cdot 100 + a_0)(b_1 \cdot 100 + b_0) =$$

$$(a_1 \times b_1) \cdot 10^4 + (a_1 \times b_0 + a_0 \times b_1) \cdot 100 + (a_0 \times b_0)$$

Le terme du milieu s'écrit :

$$a_1 \times b_0 + a_0 \times b_1 = (a_1 + a_0) \times (b_1 + b_0) - a_1 b_1 - a_0 b_0$$

On peut donc calculer aussi :

$$(74 \times 27) \cdot 10^4 + ((74 + 21) \times (27 + 09) - 74 \times 27 - 21 \times 09) \cdot 100 + (21 \times 09)$$

qui ne demande que 3 multiplications au lieu de 4 !

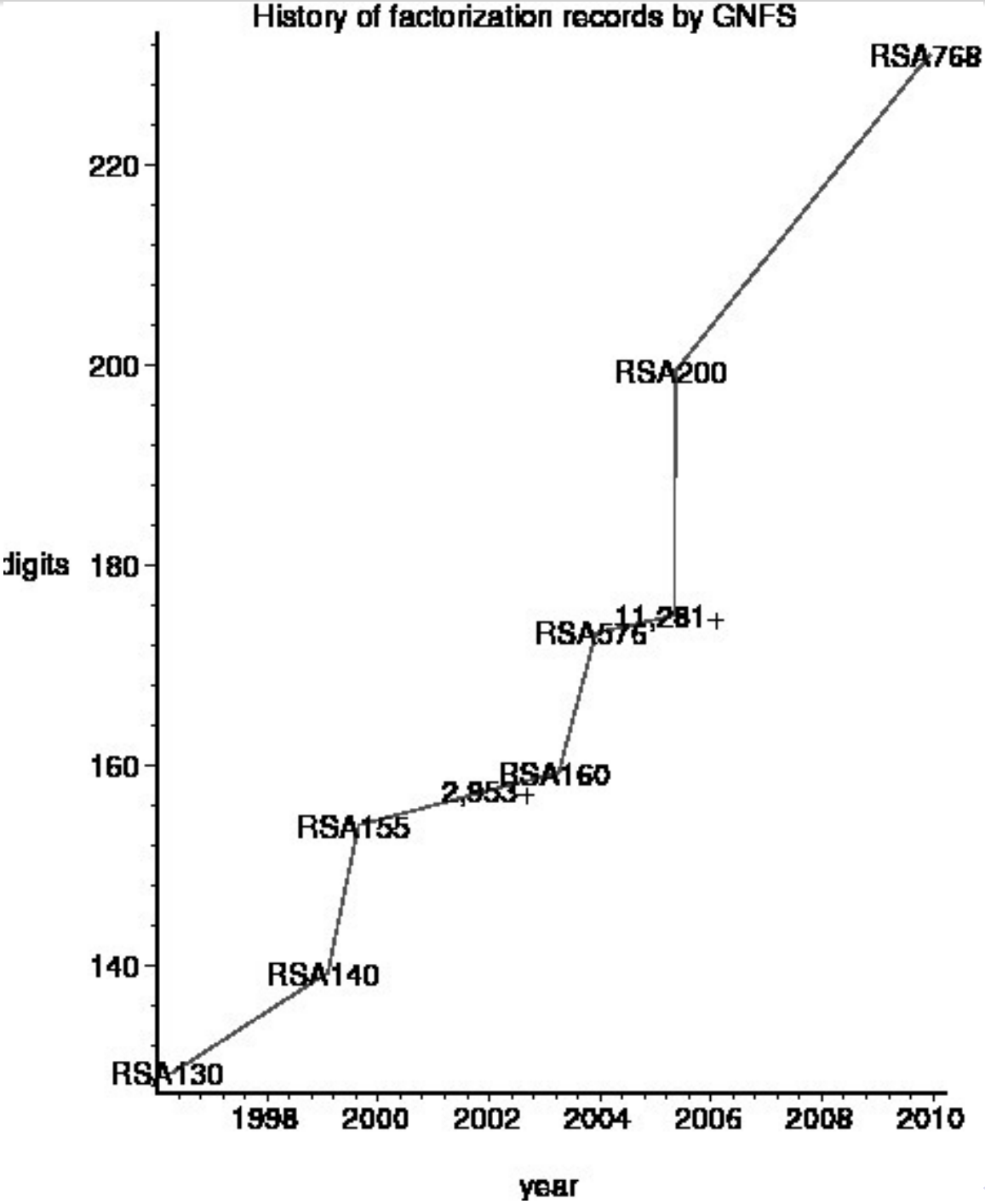
Cryptanalyse de RSA : factorisation d'entier par NFS

NFS = *Number Field Sieve* (crible algébrique en français)

Inventé par Pollard en 1988.

Utile pour factoriser un nombre RSA $n = pq$, produit de deux nombres premiers de même taille.

Record actuel : RSA-768, 232 chiffres (p et q de 116 chiffres).


History of factorization records by GNFS
RSA768
220
200
RSA200
digits
180
11,281+
RSA576
160
RSA160
2,953+
RSA155
RSA140
140
RSA130
1998
2000
2002
2004
2006
2008
2010
year

Factorisation de RSA-768

NTT : Kazumaro Aoki

EPFL : Joppe Bos, Thorsten Kleinjung, Arjen Lenstra, Dag Arne Osvik

Bonn : Jens Franke

CWI : Peter Montgomery, Andrey Timofeev

INRIA/LORIA/CARAMEL : Pierrick Gaudry, Alexander Kruppa, Emmanuel Thomé, PZ

Quelques chiffres

Environ 60 milliards de « relations ».

Temps utilisé : 1500 années cpu (2 années de temps « réel »).

Mémoire disque : 5 téra-octets.

Mémoire vive (RAM) : 1 téra-octet (maxi).

Exemple de relation

$F(104262663807, 271220)$ a 81 chiffres :

301114673492631466171967912486669486315616012885653409138028100146264068435983640

$$2^3 \cdot 3^2 \cdot 5 \cdot 1429 \cdot 51827 \cdot 211373 \cdot 46625959 \cdot 51507481$$
$$\cdot 3418293469 \cdot 4159253327 \cdot 10999998887 \cdot 11744488037 \cdot 12112730947$$

$G(104262663807, 271220)$ (42 chiffres) :

$$-350192248125072957913347620409394307733817$$

$$-1 \cdot 11 \cdot 1109 \cdot 93893 \cdot 787123 \cdot 9478097 \cdot 2934172201 \cdot 13966890601$$

RSA-768

Le 12 décembre 2009 :

RSA768 =
123018668453011775513049495838496272077285356959533479219732245215172640050726365751874520219978646938995647494277406384592519255732630345373154826850791702612214291346167042921431160222124047927473779408066535141959745985690214341 3
=
334780716989568987860441698482126908177047949837137685689124313889828837938780022876147116525317430877378144679994893
*
367460436667995904282446337996279526322791581643430876426760322838157396665112792333734171433968102700927987363089173

De la théorie à la pratique

Taille de clé recommandée pour RSA par l'ANSSI (Agence nationale de la sécurité des systèmes d'information) : *Référentiel Général de Sécurité, Annexe B1, version 1.20, 26 janvier 2010*,
`http://www.ssi.gouv.fr/IMG/pdf/RGS_B_1.pdf`

RègleFact-1

La taille minimale du module est de 2048 bits, pour une utilisation ne devant pas dépasser l'année 2020.

Taille de clé utilisée par le GIE Carte Bancaire : 960 bits (y compris pour des cartes valables jusque janvier 2014).

Comment devenir chercheur en informatique ?

Quelles études pour faire de la recherche en informatique ?

Qu'est-ce qu'une thèse ?

3 années de recherche dans un laboratoire, sous la direction d'un directeur de thèse. A l'issue de ces trois années, vous expliquez vos travaux dans un manuscrit et les défendez devant un jury expert du domaine (soutenance de thèse). On vous décerne alors le titre de **docteur ès informatique** (bac+8).

Inria
INVENTEURS DU MONDE NUMÉRIQUE

Quelques pointeurs

- le site Interstices (vulgarisation de problèmes de recherche)
- le site Fuscia (en particulier sujets de TPE/TIPE)