Mathématiques et Cryptographie

Paul Zimmermann
INRIA Nancy - Grand Est et LORIA

Colloque « Les mathématiques dans la société »
Académie Lorraine des Sciences
20 novembre 2010

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
...
```

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

(Ne pas confondre avec les machines à voter.)

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.

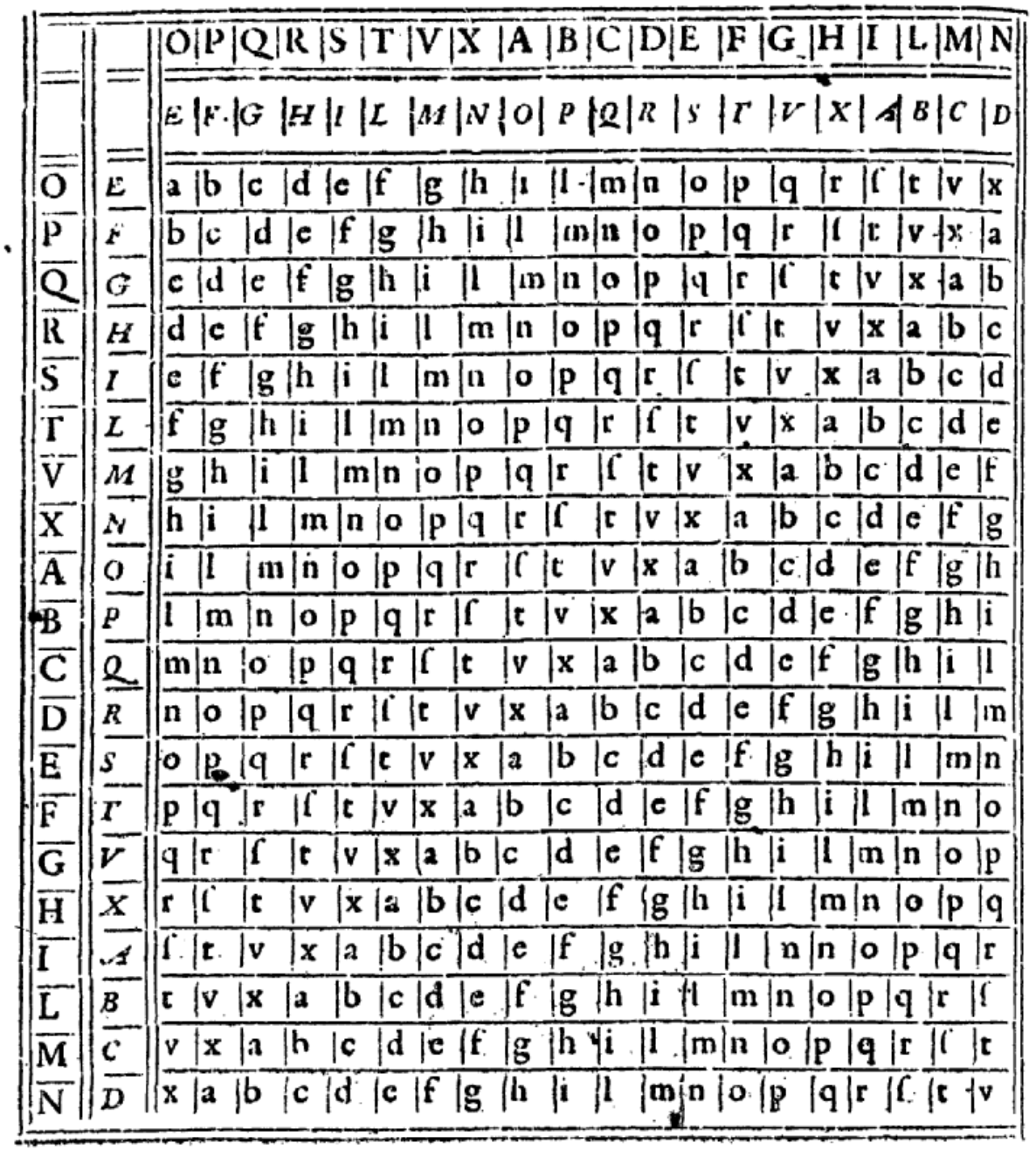
antes muerto que mudado

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 = 'ACADEMIELORRAINEDESSCIENCES'
sage: encode(message[0::4],[1]), encode(message[1::4],[7]),
      encode(message[2::4],[4]), encode(message[3::4],[2])
('BFMBEDD', 'JTVPLPL', 'EMVRWIW', 'FGTGUP')

sage: c = encode(message, [1,7,4,2])
sage: c[0::4], c[1::4], c[2::4], c[3::4]
('BFMBEDD', 'JTVPLPL', 'EMVRWIW', 'FGTGUP')
```

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)

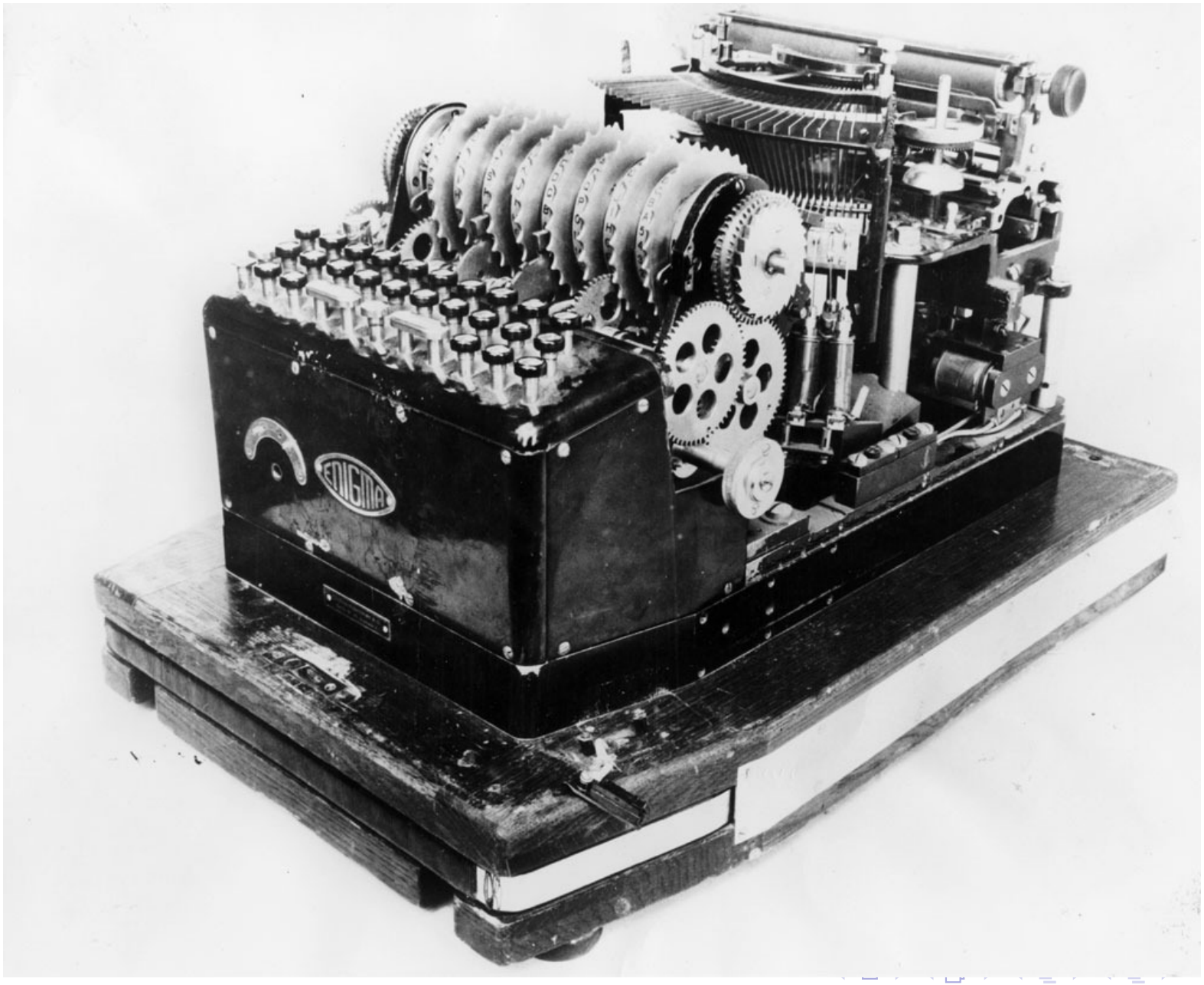
lettre	A	F	H	J	K	Q	U	Z
français	9.42	0.95	0,77	0.89	0.00	1.06	6.24	0.32
anglais	8.08	2.17	5.27	0.14	0.63	0.09	2.79	0.07

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)

Compétition SHA-3

novembre 2008 : 64 candidats initiaux

décembre 2008 : 51 qualifiés pour le 1er tour

juillet 2009 : 14 qualifiés pour le 2e tour (dont Shabal)

fin 2010 : sélection des finalistes, début du dernier tour

2012 : annonce du protocole retenu

L'algorithme RSA

Inventé par Rivest, Shamir et Adleman en 1978.

Premier protocole à clé publique connu.

Sécurité repose sur la factorisation d'entier.

RSA : fabrication de la clé

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

RSA : fabrication de la clé

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

Alice choisit $1 < e < (p-1)(q-1)$ sans facteur commun avec $(p-1)(q-1)$

RSA : fabrication de la clé

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

Alice choisit $1 < e < (p-1)(q-1)$ sans facteur commun avec $(p-1)(q-1)$

Alice calcule $d = 1/e \bmod (p-1)(q-1)$ [pgcd étendu]

RSA : fabrication de la clé

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

Alice choisit $1 < e < (p-1)(q-1)$ sans facteur commun avec $(p-1)(q-1)$

Alice calcule $d = 1/e \bmod (p-1)(q-1)$ [pgcd étendu]

Partie publique : n, e.

RSA : fabrication de la clé

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

Alice choisit $1 < e < (p-1)(q-1)$ sans facteur commun avec $(p-1)(q-1)$

Alice calcule $d = 1/e \bmod (p-1)(q-1)$ [pgcd étendu]

Partie publique : n, e.

Partie privée : p, q, d.

RSA : échange de message

Bob veut envoyer le message $0 < m < n$ à Alice.
Bob calcule :

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

Bob envoie c à Alice.

Alice calcule :

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

Or :

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

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 :

$$m^3 \bmod n = ((m^2 \bmod n) \cdot m) \bmod n$$

2. On utilise un algorithme d'exponentiation binaire :

$$m^5 = (m^2)^2 \cdot m$$

au lieu de :

$$m^5 = m \cdot m \cdot m \cdot m \cdot m$$

Exponentiation binaire

Soit à calculer m^{42}.

Exponentiation naïve : 41 multiplications.

Exponentiation binaire :

$$m_2 = m^2, m_4 = m_2^2, m_5 = m_4 \cdot m, m_{10} = m_5^2,$$

$$m_{20} = m_{10}^2, m_{21} = m_{20} \cdot m, m_{42} = m_{21}^2$$

Seulement 7 multiplications.

```
sage: 42.digits(base=2)
[0, 1, 0, 1, 0, 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, 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 :

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

Soit $ed - 1 = 2^s t$ avec t impair. On peut montrer que

$$a^{2^{s-1}t} \neq 1 \bmod n$$

pour au moins la moitié des valeurs de a. Alors

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

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

Donc le problème RSA est équivalent à la factorisation.

Factorisation par NFS

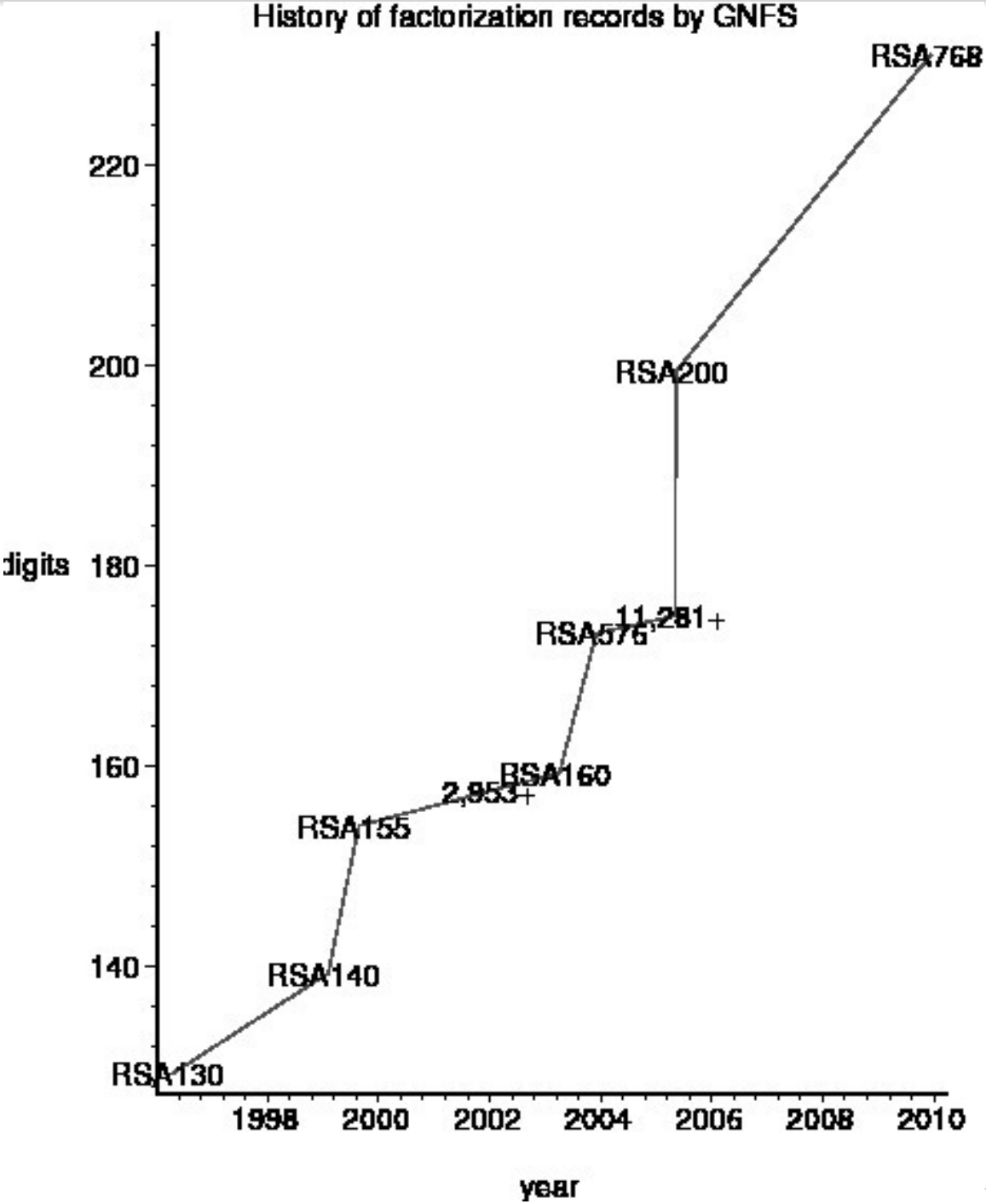
NFS = *Number Field Sieve* (crible algébrique en français)

Inventé par Pollard en 1988.

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

Complexité $e^{c(\log n)^{1/3}(\log\log n)^{2/3}}$.

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) :

$$-3501922481250729579133476204093943077338 17$$

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

RSA-768

Le 12 décembre 2009 :

RSA768 =
123018668453011775513049495838496272077285356959533479219732245215172640050726365751874520219978646938995647494277406384592519255732630345373154826850791702612214291346167042921431160222124047927473779408066535141959745985690214341 =
33478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489 *
36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917

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.

Taile de clé utilisée par le GIE Carte Bancaire : 960 bits (y compris pour des cartes fabriquées fin 2010).