

Cryptographie & Algorithmique de la Sécurité



Jean-Guillaume Dumas

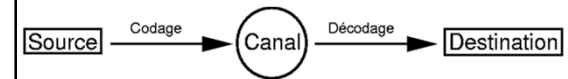


Références

<http://ljk.imag.fr/membres/Jean-Guillaume.Dumas>

- Histoire des codes secrets. *Simon Singh.*
- Théorie des codes : compression, cryptage, correction. *JGD, J-L. Roch, E. Tannier et S. Varrette. Dunod 2007.*
- Cryptographie Appliquée. *Bruce Schneier.*
- Handbook of Applied Cryptography. *A. J. Menezes, P. C. van Oorschot et S. A. Vanstone.*
<http://www.cacr.math.uwaterloo.ca/hac>
- Modern Computer Algebra. *J. von zur Gathen et J. Gerhard.*
- Cryptography, theory and practice. *Douglas Stinson.*

Notion de Code



- Le code doit répondre à différents critères :
 - Sécurité de l'information : cryptage, authentification, etc.
 - Rentabilité : compression des données
 - Tolérance aux fautes : correction/détection d'erreurs

La cryptologie

⇒ Science du secret

- Cryptographie : étude et conception des procédés de chiffrement des informations
 - Stéganographie (στεγανο-γραφην) : écriture couverte
Dissimule l'existence même d'information secrète
 - Cryptographie (κρυπτο-γραφην) : écriture secrète
Transforme un message clair en cryptogramme
- Cryptanalyse : analyse des textes chiffrés pour retrouver l'information dissimulée

Plan

- Introduction
 - Stéganographie
 - Cryptographie
 - Cryptanalyse
- Histoire des codes secrets
- Complexité, Algorithmique, Information
- Chiffrement par flots, générateurs pseudo-aléatoires
- Cryptographie à clefs secrètes
- Arithmétique des entiers et cryptographie à clef publique
- Signatures numériques, hachage
- Législation

... Stéganographie

- Information non-chiffrée
 - Connaissance de l'existence de l'information
 - =
 - Connaissance de l'information
- Exemples
 - Message couvert : tablette couverte de cire, crâne du messenger
 - Message invisible : encre sympathique (Pline 1^{er} siècle av. JC)
 - Message illisible : Micro-film sous forme de point
 - Message subliminal : traitement de texte des ministres de Mme Thatcher
- Théorie
 - Faible niveau de sécurité (cf. entropie)
 - En pratique ça marche ... (11/09/2001 ?)
 - Utilisée également pour le Watermarking (JPEG, MP3-MPEG, etc)

Cryptographie

- Information chiffrée
 Connaissance de l'existence de l'information
 \neq
 Connaissance de l'information
- Objectif
 - Permettre à Alice et Bob de communiquer sur un canal peu sûr
 - Réseau informatique, téléphonique, etc.
 - Oscar ne doit pas comprendre ce qui est échangé

Algorithmes de cryptographie

- Propriétés théoriques nécessaires :
 - Confusion
Aucune propriété statistique ne peut être déduite du message chiffré
 - Diffusion
Toute modification du message en clair se traduit par une modification complète du chiffré

Terminologie

- Texte clair
 - information qu'Alice souhaite transmettre à Bob
- Chiffrement
 - processus de transformation d'un message M de telle manière à le rendre incompréhensible
 - Fonction de chiffrement E
 - Génération d'un chiffre (message chiffré) $C = E(M)$
- Déchiffrement
 - processus de reconstruction du message clair à partir du message chiffré
 - Fonction de déchiffrement D
 - $D(C) = D(E(M)) = M$ (E est injective et D surjective)

Types de cryptographie

- En pratique E et D sont paramétrées par des clefs K_d et K_e
- Deux grandes catégories de systèmes cryptographiques
 - Systèmes à clefs secrètes (symétriques) : $K_e = K_d = K$
 - Systèmes à clefs publiques (asymétriques) : $K_e \neq K_d$
- Deux types de fonctionnement
 - Par flot : chaque nouveau bit est manipulé directement
 - Par bloc : chaque message est découpé en blocs

À quoi sert la cryptographie (CAIN...)?

- Confidentialité des informations stockées ou manipulées
 - Seuls les utilisateurs autorisés peuvent accéder à l'information
- Authentification des utilisateurs
 - L'utilisateur est-il ou non autorisé ? Pour quelle action ?
- Intégrité des informations stockées ou manipulées
 - Contre l'altération des données
- Non-répudiation des informations
 - Empêcher un utilisateur de se dédire
- Disponibilité des systèmes d'information
- Contrôle d'accès
- ...

Cryptanalyse : les menaces

- Attaque passive
 - Oscar se contente d'écouter le message
 - Menace la confidentialité de l'information :
 - Une information parvient également à une autre personne que son destinataire légitime

Cryptanalyse : les menaces

2. Attaque active

- Oscar peut modifier le contenu des messages échangés
- Menace l'intégrité de l'information :
 - L'information reçue est interprétée comme provenant d'une personne autre que son véritable auteur
 - Usurpation d'identité, altération des données, destruction, retardement, répétition (engorgement), répudiation, etc.

ALICE ← Messages → BOB

Niveaux d'attaque

- Étude de la sécurité des procédés de chiffrement

1. Texte chiffré connu : seul C est connu d'Oscar
2. Texte clair connu : Oscar a obtenu C et le M correspondant
3. Texte clair choisi : pour tout M, Oscar peut obtenir le C
4. Texte chiffré choisi : pour tout C, Oscar peut obtenir le M

- Garantir la confidentialité
 - Impossible de trouver M à partir de $C=E(M)$
 - Impossible de trouver la méthode de déchiffrement D à partir d'une séquence $\{M_1, \dots, M_n, E(M_1), \dots, E(M_n)\}$

Algorithmes d'attaque

1. Attaque exhaustive (par force brute)
 - Énumérer toutes les valeurs possibles de clefs
 - 64 bits $\rightarrow 2^{64}$ clefs = 1.844 10¹⁹ combinaisons
 - Un milliard de combinaisons/seconde \rightarrow 1 an sur 584 machines
2. Attaque par séquences connues
 - Deviner la clef si une partie du message est connu
 - ex: en-têtes de standard de courriels
3. Attaque par séquences forcées
 - Faire chiffrer par la victime un bloc dont l'attaquant connaît le contenu, puis on applique l'attaque précédente ...
4. Attaque par analyse différentielle
 - Utiliser les faibles différences entre plusieurs messages (ex: logs) pour deviner la clef

Plan

- Introduction
- Histoire des codes secrets
 - Sparte (5^{ème} siècle av JC)
 - Polybe (2^{ème} siècle av JC)
 - César (1^{er} siècle av JC), El Kindi (IX^{ème} siècle)
 - Vigenère (XVI^{ème} siècle), Babbage/Kasiski (XIX^{ème} siècle)
 - Enigma (1918)
 - Vernam (1917)
- Complexité, Algorithmique, Information
- Chiffrement par flots, générateurs pseudo-aléatoires
- Cryptographie par blocs, à clefs secrètes, à clefs publiques
- Signatures numériques, hachage
- Législation

Histoire des codes secrets

- Cryptographie Ancienne
 - Transposition Sparte (5^{ème} siècle av JC)
 - Substitution César (1^{er} siècle av JC), Vigenère (XVI^{ème} siècle),
- Cryptanalyse des codes mono et poly alphabétiques
 - El Kindi (IX^{ème} siècle)
 - Babbage/Kasiski (XIX^{ème} siècle)
- Mécanisation de la cryptographie et de la cryptanalyse
 - Enigma
 - Les bombes du Biuro Polonais et de Bletchley Park
- Vers un chiffrement parfait
 - Vernam, théorie de l'information

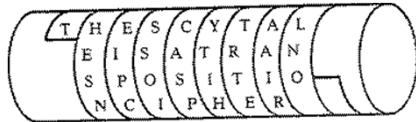
Cryptographie ancienne

```

graph TD
    A[Cryptographie] --> B[Transposition  
(mélange les lettres)]
    A --> C[Substitution]
    C --> D[Code  
(change les mots)]
    C --> E[Alphabet ou Chiffre  
(change les lettres)]
    
```

Transposition

- Chiffrement de type anagramme : mélange les lettres du message
- Sécurité théorique
 - Message de 35 lettres : 35! chiffreés possibles
- Problèmes
 - Confusion sur la syntaxe mais ... chaque lettre conserve sa valeur
 - Clé de chiffrement « complexe »
 - Ex: Scytale spartiate (5^{ème} siècle av JC)



Substitution

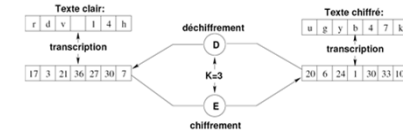
- Chiffrement en changeant d'alphabet
 - Kama sutra : mlecchita-vikalpà (art de l'écriture secrète, 4^{ème} siècle av JC)
- Sécurité théorique
 - Alphabet de 26 lettres : 26! alphabets possibles
- Problèmes
 - Confusion sur l'alphabet mais ... chaque lettre conserve sa place d'origine
 - Ex: Chiffrement de Jules César (1^{er} siècle av JC)

Alphabet clair : abcdefghijklmnopqrstuvwxyz
 Alphabet chiffré : DEFGHIJKLMNOPQRSTUVWXYZABC

Texte clair : errare humanum est, perseverare diabolicum
 Texte chiffré : HUUDUH KXPDXPHVW, SHUVHYHUUHGLDEROLFXP

Code de substitution de César

- Chiffrement par décalage avec K=3 :
 - $E_k(M) = M+K \text{ mod } n$
 - $D_k(M) = C-K \text{ mod } n$



- △ Seulement n façons différentes de chiffrer un message
 - Attaque brutale facile de nos jours
 - Réemployé par les officiers sudistes, sur les forums de news (ROT-13)

Cryptanalyse d'un code type Jules César

- « Mp iwX jegmpi hi fvmwiv gi qiwweki »
- Indices
 - Trois mots de deux lettres :
 - MP : la 2^{ème} lettre est trois plus loin ; OR ou IL
 - HI : la 2^{ème} suit la première ; DE ou TU
 - GI : la 2^{ème} est deux plus loin ; CE ou SU
 - Un seul doublon ww ne nous aide pas
 - On sait que l'algorithme est « Jules César »

• Deux indices suggèrent que $I \leftarrow E$, donc $n=4$

• « Il est facile de briser ce message »

Substitution monoalphabétique

Alphabet clair : abcdefghijklmnopqrstuvwxyz
 Alphabet chiffré : MOTSECRUVWXYZABDFGHJIKLNQP

Texte clair : l'erreur est humaine, y persévérer est diabolique
 Texte chiffré : YEGGEJG EHI UJZMVAE, P DEGHEKEGEG EHI SVMBOYVFJE

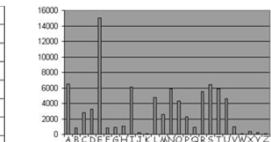
- 150 av JC, carré du grec Polybe : remplace H par ses coordonnées 23

	1	2	3	4	5		1	2	3	4	5	
Simple	1	A	B	C	D	E	1	D	I	F	C	L
	2	F	G	H	I	J	2	E	A	B	G	H
	3	K	L	M	N	O	3	J	K	M	N	O
	4	P	Q	R	S	T	4	P	Q	R	S	T
	5	U	V	X	Y	Z	5	U	V	X	Y	Z
							Avec mot de passe					

Cryptanalyse de la Substitution monoalphabétique

- Cryptanalyse par El-Kindi au IX^{ème} siècle
 - En français, 'e' apparaît le plus souvent

A	8.3944 %	N	7.5570 %
B	0.7669 %	O	5.3669 %
C	3.3297 %	P	3.2087 %
D	4.0699 %	Q	1.2613 %
E	14.5037 %	R	7.0209 %
F	1.2109 %	S	8.0091 %
G	0.9495 %	T	7.4775 %
H	0.7973 %	U	5.9808 %
I	8.1828 %	V	1.5791 %
J	0.6377 %	W	0.0067 %
K	0.0638 %	X	0.4098 %
L	5.8405 %	Y	0.3155 %
M	2.9355 %	Z	0.1240 %



Statistiques des fréquences de groupes de deux lettres : <http://mantis.free.fr/articles/freq.htm>

Briser un chiffre monoalphabétique (1/4)

- Soit le message
g cdbowaumzc ciu yd fswcd v miiyoco vci bsffydikbmuksdi icbyokumkoci iyo vci bmdmye dsd icbyokumkoci amo g yimzc vc g cdbowaumzc sd vzykic gc fcimzc asyo pyc bcyk bk dc aykiic ami cuoc octgc fdc ik gm uomdifiikisd m cuc kducobcaucc
- L'analyse fréquentielle donne:
a:2.98% b:4.26% c:15.74% d:5.96% e:0.43% f:2.98% g:2.98% h:0.0% i:9.36% j:0.00% k:5.96% l:0.00% m:6.38% n:0.00% o:6.38% p:0.43% q:0.00% r:0.00% s:2.98% t:0.43% u:4.68% v:2.13% w:1.28% x:0.00% y:5.53% z:2.13%
- Essayons avec c=>e
g edbowaumze elu yd fswed v miiyoco vei bsffydikbmuksdi icbyokumkoci iyo vei bmdmye dsd icbyokumkoci amo g yimze ve g edbowaumze sd vzykic ge feimze asyo pyc begyk bk de aykie ami euoe oetge fefe ik gm uomdifiikisd m eue kduebeauee

Briser un chiffre monoalphabétique (2/4)

- Un triplet « eue »
- trois possibilités: ère, été, Ève; t=7.47%, r=7.02%, v=1.57%
- essayons u => t
g edbowatmze eit yd fswed v miiyoco vei bsffydikbmuksdi icbyokumkoci iyo vei bmdmye dsd icbyokumkoci amo l yimze ve g edbowatmze sd vzykic ge feimze asyo pyc begyk bk de aykie ami etoe oetge fefe ik gm tomnifiikisd m ete kdtebeatee
- Trois mots de deux lettres finissant par e: ve, ge, de
- possibilités: ce, de, hé, le, me, ne
- dans le texte: d:5.96%; g:2.98%; v:2.13%;
- en français: n:7.55%; i:5.84%; d:4.07%; c:3.32%; m:2.93%; h:0.79%;
- Essayons d => n, g => l; v => d
l enbowatmze eit yn fswen d miiyoco dei bsffynkbmtksni icbyokumkoci iyo dei bmmnye nsn icbyokumkoci amo l yimze de l enbowatmze sn dezkyke le feimze asyo pyc begyk bk ne aykie ami etoe oetele fefe ik gm tomnifiikilsn m ete krtebeatee
- Considérons « eit »:
- le seul mot possible est "est", donc l => s

Briser un chiffre monoalphabétique (3/4)

- Nous avons donc maintenant
l enbowatmze est yn fswen d mssyoeo des bsffynkbmtksns sebyoktmkoes syo des bmmnye nsn sebyoktmkoes amo l ymzne de l enbowatmze sn dezkyke le feimze asyo pyc begyk bk ne aykse amz etoe oetele fefe sk im tomnifiiksskn m ete krtebeatee
- Parmi les lettres restantes du message, celles à haute fréquence sont
- m:6.38%, o:6.38%, k:5.96%, y:5.53%, a, f, s:2.98
- Parmi les lettres non identifiées, celles à haute fréquence sont
- a:8.39%, i:8.18%, r:7.02%, u:5.98%, o:5.37%
- Autres indices:
- lettre unique "m": la seule possibilité est "a"
- "o" pourrait donner "l" ou "r"; "etoe" ne peut être que "etre"
- l'hypothèse k=>l transforme sk en si; so et su ne sont pas des mots
- l'hypothèse y=>u transforme yn en un et syr en sur;
- Essayons avec m=>a, o=>r, k=>i et y=>u
l enbrwataze est un fswen d assurer des bsffynkbmtksns seburitaires sur des banaue nsn seburitaires aar l usaze de l enbrwataze sn dezuse le fessaze asur pue belui bi ne auisse aas etre retele fefe si la transmissin a ete interbeatee

Briser un chiffre monoalphabétique (4/4)

- Il nous reste un doublon "ff"
- si on retire les lettres déjà identifiées, les seules possibilités sont f=>m et f=>p
- fefe donnerait soit meme ou pepe (même ou pépé)
- Essayons avec f=>m
l enbrwataze est un mswen d assurer des bsmmnibatsns seburitaires sur des banaue nsn seburitaires aar l usaze de l enbrwataze sn dezuse le messaze asur pue belui bi ne auisse aas etre retele meme si la transmissin a ete interbeatee
- Une fois qu'on a un nombre suffisant de lettres identifiées, les autres peuvent l'être par simple inspection
- a=>p, z=>g, s=>o, w=>y, b=>c, p=>q, t=>v, e=>x
l encryptage est un moyen d assurer des communications securitaires sur des canaux non securitaires par l usage de l encryptage on deguise le message pour que celui ci ne puisse pas être revele meme si la transmission a ete interceptee

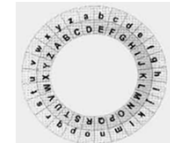
Substitution Monoalphabétique + Codes

- Marie Stuart Reine d'Ecosse (XVI^{ème} siècle)
Brisé par Thomas Phelippes (meilleur cryptanalyste d'Europe, parlant le français, l'italien, l'espagnol, le latin et l'allemand) qui réussit à briser par l'analyse des fréquences les messages chiffrés dans lesquels elle proposait d'assassiner la reine Elisabeth.
⇒ Substitutions polyalphabétiques ...



Substitutions polyalphabétiques

- Codes homophoniques (nomenclator)
- Plusieurs encodages possibles de chaque lettre
- Encodages par lettre proportionnelle à la fréquence de la lettre
⊙ Plus d'attaque fréquentielle sur les lettres
⊙ Attaques fréquentielles sur les di- ou tri-grammes
- car les lettres à faible fréquence sont toujours codées de la même manière
- Le disque de Leone Battista Alberti (1404-1472)
- Code de César avec changement de décalage régulier
- Une même lettre peut donc être codée par des caractères différents



Carré de Blaise de Vigenère (vers 1560)

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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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

Utilisation du tableau de Vigenère

- Une clef définit le décalage pour chaque lettre du message
 - Pour chaque paire, le caractère de la clef choisit une rangée du carré de Vigenère et le caractère du message choisit la colonne
 - Pour déchiffrer, c'est symétrique !

Clair	L	A	V	I	E	S	T	B	E	L	L	E	
Clef	B	O	N	J	O	U	R	B	O	N	J	O	U
Décalage	1	14	13	9	14	20	17	1	14	13	9	14	20
Chiffré	M	O	I	R	S	Y	J	U	P	R	U	Z	Y

- (A: décalage de 0, B:1, C:2, ..., Z:25) : S+R ≡ J [26]
- C'est symétrique : R-J ≡ S [26]

Trouver la longueur de la clef : Kasiski

- Charles Babbage 1854, Kasiski 1863
 - Si une même séquence se répète alors la distance entre les deux séquences est probablement un multiple de la taille de la clef
- OOBQB PQAIU NEUSR TEKAS RUMNA RRMNR ROPYO DEEAD ERUNR QLIJUG CZCCU NRTEU ARJPT MPAWU TNDQB GCCEM SOHKA RCMNB YUATM MDERD UQFWM DTFKI LROPY ARUOL FHYZS NUEQM NBFHG EILFE JXIEQ NAQEV QRREG PQARU NDXUC ZCCGP MZTFQ PMXIA UEQAF EAVCD NKQNR EYCFI RTAQZ ETQRF MDYOH PANGO LCD
- On repère les triplets ou quadruplets se répétant, puis la distance entre eux

PQA	150	DER	57	CZCC	114		
RTE	42	RUN	117	MNB	42		
ROPY	81	UNR	12	ARU	42	UEQ	54
- Pgcd(150,42,81,12,117,57,114,54) vaut 3 => longueur de la clé : 3

Trouver la longueur de la clef : Friedman

- Friedman 1920
 - Je tire une lettre au hasard dans un message : n_λ/n sont des a
 - Je tire une autre lettre au hasard à un autre endroit : $(n_\lambda-1)/(n-1)$ que ce soit un autre a
 - Probabilité que deux lettres tirées au hasard soient égales :

$$I_C = \sum_{\lambda=A}^Z \frac{n_\lambda(n_\lambda - 1)}{n(n-1)}$$

$$I_C = \sum_{\lambda=A}^Z f_\lambda \left(f_\lambda + \frac{f_\lambda - 1}{n-1} \right) \approx \sum f_\lambda^2$$

- Pour la langue française : $I_C(f) \approx 0.072786$
- Pour la langue anglaise : $I_C(e) \approx 0.065767$
- Pour un texte totalement aléatoire : $I_C(a) \approx 1/26 \approx 0.038462$

Trouver la longueur de la clef : Friedman

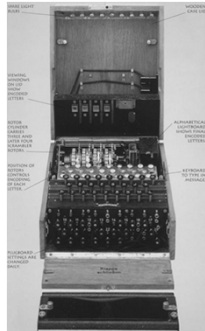
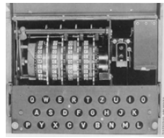
- Friedman 1920
 - Dans un texte français chiffré par Vigenère :
 - Pour une clef de longueur m
 - 1/m on a égalité des lettres du mot de passe
 - $I_C(f)$ que les lettres du clair soient aussi égales
 - 1-1/m les lettres du mot de passe sont distinctes
 - 1/26 d'avoir deux fois le même écart de lettres
- $\Rightarrow I_C(v) \approx 1/m I_C(f) + (1-1/m) 1/26$ avec m la taille de la clef
- On calcule $I_C(v)$ et on en déduit $m \approx (I_C(f) - I_C(a))/(I_C(v) - I_C(a))$
- Sur l'exemple, $I_C(v)=0.0477$
- Français $\rightarrow m=3.69$, Anglais $\rightarrow m=2.93$

Briser un code polyalphabétique

- Babbage/Kasiski
 - Si la clef est de longueur 3, analyse fréquentielle d'une lettre sur 3 !
 - Déchiffrement de 3 messages de type Jules César
- 1: OQQLITAUAMRYEDUQLUZITAPPJUDGEOAMYTDDFDKRYHYZUMFEFXOQOEUZGZXLIADQEFZTOMQAOQ
- 2: OBANSESMRNOOENLGCNERTATOCMHRNLMELWTOAQHSENHEINERGANUCPTREFVNNYIAERDHNL
- 3: BPIERKRNRRPDARRJCCRUMWNBCKSCKBAMRQMLPRLYNBGLJEAVRPRDCCMFMAQEKRCRQTFYPGC
- Friedman
 - Calcule la clef par analyse des Indices de coïncidences comparées !
 - Pour tous les décalages de k entre 0 et 25
 - Décaler chaque lettre de la première ligne de k : $L_i(k)$
 - Pour toutes les lignes i, Calculer $M_k = I_C(L_i(k)L_i)$
 - Le maximum d'une ligne de la matrice M donne le décalage entre la première et la i-ème lettre de la clef.
 - Au final, il reste seulement 26 clefs possibles, une seule d'entre elles doit donner un texte intelligible (ou avec des fréquences correctes)

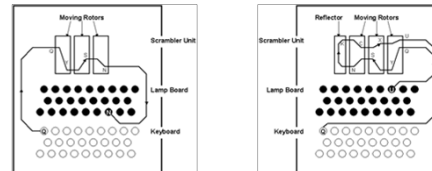
Mécanisation de la cryptographie

- Machine Enigma (Scherbius 1918)
 - Substitution polyalphabétique
 - Rotors : subs. Polyalph.
 - Connector/Reflector : substitution



Rotors et claviers

- Substitution polyalphabétique
 - 26 orientations pour chacun des 3 rotors : $26^3=17576$ alphabets
- Avec reflecteur : plus de possibilités ? ...
 - ⊙ si A->Q alors Q->A, cryptage et décryptage par la même configuration
- ... mais
 - ⊙ A ne peut jamais donner elle-même



+ Connexions à fiches

- Chacune des 6 connexions peut échanger deux lettres
 - $(26*25)/2 * (24*23)/2 ... * (16*15)/2 / (6!) = 100\,391\,791\,500$
- + échange des rotors : 123, 132, 213, 231, 312, 321
 - $100\,391\,791\,500 * 6 * 17576 \approx 10^{16} \approx 2^{53}$ clefs
- Pourquoi faire des rotors ?
 - Connexions à fiches : simple substitution
 - Les rotors tournent à chaque lettre : plus d'analyse de fréquence

La bombe de A. Turing

- 2^{53} clefs possibles
- Clef trouvée en une heure : comment ?



Biuro Szyfrów Polonais (1938)

- Réduction de l'ensemble des clefs
 - Réflecteur symétrise
 - Connexions et ordres changés chaque jour : orientation décidée à chaque message mais répétée deux fois !
 - Ex : AL-PR-TD-BW-KF-OY ; 231 ; QCW du jour
 - Envoi de PGHPGH → KIBVJE ; orientation devient PGH
 - Chaque jour une moisson de répétitions (pour chaque message)
 - 1^{ère} lettre : ABCDEFGHIJKLMNOPQRSTUVWXYZ
 - 4^{ème} lettre : FQHPLWOGBMVRXUYCZITNJEASDK
 - Orbites : A→F→W→A ; B→Q→Z→K→V→E→L→R→I→B
 - La longueur des orbites ne dépend pas des connexions !
 - Dictionnaire des 17576*6 suites d'orbites en 1 an
 - Ensuite, seules 12 des 26 lettres sont brouillées.

Les bombes

- Passage à 5 rotors différents dont seuls 3 sont utilisés
- + 10 connexions : 2^{68} clefs
 - Et $17576*(6*4*5)$ au lieu de 105456 entrées du dictionnaire
 - 20 ans !??
- Autres réductions par A. Turing (Bletchley park), ...
 - À l'aide de mots probables
 - Ex: opérations à un endroit fixé par les Anglais
 - Sachant qu'aucune lettre n'est cryptée par elle-même
- Mécanisation par A. Waterson (BTM), ...
 - Des dictionnaires réduits
 - Du déchiffrement

Vers un chiffrement parfait

- One-time-pad, Vernam (USA 1917)
 - $C = M \oplus K$ (xor bit à bit)
 - Téléphone rouge
- Vigenère avec : Longueur du mot-cléf = longueur du message !
 - Confusion totale : chiffrement de « aaaa...aaa » aléatoire
 - Diffusion totale : si le mot-cléf n'est *jamais* réutilisé
- Cryptanalyse possible uniquement si
 - Mot-cléf trivial
 - Réutilisation du mot-cléf : $(M_1 \oplus K) \oplus (M_2 \oplus K) = M_1 \oplus M_2$!!!
 ⇨ Alors des morceaux de textes en clair donnent $M_1 \oplus M_2 \oplus N \approx M_1$
- Seul code aujourd'hui Mathématiquement prouvé sûr (cf. entropie) ...

Plan

- Introduction
- Histoire des codes secrets
- Cryptographie moderne
 - Théorie de l'information
 - Théorie des codes
 - Théorie de la complexité, arithmétique
- Chiffrement par flots, générateurs pseudo-aléatoires
- Cryptographie par blocs, à clefs secrètes, à clefs publiques
- Signatures numériques, hachage
- Législation

Cryptographie moderne

- Principes de Auguste Kerckhoffs (1883)
 1. La sécurité repose sur le secret de la clef et non sur le secret de l'algorithme
 - Canal +, Cartes Bleues (LUHN-10) !!!
 2. Le déchiffrement sans la clef doit être impossible (à l'échelle humaine)
 3. Trouver la clef à partir du clair et du chiffré est impossible (à l'échelle humaine)

Théorie de l'information

- Claude Shannon 1948
- Entropie
 - Quantité d'information : $I(p) = \log_2(1/p)$
 - Entropie : quantité moyenne d'information contenue dans un message
 - Nombre de questions binaires pour déterminer la valeur d'un dé
 - Dé normal : 3 questions = $\lceil \log_2(6) \rceil \approx 2.585$
 - Dé pipé : le 1 une fois sur deux et les autres une fois sur 10
 - * $\frac{1}{2} * 1 + \frac{1}{2} * 4 \approx 2.5$
 - * $\frac{1}{2} * 1 + \frac{1}{2} * (2^5 * 3 + 3^5 * 4) \approx 2.3$
 - * $\frac{1}{2} * \log_2(2) + 5 (1/10 \log_2(10)) \approx 2.161$
 - $H(M) = \sum_p p \log_2(1/p)$
 - L'entropie est maximale lorsque toutes les probabilités sont égales
 Ex: Si l'apparition des lettres est exactement aléatoire, il est impossible d'appliquer l'attaque fréquentielle

Théorie des codes : théorèmes de Shannon 1948

- Compression : « Pour toute source X d'entropie $H(x)$ on peut trouver un code dont la longueur moyenne s'approche de $H(X)$ d'aussi prêt que l'on veut »
 - Algorithme d'Huffman, extensions de sources
- Correction d'erreurs : « Pour tout canal on peut toujours trouver une famille de codes dont la probabilité d'erreur après décodage tend vers 0 »
- Cryptage : « si un chiffrement est parfait, alors il y a au moins autant de clefs que de messages »
 - Un cryptanalyste doit obtenir de l'ordre de $H(M)$ informations pour retrouver M

Probabilité et Chiffrement

- $P(X=x)$: probabilité que X vaille x
- $P(X=x, Y=y)$: probabilité jointe que $X=x$ ET $Y=y$
 - Indépendance ssi $P(X=x, Y=y) = P(X=x)P(Y=y)$
- $P(X=x|Y=y)$: probabilité conditionnelle que $X=x$ sachant $Y=y$
 - $P(X=x, Y=y) = P(Y=y) P(X=x|Y=y) = P(X=x) P(Y=y|X=x)$ [Bayes]
 - Indépendance ssi $P(X=x|Y=y) = P(X=x)$
- M l'ensemble des messages en clair
- K l'ensemble des clefs possibles
- C l'ensemble des cryptés possibles, $C(k) = \{E_k(m), m \in M\}$
 - $P(C=c) = \sum P(K=k) P(M=D_k(c))$ pour tout $k \in K$ telle que $c \in C(k)$
 - $P(C=c|M=m) = \sum P(K=k)$ pour tout $k \in K$ telle que $m=D_k(c)$

Informations données par le chiffré

- $P(M=m|C=c) = P(C=c|M=m)P(M=m)/P(C=c)$
 - $P(M=a) = 1/4$; $P(M=b) = 3/4$
 - $P(K=k_1) = 1/2$; $P(K=k_2) = P(K=k_3) = 1/4$
- $P(C=1)=1/8$; $P(C=2) = 7/16$; $P(C=3) = 1/4$; $P(C=4) = 3/16$
- $P(M=a|C=1)=1$; $P(M=a|C=2)=1/7$; $P(M=a|C=3)=1/4$; $P(M=a|C=4)=0$
- $P(M=b|C=1)=0$; $P(M=b|C=2)=6/7$; $P(M=b|C=3)=3/4$; $P(M=b|C=4)=1$

$e_k(m)$	a	b
k_1	1	2
k_2	2	3
k_3	3	4

- La connaissance du chiffré donne une information sur le message
- Un chiffrage parfait ne devrait donner aucune information
 - (1) : $P(M=m|C=c) = P(M=m) \iff$ (2) : $P(C=c|M=m) = P(C=c)$
 - (2)>0 \rightarrow Pour m donné, $\forall c$, \exists au moins une clef pour laquelle $E_k(m) = c$
 - \rightarrow Chiffre parfait : $|K| \geq |C|$

Vernam est un code parfait

- $|K|=|M|=|C|$, les clefs sont équiprobables, $c = m \oplus k$
- Parfait ssi $P(M=m|C=c) = P(M=m)$
- Or $P(C=c) = \sum P(K=k) P(M=D_k(c))$
 - $= \sum 1/|K| P(M=D_k(c))$ *équadistribution*
 - $= 1/|K| \sum P(M=m)$ *bijectif*
 - $= 1/|K|$
- Également $P(C=c|M=m) = P(K=m \oplus c)$ *bijectif*
 - $= P(K=k) = 1/|K|$
- Donc : $P(M=m|C=c) = P(C=c|M=m)P(M=m)/P(C=c)$
 - $= P(M=m)$ *[Bayes]*

Entropie dans un Code secret symétrique

- Information moyenne contenue dans l'événement $M=m_i$
 - $H(M) = -\sum P(M=m_i) \log_2(P(M=m_i))$
 - Nombre moyen de questions à poser pour retrouver m_i
 - Perte d'entropie équivaut à un gain d'information
- $H(K,M) = -\sum \sum P(K=k,M=m) \log(P(K=k,M=m))$
- $H(K,M) \leq H(K) + H(M)$
 - égalité ssi Indépendance de K et M
 - [Bayes] : $H(X,Y) = H(Y) + H(X|Y) = H(X) + H(Y|X)$
- $H(M|K,C) = 0 = H(C|M,K)$ par le cryptage et le décryptage
- Donc $H(K|C) = H(K)+H(M)-H(C)$

Chiffrement parfait et entropie

- Chiffrement parfait :
 - Le chiffrement est parfait ssi le chiffré ne fournit aucune information sur M : $H(M|C) = H(M)$
 - \rightarrow Si $|K| < |M|$, le chiffrement n'est pas parfait
 - \rightarrow L'entropie d'un chiffrement est fonction de la taille des clefs
- Vernam, si K est vraiment aléatoire sur toute la longueur de M, alors $H(C) = H(M \oplus K) = H(M)$, et Vernam est parfait
- \rightarrow Plus l'entropie d'un chiffrement est grande, plus le chiffré ressemble à de l'aléatoire, plus la recherche exhaustive est difficile

Entropie : Mesure de l'imperfection d'un code

- Distance d'unicité $d =$ Nombre moyen d de lettres du chiffré nécessaires pour trouver la clef : $H(K|C_1...C_d) = 0$
 - Or on a toujours $H(X,Y) = H(Y) + H(X|Y)$ *[Bayes]*
 - $H(K|C_1...C_d) = H(K, C_1...C_d) - H(C_1...C_d) = H(K, M_1...M_d) - H(C_1...C_d)$
 - La clef est indépendante du message donc
 - $H(M_1...M_d) + H(K) - H(C_1...C_d) = 0$
 - Pour une suite aléatoire de lettres, $H(X_1...X_d) = d \log_2(26) = 4.7 d$
 - Pour un texte en français $H(M_1...M_d) \approx 3.97 d$
 - Pour un texte en anglais $H(M_1...M_d) \approx 4.19 d$
 - Dans un chiffré : $H(\text{langue}) \leq H(C_1...C_d) \leq H(\text{aléatoire})$
- $\Rightarrow d \geq H(K)/(4.7-3.97)$
- Exemple : la clef est une permutation de l'alphabet
 - $H(K) = -\sum_{k=1}^n P(K=k) \log_2(P(K=k)) \approx \log_2(26!) \approx 88.38$
 - $88.38 / (4.67-3.97) = 126.26$
 - \rightarrow Il faut au moins 127 lettres françaises pour trouver la clef

Utilisation : Faible niveau de sécurité de la Stéganographie

[R. Anderson, F. Petitcolas, 1998] On the limits of Steganography

- S stégotexte, M texte de couverture, X informations
 - $H(S)=H(M)+H(X)$
- Attention à la quantité d'information du chiffre
 - détection simple de stéganographie si l'entropie du stégotexte est trop élevée ...
- Ensemble de textes de couverture M, ensemble d'encodages E codant de l'information
 - Capacité de stéganographie $\leq H(E)-H(M)$

Plan

- Introduction
- Histoire des codes secrets
- Théorie de l'information
- Cryptographie symétrique
 - Chiffrement par flots (bit à bit) / par blocs
 - Générateurs pseudo-aléatoires
 - Cryptographie à clefs secrètes
 - » DES
 - » AES
- Complexité, Cryptographie asymétrique, clefs publiques/privées
- Signatures numériques, hachage
- Législation

Complexité et cryptographie

- Niveau de complexité d'une attaque
 - Comparer avec la recherche exhaustive
- Chiffrement idéal (moins que parfait !)
 - L'implémentation est possible : complexité polynomiale au pire
 - Mais toutes les attaques sont de complexité exponentielle au mieux
- Chiffrement sûr
 - Toutes les attaques *connues* sont de complexité exponentielle
- Chiffrement pratique
 - Attaquer coûte plus cher (machines, ...) que la valeur du secret
 - Attaquer prend plus de temps que la validité du secret

Cryptographie moderne

- Principes de Auguste Kerckhoffs (1883)
 1. La sécurité repose sur le secret de la clef et non sur le secret de l'algorithme
 - Canal +, Cartes Bleues !!!
 2. Le déchiffrement sans la clef doit être impossible (à l'échelle humaine)
 3. Trouver la clef à partir du clair et du chiffré est impossible (à l'échelle humaine)

Chiffrement par flot (stream cipher)

- Chiffrement à la one-time-pad
 - Taille du message M : n
 - Avec une petite clef K, générer K' de taille n
- Sécurité
 - Substitution rapide (xor)
 - Génération de clef
 - Fonction pseudo-aléatoire : devrait être impossible à prédire à l'échelle humaine
 - Principe de Kerckhoffs : la sécurité repose sur la clef

Générateurs Hardware

- Registres linéaires à décalage (LFSR)
- Registres non-linéaires à décalage (NLFSR)

Bluetooth 1/2

- Sécurisation des communications radio entre unité centrale et périphériques : 4 LSFR puis une fonction f

$$X^{25} + X^{20} + X^{12} + X^8 + 1$$

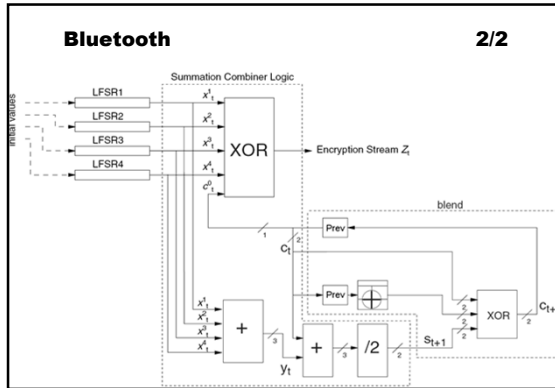
$$X^{31} + X^{24} + X^{16} + X^{12} + 1$$

$$X^{33} + X^{28} + X^{24} + X^4 + 1$$

$$X^{39} + X^{36} + X^{28} + X^4 + 1$$

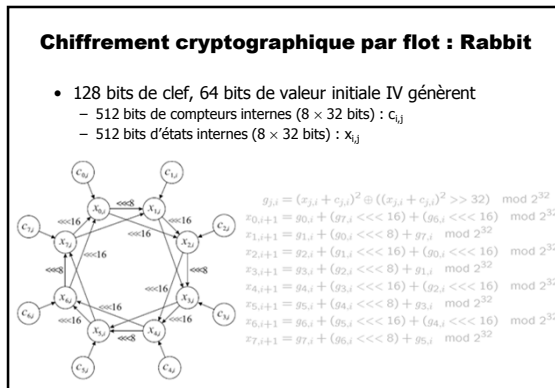
f

⇒ Registre à 25+31+33+39=128 bits
 ⇒ Période : $\text{ppcm}(2^{25}-1, 2^{31}-1, 2^{33}-1, 2^{39}-1) \approx 2^{125}$



- ### Registres à décalage
- Implémentation facile et rapide en hardware
 - Période : n registres, $T \leq 2^n - 1$
 - Sûr : RC4 avec clef de 256 bits, 2^{1700} états
 - Rapide : 10 fois plus rapide que DES par exemple
 - LFSR cassés par l'algorithme de Berlekamp-Massey 1969
 - Polynômes générateurs
 - Prévoient les prochains bits à partir des premiers
 - NLFSR, Problème : fonction f généralement secrète en contradiction avec le principe de Kerckhoffs
 - RC4, variante des NLFSR, f publiée en 1994

- ### Berlekamp-Massey
- Séquence de bits de clefs $b_0, b_1, b_2, b_3, \dots$
 - Trouver les c_i tels que $c_0 b_t = c_1 b_{t-1} + \dots + c_d b_{t-d}$
 - Polynôme annulateur LFSR $c_d X^d + \dots + c_1 X^1 + c_0$
 - Algorithme : Mise à jour de polynômes annulateurs
 1. $P(X) = 1 - b_1/b_0 X \rightarrow 1 * b_1 = b_1/b_0 * b_0$
 2. Défaut : $P(X)$ sur b_0, b_1 et $b_2 \rightarrow \delta_2 = b_2 - b_1/b_0 b_1$
 3. Augmenter $P(X)$ pour annuler b_0, b_1 et b_2
 4. $Q(X) = (1 - b_1/b_0 X) - \delta_2/b_0 X^2$
 5. Défaut : $P(X)$ sur $b_3 \rightarrow \delta_3 = Q[b_1, b_2, b_3]$
 6. $P(X) = Q(X) - \delta_3/b_3 P(X) X$
 7. etc.
 - Complexité $O(d^2)$ seulement !



- ### Rabbit : compteurs
- | | |
|--|--------------------|
| $c_{0,i+1} = c_{0,i} + a_0 + \phi_{7,i} \pmod{2^{32}}$ | $a_0 = 0x4D34D34D$ |
| $c_{1,i+1} = c_{1,i} + a_1 + \phi_{0,i+1} \pmod{2^{32}}$ | $a_1 = 0xD34D34D3$ |
| $c_{2,i+1} = c_{2,i} + a_2 + \phi_{1,i+1} \pmod{2^{32}}$ | $a_2 = 0x34D34D34$ |
| $c_{3,i+1} = c_{3,i} + a_3 + \phi_{2,i+1} \pmod{2^{32}}$ | $a_3 = 0x4D34D34D$ |
| $c_{4,i+1} = c_{4,i} + a_4 + \phi_{3,i+1} \pmod{2^{32}}$ | $a_4 = 0xD34D34D3$ |
| $c_{5,i+1} = c_{5,i} + a_5 + \phi_{4,i+1} \pmod{2^{32}}$ | $a_5 = 0x34D34D34$ |
| $c_{6,i+1} = c_{6,i} + a_6 + \phi_{5,i+1} \pmod{2^{32}}$ | $a_6 = 0x4D34D34D$ |
| $c_{7,i+1} = c_{7,i} + a_7 + \phi_{6,i+1} \pmod{2^{32}}$ | $a_7 = 0xD34D34D3$ |
- $$\phi_{j,i+1} = \begin{cases} 1 & \text{si } c_{0,i} + a_0 + \phi_{7,i} \geq 2^{32} \text{ et } j = 0 \\ 1 & \text{si } c_{j,i} + a_j + \phi_{j-1,i+1} \geq 2^{32} \text{ et } j > 0 \\ 0 & \text{sinon} \end{cases}$$

- ### Rabbit : extraction
- | | |
|--|---|
| $s_i^{[15..0]} = x_{0,i}^{[15..0]} \oplus x_{5,i}^{[31..16]}$ | • Chiffrement
- $c_i = m_i \oplus s_i$ |
| $s_i^{[31..16]} = x_{0,i}^{[31..16]} \oplus x_{3,i}^{[15..0]}$ | |
| $s_i^{[47..32]} = x_{2,i}^{[15..0]} \oplus x_{7,i}^{[31..16]}$ | • Déchiffrement
- $m_i = c_i \oplus s_i$ |
| $s_i^{[63..48]} = x_{2,i}^{[31..16]} \oplus x_{5,i}^{[15..0]}$ | |
| $s_i^{[79..64]} = x_{4,i}^{[15..0]} \oplus x_{1,i}^{[31..16]}$ | • Débit
- 3.7 cycles/bit |
| $s_i^{[95..80]} = x_{4,i}^{[31..16]} \oplus x_{7,i}^{[15..0]}$ | |
| $s_i^{[111..96]} = x_{6,i}^{[15..0]} \oplus x_{3,i}^{[31..16]}$ | |
| $s_i^{[127..112]} = x_{6,i}^{[31..16]} \oplus x_{1,i}^{[15..0]}$ | |

Autres générateurs

- Générateurs d'entropie : Linux /dev/(u)random
 - Regroupement de bruit de l'environnement, non reproductibles :
 - Clavier, souris, autres périphériques
 - Vitesse de rotation du disque
 - Interruptions
- Générateurs logiciels : faire des suites aléatoires
 - Faire de la théorie
 - Knuth : 10 générateurs, itérés et choisis au hasard
 - Boucle après 7401 nombres, période 3178 !!!
 - Générateurs congruentiels linéaires (rand, etc.)
 - $X_{n+1} = (aX_n + c) \text{ mod } M$
 - ⊙ Période au plus M-1, mais M aussi grand que l'on veut
 - ⊙ Non adaptés à la cryptographie : facilement cassables
 - À partir de chiffres DES, RSA ...
 - Récupérer des bits dans des chiffrements successifs
 - $X_{n+1} = E(X_n)$: b_n = premier bit de X_n
 - Sécurité dépend du chiffre utilisé

Générateurs congruentiels linéaires

- $X_{n+1} = (aX_n + c) \text{ mod } M$
 - ⊙ Période au plus M-1, mais M aussi grand que l'on veut
 - ⊙ Pour $M = p_1^{e_1} \dots p_k^{e_k}$ la période maximale est $\lambda(M) = \text{lcm}(2^{e_1-2}, p_1^{e_1-1}(p_1-1), \dots)$ pour a un élément primitif
 - ↳ Extensions : $X_n = a_1 X_{n-1} + \dots + a_k X_{n-k} [p]$
 - ⊙ Période p^k-1 ssi $X^k - a_1 X^{k-1} - \dots - a_k$ primitif
- ⚠ Attention aux propriétés statistiques
 - ⊙ 950706376 $X_n [2^{31}-1]$: bonnes propriétés d'uniformité [Fishman-Moore]
 - ⊙ Non adaptés à la cryptographie : facilement cassables
 - $(x_{n+1} - x_n)^2 \equiv (x_{n+2} - x_{n+1})(x_n - x_{n-1})$: permet de retrouver M
 - $\begin{bmatrix} x_n & 1 \\ x_{n+1} & 1 \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix} \equiv \begin{bmatrix} x_{n+1} \\ x_{n+2} \end{bmatrix} [m]$ retrouve a et c !

Générateurs logiciels cryptographiques

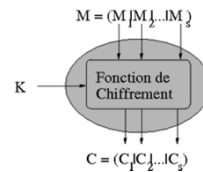
- Blum-Blum-Shub
 - p et q premiers congrus à 3 modulo 4
 - $X_{i+1} \equiv X_i^2 [pq]$
 - ↳ Casser la suite du bit de poids faible est aussi difficile que la factorisation
 - Généralisation à partir d'un code : exemple de RSA
 - n, e, d clefs RSA
 - $X_{i+1} \equiv X_i^e [n]$
 - ↳ Découvrir les $O(\log(\log(n)))$ premiers bits est aussi difficile que trouver d
- ⇒ Preuves d'équivalence nécessaires

Propriétés des « stream ciphers »

- Très proches de Vernam (pseudo-random)
 - Les plus rapides ($\times 10$ symétriques/blocs, $\times 100$ pub-key)
 - A5-GSM, RC4-WEP cassés ...
 - Pas de standard de remplacement
 - eStream 2008 → 7 recommandations, dont Rabbit
 - Synchronisation ou pré-génération de clefs
 - + État
- ⇒ Besoin de parallélisation → chiffrement par blocs
- Tout chiffrement par bloc + mode type CTR peut-être utilisé comme chiffrement par flot ...

Chiffrement par bloc

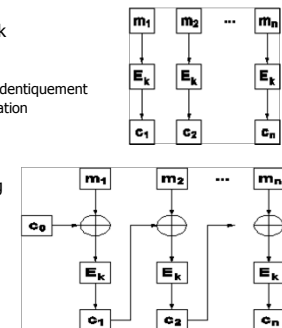
- $M = (M_1 | M_2 | \dots | M_s)$: s blocs de $r=n/s$ bits chacun



- Sécurité
 - Sur chaque bloc : $C_i = E_k(M_i)$, dépend de la fonction E
 - Pour le message : dépend également du mode de chiffrement

Cinq Modes de chiffrement (1/3)

- ECB : Electronic Code Book
 - ↳ $C_i = E_k(M_i)$
 - ↳ $M_i = D_k(C_i)$
 - Un bloc est toujours crypté identiquement
 - Aucune sécurité, pas d'utilisation
- CBC : Cipher Bloc Chaining
 - ↳ $C_i = E_k(M_i \oplus C_{i-1})$
 - ↳ $M_i = C_{i-1} \oplus D_k(C_i)$
 - Le plus utilisé



Cinq Modes de chiffrement (2/3)

3. CTR : Counter Encryption Mode
 ⇒ $C_i = M_i \oplus E_k(T+i)$
 ⇒ $M_i = C_i \oplus E_k(T+i)$

- Fait intervenir intervenir le chiffrement d'un compteur de valeur initiale T
- Totalement symétrique, facilement parallélisable
- Un même bloc n'est a priori jamais codé de la même façon

Cinq Modes de chiffrement (3/3)

4. CFB : Cipher Feedback
 ⇒ $C_i = M_i \oplus E_k(C_{i-1})$
 ⇒ $M_i = C_i \oplus E_k(C_{i-1})$

- Pas besoin de D_k
- Moins sûr, parfois plus rapide
- Réseaux

5. OFB : Output Feedback
 ⇒ $Z_i = E_k(Z_{i-1}) ; C_i = M_i \oplus Z_i$
 ⇒ $Z_i = E_k(Z_{i-1}) ; M_i = C_i \oplus Z_i$

- Totalement symétrique
- Moins de câblage
- Satellites

Data Encryption Standard

- Années 1970 Lucifer d'IBM, standard américain FIPS 46-2
 - Substitutions, Transpositions, Fonction de Feistel
 - Chiffrement par blocs de 64 bits
 - Clefs de 64 bits, 1 bit sur 8 de parité : 56 bits effectifs (plaidé par la NSA)
 - ⇒ Génération de 16 sous-clefs de 48 bits
- Structure
 - Permutation initiale
 - 16 itérations de Feistel : expansion, substitution, permutation
 - 16 sous-clefs de 48 bits dérivées de la clef initiale
 - Permutation inverse
- Après 5 tours, chaque bit du chiffré dépend de chaque bit du message en clair et de chaque bit de la clef
- Dès 1995, une puce dédiée chiffre 64 Mo/s

Détail du DES

- $L_0|R_0 = IP(M)$
- Faire 16 fois
 - $L_i = R_{i-1}$
 - $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
- $C = IP^{-1}(R_{16}|L_{16})$

- Déchiffrer : $C = DES_k(M) \Rightarrow M = DES_k^{-1}(C)$

La fonction de Feistel

- Expansion
 Les 32 bits du bloc sont étendus en 48 (duplication de certains bits)
- Mixage
 XOR avec les 48 bits d'une sous clef
- Substitution
 8 blocs de 6 bits transformés en 8 blocs de 4 bits
 8 S-boxes non linéaires, non inversibles
- Permutation
 32 bits sont permutés par une P-box

Sous clefs : itérer 16 fois
 2 blocs de 28 bits : chacun est décalé circulairement à gauche de 1 ou 2 bits
 Les 24 bits les plus à gauche et les 24 les plus à droite forment K_i

S-box

S ₅	Middle 4 bits of input																
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1100	1011	1100	1101	1110	1111	
Outer bits	00	0010	1100	0100	0001	0111	1100	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
bits	10	0100	0010	0001	1011	1100	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1100	0100	0101	0011

Propriétés cryptanalytiques

- Non linéaire (substitution différente de César ou Vernam) : pas d'attaque simple
- Spécialement conçues pour contrer la cryptanalyse différentielle [Coppersmith 94]
- Même de très petites modifications des S-box peuvent affaiblir considérablement le chiffre

Cryptanalyses du DES

- Préalcul exhaustif
 - Stocker l'application du DES sur un texte choisi avec toutes les clefs possibles
 - Lors d'une émission, retrouver le chiffre émis dans la table permet de connaître la clef
- Recherche exhaustive
 - Appliquer DES sur un texte connu jusqu'à retrouver le cryptage permet de connaître la clef
- Cryptanalyse différentielle *[Biham-Shamir 1990]*
 - Analyser les différences (B@B) de cryptage entre des textes similaires permet sélectionner des clefs probables
- Cryptanalyse linéaire *[Matsui 1993]*
 - Utiliser des approximations linéaires du chiffrement $(\Sigma M_i) \oplus (\Sigma C_i) = (\Sigma K_i)$ pour interpoler des bits de la clef

Complexité des attaques sur le DES

Méthode d'attaque	Texte connu	Texte choisi	Stockage	Calculs
Préalcul exhaustif		1	2^{56}	1 tableau
Recherche exhaustive	1			2^{55}
Cryptanalyse différentielle	2^{55}	2^{47}	Textes	2^{47} 2^{55}
Cryptanalyse linéaire	2^{43} 2^{38}		Textes	2^{43} 2^{50}

Sécurité du DES en 1996

Attaquant	Budget	Outil	Clef 40bits	Clef 56bits
Hacker	Négligeable 300€	Soft Circuit	1 semaine 5 heures	1260 ans 38 ans
PME	7500€	Circuit	12 min	18 mois
Grande entreprise	225 k€ 225 k€	Circuit ASIC	24 s 0,18 s	19 jours 3 heures
Multinationale	7,5 M€	ASIC	5 msec	6 min
État	225 M€	ASIC	0.2 msec	12 s

Double DES ?

- $C = E_2(E_1(M))$ et $M = D_1(D_2(C))$
- Clef effective de 112 bits ?
- Cassage du double DES pour un « texte clair connu » M, C par la méthode « rencontre au milieu »
 - Effectuer les 2^{56} cryptages possibles X_i de M
 - Trier ces X_i en 56×2^{56} étapes
 - Effectuer les 2^{56} décryptages possibles de C et comparer au fur et à mesure avec les X_i
- Donc en fait, seulement 112 fois plus difficile que DES et non 2^{56} fois !

Rencontre au milieu

- Pour un couple (M_1, C_1) donné du double DES :
 - Probabilité qu'un couple de clefs soit un faux positif $\leq 1/2^{64}$
 - Probabilité que la clef (K, K') soit la seule
 - $\geq (1-1/2^{64}) \wedge (2^{56} \cdot 1) \approx 1-1/2^9 = 99.61\%$
 - Avec un deuxième couple (M_2, C_2) :
 - $\geq (1-1/2^{128}) \wedge (2^{56} \cdot 1) \approx 1-3 \cdot 10^{-22} \approx 100\%$
- Pour un algorithme de chiffrement à 2^{64} clefs :
 - Probabilité que la clef soit la seule
 - $\geq (1-1/2^{64}) \wedge (2^{64} \cdot 1) \approx e^{-1} = 36.79\%$
 - Avec un deuxième couple (M_2, C_2) :
 - $\geq (1-1/2^{128}) \wedge (2^{64} \cdot 1) \approx 1-6 \cdot 10^{-20} \approx 100\%$

Triple DES ?

- Trois clefs : $C = E_1(E_2(E_3(M)))$ et $M = D_3(D_2(D_1(C)))$
- Deux clefs : $C = E_1(D_2(E_1(M)))$ et $M = D_1(E_2(D_1(C)))$
 - Compatibilité : retombe sur DES si clef 2 == clef 1
 - Clef moins longue et sécurité effective identique
- Méthode « rencontre au milieu » donne une clef effective de 112 bits, cette fois-ci.
- Triple DES seulement double la sécurité
 - Uniquement au niveau de la clef
 - Taille de blocs toujours 64 bits

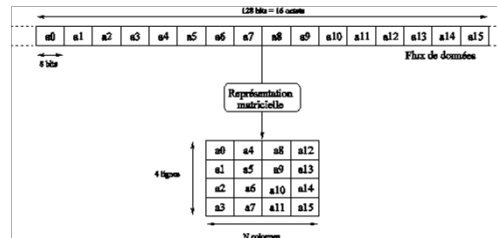
Advanced Encryption Standard

- Processus de sélection
 - Janvier 1997 : appel à candidatures
 - 5 finalistes : Rijndael, Serpent, Twofish, RC6, Mars
 - Août 2000 : sélection de Rijndael

Algorithme	Taille de clef	Cycle/octet	
		Cryptage	Décryptage
DES	56	59	59
3-DES	112	154	155
IDEA	128	56	56
Grand cru	128	1250	1518
RC6	256	18	17
Rijndael	256	34	35
Serpent	256	68	80
Mars	256	31	30
Twofish	256	29	25

AES

- Représentation Matricielle d'un flux de 128 bits = 16 octets = 4 blocs



- Configurations ($N_k = |clef|$, $N_b = \#blocs$, $N_t = \#tours$) \times 32 bits :
- AES-128 (4,4,10); AES-192 (6,4,12); AES-256 (8,4,14)

Tour de Rijndael-128

Faire 10 fois

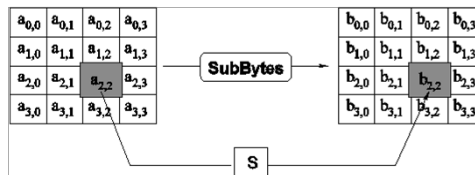
- AddRoundKey : \oplus bit à bit entre bloc et partie de la clef

Blocs sous forme de matrice (ex: 128 bits => 4x4 octets)
 octets considérés comme des éléments du corps fini GF(256)
 avec $GF(256) \cong Z/2Z[X]$ modulo $P_8 = X^8 + X^4 + X^3 + X + 1$

- SubBytes : pour chaque octet $S(a) = A(a^{-1} GF(256)) + C \text{ mod } 2$
- ShiftRow : chaque ligne est décalée de i
- MixColumn : brouillage des colonnes (code correcteur linéaire)
 Chaque colonne, polynôme de $GF(256)[Y]$, est multipliée par $([03]Y^3 + Y^2 + Y + [02]) \text{ modulo } (Y^4 + [01])$

SubBytes

- SubBytes : pour chaque octet
 $S(a) = A(a^{-1} GF(256)) + C \text{ mod } 2$
 $S(0) = C$



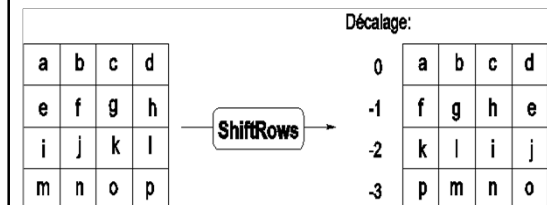
- $b_{i,j} = S(a_{i,j})$

SubBytes

- Effectuée en pratique par lecture de la table S de correspondance :

63	7c	77	7b	f2	6b	6f	c5	30	1	67	2b	fe	d7	ab	76
ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
d0	ef	aa	fb	43	4d	33	85	45	19	2	7f	50	3c	9f	a8
51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
cd	c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	b	db
e0	32	3a	a	49	6	24	5c	c2	d3	ac	62	91	95	e4	79
e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	8
ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
70	3e	b5	66	48	3	f6	e	61	35	57	b9	86	c1	1d	9e
e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
8c	a1	89	d	bf	e6	42	68	41	99	2d	f	b0	54	bb	16

ShiftRows



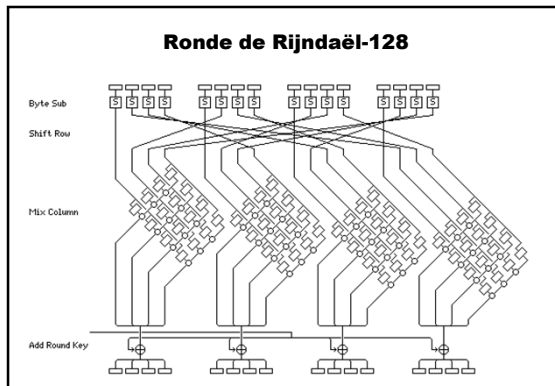
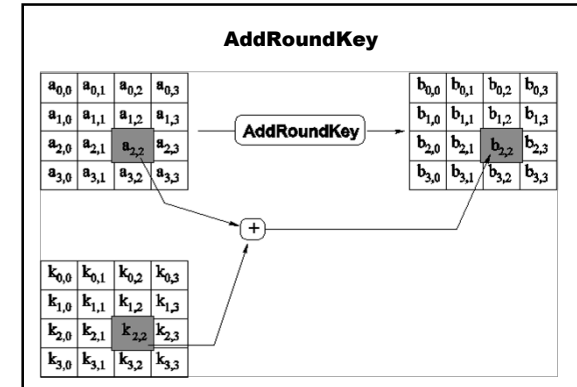
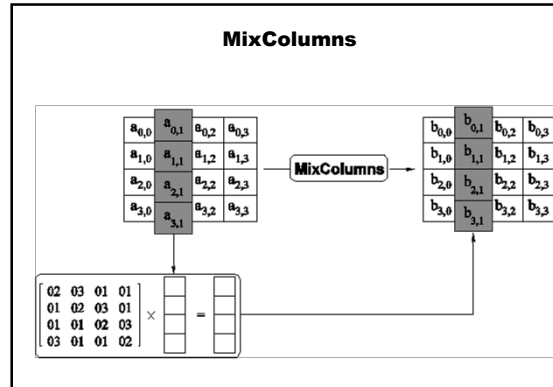
MixColumn

- Chaque colonne C , polynôme de $GF(256)[Y]$, est multipliée par $G = ([03]Y^3 + Y^2 + Y + [02])$ modulo $(Y^4 - [01])$
- Multiplication de Matrices : $A_G C$

$$A_G = \begin{bmatrix} [2] & [3] & [1] & [1] \\ [1] & [2] & [3] & [1] \\ [1] & [1] & [2] & [3] \\ [3] & [1] & [1] & [2] \end{bmatrix}$$

avec les monômes rangés de haut en bas par degré croissant

- $M = C \times G \text{ mod } Y^4 - [1]$: code 2-correcteur d'erreur cyclique sur les colonnes :
 - Détection jusqu'à 4 octets erronés dans M
 - Correction automatique si 2 octets de M sont modifiés
 - Bonne propriétés de diffusion cryptographique
 - $M \text{ mod } Y^4 - [1]$: addition des 8 octets deux à deux : conservation de la diffusion



Pseudo code AES

Cryptage A.E.S. $E_K(M)$

```

A:=M;
KeySchedule(K);
A:=AddRoundKey(A,K0);
Pour i = 1 à 9 Faire {
  A:=SubBytes(A);
  A:=ShiftRows(A);
  A:=MixColumns(A);
  A:=AddRoundKey(A,Ki);
}
A:=SubBytes(A);
A:=ShiftRows(A);
A:=AddRoundKey(A,K10);
Return C:=A;
                    
```

Décryptage A.E.S. $D_K(C)$

```

A:=C;
KeySchedule(K);
A:=AddRoundKey(A,K10);
A:=InverseShiftRows(A);
A:=InverseSubBytes(A);
Pour i = 1 à 9 Faire {
  A:=AddRoundKey(A,K10-i);
  A:=InverseMixColumns(A);
  A:=InverseShiftRows(A);
  A:=InverseSubBytes(A);
}
A:=AddRoundKey(A,K0);
Return M:=A;
                    
```

Extension de Clef

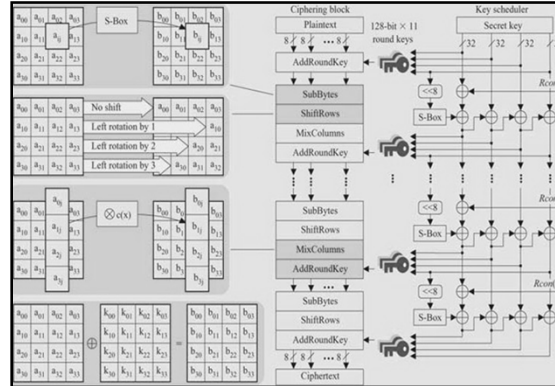
- N_k premières colonnes
- Colonnes suivantes obtenues par rondes d'après les colonnes précédentes :
 - SubWord : Sbox(octets de colonne)
 - RotWord : permutation circulaire d'une colonne
 - Rcon[i] = $[X^{i-1}, 00, 00, 00]$: constantes de tour
- Exemple : $c4 = c0 \oplus \text{SubWord}(\text{RotWord}(c3)) \oplus \text{Rcon}[1]$

Pseudo code KeySchedule

KeySchedule(*K*)

```

W0||W1||W2||W3 := K0
Pour i = 1 à 10 Faire {
    T := W4i-1 <<< 8; // décalage circulaire de 8 bits vers la gauche
    T := SubBytes(T); // 4 appels SubBytes
    T := T ⊕ (Xi mod P8); // addition sur chaque octet
    W4i := W4i-4 ⊕ T;
    W4i+1 := W4i-3 ⊕ W4i;
    W4i+2 := W4i-2 ⊕ W4i+1;
    W4i+3 := W4i-1 ⊕ W4i+2;
    Ki = W4i||W4i+1||W4i+2||W4i+3;
}
    
```



Implémentation

```

UINT32 m_uRounds;
UINT8 m_expandedKey[ MAX_ROUNDS+1][4][4];
static UINT8 T1[256][4] = { 0xc5,0x63,0x63,0xa5, 0x68,0x7c,0x7c,0x84, ... };
static UINT8 T2[256][4] = { 0xa5,0xc5,0x63,0x63, 0x84,0x84,0x7c,0x7c, ... };
static UINT8 T3[256][4] = { 0x63,0xa5,0xc5,0x63, 0x7c,0x84,0x84,0x7c, ... };
static UINT8 T4[256][4] = { 0x63,0x63,0xa5,0xc5, 0x7c,0x7c,0x84,0x84, ... };

void Rijndael::encrypt(const UINT8 a[16], UINT8 b[16]) {
    int r;
    UINT8 temp[4][4];

    // XOR with key
    *((UINT32*)temp[0]) = *((UINT32*)a) ^ *((UINT32*)m_expandedKey[0][0]);
    *((UINT32*)temp[1]) = *((UINT32*)a+4) ^ *((UINT32*)m_expandedKey[0][1]);
    *((UINT32*)temp[2]) = *((UINT32*)a+8) ^ *((UINT32*)m_expandedKey[0][2]);
    *((UINT32*)temp[3]) = *((UINT32*)a+12) ^ *((UINT32*)m_expandedKey[0][3]);

    // First round
    *((UINT32*)b) = *((UINT32*)temp[0][0])
        ^ *((UINT32*)T1[temp[1][1]])
        ^ *((UINT32*)T2[temp[2][2]])
        ^ *((UINT32*)T4[temp[3][3]]);

    *((UINT32*)b+4) = *((UINT32*)temp[2][1])
        ^ *((UINT32*)T2[temp[2][1]])
        ^ *((UINT32*)T3[temp[3][2]]);
}
    
```

Sécurité de l'AES

- Propriétés cryptanalytiques
 - S : sans point fixe ni opposé, ni inverse
 - ShiftRow diffuse les données en séparant les consécutifs
 - MixColumn : chaque bit de sortie dépend de tous les bits en entrée (code correcteur linéaire sur chaque colonne)
- Implémentations FPGA
 - Jusqu'à 21.54 Go/s pour le cryptage [Hodjat-Verbauwhede 2004]
- Sécurité
 - Aucune attaque significative révélée
 - [Courtois, Pieprzyk, 02, 05] attaques en $\approx O(2^{100})$ sur AES-128
 - ...

Applications utilisant Rijndael

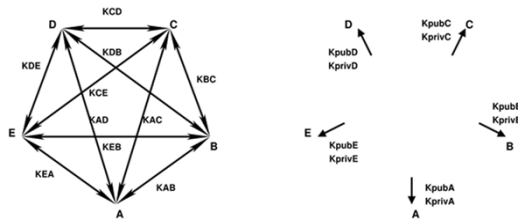
- SONET (Synchronous Optical NETWORK)
- Routeurs Internet
- Switch Ethernet ATM (Asynchronous Transfert Mode)
- Communications Sattelites
- VPN (Réseaux privés virtuels)
- Téléphonie mobile
- Transactions électroniques

Plan

- Introduction
- Histoire des codes secrets
- Théorie de l'Information
- Confidentialité
 - Clefs symétriques
 - Clefs asymétriques
 - Arithmétique et complexité
 - Théorèmes de Fermat, Chinois, RSA
 - Logarithme discret (DLP)
 - Clefs publiques/privées
- Intégrité, Authentification, Non-Répudiation
- Législation

Clef publique / clef privée

- Tout le monde peut envoyer (avec la clef publique)
- Seul le destinataire peut déchiffrer (avec la clef privée)



Entropie et clefs publiques ?

- K_d est secrète, K_e est publique
- Un cryptosystème à clef publique parfait devrait vérifier $H(M | C = EK_e(M)) = H(M)$
- Dans un cryptosystème à clef secrète les fonctions de chiffrement et déchiffrement sont inconnues car la clef est secrète
- Or dans un cryptosystème à clef publique le déchiffrement est l'inverse du chiffrement, et le chiffrement est explicite E_{K_e} est explicite car K_e publique et $D_{K_d} = E_{K_e}^{-1}$
- ⊗ Du point de vue de la théorie de l'information, on a donc
 - ⊗ $H(M|C) = 0$ et même $H(M) = 0$ si le système pouvait être parfait
 - ⊗ $H(K_d | K_e) = 0$ dans tous les cas

Il n'y a aucun secret sur la clef privée ni sur le message !!!
- ⇒ La théorie de l'information est insuffisante, l'incertitude sur la clef privée n'est pas pertinente pour mesurer la fiabilité d'un cryptosystème à clef publique, la caractérisation doit plutôt se faire par la *théorie de la complexité*

Théorie de la complexité

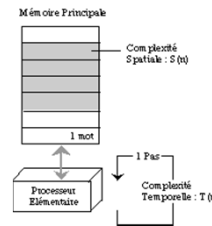
- Méthodologie pour analyser la complexité de calcul des algorithmes
 - Complexité en temps (ou en nombre d'opérations)
 - Complexité en mémoire (espace de stockage nécessaire)
- Complexité exprimée comme fonction de la taille du paramètre d'entrée
- Complexité d'un problème : complexité de l'algorithme permettant de résoudre l'instance la plus difficile
- Classification :
 - Problèmes solubles (en temps polynomial) : classe P
 - Problèmes difficiles (solubles en temps exponentiel) : classe NP
 - Problèmes indécidables

Ordres de grandeur

- Taille d'un entier n : $t_n = \lceil \log_2(n) \rceil$ bits $\rightarrow n = O(\exp(t_n))$
 - 128 bits = 39 chiffres, 512 bits = 155 chiffres, 1024 bits = 309 chiffres
- Vitesse des ordinateurs
 - 1 GHz = 10^9 secondes
 - Lumière du projecteur à la salle : 3m à 300000 km/s = 10^{-8} s
 - Ce portable fait donc 30 additions le temps que la lumière arrive du projecteur !
- Age de l'univers : 15 milliards * 365 * 24 * 3600 $\approx 5.10^{17}$ s
- Nombre d'électrons dans l'univers $\approx 10^{28}$ e/g * 10^{33} g * 10^{11} * 10^{12} : 10^{84}
- Complexité d'un algorithme pour des nombres de 128 bits
 - $t_n = 128$ opérations, $t_n^2 = 16384$ ops, $t_n^3 = 2.10^6$ ops, $t_n^4 = 3.10^8$ ops
 - $n = 10^{39}$ opérations \rightarrow 2 millions de fois l'âge de l'univers sur un million de PC à 1GHz

Complexité asymptotique

- $S(n)$ = nombre de mots mémoire nécessaires à l'implantation de l'algorithme
 - Trier n "int", $S(n)=n$
 - Trier des noms de 20 caractères ASCII sur machine 32bits, $S(n)=5n$
- $T(n)$ = nombre d'opérations nécessaires à l'exécution de l'algorithme
- Équivalence asymptotique
 - $f(n) = O(g(n))$ ssi $\exists N, k$ tels que
 - $\forall n > N, f(n) \leq k g(n)$ et $g(n) \leq k f(n)$
 - $5x^2 + 3x + 1 = O(x^2 + 700) = O(x^2)$



Calcul sur les entiers

- 32 (ou 64) bits : codent entiers de 0 à $2^{32}-1$
 - addition (≈ 1 cycle)
 - soustraction (≈ 1 cycle)
 - multiplication (≈ 1 cycle)
 - division (≈ 10 cycles)
- Pour 1 GHz ≈ 1 milliard d'additions/multiplications à la seconde

Entiers en précision arbitraire

- Ex: C/C++ Gnu Multiprecision Package (GMP)
 - Liste de mots non signés de 32 bits, plus le signe.
- Addition/soustraction d'entiers bornés par n : $O(t_n = \log_2(n))$ opérations
- Multiplication :
 - $O(t_n^2)$ opérations (méthode classique)
 - Karatsuba $O(t_n^{1.585})$
 - Toom-Cook $O(t_n^{1.465})$
 - Schönhage & Strassen $O(t_n \log(t_n) \log \log(t_n))$.
- En pratique : classique jusqu'à 32 mots (=300 chiffres), Karatsuba jusqu'à 256 mots, première FFT à partir de 10000 mots.
- Division : cf. multiplication, + divisions sur 32 bits

Arithmétique classique

- MAops = million d'opérations arithmétiques / seconde
 - Sur un Pentium IV, 2.4GHz, max \approx 2400 MAops
- Mesures : comptent accès mémoire, copie, affectation, opération, ...
- Sur 32 bits machine (long int) :

- Addition, Soustraction, Multiplication :	365 MAops		
- Négation :	915 MAops		
- AXPYIN ($r += a \times b$) :	711 MAops		
- Division :	81 MAops		
- Avec GMP,

	32 bits	300 bits	3000 bits
- Addition, Soustraction :	2.2	1.7	0.64
- Multiplication :	2.2	0.6	0.017
- Négation :	4.4	2.6	0.85
- AXPYIN : ($r += a \times b$) :	4.5	1.1	0.035
- Division :	2.2	0.7	0.03

Entiers modulaires

- Notation : $a \equiv b [m]$, (a est congru à b modulo m) si (a-b) est divisible par m. ex: $11 \equiv 5 [6]$
- $Z/nZ = \{0, 1, \dots, n-1\}$
- Opérations modulaires classiques :
 - Addition/Soustraction : addition d'entiers + décalage de m
 - $r = a+b$
 - si ($r > m$) Alors $r = r - m$
 - Multiplication : multiplication d'entiers + division d'entiers
 - $r = a \times b$
 - $r = r \% m$ // Calcul du reste de la division de r par m
 - Division : pgcd + division d'entiers quand valide

Algorithme d'Euclide (-325/-265) étendu

- Bézout : a et b premiers, $\exists (u,v), a u + b v = 1 = \text{pgcd}(a,b)$
- Algorithme d'Euclide pour le pgcd :
- $a = b q_1 + r_1; b = r_1 q_2 + r_2; r_1 = r_2 q_3 + r_3 \dots$
- Algorithme d'Euclide étendu, calcule aussi u et v
 - $1 a + 0 b = a$
 - $0 a + 1 b = b$
 → Appliquer le pgcd sur 1. et 2.
- Complexité classique : $O(t_n^2)$, ... FFT en $O(t_n \log(t_n)^2)$
- Application : calcul de la division dans un corps premier :
 - $a u \equiv 1 [b]$, i.e. u est l'inverse de a modulo b !
 - Corollaire : Z/pZ est un corps pour p premier (tout non nul plus petit que p est inversible, puisque premier avec p)

Arithmétique MODULAIRE classique

- MAops = million d'opérations arithmétiques / seconde
 - Sur un Pentium IV, 2.4GHz, max \approx 2400 MAops
- Mesures : comptent accès mémoire, copie, affectation, opération, ...
- Sur 16 bits machine (long int) :

- Addition, Soustraction :	300 MAops		
- Multiplication :	68 MAops		
- Négation :	475 MAops		
- AXPYIN ($r += a \times b$) :	135 MAops		
- Division (Bézout) :	1.6 MAops		
- Avec GMP,

	32 bits	300 bits	3000 bits
- Addition, Soustraction :	1.8	1.5	0.64
- Multiplication :	1.7	0.3	0.007
- Négation :	1.7	1.4	0.64
- AXPYIN :	3	0.55	0.013
- Division :	0.46	0.03	0.001

Petit théorème de Fermat

- Indicateur d'Euler : $\phi(n)$ nombre de premiers avec n
 - c'est le cardinal de Z/nZ^*
- Algorithme :
 - p premier, $\phi(p) = p - 1, \phi(p^k) = (p-1)p^{k-1}$
 - m, n premiers entre eux $\phi(mn) = \phi(m)\phi(n)$
- Théorème d'Euler (1707-1783) : $a^{\phi(n)} \equiv 1 [n]$, pour a inversible modulo n
 - $G_n = \{y, y=ax, x \in Z/nZ^*\}$ Preuve de Lagrange (1736-1813)
 - $G_n = Z/nZ^*$, car a, x et y inversibles
 - $\prod_{x \in Z/nZ^*} x = \prod_{y \in G_n} y = \prod_{x \in Z/nZ^*} ax$
 - $a^{|Z/nZ^*|} = 1$
- Théorème de Fermat (1601-1665) : $a^p \equiv a [p]$ pour p premier

Théorème Chinois

(Qin Jiu-Shao XIII^e siècle)

- Soient m_1, \dots, m_n positifs et premiers entre eux
- M leur produit
- Alors, pour tous résidus a_1, \dots, a_n il existe un unique $x < M$ tel que les restes de la division de x par m_i coïncident avec les a_i :

$$\forall a_1, \dots, a_k, \exists ! x < M, x \equiv a_i [m_i]$$

- Construction (par Lagrange (1736-1813)) :
 - Posons $M_i = M/m_i$,
 - D'après l'AEE, $\exists y_i$ tel que $y_i M_i \equiv 1 [m_i]$
 - Alors $x \equiv \sum a_i y_i M_i [M]$ convient !
- Construction itérée par Newton (1643-1727) (différences multipliées)

Le théorème RSA

- Théorème [Rivest-Shamir-Adleman 1978] : pour p et q premiers, on pose $n=pq$ et on a

$$\forall a \in \mathbb{Z}/n\mathbb{Z}, a^{1+k(p-1)(q-1)} \equiv a \pmod n$$

- Preuve : petit théorème de Fermat + théorème chinois
- Choix d'un entier e premier avec $\phi(pq) = (p-1)(q-1)$
- Algorithme d'Euclide étendu calcule les coefficients de Bézout d et k tels que $ed - (p-1)(q-1)k = 1$

Preuve Rivest-Shamir-Adleman

- Conséquence des Théorèmes Chinois et d'Euler :
 - p et q premiers
 - $n = pq$, on a : $a^{k\phi(n)+1} \equiv a [n]$

- Dans le cas a inversible : Euler donne $a^{\phi(n)} \equiv 1 [n]$
- a non-inversible modulo n :
 - Si $a \equiv 0 [p]$, alors $a^{\phi(n)} \equiv 0 \equiv a [p]$
 - Sinon a inversible mod p et $a^{p-1} \equiv 1 [p]$ donc $a^{(p-1)k(q-1)+1} \equiv a [p]$
 - De même $a^{(q-1)k(p-1)+1} \equiv a [q]$
 - Par théorème chinois, on alors $a^{k\phi(n)+1} \equiv a [pq]$

- Intérêt : avec e et $d = k\phi(n) + 1$, e , n sont *publics*, d est *secret*
 - Codage de $a \rightarrow x \equiv a^e [n]$
 - Décodage de $x \rightarrow x^d \equiv a^{ed} \equiv a [n]$

Le code RSA

- Les entiers n et e choisis sont la clef PUBLIQUE
- L'entier d associé est la clef PRIVÉE secrète
 p et q sont également secrets
- Le chiffrement d'un message M s'effectue par découpage en blocs $m_i < pq$, puis exponentiation rapide avec la clef publique : $c_i = m_i^e \pmod n$
- Le déchiffrement d'un chiffré C s'effectue également par exponentiation rapide avec la clef privée :
 $m_i = c_i^d \pmod n$

Implémentation de RSA :

- Construire p et $q \rightarrow 1^\circ$ Test de primalité
- $(p-1)(q-1) = \phi(n)$
- Construire e et d :
 - On choisit e (pas trop petit) premier avec $\phi(n)$
 - $\rightarrow 2^\circ$ Algorithme d'Euclide étendu donne d et k tq $ed + k\phi(n) = 1$
- Coder et décoder $\rightarrow 3^\circ$ élévation à la puissance
- Casser le code
i.e. : Trouver d , ne connaissant que e et $n \rightarrow$ calculer $\phi(n) = (p-1)(q-1)$
Donc *factoriser* $n \rightarrow 4^\circ$ Factorisation d'entiers ?

Élévation à la puissance

- Ex: a^{11} , 10 multiplications ?
- $a^{11} = a^{10} a = (a^5)^2 a = ((a^2)^2 a)^2 a$: 5 multiplications !

```
Integer PuissanceModuloRecursive( Integer c, Integer d, Integer n ) {
    if ( d == 0 ) return Integer(1);

    // d >> 1 : décalage == partie entière de d/2
    Integer s = PuissanceModuloRecursive( c, d >> 1, n );
    s = ( s * s ) % n ; // carré

    if ( d & 1UL ) // d est impair
        s = ( s * c ) % n ; // multiplication

    return s;
}
```

© Complexité : **O(log₂(d))** multiplications

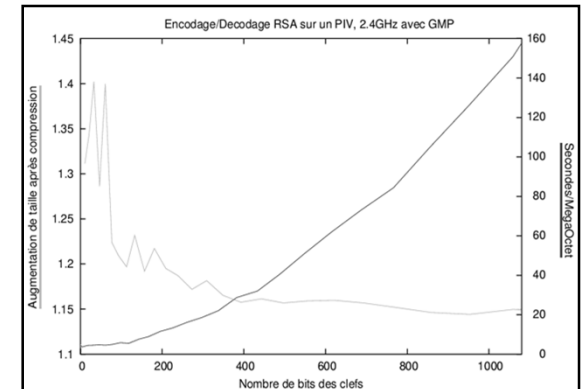
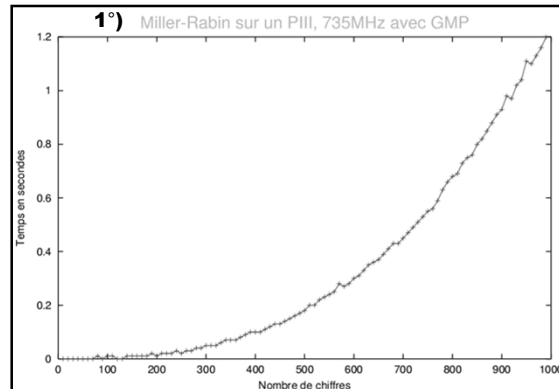
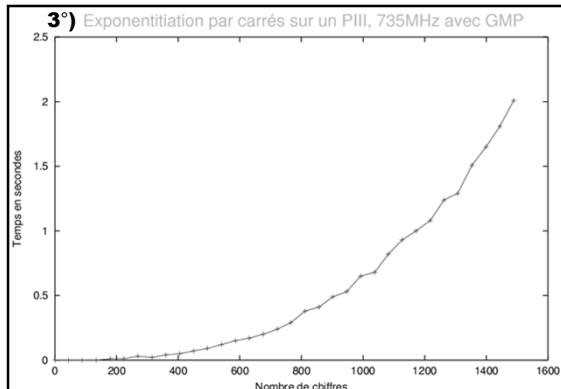
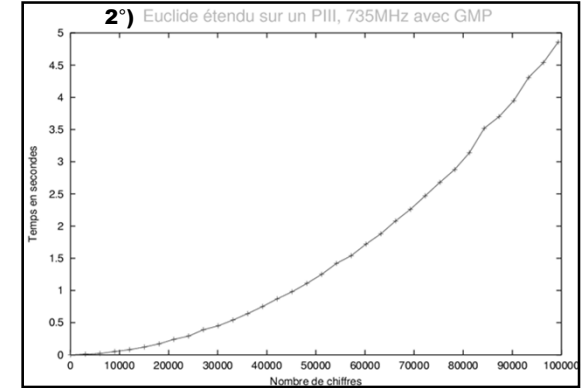
Tests de primalité

- Crible d'Eratosthène déterministe
 - Marquer tous les multiples jusqu'à \sqrt{n}
 - $O(n)$ additions, exponentiel, inutilisable au delà de 10 chiffres
- Miller/Rabin probabiliste polynomial
 - Tirage aléatoire d'un témoin
 - Si n est premier, le témoin indiquera *premier*
 - Si n est non premier, 3/4 des témoins indiqueront *nonpremier*
 - MAIS, 1/4 des témoins donneront une fausse indication
 - Répété k fois, probabilité de réponse *premier* incorrecte = 2^{-2k}
- Déterministe polynomial
 - Miller/Rabin avec suffisamment de témoins, sous ERH
 - Agrawal, Kayal et Saxena, sans ERH

Miller-Rabin

- Si n est impair, $n - 1 = t \cdot 2^s$, avec t impair
 - Pour a quelconque on a alors :

$$a^{(n-1)} - 1 = (a^t)^{2^s} - 1 = (a^t - 1)(a^t + 1)(a^{2t} + 1) \dots (a^{(2^{s-1})t} + 1)$$
 - Or, si n est premier alors Euler donne $a^{n-1} \equiv 1[n]$, donc
 - Soit $a^t \equiv 1[n]$.
 - Soit $a^{2^i t} \equiv -1[n]$ avec $0 \leq i < s$.
- ⇒ Test de Miller-Rabin : tirer a au hasard et vérifier Fermat



Casser RSA

- Déchiffrer un message secret sans la clef privée
 - Connaissant c , calculer m_i directement :
 - $c_i \equiv m_i^e \pmod n$ se transforme en passant au logarithme en :
 - $\log_p(c_i) \equiv e \log_p(m_i) \pmod{\phi(n)}$
 - ⊗ Il est déjà difficile de calculer $\log_p(c_i)$
 - C'est plus simple de trouver la clef privée
 - ⇒ Il faut calculer les coefficients de Bézout de e et $\phi(n)$
- ⇒ Il faut calculer $\phi(n)=(p-1)(q-1)$ sans connaître ni p ni q
 ⇒ Factoriser n

Paramètres intéressants

- Chiffrement et déchiffrement faciles
 - Exponentiation rapide récursive par carrés :
 - $5^{11} [7] \equiv ((5^2)^2 5)^2 5 \equiv ((4)^2 5)^2 5 \equiv (2 \times 5)^2 5 \equiv 2 \times 5 \equiv 3$
 - Complexité $O(\log(n))$ multiplications de grands entiers soit $O(\log^3(n))$
 - En pratique une exponentiation à 60 000 chiffres (200 000 bits) en 1 seconde
- Cassage trop difficile
 - Meilleure factorisation connue en complexité $\approx n^{1/6}$
 - En pratique plusieurs mois pour casser du RSA à 200 chiffres

Factorisation d'entiers : les records

- Pollard rho : quelques secondes jusqu'à 30 chiffres
- Variantes si $n-1$, $n+1$ sont faciles à factoriser
- Courbes Elliptiques :
 - quelques secondes jusqu'à 50 chiffres
 - record facteur premier de 67 chiffres au 08/2006
- Number Field Sieve (crible de corps de nombres) :
 - 313 chiffres (1039 bits), par SNFS, au 05/2007
 - RSA-768 bits 12/2009

Un « bon » algorithme de factorisation à la main $O(n^{1/2})$

1. Prendre un entier au hasard parmi $2 \dots n-1$
 - ⊗ Une chance sur p de tomber sur un multiple d'un facteur premier p de n
 2. Quand on a un multiple m de p : $\text{pgcd}(m,n) = p$ factorise n
 - ⊗ Espérance du nombre de tirages aléatoires et de calculs de pgcd : $p/2$
- ⊗ Complexité identique à celle du crible d'Ératosthène !

Paradoxe des anniversaires

- Population de k personnes, n jours dans l'année
 - Nombres de combinaisons d'anniversaires différents : A_n^k
 - Probabilité pour deux personnes au moins aient leurs anniversaires le même jour : $1 - \frac{A_n^k}{n^k}$
 - $k=9 \Rightarrow P = 0.1$; $k=23 \Rightarrow P > 0.5$; $k = 60 \Rightarrow P > 0.994$
- Plus généralement, si $k > 1.18\sqrt{n}$, alors plus d'une chance sur deux de collision !

Algorithme rho de Pollard $O(n^{1/2})$

- Factorisation par pgcd
 - On considère une suite $u_{i+1} \equiv f(u_i) \pmod n$ (souvent $f(x) = x^2+c$)
 - Soit p le plus petit facteur de n . Il existe i et j tels que $u_i \equiv u_j \pmod p$.
 - Donc $(u_i - u_j)$ est un multiple de p . En général non multiple de n .
 - Il suffit donc de faire $\text{pgcd}(u_i - u_j, n)$ pour trouver un facteur
 - Essayer tous les cas i et j : $O(\sqrt{n})$ opérations, $O(\sqrt{n} \log(n))$ espaces mémoire
- Algorithme Rho : recherche de cycle [Floyd]
 - ⊗ L'idée c'est de ne pas garder tous les u_i précédents mais seulement $u_0, u_1, u_2, u_4, u_8, u_{16}$, etc.
 - Car si $u_i \equiv u_j$, alors $u_{i+k} \equiv u_{j+k} \forall k$.
 - Garder successivement seulement les indices 2^i
 - ⊗ Au pire le nombre d'appels est doublé
 - ⊗ Complexité exponentielle conservée $O(\sqrt{p}) = O(n^{1/4})$ avec seulement $O(\log(n))$ espaces mémoire



Crible quadratique $\approx O(n^{1/6})$

- Idée : Si $x^2 \equiv y^2 \pmod n$, avec $|x| \neq |y|$
alors $(x-y)(x+y) \equiv 0 \pmod n$
donc $\gcd(x-y, n)$ est un facteur non-trivial de n .
- $87^2 = 7429 + 140$ et $140 = 2^2 \cdot 5 \cdot 7$ par l'algorithme rho
- $88^2 = 7429 + 315$ et $315 = 3^2 \cdot 5 \cdot 7$ par l'algorithme rho
- $(87 \cdot 88)^2 \equiv (2^2 \cdot 3^2 \cdot 5^2 \cdot 7^2)^2 \pmod{7429}$
 - $x = 87 \cdot 88 \equiv 227$
 - $y = 2^2 \cdot 3^2 \cdot 5 \cdot 7 = 210$
 - $7429 = 17 \cdot 347$

Phase de Crible

	Exposant de 2	Exposant de 3	Exposant de 5	Exposant de 7	
83 ² mod 7429	2	3	1	0	...
87 ² mod 7429	2	0	1	1	...
88 ² mod 7429	0	2	1	1	...
...

Phase d'algèbre linéaire

- $(87 \cdot 88)^2$ est un carré ssi ligne 87² + ligne 88² est paire
- Trouver un vecteur binaire x tq $x \cdot M$ est pair
- Trouver un vecteur x tq $x \cdot M \equiv 0$ modulo 2.

Logarithme discret

- Connaissant n , a et b , calculer x tel que $b \equiv a^x \pmod n$?
 - Exemple : $6^x \equiv 8 \pmod{11}$?
 - Comment faire ?
 - Énumération de tous les exposants : $O(n)$ multiplications
 - Baby step / Giant step [Shanks], [Pollard] : $O(\sqrt{n})$
 - ...
- x est le logarithme (discret) de b
- $x = \log_a(b) \pmod{\varphi(n)}$

$6^7 \equiv 6^2 6^2 6^2 \equiv 3^3 6 \equiv 5 \cdot 6 \equiv 8 \pmod{11}$

Théorème du logarithme discret

- Si g est une racine primitive de Z/mZ
- $\forall x, y \in \mathbb{N}$
 $g^x \equiv g^y \pmod m$ ssi $x \equiv y \pmod{\varphi(m)}$
- \Rightarrow si $g^x \equiv g^y$ alors $g^{x-y} \equiv 1$, et donc $x-y$ doit être un multiple de $\varphi(m)$, car si $g^r \equiv 1$, alors $t = q\varphi + r$ et donc $g^t \equiv 1$ avec $r < \varphi(m)$
- $\Rightarrow g^{\varphi(m)} \equiv 1$, donc si $x = y + k\varphi$, alors $g^x \equiv g^y$

Chiffrement ElGamal

- Clef privée d'Alice x
- Clef publique $(p, g, h = g^x \pmod p)$, avec g un générateur

Alice

- Reçoit (c_1, c_2)
- Calcule $m = (c_2 \cdot c_1^{-x}) \pmod p$

Bob

- Choisit aléatoirement k
- Calcule $c_1 = g^k \pmod p$
- Calcule $c_2 = (m \cdot h^k) \pmod p$
- Envoie (c_1, c_2)

Diffie-Hellman

- Calculer $\log_g(c) \pmod m$ est difficile
- Meilleure complexité connue $\approx m^{1/3}$
- \Rightarrow Échange de clefs secrètes sur un canal non sécurisé [Diffie-Hellman 1976], m et g sont publics

Alice

- Choisit aléatoirement a
- Calcule $u = g^a \pmod m$
- Reçoit v
- Calcule $K = v^a \pmod m$

Bob

- Choisit aléatoirement b
- Calcule $v = g^b \pmod m$
- Reçoit u
- Calcule $K = u^b \pmod m$

Pour échanger une clef secrète

- Envoi d'une partie de la clef
 - Alice choisit pseudo-aléatoirement x et envoie $a \equiv g^x [n]$
 - Bob choisit pseudo-aléatoirement y et envoie $b \equiv g^y [n]$
 - Construction de la clef finale
 - Alice calcule $K \equiv b^x [n]$
 - Bob calcule $K \equiv a^y [n]$
- ☉ À aucun moment la clef finale n'a transité
- Pour calculer K à partir de a et b , il faut résoudre un logarithme discret
- ☉ Inefficace contre un adversaire actif (man-in-the-middle)

Des corps finis plus grands ?

- Le calcul du logarithme discret dépend du nombre d'éléments dans le corps
- Travailler avec des nombres premiers plus grands
 - ☉ Même les calculs d'exponentiation deviennent lents
 - Travailler dans une extension
 - avec caractéristique accélérant les calculs $\rightarrow GF(2^k)$
 - ❖ Attention, $GF(2^k) \neq Z/2^kZ$

Quelques propriétés des corps finis

- Q et Z/pZ : corps premiers
- $GF(q)$: corps fini ou corps de Galois à q éléments
- $K = \{k \in N^*, k \cdot 1_K = 0\}$; Caractéristique du corps :
 - 0 si K est vide (ex: R, C, Q)
 - Plus petit élément de K (ex: p pour Z/pZ)
 - Caractéristique : 0 ou un nombre premier !
- Cardinal d'un corps fini = puissance de sa caractéristique
- L'ensemble des inversibles d'un corps fini $GF(p^k)^*$ est cyclique de cardinal $p^k - 1$
- L'ordre d'un élément est la pp puissance tq $x^e = 1$, il divise $p^k - 1$

Revenons à l'arithmétique modulaire sur des mots machine

implémentation classique

- Utilise division système pour le calcul de la multiplication
- AXPY:** $r = (a \times x + y)$; $r = (r < p ? r : r \% p)$;
- Addition, soustraction sont rapides
 - Multiplication est lente
 - Requis : $(p^2 + p) < \text{taille des mots}$

Implémentation avec inverse numérique

- Stocke une représentation numérique de l'inverse de p
- AXPY:** $r = (a \times x + y)$; $r -= \text{floor}(r \times 1/p) \times p$
- Multiplication parfois plus rapide (pas de division, mais floor est assez lent également)
 - Requiert aussi : $(p^2 + p) < \text{taille des mots}$

Implémentation avec inverse numérique

- Stocke une représentation numérique de l'inverse de p
- AXPY:** $r = (a \times x + y)$; $r -= \text{floor}(r \times 1/p) \times p$
- Multiplication parfois plus rapide (pas de division, mais floor est assez lent également)
 - Requiert aussi : $(p^2 - p) < \text{taille des mots}$
 - Variante existe avec fmod (reste flottant), attention aux arrondis.

Petits corps premiers : générateurs 1/2

- Pré calcul de 3 tables
⇒ Moins de 2s de calcul pour des corps avec moins de 2²⁴ éléments
- 1) Correspondance entre x et i : t₁[x] = i, tq x = gⁱ
- 2) Correspondance entre i et x : t₂[i] = x, tq x = gⁱ
- 3) Table des « Plus un » : t₃[i] = j, tq 1+gⁱ = g^j
- [Conway] : faire les opérations sur les indices !
 - a × x : (gⁱ × g^j) % m = g^{i+j ±(m-1)}
 - 0 et 1 ont des valeurs particulières, par exemple 0 et m-1
- [Imamura], [Hubert], [Douillet] → réduisent la taille des tables

Petits corps premiers : générateurs 2/2

a = gⁱ; x = g^j; y = g^k;
 a × x + y = gⁱ × g^j + g^k = g^{k(1+g^{i+j+k})}

AXPY: r = i+j-k;
 r = t₃[r] + k;
 et tests pour zéro, indices modulo m-1 ...

- Absolument aucune multiplication ni division système
- Chaque opération est une combinaison de tests, additions et accès aux tables.

Réduction de Montgomery

- La division système est remplacée par des décalages et des masques

```
#define MASK 65535UL
#define B 65536UL
#define HALF_BITS 16
/* nim is precomputed to -1/p mod B
with the extended gcd */
```

AXPY:

1. c = (a × x + y);
2. unsigned long c0 (c & MASK); /* c mod B */
3. c0 = (c0 * nim) & MASK; /* -c/p mod B */
4. c += c0 * p; /* c = 0 mod B */
5. c >>= HALF_BITS; /* high bits of c */
6. return (c > p ? c - p : c); /* 0 < c < 2p */

- Aucune division
- Premiers plus petits: Taille(p²+p*half_word_size) < taille des mots
 - (p<40499 pour 32 bits, p<2654435761 pour 64 bits)

Sun Ultra 250 MHz

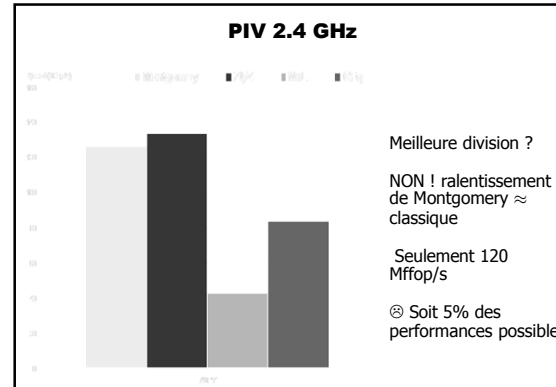
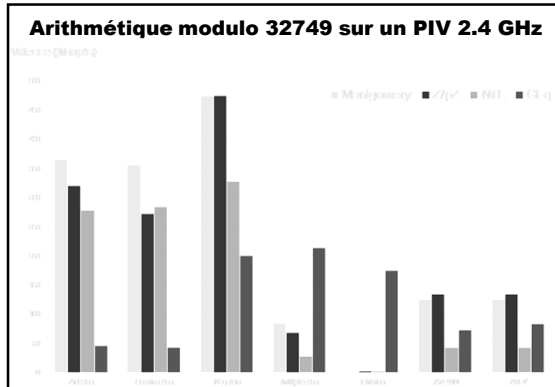
Mémoire rapide / arithmétique machine :
 ⇒ Faire des Tables
 ⇒ Tabuler p² valeurs est possible, au prix de :
 ⇒ Tables énormes / défauts de cache
 ⇒ additions supplémentaires
 [Kawame-Muraio, MBLAS]

Arithmétique modulo 32749 sur un PIII 993 MHz

Choix d'implémentation dépendant des machines/algothmes

PIII 993 MHz

Meilleure arithmétique, Division toujours lente:
 ⇒ Montgomery gagne



Anneau des polynômes sur un corps abélien

- G un corps, alors pour $a_i \in G$, on écrit $P = \sum a_i X^i \in G[X]$
- Dans cet anneau, il existe une division euclidienne :
 - $\forall A, B \in G[X] : \exists! (Q, R) \in G[X]$ avec $\deg(R) < \deg(B)$ tels que $A = B \cdot Q + R$
 - Il y a donc un pgcd, et l'Algorithme d'Euclide étendu est valide.
 - Deux polynômes sont premiers entre eux si leur pgcd $\in G$
 - Un polynôme est irréductible si il est premier avec tous ceux de degré inférieur
- Il y a une factorisation (unique à un facteur de G près) en polynômes irréductibles.
 - ⊗ Elle est polynomiale ! [Berlekamp], [Cantor-Zassenhaus]
 - Ex: Dans $Z/3Z$, on a $X^2+X+1=(X-1)(X^2+X-1)$

Coût des opérations polynômiales

- Addition : d additions du corps
- Multiplication
 - Classique : $O(d^2)$ additions/multiplications du corps
 - Karatsuba : $O(d^{1.585})$ additions/multiplications du corps
 - FFT : $O(d \log(d))$ additions/multiplications du corps
- Euclide : $O(\text{Mult}(d) \log(d))$ opérations du corps
- Irréductibilité [Ben-Or]
 - $O(d \text{ Mult}(d) \log(q)) = O(d^2 \log(q))$ opérations de $GF(q)$
- Factorisation
 - [Cantor-Zassenhaus] : $O(d^2 \log(q))$ opérations de $GF(q)$
 - [Berlekamp] : $O(d^3 + \text{Mult}(d) \log(q))$ opérations de $GF(q)$

Produit rapide de polynômes par la DFT

- $DFT(P) = [P(w^0); P(w); \dots; P(w^{n-1})]$
- Donc $DFT(P) \cdot DFT(Q) = [P(w^0) \cdot Q(w^0); \dots; P(w^{n-1}) \cdot Q(w^{n-1})]$
 $= [PQ(w^0); PQ(w); \dots; PQ(w^{n-1})]$
 $= DFT(PQ)$
- Or, si w est une racine primitive n-ième, la DFT est inversible
- D'où l'algorithme :
 - $PQ = DFT^{-1}(DFT(P) \cdot DFT(Q))$
 - Trois transformées en $O(n \log(n))$ et n produits termes à termes

Produit rapide de polynômes

ω racine n^{ème} de l'unité

1. $H = DFT(P) = [\dots ; \sum_j p_j \omega^{kj} ; \dots]$
 - $H_k = P(\omega^k)$
2. $DFT(P \cdot Q) = DFT(P) \cdot DFT(Q)$ terme à terme
 - $C_k = H_k \cdot G_k = P(\omega^k) \cdot Q(\omega^k) = PQ(\omega^k)$
3. $PQ = DFT^{-1}(DFT(P) \cdot DFT(Q))$
 $= [\dots ; \sum_k C_k \omega^{kj} ; \dots]$

⇒ Complexité : 3 transformations $O(n \log(n))$
 1 produit terme à terme $O(n)$

DFT et racines n-ièmes de l'unité

- Racine n-ième : racine de X^n-1
- Racine primitive : n est le plus petit ordre
- Théorème : racine primitive n-ième $\in F_q$ ssi $n \mid q-1$
 - Ex : $786433-1 = 2^{18} \cdot 3$
- Sinon corps de scindage de X^n-1
 - Ex : racine 4 ième modulo 31 : pas de racine 4-ième dans $Z/31Z$
 - $X^4-1 = (X-1)(X+1)(X^2+1)$
 - On se place dans le corps $Z/31Z[X]/X^2+1$
 - On prend une racine de X^2+1 dans ce corps : i
 - Alors $i^2=-1=30, i^3=-i, i^4=1$

Anneau quotient $G[X]/P$

- Pour G un corps, P un polynôme de degré d
- l'ensemble des polynômes de degré strictement inférieur à d et muni de l'addition et de la multiplication modulo P est un anneau commutatif
- Si $|G|=q$, alors c'est un anneau à q^d éléments
- Si P irréductible, Euclide nous indique que c'est un corps
- Ex: $Z/3Z[X] / (X^2+X-1) = \{0,1,2,X,X+1,X-1,2X,2X+1,2X-1\}$
 - Et $(X+1)(2X+1) = 2X^2+1 = 2(X^2+X-1)+X = X$
- Proposition : $\forall p$ premier, $\forall d > 0$, Il existe des irréductibles
- Corollaire : Il existe un corps fini de cardinal p^d
- Théorème : ce sont les seuls (à iso près)

Test d'irréductibilité

- Théorème : $(X^q)^r - X$ est le produit de tous les polynômes unitaires irréductibles de $GF(q)[X]$ dont le degré divise r.
- Test d'irréductibilité [Ben-Or] pour $P \in GF(q)[X]$
 - Si $\text{pgcd}(P, P^q) \neq 1$ Alors renvoyer « non »
 - // *Maintenant P est sans carrés*
 - $W = X$
 - Pour $d=1$ jusqu'à $d^oP / 2$ Faire
 - $W \equiv W^q [P]$
 - Si $\text{pgcd}(W-X, P) \neq 1$ Alors renvoyer « non »
 - Renvoyer « oui »

Construction d'un corps fini

- Combien d'irréductibles de de degré d ?
 - près d'un sur d
- Tirage aléatoire : avec une espérance de d tirages
- L'arithmétique classique est plus rapide si le polynôme irréductible est creux
 - Ex: $AB=HX^d+L \text{ mod } (X^d+a) = -aH + L$
 - Rechercher d'abord des polynômes sous la forme X^d+a, X^d+bX^k+a
- Une fois que l'on a un polynôme irréductible, on a les opérations arithmétiques, le corps est construit !

Deuxième construction : générateurs

- $GF(q^d)^*$ reste cyclique
 - © Il existe des générateurs
- Ex: dans $Z/3Z[X] / (X^2+X-1) = \{0,1,2,X,X+1,X+2,2X,2X+1,2X+2\}$
 - $(X+1)^0 = 1$
 - $(X+1)^1 = X+1$
 - $(X+1)^2 = X^2+2X+1 = X+2$
 - $(X+1)^3 = (X+2)(X+1) = 2X$
 - $(X+1)^4 = 2$
 - $(X+1)^5 = 2X+2$
 - $(X+1)^6 = 2X+1$
 - $(X+1)^7 = X$
 - $(X+1)^8 = 1$

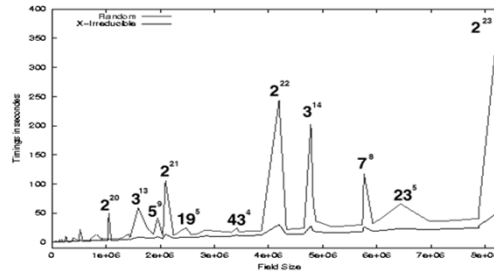
Tester un générateur

1. Exhaustif, on calcule toutes les puissances de R jusqu'à trouver 1
2. L'ordre de R divise q^d-1 , donc
 - Si $\forall p$ premier et divisant $q^d-1, R^{(q^d-1)/p} \not\equiv 1 [P]$
 - Alors R est un générateur
- On a remplacé des opérations polynomiales par des opérations sur des indices
 - accélération d'un facteur au moins d^2 !

Polynômes primitifs

- Calcul des tables : calculer toutes les puissances de R
- Plus rapide si P est creux et R est le plus simple possible
- ⊙ Il existe des P tels que X est un générateur
Appelés primitifs, ou X-irréductibles
- Il y en a $\phi(q^d-1)/d$ parmi p^r polynômes
Pour $p < 2^{32}$, cela donne une espérance $< 12d$

Efficacité de X-irréductibilité



Temps de génération des tables de logarithme discret sur PII, 333 MHz

Construction des grands corps finis

- Construction avec tables :
 - ⊙ Arithmétique rapide
 - ⊙ Besoin de $O(p^k)$ unités de mémoire
 - limitée par la taille de RAM
- Construction polynomiale :
 - ⊙ Pas de problème de mémoire
 - ⊙ Arithmétique polynomiale
- Construction d'un grand corps $GF(p^d)$ avec $d=kr$
 - Construire $GF(p^k)$ efficace et tenant en mémoire
 - Trouver un polynôme P irréductible creux de degré r dans $GF(p^k)$
 - $GF(p^d) \cong GF(p^k)[X] / P$

Casser le Logarithme Discret

- Recherche exhaustive : $O(n)$ multiplications
- [Shanks'71] Baby Step Giant Step $O(n^{1/2})$ multiplications, espace aussi en $O(\sqrt{n})$
- [Pollard'78] méthode Rho $O(n^{1/2})$ multiplications, $O(\log(n))$ mémoire
- [Pohlig-Hellman'78] Si il faut factoriser $n = |G| = \prod p_i^{e_i}$
- Alors $O(\sum e_i (\log(n) + \sqrt{p_i}))$
- Les calculs d'index : la meilleure complexité actuelle
 - [Coppersmith'86] pour $GF(2^c)^*$: $L_2^c[1/3, c < 1.587] \approx O(n^{1/3})$
 - NFS pour Z/pZ^* : $L_p[1/3, 1.923] \approx O(n^{1/3})$
 - Non applicable dans tous les groupes (comme les courbes elliptiques sur un corps fini)

Baby step Giant step

- $h=g^x$ dans un groupe G de cardinal n. Soit $m = \lceil \sqrt{n} \rceil$
- x pourra s'écrire $x = i m + j$ donc $g^x = g^{im+j}$
- Ou encore $h (g^{-m})^i = g^j$
 1. On calcule et on stocke les m premiers g^j $O(m)$ mult
 2. On trie les m premiers g^j $O(m \log(m))$ comp
 3. Inverser, $y = g^{-m}$ $O(\log(m)^2)$ mult
 4. Pour i de 1 à m $O(m \log(m))$ comp
si h y est un des g^j renvoyer $x = i m + j$
- Complexité : $O(\sqrt{n})$ opérations arithmétiques, $O(\sqrt{n} \log(n))$ tests
- ⊙ $O(\sqrt{n})$ espaces mémoire

Méthode rho !

- Il faut trois sous ensembles S_1, S_2, S_3 de G, $n = |G|$
Ex: dans Z/pZ , $S_1 = \{u \equiv 1 [3]\}$, $S_2 = \{u \equiv 0 [3]\}$, $S_3 = \{u \equiv 2 [3]\}$
- $$u_{k+1} = f(u_k) = \begin{cases} hu_k & \text{si } u_k \in S_1 \\ u_k^2 & \text{si } u_k \in S_2 \\ gu_k & \text{si } u_k \in S_3 \end{cases}$$
- Donc $u_i = g^{a_i} h^{b_i}$
- Or, il existe i tel que $u_i = u_{2i}$ mais la plupart du temps $b_i \neq b_{2i}$!
- Dans ce cas $g^{a_i} h^{b_i} = g^{2a_i} h^{2b_i}$, ou encore $h^{b_i-b_{2i}} = g^{2a_i-a_i}$, or $h = g^c$, donc les indices doivent vérifier $(b_i-b_{2i}) x \equiv a_{2i}-a_i [n]$
- Ce qui nous donne $x \equiv (a_{2i}-a_i)(b_i-b_{2i})^{-1} [n]$.
- Complexité : $O(\sqrt{n})$ en général si f est bien aléatoire, $O(\log(n))$ mémoire

Casser le logarithme discret par le calcul d'index (crible)

- Trouver x , tel que $a \equiv g^x$ dans le corps fini à q éléments ?
1. Trouver des équations du type : $x_1 x_2 = y_1 y_2$
 2. Passer au logarithme :
 - $\log_g(x_1) + \log_g(x_2) = \log_g(y_1) + \log_g(y_2)$
 - Certains sont connus : par exemple $\log_g(1)=0, \log_g(g)=1$
 - De proche en proche on peut donc déterminer par résolution de système linéaire, une BASE de logarithmes discrets connus
 3. Il ne restera plus qu'à exprimer a dans cette base :
 - Choisir y au hasard
 - Décomposer ag^y dans la BASE
 - Si cela ne réussit pas recommencer au 3. ou augmenter la BASE

Exemple de calcul d'index

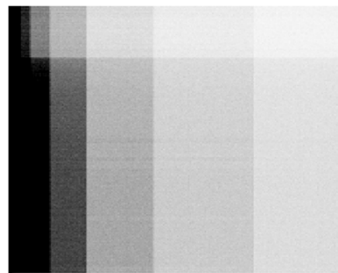
- Trouver x , tel que $17 \equiv 3^x \pmod{1013}$?
- On sait que 1013 est premier et que 3 est générateur : $Z_{1013}^* = \{3\}$
- Après tirages aléatoires on a retenu {309, 311, 503, 511, 703} dont la factorisation de g^x est simple :
 - $3^{309} \equiv 132 = 2^2 \cdot 3 \cdot 11 \pmod{1013}$
 - $3^{311} \equiv 175 = 5 \cdot 7 \pmod{1013}$
 - $3^{503} \equiv 75 = 3 \cdot 5^2 \pmod{1013}$
 - $3^{511} \equiv 770 = 2 \cdot 5 \cdot 7 \cdot 11 \pmod{1013}$
 - $3^{703} \equiv 330 = 2 \cdot 3 \cdot 5 \cdot 11 \pmod{1013}$
- On a à résoudre le système linéaire suivant ($\log_3 3=1$):

$$\begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 2 & 1 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \log_3 2 \\ \log_3 5 \\ \log_3 7 \\ \log_3 11 \end{bmatrix} = \begin{bmatrix} 308 \\ 311 \\ 502 \\ 511 \\ 702 \end{bmatrix} \quad [1012 = 2^2 \cdot 11 \cdot 23]$$

Solution de l'exemple

- Différence des deux dernières lignes
 - $\log_3 7 = 511 - 702 = -191 \equiv 821 \pmod{1012}$
- $$\begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} \log_3 2 \\ \log_3 5 \\ \log_3 11 \end{bmatrix} = \begin{bmatrix} 308 \\ 502 \\ 702 \end{bmatrix}$$
- $2 L_3 - L_2 - L_1$ donnent : $\log_3 11 = 594, 2 \log_3 2 = 726$ et $2 \log_3 5 = 502$, mais 2 n'est pas inversible modulo 1012
 - donc $\log_3 2 = 363$ ou $363 + 506$, et $\log_3 5 = 251$ ou $251 + 506$
 - Par exponentiation rapide on vérifie que $\log_3 2 = 326$ et $\log_3 5 = 757$
 - Il reste à trouver $17 \cdot 3^y \equiv 17 \cdot 2^2 \cdot 3^2 \cdot 5^2 \cdot 7^2 \cdot 11^2$
 - Au hasard, $17 \cdot 3^{200} \equiv 480 = 2^5 \cdot 3 \cdot 5^2 \pmod{1013}$
 - Donc $\log_3 17 = 5 \cdot 363 + 1 + 3 \cdot 757 - 200 = 349$

Matrice creuse de Crible



[Thomé 2005]

Noir \Leftrightarrow densité $\geq 0.01\%$

La recette

- Un algorithme de factorisation simple : rho de Pollard, ECM
- Fabrication massivement parallèle des matrices [Crible]
- Reconstruction par restes chinois
- Filtrage : élimination de Gauß modulaire structurée sur matrices gigantesques
- Méthode itérative : Block Coppersmith
- Multiplication de polynômes matriciels par FFT
- Noyau d'algèbre linéaire dense sur corps finis

Exemple SNFS-1039 bits

- Les ingrédients
 - Matrice initiale de 16 570 808 010 lignes
 - Après filtrage de Gauß : 71 573 531 \times 71 773 331 avec 9 681 804 348 coefficients non nuls
 - Wiedemann en blocs de taille 256
- La cuisson
 - Multiplications matrices-blocs : 59 jours sur un cluster de 110 Pentium D à 2.3GHz + 96 DualCore2Duo à 4.2.66 GHz
 - Polynôme générateur matriciel : 8 heures sur un cluster de 64 cœurs
- Le gâteau
 - 50 dépendances linéaires
 - Pour 47 solutions du système linéaire

La fève

- $2^{1039} - 1 = 5080711 \cdot$
 $558536666199362912607492046583159449686465270184$
 $88637648010052346319853288374753 \cdot$
 $207581819464423827645704813703594695162939708007$
 $395209881208387037927290903246793823431438841448$
 $348825340533447691122230281583276965253760914101$
 $891052419938993341097116243589620659721674811617$
 $49004803659735573409253205425523689$
[K. Aoki, J. Franke, T. Kleinjung, A. K. Lenstra, D. A. Osvik 05-2007]
 « Les facteurs premiers sont disproportionnés, l'effort de calcul équivaut
 environ à l'effort nécessaire pour casser RSA-768 »

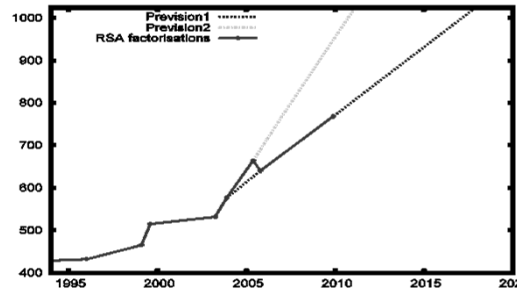
Exemple RSA-768 bits

- Les ingrédients
 - Matrice initiale de 64 334 489 730 lignes eq. 1500 AMD64 année
 - Après filtrage de Gauß : 192 796 550 × 192 795 550 avec 27 797 115 920 coefficients non nuls
 - Wiedemann en blocs de taille 512
- La cuisson
 - Multiplications matrices-blocs et Polynôme générateur matriciel : 119 jours sur un cluster de 110 Pentium D à 2×3GHz + 56 Dual hex core + des ALADDIN-GSK
- Le gâteau
 - 512 dépendances linéaires
 - Pour 460 solutions du système linéaire

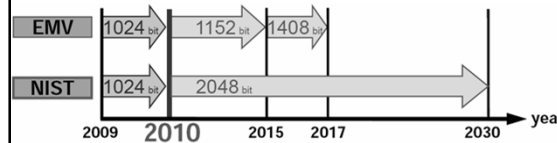
RSA-768, 232 chiffres par NFS, deux facteurs de 384 bits et 116 chiffres

- 123018668453011775513049495838496272077285356959
 $533479219732245215172640050726365751874520219978$
 $646938995647494277406384592519255732630345373154$
 $826850791702612214291346167042921431160222124047$
 $9274737794080665351419597459856902143413 =$
 $334780716989568987860441698482126908177047949837$
 $137685689124313889828837938780022876147116525317$
 $43087737814467999489 \times$
 $367460436667995904282446337996279526322791581643$
 $430876426760322838157396665112792333734171433968$
 10270092798736308917
[Kleinjung, Aoki, Franke, Lenstra, Thomé, Bos, Gaudry, Kruppa, Montgomery, Osvik, Riele, Timofeev, Zimmermann, 12 Déc 2009]

Clefs RSA sur 1024 bits ?



Recommandations EMV, NIST



Niveau de Sécurité estimés *[Nessie]*

bits	80 (SKIPJACK)	112 (3-DES)	128 AES-small	192 AES-medium	256 AES-large
RSA	1024	2048	3072	7168	13312
DLP	1024	2048	3072	7168	13312
EC DLP	192	224	256	384	521

- Factorisation $\approx \text{Ln}[1/3, 1.923+\alpha(1)] = \exp((1.923+\alpha(1)) \ln(n)^{1/3} \ln(\ln(n)))^{2/3}$
- Calcul d'index pour le log discret $\approx \text{Lp}[1/3, 1.923+\alpha(1)]$
- DLP sur courbes elliptiques : Pollard rho $\approx O(n^{1/2})$

Plan

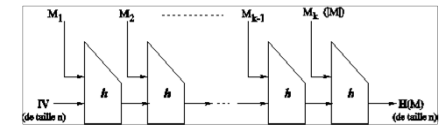
- Introduction
- Histoire des codes secrets
- Complexité, Algorithmique, Information
- Confidentialité
- Intégrité, Authentification, Non-Répudiation
 - Fonctions de hachage cryptographique
 - Paradoxe des anniversaires
 - MAC, Signatures numériques
 - Architectures de Chiffrement
- Législation

Fonction de hachage : résumé

- $H : \{0,1\}^* \rightarrow \{0,1\}^n$
- Opère sur un message de longueur quelconque et fournit une valeur de hachage (le résumé) de longueur fixe n
 - Uniformité : $\text{Probabilité}(H(M)=i) = 1/2^n$
 - Sécurité : à partir du résumé il est impossible de retrouver le texte
- Niveaux de sécurité
 - Résistance à la préimage : dût de trouver x tq $y=H(x)$
 - Résistance à la 2nde préimage : ayant x, dût de trouver x' tq $H(x)=H(x')$
 - Résistance aux collisions : dût de trouver x et x' tq $H(x)=H(x')$
- MD5 (128 bits [collisions 2004]), SHA-1 160 bits [collisions en théoriquement 2^{51} opérations], etc.

Calcul d'empreinte

- Itération d'une ronde $h : \{0,1\}^b \times \{0,1\}^n \rightarrow \{0,1\}^n$



- h résiste aux collisions => H aussi [Merkle-Damgård]
 - Sinon : collision sur h à la première différence
- Attaque par force brute
 - Combien de x doit-on générer pour trouver une collision ?
 - ⇒ Entre 2 et 2^{n+1}

Paradoxe des anniversaires

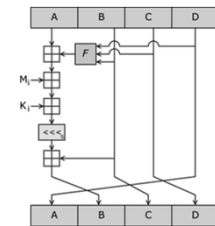
- Population de k personnes, n jours dans l'année
 - Nombres de combinaisons d'anniversaires différents : $A_i^k = \frac{n!}{(n-k)!}$
 - Probabilité pour deux personnes au moins aient leurs anniversaires le même jour : $1 - \frac{A_i^k}{n^k}$
 - $k=9 \Rightarrow P = 0.1$; $k=23 \Rightarrow P > 0.5$; $k=60 \Rightarrow P > 0.994$
- Plus généralement, si $k > 1.18\sqrt{n}$, alors plus d'une chance sur deux de collision !

Attaque de Yuval

- Entrées : x_1 légitime, x_2 frauduleux, h sur n bits
- Sorties : $x_1' \approx x_1$ et $x_2' \approx x_2$ tq $h(x_1')=h(x_2')$
 1. Générer $t = 2^{n/2}$ modifications mineures de x_1 , notées x_1'
 2. Pour chacune, calculer $h(x_1')$
 3. Générer des x_2' , modifications mineures de x_2 , jusqu'à collision avec un x_1'
- Application : Répudiation
 - ☛ envoyer x_2' et soutenir que en fait x_1' avait été envoyé
- Fiabilité des fonctions de hachage en cryptologie :
 - ⇒ Bloc de hachage > 128 bits

MD5

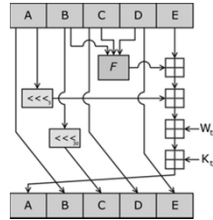
- Blocs de 512 bits vers 128
 - 16 sous blocs M_i de 32 bits
 - 64 constantes fixées K_i
- 64 rondes sur 4 sous blocs de 32 bits
 - 16 rondes où F est $D \oplus (B \text{ and } (C \oplus D))$
 - 16 rondes où F est $C \oplus (D \text{ and } (B \oplus C))$
 - 16 rondes où F est $B \oplus C \oplus D$
 - 16 rondes où F est $C \oplus (B \text{ or } D)$



- Δ Collisions
 - Analytiques :
 - 1 heure sur un cluster [Wang/Feng/Lai/Yu 2004]
 - Deux signatures équivalentes en quelques heures sur 1 PC [Klima 2005]
 - Force brute par le paradoxe des anniversaires faisable en théorie au moins

SHA-1

- Blocs de 512 bits vers 160
 - 16 sous blocs M_i de 32 bits
 - Étendus en 80 nouveaux blocs W_t
 - 4 constantes fixées K_t
- 80 rondes sur 5 sous blocs de 32 bits
 - 20 rondes où F est $D \oplus (B \wedge (C \oplus D))$
 - 20 rondes où F est $B \oplus C \oplus D$
 - 20 rondes où F est $(B \wedge C) \vee (D \wedge (B \vee C))$
 - 20 rondes où F est $B \oplus C \oplus D$



- Δ Collisions ?
- Force brute : $O(2^{80})$
 - [Manuel 2008] : collision en $O(2^{51})$ appels à h
 - Prévention de la copie de la Xbox MS repose sur SHA-1
 - Standard de signatures électroniques utilise SHA-1

⇒ SHA-256, ...

- Blocs de 512 bits vers 256
 - 16 sous blocs M_i de 32 bits
 - Étendus en 64 nouveaux blocs W_t
 - 64 constantes fixées K_t
- 64 rondes sur 8 sous blocs de 32 bits
 1. $S = (A \ggg_2) \oplus (A \ggg_{13}) \oplus (A \ggg_{22})$
 2. $M = (A \wedge B) \vee (B \wedge C) \vee (C \wedge A)$
 3. $T = S \vee M$
 4. $U = (E \ggg_6) \oplus (E \ggg_{11}) \oplus (E \ggg_{25})$
 5. $C = D \oplus (E \wedge (F \oplus G))$
 6. $V = H \vee U \vee C \vee K_t \vee W_t$
 7. $H = G; G = F; F = E; E = D \vee V; D = C; C = B; B = A; A = T \vee V$

SHA-3

- 9 décembre 2010 : 5 finalistes → printemps 2012
 - BLAKE (Suisse-UK), Grøstl (Danemark-Autriche), JH (Singapour), Keccak (STmicro), Skein (US)

Cycles/octet	SHA	Whirlpool	BLAKE	Grøstl	JH	Keccak	Skein
256-bit hash	17.44		8.24	13.73	13.71	13.68	9.65
512-bit hash	11.65	24.11	7.95	18.29	13.75	12.80	7.82
attaques			?	?	preimage < générique	?	?

Intégrité

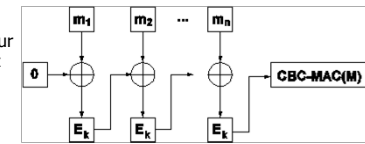
- MAC (Message Authentication Code)
 - Fonction à sens unique paramétrée avec une clef secrète
 - $r = f_k(M)$
 - Seul le possesseur de la clef k peut vérifier l'empreinte :
1. Alice et Bob possèdent une clef k
 2. Alice envoie M et r
 3. Bob vérifie que le r reçu est bien égal à $f_k(M)$ pour le M reçu
- ⇒ Intégrité : Tous les possesseurs de k peuvent vérifier que M n'a pas été modifié puisqu'il correspond à r
- + Authentification si la clef n'est distribuée qu'à deux personnes

MAC

- NE PAS UTILISER
 - $H(K||Message)$
 - $H_k(Message) : H(Message)$ avec $IV = K$
- ⇒ On peut ajouter ce que l'on veut à la fin du message en itérant H
- $H(Message||K)$
- ⇒ Collision sur H donne un MAC valide : $H(x)=H(y) \rightarrow H(xK)=H(yK)$
- Plutôt
 - Enveloppe : $H(K_1||Message||K_2)$
 - NMAC : $H_{K_1}(H_{K_2}(Message))$
 - Hybride : $H_{K_1}(Message||K_2)$
 - Ou encore HMAC ...

CBC-MAC

- AES : 275 Mbits/seconde sur un PIII 500 MHz
- CBC-MAC-AES : 234 Mbits/seconde



HMAC

- $HMAC_k(M) = H(K \oplus pad_1 || H(K \oplus pad_2 || M))$
- MD5-MAC 5 à 20% plus lent que MD5 seul

Signatures numériques

- Authentique : convaincre le destinataire que le signataire a délibérément signé un document
- Infalsifiable : preuve que le signataire a délibérément signé
- Non réutilisable : attachée à un document
- Inaltérable : aucune modification du document signé
- Non reniable : le signataire ne peut répudier le document

Intégrité + Authentification + Non-répudiation

Signatures

- MAC sont symétriques
- Signatures sont asymétriques

Signatures RSA

- Hachage + Clef publique
 - Crypter $H(M)$ avec la clef privée de l'envoyeur
 - Vérifier en décryptant et recalculant le hash
- + Interdire les rejeux : Crypter plutôt $H(M || \text{Date})$

DSS/DSA : SHA-1 + ElGamal/Log discret

- Choisir q premier de 160 bits
 - Trouver $p = kq + 1$ premier de 512 à 1024 bits
 - Tant que $(g \neq 1)$ choisir a et calculer $g \equiv a^k [p]$
 - Choisir x de 160 bits : clef privée
 - $y \equiv g^x [p]$: p, q, g, y publics
1. Alice choisit k aléatoire inférieure à q
 2. Alice calcule et envoie
 - $r = (g^k \text{ mod } p) \text{ mod } q$
 - $s = (k^{-1} (\text{SHA1}(\text{Message}) + xr) \text{ mod } q)$
 3. Bob vérifie la signature ssi $(v == r)$
 - $w \equiv s^{-1} [q]$
 - $u \equiv \text{SHA1}(\text{Message}) w [q]$
 - $t \equiv r w [q]$
 - $v = (g^{u_1} y^{u_2} \text{ mod } p) \text{ mod } q$

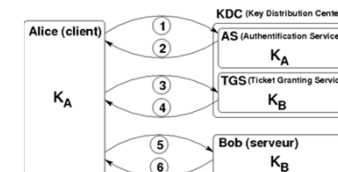
Architectures et protocoles

- Échanger des clefs
 - Diffie-Hellman : man-in-the-middle
 - Échange « out-of-band »
 - Tiers de confiance
 - Clefs secrètes : Kerberos [MIT]
 - Clefs publiques : certificats, PKI
- Sessions cryptées : systèmes hybrides de chiffrement
 - PGP
 - SSH

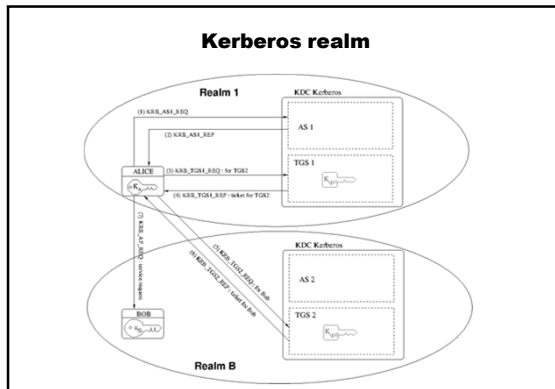
Échange de clef

- Challenge
 - KDC envoie un « challenge », nombre aléatoire que le client doit crypter avec sa clef secrète : authentification
- Tiers de confiance
 - Pas besoin d'échanger entre tous les utilisateurs possibles, juste avec le KDC
 - Le KDC peut alors distribuer des clefs de session ciblées
- Horodatage
 - Empêche les re-jeux

Kerberos



- Kerberos = Challenge + Tiers + Horodatage
1. Bonjour
 2. $K_{A,A} || \text{challenge}$
 3. Je veux parler à Bob || réponse challenge datée
 4. $K_{A,B} || \text{Ticket daté pour Bob contenant } E_{K_B}(K_{A,B})$
 5. Ticket pour Bob || Je suis Alice daté
 6. Bonjour Alice daté



- ### Certificats Numériques
- Comment rattacher une clef publique à son propriétaire ?
 - Certificat numérique : association identité + clef
 - Comment authentifier un certificat ?
 - Les certificats sont signés par des Autorités
 - Comment récupérer un certificat ?
 - Des annuaires sont maintenus
 - Comment gérer / sécuriser ce mécanisme
 - Auto signatures des Autorités
 - Architectures à clef publiques : PKI ...

- ### Système hybride : clef secrète / clef publique
- Initiée par PGP [Zimmermann]
 - Cryptographie secrète : rapide mais problème du tiers
 - Cryptographie publique : lente
 - SSH
 1. Client contacte le serveur
 2. Serveur envoie ses clefs publiques H fixe et S changée régulièrement + challenge
 3. Client vérifie H correspond à sa BD
 4. Client double crypte clef secrète de session par H et S
 5. Confirmation du serveur
 ⇒ Connexion sécurisée établie

- ### Preuves de sécurité : IND-CCA
- IND : indistinguabilité des chiffrés (sécurité sémantique)
 - Aucune attaque ne peut déduire un seul bit d'information sur le clair associé à un chiffré donné
 - Étant donnés deux clairs et un chiffré correspondant à l'un des deux clairs, il est impossible de déterminer auquel le chiffré correspond
 - CCA : Attaque à texte Chiffré Choisi
 - Capacité pour l'attaquant de déchiffrer tout chiffré de son choix sauf pour certains chiffrés prescrits
 - IND-CCA : résistance IND, même si l'attaquant a CCA

Chiffrement probabiliste : RSA-OAEP

- Optimal Asymmetric Encryption Padding
 - Chiffrement non déterministe
- $OAEP(m) = [X || Y]$ (plus grand que m)
 $= [M \oplus G(r) || r \oplus H(M \oplus G(r))]$
- Avec H et G deux fonctions de hachage cryptographique
- m récupéré par $Y \oplus G(H(X) \oplus Y)$

IND-CCA prouvé seulement pour e=2 et e=3 ...

- ### Simple-RSA (RSA-KEM)
- Chiffrement
 - R aléatoire
 - $C_0 \equiv R^e \pmod n$
 - $K_0 || K_1 = G(R)$
 - $C_1 = M \oplus K_0$
 - $T = MAC_{K_1}(C_1)$
 - Chiffré : (C_0, C_1, T)
 - Déchiffrement
 - $R = C_0^d \pmod n$
 - $K_0 || K_1 = G(R)$
 - Vérification $T \stackrel{?}{=} MAC_{K_1}(C_1)$
 - $M = C_1 \oplus K_0$

Plan

- Introduction
- Histoire des codes secrets
- Complexité, Algorithmique, Information
- Confidentialité
 - Flots
 - Blocs
 - Clefs secrètes
 - Clefs publiques
- Intégrité, Authentification, Non-répudiation

- Un peu de Législation

- Législation française sur la cryptologie**
- 1999 : Clefs secrètes sur plus de 40 bits sont autorisées
→ Jusqu'à 128 bits uniquement après autorisation
Flou juridique sur les clefs publiques

 - Loi du 13 mars 2000
« La signature électronique est admise en preuve au même titre que l'écrit sur support papier »

 - Loi du 21 juin 2004, article 30
 - « Authentification et intégrité sont libres »
 - « Tout autre importation de cryptologie depuis l'UE est soumise à déclaration préalable auprès du premier ministre »
 - « Les caractéristiques et le code source sont tenus à la disposition du premier ministre »
 - « Toute exportation de cryptologie vers l'UE est soumise à autorisation du premier ministre »

Utilisation, transfert depuis un État membre de la communauté européenne, importation et fourniture de moyens de cryptologie en France

http://www.ssi.gouv.fr/fr/reglementation/regl_crypto.html

Moyen de cryptologie	UTILISATION	IMPORTATION ET TRANSFERT DEPUIS UE	FOURNITURE
transporté ou importé pour usage exclusivement personnel		LIBRE	SANS OBJET
assurant exclusivement des fonctions d'authentification ou de contrôle d'intégrité		LIBRE	
autre	LIBRE	DÉCLARATION	

Exportation et transfert de moyens de cryptologie depuis la France

http://www.ssi.gouv.fr/fr/reglementation/regl_crypto.html

Moyen de cryptologie	TRANSFERT VERS UN ÉTAT de l'UE	EXPORTATION VERS SEPT ÉTATS IDENTIFIÉS (1)	EXPORTATION VERS D'AUTRES ÉTATS
authentification ou contrôle d'intégrité	LIBRE		
de type grand public	DECLARATION		
employant des clés cryptographiques de grande taille (2)	DECLARATION	DECLARATION [Licence générale communautaire]	AUTORISATION [Licence individuelle ou globale]
permettant la cryptanalyse	AUTORISATION [Licence individuelle ou globale]		

(1) Australie, Canada, États-Unis d'Amérique, Japon, Nouvelle-Zélande, Norvège et Suisse
 (2) a) un algorithme cryptographique symétrique employant une clé de longueur supérieure à 56 bits
 b) un algorithme cryptographique asymétrique fondé soit sur la factorisation d'entiers de taille supérieure à 512 bits, soit sur le calcul de logarithme discret dans un groupe multiplicatif d'un corps fini de taille supérieure à 512 bits ou dans un autre type de groupe de taille supérieure à 112 bits.