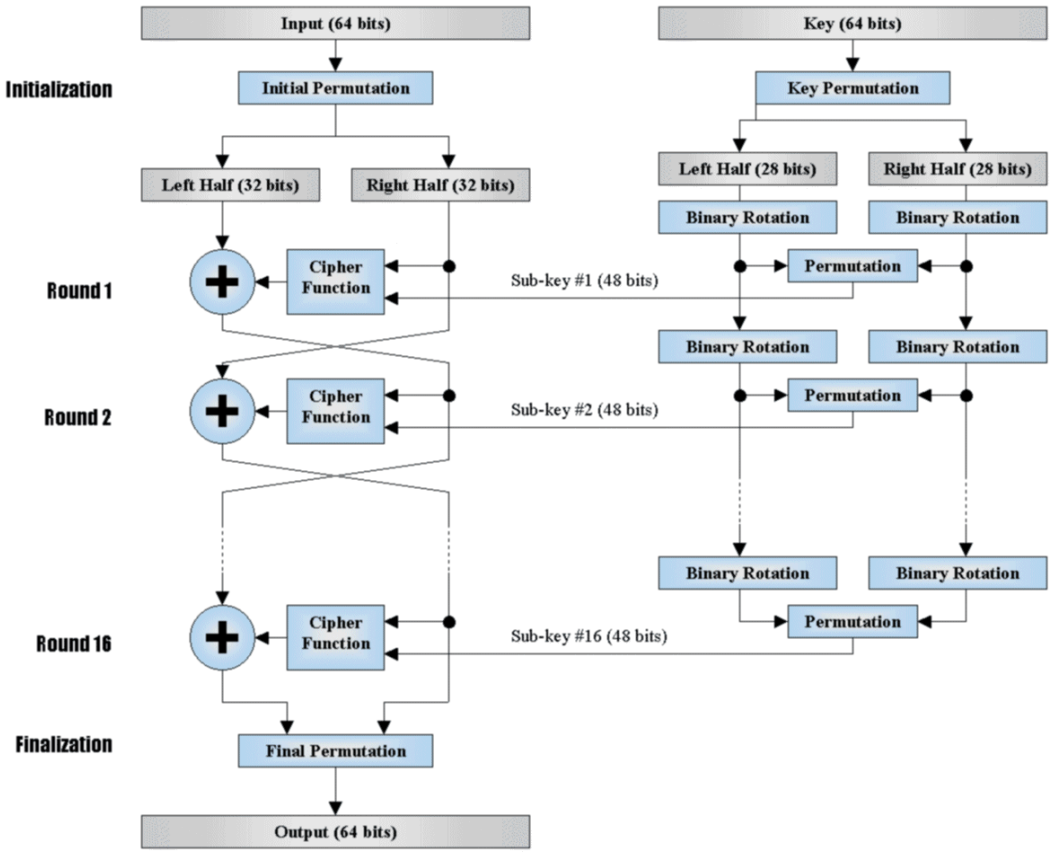
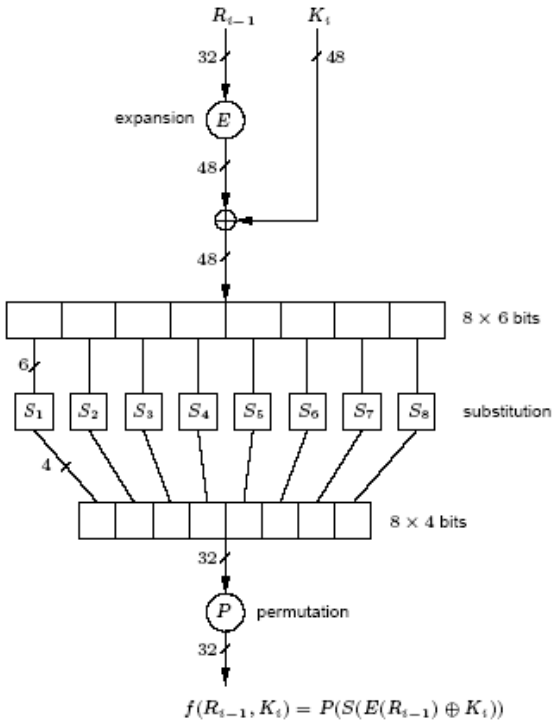
Data Encryption Standard (**DES**)

- DES a été l'algorithme cryptographique le plus important de ces derniers 30 ans.
- DES est un *Feistel Cipher* avec des blocs de 64 bits (taille nominale).
- La taille effective de la clé est de 56 bits (Un total de 64 bits avec 8 bits de parité).
- L'algorithme est constitué de **16** étapes avec **16** sous-clés de 48 bits générées (une clé par étape).
- DES fonctionne dans les 4 modes: ECB, CBC, CFB et OFB.
- Le schéma de fonctionnement de DES est le suivant:

DES: Schéma de Fonctionnement



DES: *Cipher Function*



(Source [Men97])

Figure 7.10: *DES inner function f.*

DES: Tables

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

Table 7.2: *DES initial permutation and inverse (IP and IP^{-1}).*

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

(Source [Men97])

Table 7.3: *DES per-round functions: expansion E and permutation P.*

DES: *S-boxes*

S-Box 1: Substitution Box 1

Row / Column	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

S-Box 2: Substitution Box 2

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

DES: Fonctionnement

Cipher Fonction

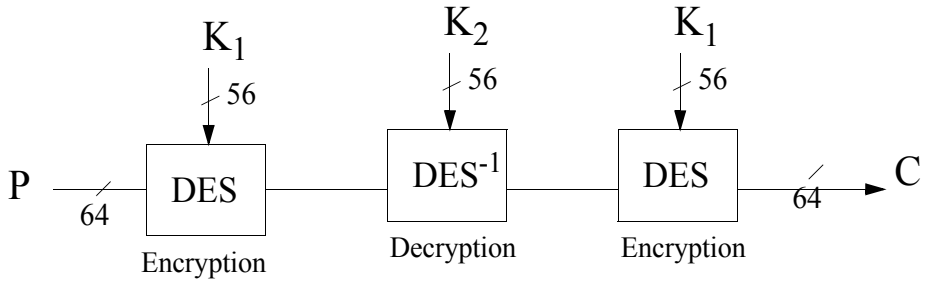
- **Expansion E**: Les 32 bits de l'entrée sont transformés en un vecteur de 48 bits en utilisant la table **E** (page 60). La première ligne de cette table indique comment sera généré le premier sous-bloc de 6 bits: on prendra en premier le 32^{e} bit et après les bits 1,2,3,4,5. Le deuxième sous-bloc commence par le 4^{e} bit ensuite les bits 5,6,7,8,9 et ainsi de suite...
- ***Key addition***: XOR du vecteur de 48 bits avec la clé.
- **S-boxes**: On applique **8 S-boxes** sur le vecteur de 48 bits résultant du XOR précédent. Chacune de ces S-boxes prend un sous-bloc de 6 bits et le transforme en un sous-bloc de 4 bits. Les S-boxes 1 et 2 sont présentées en page 61. L'opération s'effectue de la manière suivante: Si on dénote les 6 bits d'input de la S-box comme: $a_1a_2a_3a_4a_5a_6$. La sortie est donnée par le contenu de la cellule située dans la ligne $a_1 + 2a_6$ et la colonne $a_2 + 2a_3 + 4a_4 + 8a_5$.
- **Permutation P**: La permutation P (page 60) fonctionne comme suit: Le premier bit est envoyé à la 16^{e} position, le deuxième à la 7^{e} position et ainsi de suite.

Permutations IP et IP^{-1}

- Agissent respectivement au début et à la fin du traitement du bloc et sur l'ensemble des 64 bits (voir les tables en page 60 pour les détails).

DES et Triple-DES

- La taille de l'ensemble de clés ($\{0,1\}^{56}$) constitue la plus grande menace qui pèse sur DES avec les ressources de calcul actuels. En 1999 il a suffit de 24 heures pour trouver la clé à partir d'un *known plaintext* en utilisant une technique brute force massivement parallèle (100'000 PCs connectés sur Internet)[1].
- **Triple DES** nous met à l'abri de ces attaques *brute force* en augmentant l'espace des clés possibles à $\{0,1\}^{112}$. Schématiquement, il fonctionne de la manière suivante:



- Cette alternative permet de continuer à utiliser les "boîtes" DES (hardware et software) en attendant une migration vers AES.
- Le niveau de sécurité obtenue par cette solution est très satisfaisant.
- L'impact en termes de performances de trois exécutions successives de DES reste un inconvénient pour certaines applications.

1. http://www.rsasecurity.com/company/news/releases/pr.asp?doc_id=462

UNIVERSITÉ DE GENÈVE

DES: propriétés

- **DES n'est pas un groupe** (au sens algébrique) avec la composition: En d'autres termes, DES étant une permutation: $\{0,1\}^{64} \rightarrow \{0,1\}^{64}$, si DES était un groupe pour la composition, ceci voudrait dire que:

  $$\forall (K_1,K_2), \exists K_3 \text{ t.q. } E_{K_2}(E_{K_1}(x)) = E_{K_3}(x)$$

  Cette propriété permet d'assurer que l'encryption composée (comme *Triple-DES*) augmente considérablement la sécurité de DES. Si DES était un groupe, la recherche exhaustive sur l'ensemble de clés possibles ($\{0,1\}^{56}$) permettrait de "casser" l'algorithme indépendamment du nombre d'exécutions consécutives de DES.
- **Clés faibles et mi-faibles** (***weak*** *and* ***semi-weak*** *keys*):
  - Une clé **K** est dite **faible** si $\mathbf{E_K(E_K(x)) = x}$.
  - Une paire de clés $\mathbf{(K_1, K_2)}$ est dite **mi-faible** si $\mathbf{E_{K_1}(E_{K_2}(x)) = x}$.
- Les clés faibles ont la particularité de générer de sous-clés identiques par paires ($k_1 = k_{16}$, $k_2 = k_{15}$,..., $k_8 = k_9$), ce qui facilite la cryptanalyse.
- DES a 4 clés faibles (et 6 paires de clés mi-faibles):

Clés faibles pour DES:

```
0101 0101 0101 0101
0101 0101 FEFE FEFE
FEFE FEFE FEFE FEFE
FEFE FEFE 0101 0101
```

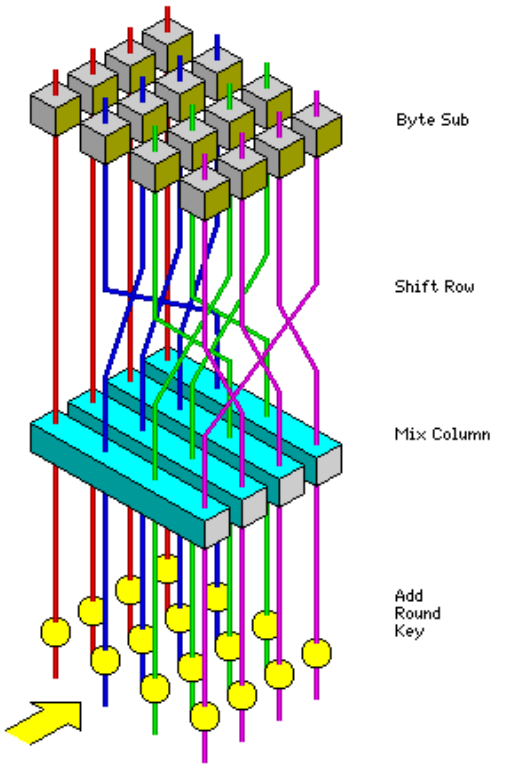

UNIVERSITÉ DE GENÈVE

Advanced Encryption Standard (AES)

- Adopté comme standard en Novembre 2001[1], conçu par *Johan Daemen* et *Vincent Rijmen*[2] *(*d'où son nom original *Rijndael).*
- Il s'agit également d'un *block cipher itératif* (comme DES) mais pas d'un *Feistel Cipher*
  - Blocs *Plaintext/Ciphertext*: 128 bits.
  - Clé de longueur variable: 128, 192, ou 256 bits.
- Contrairement à DES, AES est issu d'un processus de consultation et d'analyse ouvert à des experts mondiaux.
- Techniques semblables à DES (substitutions, permutations, XOR…) complémentées par des opérations algébriques simples et très performantes.
- Toutes les opérations s'effectuent dans le corps GF(2^8): le corps fini de polynômes de degré ≤ 7 avec des coefficients dans GF(2).
- En particulier, un byte pour AES est un élément dans GF(2^8) et les opérations sur les bytes (additions, multiplications,...) sont définies comme sur GF(2^8).
- ~2 fois plus performant (en software) et ~10^{22} fois (en théorie...) plus sûr que DES...
- Évolutif: La taille de la clé peut être augmentée si nécessaire.

1. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
2. http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael-ammended.pdf

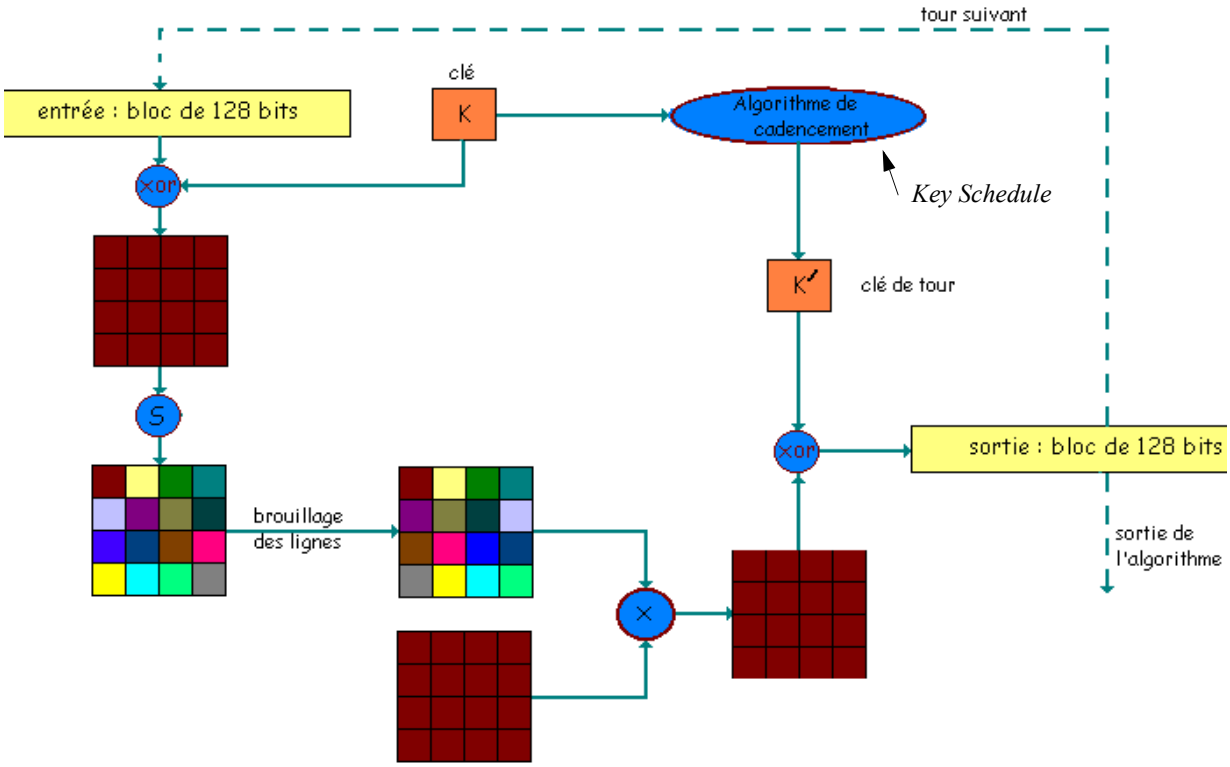
Détail d'une Etape (*round*) AES



L'unité de base sur laquelle s'appliquent les calculs est une matrice de 4 lignes et 4 colonnes (dans le cas d'une clé de 128 bits) dont les éléments sont des bytes:

- ***ByteSub***: Opération non linéaire (S-box) conçu pour résister à la cryptanalyse linéaire et différentielle.
- ***ShiftRow***: Permutation des bytes introduisant des décalages variables sur les lignes.
- ***MixColumn***: Chaque colonne est remplacée par des combinaisons linéaires des autres colonnes (multiplication des matrices!)
- ***AddRoundKey:*** XOR de la matrice courante avec la sous-clé correspondante à l'étape courante.

Détail d'une étape (*round*) AES (II)



Source: Un nouvel algorithme de chiffrement, par Joan Daemen et Vincent Rijmen. Pour la Science,Dossier l'art du secret.

AES: Fonctionnement Global

- Le nombre d'étapes d'AES varie en fonction de la taille de la clé. Pour une clé de 128 bits, il faut effectuer 10 étapes. Chaque augmentation de 32 bits sur la taille de la clé, entraîne une étape supplémentaire (14 étapes pour des clés de 256 bits).
- La decryption consiste en appliquer les opérations inverses dans chacune des étapes (*InvSubBytes*, *InvShiftRows*, *InvMixColumns*). *AddRoundKey* (à cause du XOR) est sa propre inverse.
- Le ***Key Schedule*** consiste en:
  - *Une opération d'expansion de la clé principal*. Si N_e est le nombre d'étapes (dépendant de la clé), une matrice de 4 lignes et 4 * (N_e + 1) colonnes est générée.
  - *Une opération de sélection de la clé d'étape:* La première sous-clé sera constituée des 4 premières colonnes de la matrice générée lors de l'expansion et ainsi de suite.
- *Pseudo-code pour AES*:

```
Rijndael(State,CipherKey)
{
    KeyExpansion(CipherKey,ExpandedKey);          // Key Schedule
    AddRoundKey(State,ExpandedKey[0..3]);         // Premier XOR avec la 1ère sous-clé
    For( i=1 ; i<Ne ; i++ ) Round(State,ExpandedKey[4*i ... (4*i)+3]);// Ne - 1 étapes
    FinalRound(State,ExpandedKey[4*Ne ... 4*Ne+3]).   // Dernière étape (pas de MixColumn)
}
```

AES: Remarques finales et Attaques

- La plus grande force de AES réside dans sa simplicité et dans ses performances, y compris sur des plate-formes à capacité de calcul réduite (p.ex. des cartes à puces avec des processeurs à 8 bits).
- Depuis sa publication officielle, des nombreux travaux de cryptanalyse ont été publiés avec des résultats très intéressants. En particulier, N. Courtois et P.Pieprzyk[1] ont présenté une technique appelée XSL permettant de représenter AES comme un système de 8000 équations quadratiques avec 1600 inconnues binaires. L'effort nécessaire pour casser ce système est estimé (il s'agit encore d'une conjecture...) à 2^{100}.
- Ces attaques se basent sur le caractère fortement algébrique (et largement contesté...) de AES. De plus, il suffit de quelques *known plaintexts* pour les mettre en place, ce qui les distingue des attaques linéaires et différentielles (page 70).
- Les prochaines années risquent d'apporter des développements très intéressants dans la cryptanalyse de AES et d'un bon nombre de *block ciphers* itératifs.
- Même si AES reste un algorithme robuste à ce jour, il convient de surveiller de près les travaux de recherche algébrique[2] qui pourraient révolutionner toutes les théories de conception actuelles relatives aux *block ciphers*.

1. Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. Asiacrypt 2002.
2. http://www.cryptosystem.net/aes/

Techniques de Cryptanalyse sur les *Block Ciphers*

Cryptanalyse Différentielle[1]

- Il s'agit d'une attaque *chosen plaintext* qui s'intéresse à la propagation des différences dans deux *plaintexts* au fur et à mesure qu'ils évoluent dans les différentes étapes de l'algorithme.
- Il attribue des probabilités aux clés qu'il "devine" en fonction des changements qu'elles induisent sur les ciphertexts. La clé la plus probable a des bonnes chances d'être la clé correcte après un grand nombre de couples *plaintext/ciphertext*.
- Nécessite 2^{47} couples *chosen plaintext* pour obtenir des résultats corrects.

Cryptanalyse Linéaire[2]

- Il s'agit d'une attaque *known plaintext* qui crée un simulateur du bloc à partir des approximations linéaires. En analysant un grand nombre de paires *plaintext/ciphertexts*, les bits de la clé du simulateur ont tendance à coïncider avec ceux du *block cipher* analysés (calcul probabiliste)
- Nécessite 2^{38} *known plaintexts* pour obtenir une probabilité de 10% de deviner juste et 2^{43} pour un 85%!
- Il s'agit de l'attaque analytique la plus puissante à ce jour sur les *block ciphers*.

1. Biham, E. and A. Shamir. (1990). Differential Cryptanalysis of DES-like Cryptosystems. Advances in Cryptology — CRYPTO '90. Springer-Verlag. 2–21.
2. Mitsuru Matsui: Linear Cryptoanalysis Method for DES Cipher. EUROCRYPT 1993: 386-397.

Techniques de Cryptanalyse sur les *Block Ciphers* (II)

- La mise en pratique des attaques différentielles et linéaires présente des difficultés dans la parallélisation des calculs par rapport à une recherche exhaustive de la clé.
- Ces deux attaques sont très sensibles au nombre d'étapes du *block cipher*: les chances de réussite augmentent exponentiellement au fur et à mesure que le nombre d'étapes de l'algorithme diminue.
- Une conjecture très répandue parmi les cryptographes est que ces attaques, à l'époque inédites, étaient connues par les concepteurs des DES. En particulier, le design des S-boxes offre une résistance très grande aux deux techniques.

Attaque Meet-in-the-Middle

- S'applique aux constructions du type $E_{K_2}(E_{K_1}(x))$. L'espace de clés pour cette solution est de $\{0,1\}^{112}$. On construit d'abord deux listes L_1 et L_2 de 2^{56} messages de la forme: $L_1 = E_{K_1}(x)$ et $L_2 = D_{K_2}(x)$ avec E et D les opérations d'encryption et decryption respectivement. Il suffit alors d'identifier un élément qui se répète dans les deux listes et les K_1 et K_2 associées à cet élément seront (en toute vraisemblance) les clés recherchées!
- Effort nécessaire à réaliser les attaques: 2^{57} opérations pour établir les deux listes + 2^{56} blocs de 64 bits de stockage pour mémoriser les résultats intermédiaires... nettement inférieur au 2^{112} estimé intuitivement...
- Ces techniques *meet-in-the-middle* sont aussi appliquées à la cryptanalyse interne des *block ciphers*.