Théorie de Shannon (Confusion et Diffusion)

Dans "Communication Theory of secrecy systems" publié en 1949, Claude Shannon a introduit des principes sur comment construire des bons systèmes de chiffrement.

Il a en particulier introduit deux propriétés dont chaque bon système de chiffrement doit posséder: la confusion et la diffusion.

Confusion: Rendre les relations entre les bits de la clé, les bits du message clair et les bits du message chiffré le plus complèxe possible.

Exemple:

Clé secrète: $K = (k_1, k_2, k_3, k_4)$

Message clair: $m = (m_1, m_2, m_3, m_4)$

Message chiffré: $c = (c_1, c_2, c_3, c_4)$

$$c_1 = m_1 \oplus m_2 \oplus k_1 \oplus k_2$$
$$c_2 = m_2 \oplus m_3 \oplus k_2 \oplus k_3$$
$$c_3 = m_3 \oplus m_4 \oplus k_3 \oplus k_4$$
$$c_4 = m_4 \oplus m_1 \oplus k_4 \oplus k_1$$

Est-ce que ce chiffrement offre une bonne confusion?

Réponse: Non. Dans le cadre d'une attaque à texte clair connu, en attaquant pourrait construire un système linéaire avec la clé K comme inconnue, le résoudre et récupérer la clé.

Pour qu'un système offre une bonne confusion, il faut que les bits du texte chiffré dépendent de façon non-linéaire de la clé.

Le système par exemple suivant offre une meilleure confusion

$$c_1 = m_1 k_1 k_2 \oplus m_2 m_3 k_3 k_4 \oplus m_1 k_1 \oplus k_2$$
$$c_2 = m_1 m_3 k_1 k_2 k_3 \oplus m_2 k_1 k_4 \oplus m_4 k_2$$
$$c_3 = m_3 k_2 k_4 + m_1 m_4 k_1 k_3 + m_3 k_2 + k_4$$
$$c_4 = m_3 m_4 k_2 k_3 k_4 + m_1 m_2 k_3 k_4 + m_3 m_4 k_1 k_2$$

Diffusion: Chaque bit du texte clair et chaque bit de la clé doivent avoir une influence sur une grande partie du texte chiffré.

(La modification d'un bit du message clair ou de la clé doit entraîner la modification de nombreux bits du message chiffré correspondant.)

L'exemple précédant était un bon exemple de diffusion puisque chaque bit du message clair et chaque bit de la clé affecte la moitié des bits du message chiffré.

Assurer la confusion dans la pratique. On veut que chaque bit du chiffré ait une relation non-linéaire avec les bits du message clair et de la clé.

Pour cela nous utilisons des substitutions.

Dans les chiffrements modernes: Boîtes-S (Sbox en anglais) des 3 à 9 bits.

Exemple d'une boîte-S: $S : \{0,1\}^4 \to \{0,1\}^4$

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S(x)	8	6	7	9	3	12	10	15	13	1	14	4	0	11	5	2

$$y_0 = x_0x_2 + x_1 + x_2 + x_3$$
$$y_1 = x_0x_1x_2 + x_0x_1x_3 + x_0x_2x_3 + x_1x_2 + x_0x_3 + x_2x_3 + x_0 + x_1 + x_2$$
$$y_2 = x_0x_1x_3 + x_0x_2x_3 + x_1x_2 + x_1x_3 + x_2x_3 + x_0 + x_1 + x_3$$
$$y_3 = x_0x_1x_2 + x_1x_3 + x_0 + x_1 + x_2 + 1$$

Assurer la diffusion dans la pratique: Construire des boîtes-S de grande taille est très coûteux. Besoin de mélanger les sorties des boîtes-S entre elles. On utilise alors dans la pratique des fonctions linéaires.

Chiffrement par produit (Claude Shannon): Un chiffrement basé seulement sur la substitution ou seulement sur la permutation ne permet pas d'atteindre un niveau de sécurité élevé.

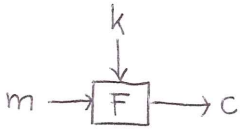
Idée de Shannon: Combiner la diffusion et la confusion pour obtenir un chiffrement robuste.

Chiffrements par blocs : Le chiffrement par blocs est une de deux grandes familles de chiffrements symétriques, l'autre étant le chiffrement à flot.

Dans un système par blocs, le texte clair est découpé en blocs de taille fixe et chiffré bloc par bloc.

Traitement d'un seul bloc: Chiffrer un bloc de message m à un bloc de chiffré c à l'aide d'une clé k.

$$F : \{0,1\}^n \times \{0,1\}^K \to \{0,1\}^n$$
$$(m,k) \mapsto F(m,k) = c$$



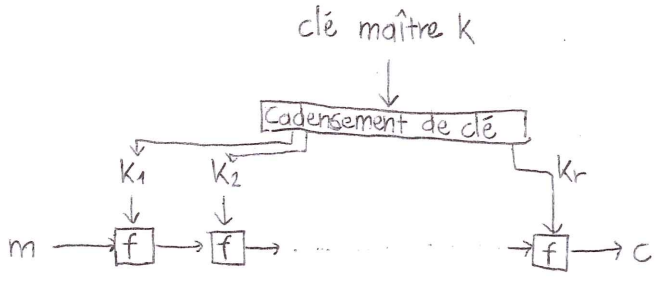
Deux paramètres importants:

- La taille des blocs, n. (Aujourd'hui : 64 ou 128 bits)
- La taille de la clé, K. (" : 128-256 bits)

La taille de la clé il faut qu'elle soit telle qu'une recherche exhaustive sur la clé (avec la connaissance par exemple d'un couple clair-chiffré) ne soit pas possible.

Comment construire un chiffrement par blocs? → Construction itérative.

Itérer une fonction de tour plusieurs fois. La fonction f^r est supposée robuste pour r grand. On appelle f, "fonction de tour".



Utiliser un algorithme de cadencement des clés afin d'éteindre la clé maître à une séquence de r sous-clé.

Avantages: - Implémentation compacte.
- Analyse de sécurité plus facile.

Réseau de Feistel

Introduit par Horst Feistel au débat des années 1970.

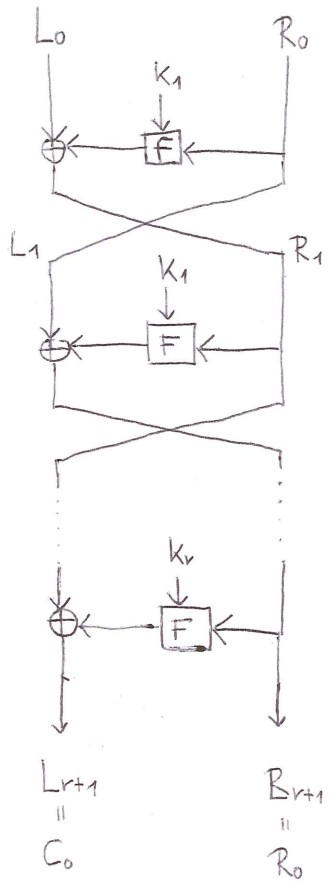
- Couper en deux le bloc de message clair:

  $m = (L_0, R_0)$

- Pour chaque tour $i = 0, \dots, r$ faire:
  - $L_{i+1} = R_i$
  - $R_{i+1} = L_i \oplus F(k_{i-1}, R_1)$

- Bloc de chiffré $c = (C_0, R_0) = (L_{r+1}, R_{r+1})$

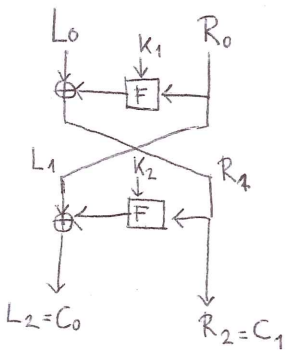
Chiffrement



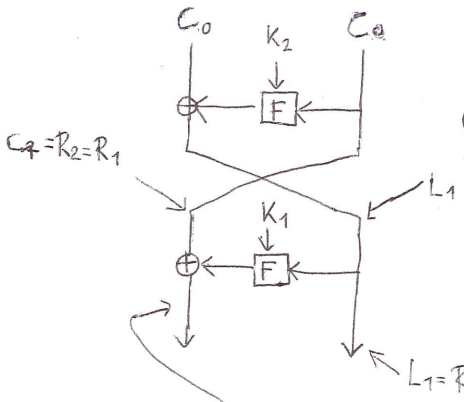
Propriété des Feistels: Déchiffrement avec $K = (k_1, k_2, \dots, k_r)$ est égal au chiffrement avec $K' = (k_r, \dots, k_1)$.

Démonstration pour un réseau de Feistel à 2 tours.

Chiffrement



Déchiffrement



$C_0 \oplus F(k_2, C_1) = L_1 \oplus F(k_2, R_1) \oplus F(k_2, R_1) = L_1$

On peut établir les égalités suivantes:

- $C_1 = R_2 = R_1$
- $C_0 = L_2 = L_1 \oplus F(k_2, R_1)$
- $L_1 = R_0$
- $R_1 = L_0 \oplus F(k_1, R_0)$

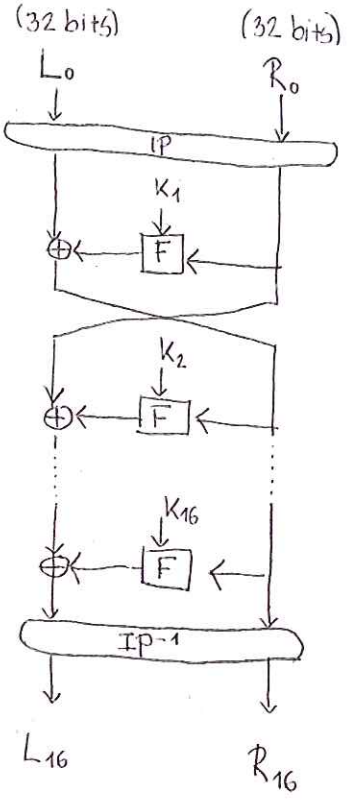
$R_1 \oplus F(k_1, L_1) = R_1 \oplus F(k_1, R_0) = R_1 \oplus L_0 \oplus F(k_1, R_0) \oplus F(k_1, R_0) = L_0$.

Data Encryption Standard (DES)

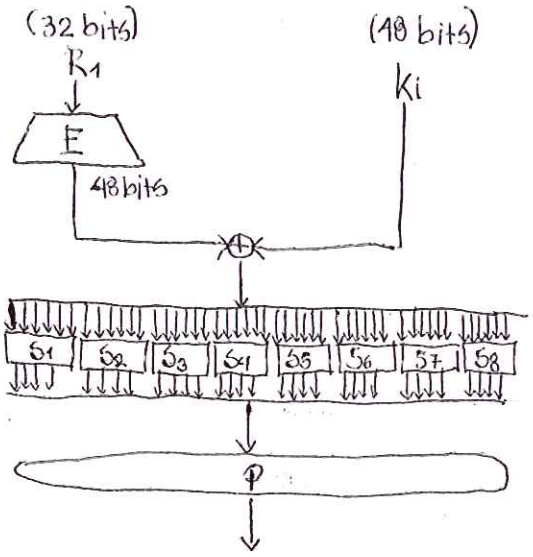
- 1973: Le National Bureau of Standards américain demande la création d'un algorithme de chiffrement pour le milieu des affaires.
- IBM propose Lucifer, conçu en 1971 par Horst Feistel.
- La NSA demande de faire des modifications à Lucifer.
- Des est créé et devient un standard en 1977 (FIPS PUB 46)

Structure du DES

- Taille des blocs: 64 bits
- Longueur de la clé: 56 bits
- 16 tours



Fonction F



La fonction F est basée sur les principes de Shannon (substitution-permutation)

Les choix peuvent paraître arbitraires mais:

- Toutes les briques sont très simples à coder et efficaces en hardware.
- Les boîtes-S ont été choisies selon différents critères pour apporter la résistance aux attaques statistiques et autres
- L'expansion et la permutation garantissent une diffusion rapide.

La fin du DES

1992 : Cryptanalyse différentielle (attaque théorique, 2^{47} textes claires choisis)

1994 : Cryptanalyse linéaire (attaque pratique, une clé est récupérée)

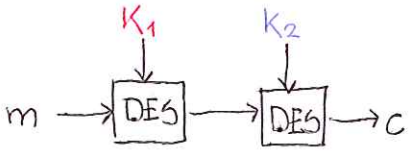
1997: DESCHALL Project (projet de force brute à travers le réseau). Un message chiffré avec DES est retrouvé.

1999: Deep Crack et distributed.net cassent une clé en moins de 23 heures

2004: Le standard est abandonné.

Faiblesse principale du DES: La clé est trop courte.

Double DES : Une idée pour rallonger la clé du DES était de chiffrer chaque message avec DES appliqué deux fois consécutives avec deux clés différentes:



$$c = DES_{K_1}(DES_{K_2}(m))$$

Avec cette méthode on obtient un chiffrement avec

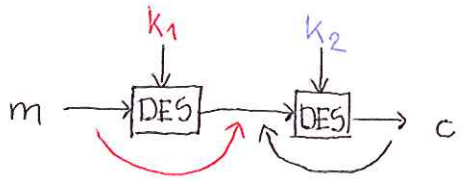
- Taille des blocs: 64 bits
- Taille de la clé: 112 bits.

De cette façon, une recherche exhaustive sur Double DES devrait tester 2^{112} clés dans le pire cas. Cependant, cette idée pour accroître la sécurité du DES s'est avéré être mauvaise, car il existe une attaque du chiffrement double qui diminue considérablement la charge de travail d'une recherche exhaustive. Cette attaque s'appelle "attack meet-in-the middle". (attaque par le milieu, en français)

L'attaque par le milieu

Cette attaque vise à retrouver les clés (k_1, k_2) du chiffrement.

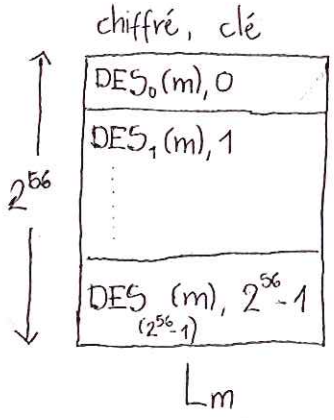
On suppose que l'attaquant possède un couple clair-chiffré (m, c). Le message m a été chiffré en c, avec une clé inconnue (k_1, k_2) que l'attaquant cherche à retrouver.

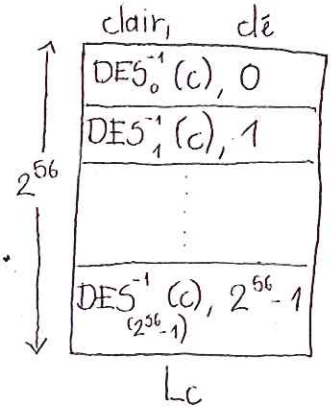


L'attaquant commence par construire deux listes L_m et L_c de la façon suivante:

- Pour tout $k \in \mathcal{K}$, il calcule $DES_k(m)$ et il stocke le résultat dans la liste L_m.
- Pour tout $k \in \mathcal{K}$, il calcule $DES_k^{-1}(c)$ et il stocke le résultat dans la liste L_c.

Les deux listes sont de taille $|\mathcal{K}| = 2^{56}$ chacune.





Soit t un élément commun des deux listes, c'est-à-dire

$t = DES_{k_1}(m)$, pour une clé k_1.

$t = DES^{-1}_{k_2}(c)$, pour une clé $k_2 \Rightarrow c = DES_{k_2}(t)$.

Le couple (k_1, k_2) est une clé candidate pour la clé secrète recherchée.

En effet, $c = DES_{k_2}(t) = DES_{k_2}(DES_{k_1}(m))$

Attention: Il peut avoir plusieurs éléments communs dans ces deux listes. De façon générale ceci dépend de la taille du bloc, n et la taille de la clé, k.

Pour DES, il y a en moyenne, 2^{48} collisions.

Afin de déterminer la bonne clé parmi les clés candidates, un attaquant peut utiliser un deuxième couple clair-chiffré.

Complexité en mémoire: Stockage des deux tables: $2 \times 2^{56} = 2^{57}$ couples (chiffré, clé).

Complexité en temps: Afin de pouvoir trouver les éléments communs dans les deux tables de façon efficace, il faut trier les deux tables. Avec un algorithme de tri rapide (par exemple quicksort), la complexité pour trier une liste de taille n est de $O(n \log n)$. Donc ici, pour trier les deux listes, la complexité est de l'ordre de:

$$2 \cdot 2^{56} \cdot \log_2(2^{56}) = 2 \cdot 56 \cdot 2^{56} \simeq 2^{63}$$

La complexité pour trouver des éléments communs dans les deux listes triées est de l'ordre de $O(n) \simeq 2^{56}$.

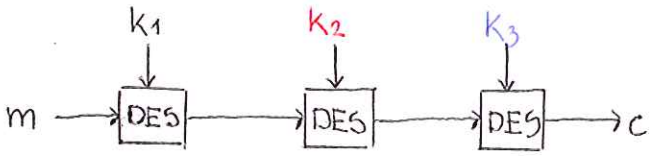
Donc la complexité en temps de l'attaque est autour de 2^{63} opérations.

- On voit alors qu'avec cette attaque, la sécurité du double DES n'atteint pas les 112 bits, mais est autour de 63 bits.
- Cette attaque peut être vu comme un compromis temps-mémoire.

	Temps	mémoire
Attaque par force brute	2^{112}	0
Attaque par dictionnaire	0	2^{112}
Attaque par le milieu	$\simeq 2^{63}$	2^{57}

Attaque par dictionnaire: On chiffre un message m, avec toutes les 2^{112} clés possibles et on sauvegarde les 2^{112} chiffrés accompagnés de la clé correspondante. Pour retrouver la bonne clé, on a qu'à faire une recherche dans la table

Triple DES (3DES)



Il existe plusieurs variantes:

- 2-clés ENC-DEC-ENC (EDE$_2$) $c = ENC_{K_1}(DEC_{K_2}(ENC_{K_1}(m)))$
- 2-clés ENC-ENC-ENC (EEE$_2$) $c = ENC_{K_1}(ENC_{K_2}(ENC_{K_1}(m)))$
- 3-clés ENC-DEC-ENC (EDE$_3$) $c = ENC_{K_3}(DEC_{K_2}(ENC_{K_1}(m)))$
- 3-clés ENC-ENC-ENC (EEE$_3$) $c = ENC_{K_3}(ENC_{K_2}(ENC_{K_1}(m)))$

L'emploie du triple-DES permet d'évite les problèmes lié à la taille de la clé trop courte du DES.

Désavantage: Permormance. Le triple-DES est trois fois plus lent que DES. En 2000, il y a eu une migration vers un algorithme plus récent, le AES (Advanced Encryption Standard).

Modes opératoires (Comment chiffrer des messages longs?)

Les algorithmes de chiffrement par blocs chiffrent les messages en les découpant en blocs de taille fixe. Un mode opératoire est la manière de traiter les blocs clairs et chiffrés afin de former le texte chiffré final.

Il existe aujourd'hui cinq modes opératoires standarisés. Chaque mode a des propriétés de sécurité différentes et une performance différente.

- Electronic Code Book Mode (ECB)

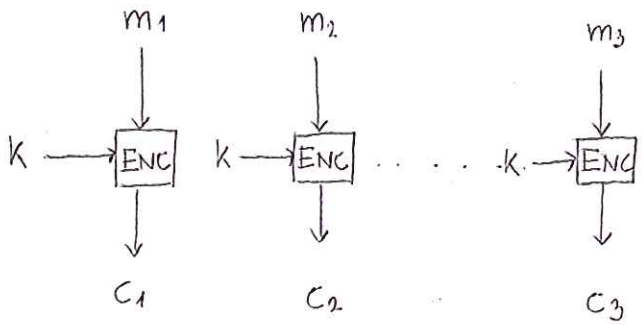
Il s'agit du mode opératoire le plus naturel et simple. Avec ce mode de chiffrement tous les blocs clairs sont chiffrés indépendamment les uns des autres en utilisant la même clé.

Soit m le message à chiffrer. Ce message est découpé en blocs de taille égale $m_1, m_2, \ldots, m_t$. Pour chaque bloc on calcule

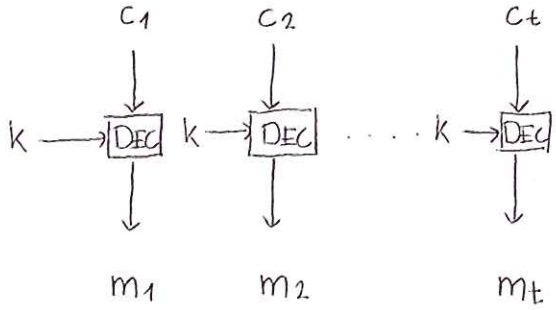
$$c_i = ENC_K(m_i), \quad i = 1, \ldots, t.$$

Le chiffré c est simplement $c = c_1 c_2 \cdots c_t$, la concaténation des t blocs.

Chiffrement



Déchiffrement



Avantages

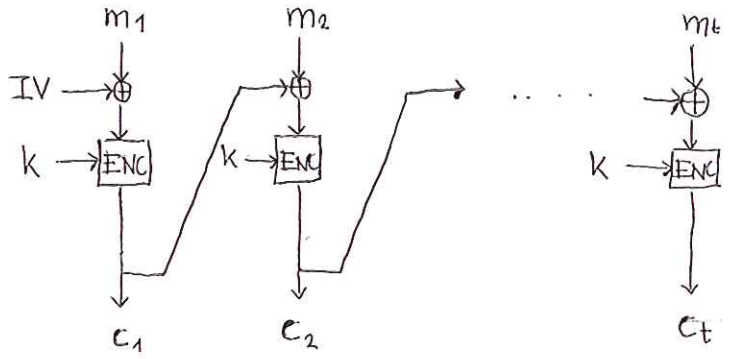
1. Le chiffrement et le déchiffrement sont parallélisables.
2. Possibilité d'un déchiffrement partiel des données.
3. Pas de propagation d'erreurs.

Désavantage: (Déchiffrement de message longs)

Des blocs de message identiques sont chiffrés en blocs de chiffré identiques.

Cipher Block Chaining (CBC)

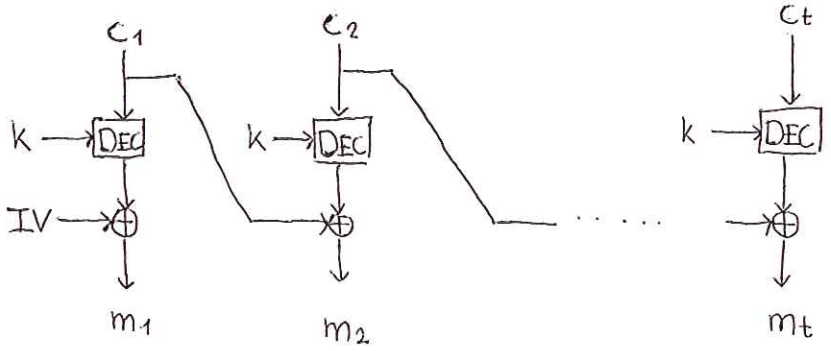
Chiffrement



$$c_1 = ENC_K(m_1 \oplus IV)$$
$$c_i = ENC_K(m_i \oplus c_{i-1}),\ i = 2, \dots, t$$

Un bloc de chiffré c_i dépend de $m_1, \dots, m_j$.

Déchiffrement



Le déchiffrement est parallélisable.

Désavantage: Le chiffrement ne peut pas être parallélisé.

Fuite d'information dans le mode CBC:

Si $c_i = c_j \Rightarrow ENC_K(m_i \oplus c_{i-1}) = ENC_K(m_j \oplus c_{j-1})$

$\Rightarrow m_i \oplus c_{i-1} = m_j \oplus c_{j-1}$

$\Rightarrow \boxed{m_i \oplus m_j = c_{i-1} \oplus c_{j-1}}$