La cryptographie symétrique

Grands nombres

Probabilité de mourir foudroyé	1/9 milliards (2^{33})
Probabilité de gagner au lotto américain	1/4000000 (2^{22})
Temps d'ici à ce que le soleil explose	10^9 (2^{30}) années
Age de la terre	10^9 (2^{30}) années
Age de l'univers	10^{10} (2^{34}) années
Nombre d'atomes constituant la terre	10^{51} (2^{170})
Nombre d'atomes constituant la galaxie	10^{67} (2^{223})
Nombre d'atomes constituant l'univers	10^{77} (2^{265})
Durée de vie de l'univers	10^{11} (2^{37}) années

Chiffrement par blocs

Chiffrements par blocs

Réseaux de Shannon

L'idée générale du chiffrement par blocs est la suivante:

Remplacer les caractères par un code binaire

Découper cette chaîne en blocs de longueur donnée

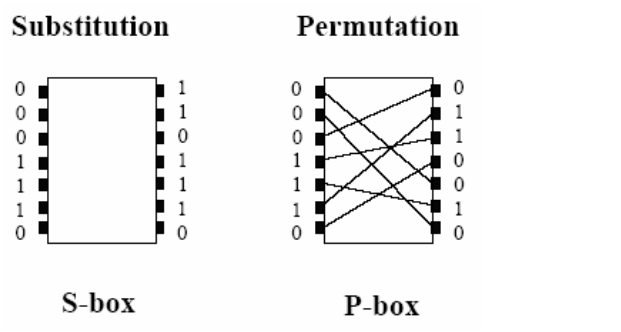
Chiffrer un bloc en l'"additionnant" bit par bit à une clef.

Déplacer certains bits du bloc.

Recommencer éventuellement un certain nombre de fois l'opération 3. On appelle cela une **ronde**.

Passer au bloc suivant et retourner au point 3 jusqu'à ce que tout le message soit chiffré.

Réseaux S-P de Shannon



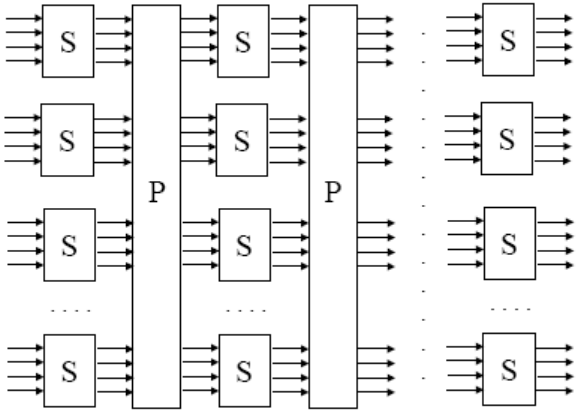
Chiffrement par substitution

Les substitutions consistent à remplacer des symboles ou des groupes de symboles par d'autres symboles ou groupes de symboles dans le but de créer de la confusion.

Chiffrement par transposition

Les transpositions consistent à mélanger les symboles ou les groupes de symboles d'un message clair suivant des règles prédéfinies pour créer de la diffusion. Ces règles sont déterminées par la clé de chiffrement. Une suite de transpositions forment une permutation.

Réseaux S-P de Shannon

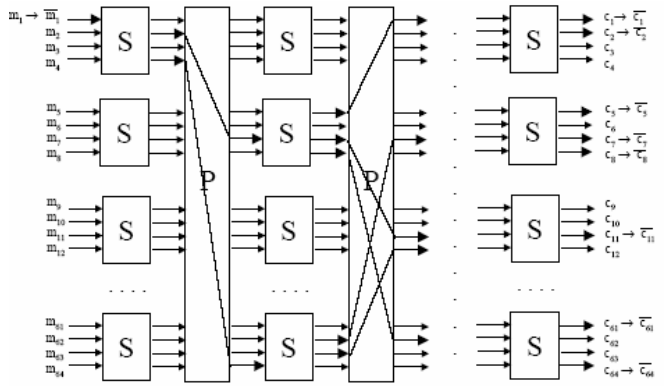


Chiffrement par produit : la combinaison des deux

Le chiffrement par substitution ou par transposition ne fournit pas un haut niveau de sécurité, mais en combinant ces deux transformations, on peut obtenir un chiffrement robuste. La plupart des algorithmes de cryptage par clés symétriques utilisent le chiffrement par produit.

Un « round » est complété lorsque les deux transformations ont été faites une fois (substitution et transposition).

Effet d'avalanche



Effet d'avalanche :

Propriété des chiffrements par blocs composés de couches (layers) ou "rounds" avec un petit changement à l'entrée.

Le changement d'un simple bit d'entrée produit généralement de multiples changements de bits après un round, plusieurs autres changements de bits après un autre round jusqu'au changement éventuel de la moitié du bloc.

Structures de Feistel

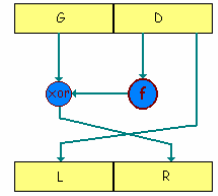
•Structure décrite en 1973 (Feistel – IBM)

•Adapte la structure de Shannon afin de rendre la structure inversible ce qui permet de réutiliser le matériel de chiffrement pour déchiffrer un message. La seule modification s'effectue sur la manière dont la clé est utilisée

•Une couche de S-Box et de P-Box est utilisée par Round

Feistel - principe

- Le texte est chiffré à partir de la même transformation sur le texte clair dans chaque round.



Description du schéma :

Dans une construction de Feistel, le bloc d'entrée d'un round est séparé en deux parties.

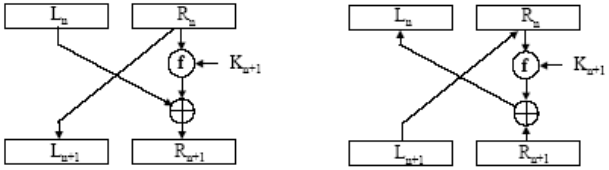
•La fonction de chiffrement est appliquée sur la première partie du bloc

•et l'opération binaire OU-Exclusif est appliquée sur la partie sortante de la fonction et la deuxième partie.

•Ensuite les deux parties sont permutées et le prochain round commence.

Feistel déchiffrement

- L'avantage est que la fonction de chiffrement et la fonction de déchiffrement sont identiques. Ainsi la fonction n'a pas à être inversible, c'est la structure qui l'est



Feistel – Choix des paramètres

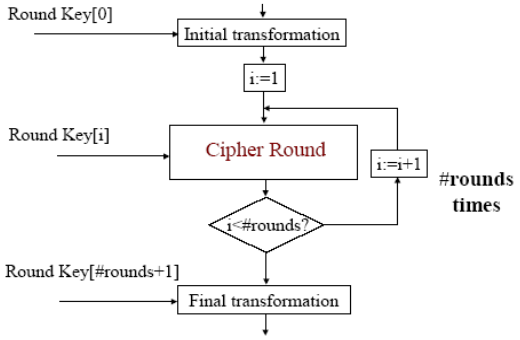
- La réalisation d'un tel réseau dépend des choix effectués pour les paramètres suivants :
  - Taille du bloc :
    - ↗ taille → ↗ sécurité
  - Taille de clé :
    - ↗ taille → ↗ sécurité
  - Nombre de cycle :
    - ↗ nombre → ↗ sécurité
  - Algorithme de génération des sous clés :
    - ↗ complexité → ↗ difficultés de cryptanalyse
  - Vitesse de chiffrement/déchiffrement logiciel

D.E.S. (Data Encryption Standard)

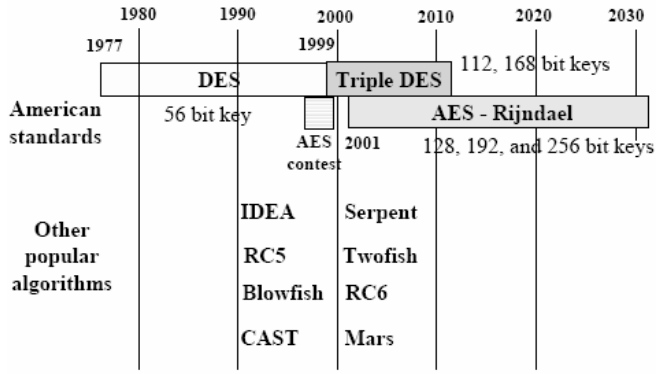
Présentation - objectifs

Le D.E.S. (Data Encryption Standard, c'est-à-dire Standard de Chiffrement de Données) est un standard mondial depuis plus de 15 ans. Bien qu'il soit un peu vieillissant, il résiste toujours très bien à la cryptanalyse et reste un algorithme très sûr.

Au début des années 70, le développement des communications entre ordinateurs a nécessité la mise en place d'un standard de chiffrement de données pour limiter la prolifération d'algorithmes différents ne pouvant pas communiquer entre eux. Pour résoudre ce problème, L'Agence Nationale de Sécurité américaine (N.S.A.) a lancé des appels d'offres. La société I.B.M. a développé alors un algorithme nommé Lucifer, relativement complexe et sophistiqué. Après quelques années de discussions et de modifications, cet algorithme, devenu alors D.E.S., fut adopté au niveau fédéral le 23 novembre 1976.



Chiffrement symétriques les plus fréquents



DES – Cahier des charges

- haut niveau de sécurité
- complètement spécifié et facile à comprendre
- sécurité indépendante de l'algorithme lui-même
- disponible à tous
- adaptable à diverses applications
- possibilité d'implantation économique en matériel
- efficace d'utilisation, validable, et exportable

Propriétés du DES

- Aléatoire
- Propriété d'avalanche
- Propriété de complétude
  - Chaque bit de sortie est une fonction complexe de tous les bits d'entrée
- Non-linéarité
  - La fonction de chiffrement est non- affine pour toute valeur de la clé
- Immunité contre la corrélation

Transformation linéaire :

transformation qui respecte la condition :

$T(X_{[m \times 1]}) = Y_{[n \times 1]} = A_{[n \times m]} * X_{[m \times 1]}$

ou

$T(X_1 \oplus X_2) = T(X_1) \oplus T(x_2)$

La non-linéarité d'une fonction est la mesure du nombre de bits qui doivent changer dans la table de vérité d'une fonction booléenne pour s'approcher d'une « affine function »

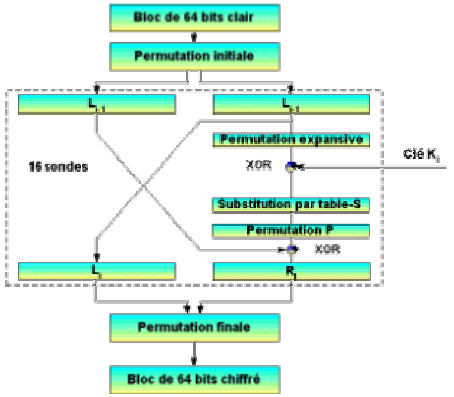
Transformation affine

transformation qui respecte la condition :

$T(X_{[m \times 1]}) = Y_{[n \times 1]} = A_{[n \times m]} * X_{[m \times 1]} \oplus B_{[n \times 1]}$

D.E.S. (Data Encryption Standard)

Algorithme



Etapes de l'algorithme :

1. Diversification de la clé : fabrication de 16 sous-clés
2. Permutation initiale
3. Calcul médian (16 fois) : application d'un algorithme complexe appliqué en fonction de la clé
4. Permutation finale

DES – principes généraux

- La clé
  - Constituée de 64 bits dont 56 sont utilisés aléatoirement dans l'algorithme
  - Les 8 autres peuvent être utilisés pour la détection d'erreurs
  - Chacun des 8 bits est utilisé comme bit de parité des 7 groupes de 8 bits.
  - Nombre total de clés : 2^{56}

Permutation initiale (IP)

- Les 64 bits du bloc d'entrée subissent la permutation

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

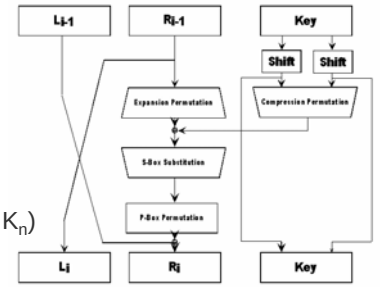
Le premier bit sera le bit 58, le second le bit 50, etc.

Calcul médian

- 16 itérations identiques
- Traite 2 blocs à la fois
  - Bloc de 32 bits (données)
  - Bloc de 48 bits (clés)
- Résultat = bloc de 32 bits
- 64 bits initiaux sont divisés en 2 bloc (L et R)
  - L ∋ bits pairs, R ∋ bits impairs

Calcul médian

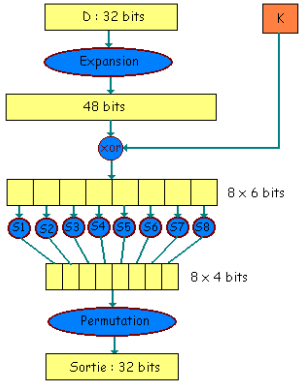
- Itérations
  - $T_n = L_n R_n$
    - $L_n = t_1 \ldots t_{32}$
    - $R_n = t_{33} \ldots t_{64}$
  - Où
    - $L_n = R_{n-1}$
    - $R_n = L_{n-1} \oplus F(R_{n-1}, K_n)$
    - $K_n = G(K,n)$



Calcul médian

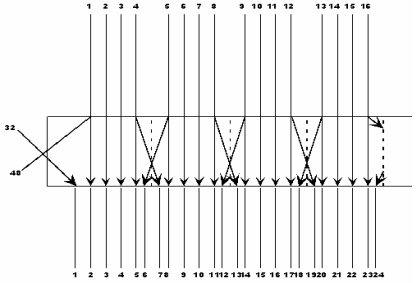
- Fonction F :
  1. Expansion
  2. Ajout de la clé
  3. Transformations

     (S-Box, P-Box)
  4. Assemblage



Expansion

- Les 32 bits sont étendus à 48 bits grâce à une table d'expansion → $E(R_{n-1})$



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

« effet d'avalanche »

Addition de la sous-clé

- Le résultat de l'expansion est additionné (xor) à la sous-clé K_n correspondant à l'itération

$$E(R_{i-1}) \oplus K_i = B_1 B_2 \quad \ldots \quad B_8$$

B_j bloc de 6 bits

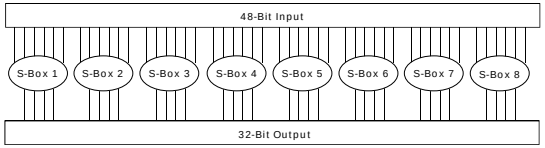
$B_j = b_1 b_2 b_3 b_4 b_5 b_6$

Transformations – S-box

- Chaque bloc B_j constitue ensuite l'entrée de l'opération de substitution réalisée sur base des S-Box

$$B_j \rightarrow S_j(B_j)$$

Bloc de 6 bits — Bloc de 4 bits



Transformations – S-box

- L'opération de substitution consiste pour chaque S-box à calculer:
  - b_1b_6 = n° de ligne
  - $b_2b_3b_4b_5$ = n° de colonne

N° de colonne

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

N° de ligne

Les 8 S-Boxes du DES

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
0	15	1	2	14	6	11	3	4	9	7	2	13	2	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	1	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	122	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

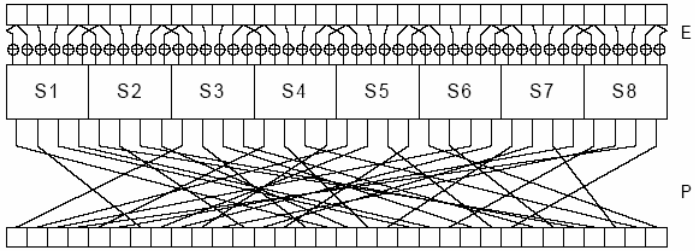
Transformations – P-Box

- L'opération de permutation est réalisée sur le résultat de la substitution des S-box et est basée sur la table suivante:

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

- Le résultat de cette dernière permutation est noté $F(R_{n-1},K_n)$

Résultat du calcul médian pour une ronde



Permutation finale (IP^{-1})

- Permutation inverse de la permutation initiale

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

Calcul de la clé (G(K,n))

- Clé initiale de 64 bits

1. Réduction à 56 bits
   - Les bits de parité sont enlevé
   - Permutation de la clé

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

Calcul de la clé (G(K,n))

2. Division de la clé
   - Division en 2 sous clés de 28 bits
3. Rotation de la clé
   - A chaque tour, chaque sous clé subit une rotation d'un ou 2 bits vers la gauche
4. Regroupement des 2 sous clés

Iteration i	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	22
16	1

Calcul de la clé (G(K,n))

5. Réduction
   - La sous clé résultante (56 bits) est réduite à une sous clé de 48 bits sur base de la table ci-contre
6. Le résultat de cette réduction est la sous-clé K_n additionnée avec $E(R_{n-1})$

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Déchiffrement

- Appliquer le même algorithme mais à l'inverse en tenant bien compte que chaque itération du déchiffrement traite les mêmes paires de blocs utilisés dans le chiffrement
- $R_{n-1} = L_n$
- $L_{n-1} = R_n \oplus f(L_n, K_n)$

D.E.S. (Data Encryption Standard)

Faiblesses et améliorations

DES – considérations sur la sécurité

- Questions relatives à la conception de l'algorithme
  - caractère confidentiel de la conception
  - présence de "trappes"?
  - possibilité d'une faiblesse fondamentale?
- Le nombre d'itérations (16) est il suffisant ?
- La taille de la clef (56 bits) est elle suffisante?
  - originalement, Lucifer prévoyait 128 bits
  - possibilité d'une attaque « force brute » réussie?
  - possibilité d'une attaque de type « parallèle »
  - possibilité de réussite d'une attaque de type "texte en clair choisi"
- Toutes ces questions ont reçu des réponses satisfaisantes

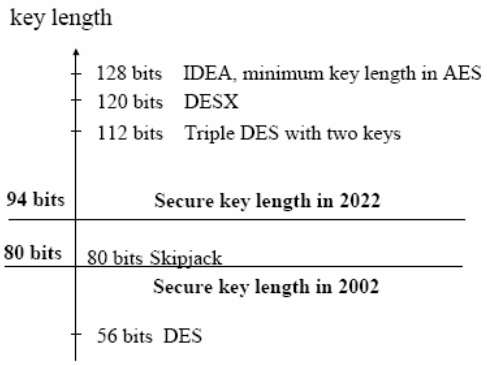
DES - Faiblesses des clés

- Problème des compléments
  - si c = DES(p, k), alors !c = DES(!p, !k)
  - n'est pas un problème sérieux
- Clefs "faibles"
  - 0x0101….0101, 0xFEFE….FEFE, 0x1F1F….1F1F, 0xE0E0….E0E0
- Clefs "semi-faibles"
  - paires de clefs dont la deuxième peut décrypter un message encrypté par la première; p.ex. 0x01FE…01FE et 0xFE01….FE01
  - il existe six paires de ce genre
- Existence possible de paires de clés qui génèrent le même texte encrypté à partir du même texte en clair ("clustering")

Clé complémentaires : 2^{55} clés à tester

64 clés faibles en tout

Longueurs de clés sures pour 20 ans



6 attaques sur le DES

- Recherche exhaustive
  - Une clé après l'autre, on essaie de déchiffrer un bloc de données, l'information recueillie sur le texte clair nous permettant de reconnaître la bonne
  - besoin, en moyenne, de 2^{55} essais
- Une machine dédiée
  - **Deep Crack**, tel est le nom de cette machine extraordinaire, a coûté moins de 210'000$.
  - un temps de recherche moyen situé entre 4 et 5 jours. (1998)

Recherche exhaustive

DES est un algorithme capable de chiffrer un bloc de données P de 64 bits à l'aide d'une clé secrète K de 56 bits. Le résultat est un bloc de données chiffrées de 64 bits que nous noterons C. L'opération de déchiffrement *P=DK(C)* est, grâce à la structure même de l'algorithme, quasiment identique à l'opération de chiffrement *C = EK(P)*, la seule différence étant une légère modification de la préparation de la clé.

Imaginons que nous disposions d'un bloc de données chiffrées *C'* et que nous voulions trouver la clé secrète correspondante. Si nous disposons d'un peu d'information sur la structure ou le contenu des données en clair (texte ASCII, image JPEG, paquet de réseau ayant une structure connue,...), la méthode la plus simple est une recherche exhaustive de la clé correspondante parmi les $2^{56} \simeq 7{,}2 * 10^{16}$ possibilités. Le principe, très simple, est le suivant: une clé après l'autre, on essaye de déchiffrer le bloc de données, l'information sur le texte clair nous permettant de reconnaître la bonne et donc d'interrompre la recherche. Nous aurons besoin, en moyenne, de 2^{55} essais avant de terminer notre attaque.

Une machine dédiée

La complexité, mesurée en terme de quantité de calcul, d'une recherche exhaustive est certes énorme, mais à présent, grâce aux progrès de la technologie des microprocesseurs depuis 1977, elle est loin d'être inaccessible. DES est un algorithme orienté hardware qui a le gros désavantage pour le cryptanalyste d'être très lent en software. Une plate-forme PC actuelle, à savoir un processeur Intel Pentium III 666MHz, peut examiner environ 2 millions de clés par seconde, ce qui implique un temps de recherche moyen de 600 années pour un seul PC.

Une solution matérielle a été proposée et réalisée par l'EFF (Electronic Frontier Fundation) en 1998, dans le seul but de prouver que DES n'est pas (ou plus) du tout un algorithme sûr. **Deep Crack**, tel est le nom de cette machine extraordinaire, a coûté moins de 210'000$. Elle est constituée de 1536 chips qui sont capables de décrypter un bloc en 16 cycles d'horloge, le tout étant cadencé à 40 MHz. Ces caractéristiques lui donnent la possibilité d'examiner 92 milliards de clés par seconde, ce qui donne un temps de recherche moyen situé entre 4 et 5 jours.

Vu le budget modeste de l'EFF, il n'est pas difficile de tirer des conclusions alarmistes sur la sécurité de DES vis-à-vis d'organismes gouvernementaux (tel la NSA, organe américain chargé de l'espionnage électronique) ou d'organisations criminelles.

6 attaques sur le DES

- Un gigantesque cluster
  - Regroupement d'ordinateur sur le net qui partagent leur puissance de calcul
  - Le 19 janvier 1999, RSA Labs, a cassé une clé en moins de 23 heures.
- Un compromis temps-mémoire
  - Générer un dictionnaire de paire (K,C)
  - besoin énorme de mémoire (1.5 milliard de Go), ainsi que de beaucoup de temps pour construire cette table.
  - Cette attaque demande environ 1000 Go de capacité de stockage et 5 jours de calculs sur un simple PC

Un gigantesque cluster

Il n'est même pas nécessaire de disposer de gros moyens financiers pour pouvoir casser DES. De la bonne volonté et quelques bénévoles sont suffisants. Le 19 janvier 1999, dans le cadre d'un concours sponsorisé par une des entreprises majeures en sécurité informatique, RSA Labs, a cassé une clé en moins de 23 heures. L'organisation **distributed .net** regroupe des milliers d'ordinateurs (de la machine la plus simple aux serveurs multiprocesseurs les plus puissants) sur Internet qui fournissent gracieusement leur puissance de calcul à disposition lorsqu'ils sont inactifs. Plus de 100'000 ordinateurs ont reçu un certain nombre de clés à contrôler via le réseau, ce qui a permis un taux de traitement de 250 milliards de clés par seconde.

Un compromis temps-mémoire

Nous avons déjà abordé l'idée de recherche exhaustive de clé: on a à disposition un couple texte clair - texte chiffré et on essaye les 2^{56} clés possibles. Cette méthode ne demande quasiment aucune mémoire, mais en contrepartie, on devra essayer en moyenne 2^{55} clés avant de tomber sur la bonne. D'un autre côté, il serait imaginable, pour un texte clair x donné, de pré-calculer le texte chiffré $y = E_K(x)$ correspondant pour toutes les 2^{56} clés K, et de stocker les paires (y_K, K) triées par leur première coordonnée.

Plus tard, lorsque l'on obtient un texte chiffré y à partir de x, il est possible, par une simple recherche dans notre table, de retrouver la clé correspondante. Nous pouvons noter que cette recherche demande un temps constant; par contre, nous avons un besoin énorme de mémoire (1.5 milliard de Go), ainsi que de beaucoup de temps pour construire cette table. De plus, le bénéfice ne devient effectif que si l'on doit chercher plus d'une clé à partir du texte chiffré d'un même message.

Un algorithme, dit de *compromis temps-mémoire*, a été proposé en 1980 par Hellman. Il combine à la fois une demande de mémoire moindre que celle de notre proposition, ainsi qu'un temps de calcul inférieur à celui d'une recherche exhaustive. Cette attaque demande environ 1000 Go de capacité de stockage et 5 jours de calculs sur un simple PC.

6 attaques sur le DES

- Cryptanalyse différentielle
  - possibilité de produire au moyen de textes clairs choisis les textes chiffrés correspondants avec cette clé inconnue
  - La meilleure attaque différentielle connue demande actuellement 2^{47} textes clairs choisis
- Cryptanalyse linéaire
  - l'attaque la plus efficace connue à ce jour contre DES
  - dictionnaire de (M,C) $\rightarrow$ attaque à texte clair connu
  - statistiques sur un flot de 2^{43} couples de données (soit la bagatelle de deux fois 64 To).

Cryptanalyse différentielle

En 1990, deux chercheurs israéliens du Weitzmann Institute, Biham et Shamir, ont présenté une nouvelle attaque, la cryptanalyse différentielle. En utilisant cette méthode, les deux chercheurs ont proposé pour la première fois une façon de casser DES qui demande moins de temps de travail qu'une recherche exhaustive.

Imaginons que nous disposions d'un boîtier électronique capable de chiffrer des données avec une clé inconnue *câblée* dans le matériel; de plus, nous ne disposons pas du matériel nécessaire pour récupérer *physiquement* la clé dans le boîtier. Il nous est cependant possible de produire au moyen de textes clairs choisis les textes chiffrés correspondants avec cette clé inconnue.

La meilleure attaque différentielle connue demande actuellement 2^{47} textes clairs choisis. La phase d'analyse calcule la clé à l'aide de cet énorme amas de données. Une propriété intéressante de cette attaque est qu'il est possible de la monter même si le nombre de données disponibles est petit; la probabilité de succès augmente linéairement avec ce nombre.

Cryptanalyse linéaire

Une autre attaque théorique importante est la cryptanalyse linéaire. Elle a été proposée par Matsui, de Mitsubishi Electronics, en 1993. Bien qu'elle ne soit que théoriquement utile dans le monde réel, c'est l'attaque la plus efficace connue à ce jour contre DES.

Imaginons le scénario suivant: nous disposons d'un grand nombre de couples texte clair - texte chiffré avec une clé identique. Nous pouvons dans ce cas procéder à ce que l'on nomme une *attaque à texte clair connu*.

Ainsi, il est possible d'exploiter une faiblesse d'une des briques composantes de DES en effectuant des statistiques sur un flot de 2^{43} couples de données (soit la bagatelle de deux fois 64 To).

DES – attaques sur le DES

- Cryptanalyse linéaire ou différentielle
  - Attaque impraticable pour des chiffrements correctement conçus
  - Outil parfait pour comparer la résistance de divers chiffrements
  - Résistance contre ces attaques ne signifie pas résistance contre toutes les autres méthodes inconnues.

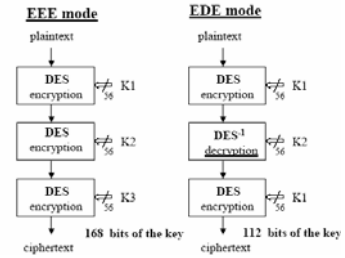
2DES – 3DES

- Double DES
  - Choisir deux clefs k_1 et k_2
  - Encrypter deux fois: $E(k_2, E(k_1, m))$
  - Il a été prouvé que 2DES était équivalent à un DES avec clé de 57 bits $\rightarrow$ seulement deux fois plus de travail pour briser
- Attaque sur le 2DES (meet-in-the-middle)
  - Soit des paires (M_i,C_i)
  - On calcule $X_z = E_{Kz}(M_i)\ \forall\ z$ et $Y_W = D_{Kw}(C_i)\ \forall\ w$
  - On cherche z = w
  - Donc $X=2^n$ opérations, $Y=2^n$ opération $\Rightarrow$ attaque = 2^{n+1}

Requiert 10^{17} bytes pour stocker ces blocs

2DES – 3DES

- Triple DES
  - 2 clefs, 3 opérations:
    - $E(k_1, D(k_2, E(k_1, m)))$
  - Équivalent à doubler la taille effective de la clé (longueur sure)
  - Très robuste et effectif contre toutes les attaques faisables connues
  - Très lent car triple les opérations



Sources

- [stallings] – ch3
- [schneier] – ch 12
- [natkin] – 2.2
- http://www.bibmath.net/crypto/moderne/indexmoderne.php3
- http://www.ssh.fi/support/cryptography/algorithms/symmetric.html
- http://home.ecn.ab.ca/~jsavard/crypto/comp04.htm
- http://www.uqtr.ca/~delisle/Crypto/prives/

Questions

- Expliquer :
  - Réseaux de Shannon – Feistel
  - Algorithme du DES
    - Permutation, calcul médian
    - Calcul de la clé
  - Faiblesses et améliorations du DES