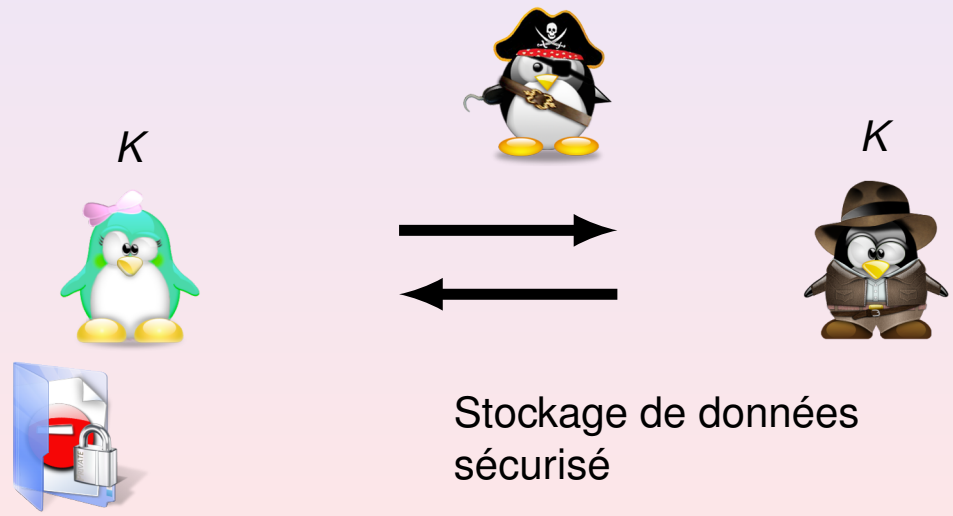
Introduction à la cryptographie: Confidentialité symétrique (Cours 2-3-4) .

Michaël Quisquater (Maître de Conférences,UVSQ)

Confidentialité

S'assurer du caractère secret de l'information



Stockage de données sécurisé

Première partie I

Sécurité des algorithmes de chiffrement symétriques

Plan de la première partie

Notion de sécurité

Notion de sécurité : pas de sens absolu ! Uniquement le sens qu'on lui donne (question de confiance en une technologie).

Différentes notions de sécurité :

- Un algorithme est sûr si il ne dispose pas de faiblesses majeures
- Un algorithme est sûr si il résiste aux attaques connues (temps, mémoire, données)
- Un algorithme est sûr si on peut prouver sa sécurité dans un modèle donné (non vu)

Contexte d'attaque

Pour déterminer si un algorithme résiste à une attaque, il faut préciser le contexte.
Il faut distinguer différents niveaux d'attaques :

- Attaque à texte chiffré seul : Eve ne connaît que le (ou les) chiffré(s) qui passe(nt) sur la ligne ,
- Attaque à clair connu : Eve a accès à des couples de messages (clairs, chiffrés) et utilise ces informations pour déchiffrer un message chiffré particulier ,
- Attaque à clair choisi : Eve peut choisir un (des) message(s) clair(s) et en obtenir le(s) chiffré(s). Elle utilise ces pairs (clair/chiffré) pour déchiffrer un message chiffré particulier ,

Contextes d'attaque

- Attaque à clair choisi adaptative : C'est une attaque à clair choisi dans laquelle le choix des clairs dépend des chiffrés précédemment obtenus ,
- Attaque à chiffré choisi : Eve peut choisir un (des) message(s) chiffré(s) et en obtenir le(s) clair(s). Elle utilise ces pairs (clair/chiffré) pour déchiffrer un message chiffré particulier ,
- Attaque à chiffré choisi adaptative : C'est une attaque à chiffré choisi dans laquelle le choix des chiffrés dépend des clairs précédemment obtenus .

Mesure de l'efficacité d'une attaque

Complexité d'une attaque d'un schéma de chiffrement symétrique.
On distingue trois types de complexité :

- Complexité des données : C'est le nombre moyen d'unités de données (blocs, bits) nécessaires pour mener à bien l'attaque ,
- Complexité en espace : c'est le nombre moyen d'unités de données que l'on doit stocker durant l'attaque ,
- Complexité en temps : c'est le nombre moyen d'opérations nécessaires à l'attaque.

Mesure de l'efficacité d'une attaque (suite)

La complexité d'une attaque est alors définie comme la "plus" (à définir !) coûteuse des trois. Un algorithme est dit résistant à une attaque si ce nombre est trop élevé (dépend des machines disponibles) .

Deuxième partie II

Algorithme de chiffrement symétrique par blocs

Plan de la seconde partie

Définition d'un algorithme de chiffrement par bloc

Soit $\mathbb{Z}_2^n$ l'espace des messages (clairs et chiffrés) et $\mathcal{K}$ l'espace des clés

Une primitive de chiffrement par bloc (de taille n) est une paire d'algorithmes (E, D).

Les algorithmes

$$E : \mathbb{Z}_2^n \times \mathcal{K} \to \mathbb{Z}_2^n$$

et

$$D : \mathbb{Z}_2^n \times \mathcal{K} \to \mathbb{Z}_2^n$$

sont tels que pour tout $K \in \mathcal{K}$, $E(\cdot, K)$ (noté $E_K(\cdot)$) est une permutation de $\mathbb{Z}_2^n$ et $D(\cdot, K)$ est son inverse (noté $D_K(\cdot)$).

Définition d'un algorithme de chiffrement par blocs (suite)

Remarques :

- la taille des messages auxquels s'applique l'algorithme est donc fixée ,
- la définition de l'espace des messages peut se généraliser .

Principe de conception et sécurité : héritage de l'âge artisanal et technique

1. Notion de substitution et de transposition
2. Principe de Kerckhoffs
3. Masque jetable
4. Notion de confusion et diffusion
5. Théorie de l'information (entropie, information mutuelle et chiffrement parfait)

Principe des conceptions : algorithmes itératifs

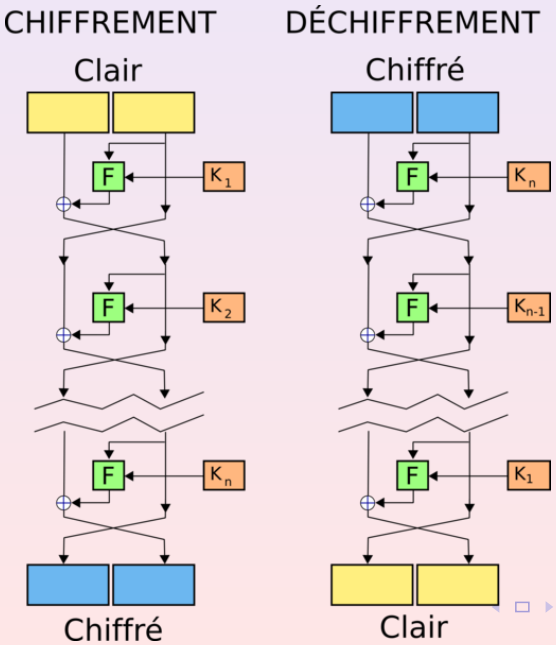
L'âge moderne : utilisation des circuits électriques et des ordinateurs

Description de l'algorithme soit la plus régulière possible : facile pour l'implémentation software et hardware.

Algorithme de déchiffrement soit similaire à l'algorithme de chiffrement (→ algorithme itératif ou produit)

Remarque : Les sous-clés K_i sont déduites à partir d'une clé maître via un algorithme dit de cadencement de clés.

Réseau de Feistel : algorithme de itératif avec tour de Feistel



Réseau de Feistel : algorithme de itératif avec tour de Feistel

Tour de Feistel :

Remarques :

- $\oplus$ est le XOR, S_i sont des fonctions booléennes non-linéaires, LT est une fonction linéaire ,
- Le DES est contruit sur ce modèle (moyennant l'ajout d'une fonction linéaire en amont de F) .

Réseau de Feistel : algorithme de itératif avec tour de Feistel

Remarques

- Le tour de Feistel est une permutation, peu importe si F est une permutation : pas de contrainte de bijection sur les S_i et LT (voir TD)
- L'inverse d'un tour de Feistel à une forme similaire à celui d'un tour de Feistel (voir TD).

Réseau SP (substitution-permutation) : algorithme itératif avec un tour SP

Tour SP :

Remarque :

- Le standard américain actuel (AES) est basé sur ce modèle .

Réseau SP (substitution-permutation) : algorithme itératif avec un tour SP

- Il faut que les S_i et LT soient bijectifs
- Pour que l'algorithme de déchiffrement soit similaire à l'algorithme de chiffrement, on veillera à ce que le tour SP soit "facilement" inversible (un bon choix est de choisir les S_i et LT comme des produits d'involutions) ,

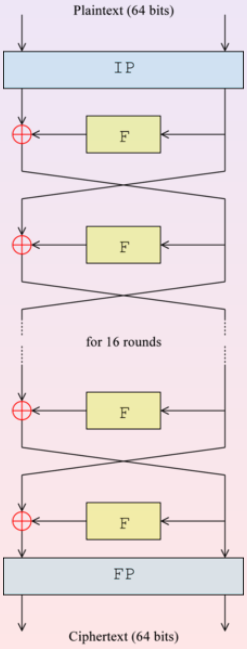
DES (Data Encryption Standard)

En mai 1973, le National Bureau of Standards américain demande la cration d'un chiffrement utilisable par les entreprises. En bonne logique, cet algorithme aurait dû être sélectionné par le NBS. En pratique, ce fut presque le cas : la NSA demanda à ce que Lucifer soit modifié, par ses soins. Ainsi fut créé le DES, qui fut adopté comme standard en novembre 1976.

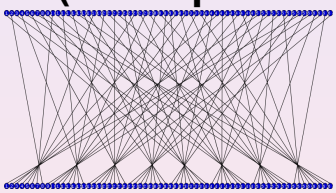
- taille de bloc : 64 bits ,
- taille de la clé : 56 bits ,
- réseau de Feistel, 16 tours .

DES (Data Encryption Standard) (Structure globale)

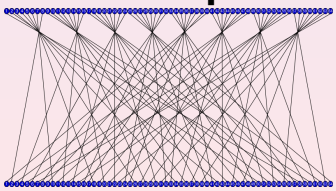
Schéma global de l'algorithme de chiffrement :



IP (initial permutation)



FP : Final permutation

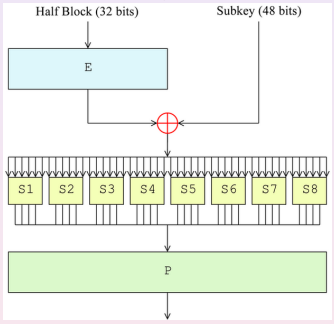


Remarque : le design de IP et FP est inconnu (pas de justification)

DES (Data Encryption Standard) (fonction F)

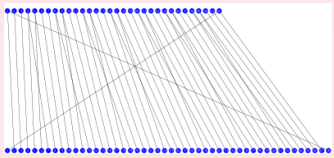
La fonction F :

32 bits $\Rightarrow$ 32 bits



E (Expansion) :

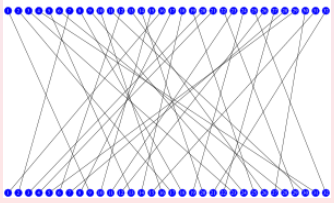
32 bits $\Rightarrow$ 48 bits



S_i (Sbox) :

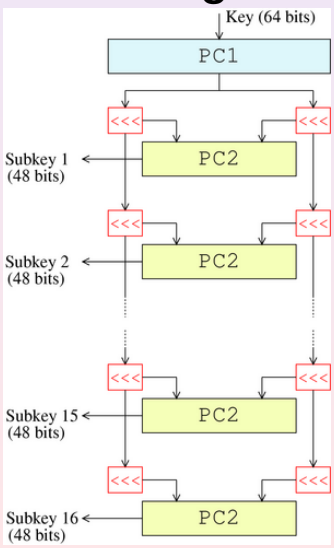
Fonctions non-linéaires de 6 bits $\Rightarrow$ 4 bits

P (Permutation) : 32 bits $\Rightarrow$ 32 bits



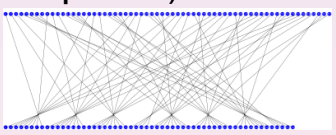
DES (Data Encryption Standard) (algorithme de cadencement de clé)

Structure globale :



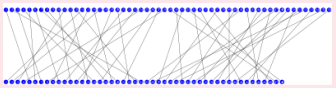
PC1 (permuted choice)

64bits $\Rightarrow$ 56 bits (on enlève les bits de parité)



PC2 (permuted choice)

56bits $\Rightarrow$ 48bits



Propriétés du DES

- Complémentation : $DES_K(m) = DES_{\overline{K}}(\overline{m})$ pour tout message m et clé K ,
- Le DES ne forme pas un groupe (par exemple, pour tout K_1, K_2 il n'existe pas toujours K_3 tels que $DES_{K_1}(DES_{K_2}) = DES_{K_3}$)

AES (Advanced Encryption Standard)

- 1997 : le National Institute of Standards and Technology (NIST,US) fait un appel d'offre pour remplacer le DES. (procédure très différentes de celle du DES qui était une collaboration entre IBM et la NSA).
- Cahier des charges :
  - taille de bloc=128 bits,
  - 3 longueurs de clé=128,192,256 bits,
  - niveau de sécurité équivalent au "triple DES" (voir plus loin) mais plus rapide,
  - Si sélection, le brevet potentiel devait être abandonné.

AES (Advanced Encryption Standard)

- 1998 : le NIST annonce 15 candidats
- 1999 : 5 finalistes retenus (MARS,RC6, Rijndael, Serpent et Twofish) pour une évaluation plus poussée
- 2000 : le NIST retient Rijndael et le baptise AES

AES (Advanced Encryption Standard)

Les concepteurs de Rijndael sont Joan Daemen et Vincent Rijmen (Belgique).

- Taille(s) du bloc : 128 bits
- Longueur(s) de la clé : 128, 192, 256 bits
- Structure de Réseau : (SP) substitution/permutation
- Nombre de tours : 10,12 ou 14 selon la taille de la clé (resp. 128, 192 et 256 bits)

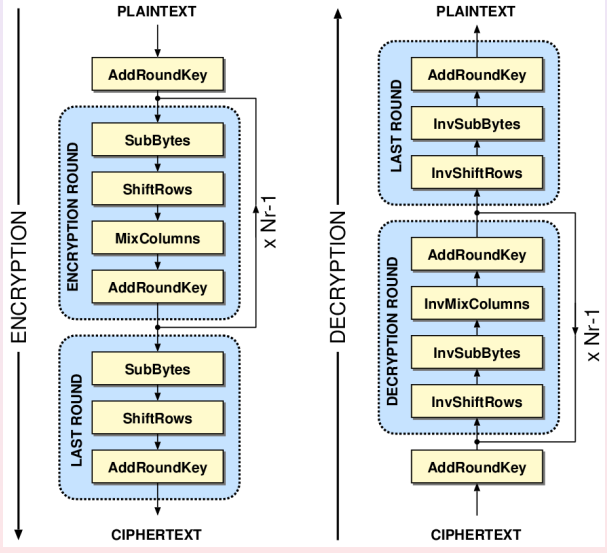
AES (Advanced Encryption Standard)

Trois critères ont été respectés lors de sa conception :

1. Résistance à toutes les attaques connues,
2. Rapidité du code sur la plus grande variétés de plates-formes (logicielles et matérielles) possible,
3. Simplicité dans la conception.

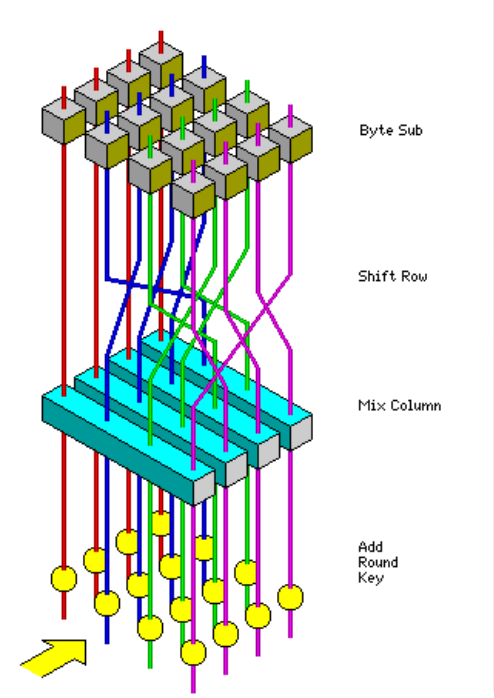
AES (Advanced Encryption Standard)

Schéma global



Remarque : Nr est le nombre de tours

AES (Advanced Encryption Standard) : Structure d'un tour



4 étapes :

- ByteSub
- ShiftRow
- MixColumn
- AddRoundKey

AES (Advanced Encryption Standard) : Structure d'un tour

Décrivons chacune des étapes plus précisément.

Soit $\mathbb{F}_{2^8}$, le corps (champ) à 2^8 éléments. Le polynôme irreductible utilisé est

$$X^8 + X^4 + X^3 + X + 1 \in \mathbb{Z}_2[X]\,.$$

Pour ce faire, représentons le vecteur d'état

$$(m_0, m_1, \cdots, m_{15}) \in (\mathbb{F}_{2^8})^{16}$$

(valeur que prend le message chiffré à chaque endroit de l'algorithme de chiffrement) par la matrice

$$M = \begin{pmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{pmatrix} \in (\mathbb{F}_{2^8})^{4\times 4}\,.$$

AES (Advanced Encryption Standard) : Structure d'un tour

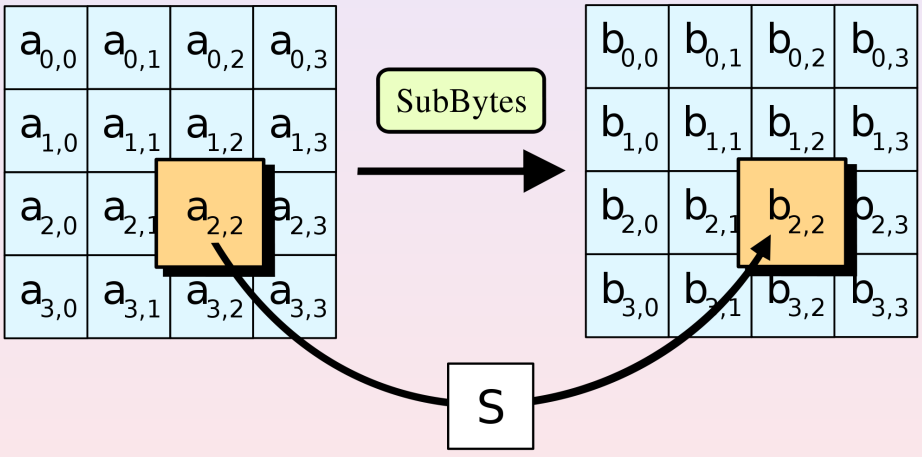
De même, représentons la sous-clé

$$K_i = (k_0^i, k_1^i, \cdots, k_{15}^i) \in (\mathbb{F}_{2^8})^{16}.$$

par la matrice

$$K_i = \begin{pmatrix} k_0^i & k_4^i & k_8^i & k_{12}^i \\ k_1^i & k_5^i & k_9^i & k_{13}^i \\ k_2^i & k_6^i & k_{10}^i & k_{14}^i \\ k_3^i & k_7^i & k_{11}^i & k_{15}^i \end{pmatrix} \in (\mathbb{F}_{2^8})^{4\times 4}.$$

AES (Advanced Encryption Standard) : SubBytes



AES (Advanced Encryption Standard) : SubBytes (suite)

Cette étape consiste donc à appliquer à chaque élément de l'état du début du tour la fonction

$$S : \mathbb{F}_{2^8} \longrightarrow \mathbb{F}_{2^8} : x \mapsto \begin{cases} A \cdot X^{-1} + b & \text{si } x \neq 0 \\ b & \text{si } x = 0 \end{cases}$$

où X^{-1} est la fonction inverse dans $\mathbb{F}_{2^8}$.

AES (Advanced Encryption Standard) : SubBytes (suite)

La matrice A est donnée par

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Le vecteur b par :

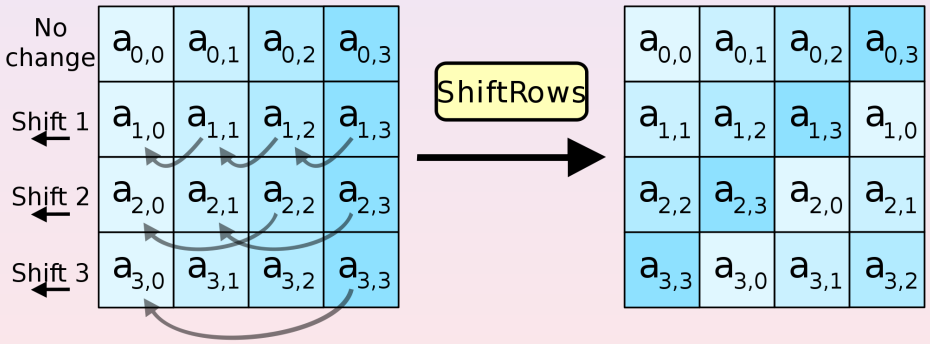
$$b = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Remarque :

- La matrice A est inversible, donc cette étape est inversible.
- La matrice A est circulante et donc il ne faut garder qu'une ligne en mémoire.

AES (Advanced Encryption Standard) : ShiftRows

Cette étape consiste à modifier la matrice d'état de la façon suivante :



AES (Advanced Encryption Standard) : MixColumns

Considérons la matrice d'état à ce stade :

$$S = \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix} \in (\mathbb{F}_{2^8})^{4\times 4}.$$

Cette étape consiste à mutliplier à gauche par la matrice C (i.e. $C \cdot S$) où

$$S = \begin{pmatrix} '02' & '03' & '01' & '01' \\ '01' & '02' & '03' & '01' \\ '01' & '01' & '02' & '03' \\ '03' & '01' & '01' & '02' \end{pmatrix} \in (\mathbb{F}_{2^8})^{4\times 4}.$$

où $'ij'$ est la notation hexadécimale de l'élément du corps fini $\mathbb{F}_{2^8}$.

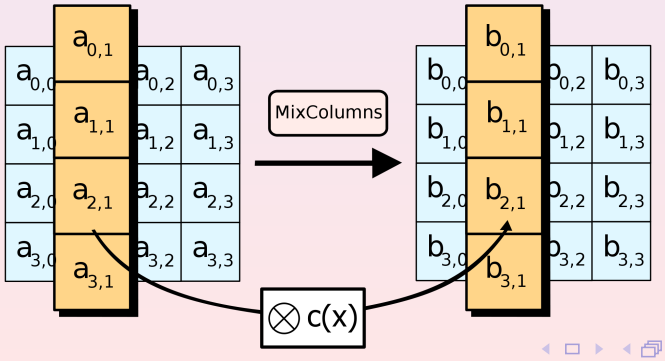
AES : MixColumns (suite)

Notons que l'on cette opération peut également être obtenue en représentant les colonnes par des polynômes à coefficients dans $\mathbb{F}_{2^8}$.

Exemple :

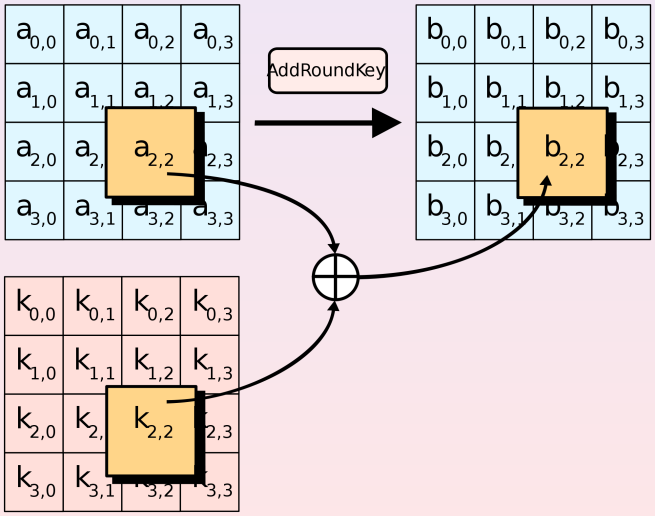
$$s(x) = s_3x^3 + s_2x^2 + s_1x + s_0$$
$$c(x) =' 03'x^3 +' 01'x^2 +' 01'x +' 02'$$

On considère le produit $c(x) \cdot s(x) \bmod X^4 + 1$.



AES (Advanced Encryption Standard) : AddRoundKey

Il s'agit d'additionner la matrice d'état et la matrice des sous-clés K_i. Cela revient à faire un XOR avec la sous-clé.



AES (Advanced Encryption Standard)

Remarques :

- SubByte se charge de la confusion,
- ShiftRow et MixColumns de la diffusion,
- Le dernier tour ne comprend pas de MixColumns.

AES (Advanced Encryption Standard) : Algorithme de cadencement de clé

Soit $K_i \in (\mathbb{F}_{2^8})^{16}$ la sous-clé du tour i. Par définition K_0 est la clé maître (mais aussi la première sous-clé). Les autres clés se déduisent de la récurrence :

Troisième partie III

Attaques de la primitive de chiffrement par blocs

Plan de la troisième partie

Attaque "Brute-Force" ou "recherche exhaustive"

Données :

- Soit $E_{(\cdot)}(\cdot)$ un algorithme de chiffrement tel que le DES.
- Soit P_0 un texte clair quelconque et $C_0 = E_{K_0}(P_0)$
  $\rightarrow$ attaque à texte clair connu.

Objectif : Trouver K_0 tel que $C_0 = E_{K_0}(P_0)$.

Attaque "Brute-Force" ou "recherche exhaustive"

Algorithme 1 : "Recherche exhaustive"

Données : (m, c) avec $E_{K_0}(m) = c$ ((m', c') avec $E_{K_0}(m') = c'$)
Résultat : K_0
pour $K \in \mathcal{K}$ **faire**
  si $E_K(m) = c$ *(et* $E_K(m') = c'$*)* **alors**
    $K \rightarrow Temp$
  fin
fin
retourner *Temp*

Remarque : Il est possible de montrer qu'il est très peu probable que plusieurs clés passe les deux tests ; $\#Temp = 1$

Attaque "Brute-Force" ou "recherche exhaustive" : Machine à casser le DES

ChallengeS (RSA Security, 1997) : Déchiffrer un message commençant par "The unknown message is :" et codé en ASCII.

Récompense : des $ et la "gloire" ;-).

DES challenge I : relevé en 96 jours par le projet DESCHALL sous la direction de Rocke Verser (Colorado). Appel aux ressources de calcul des particuliers (76000 IP's). Sur base volontaire. Motivation : gagnant=4000$.

Attaque "Brute-Force" ou "recherche exhaustive" : Machine à casser le DES

DES challenge II-1 : relevé par distributed.net en 41 jours (janvier et février 1998) en utilisant un principe similaire au projet DESCHALL. Serveur gérant l'attaque à lui été attaqué ;-).

DES challenge II-2 : relevé par Electronic Frontier Foundation (EFF) en 56 heures (17 juillet 1998). Construction d'une machine dédiée (deep crack, 1800 puces). Coût : 250.000$. Concepteur de la machine : Paul Kocher. Prime : 10000 $.

DES challenge III : relevé en 1999 en 22 heures 15 minutes en associant les efforts de Deep Crack et distributed.net.

Attaque "Brute-Force" ou "recherche exhaustive"

Conséquence

A la suite du dernier défit, le DES a été réaffirmé comme standard fédéral mais avec la recommandation d'utiliser le triple DES (voir plus loin).

Attaque "Brute-Force" ou "recherche exhaustive"

Aujourd'hui :

COPACOBANA (Cost-Optimized Parallel Code Breaker).

- 120 FPGAs (Xilinx Spartan3-1000) + carte de contrôle (Xilinx FPGA avec MicroBlaze)
- Consommation de l'ordre de Deep Crack.
- Coût : moins de 10.000$.

15 mars 2007 : attaque par recherche exhaustive du DES en

6 jours et demi en moyenne.

Remarque : FPGAs : circuits entièrement programmables avec des fichiers binaires obtenus à partir de fichier VHDL.

Attaque "Brute-Force" ou "recherche exhaustive"

Temps=pire ennemi de la sécurité d'un algorithme donné à cause de la loi de Moore.

Objectif de la cryptanalyse linéaire

L'objectif de la cryptanalyse linéaire consiste à trouver de l'information sur la clé K à partir de nombreux couples de clairs/chiffrés. Il s'agit donc d'une attaque à clairs connus.

Cette attaque est probabiliste. Cela signifie qu'elle est sujet à une probabilité de succès.

Cette attaque a été inventée par Matsui en 1993. Depuis lors, de nombreuses variantes ont été imaginées.

Cryptanalyse linéaire : cas d'école

Imaginer un algorithme de chiffrement E_K ayant la propriété particulière suivante :

- Si le poids de Hamming de la clé K est pair, alors les messages de poids de Hamming pair (resp. impair) ont des chiffrés de poids de Hamming pair (resp. impair).
- Si le poids de Hamming de la clé K est impair, alors les messages de poids de Hamming pair (resp. impair) ont des chiffrés de poids de Hamming impair (resp. pair).

Cryptanalyse linéaire : cas d'école (suite)

En résumé :

K	wt pair	wt impair
	$P \mapsto P$	$I \mapsto P$
	$I \mapsto I$	$P \mapsto I$

Affirmation : L'observation d'une seule paire de clair/chiffré nous donne alors la parité de la clé K.

Explication : En effet, si la somme du poids de Hamming (nombre de composantes non-nulles) du message et du poids de Hamming du chiffré est pair (resp. impair) alors on sait que le poids de Hamming de la clé est pair (resp. impair).

Cryptanalyse linéaire : cas d'école (exemple)

Exemple :

Considérons l'algorithme de chiffrement $E_K : \mathbb{Z}_2^2 \to \mathbb{Z}_2^2$ défini par

$$\begin{pmatrix} M_0 \\ M_1 \end{pmatrix} \to \begin{pmatrix} C_0 \\ C_1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} M_0 \\ M_1 \end{pmatrix} + \begin{pmatrix} K_0 \\ K_1 \end{pmatrix}$$

Cryptanalyse linéaire : cas d'école (exemple)

Cas 1 : $(K_0, K_1) = (0, 0)$: poids de Hamming pair.

Nous avons,

x	E_K
$(0,0)$	$(0,0)$
$(0,1)$	$(1,0)$
$(1,0)$	$(0,1)$
$(1,1)$	$(1,1)$

Conclusion :

- $P \mapsto P$,
- $I \mapsto I$.

Cryptanalyse linéaire : cas d'école (exemple)

Cas 2 : $(K_0, K_1) = (1, 1)$: poids de Hamming pair.

Nous avons,

x	E_K
$(0,0)$	$(1,1)$
$(0,1)$	$(0,1)$
$(1,0)$	$(1,0)$
$(1,1)$	$(0,0)$

Conclusion :

- $P \mapsto P$,
- $I \mapsto I$.

Cryptanalyse linéaire : cas d'école (exemple)

Cas 3 : $(K_0, K_1) = (0, 1)$: poids de Hamming impair.

Nous avons,

x	E_K
$(0,0)$	$(0,1)$
$(0,1)$	$(1,1)$
$(1,0)$	$(0,0)$
$(1,1)$	$(1,0)$

Conclusion :

- $P \mapsto I$,
- $I \mapsto P$.

Cryptanalyse linéaire : cas d'école (exemple)

Cas 4 : $(K_0, K_1) = (1, 0)$: poids de Hamming impair.

Nous avons,

x	E_K
$(0,0)$	$(1,0)$
$(0,1)$	$(0,0)$
$(1,0)$	$(1,1)$
$(1,1)$	$(0,1)$

Conclusion :

- $P \mapsto I$,
- $I \mapsto P$.

Cryptanalyse linéaire : cas d'école (exemple)

En conclusion, nous avons

$$C_1 \oplus C_0 \oplus M_1 \oplus M_0 = K_0 \oplus K_1 \,. \qquad (1)$$

Par conséquent, si nous observons un couple de clair/chiffré $(\underline{m}, \underline{c})$, nous pouvons déterminer $k_0 \oplus k_1$ c'est-à-dire un bit de la clé.

L'algorithme de chiffrement précédent est affine. Ce n'est pas le cas en général. Par conséquent, il est peu probable que l'on trouve une expression du type (1)

Cryptanalyse linéaire : cas d'école (formalisation)

Soit $\alpha, X \in \mathbb{Z}_2^n$. Dénotons par $\alpha \cdot X = \oplus_{i=1}^{n} \alpha_i \cdot X_i$ avec $\alpha = (\alpha_1, \cdots, \alpha_n)$ et $X = (X_1, \cdots, X_n)$.

Si $\overline{1}$ dénote le vecteur tout à 1, M un message et C son chiffré correspondant lorsque la clé K est utilisée, nous pouvons exprimer la condition sur les poids de Hamming par :

$$\overline{1} \cdot M \oplus \overline{1} \cdot C = \overline{1} \cdot K \, .$$

$\rightarrow$ Généralisation !

Remarque : $\overline{1} \cdot X$ est la parité de X (vaut 1 si le poids de Hamming est impair et 0 sinon)

Cryptanalyse linéaire : cas d'école (suite)

En pratique, il est très peu probable qu'une telle propriété existe !

Cryptanalyse linéaire : cas d'école (suite)

Solution intermédiaire ?

Dénotons la clé par K, un message par M et un chiffré par C.

On peut imaginer qu'il existe des algorithmes ayant la propriété que si la parité de K est nulle (resp. un), la parité de $M + C$ est plus souvent nulle (resp. un) que un (resp. nulle) pour l'ensemble des messages M.

$\rightarrow$ Calculer la parité de $M + C$ pour de nombreuses paires de clairs/chiffrés et déduire la parité de la clé !

La cryptanalyse linéaire tire parti de ce genre de phénomène.

Principe de base de la cryptanalyse linéaire

La cryptanalyse consiste à considérer des expressions plus générales de la forme :

$$\alpha \cdot M + \beta \cdot C = \gamma \cdot K \quad \text{avec probabilité } p\,.$$

avec $\alpha, \beta, M, C \in \mathbb{Z}_2^n$ et $\gamma, K \in \mathbb{Z}_2^k$.

- α est appelé le masque du texte clair,
- β est appelé le masque du texte chiffré,
- γ est appelé le masque de la clé.

En évaluant cette expression en de nombreuses paires de clairs/chiffrés (m^i, c^i), on peut obtenir de l'information sur la clé utilisée. Cette expression est appelée caractéristique linéaire.

Algorithme 1

- Chiffrer n messages m^i et dénotons par c^i les chiffrés correspondants (la clé K utilisée est unique et inconnue),
- Evaluer le membre de gauche de la caractéristique linéaire pour chaque paire (m^i, c^i) et compter le nombre de fois que $\alpha \cdot M + \beta \cdot C$ prend la valeur 0, soit t_n ce nombre.
- Si $t_n > n/2$ et $p > 1/2$ alors $\gamma \cdot K = 0$
  Si $t_n > n/2$ et $p < 1/2$ alors $\gamma \cdot K = 1$
  Si $t_n < n/2$ et $p > 1/2$ alors $\gamma \cdot K = 1$
  Si $t_n < n/2$ et $p < 1/2$ alors $\gamma \cdot K = 0$

Remarque : Notons que si l'expression $\alpha M + \beta C = \gamma K$ est vérifiée avec une probabilité p alors $\alpha M + \beta C = \gamma K + 1$ est vérifiée avec une probabilité $1 - p$.

Probabilité de succès de l'algorithme 1

On peut montrer que pour l'algorithme 1 ait une bonne probabilité de succès, il faut prendre un nombre de paires clairs/chiffrés n de l'ordre de $\frac{1}{(p-\frac{1}{2})^2}$.

Remarque :

- L'algorithme résiste d'autant mieux à l'attaque que n doit être grand c'est à dire p proche de $1/2$.
- Notons que l'on peut généraliser l'algorithme en considérant plusieurs caractéristiques, ce qui permet de retrouver plusiseurs bits de la clé

Questions :

1. Comment déterminer la probabilité de succès de l'algorithme en fonction de n et de p ?
2. Comment déterminer une caractéristique linéaire ?
3. Comment trouver la caractéristique linéaire qui nous donne une probabilité de succès la plus élevée pour n fixé ?

Conception d'algorithme de chiffrement résistant à la crypt. linéaire

L'objectif est de minimiser le biais de la meilleure caractéristique. Cela impose des conditions sur les propriétés des Sboxes et les permutations linéaires.

En particulier, il faut que les permutations aient une bonne diffusion et que les Sboxes aient une bonne non-linéarité (il faut que $P(\alpha x \oplus \beta S(x) = 0)$ soit la plus proche possible de 1/2 pour tout α et β non-nuls simultanément).

Cette théorie sort du cadre de ce cours.

Conclusion

- Attaque à textes clairs connus.
- le nombre de paires clairs/chiffrés nécessaires peut être très important (ex. DES : 2^{42})
- Il existe des généralisations permettant d'obtenir de nombreux bits (ex. 26 pour le DES). Les autres bits se trouvent par recherche exhaustive.
- Domaine encore en développement (dernière publication en 2012).

Cryptanalyse différentielle

La cryptanalyse différentielle découverte par Eli Biham et Adi Shamir en 1991 permet de trouver la clé en utilisant 2^{47} textes clairs.

Des raffinements de cette attaque ont été étudié. En particulier, comment transformer cette attaque en attaque à texte connu ou à chiffré seul (en supposant que le texte anglais est de l'ASCII en anglais P.ex). Voir Biryukov et Kushilevitz (crypto 98 : "From differential cryptanalysis to ciphertext-only attacks")

Tickling Attack

L'attaque-T (Tickling attack) est une variante de la cryptanalyse différentielle. Elle a été découverte lors de la conception du DES par les chercheurs d'IBM et révélée par Don Coppersmith au milieu des années 90. Pendant une vingtaine d'années, le silence a été complet sur cette découverte. A l'époque, elle avait incité les concepteurs de DES à renforcer le contenu des tables de substitution (au lieu de l'affaiblir comme la rumeur le laissait entendre).

Attaque algébrique

L'attaque algébrique permet de retrouver la clé d'un DES réduit à 6 tours à partir d'un seul couple clair/chiffré.

Voir la presentation à la rump session d'asiacrypt 2006 de Nicolas Courtois et Gregory Bard.

Attaque par le milieu

Si la taille de l'espace des clés d'un algorithme donné s'avère trop faible au regard de la recherche exhaustive, on peut être tenté de pallier à ce problème en construisant un nouvel algorithme de chiffrement en utilisant plusieurs fois cet algorithme. Ex. DES.

La première idée est de chiffrer un message en chiffrant deux fois un message en utilisant le DES dont les clés sont différentes, i.e.

$$DoubleE_{K_1,K_2}(\cdot) = E_{K_2}(E_{K_1}(\cdot)).$$

Cette méthode semble à première vue doubler la taille de l'espace des clés résultants.

Attaque par le milieu (suite)

L'attaque meet-in-the-middle montre que la sécurité d'un tel algorithme ("double DES") n'est pas vraiment supérieur à celle d'un DES simple.

Décrivons une attaque à texte clair connu contre ces algorithmes "doubles".

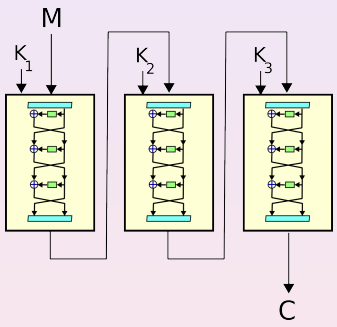
Attaque par le milieu (suite)

Soit (m_0, c_0) avec $c_0 = DoubleE_{K_1,K_2}(m_0)$. On cherche (K_1, K_2).

- Chiffrer m_0, i.e. $E_K(m_0)$, pour les différentes clés $K \in \mathcal{K}$ (liste A : $\mid \mathcal{K} \mid$ chiffrements) et déchiffrons c_0, $E_K^{-1}(c_0)$ avec l'ensemble des clés $K \in \mathcal{K}$ (liste B : $\mid \mathcal{K} \mid$ déchiffrements).
- Trouver l'ensemble des paires de clés (K_1, K_2) telles que $E_K(m_0) = E_K^{-1}(c_0)$. Cela revient à trouver l'ensemble des collisions présentes entre la liste A et B. Pour trouver ces collisions, il suffit de concaténer les deux listes et les classer (de l'ordre de $Log_2(\mid \mathcal{K} \mid) \cdot \mid \mathcal{K} \mid$).

Attaque par le milieu (suite)

L'idée naturelle est alors d'utiliser trois fois le DES : Triple DES ou 3DES. Il existe deux version :



- 3 clés différentes : $DES_{K_3}(DES_{K_2}^{-1}(DES_{K_1}(\cdot)))$.
- 2 clés différentes : $DES_{K_1}(DES_{K_2}^{-1}(DES_{K_1}(\cdot)))$.

Cette utilisation de trois DES a été développée par Walter Tuchman (chef du projet DES chez IBM). Le mode d'usage rend compatible cet algorithme avec DES quand on utilise trois fois la même clé.

Quatrième partie IV

Modes opératoires des algorithmes de chiffrement par bloc

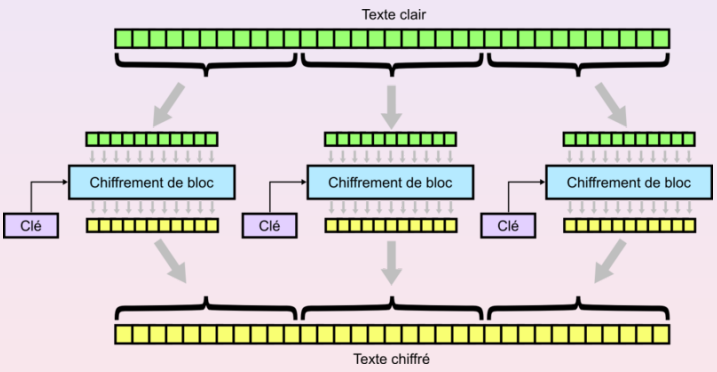
Plan de la quatrième partie

Quid si le bloc est trop petit ?

Idée : découper le message en petits morceaux et les chiffrer séparément
Un algorithme de chiffrement par bloc permet le chiffrement de messages de longueurs relativement courtes, i.e. 64 ou 128 bits généralement. En pratique, on est amené à chiffrer des messages de longueurs beaucoup plus importantes. A cette fin on utilise un procédé, appelé mode, qui permet de chiffrer de chiffrer des messages de longueurs importantes à partir de l'algorithme de chiffrement.
Considérons un message de longueur t. Pour utiliser un algorithme de chiffrement par bloc de longueur n (m_i), on peut découper ce message en $\lfloor t/n \rfloor + 1$ blocs de longueur n.

Mode ECB (electronic codebook)

On applique ensuite chaque bloc à l'algorithme i.e.
$c_i = E_K(m_i)$.



Avantages et inconvénients :

- Si on veut modifier un bloc, on ne doit pas tout rechiffrer
- Attaque par motif
- Pas utilisé en pratique

Attaque par motif

Image originale

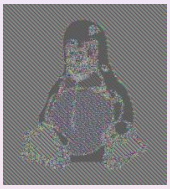


Image chiffrée
par le mode ECB

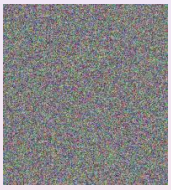
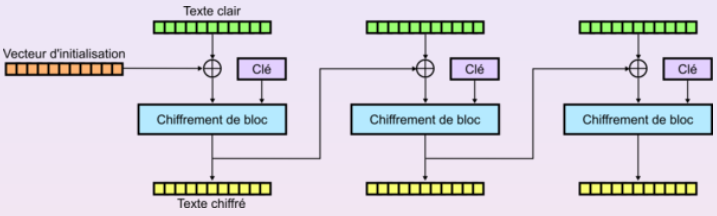


Image chiffrée
via un autre mode

Remarque : Pour éviter les attaques par motifs, il faut que le chiffrement de blocs identiques $\Rightarrow$ chiffrés différents.

Mode CBC (cipher block chaining)

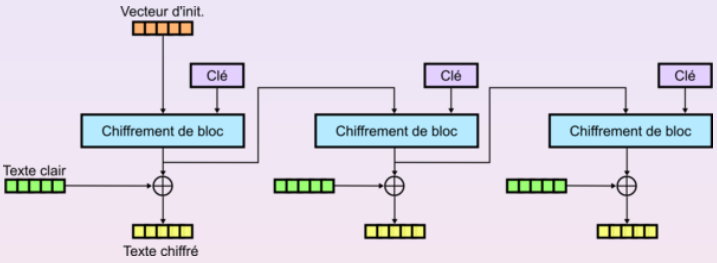


On choisit un vecteur d'initialisation IV de n bits et on chiffre les blocs m_i de la façon suivante :

$$c_0 = IV \quad c_i = E_K(m_i \oplus c_{i-1})\,.$$

- différents IV mène à des schémas de chiffrement différents ,
- Il y a propagation des erreurs ,
- Non flexible pour le chiffrement (si on modifie un bloc, il faut chiffrer à nouveau tous les suivants).

Mode OFB (output feedback mode)



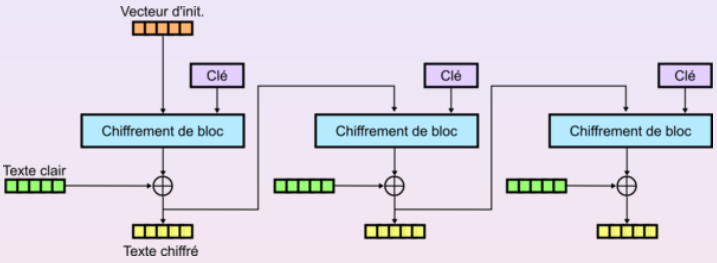
On choisit un vecteur d'initialisation IV de n bits et on chiffre les blocs m_i de la façon suivante. On génère un "flot" de clés

$$z_0 = IV \quad z_i = E_K(z_{i-1})$$

Ensuite, on obtient les chiffrés $c_i = m_i \oplus z_i$.

- IV doit être changée à chaque message (attaque possible sinon) ,
- pas de propagation d'erreurs .

Mode CFB (cipher feedback mode)



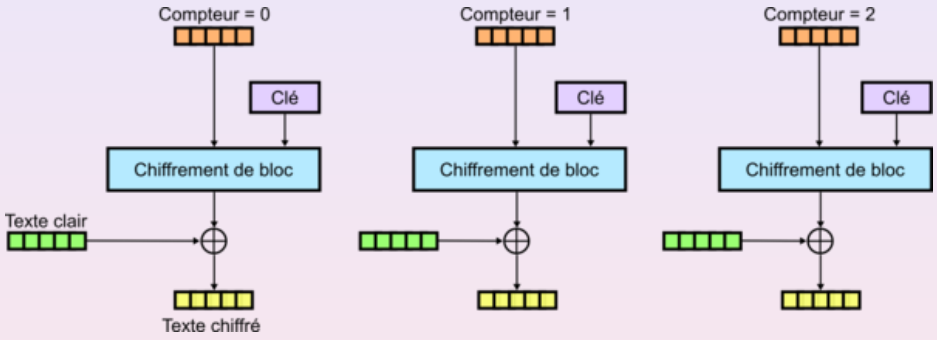
On choisit un vecteur d'initialisation IV de n bits et on chiffre les blocs m_i de la façon suivante. Ce mode fonctionne sur le même principe que le mode OFB mais ici le chiffre sert à générer le flot de cés .

$$c_0 = IV \quad z_i = E_K(c_{i-1}) .$$

Ensuite, on obtient les chiffrés $c_i = m_i \oplus z_i$

- Même dpendance entre les blocs que dans le mode CBC .

Mode CRT (counter mode)



- pas de notion d'IV mais les compteurs fournissent beaucoup d'aléas à travers l'algorithme de chiffrement,
- Il y n'a pas propagation des erreurs ,
- Fexible pour le chiffrement (si on modifie un bloc, il ne faut pas chiffrer à nouveau tous les suivants) .

Source des images :

- png factory
- tux factory
- wikipedia