Introduction à la cryptographie

Sébastien VARRETTE

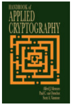
Université du Luxembourg - Laboratoire LACS, LUXEMBOURG
CNRS/INPG/INRIA/UJF - Laboratoire LIG-IMAG
Sebastien.Varrette@imag.fr
http://www-id.imag.fr/~svarrett/

Cours "Cryptographie & Securité Réseau" Master Info
Université de Yaoundé

Déroulement du cours

- Cours intensif (∼40h) sur 2 semaines (17-28 avril)
- Objectif du cours
  - Introduction/Sensibilisation à la cryptographie
    - Cryptographie à clé secrète
    - Cryptographie à clé publique
  - Programmation en C de divers algorithmes
  - Signatures Electroniques
  - Architectures PKI
  - Sécurité Systèmes & Réseaux
    - Programmation sécurisée en C
    - Manipulation sur machine (environnement Linux)

Quelques références bibliographiques. . . (avant que j'oublie)



MENEZES A. J., VANSTONE S. A. and OORSCHOT P. C. V., *Handbook of Applied Cryptography*, Computer Sciences Applied Mathematics Engineering, CRC Press, Inc., 1st edition, 1996,
http://www.cacr.math.uwaterloo.ca/hac/

SCHNEIER B., *"Cryptographie Appliquée"*, Vuibert, Wiley and International Thomson Publishing, NY, 2nd edition, 1997.
http://www.schneier.com/book-applied.html

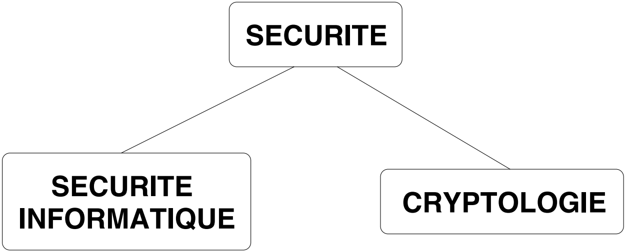
STINSON D.R, *Cryptography : Theory and Practice*, Chapman & Hall/CRC Press, 2nd edition, 2002.
http://www.cacr.math.uwaterloo.ca/~dstinson/CTAP2/CTAP2.html

EBRAHIMI T., LEPRÉVOST F. and WARUSFELD Ed., *Cryptographie et Securité des systèmes et réseaux*, Hermes/Lavoisier,
http://www-id.imag.fr/~svarrett/book_secu_mult.html

Plan

Principes généraux de la cryptographie

Notion de *Sécurité*



Notion de *Cryptologie*

"Science du secret" avec deux composantes complémentaires

1. la **cryptographie** : étude et conception des procédés de chiffrement des informations
2. la **cryptanalyse** : analyse des textes chiffrés pour retrouver les informations dissimulées

Notion de *Cryptologie*

"Science du secret" avec deux composantes complémentaires

1. la **cryptographie** : étude et conception des procédés de chiffrement des informations
2. la **cryptanalyse** : analyse des textes chiffrés pour retrouver les informations dissimulées

- Bien distinguer cryptographie/stéganographie :
  - *cryptographie* : transforme un message clair en cryptogramme
  - *stéganographie* : dissimule l'existence même de l'information secrète (encre sympatique etc...)

> Je suis très émue de vous dire que j'ai
> bien compris, l'autre jour, que vous avez
> toujours une envie folle de me faire
> danser. Je garde un souvenir de votre
> baiser et je voudrais que ce soit
> là une preuve que je puisse être aimée
> par vous.[...]

Illustration de l'utilisation de la stéganographie : extrait d'une lettre de Georges Sand.

Un mot sur la stéganographie...

- Information non-chiffrée

  Connaissance de l'existence de l'information

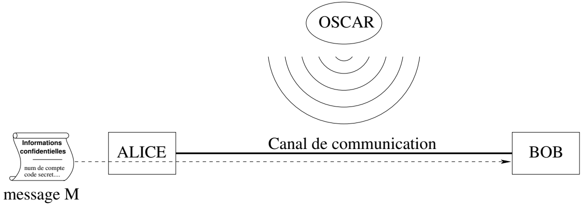
  =

  Connaissance de l'information

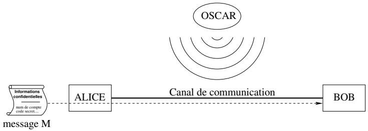
- Exemples :
  - Message couvert : tablette couverte de cire, crâne du messager
  - Message invisible : encre sympathique (Pline 1er siècle av. JC)
  - Message illisible : Micro-film sous forme de point
  - Message subliminal : traitement de texte des ministres de M. Thatcher
- Théorie :
  - Faible niveau de sécurité
  - En pratique ça marche ...(11/09/2001 ?)
  - Utilisée également pour le Watermarking (JPEG, MP3-MPEG, etc)

Terminologie (1)

- Protagonistes traditionnels :
  - **Alice** et **Bob** : souhaitent se transmettre des informations
  - **Oscar** : un opposant qui souhaite espionner Alice et Bob
- Objectif fondamental de la cryptographie
  - permettre à Alice et Bob de communiquer sur un canal peu sûr
  - Oscar ne doit pas comprendre ce qui est échangé.

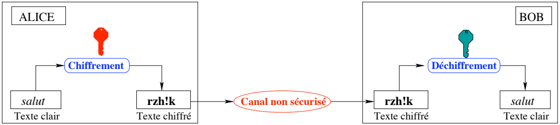


Terminologie (2)



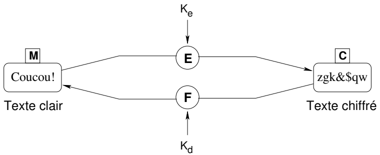
- **Texte clair** : information qu'Alice souhaite transmettre à Bob
  - Ex : texte en français, donnée numérique etc...
- **Chiffrement** : processus de transformation d'un message M de telle manière à le rendre incompréhensible
  - Basé sur une *fonction de chiffrement* E
  - On génère ainsi un **message chiffré** $C = E(M)$
- **Déchiffrement** : processus de reconstruction du message clair à partir du message chiffré
  - Basé sur une fonction de déchiffrement D
  - On a donc $D(C) = D(E(M)) = M$ (D et E sont injectives)

Algorithmes de cryptographie



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

Relation fondamentale



- En pratique : E et D sont paramétrées par des clés K_e et K_d

$$\begin{cases} E_{K_e}(M) = C \\ D_{K_d}(C) = M \end{cases} \tag{1}$$

- $K_e, K_d \in \{\text{espace des clés}\}$.
- Définit deux catégories de systèmes cryptographiques :
  1. Systèmes à clé secrète (ou symétriques) $(K_e = K_d = K)$
  2. Systèmes à clé publique (ou asymétriques) $(K_e \neq K_d)$

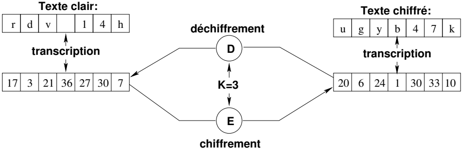
Exemple : representation mathématique de E et D

- Pour permettre l'analyse des systèmes cryptographiques
  - représentation mathématique des messages M et C. Ex :

a	b	...	y	z	0	1	...	9	␣	.
0	1	...	24	25	26	27	...	35	36	37

    Ici : vocabulaire de $n = 38$ caractères. (code ASCII : n=256)
  - fonctions E et D vues comme des fonctions mathématiques
    - basée le plus souvent sur l'arithmétique modulaire
    - Exemple : chiffrement par décalage

$$\begin{cases} E_K(M) = M + K \mod n \\ D_K(C) = C - K \mod n \end{cases}$$



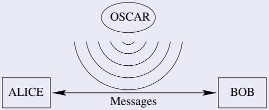
A quoi sert la cryptographie ?

CAIN (Confidentialité - Authentification - Intégrité - Non-répudiation)

- **Confidentialité** des informations stockées/manipulées
  - utilisation d'un algorithme de chiffrement.
  - *empêcher l'accès* aux infos pour ceux qui ne sont pas autorisés.
- **Authentification** d'utilisateurs/de ressources
  - utilisation d'algorithmes d'authentification.
  - Alice s'identifie à Bob en prouvant qu'elle connaît un secret S, (ex : un mot de passe).
- **Intégrité** des informations stockées/manipulées
  - vérifier que les infos transmise n'ont pas subie d'*altérations*
- **Non-répudiation** des informations
  - utilisation d'algorithmes de signatures
  - empêcher un utilisateur de se dédire

Les grands types de menaces

Menaces passives

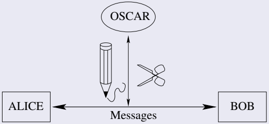


- Oscar ne fait qu'écouter le message.
- menace la *confidentialité*

- une information sensible parvient également à une autre personne que son destinataire légitime.

Les grands types de menaces (2)

Menace active



- Oscar peut modifier le contenu des messages échangés.
- menace l'*intégrité* de l'information.

- Exemple d'attaques actives :
  - l'usurpation d'identité (de l'émetteur ou du recepteur)
  - l'altération/modification du contenu des messages ;
  - la destruction de messages/ le retardement de la transmission ;
  - la répétition de messages (jusqu'à engorgement)
  - la répudiation de message : l'émetteur nie avoir envoyé le message.

Les attaques sur un chiffrement

- *Cryptanalyse* : étude de la sécurité des procédés de chiffrement utilisés en cryptographie
- Niveaux d'attaques possibles :
  1. *Texte chiffré connu* : Seul C est connu d'Oscar
  2. *Texte clair connu* : Oscar connaît C et M correspondant
  3. *Texte clair choisi* : $\forall M$, Oscar peut obtenir C
  4. *Texte chiffré choisi* : $\forall C$, Oscar peut obtenir M
- garantir la confidentialité $\Longrightarrow$ Oscar ne peut pas :
  - trouver M à partir de $E(M)$
  - trouver la méthode de déchiffrement D à partir d'une séquence $\{M_i, \mathrm{E}(M_i)\}$.

Algorithmes d'attaques

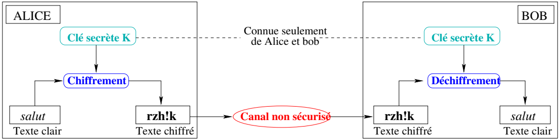
1. Attaque brutale
   - Énumérer toutes les valeurs possibles de clefs
   - 64 bits $\Longrightarrow 2^{64}$ clefs $= 1.844 * 10^{19}$ combinaisons
     - Un milliard de combinaisons/s $\Rightarrow$ 1 an sur 584 machines
2. Attaque par séquences connues
   - Deviner la clef si une partie du message est connue
     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

Bref historique des codes secrets...

- Cryptographie Ancienne
  - Transposition Sparte (5ème siècle av JC)
  - Substitution César (1er siècle av JC), Vigenère (XVI ème)
- Cryptanalyse des codes mono et poly alphabétiques
  - El Kindi (IXème siècle)
  - Babbage/Kasiski (XIXème siècle)
- Mécanisation de la cryptographie et de la cryptanalyse
  - Enigma (1918)
- Vers un chiffrement parfait : Vernam, théorie de l'information
- Standard de chiffrement à clé secrète : DES (1977), AES(2000)
- Cryptographie à clé publique (1976)

Cryptographie à clé secrète

Principe



- $K_e = K_d = K$ (clé privée convenue secrètement par Alice et Bob)
  - En pratique : grande efficacité en terme de temps de calculs
  - Inconvénient : la clé K doit rester secrète.
- **Analogie : coffre-fort !**
- Historiquement le premier type de chiffrement utilisé.
- Fourni le seul chiffrement théoriquement indéchiffrable
  - Chiffrement de Vernam (ou one-time password)
  - Démonstration du mathématicien Claude Shannon (1949)

Chiffrement symétrique : outils de base utilisés (1)

- A la base des chiffrements à clé secrète :
  - Substitution : **remplacer** chaque élément par un autre.

M	E	S	S	A	G	E		S	E	C	R	E	T
N	F	T	T	B	H	F		T	F	D	S	F	U

  - Transposition (ou permutation) : **changer** l'ordre des éléments

M	E	S	S	A	G	E		S	E	C	R	E	T
E	M	S	S	G	A	E		E	S	R	C	T	E

Chiffrement symétrique : outils de base utilisés (2)

- Autres opérations utiles :
  - Arithmétique modulaire dans $\mathbb{Z}_n$ $a, b, n \in \mathbb{N}$, avec $n \geq 2$.

    $$a \equiv b \mod n \iff n \text{ divise } a - b$$

    En pratique : $b =$ reste de la division euclidienne de a par n.
    $5 \equiv 1 \mod 4$ et $-3 \equiv 125 \mod 128$
  - Notions $\pm$ associées : Primalité, Euclide, Th. des restes chinois, Gauss, Euler...
  - opération XOR (ou exclusif $\oplus$)

$\oplus$	0	1
0	0	1
1	1	0

- Opération bijective (bijection inverse : $\oplus$!)
- correspond à une addition bit-à-bit modulo 2.

Les premiers procédés

Initialement, le secret echangé était la technique mise en oeuvre

- 400 av JC : esclave envoyé à Aristogoras par Histaïus
- Ve av JC : premières transpositions monoalphabétiques
    - Chiffrement de type anagramme : mélange les lettres du message
    - Confusion sur la syntaxe mais chaque lettre conserve sa valeur
    - Clé de chiffrement complexe
    - Principe des scytales spartiate (coms entre chefs des armées)

Les premiers procédés (2)

- IVe : premières substitution
  - Chiffrement par changement d'alphabet
  - Ex : Kama-sutra : recommande aux femmes de maîtriser le *mlecchita-vikalpà* (art de l'écriture secrète)
- 150 avant JC : carré de Polybe

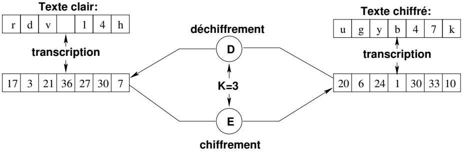
	1	2	3	4	5
1	a	b	c	d	e
2	f	g	h	i	j
3	k	l	m	n	o
4	p	q	r	s	t
5	u	v	x	y	z

- alphabet de 25 lettres (pas de 'w' ou i et j regroupés)
- remplace les lettres par des chiffres
- codage d'une lettre = coordonnée dans le tableau
  - A $\longrightarrow$ 11
  - B $\longrightarrow$ 12 ...

Le chiffrement de César...

- Chiffrement par décalage avec $K = 3$.

$$\begin{cases} E_K(M) = M + K \mod n \\ D_K(C) = C - K \mod n \end{cases}$$



- Seulement n façons différentes de chiffrer un message
  - code très peu sûr (recherche exhaustive facile)
  - avantage de la simplicité
    - employé par les officiers sudistes (guerre de Sécession)
    - réemployé sur les forums de News : `ROT-13` ($K = 13$)
- Généralisation : chiffrement affine
  - $E_{(a,b)}(M) = a * M + b \mod n$ (pour $a \in \mathbb{Z}_n^{\times}$)
  - cf TD

Cryptanalyse des substitutions mono-alphabétique

- Rappel : substitution mono-aphabétique : on remplace chaque lettre par une lettre différente.

M	E	S	S	A	G	E		S	E	C	R	E	T
↓	↓	↓	↓	↓	↓	↓		↓	↓	↓	↓	↓	↓
N	F	T	T	B	H	F		T	F	D	S	F	U

- Nombre de possibilités (alphabet de 26 lettres) ?

Cryptanalyse des substitutions mono-alphabétique

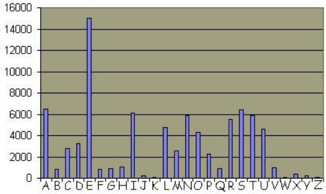
- Rappel : substitution mono-aphabétique : on remplace chaque lettre par une lettre différente.

M	E	S	S	A	G	E		S	E	C	R	E	T
N	F	T	T	B	H	F		T	F	D	S	F	U

- Nombre de possibilités (alphabet de 26 lettres) ?
  - chiffrement de 'A' : 26 possibilités
  - chiffrement de 'B' : 25 possibilités
  - ... $\longrightarrow 26! \simeq 4 * 10^{26}$ possibilités
  - Ordre de grandeur de comparaison : plier 50 fois sur elle-même une feuille de papier (épaisseur : 1 dixième de mm)
    $\longrightarrow$ épaisseur de la feuille :
    2^{50} dixième de millimètre $\simeq 1,1 * 10^{8}$ km
    (110 millions de km $\simeq$ 300 fois distance Terre/Lune)

Cryptanalyse des substitutions mono-alphabétique

- MAIS ne cache pas la fréquence d'apparition des symboles !



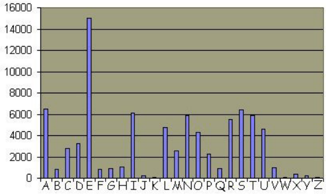
En français, la lettre 'e' apparaît le plus souvent etc...

- Exemple : cryptanalyse du texte suivant :

```
HQYRBHU GX UHQIRUW DYHF GHV DUPHV
```

Cryptanalyse des substitutions mono-alphabétique

- MAIS ne cache pas la fréquence d'apparition des symboles !



En français, la lettre 'e' apparaît le plus souvent etc...

- Exemple : cryptanalyse du texte suivant :

`HQYRBHU GX UHQIRUW DYHF GHV DUPHV`

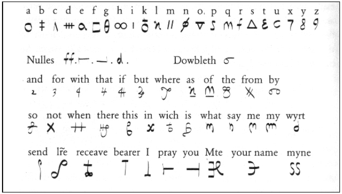
  - Réponse : envoyer du renfort avec des armes

- cryptanalyse proposée par Al Kindi (un savant arabe) au IX^{e} s.
- cf TD

Exemple de chiffrement par code

Chiffrement par code = chiffrement de mots plutôt que de lettres

- Ex : Code de Marie Stuart (XVI^e s.) (nomenclateur)
  - Décapitée pour avoir utilisé un code trop faible quand elle tenta d'assassiner la Reine d'Angleterre.
  - Illustration d'une attaque active
    - Walsingham fait ajouter un P.S aux lettres interceptées pour obtenir les noms des conspirateurs
    - Trop confiante dans son code, Marie Stuart le fit



La cryptographie par substitution polyalphabétique (1)

- Méthode utilisée par Vigenère (1586)
  - la clef définit le décalage pour chaque lettre du message

Texte clair:	L	A		V	I	E		E	S	T		B	E	L	L	E
transcription																
	11	0		21	8	4		4	18	19		1	4	11	11	4
Clé:	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
chiffrement																
	12	14		8	17	18		24	9	20		15	17	20	25	24
transcription																
Texte chiffré:	M	O		I	R	S		Y	J	U		P	R	U	Z	Y

'A' : décalage de 0
'B' : décalage de 1
...
'Z' : décalage de 25
Ex : chiffrement de "`La vie est belle`"
avec la clé "`bonjour`"

La cryptographie par substitution polyalphabétique (2)

- Procédé de Vigenère résistera jusqu'au milieu du XIX^e s.
  - Cryptanalyse de Babbage (1854) et Kasiski (1863)
  - But : se ramener à la cryptanalyse de substitution simple
  - Exemple :

```
   ENVOYER LA CAVALERIE
   CLEFCLE FC LEFCLEFCL
→  GYZTAPD QC NEACWIWKP
```

1. Déterminer la taille de la clé (méthode de Kasiski) : 4

La cryptographie par substitution polyalphabétique (2)

- Procédé de Vigenère résistera jusqu'au milieu du XIX^e s.
  - Cryptanalyse de Babbage (1854) et Kasiski (1863)
  - But : se ramener à la cryptanalyse de substitution simple
  - Exemple :

```
   ENVOYER LA CAVALERIE
   CLEFCLE FC LEFCLEFCL
⟶ GYZTAPD QC NEACWIWKP
```

1. Déterminer la taille de la clé (méthode de Kasiski) : 4
2. On réarrange le cryptogramme par groupe de 4 lettres :

```
GYZT
APDQ
CNEA
CWIW
KP
```

La cryptographie par substitution polyalphabétique (2)

- Procédé de Vigenère résistera jusqu'au milieu du XIX^e s.
  - Cryptanalyse de Babbage (1854) et Kasiski (1863)
  - But : se ramener à la cryptanalyse de substitution simple
  - Exemple :

```
   ENVOYER LA CAVALERIE
   CLEFCLE FC LEFCLEFCL
⟶ GYZTAPD QC NEACWIWKP
```

1. Déterminer la taille de la clé (méthode de Kasiski) : 4
2. On réarrange le cryptogramme par groupe de 4 lettres :

```
GYZT
APDQ
CNEA
CWIW
KP
```

3. Pour chaque colonne, cryptanalyse de substitution simple

Calculer la longueur d'une clé de Vigenère

- Considérons par exemple le message codé suivant :

  CS AZZMEQM, CO XRWF, CS DZRM GFMJECV. X'IMOQJ JC LB NLFMK CC LBM WCCZBM KFIMSZJSZ CS URQIUOU. CS ZLPIE ECZ RMWWTV, SB KCCJ QMJ FCSOVJ GCI ZI ICCKS...

- Idée : une séquence se répète $\longrightarrow$ la distance entre 2 séquences est probablement un multiple de la taille de la clef

Séquence	Position	Distance	Décomposition
COX	11-140	129	3.43
FCS	16-99	83	83
ZRM	20-83	63	$3^2 7$
FMJ	24-162	138	2.3.23
CLB	37-46	9	3^2
KCC	44-92	48	$2^3 3$
WTV	87-133	46	2.23
CCJ	93-126	33	3.11
ICC	110-155	45	$3^2.5$
MJI	136-163	27	3^3
pgcd pour les triplets pertinents :			3

Cryptanalyse classique par analyse de fréquence en regroupant par paquet de 3

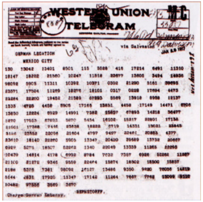
C	S	A
Z	Z	M
E	Q	M
C	O	X
⋮	⋮	⋮

LE SILENCE, LA PAIX, LE VIDE PRESQUE. J'AVAIS VU UN FURET OU UNE FOUINE TRAVERSER LE MACADAM. LE RUBAN QUI DEFILE, ET TOUS CES RUBANS SUR LA ROUTE...

- Méthode moderne (Friedman 1920) : calcul d'indices de coïncidences.

Enjeux de la cryptanalyse : le télégramme de Zimmermann

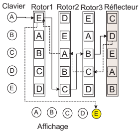
- 1917 : la guerre s'enlise
- Les Etats-Unis de Wilson sont restés neutre
- L'état-major allemand veut lancer la guerre sous-marine totale
  - Pb : peut déclencher entrée en guerre des USA
  - Idée de Zimmermann (ministre aff. étr.) : occuper les USA avec le Mexique et le Japon $\Rightarrow$ soutien financier alld à ces insurrections.
  - Z. envoit un télégramme chiffré à l'ambassade d'allemagne aux USA



- Télégramme déchiffré par le bureau 40 (Montgomery & al.)
- Permet l'entrée en guerre des USA contre l'allemagne.

Mécanisation de la cryptographie : Enigma

- Machine Enigma (Scherbius 1918)
  - Substitution polyalphabétique
    - 26 orientations pour 3 rotors : $26^3 = 17576$ alphabets
  - Réflécteur : cryptage/décryptage : même config
    - Connector/Reflector : substitution



- Brisé par l'équipe polonaise (Marian Rejewski) en 1933.
- Renforcé par les allemands pdt la 2^{eme} guerre (avec 5 rotors)
- Cassé par les "bombes" de Turing (Bletchley) (dico...)

Notion de sécurité inconditionnelle

- Cryptanalyses précédentes utilisent la répétition de la clé

Definition (Sécurité inconditionnelle)

la connaissance du message chiffré n'apporte aucune information sur le message clair.

- seule attaque possible : recherche exhaustive de clé secrète
- la clé secrète doit être au moins aussi longue que le texte clair

Notion de sécurité inconditionnelle

- Cryptanalyses précédentes utilisent la répétition de la clé

Definition (Sécurité inconditionnelle)

la connaissance du message chiffré n'apporte aucune information sur le message clair.

- seule attaque possible : recherche exhaustive de clé secrète
- la clé secrète doit être au moins aussi longue que le texte clair

Existe t il un sytème cryptographique inconditionnellement sûr ?

Système de Vernam (One time pad) (1917)

- Relation fondamentale :

$$\forall M, K / |M| = |K|, \ (M \oplus K) \oplus K = M$$

Fonctions de chiffrement/déchiffrement : $\begin{cases} E_K(M) = M \oplus K \\ D_K(C) = C \oplus K \end{cases}$

- Vigenère avec : Longueur mot-clef = longueur message !
  - Confusion totale : chiffrement de "aaaa...aaa" aléatoire
  - Diffusion totale : si le mot -clef n'est jamais réutilisé
- Pour un message M de n bits, clé K de n bits.

$$\begin{array}{rcl} M & = & 1000011 \\ K & = & 1101000 \\ \hline C = M \oplus K & = & 0101011 \end{array}$$

- **Si** K est totalement aléatoire et n'est utilisée une seule fois **alors** Oscar n'obtient aucune information sur M à partir de C.

Systèmes cryptographiques pratiquement sûr

- Vernam : seul système prouvé inconditionnellement sûr
  - MAIS problème du caractère aléatoire et du stockage de K
  - tous les autres systèmes sont théoriquement cassables

Definition (chiffrement pratiquement sûr)

un message chiffré ne permet de retrouver ni la clé secrète ni le message clair *en un temps humainement raisonnable*.

$\Longrightarrow$ permet d'utiliser des clés plus petites (56, 128 bits...)

Question : Pourquoi ne pas tester toutes les clés possibles ?

Systèmes cryptographiques pratiquement sûr

- Vernam : seul système prouvé inconditionnellement sûr
  - MAIS problème du caractère aléatoire et du stockage de K
  - tous les autres systèmes sont théoriquement cassables

Definition (chiffrement pratiquement sûr)

un message chiffré ne permet de retrouver ni la clé secrète ni le message clair *en un temps humainement raisonnable*.

$\Longrightarrow$ permet d'utiliser des clés plus petites (56, 128 bits...)

Question : Pourquoi ne pas tester toutes les clés possibles ?

Réponse : ce serait trop long a tester sur ordinateur !

Ex : portable 1Ghz $\longrightarrow 10^9$ op/s ; clé : 128 bits soit

$2^{128} \simeq 3,4 * 10^{38}$ possibilités $\Longrightarrow 3,4 * 10^{29}$ s

Systèmes cryptographiques pratiquement sûr

- Vernam : seul système prouvé inconditionnellement sûr
  - MAIS problème du caractère aléatoire et du stockage de K
  - tous les autres systèmes sont théoriquement cassables

Definition (chiffrement pratiquement sûr)

un message chiffré ne permet de retrouver ni la clé secrète ni le message clair *en un temps humainement raisonnable*.

$\Longrightarrow$ permet d'utiliser des clés plus petites (56, 128 bits...)

Question : Pourquoi ne pas tester toutes les clés possibles ?

Réponse : ce serait trop long a tester sur ordinateur !

Ex : portable 1Ghz $\longrightarrow 10^9$ op/s ; clé : 128 bits soit $2^{128} \simeq 3,4 * 10^{38}$ possibilités $\Longrightarrow 3,4 * 10^{29}$ s

- 10^{22} ordi pdt 1 an (il y a $\simeq 10^9$ PC ds le monde en 2004)
- Age de l'univers : 15 milliard * 365 * 24 * 3600 $\simeq 4,7 * 10^{17}$s

Cryptographie moderne

Principe 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
    - Contre-exemple : GSM et surtout CSS (Content Scrambling System - protection des DVD)
        - cf `http://www.lemuria.org/decss/`
2. Le déchiffrement sans la clef doit être impossible
    - en un temps raisonnable
3. Trouver la clef à partir du clair et du chiffré est impossible
    - en un temps raisonnable

Auguste Kerckhoffs, *La cryptographie militaire*, Journal des sciences militaires, vol. IX, pp. 5-83, jan. 1883, pp. 161-191, févr. 1883.

Théorie de l'Information

- Claude Shannon (1948)
- Source d'information $(\mathcal{S}, \mathcal{P})$ sans mémoire
  - $\mathcal{S} = \{s_1, \dots, s_n\}$, $\mathcal{P} = \{p_1, \dots, p_n\}$
  - p_i : probabilité d'occurence de s_i dans une emission
- Notion d'entropie
  - Quantité d'information de $s_i \in \mathcal{S}$: $I(p_i) = \log_2\left(\frac{1}{p_i}\right)$
  - Quantité d'information d'une source $(\mathcal{S}, \mathcal{P})$
    - $H(\mathcal{S}) = -\sum_{i=1}^{n} p_i \log_2(p_i) = \sum_{i=1}^{n} p_i \log_2\left(\frac{1}{p_i}\right)$
    - Nombre moyen de question à poser pour déterminer la valeur obtenue
  - Dé non pipé : $H(\mathcal{S}) = \sum_{i=1}^{6} \frac{1}{6} \log_2(6) \simeq 2.58$
  - Dé pipé $(p_1 = \frac{1}{2}\ ;\ p_i = \frac{1}{10} \forall i \in [2, 6])$
    $H(\mathcal{S}) = \frac{1}{2}\log_2(2) + 5 * \frac{1}{10}\log_2(10) \simeq 2,161$
- L'entropie est maximale lorsque toutes les proba sont égales !
  - Ex : Si l'apparition des lettres est exactement aléatoire, il est impossible d'appliquer l'attaque fréquentielle

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
  - Ex : quicksort : tri moyen en $\mathcal{O}(n \log n)$
- 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 expo) : classe NP
  - Problèmes indécidables

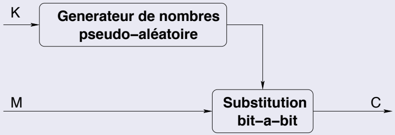
Complexité & Cryptographie

- Niveau de complexité d'une attaque
  - Comparer avec la recherche exhaustive
- Chiffrement idéal (moins que parfait !) - pratiquement sûr
  - L'implémentation est possible : complexité polynomiale au pire
  - 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
- Attention au *paradoxe des anniversaires* pour les complexités pratique !

Classes de chiffrements symétriques

Chiffrement symétrique par flot (stream cypher - statefull)

- Traitement à la volée ; chiffrement à la one-time pad :
  - $|M| = n$ et avec une petite clé K, générer $K' \ / \ |K'| = n$

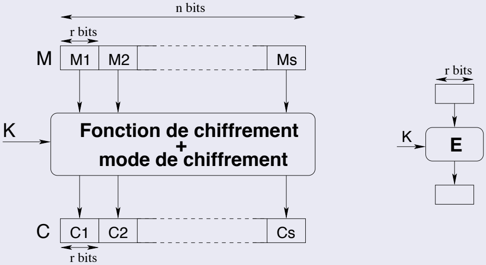


- Sécurité :
  - Substitution rapide ($\oplus$ typiquement)
  - Générateur pseudo aléatoire : "impossible" à prédire
  - Kerckhoffs : la sécurité repose sur le générateur de clé !
- Ex : LFSR, RC4 (Rivest), Py (Biham), E0[Bluetooth], A5/3[GSM]

Classes de chiffrements symétriques (2)

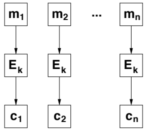
Chiffrement symétrique par bloc (bloc cypher - stateless)

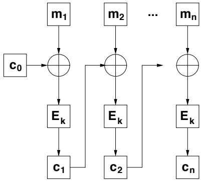
- $M = M_1 \bullet M_2 \bullet \ldots \bullet M_s$: s blocs de $r = \frac{n}{s}$ bits



- Sécurité :
  - Pour chaque bloc : $C_i = E_K(M_i)$ dépend de E
  - Pour chaque message : dépend aussi du mode de chiffrement !
- Ex : DES, AES, IDEA, BLOWFISH, RC6

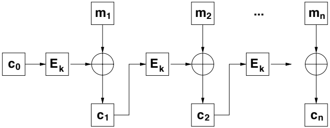
Les modes de chiffrement



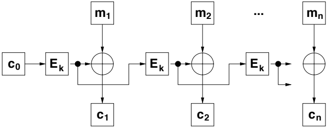


- Mode ECB (Electronic Code Book)
  - $C_i = E_K(M_i)$
  - $M_i = D_K(C_i)$
  - Un bloc est toujours chiffré identiquement
  - Aucune sécurité, pas d'utilisation
- Mode CBC (Cipher Bloc Chaining)
  - $C_i = E_k(M_i \oplus C_{i-1})$
  - $M_i = C_{i-1} \oplus D_k(C_i)$
  - Mode le plus utilisé

Les modes de chiffrement (2)



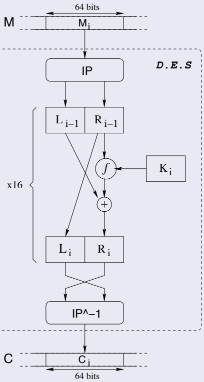
- Mode CFB (Cipher FeedBack)
  - $C_i = M_i \oplus E_K(C_{i-1})$
  - $M_i = C_i \oplus E_K(C_{i-1})$
  - Pas besoin de D_K !
  - Moins sûr, parfois plus rapide
  - Utilisé dans les réseaux



- Mode OFB (Output FeedBack)
  - $Z_i = E_K(Z_{i-1})$; $C_i = M_i \oplus Z_i$
  - $Z_i = E_K(Z_{i-1})$; $M_i = C_i \oplus Z_i$
  - Variante du mode précédent
  - Totalement symétrique
  - Moins de cablage
  - Utilisé dans les satellites

Les standard de chiffrement par bloc

D.E.S (Data Encryption Standard) - 1977



- Standard américain FIPS 46-2
- Chiffrement par blocs de 64 bits
- Clé de 64 bits dont 8 bit de parité :
  - 56 bits effectifs (plaidé par la NSA)
  - Diversification en 16 sous-clés de 48 bits
- Structure générale :
  - Permutation initiale IP
  - 16 rondes "de Feistel" :

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \end{cases}$$

  - Permutation finale IP^{-1}

Un mot sur les permutations de DES

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

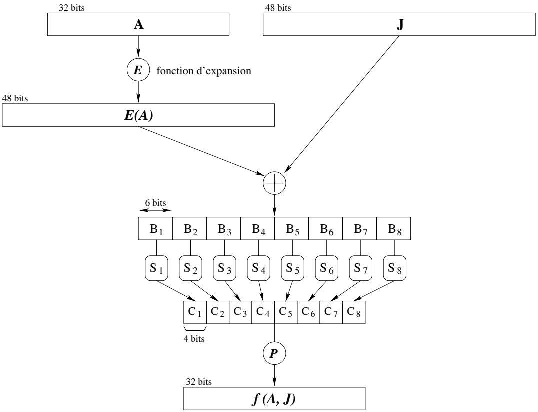
IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

- Signification dans le calcul de $y = IP(x)$:
  - le 58^e bit de x est le premier de y
  - le 50^e bit de x est le second de y
  - etc...

E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Détail de la fonction $f(A, J)$ de DES



Détail du calcul autour des S-Box de DES

- 8 "boîtes-S" $S_1, S_2, \ldots, S_8$
  - tableaux 4×16 entiers compris entre 0 et 15

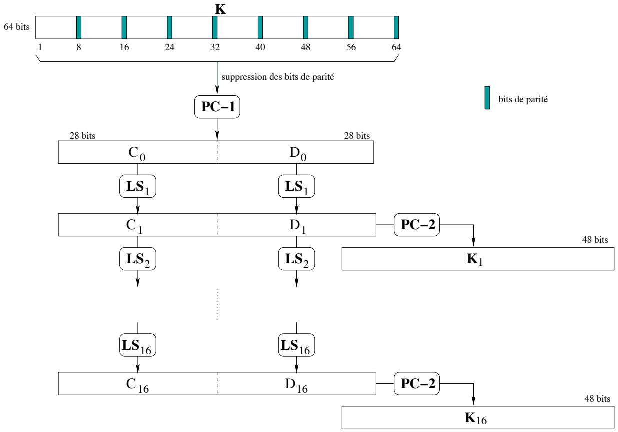
S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

- Soit une sous-chaîne de six bits $B_i = b_1.b_2.b_3.b_4.b_5.b_6$.
- Calcul de la chaîne de quatre bits $S_i(B_j)$:
  - $b_1.b_6$ = indice l de S_i à considérer $0 \leq l \leq 3$.
  - $b_2.b_3.b_4.b_5$ = colonne c de S_i à considérer $0 \leq c \leq 15$
  - A l'intersection de l et c : $C_i = S_i(B_i)$!
- On obtient ainsi $C = C_1.C_2 \ldots C_8$, chaîne de 32 bits
- On applique la permutation P : $P(C) = f(A, J)$.

Caractéristiques des boîtes S de DES

- Propriétés cryptanalytiques intéressantes
  - Non linéaire (substitution différente de César ou Vernam) : pas d'attque simple
  - Spécialement conçues pour contrer la cryptanalyse différentielle [Coppersmith 94]
  - Même de très petites modifications des S -box peuvent affaiblir considérablement le chiffre
- A la base de controverse autour de DES
  - cf secret entourant la génération de $\{S_i\}_{1\leq i\leq 8}$ et de P

Diversification des clés dans DES



Caractéristiques de DES

- Après 5 tours, chaque bit du chiffré dépend de chaque bit du message en clair et de chaque bit de la clef.
- Résultat du chiffrement statistiquement "plat"
- Quelques exemples d'utilsation :
  - Cartes de crédit : *UEPS* (Universal Electronic Payment System)
  - Protocole d'authentification sur réseaux : `Kerberos`
  - Messagerie éléctronique : *PEM* (Privacy-Enhanced Mail)
- Implémentation hardware aisée
  - Opération facilement implémentables : cf TP :-)
  - Puce spécifique bas de gamme[1] ($\simeq$ 60 euros) : $\simeq$ 190 Mo/s.
- Propriété de completion et clés faibles (pas une menace) :

$$DES_K(M) = C \Longrightarrow DES_{\bar{K}}(\bar{M}) = \bar{C}$$

[1] voir par exemple `http://hifn.com/products/7955-7956.html`

Cryptanalyse de DES

- Précalcul exhaustif
  - Stocker le résultat de DES sur un texte choisi $\forall K$
- Recherche exhaustive
  - Chiffrer un texte connu jusqu'à retrouver le chiffrement
  - permet de connaître la clef
- Assez peu de progrès au début - attaques sur 8/16 rondes (1975-1990)
- **Cryptanalyse différentielle** (16 rondes) [Biham-Shamir 1990]
  - Etude des différences de chiffrement entre des textes similaires
  - Permet de sélectionner des clefs probables
- **Cryptanalyse linéaire** (16 rondes) [Matsui 1993]
  - Utiliser des relations linéaires pour interpoler des bits de la clef

Complexité et coût des attaques sur DES

Méthode d'attaque	Texte connu	Texte choisi	Stockage	Calculs
Précalcul exhaustif		1	2^{56}	1 tableau
Recherche exhaustive	1			2^{55}
Crypta. différentielle	2^{47} puis 2^{36}		Textes	2^{47} puis 2^{36}
Crypta. linéaire	2^{55}	2^{47}	Textes	2^{47}

Cout des attaques sur DES en 1996

Attaquant	Budget	Outil	Clé 56 bits
Hacker	300 euro	Soft circuit	38 ans
PME	7500 euro	Circuit	18 mois
Gde Entreprise	225 Keuro	Circuit ASIC	19 j. 3 h
Multinationale	7,5 Meuro	ASIC	6 min
Gouvernement	225 Meuro	ASIC	12 s

Et aujourd'hui ?

- Problème du DES : clé devenu trop petite !
  - cassable en 8h avec 100 PCs ($2^{56} \simeq 7,2 * 10^{16}$) et $\frac{7,2*10^{16}}{10^9*3600*24*100} \simeq 8$ heures
- Solution 1 : double DES ?
  - $C = E_2(E_1(M))$ et $M = D_1(D_2(C))$
  - Pb : cassage effectif le rend seulement 56 fois plus difficile que DES et non 2^{56} fois
- Solution 2 : triple DES ?
  - 3 clefs : $C = E_1(E_2(E_3(M)))$ et $M = D_3(D_2(D_1(C)))$
  - 2 clefs : $C = E_1(D_2(E_1(M)))$ et $M = D_1(E_2(D_1(C)))$
    - retombe sur DES si $K_2 == K_1$
    - Clef moins longue et sécurité effective identique
    - Attaque similaire double DES $\Rightarrow$ clé effective 112 bits
    - CC : Triple DES double seulement la sécurité !

Principe général de la cryptanalyse statistique sur les chiffrements par bloc

1. Etude d'une version réduite en ronde
2. Etude la propagation de propriété non aléatoire à travers les rondes
3. Etre capable de détecter un chiffré d'une permutation aléatoire
4. Ajouter quelques roundes en début/fin tout en assurant la découverte de la clé
5. Compromis entre compléxité des données et temps d'analyse

Illustration sur des études de cas

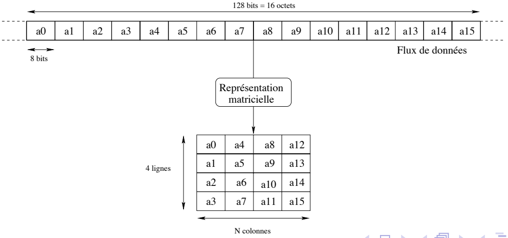
- FEAL-4 (Fast Data Encipherment Algorithm - Miyagushi 87)
  - 4 tours, blocs et clé de 64 bits
  - 88 : 100 à 10000 textes choisis
  - 90 : 20 textes choisis
  - 92 : 5 textes connus
- FEAL-8/FEAL-N/FEAL-NX (Miyagushi 90)
  - 90 : 20.000 textes choisis (Découverte cryptanalyse différentielle !)
  - 92 : 2^{15} textes connus (Découverte cryptanalyse linéaire !)
  - 96 : 12 textes choisis
- IDEA (8 rondes) (Lai, Massey 91)
  - Blocs de 64 bits, clé de 128 bits, 8 rondes
  - 90 : PES ; 91 : PES cassé $\Longrightarrow$ IPES=IDEA
  - 93 : Cassage sur 2 rondes
  - 97 : Cassage sur 3 rondes
  - 2003 : Cassage sur 5 rondes (2^{24} textes clairs)

Advanced Encryption Standard - AES (2000)

- 1996 : Evaluation DES $\Longrightarrow$ Il faut un remplacant !
- 1997 : Appel à candidature international remember Kerckhoffs :-)
  - 15 propositions ; 5 finalistes (1999) :
    1. Rijndael (Daemen,Rijmen BE) 10/12/14 rondes
       Bloc : 128 bits ; Clé : 128/192/156 bits
    2. Serpent (Anderson,Biham,Knudsen UK) 32 rondes
       Bloc : 128 bits ; Clé : 128/192/156 bits (en fait : $n = 8x \in [0, 2048]$)
    3. Twofish (Schneier&al US) 16 rondes
       Bloc : 128 bits ; Clé : 128/192/156 bits
    4. RC6 (Rivest US) 20 rondes
       Bloc : 128 bits ; Clé : 128/192/156 bits (en fait : $n = 8x \in [0, 2048]$)
    5. MARS (Coppersmith/IBM US) 16 rondes
       Bloc : 128 bits ; Clé : 128$\rightarrow$448 bits (128+32k bits)
- 2000 : Standard NIST : AES-Rijndael

Les conventions dans AES

- E/S : blocs de 128 bits ($N_b = 4$)
- Clé : 128, 192 ou 256 bits ($N_k = 4, 6$ ou 8)
- Nb rondes N_r : dépend de N_b et N_k ($N_r \in \{10, 12, 14\}$)
- 1 octet = élément du corps fini à 256 éléments $\mathbb{F}_{256}$
  - Rappel : p premier, $\mathbb{F}_{p^m}$ isomorphe à $\mathbb{F}_p[X]/g(X)$
  - $g(X)$: polynôme irréd. sur $\mathbb{F}_p[X]$ de degré m
  - Dans AES : $p = 2$, $m = 8$ et $g(X) = X^8 + X^4 + X^3 + X + 1$
- Interprétation matricielle d'un bloc (ex avec 16 octets) :



AES-Rijndael

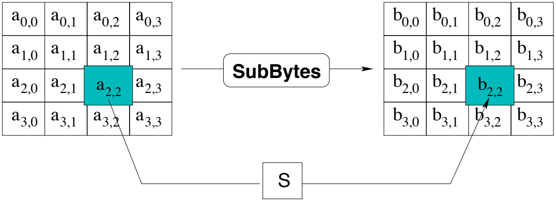
- Standart NIST 2000 ;
- Bloc : 128 bits ; Clé : 128/192/256 bits
- Structure générale :
  1. `AddRoundKey` Addition initiale de clé
  2. $N_r - 1$ rondes, chacune constituées de 4 étapes :
     - `SubBytes` : substitution non-linéaire via *S-Box*.
     - `ShiftRows` : transposition matricielle par décalage à gauche
     - `MixColumns` : produit matriciel sur colonne
     - `AddRoundKey` Addition avec les octets des sous-clé
  3. `FinalRound` : ronde finale (sans `MixColumns`)

AES : pseudo-code de la fonction de chiffrement

```
AES_Encrypt(State, K) {
       KeyExpansion(K, RoundKeys);
       /* Addition initiale */
       AddRoundKey(State, RoundKeys[0]);
       /* Les Nr-1 rondes */
       for (r=1; i<Nr; r++) {
           SubBytes(State);
           ShiftRows(State);
           MixColumns(State);
           AddRoundKey(State, RoundKeys[r]);
       }
       /* FinalRound */
       SubBytes(State);
       ShiftRows(State);
       AddRoundKey(State, RoundKeys[Nr]);
  }
```

Etape SubBytes

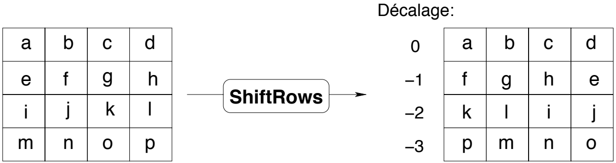
- Substitution de chaque élément de la matrice via une SBox
- SBox dérive de la fontion inverse $t : a \longrightarrow a^{-1}$ sur $\mathbb{F}_{256}$.
  - fonction bien connue pour sa non-linéarité
  - on combine avec une transformation affine inversible f :
    - $\texttt{SBox}[a] = f\left(t(a)\right) \forall a \in \mathbb{F}_{256}$
    - $\texttt{SBox}^{-1}[a] = t^{-1}\left(f^{-1}(a)\right) = t\left(f^{-1}(a)\right) \forall a \in \mathbb{F}_{256}$



Etape Shiftrows

- Opération sur les lignes de matrice
  - La ligne i est décalé de C_i éléments à gauche
  - Le nombre de décalage dépend de N_b (Rijndael) :

N_b	C_0	C_1	C_2	C_3
4	**0**	**1**	**2**	**3**
5	0	1	2	3
6	0	1	2	3
7	0	1	2	4
8	0	1	3	4

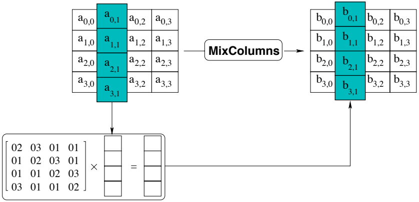


  - Opération inverse : la ligne i est décalée à droite de C_i éléments.

Etape MixColumns

- Operation sur les colonnes de la matrice
  - Considéré un polynôme $a(x)$ de degré 3 dans $\mathbb{F}_{256}[X]$
  - Réalise l'opération : $(03x^3 + x^2 + x + 02) \times a(x) \mod (x^4 + 1)$
  - Matriciellement :

$$b(x) = c(x) \times a(x) \mod (x^4+1) \Longleftrightarrow \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$



- Bonne propriétés de diffusion cryptographique

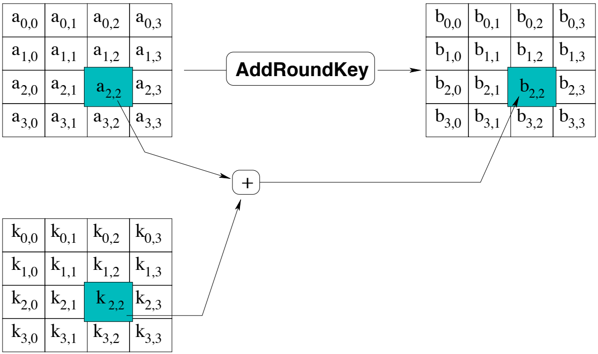
Etape $\texttt{MixColumns}^{-1}$

- idem mais en utilisant la multiplication par $d(x) = c^{-1}(x)$
  - $(03x^3 + x^2 + x + 02) \times d(x) \equiv 01 \mod (x^4 + 1)$
  - $d(x) = 0\mathrm{B}x^3 + 0\mathrm{D}x^2 + 09x + 0\mathrm{E}$
- Matriciellement :

$$b(x) = d(x) \times a(x) \mod (x^4 + 1) \Longleftrightarrow \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0\mathrm{E} & 0\mathrm{B} & 0\mathrm{D} & 09 \\ 09 & 0\mathrm{E} & 0\mathrm{B} & 0\mathrm{D} \\ 0\mathrm{D} & 09 & 0\mathrm{E} & 0\mathrm{B} \\ 0\mathrm{B} & 0\mathrm{D} & 09 & 0\mathrm{E} \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Etape AddRoundKey

- Addition matricielle dans $\mathbb{F}_{256}$ avec une sous-clé



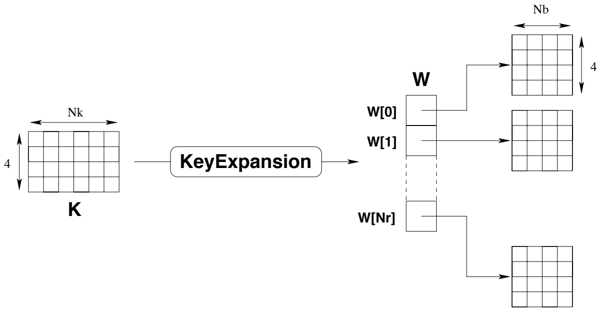
$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

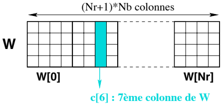
La diversification de la clef `KeyExpansion` dans AES

- Clé de chiffrement : $4N_k$ octets
- Extension en une clé étendue de $4N_b(N_r + 1)$ octets
  - on dispose ainsi de $N_r + 1$ clés de rondes de $4Nb$ octets



La diversification de la clef `KeyExpansion` dans AES (2)

- Matrice de clé K ou W = succession de colonnes de 4 octets



  - `c[i]` (resp. `k[i]`) : $(i+1)^{\text{ème}}$ colonne de `W` (resp. `K`).
- Algorithme opère sur les éléments `c[i]` et `k[i]`
- Utilise les éléments suivants :
  - `SubWord` : applique `Sbox` aux 4 octets du mot en entrée
  - `RotWord` :
    - Input : mot de 4 octets $a = [a_0, a_1, a_2, a_3]$
    - effectue une permutation circulaire et renvoit $[a_1, a_2, a_3, a_0]$
  - Tableau de constantes de rondes `Rcon[i]`
    - indépendant de N_k
    - défini récursivement par :

$$\texttt{Rcon}[i] = [x^{i-1}, 00, 00, 00], \forall i \geq 1$$

La diversification de la clef KeyExpansion dans AES (3)

```
KeyExpansion(K, W) {
       /* Recopie directe des Nk premiere colonnes */
       for (i=0; i<Nk; i++) c[i] = k[i];
       for (i=Nk; i<Nb*(Nr+1); i++) {
           tmp = c[i-1];
           if (i mod Nk == 0)
               tmp = SubWord(RotWord(tmp)) + Rcon[i/Nk];
           else if ((Nk > 6) && (i mod Nk == 4)) // Cas Nk > 6
               tmp = SubWord(tmp);
           c[i] = c[i-Nk] + tmp;
       }
   }
```

AES : pseudo-code de la fonction de déchiffrement

- Utilise `SubBytes`$^{-1}$, `ShiftRows`$^{-1}$, `MixColumns`$^{-1}$, et `AddRoundKey`
- Traitement des clés inchangé

```
AES_Decrypt(State, K) {
       KeyExpansion(K, RoundKeys);
       AddRoundKey(State, RoundKeys[Nr]); /* Addition initiale */
       /* Les Nr-1 rondes */
       for (r=Nr-1; i>0; r--) {
           InvShiftRows(State);
           InvSubBytes(State);
           AddRoundKey(State, RoundKeys[r]);
           InvMixColumns(State);
       }
       /* FinalRound */
       InvShiftRows(Out);
       InvSubBytes(Out);
       AddRoundKey(Out,RoundKeys[0]);
   }
```

- Ici, séquence des transformations $\neq$ celle du chiffrement
- Il existe une version qui respecte la séquence de transformations du chiffrement

Sécurité de l'AES

- Propriétés cryptanalytiques
  - `SBox` : 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 "simple" efficace
  - cf TP :-)
  - FPGA : jusqu'à 21.54 Go/s pour le chiffrement
- Cryptanalyse :
  - Aucune attaque significative révélée
  - MAIS seulement 5 ans de recherche
  - (to be continued)

Quelques 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

Projets de standardisation/Recommendations

- NIST (National Institute of Standards and Technology) (US00)

  ⟹ AES-Rijndael (Bloc : 128 bits ; Clé : 128/192/256 bits)
  25 cycles/octet sur un PIII/Linux.

- KICS (Korean Information and Communication Standards) (Corée01)

  ⟹ SEED (Bloc : 128 bits ; Clé : 128 bits)
  45 cycles/octet sur un PIII.

  ⟹ ARIA (proposition) (Bloc : 128 bits ; Clé : 128 bits)
  37 cycles/octet sur un PIII.

Projets de standardisation/Recommendations (2)

- NESSIE (New European Schemes for Signatures, Integrity and Encryption)(EU03)
  - Chiffrement symétriques par bloc :
    - $\Longrightarrow$ MISTY1 (Bloc : 64 bits ; Clé : 128 bits)
      47 cycles/octet sur un PIII/Linux.
    - $\Longrightarrow$ AES-Rijndael
    - $\Longrightarrow$ Camellia (Bloc : 128 bits ; Clé : 128/192/256 bits)
      35 cycles/octet sur un PIII/Linux.
    - $\Longrightarrow$ SHACAL-2 (Bloc : 256 bits ; Clé : 512 bits)
      44 cycles/octet sur un PIII/Linux.
  - Fonctions de hachage à sens unique
    - $\Longrightarrow$ Whirlpool
    - $\Longrightarrow$ SHA-256, SHA-384 et SHA-512 (clairvoyant ! cf SHA-1)
  - Chiffrement asymétriques : PSEC-KEM, RSA-KEM, ACE-KEM
  - Signature électronique : RSA-PSS, ECDSA, SFLASH

Projets de standardisation/Recommendations (3)

- CRYPTREC (Cryptography Research and Evaluation Committee) (Jap03)
  - Chiffrement symétrique (blocs : 64 bits, clé $\geq$ 128 bits) :
    $\Longrightarrow$ CipherUnicorn-E, Hierocrypt-L1, MISTY1, Triple DES
  - Chiffrement symétrique (blocs : 128 bits, clé $\geq$ 128 bits) :
    $\Longrightarrow$ AES, Camelia, CipherUnicorn-A, Hierocrypt-3, SC2000
  - Chiffrement symetrique par flot :
    $\Longrightarrow$ Mugi, Multi-S01, RC4
  - Fonctions de hachage à sens unique
    $\Longrightarrow$ RIPEMD-160, SHA-1, SHA-256/384/512

Cryptographie à clé publique

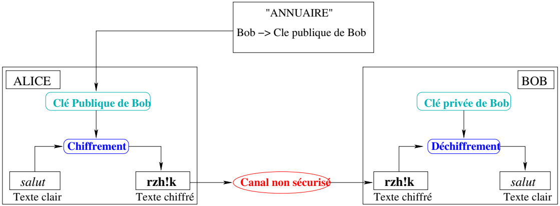
Motivations

- Systèmes cryptographiques à clé secrètes
  - pratiquement sûrs
  - efficaces en termes de temps de calcul.
- Mais nouvelles interrogations :
  - Avant d'utiliser un système de chiffrement à clé secrète, comment convenir d'une clé ?
  - Comment établir une communication sécurisée entre deux entités sans échange préalable de clef ?

$\Longrightarrow$ Solution apportée par Diffie et Hellman (1976)

  - systèmes cryptographiques à clé publique

Principe



Equation fondamentale :

$$\begin{cases} E_{K_e}(M) = C \\ D_{K_d}(C) = M \end{cases}$$

ici : $K_e \neq K_d$,

- K_e publique (connue de tous)
- K_d secrète (connue seulement de Bob)

- **Analogie : Boite aux lettres**
  - toute personne peut envoyer du courrier à Bob ;
  - seul Bob peut lire le courrier déposé dans sa boîte aux lettres.

Quelques pré-requis mathématiques

Theorem (Euclide)

Soit a, b $\in \mathbb{N}$ / $a \leq b$. *Soit r le reste de la division euclidienne de a par b. Alors pgcd(a,b)= pgcd(b,r).*

- Algorithme :
  - Tq $b \neq 0$:$(a, b) \longrightarrow (b, a \mod b)$
  - si $b = 0$ renvoyer a
- Exemple : $pgcd(42, 30) = 6$

$$
\begin{aligned}
(42, 30) &\longrightarrow (30, 12) \\
(30, 12) &\longrightarrow (12, 6) \\
(12, 6) &\longrightarrow (6, 0)
\end{aligned}
$$

Quelques pré-requis mathématiques (2)

Theorem (Bezout)

Soient $a,\ b \in \mathbb{Z}$ et $d = pgcd(a, b)$. Alors $\exists (u, v) \in \mathbb{Z}^2$ tels que

$$au + bv = d$$

Les entiers u et v sont appelés coefficients de Bezout.

- Calcul pratique : Algorithme d'Euclide Etendu

$$\begin{array}{lccccccc}
(E_0): & 1 \times a & + & 0 \times b & = & a \\
(E_1): & 0 \times a & + & 1 \times b & = & b \\
(E_{i+1}) = (E_{i-1}) - q_i(E_i) & u_i \times a & + & v_i \times b & = & r_i
\end{array}$$

Application sur le calcul d'inverse modulaire

Pb : calcul de $17^{-1} \mod 50$

- Calcul des coefficients de Bezout pour $a = 50$ et $b = 17$

(E_0) :	1×50	$+$	0×17	$=$	50	
(E_1) :	0×50	$+$	1×17	$=$	17	$q_1 = 50/17 = 2$ $r_1 = 50\%17 = 16$
$(E_2) = (E_0) - 2 \times (E_1)$	1×50	$+$	$(-2) \times 17$	$=$	16	$q_2 = 17/16 = 1$ $r_2 = 17\%16 = 1$
$(E_3) = (E_1) - 1 \times (E_2)$	$(-1) \times 50$	$+$	3×17	$=$	1	$q_3 = 16/1 = 16$ $r_3 = 16\%1 = 0$

- Bilan : $(-1) \times 50 + 3 \times 17 = 1 \Longrightarrow 17^{-1} = 3 \mod 50$

Exponentiation rapide modulaire : calcul de $a^e \mod n$

- Basé sur la remarque suivante :
  - si e est pair, $a^e = \left(a^{\frac{e}{2}}\right)^2$
  - si e est impair $a^e = \left(a^{\frac{e}{2}}\right)^2 \cdot a$

Algorithme d'exponentiation rapide modulaire

1. Décomposer e en binaire : $e = \sum_{i=0}^{k} e_i 2^i$
2. Calcul de $\{a^{2^i} \mod n\}_{0 \leq i \leq k}$
   - Utiliser la relation : $a^{2^{i+1}} = \left(a^{2^i}\right)^2 \mod n$
3. En déduire $a^e = \prod_{i=0}^{k} \left(a^{2^i}\right)^{e_i}$

Exponentiation rapide modulaire : exemple

Calcul de $51447^{21} \mod 17 \qquad (E)$

Exponentiation rapide modulaire : exemple

Calcul de $51447^{21} \mod 17 \qquad (E)$

- $51447 = 3026 \times 17 + 5$ donc $(E) \iff 5^{21} \mod 17$

Exponentiation rapide modulaire : exemple

Calcul de $51447^{21} \mod 17 \qquad (E)$

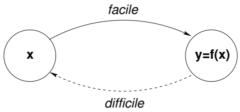
- $51447 = 3026 \times 17 + 5$ donc $(E) \Longleftrightarrow 5^{21} \mod 17$

1. Décomposition de 21 en binaire : $21 = 2^4 + 2^2 + 2^0$
2. Calcul de $\{5^{2^i} \mod 17\}_{0 \leq i \leq 4}$
   - $i = 0 : 5^{2^0} \equiv 5 \mod 17$
   - $i = 1 : 5^{2^1} = 5^2 = 25 \equiv 8 \mod 17$
   - $i = 2 : 5^{2^2} = 8^2 = 64 \equiv 13 = -4 \mod 17$
   - $i = 3 : 5^{2^3} = (-4)^2 \equiv 16 = -1 \mod 17$
   - $i = 4 : 5^{2^4} = (-1)^2 \equiv 1 \mod 17$
3. On en déduit :

$$\begin{aligned} 5^{21} &= 5^{2^4} \times 5^{2^2} \times 5^{2^0} \\ &= 1 \times (-4) \times 5 \\ &= -20 \equiv 14 \mod 17 \end{aligned}$$

Base de la cryptographie à clé publique : fonction a sens unique avec trappe

- Cryptographie à clé publique se base sur des problèmes mathématiques réputés difficiles.
- Fonction à sens unique :



Ex : factorisation d'entiers.

- Fonction à sens unique avec trappe.
  - la connaissance de la trappe (clé) facilite le calcul inverse !

Le cryptosystème RSA

Génération des clefs

- Bob choisit au hasard deux nombres premiers p et q.
  - Bob calcule $n = p.q$
  - Indicatrice d'Euler : $\varphi(n) = (p-1)(q-1)$
- Bob choisit au hasard un entier e (impair) tel que
  $\begin{cases} 1 < e < \varphi(n) \\ pgcd(e, \varphi(n)) = 1 \end{cases}$
- Bob calcule alors l'entier $1 < d < \varphi(n)$ tel que

$$ed \equiv 1 \bmod \varphi(n).$$

- **Clef publique** : (n, e) (e : exposant RSA ; n : module RSA)
- **Clef secrète** : d.

Le cryptosystème RSA (2)

Chiffrement RSA

- Alice récupère la clef publique (n, e) de Bob
- Pour chiffrer le message M entier tel que $0 \leq M < n$:

$$C = M^e \ mod \ n$$

- Alice envoit le message chiffré C à Bob.

Le cryptosystème RSA (3)

Déchiffrement RSA

- Pour déchiffrer le message C reçu d'Alice, Bob calcule

$$C^d = M \ mod \ n$$

En effet, $\exists k \in \mathbb{Z}$ tel que :

$$\begin{aligned} C^d &\equiv M^{e.d} \mod n \\ &\equiv M^{1+k.\varphi(n)} \mod n \\ &\equiv M.\left(M^{\varphi(n)}\right)^k \equiv M \mod n = M \end{aligned}$$

Le cryptosystème RSA : Exemple

Prenons $p = 47$ et $q = 59$.

- On calcule $n = p.q = 47.59 = 2773$
- On choisit e, premier par rapport à $\varphi(n)$. Ex : $e = 17$.
- On calcule alors, par l'algorithme d'Euclide étendu[2], d tel que $d.e \equiv 1 \mod (p-1)(q-1)$, soit $d = 157$.

Clef publique : $(e, n) = (17, 2773)$
Clef privé : $d = 157$.

- Chiffrement du message $M = 01000010 = 66$:

$$C \equiv M^e \mod n \equiv 66^{17} \mod 2773 = 872$$

- Déchiffrement de C :

$$C^d \mod n \equiv 872^{157} \mod 2773 \equiv 66$$

[2] sous Maple : **igcdex**

Sécurité du cryptosystème RSA

- Le vrai but de l'attaquant : découvrir le texte en clair !
- Calculer d à partir de $(n, e) \iff$ factoriser n.
  - $\Longleftarrow$: trivial (cf génération des clefs)
  - $\Longrightarrow$: Soit $s = \max\{t \in \mathbb{N} \ : \ 2^t | ed - 1\}$. On pose $k = \frac{ed-1}{2^s}$.
    Alors, soit $a \in \mathbb{Z}$ est premier avec n.
    - l'ordre de a^k dans $\mathbb{Z}_n \in \{2^i \ ; \ 0 \leq i \leq s\}$ ($a^{\varphi(n)} \equiv 1 \mod n$)
    - si l'ordre de $a^k \mod p \neq$ l'ordre de $a^k \mod q$, alors

    $$\exists t \in [0, s[\ / \ 1 < \mathrm{pgcd}(a^{2^t k} - 1, n) < n$$

    On a ainsi trouvé un facteur non trivial de n.
- Algo probabiliste.
- (Coron2004) : algo déterministe presque général$(e, d < \varphi(n))$
  - Toujours d'actualité : Casser RSA aussi dur que factoriser n ?

Sécurité du cryptosystème RSA

- Limites actuelles de factorisation : $\simeq$ 200 chiffres
- Record actuel[3] : RSA200 (200 chiffres décimaux)
  - Bahr, Boehm, Franke and Kleinjung - 9 mai 2005.
- Si la clef secrète d est petite (de l'ordre de $n^{1/4}$) :
  - attaque utilisant l'algorithme des fractions continues (algorithme LLL)
  - permet de calculer d à partir de n et e.

[3] http://www.loria.fr/~zimmerma/records/factor.html

DLP & Diffie-Hellman

- Autre problème difficile : **D**iscret **L**ogarithme **P**roblem

Definition (Logarithme discret)

Soit $G = <g> = \{g^i\}_{0 \leq i < n}$ un groupe monogène fini d'ordre n.
Soit $h \in G$. Alors le *logarithme discret de h en base g*, noté $\log_g h$, est l'unique entier x tel que $h = g^x$ $(0 \leq x < n)$.

DLP consiste alors à résoudre le problème suivant :

$$\text{Etant donné } G, g, h, \text{ trouver } x = \log_g h.$$

Exemple : $p = 97$ et $G = \mathbb{Z}/_{97}\mathbb{Z}^* = \{1, 2, \ldots, 96\} = \{5^i\}_{0 \leq i < 96}$
$5^{32} \equiv 35 \mod 97 \Longrightarrow \log_5 35 = 32$ dans $\mathbb{Z}/_{97}\mathbb{Z}^*$.

Protocole d'échange de clefs de Diffie-Hellman

Alice et Bob veulent partager une clef secrète K.
On suppose que les données G, $n = |G|$ et g sont publiques.

- Alice choisit un entier $1 \leq a \leq n-1$ au hasard.
- Alice calcule $A = g^a$ et l'envoie à Bob.
- Bob choisit un entier $1 \leq b \leq n-1$ au hasard.
- Bob calcule $B = g^b$ et l'envoie à Alice.
- Alice est en mesure de calculer B^a et Bob de calculer A^b.

La clef commune est donc

$$K = g^{ab} = A^b = B^a.$$

Protocole d'échange de clé de Diffie-Hellman

Alice		Bob
génère a $A = g^a \mod p$		génère b $B = g^b \mod p$
	$\xrightarrow{A}$	
	$\xleftarrow{B}$	
(dispose de $[a, A, B, p]$) Clé secrète : $K = B^a \mod p$		(dispose de $[b, A, B, p]$) Clé secrète : $K = A^b \mod p$

Sécurité de DH

- Problème de DH :
  - connaissant G, g, $A = g^a$ et $B = g^b$, calculer $K = g^{ab}$.
- A l'heure actuelle, résoudre DLP est la seule méthode générale connue pour résoudre DH.
  - MAIS : pas de preuve que résoudre DLP $\Longleftrightarrow$ résoudre DH.
- Choix du groupe G : $G = \mathbb{F}_p^*$, $G = E(\mathbb{F}_p)$, etc.
  - Attention au bon choix des paramètres.

Le cryptosystème de El Gamal

Données publiques pré-requise :

- $(G, .) =< g >$ un groupe cyclique d'ordre n

Génération des clefs

- Bob choisit $a \in [1, n-1]$ et calcule $A = g^a$ dans G.
- Clef publique : (G, g, n, A).
- Clef secrète : a.

Le cryptosystème de El Gamal (2)

Chiffrement

Alice souhaite envoyer le message $M \in G$ à Bob

- Alice récupère la clef publique (G, g, n, A) de Bob.
- Alice choisit au hasard $k \in [1, n-1]$
- Le message chiffré qu'Alice envoit à Bob est $C = (y_1, y_2)$ avec

$$\begin{cases} y_1 = g^k \\ y_2 = M.A^k \end{cases}$$

Le cryptosystème de El Gamal (3)

Déchiffrement

- Bob recoit le message chiffré $C = (y_1, y_2)$
- Il lui suffit alors de calculer

$$M = y_2.(y_1^a)^{-1} = y_2.y_1^{n-a}$$

En effet :

$$\begin{aligned} y_2.y_1^{n-a} &= M.A^k.\left(g^k\right)^{n-a} \\ &= M.g^{a.k}.g^{k.n}.g^{-ka} \\ &= M.g^{a.k}.(g^n)^k.g^{-ka} \\ &= M.g^{a.k}.g^{-ka} = M \end{aligned}$$

Sécurité du cryptosystème de El Gamal

- Résoudre DLP dans $G \Longrightarrow$ Casser El Gamal dans G
  - l'attaquant peut alors calculer a à partir de A (public).
- La réciproque n'est pas encore prouvée !

Cas particulier de $G = \mathbb{F}_p^*$:

- utiliser un nombre premier p de 1024 bits choisis uniformément
- permet de résister aux méthodes actuelles de résolution de DLP sur $\mathbb{F}_p^*$

Fonctions de hachage et signatures électroniques

Notion de fonction de hachage

Definition (Fonction de Hachage)

Une fonction de hachage H est une application facilement calculable qui transforme une chaîne binaire de taille quelconque t en une chaine binaire de taille fixe n, appelée *empreinte de hachage*.

- En général, $t > n$: H est surjective
- On parle de *collision* entre x et x' lorsque

$$\begin{cases} x \neq x' \\ H(x) = H(x') \end{cases}$$

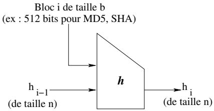
- Si y est tel que $y = H(x)$, alors x est appelé *préimage* de y

Propriétés des fonction de hachage

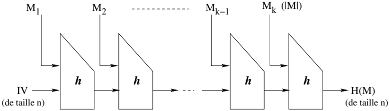
- Propriétés de base : compression et facilité de calcul.
- Propriétés additionnelles :
  - Résistance à la préimage
    - étant donné y, il est difficile de trouver x tel que $y = H(x)$
  - Rrésistance à la seconde préimage
    - étant donné x, il est difficile de trouver $x' \neq x$ tel que $H(x) = H(x')$
  - Résistance à la collision
    - il est difficile de trouver x et x' tels que $H(x) = H(x')$.
- Fonction de Hachage à Sens Unique
  - résistance à la préimage et à la seconde préimage
- Fonction de Hachage résistante aux collisions
  - résistance à la seconde préimage et à la collision

Construction d'une fonction de hachage

- Définir une fonction de compression h



- Pour calculer l'empreinte d'un message M :
  - Application d'un *padding* à M pour que $|M| = k.b$
  - Découpage du message M en blocs de tailles b
    $$M = M_1 M_2 \dots M_{k-1} M_k \text{ avec } |M_i| = b \quad \forall i \in [1, k]$$
  - Itération de la fonction h (IV : Initial Value) :



- Exemples connus : MD5, SHA-1, SHA-2, Whirlpool...

Idée générale des signatures électroniques

But des signatures manuscrites :

- prouver l'identité de leur auteur **et/ou**
- l'accord du signataire avec le contenu du document

La signature éléctronique dépend du signataire **et** du document !

Objectifs d'une signature éléctronique

- Une signature est authentique.
- Une signature ne peut être falsifiée (imitée).
- Une signature n'est pas réutilisable sur un autre document.
- Un document signé est inaltérable.
- Une signature ne peut pas être reniée.

Idée générale des signatures électroniques (2)

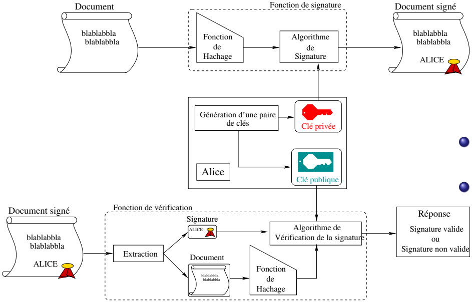
- Réalisation pratique :
  - Cryptosystèmes à clef secrète (et arbitre)
  - Cryptosystèmes à clef publique + fonction de hachage
- On préfère signer le hachage d'un document
  - Taille fixe suffisamment petite pour être utilisée efficacement par un cryptosystème à clé publique

Idée générale des signatures électroniques (3)

- Protocole de signature électronique sûr :
  - Impossible de falsifier la signature $s(M)$ d'un document M
    - sans connaitre la clef secrète K (resp. K_d)
    - même en disposant de signatures d'autres documents.
    - Attention : impossibilité *pratique*
- Il existe d'autres conditions nécessaires de sécurité !
  - relève davantage des architectures de sécurité des cryptosystèmes à clef publiques (PKI) ou du secret entourant la clé secrète.

Idée générale des signatures électroniques (4)

- Signature utilisant un cryptosystème à clef publique :



- Alice signe M en utilisant :
  - $h_M = H(M)$ le hachage de M
  - sa clé secrète K_d.
  - la fonction de déchiffrement D.
  - Résultat : $s(M) = D_{K_d}(h_M)$
- Document signé : $[M, s(M)]$
- Vérification de $[M, s(M)]$:
  - utilise la clé publique K_e d'Alice et la fonction de chiffrement E
  - $E_{K_e}(s(M)) = h_M =? \ H(M)$
  - Seule Alice a pu générer $s(M)$

Signature RSA

Génération des paramètres

Identique à la génération des clefs de RSA !

- Alice choisit au hasard deux nombres premiers p et q.
  - Alice calcule $n = p.q$
  - Indicatrice d'Euler : $\varphi(n) = (p-1)(q-1)$
- Alice choisit au hasard un entier e (impair) tel que $1 < e < \varphi(n)$ et $\text{pgcd}(e, \varphi(n)) = 1$
- Alice calcule alors l'entier d tel que $e.d \equiv 1 \mod \varphi(n)$.

Clef publique : (n, e)
Clef secrète : d

On suppose disposer d'une fonction de hachage à sens unique H connue publiquement.

Signature RSA (2)

Génération d'une signature RSA

Alice souhaite signer un document M

- Alice calcule $h_M = H(M)$ (on suppose $0 \leq h_M < n$)
- Signature de M : $s(M) = (h_M)^d \mod n$
- Le document signé est alors $[M, s(M)]$.

Signature RSA (3)

Vérification d'une signature RSA

- Bob reçoit un document signé $[\tilde{M}, s(M)]$ d'Alice.
  - Ce document est potentiellement altéré/illégitime
- Il récupère la clé publique d'Alice (n, e)
- Il calcule $h_{\tilde{M}} = H(\tilde{M})$
- Il vérifie l'identité : $s(M)^e \equiv h_{\tilde{M}} \mod n$

En effet : $s(M)^e \equiv (h_M)^{e.d} \mod n \equiv h_M \mod n = h_M$ et si le document est authentique : $h_M = h_{\tilde{M}}$.

- La sécurité est donc celle du cryptosystème RSA.
- Présentation simpliste et en l'état sujette à des attaques

Signature El Gamal

Génération des paramètres

- Alice choisit :
  - un nombre premier p
  - g une racine primitive modulo p.
  - un entier $a \in \{1, \dots, p-2\}$ au hasard
- Elle calcule alors $A = g^a \mod p$.

Clef publique : (p, g, A).
Clef secrète : a.

On suppose disposer d'une fonction de hachage à sens unique H connue publiquement.

Signature El Gamal (2)

Génération d'une signature El Gamal

Alice souhaite signer un document M

- Alice calcule $h_M = H(M)$ (on suppose $0 \leq h_M < p$)
- Elle choisit au hasard un entier $k \in [1, p-2]$ tel que $\mathrm{pgcd}(k, p-1) = 1$ ($\Longrightarrow k^{-1} \in \mathbb{Z}_{p-1}$ existe).
- Signature de M : $s(M) = (r, s)$ avec

$$\begin{cases} r = g^k \mod p \\ s = k^{-1}(h_M - a.r) \mod (p-1) \end{cases}$$

- Le document signé est alors $[M, s(M)]$.

Signature El Gamal (3)

Vérification d'une signature El Gamal

- Bob reçoit un document signé $[\tilde{M}, s(M)]$ d'Alice.
  - Rappel : $s(M) = (r, s)$
  - Ce document est potentiellement altéré/illégitime
- Il récupère la clé publique d'Alice (p, g, A)
- Il calcule $h_{\tilde{M}} = H(\tilde{M})$
- Il vérifie l'identité : $A^r r^s \equiv g^{h_{\tilde{M}}} \mod p$

En effet,

$$\begin{aligned} A^r r^s &\equiv g^{a.r}.g^{kk^{-1}(h_M - a.r)} \mod p \\ &\equiv g^{h_M} \mod p \end{aligned}$$

Si le document est authentique : $h_M = h_{\tilde{M}} \Rightarrow g^{h_M} \equiv g^{h_{\tilde{M}}} \mod p$

Sécurité des signatures El Gamal

- Sécurité intimement liée à DLP dans $\mathbb{F}_p^*$
    - Résolution de DLP dans $\mathbb{F}_p^*$
      $\Longrightarrow$ possibilité de calculer a à partir de A
      $\Longrightarrow$ possibilité d'impersonaliser Alice
- Attention au choix des paramètres.

Le standard DSA

Génération des paramètres

- Alice génère un nb premier **q** de 160 bits ($2^{159} \leq q < 2^{160}$)
- Elle génère un nb premier **p** de 512 à 1024 bits vérifiant :

$$\begin{cases} \exists t \in [0,8] \ / \ 2^{511+64t} < p < 2^{512+64t} \\ q | (p-1) \end{cases} \tag{2}$$

- Soit $\tilde{g}$ une racine primitive modulo p
- Un générateur du sous-groupe de $\mathbb{F}_p^*$ d'ordre q est alors

$$\mathbf{g} = \tilde{g}^{\frac{p-1}{q}} \mod p$$

(2) assure que $\mathbb{F}_p^*$ possède un sous-groupe d'ordre q

Le standard DSA (2)

Génération des paramètres (suite)

Une fois choisis (p, q, g) :

- Alice choisit $a \in \{1, \ldots, q-1\}$
- Elle calcule $A = g^a \mod p$

Clef publique : (p, q, g, A).
Clef secrète : a

Le problème du logarithme discret sous-jacent se passe dans le groupe d'ordre q.

Le standard DSA (3)

Génération d'une signature DSA

Alice souhaite signer un document M :

- Alice calcule $h_M = H(M)$ (on suppose $1 \leq h_M < q - 1$)
- Elle choisit un entier $k \in \{1, \ldots, q-1\}$
- Signature de M : $s(M) = (r, s)$ avec

$$\begin{cases} r = (g^k \mod p) \mod q \\ s = k^{-1}(h_M + a.r) \mod q \end{cases}$$

- Le document signé est alors $[M, s(M)]$.

Le standard DSA (4)

Vérification d'une signature DSA

- Bob reçoit un document signé $[\tilde{M}, s(M) = (r, s)]$ d'Alice.
- Il récupère la clé publique d'Alice (p, q, g, A)
- Il vérifie que les formats sont respectés : $1 \leq r, s \leq q - 1$
- Il calcule $\tilde{h_M} = H(\tilde{M})$
- Il vérifie l'identité :

$$r \equiv \left[\left(g^{s^{-1}\tilde{h_M}} \mod q \right) . \left(A^{rs^{-1}} \mod q \right) \mod p \right] \mod q.$$

Ce qu'il nous reste à voir...

- Développement sur les cryptosystèmes à clé publique
  - Primalité
  - Résolution des problèmes sous-jacent : Factorisation, DLP
- Les architectures PKI
- Sécurité informatique
  - Sécurité de la programmation
  - Sécurité des infrastructures