Chiffrements Symétriques

Pascal Lafourcade

UNIVERSITÉ Clermont Auvergne

LIMOS

2022-2023

Chiffrement Symétrique

Exemples

- Chiffrement par bloc (taille fixe des messages) : DES AES
- Chiffrement par flots (stream cipher) (taille illimité des messages) RC4, FISH, ChaCha, Salsa20, A5/1

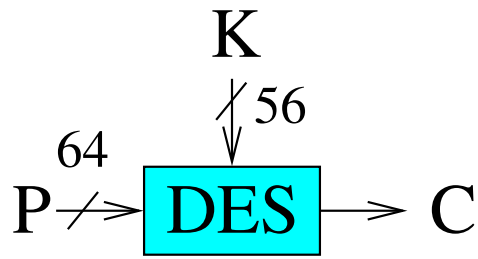
Plan

Plan

Data Encryption Standard (DES), (Appel du NIST de 1973)

Lucifer conçu en 1971 par Horst Feistel (IBM).

- Chiffrement par block de 64-bit avec une clé de 56 bits.



- Premier standard cryptographique
  - 1977 US federal standard (US Bureau of Standards)
  - 1981 ANSI

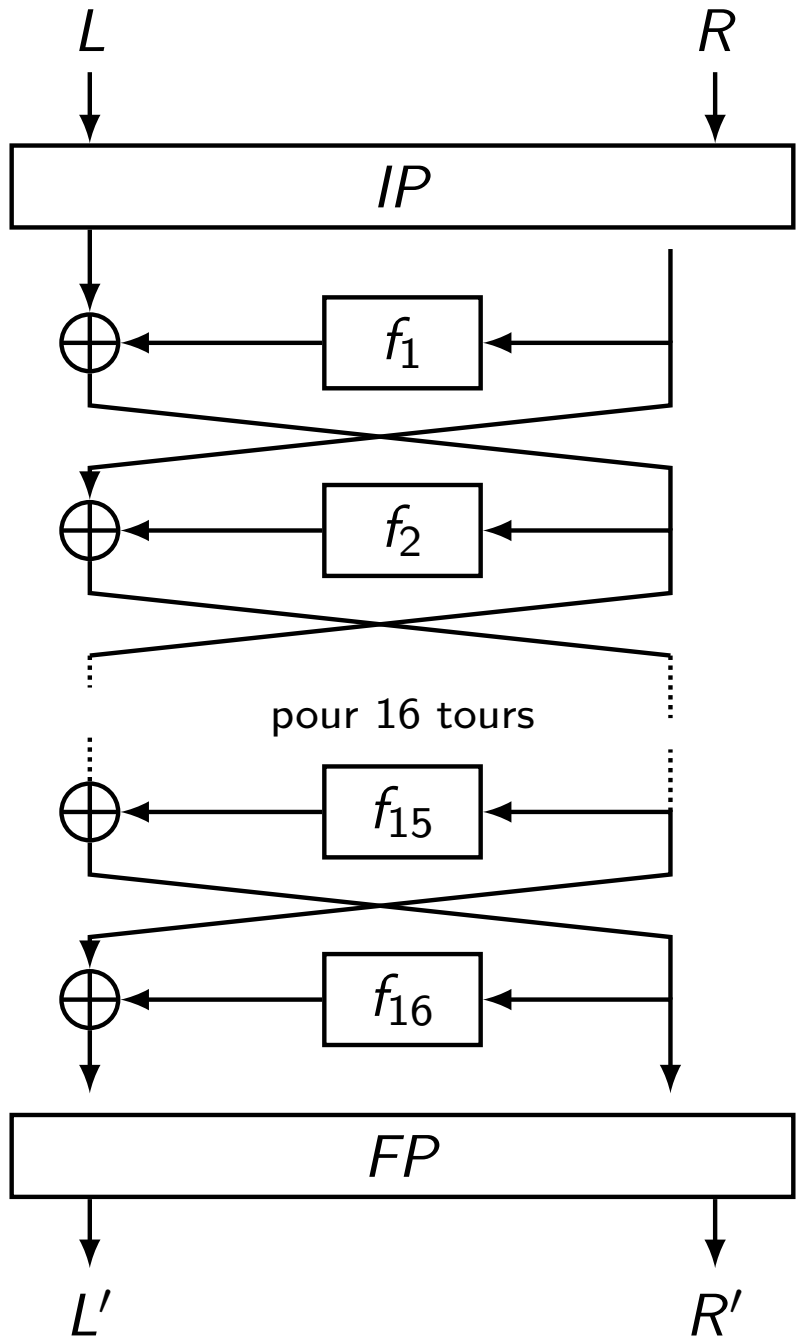
DES — Principes

- Diviser le message en deux parties de 32 bits : la droite (R_i) et la gauche (L_i)
- 16 tours de Feistel
- Une dérivation de 16 sous-clés K_i à partir de K
- Une permutation initiale et son inverse pour finir
- Une fonction f constituée de 2 permutations et d'une S-box (une substitution)

$$L_{i+1} = R_i$$

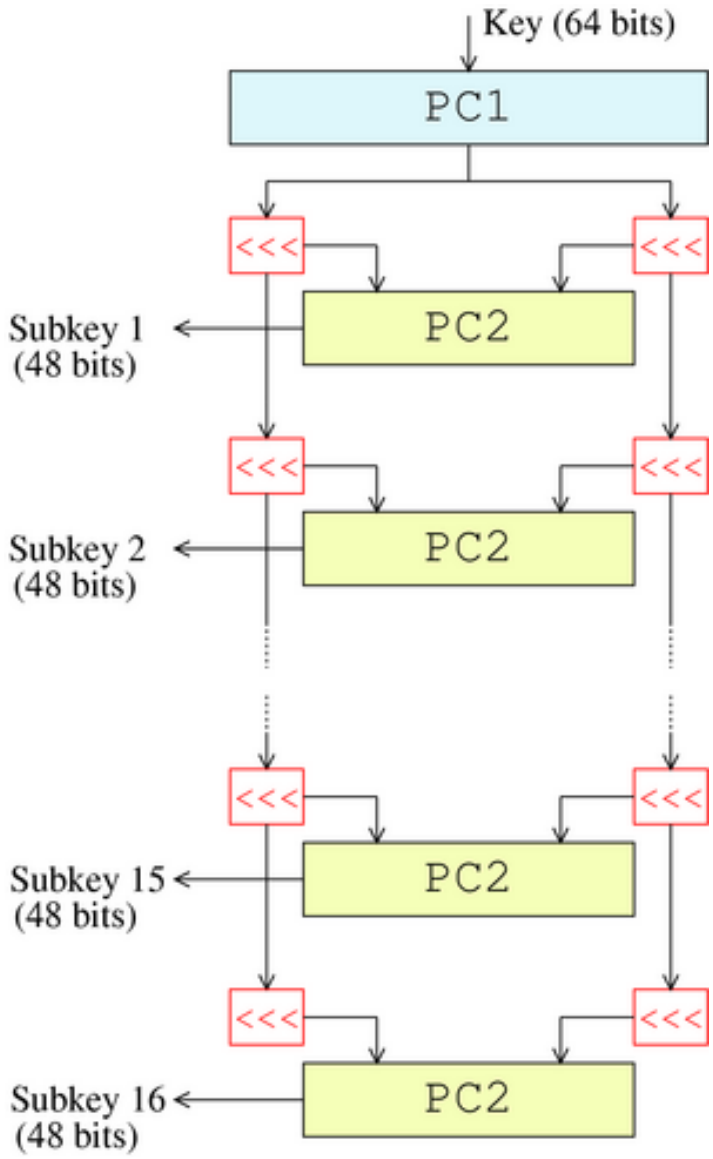
$$R_{i+1} = L_i \oplus f(R_i, K_i)$$

DES, vue d'ensemble



DES — Génération des sous-clé

Les bits, 8, 16,24, 32, 40, 48, 56 et 64 sont des bits de parité.



DES — Génération des sous-clé

Permutation Choice 1 (PC1) (8*7 = 56 = 28 + 28) :

Left

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

Right

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

Permutation Choice 2 (PC-2) (8*6 = 48)

8 bits non utilisés : 9, 18, 22, 25, 35, 38, 43, et 54

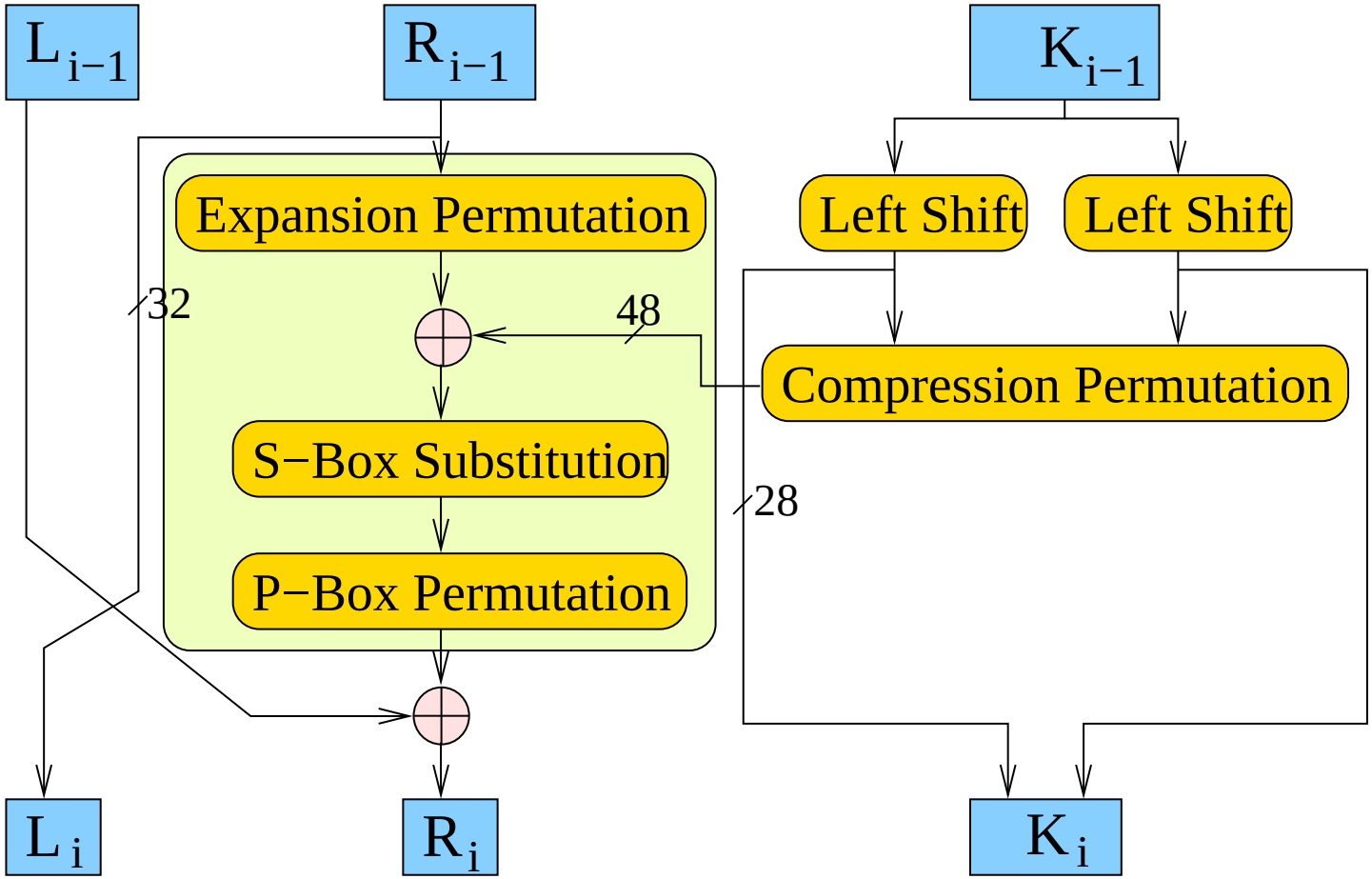
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

DES — Avant chaque tour de génération de sous-clé

Chaque moitié de la clé est décalée à gauche d'un nombre de places en fonction du tour.

# Rds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Left	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES — 1 tour



$(b_1 b_6, b_2 b_3 b_4 b_5)$, $b_1 b_6$ represente la valeurs de la ligne et $b_2 b_3 b_4 b_5$ la valeur de la colonne de la S-Box S_j.

S-Boxes: S1, S2, S3, S4

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

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

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

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

S-Boxes: S5, S6, S7 and S8

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

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

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

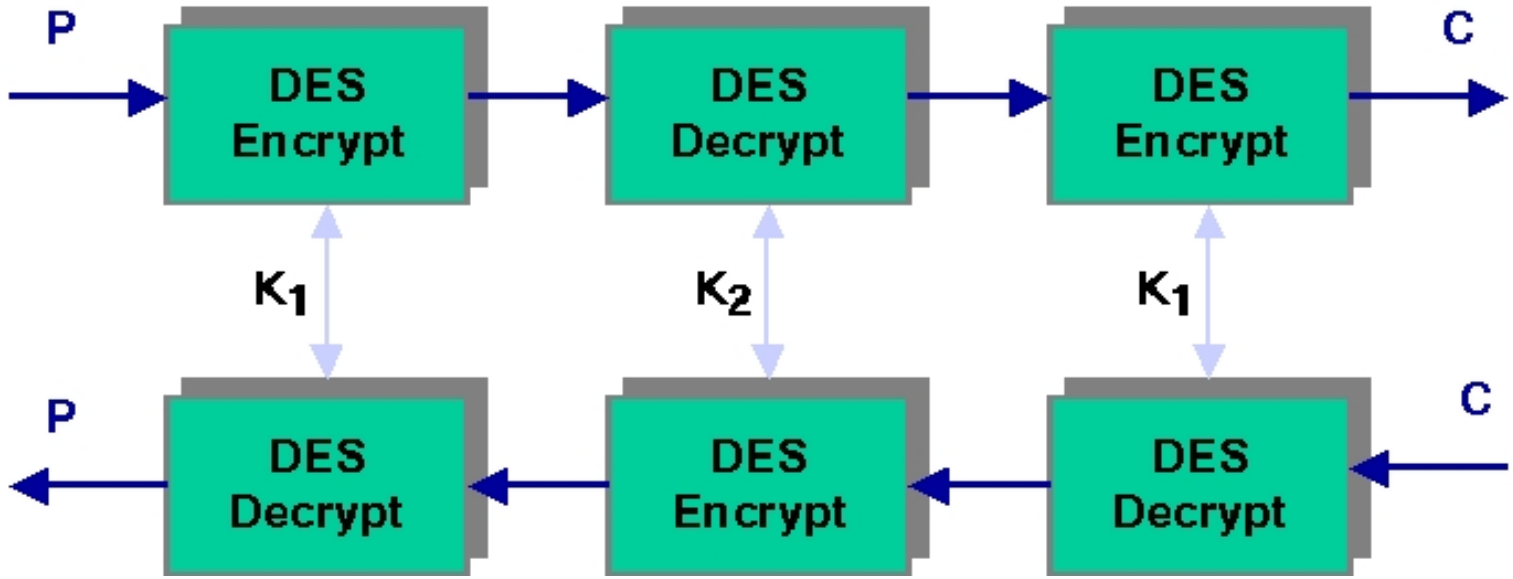
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Permutation P

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

Triple DES

- Utiliser 3 chiffrements consécutifs.



- Compatibilité avec DES si $(K_2 = K_1)$.
- Brute-force attaque avec 2^{112} opérations.

Advanced Encryption Standard (2002)

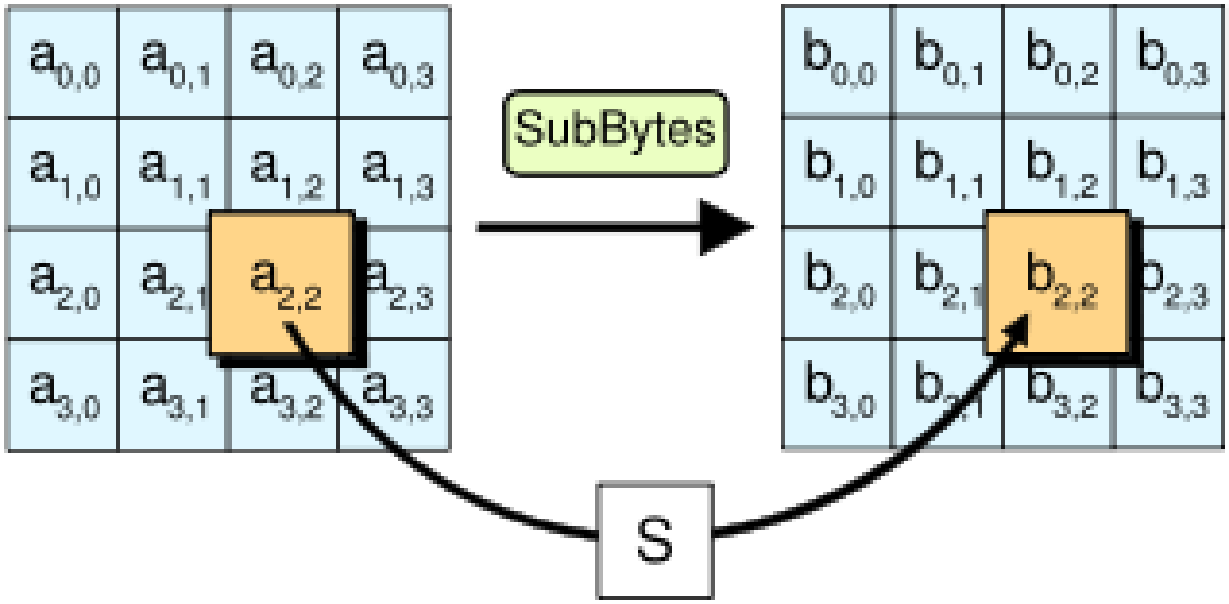
- Rijndael Conçu par deux cryptographes Belges Joan Daemen et Vincent Rijmen
- Block de 128 bits, clé de 128, 192, ou 256 bits.

Taille de la clé	Nombre de tours
128	10
192	12
256	14

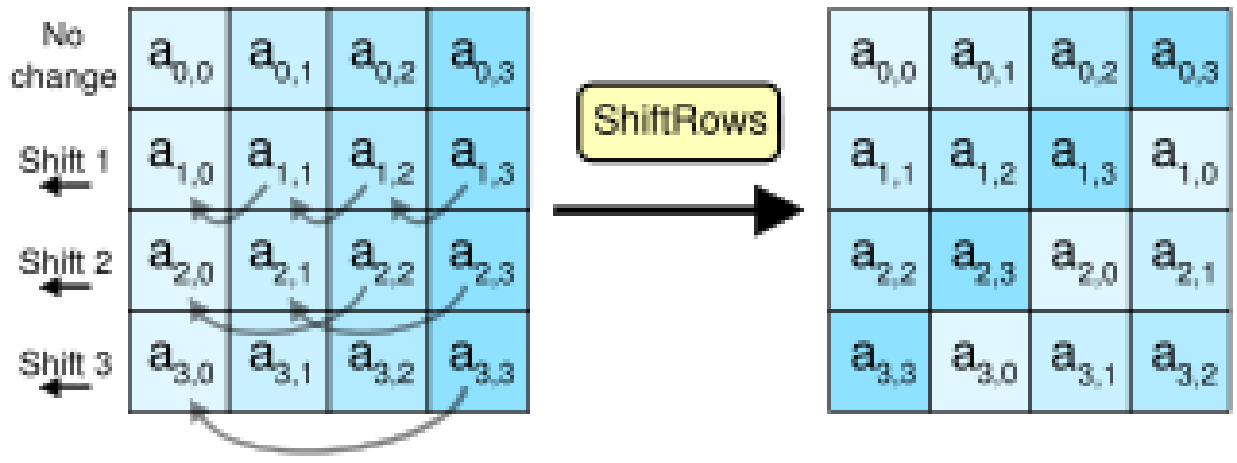
AES:Chiffrement

- Dérivation de la clé
- Tour initial : AddRoundKey
- Tours :
  1. SubBytes
  2. ShiftRows
  3. MixColumns
  4. AddRoundKey
- Tour final :
  1. SubBytes
  2. ShiftRows
  3. AddRoundKey

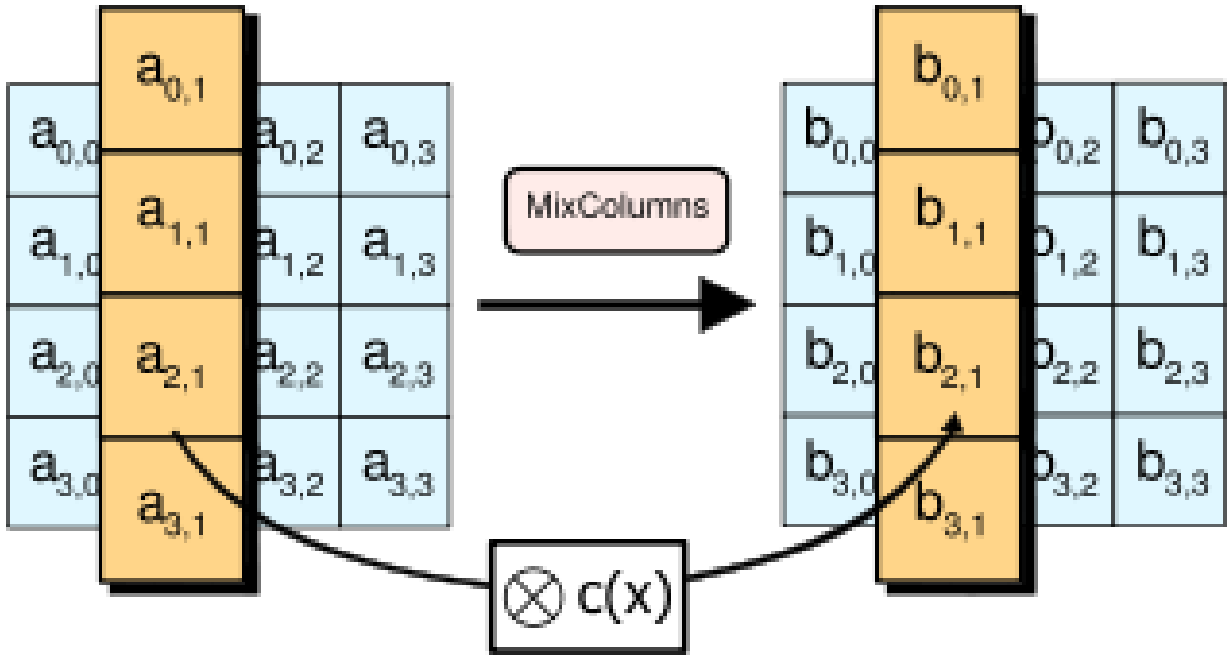
AES: SubBytes



AES: ShiftRows

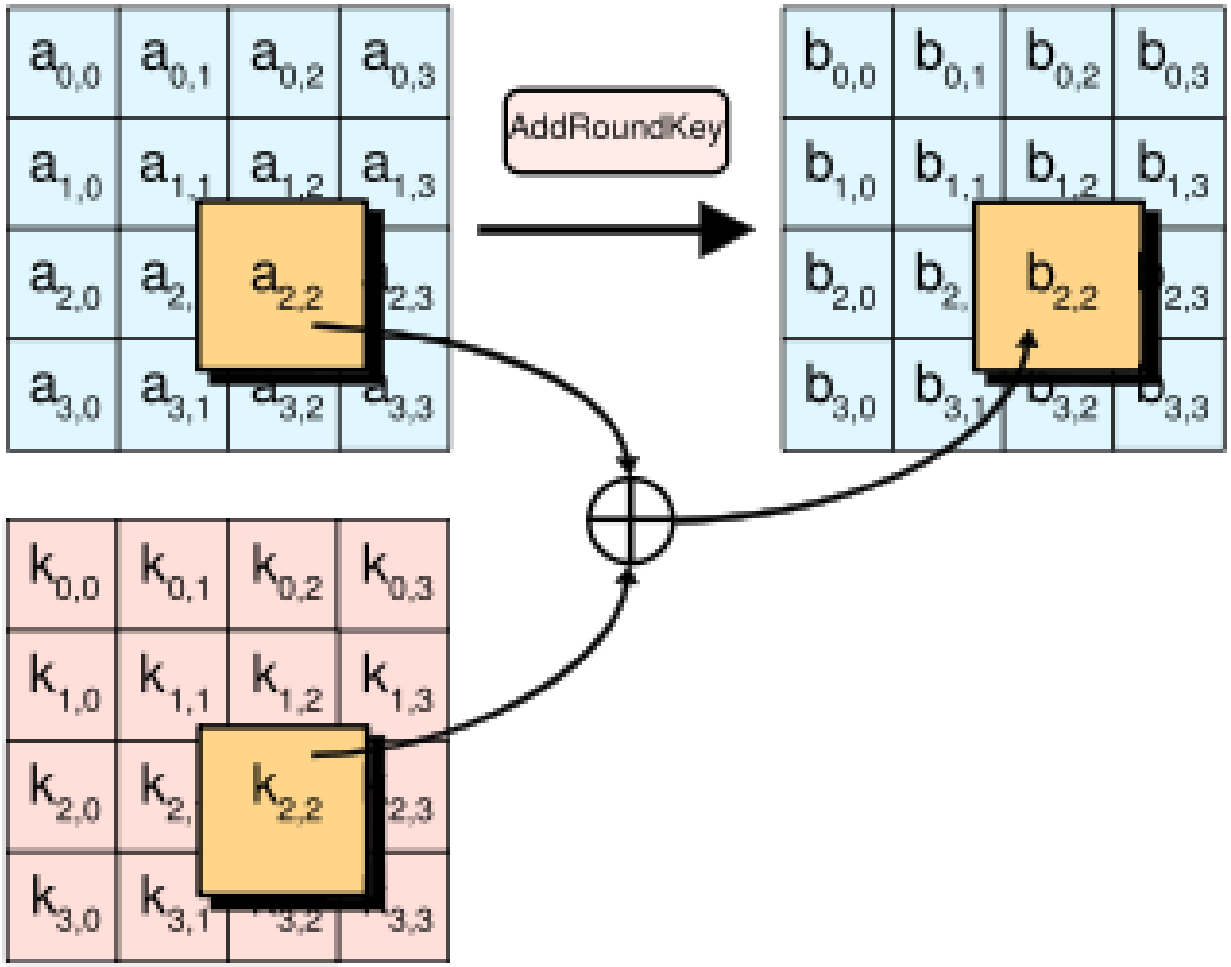


AES: MixColumns



$$C(x) = 3x^3 + x^2 + x + 2 Modulo : x^4 + 1$$

AES: AddRoundKey



Plan

IDEA: International Data Encryption Algorithm 1991

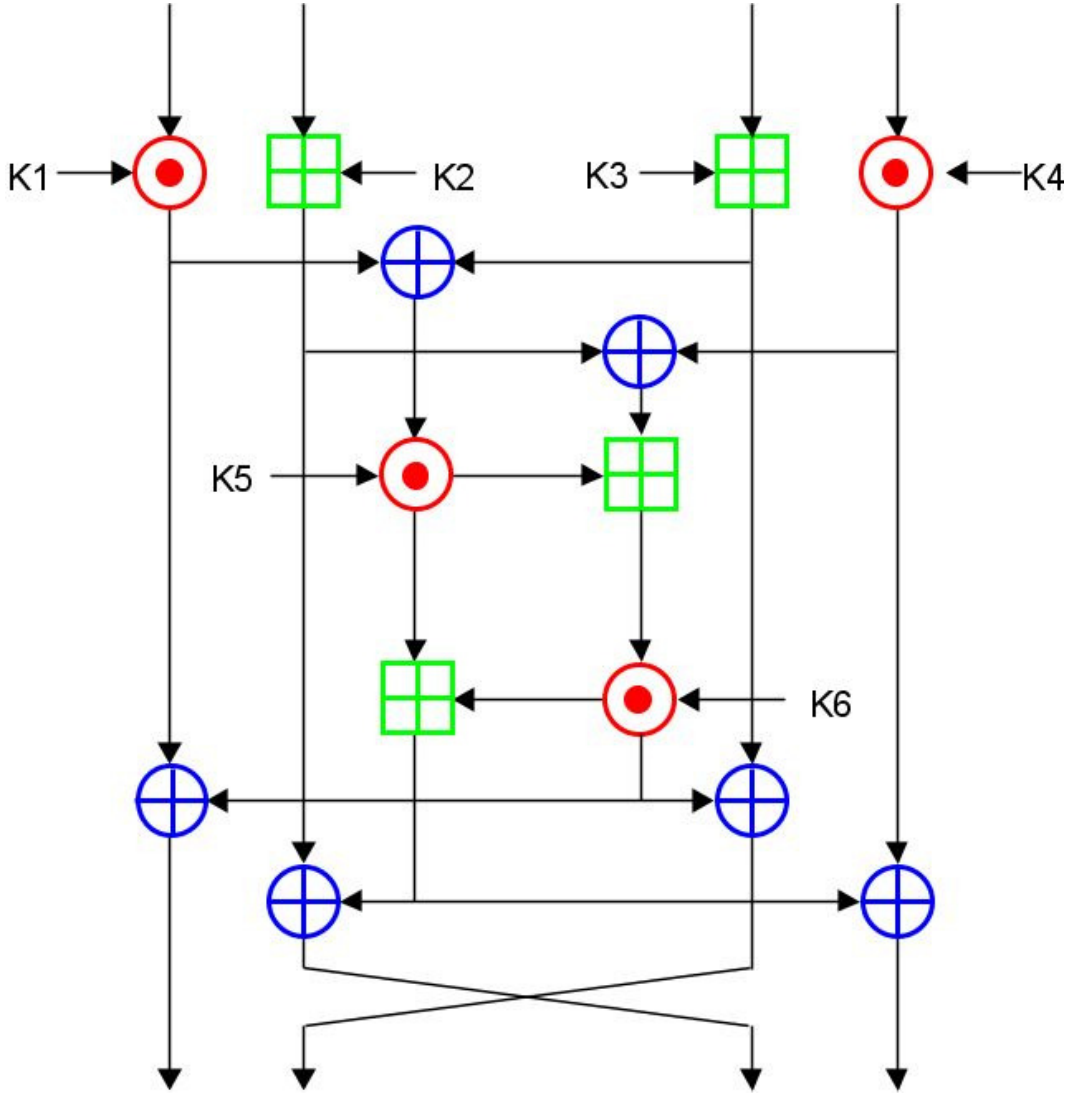
Conçu par Xuejia Lai et James Massey de l'ETH Zurich, utilisé dans Pretty Good Privacy (PGP) v2.0
IDEA utilise :

- un message de 64-bit, coupé en 4 blocs de 16 bits
- une clé de 128-bit et 8.5 tours.

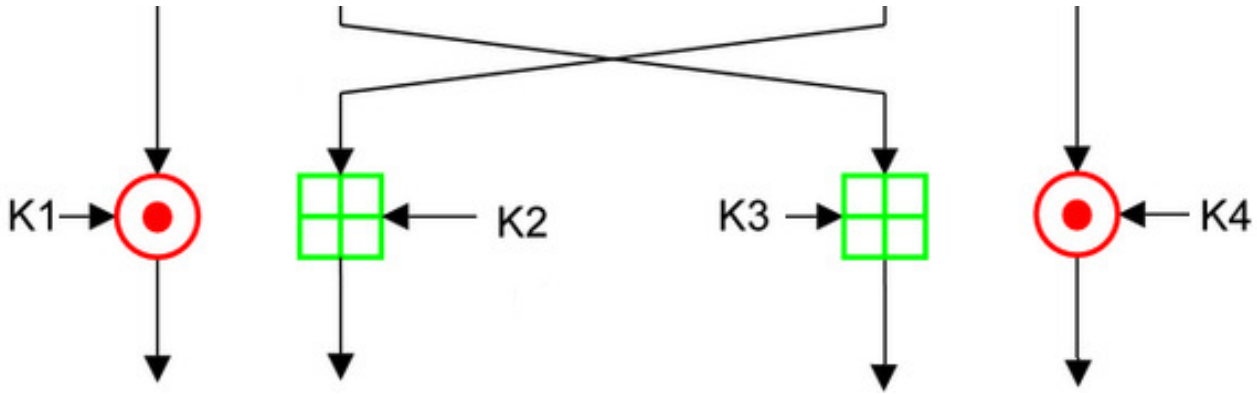
Notation

- Ou eXclusive XOR (dénoté par $\oplus$).
- Addition modulo 2^{16} (dénoté par$\boxplus$).
- Multiplication modulo $2^{16} + 1$, (dénoté par $\odot$).

IDEA



IDEA demi-tour final



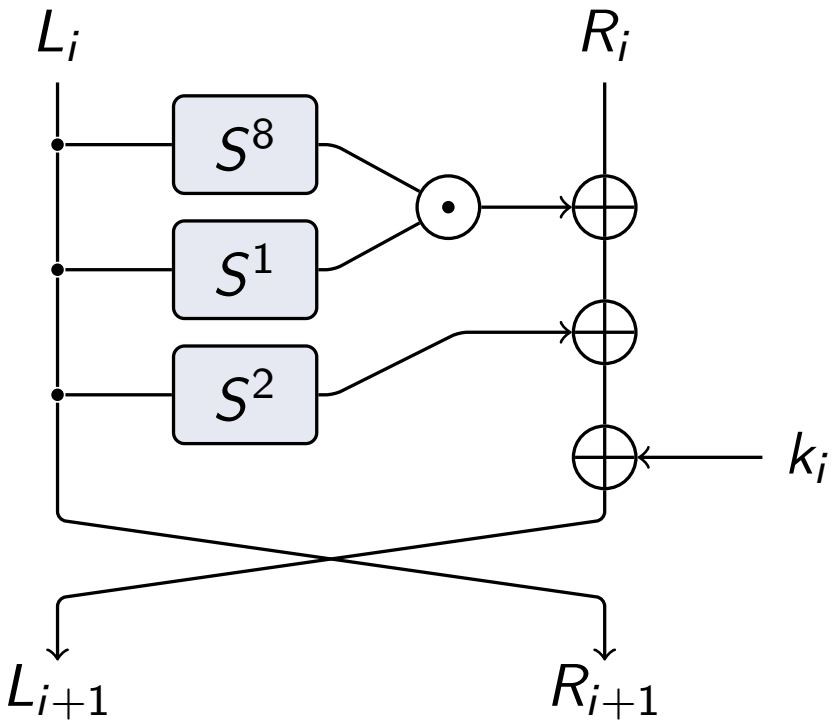
La meilleur attaque connue est sur 6 tours par Biham, E. and Dunkelman, O. and Keller, N. "A New Attack on 6-Round IDEA", 2007.

SIMON

Proposé par la NSA en Juin 2013.

Block (bits)	Clé (bits)	Tours
32	64	32
48	72	36
	96	36
64	96	42
	128	44
96	96	52
	144	54
128	128	68
	192	69
	256	72

Un tour de SIMON



S^i = décalage gauche de i bits
AND bit à bit $\odot$ et XOR $\oplus$.

SIMON key schedule, $c = 2^n - 4$

$$k_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus k_i \oplus \left(I \oplus S^{-1}\right)\left(S^{-3} k_{i+1}\right), & m = 2 \\ c \oplus (z_j)_i \oplus k_i \oplus \left(I \oplus S^{-1}\right)\left(S^{-3} k_{i+2}\right), & m = 3 \\ c \oplus (z_j)_i \oplus k_i \oplus \left(I \oplus S^{-1}\right)\left(S^{-3} k_{i+3} \oplus k_{i+1}\right), & m = 4 \end{cases}$$

$(z_j)_i$ est défini par une séquence de bits pour chaque paramètre.

D'autres chiffrements symétriques

Blowfish, Serpent, Twofish, 3-Way, ABC, Akelarre, Anubis, ARIA, BaseKing, BassOmatic, BATON, BEAR and LION, C2, Camellia, CAST-128, CAST-256, CIKS-1, CIPHERUNICORN-A, CIPHERUNICORN-E, CLEFIA, CMEA, Cobra, COCONUT98, Crab, CRYPTON, CS-Cipher, DEAL, DES-X, DFC, E2, FEAL, FEA-M, FROG, G-DES, GOST, Grand Cru, Hasty Pudding Cipher, Hierocrypt, ICE, IDEA, IDEA NXT, Intel Cascade Cipher, Iraqi, KASUMI, KeeLoq, KHAZAD, Khufu and Khafre, KN-Cipher, Ladder-DES, Libelle, LOKI97, LOKI89/91, Lucifer, M6, M8, MacGuffin, Madryga, MAGENTA, MARS, Mercy, MESH, MISTY1, MMB, MULTI2, MultiSwap, New Data Seal, NewDES, Nimbus, NOEKEON, NUSH, Q, RC2, RC5, RC6, REDOC, Red Pike, S-1, SAFER, SAVILLE, SC2000, SEED, SHACAL, SHARK, Skipjack, SMS4, Spectr-H64, Square, SXAL/MBAL, TEA, Treyfer, UES, Xenon, xmx, XTEA, XXTEA, Zodiac.

Plan

Meet-in-the-middle Attack

Double DES avec k_1 et k_2

$C = ENC_{k_2}(ENC_{k_1}(P))$

$P = DEC_{k_1}(DEC_{k_2}(C))$

Brute force attaque : $2^{k1} * 2^{k2} = 2^{k1+k2}$

Si $k = |k_1| = |k_2|$ alors 2^{2k}

Meet-in-the-middle Attack

Double DES avec k_1 et k_2

$C = ENC_{k_2}(ENC_{k_1}(P))$

$P = DEC_{k_1}(DEC_{k_2}(C))$

Brute force attaque : $2^{k1} * 2^{k2} = 2^{k1+k2}$

Si $k = |k_1| = |k_2|$ alors 2^{2k}

Observation

$$DEC_{k_2}(C) = DEC_{k_2}(ENC_{k_2}[ENC_{k_1}(P)])$$
$$= ENC_{k_1}(P)$$

Meet-in-the-middle Attack

Double DES avec k_1 et k_2

$C = ENC_{k_2}(ENC_{k_1}(P))$
$P = DEC_{k_1}(DEC_{k_2}(C))$
Brute force attaque : $2^{k1} * 2^{k2} = 2^{k1+k2}$
Si $k = |k_1| = |k_2|$ alors 2^{2k}

Observation

$$DEC_{k_2}(C) = DEC_{k_2}(ENC_{k_2}[ENC_{k_1}(P)])$$
$$= ENC_{k_1}(P)$$

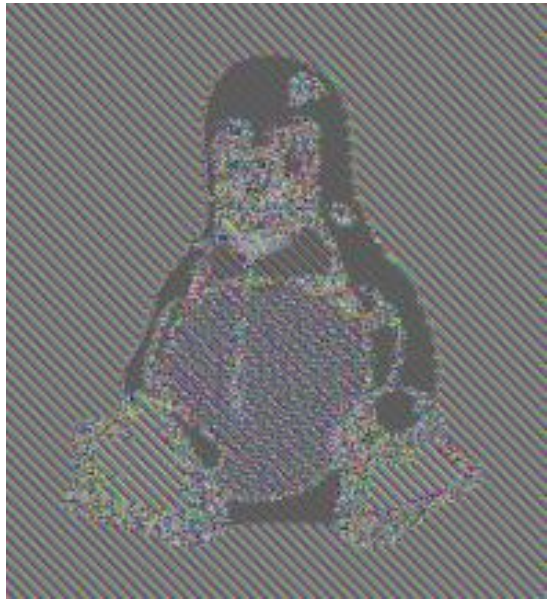
MITM Attack

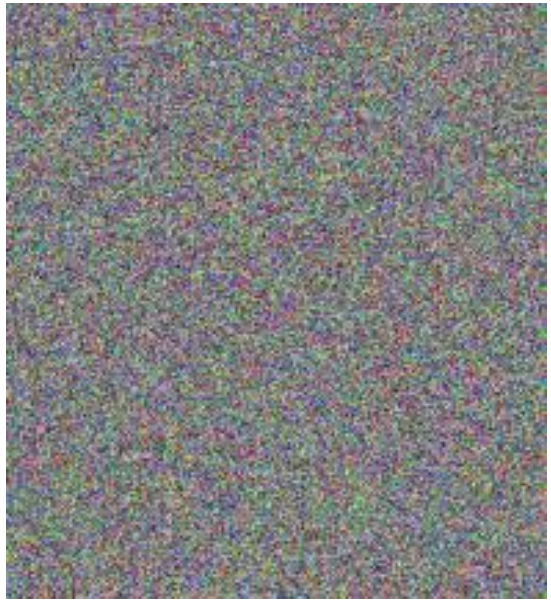
- $ENC_{k_1}(P)$ pour toutes les valeurs de k_1
- $DEC_{k_2}(C)$ pour toutes les valeurs de k_2,

Pour un total de $2^{|k_1|} + 2^{|k_2|}$.
Si $k = |k_1| = |k_2|$ alors 2^{k+1}

Plan

Chiffrer est-ce toujour sûr ?





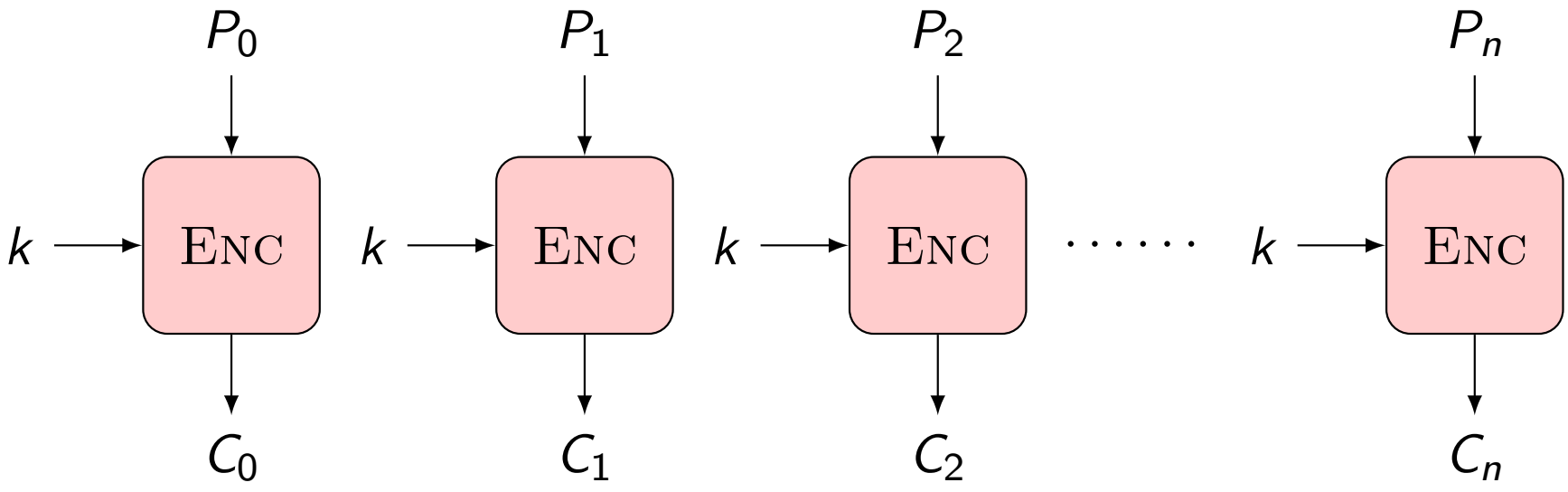
Electronic Book Code (ECB)

Chaque bloc de la même longueur est chiffré séparément en utilisant la même clé K.
Dans ce mode, seul le bloc dans lequel un bit est retourné a un contenu modifié. Les autres blocs ne sont pas affectés.

ECB Chiffrement Algorithm

algorithm $E_K(M)$
if ($|M|$ mod $n \neq 0$ or $|M| = 0$) then return FAIL
Break M into n-bit blocks $M[1] \ldots M[m]$
for $i = 1$ to m do $C[i] = E_K(M[i])$
$C = C[1] \ldots C[m]$
return C

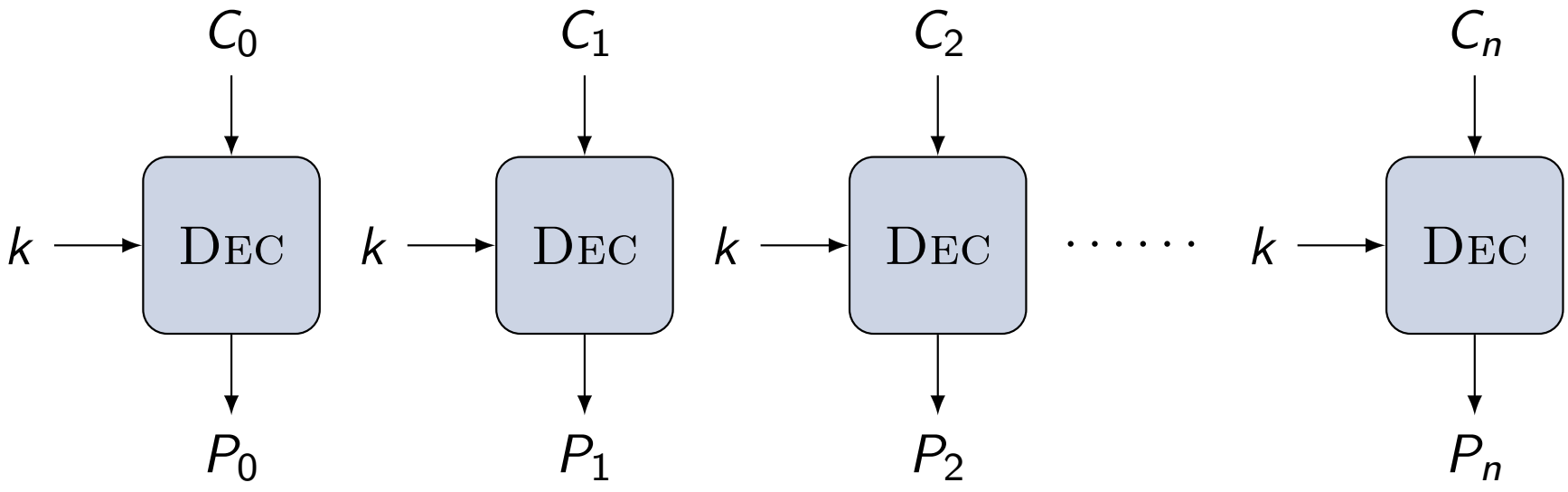
ECB Chiffrement



ECB Déchiffrement Algorithm

algorithm $D_K(C)$
if ($|C|$ mod $n \neq 0$ or $|C| = 0$) then return FAIL
Break C into n-bit blocks $C[1] \ldots C[m]$
for $i = 1$ to m do $M[i] = D_K(C[i])$
$M = M[1] \ldots M[m]$
return M

ECB Déchiffrement



Cipher-block chaining (CBC)

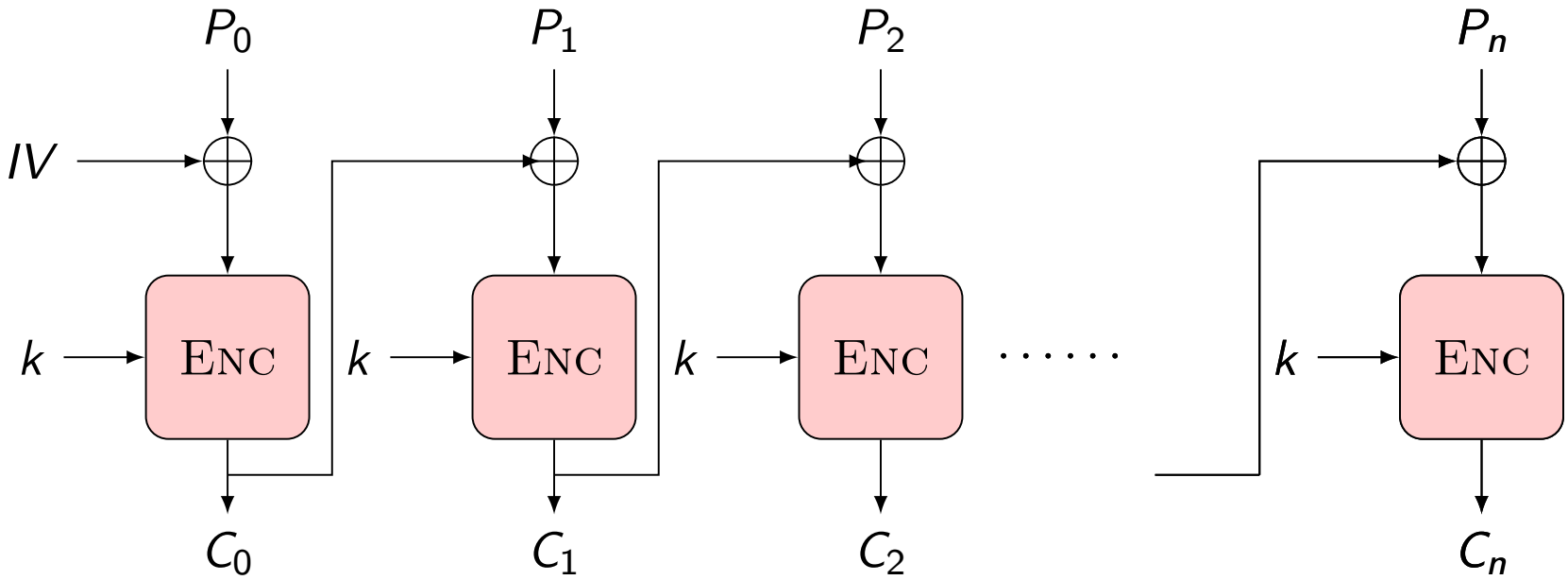
Chiffrement :

$$C_i = E_K(P_i \oplus C_{i-1}), C_0 = IV$$

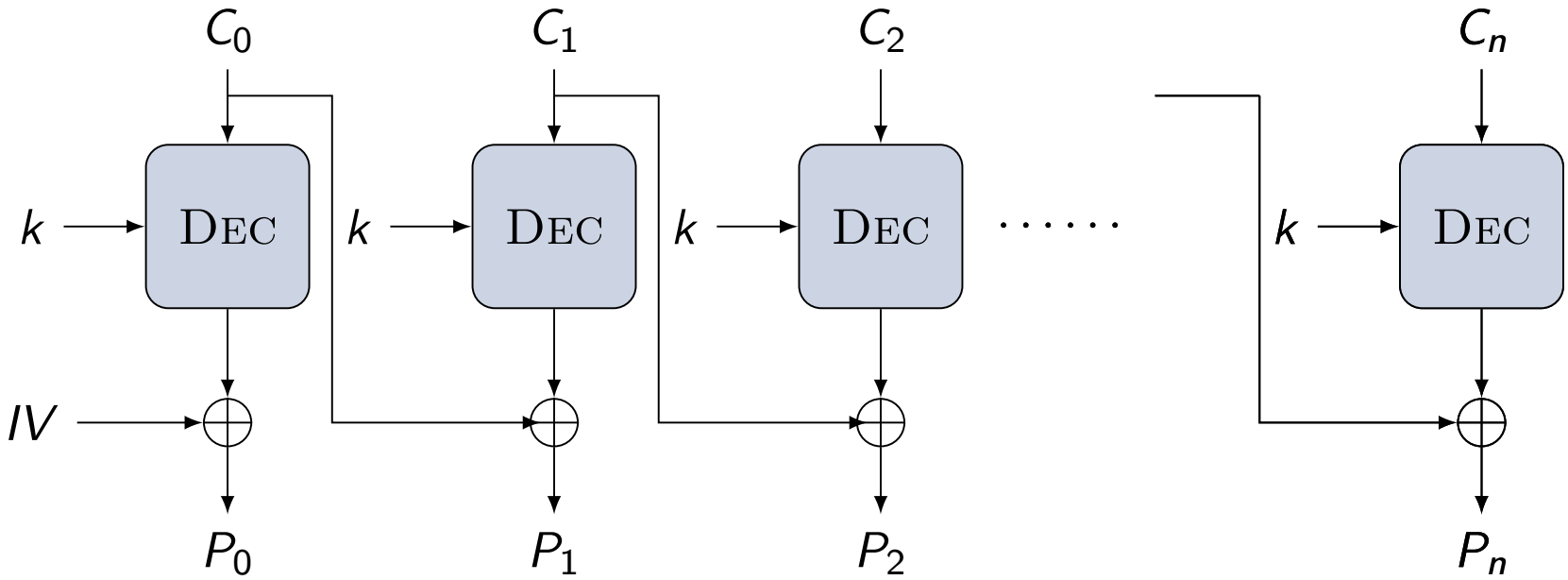
Déchiffrement:

$$P_i = D_K(C_i) \oplus C_{i-1}, C_0 = IV$$

CBC Chiffrement



CBC Déchiffrement



The cipher feedback (CFB)

Chiffrement :

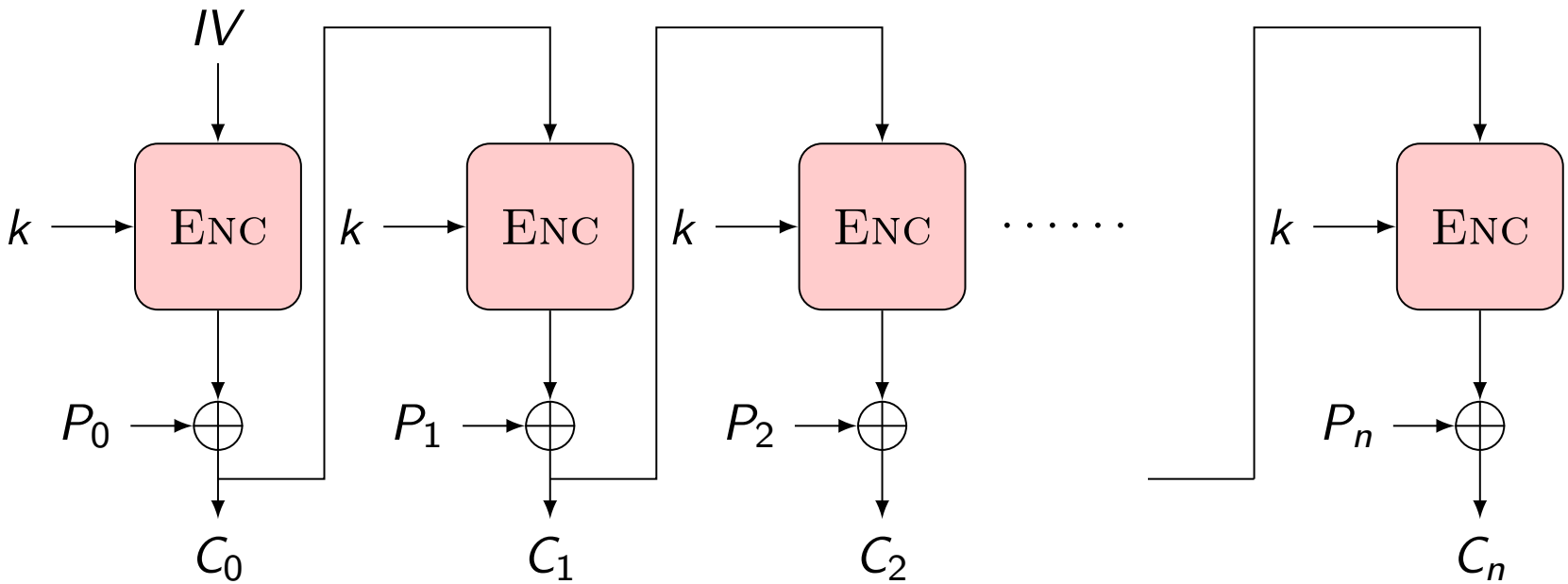
$$C_i = E_K(C_{i-1}) \oplus P_i$$

Déchiffrement :

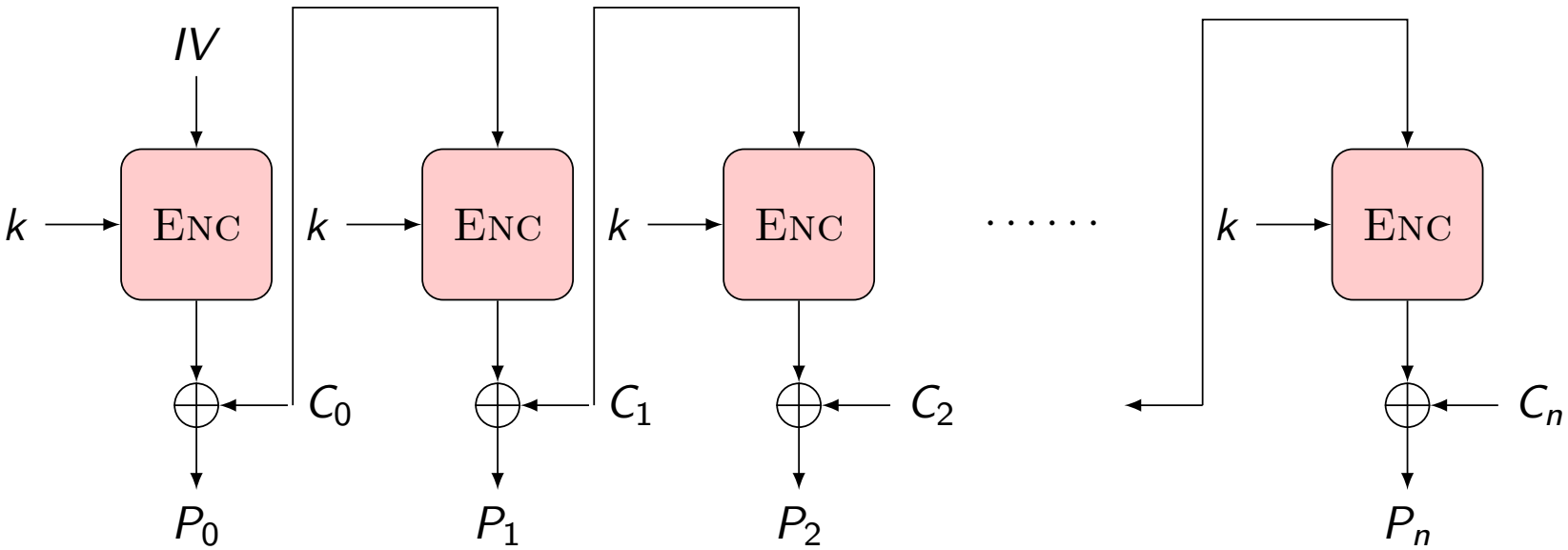
$$P_i = E_K(C_{i-1}) \oplus C_i$$

$$C_0 = \text{IV}$$

CFB Chiffrement



CFB Déchiffrement



Counter Mode (CTR)

Chiffrement :

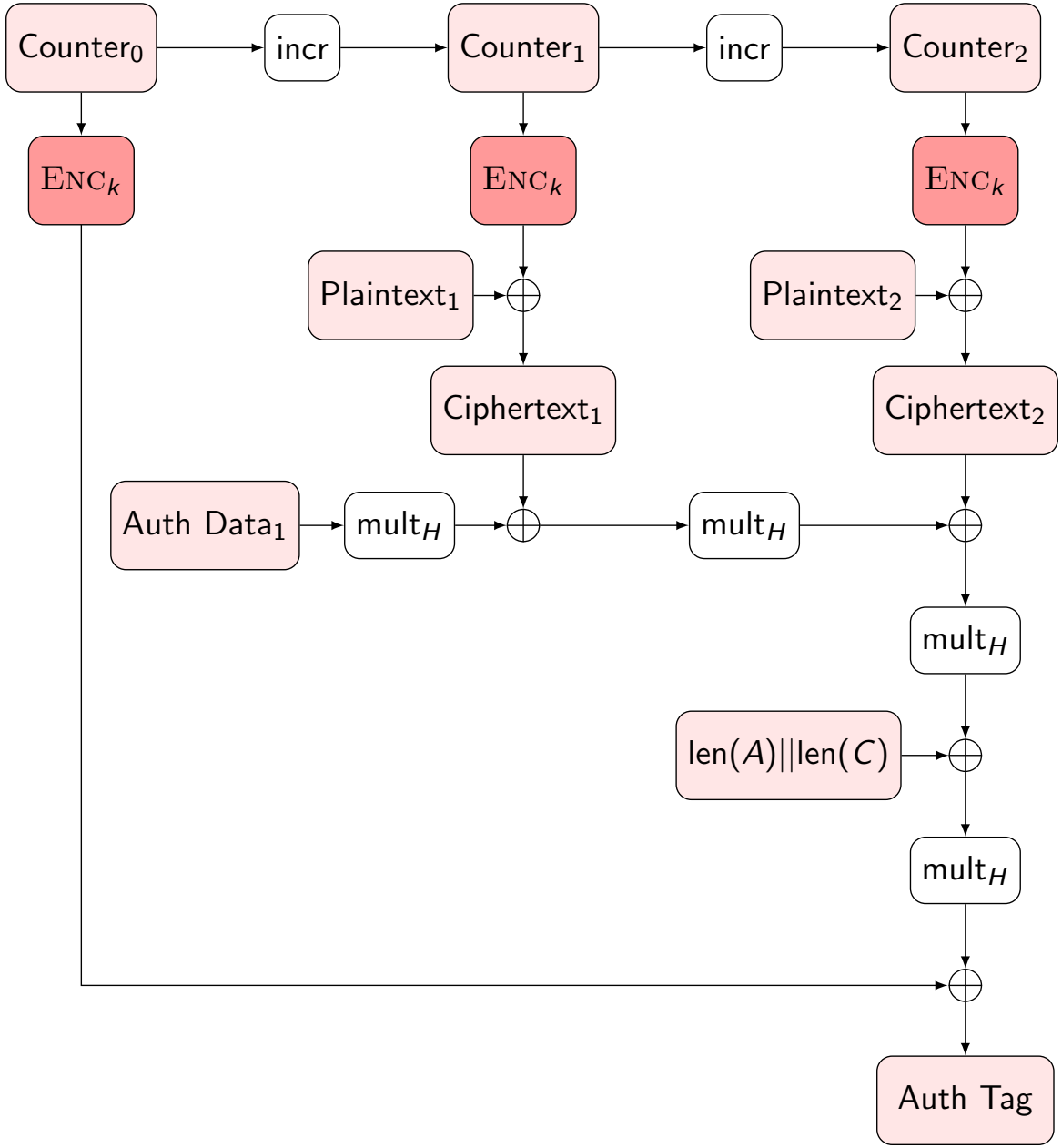
$$C_0 = IV$$

$$C_i = P_i \oplus E_k(IV + i - 1)$$

Déchiffrement :

$$P_i = C_i \oplus E_k(IV + i - 1)$$

GCM Galois/Counter Mode par D. McGrew et J. Viega

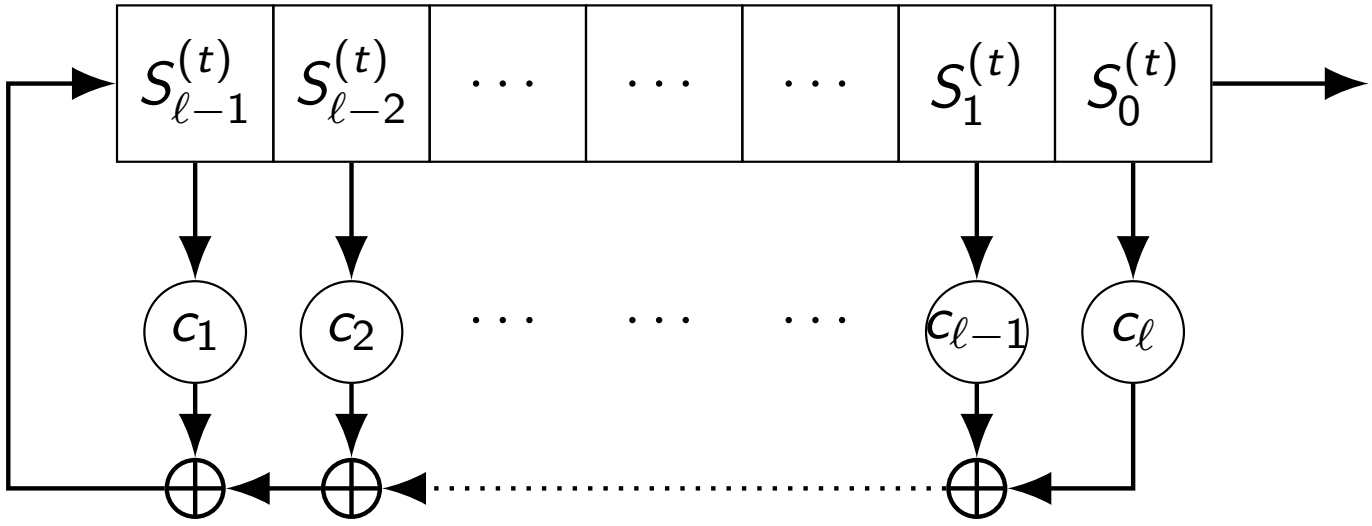


Plan

Chiffrement par flots

Les messages sont chiffrés par un XOR avec une clé.
Une graine (clé partagée) permet de générer un flot de bits pseudo-aléatoires servant de clé OTP.

Linear Feedback Shift Register



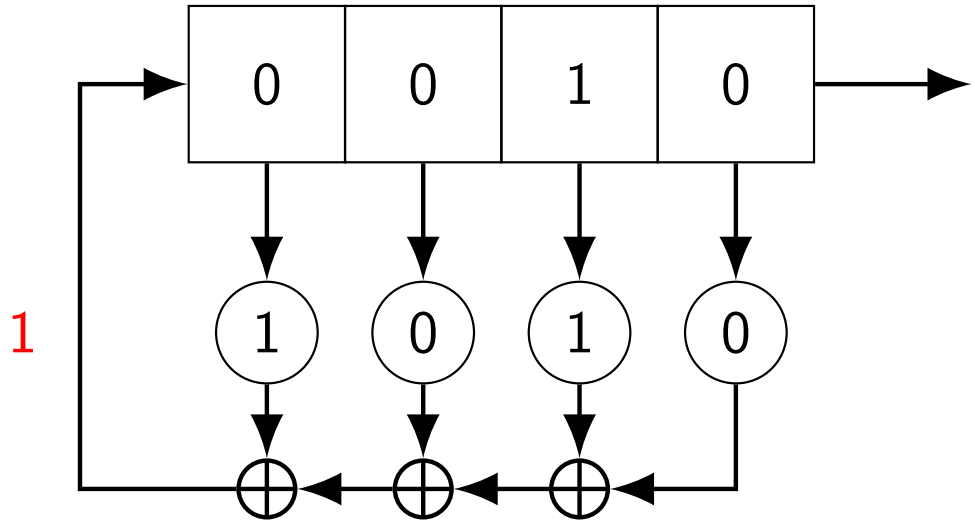
- Longueur du registre est ℓ, $s^{(0)}$ est la graine (seed)
- $\forall c_i \in \{0, 1\}$

$$\forall t \geq 0, s_{\ell-1}^{(t+1)} = \sum_{i=1}^{\ell} c_i s_{\ell-i}^{(t)}$$

$$Shift : s_i^{(t+1)} = s_{i+1}^{(t)}, \forall i, 0 \leq i \leq \ell - 2$$

Exemple

Graine $s^{(0)} = 0010$ et $c_1 = 1$ $c_2 = 0$ $c_3 = 1$ and $c_4 = 0$

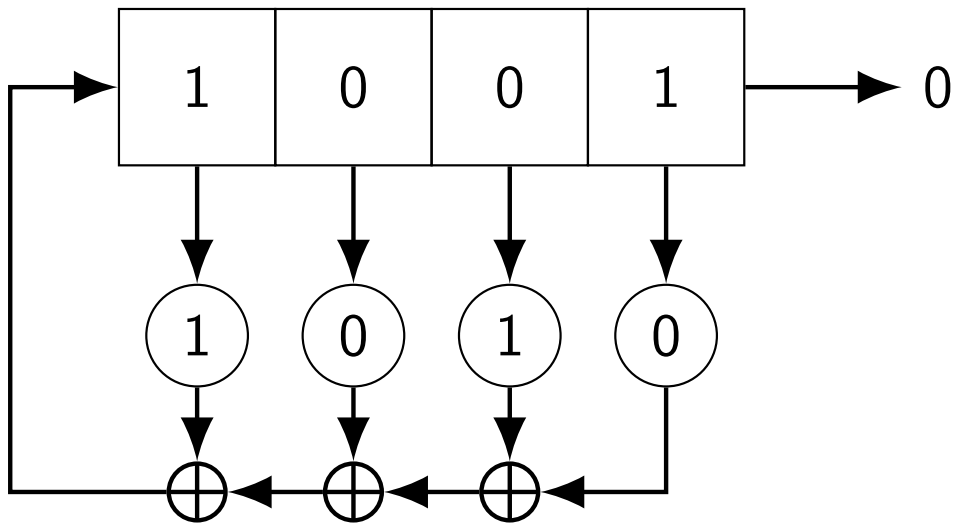


$$
\begin{aligned}
s_3^{(1)} &= (s_3^{(0)} \cdot c_1) \oplus (s_2^{(0)} \cdot c_2) \oplus (s_1^{(0)} \cdot c_3) \oplus (s_0^{(0)} \cdot c_4) \\
&= (0 \cdot 1) \oplus (0 \cdot 0) \oplus (1 \cdot 1) \oplus (0 \cdot 0) \\
&= 1
\end{aligned}
$$

Example first output bit

$c_1 = 1$ $c_2 = 0$ $c_3 = 1$ and $c_4 = 0$

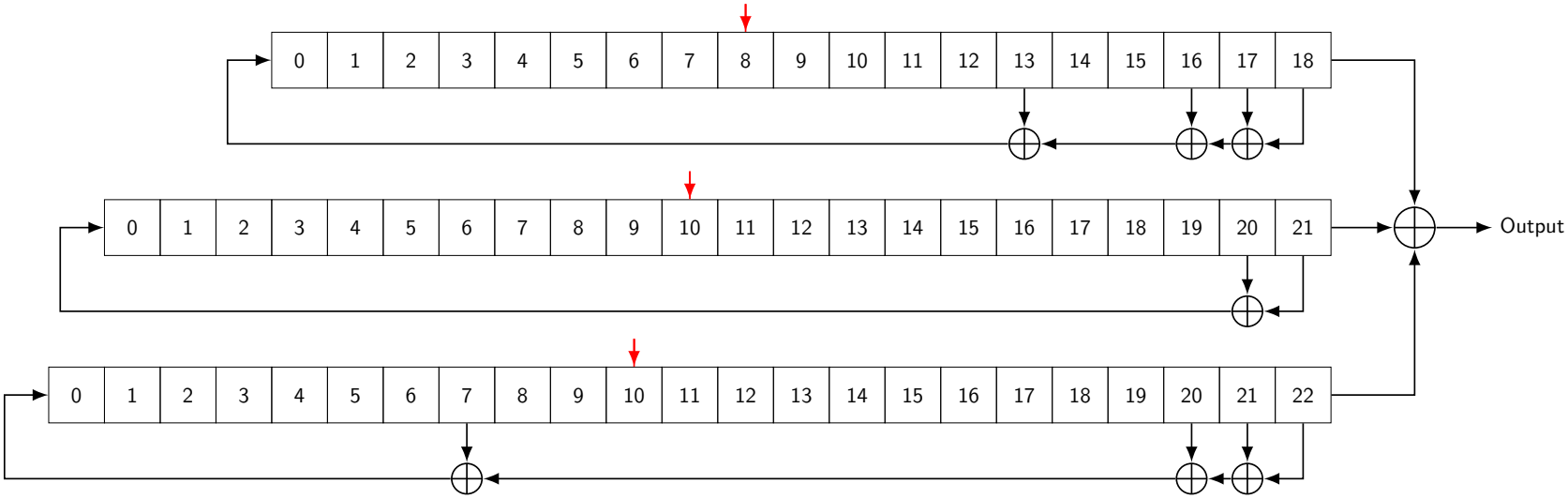
$$s_2^{(1)} = s_3^{(0)}, s_1^{(1)} = s_2^{(0)}, \text{ and } s_0^{(1)} = s_1^{(0)}$$



A5/1 used for GSM in Europe 1994

Les bits rouges déterminent la majorité des 3 valeurs.

Les registres gagnants sont décalés.



$$x^{19} + x^{18} + x^{17} + x^{14} + 1$$

$$x^{22} + x^{21} + 1$$

$$x^{23} + x^{22} + x^{21} + x^{8} + 1$$

Attaques on A5/1

- 1997, Golic attaque en $2^{40.16}$
- 2000, Alex Biryukov, Adi Shamir and David Wagner : few minutes with 2 minutes of plain communication (using in total 300 Go data, in 2^{48} steps).
- 2000 Eli Biham et Orr Dunkelman attack in $2^{39.91}$ with $2^{20.8}$ bits fo data.
- Améliorations par Maximov et al. Maximov, Alexander; Thomas Johansson; Steve Babbage (2004). "An Improved Correlation Attack on A5/1". Selected Areas in Cryptography 2004: 1–18.
  Barkan, Elad; Eli Biham (2005). "Conditional Estimators: An Effective Attack on A5/1". Selected Areas in Cryptography 2005: 1–19.
- 13 décembre 2013, avec les affiramtions de Snowden, NSA peut écouter les communications GSM.

RC4 by Ron Rivest in 1987

"Rivest Cipher 4" ou "Ron's Code" est un chiffrement par flots utilisé dans les premières versions de TLS (Transport Layer Security) et dans le protocole WEP (Wired Equivalent Privacy).
Il comporte

- Un algorithme de dérivation de clé (KSA)
- Un algorithme de génération de pseudo-aléatoire (PRGA)

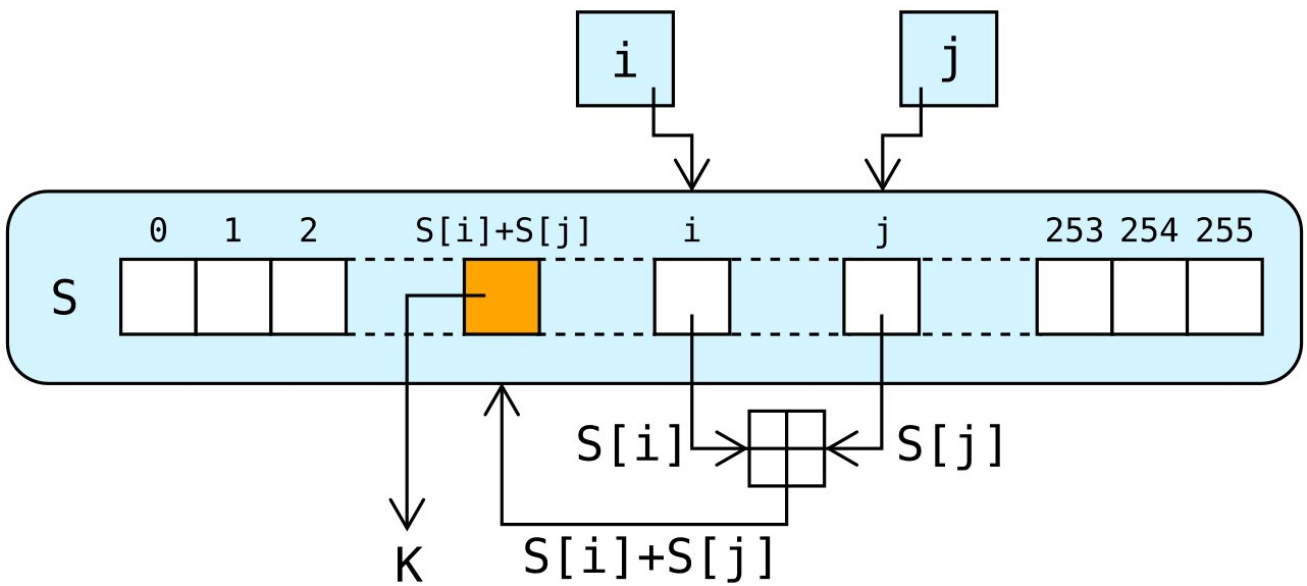
KSA utilise une clé de longueur entre 40 – 128 bits

- Le tableau "S" est initialisé avec la permutation identité
- 256 itérations avec des mélanges de bytes de la clé sont réalisés en même temps.

```
j := 0
for i from 0 to 255
   j := (j + S[i] + key[i mod keylength]) mod 256
   swap values of S[i] and S[j]
endfor
```

Pseudo-Random Generation Algorithm (PRGA)

```
i := 0; j := 0;
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
```



Recent attacks on RC4

- Fluhrer, Mantin and Shamir attack 2001
- Klein's attack 2005
- John Leyden (2013-09-06). "That earth-shattering NSA crypto-cracking: Have spooks smashed RC4?"
- "Fresh revelations from whistleblower Edward Snowden suggest that the NSA can crack TLS/SSL connections, the widespread technology securing HTTPS websites and virtual private networks (VPNs)."
- " Attack relies on statistical flaws in the keystream generated by the RC4 algorithm. It relies on getting a victim to open a web page containing malicious JavaScript code that repeatedly tries to log into Google's Gmail, for example. This allows an attacker to get hold of a bulk of traffic needed to perform cryptanalysis."

  Nadhem AlFardan, Dan Bernstein, Kenny Paterson, Bertram Poettering and Jacob Schuldt. "On the Security of RC4 in TLS". Royal Holloway University of London. Retrieved March 13, 2013.

RC4 bad

```
int main (int argc , char * argv []) {
    unsigned char S [256] , c;
    unsigned char key [] = KEY;
    int klen = strlen ( key );
    int i,j,k;

    /* Init S[] */
    for (i =0; i <256; i++)
        S[i] = i;

    /* Scramble S[] with the key */
    j = 0;
    for (i =0; i <256; i++) {
         j = (j+S[i]+ key [i% klen ]) % 256;
         S[i] ^= S[j];
         S[j] ^= S[i];
         S[i] ^= S[j];
     }
     /* Generate the keystream and cipher the input stream */
     i = j = 0;
     while ( read (0, &c, 1) > 0) {
          i = (i +1) % 256;
          j = (j+S[i]) % 256;
          S[i] ^= S[j];
          S[j] ^= S[i];
          S[i] ^= S[j];
          c ^= S[(S[i]+S[j]) % 256];
          write (1, &c, 1);
     }}
```

RC4 Good

```
int main (int argc , char * argv []) {
    unsigned char S [256] , c;
    unsigned char key [] = KEY;
    int klen = strlen ( key );
    int i,j,k;

    /* Init S[] */
    for (i =0; i <256; i++)
        S[i] = i;

    /* Scramble S[] with the key */
    j = 0;
    for (i =0; i <256; i++) {
         j = (j+S[i]+ key [i% klen ]) % 256;
         k = S[i];
         S[i] = S[j];
         S[j] = k;
     }
     /* Generate the keystream and cipher the input stream */
     i = j = 0;
     while ( read (0, &c, 1) > 0) {
          i = (i +1) % 256;
          j = (j+S[i]) % 256;
          k = S[i];
          S[i] = S[j];
          S[j] = k;
          c ^= S[(S[i]+S[j]) % 256];
          write (1, &c, 1);
     }}
```

Swap

Classical (avec une variable en plus)

```
tmp = a
a = b
b = tmp
```

avec des additions et soustractions

```
a = a+b
b = a-b
a = a-b
```

avec le XOR

```
a = a^b
b = a^b
a = a^b
```

Swap

Version buguée

```
S[i] = S[i]^S[j]
S[j] = S[i]^S[j]
S[i] = S[i]^S[j]
```

car quand $i = j$, on a

```
S[i] = S[i]^S[i]
S[i] = S[i]^S[i]
S[i] = S[i]^S[i]
```

- Au lieu d'échanger les valeurs, on les met à 0
- L'état de RC4 se remplit progressivement de 0
- Le flot se remplit de 0
- Le chiffrement n'est plus utile

Plan

Aujourd'hui

1. Chiffrements symmétriques
2. Par bloc : AES, DES, IDEA, Simon
3. Modes : ECB, CBC ...
4. Flots : LSFR, RC4

Merci pour votre attention.

Questions ?

Ron Rivest

“Once you have something on the Internet, you are telling the world, please come hack me.”