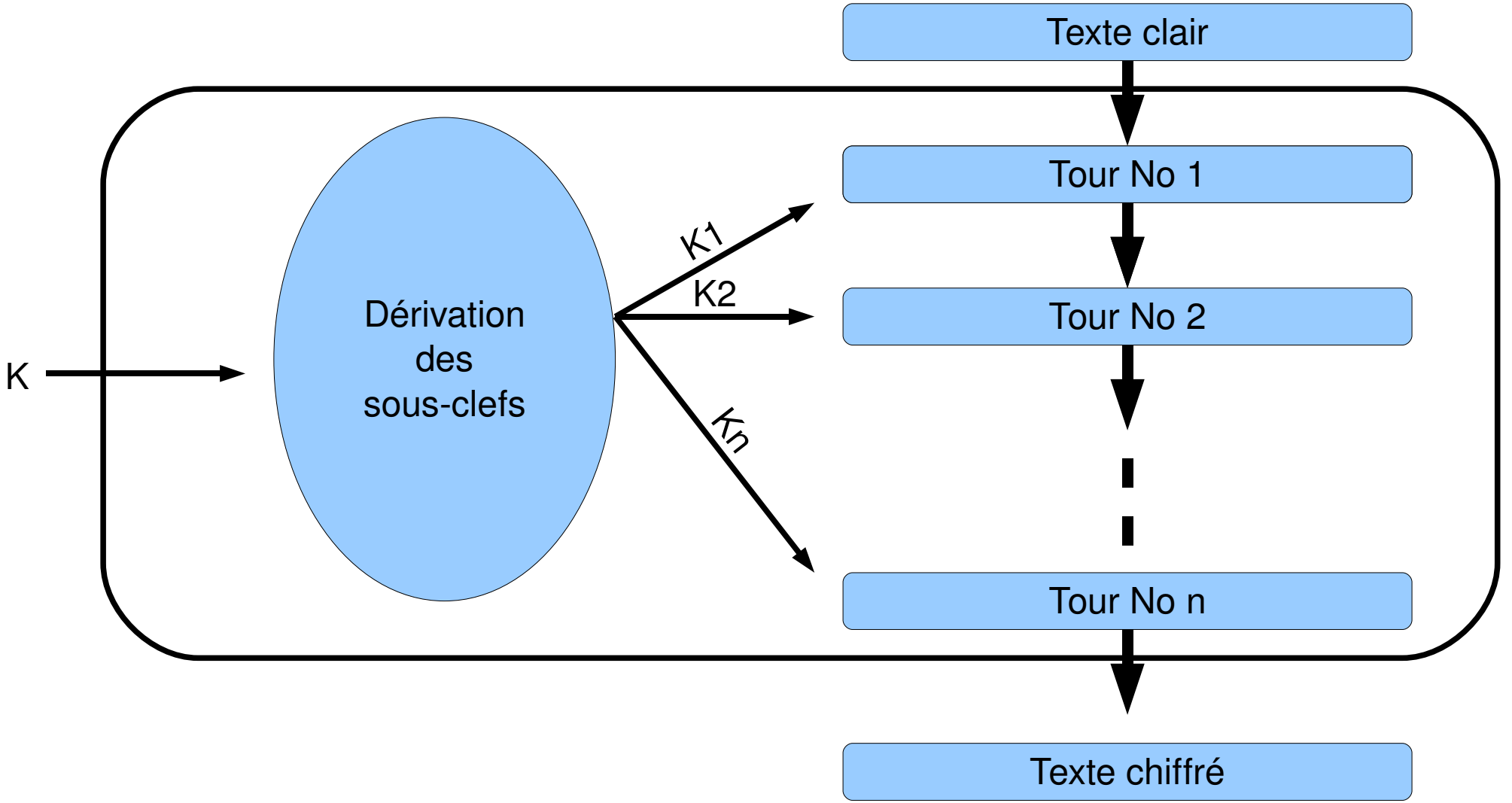
Construction des chiffrements par bloc

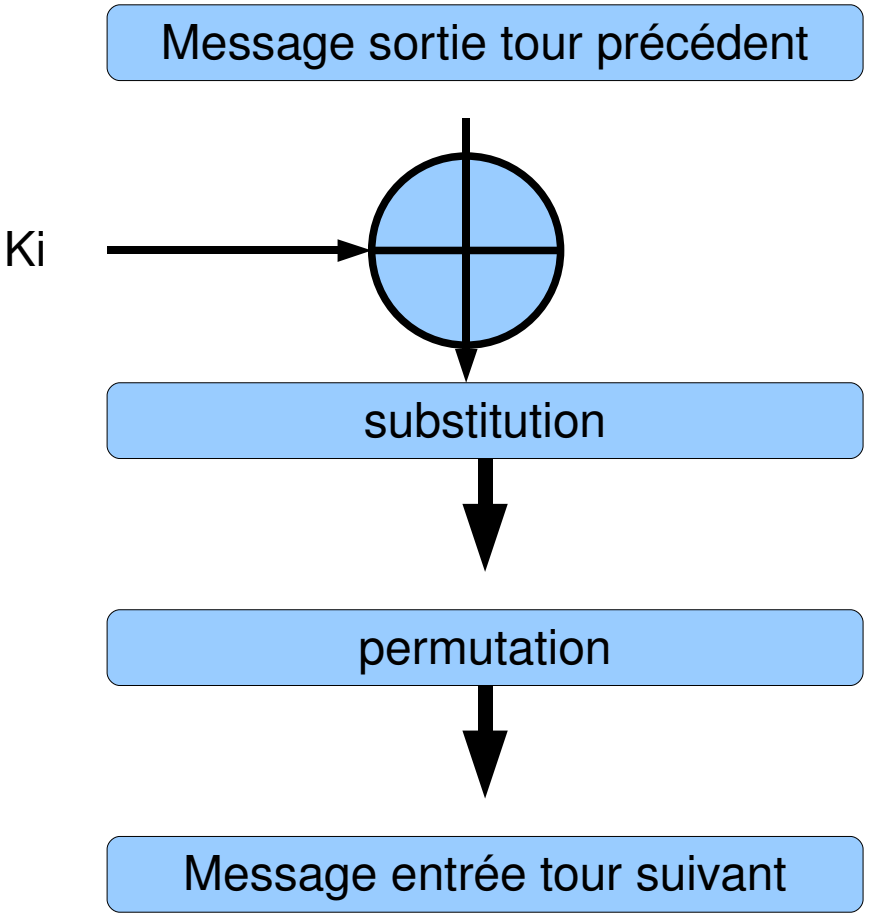
- Idée: itérer suffisamment de fois des opérations simples ***judicieusement choisies*** permet d'avoir une excellente sécurité
- Le chiffrement est une succession d'étapes similaires (appelées « tour ») utilisant chacun une sous-clef
  - Construction itérative et modulaire
  - Les sous-clefs sont dérivées de la clef secrète
  - Les fonctions utilisées par un tour doivent être optimisées et sont en général des opérations simples

Construction des chiffrements par bloc



Méthode par substitutions/permutations

- Décomposition d'un tour:
  - Utilisation de la clef Ki
  - Fonction de substitution
  - Fonction de permutation

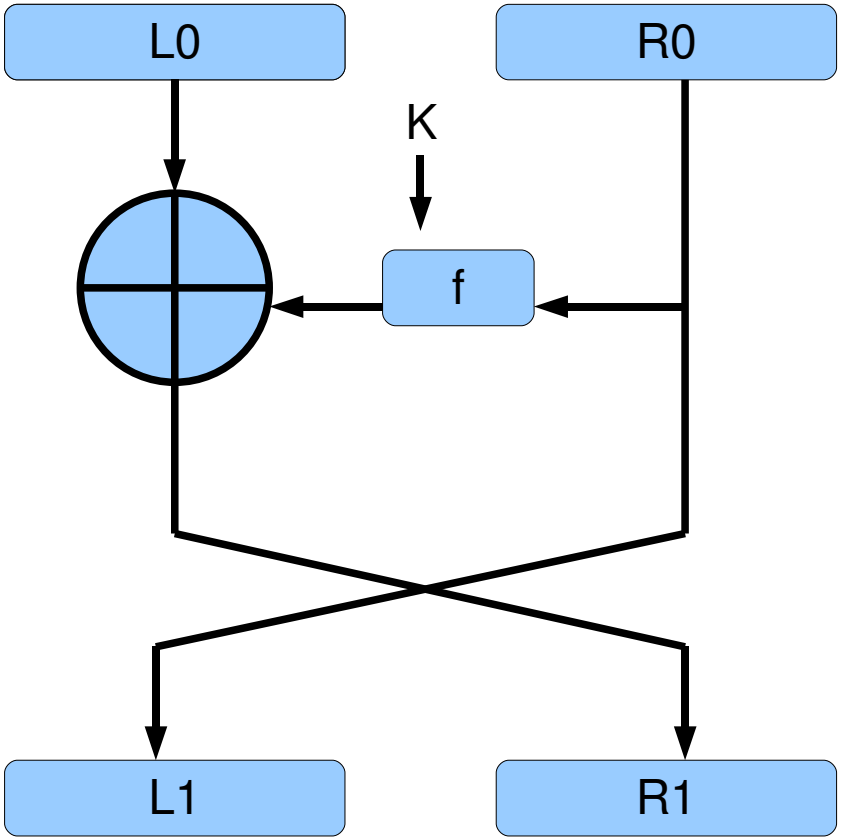


Substitution: un symbole du texte clair est remplacé par un autre symbole. Cf tables de substitution du DES (S-Boxes). Ajoute de la confusion.
permutation: on échange entre eux les symboles du texte en clair. Ajoute de la diffusion

Schéma de Feistel

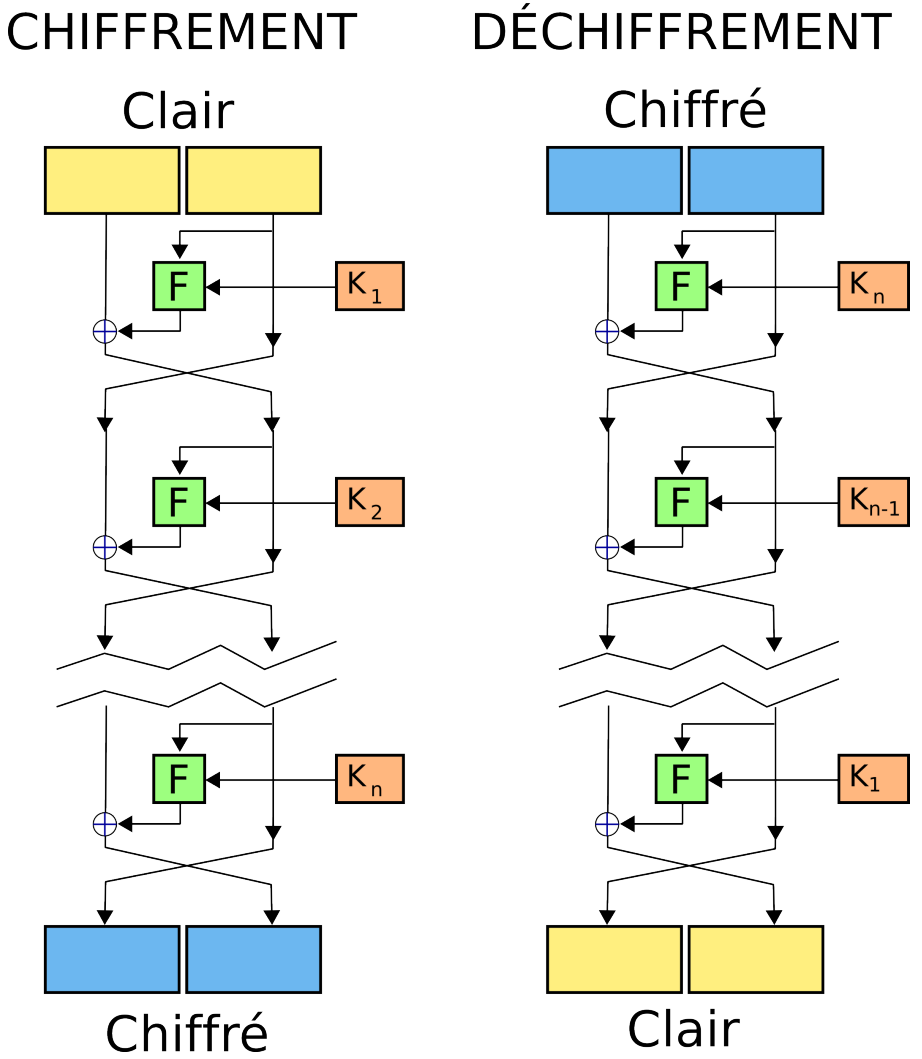
- Chiffrement:
  - L1=R0
  - R1=L0 ⊕ f(R0, K)
- Déchiffrement:
  - R0=L1
  - L0=R1⊕f(R0, K)

La fonction f est appelée fonction de confusion



Réseau de Feistel

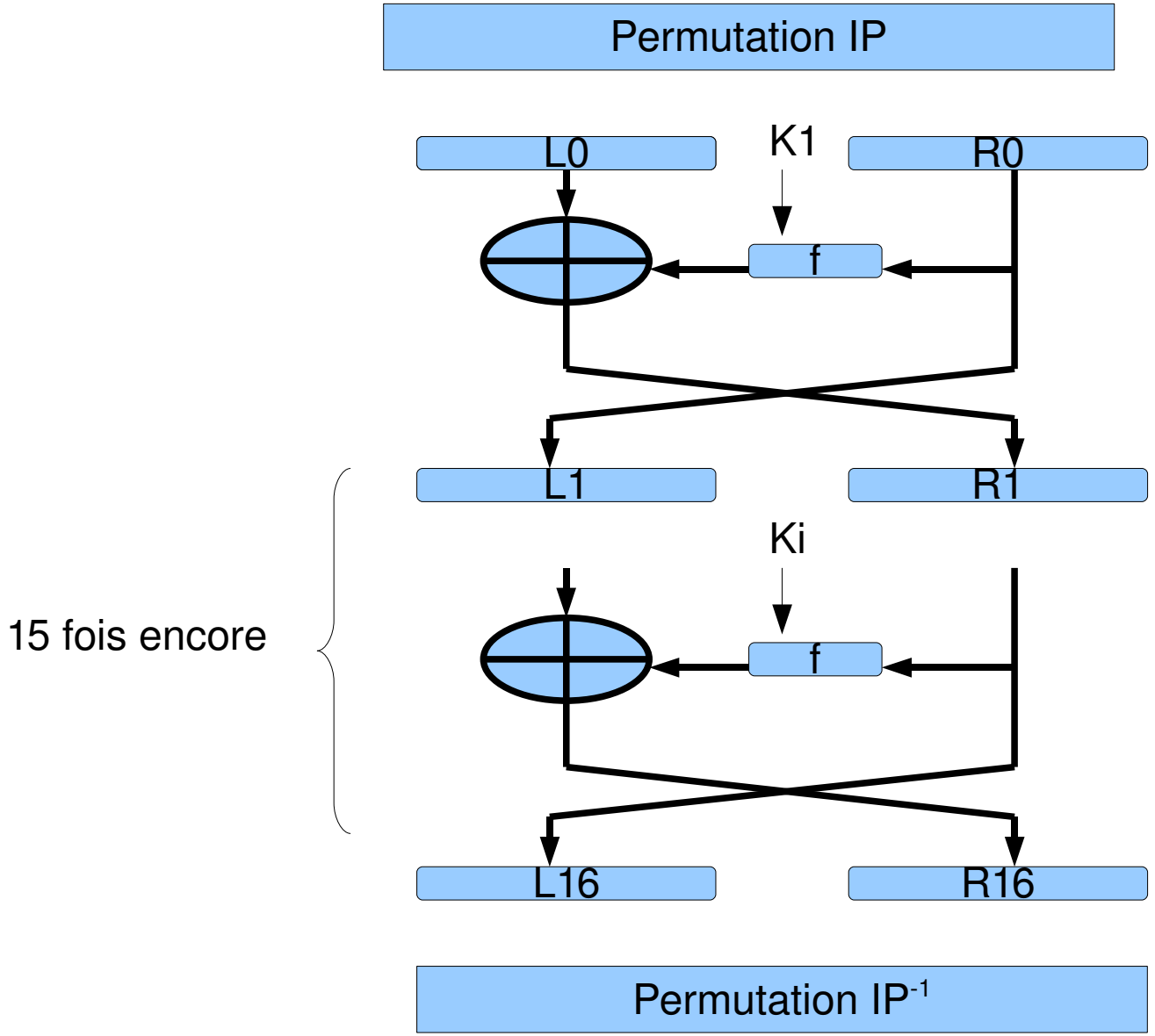
- Constitué de plusieurs tours (ou rondes) ou étages de Feistel
- Chiffrement et déchiffrement sont efficaces si chaque tour l'est
- La complexité du décryptage croit de façon exponentielle avec le nombre de tour



DES: Data Encryption Standard

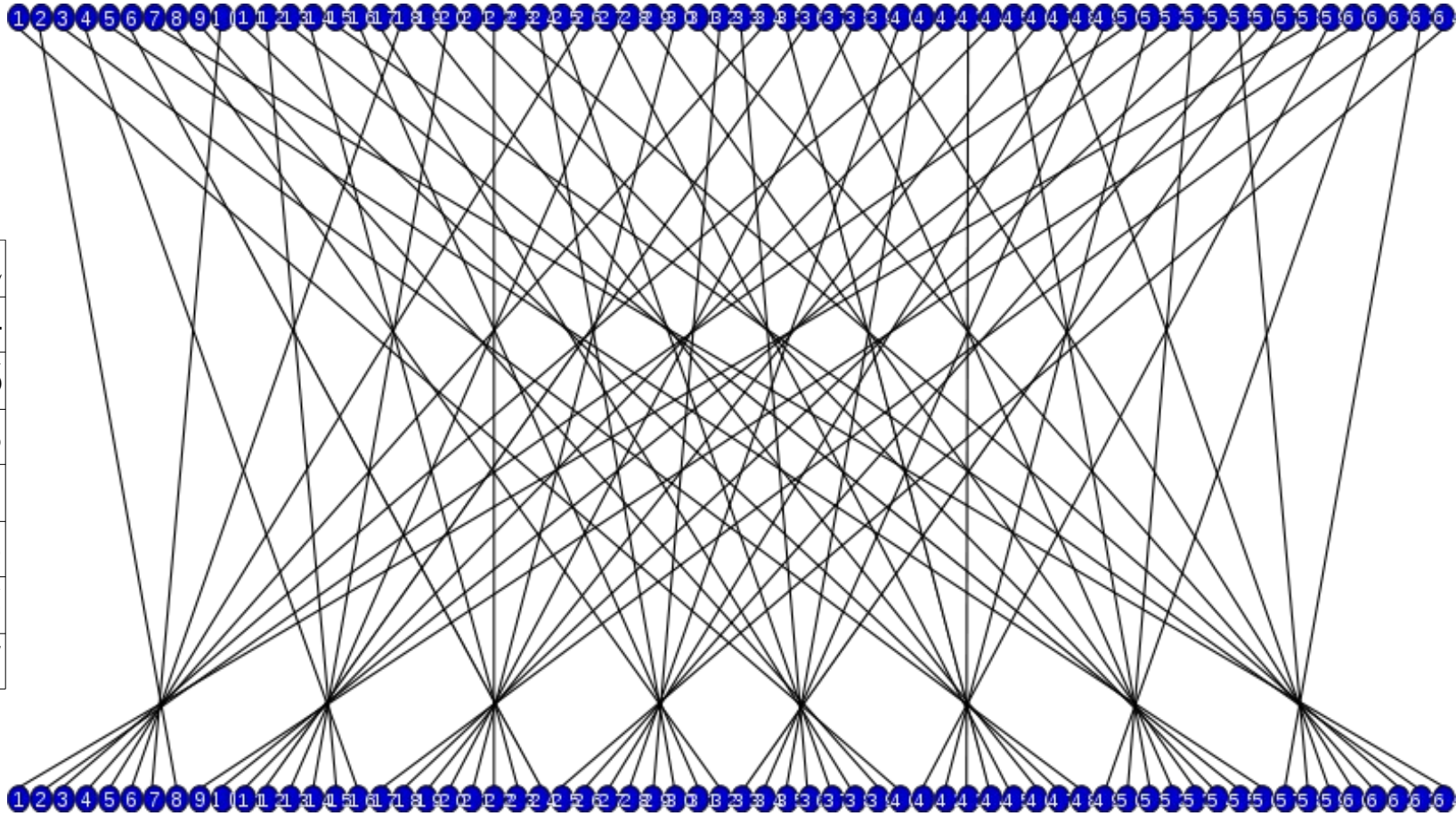
- Propriétés:
  - Algorithme à clef privée
  - Chiffrement par blocs de 64 bits
  - Feistel 16 tours avec des sous-clefs de 48 bits
  - Conçu dans les années 1970
  - Considéré comme obsolète de nos jours
- Conception:
  - À partir de l'algorithme Lucifer (IBM, 1970)
  - Corrigé par la NSA :
    - Clefs de 56 bits au lieu de 64
    - Boîtes S corrigées
    - Les but de la NSA ne seront découverts qu'en 1990
  - Standard FIPS 46-2 en 1977

DES: schéma


Permutation IP
L0
K1
R0
f
L1
R1
15 fois encore
Ki
f
L16
R16
Permutation IP-1

DES: permutation initiale IP

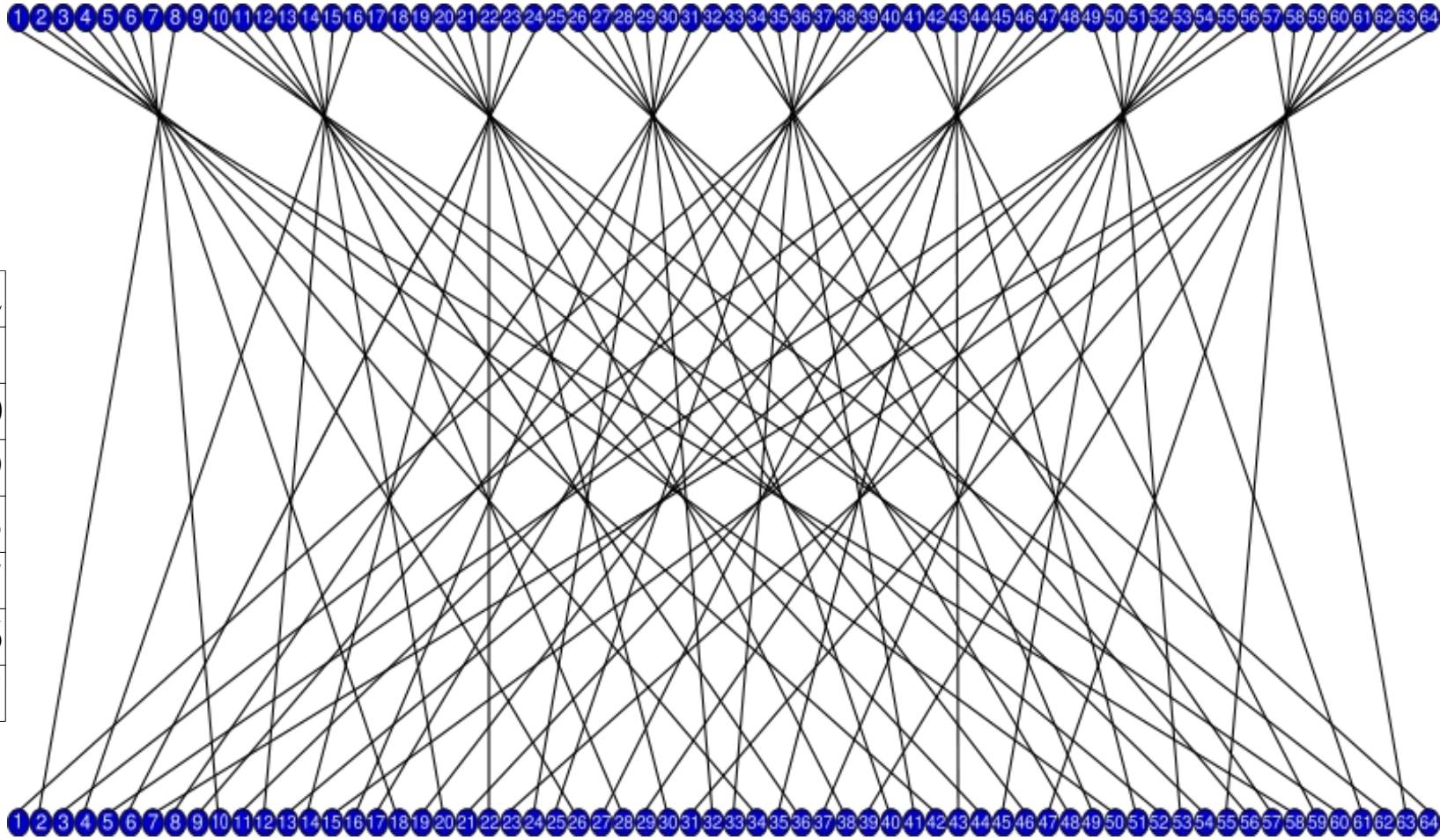
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7



Le 58e bit devient le premier, le 50e bit devient le deuxième, ..le 1er bit devient le 40e
si m=p1p2...p64 alors P(m)=p58p50...p7

DES: permutation finale

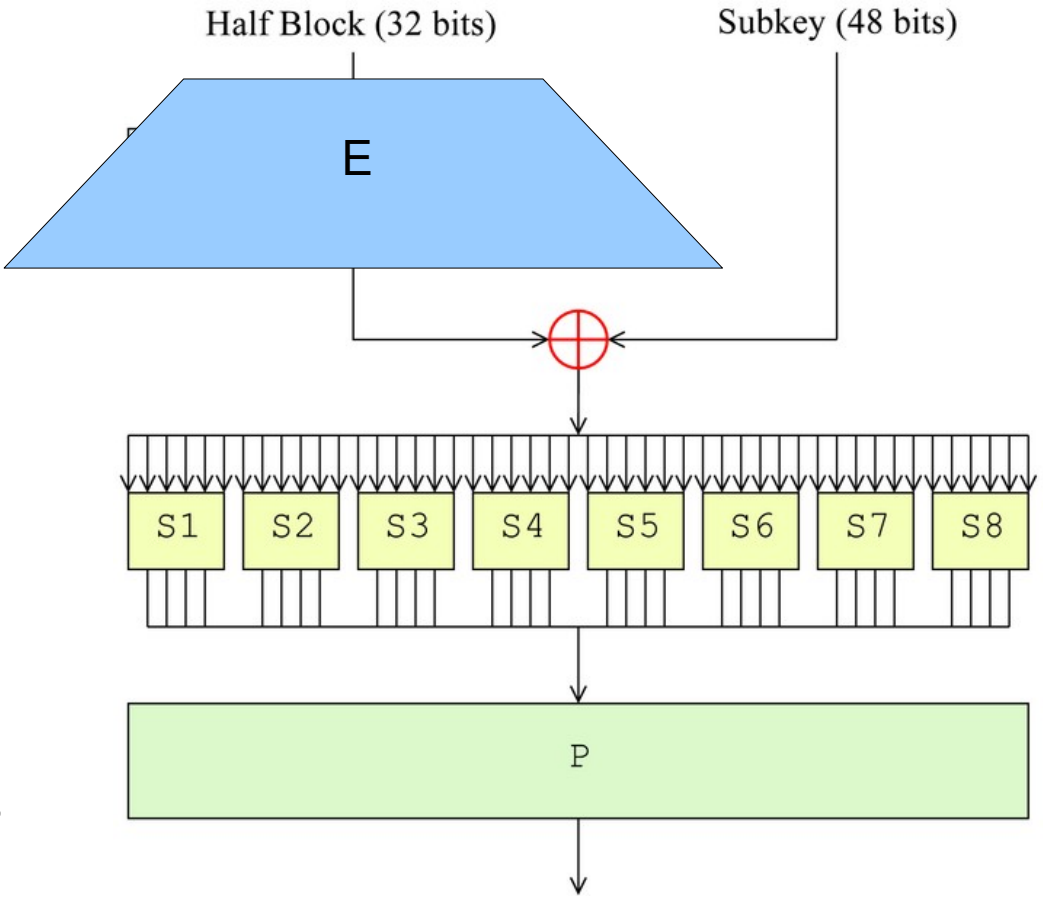
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25



Le 40e bit devient le premier, le 8e bit devient le deuxième, ...le deuxième bit devient le 50e, ... cette permutation est l'inverse de la première.

DES: fonction f

- La fonction f est une fonction de 32 bits vers 32 bits constituée:
  - d'une expansion de Ri de 32 bits vers 48 bits
  - d'un XOR avec 48 bits dérivés de la clef
  - de l'application de 8 sous-fonctionsde 6 bits vers 4 bits : les boîtes S
  - d'une permutation des 32 bits sortants

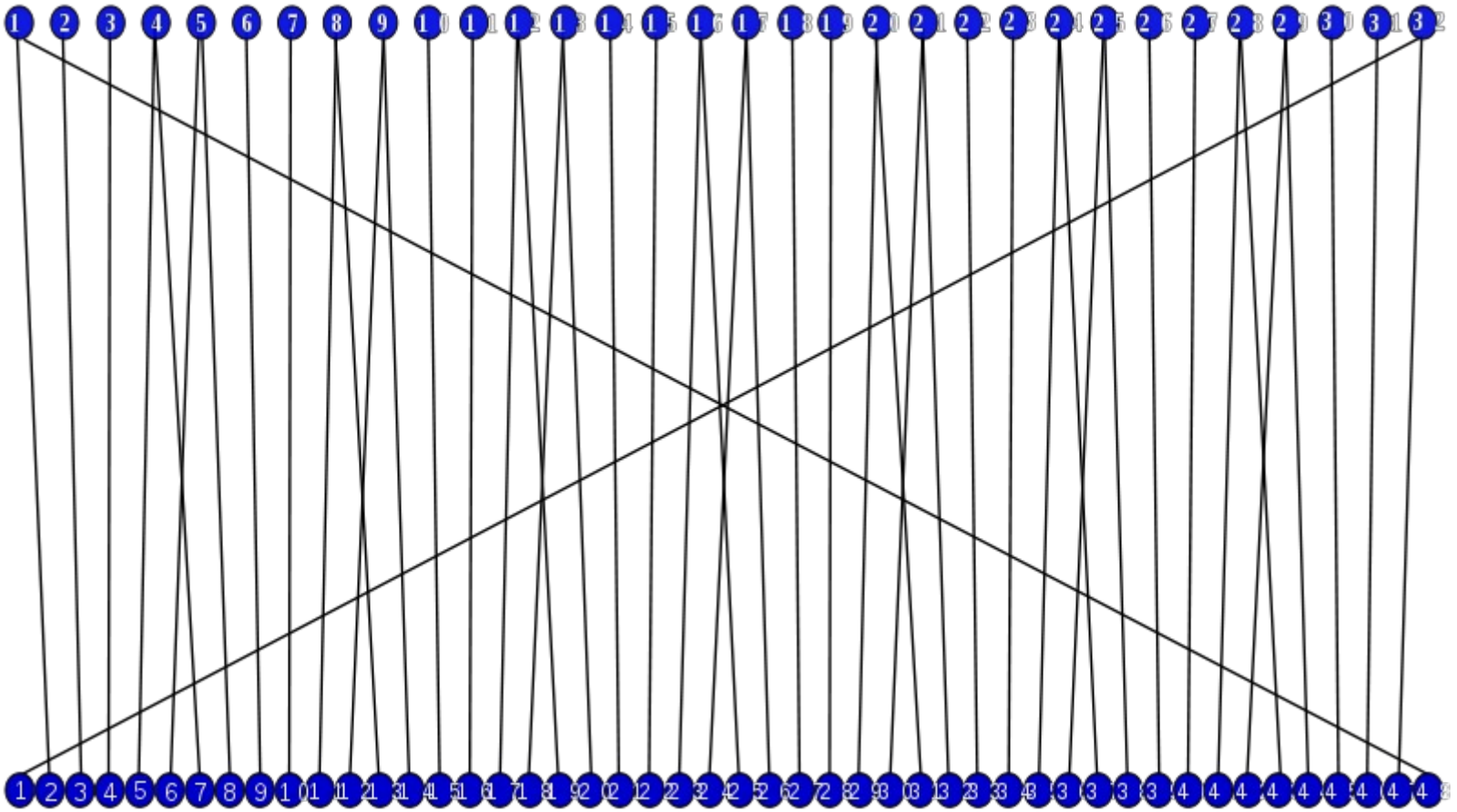


DES: fonction f

- La fonction f n'est pas inversible (elle n'a pas besoin de l'être)
- L'expansion duplique certains bits
- Les boîtes assurent la non linéarité (6 bits -> 4 bits pour chaque boîte)
- La permutation renforce la diffusion (tout comme le fait le croisement du schéma de Feistel)

DES: expansion E

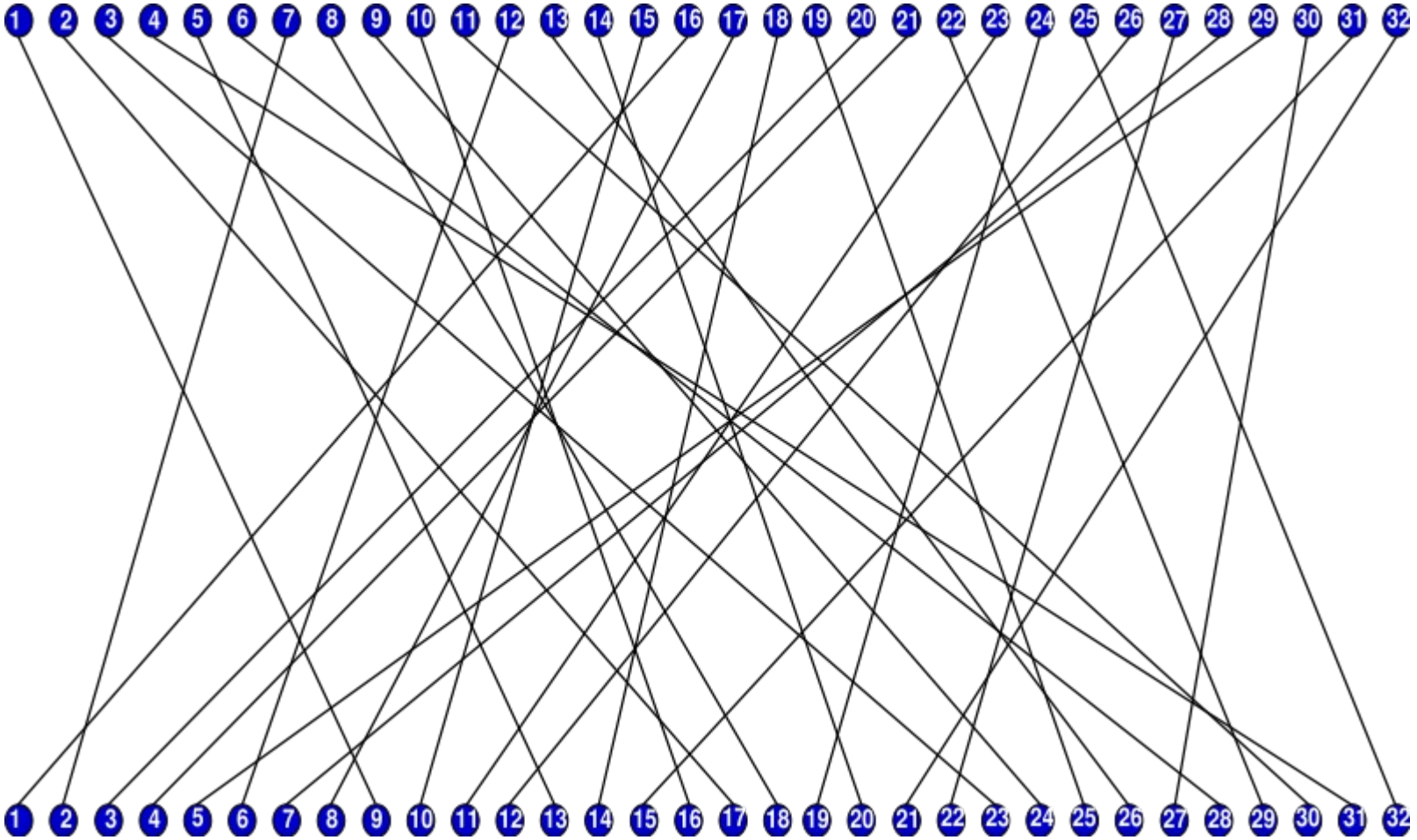
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1



Le 32e bit devient le premier, le 1er bit devient le 2e, ...

DES: permutation 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



Le 16e bit devient le 1er, le 7e bit devient le 2e, ...

DES: boîtes de substitutions

Utilisation des boîtes-S:

si B=b1b2b3b4b5b6

s(B) est le mot binaire situé à la ligne b1b6 et à la colonne b2b3b4b5.

La numérotation des lignes et des colonnes commence à 0.

Exemple: S1(<u>1</u>11001<u>1</u>):

- Ligne 11=ligne3
- Colonne 1100=colonne 12
- S1(111001)=10=1010

S1															
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

S2															
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

S3															
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

S4															
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

DES: boîtes de substitutions

S5															
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

S6															
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

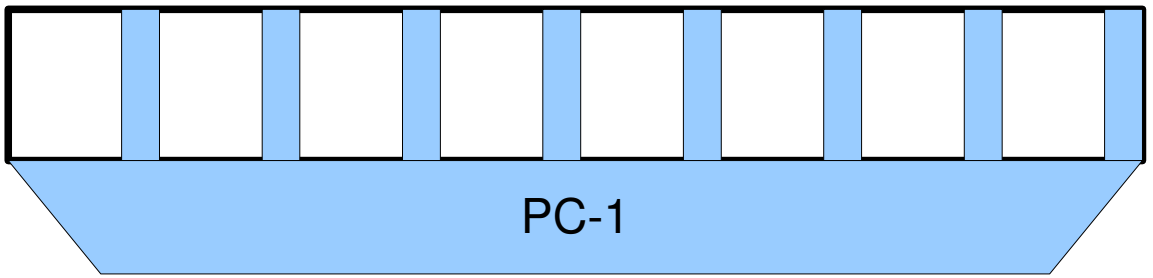
S7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8															
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

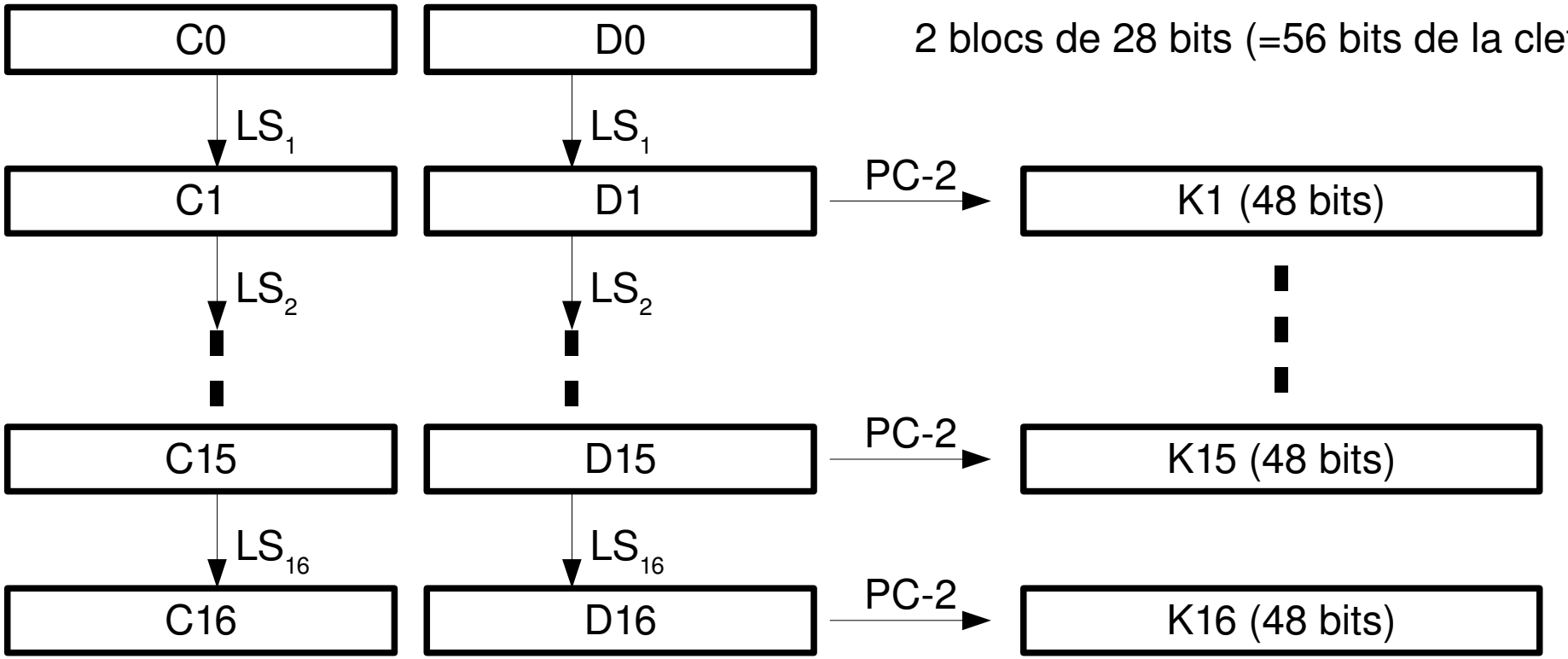
DES: génération des clefs des rondes

- Décalage: LSI : 28 bits -> 28 bits
  - $LS_I(C)$ décalage vers la gauche de V_i bits
  - V_i=1 si i =1,2,9 ou 16
  - V_i=2 sinon
- PC1: une fonction qui retourne 2 mots de 28 bits à partir d'un mot de 64 bits
  - Elle supprime les bits de parité (8, 16, ..., 64)
  - On obtient la clef de 56 bits en 2 mots de 28 bits
- PC2:
  - 28 bitsx28bits -> 48 bits
  - Son résultat fait 48 bits et sert de paramètre à la fonction f de DES

DES : génération des clefs



64 bits donc 8 bits de parité
répartis aux positions 8, 16, ... 64

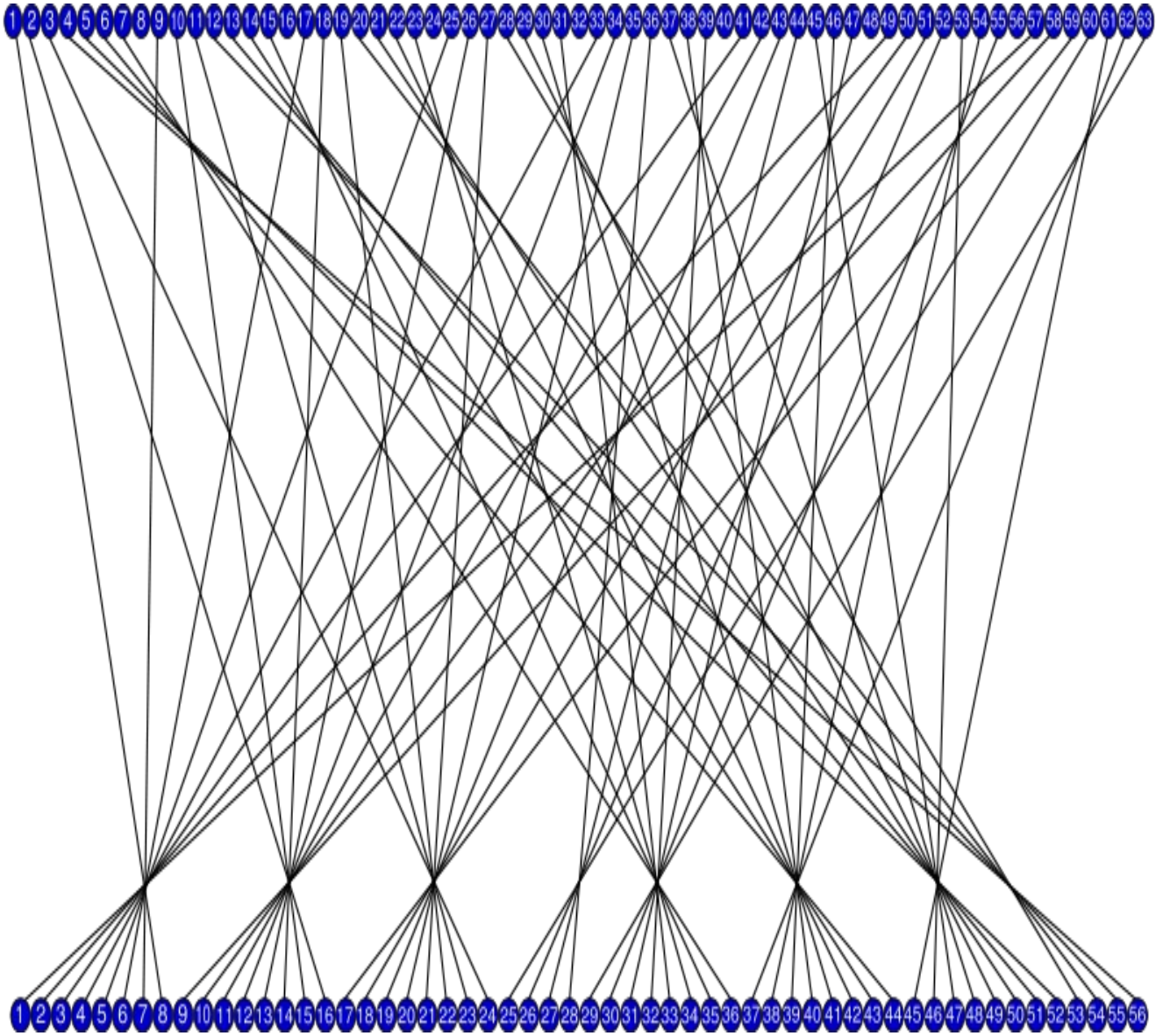


DES: génération des clefs des rondes PC1

Gauche						
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
Droite						
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

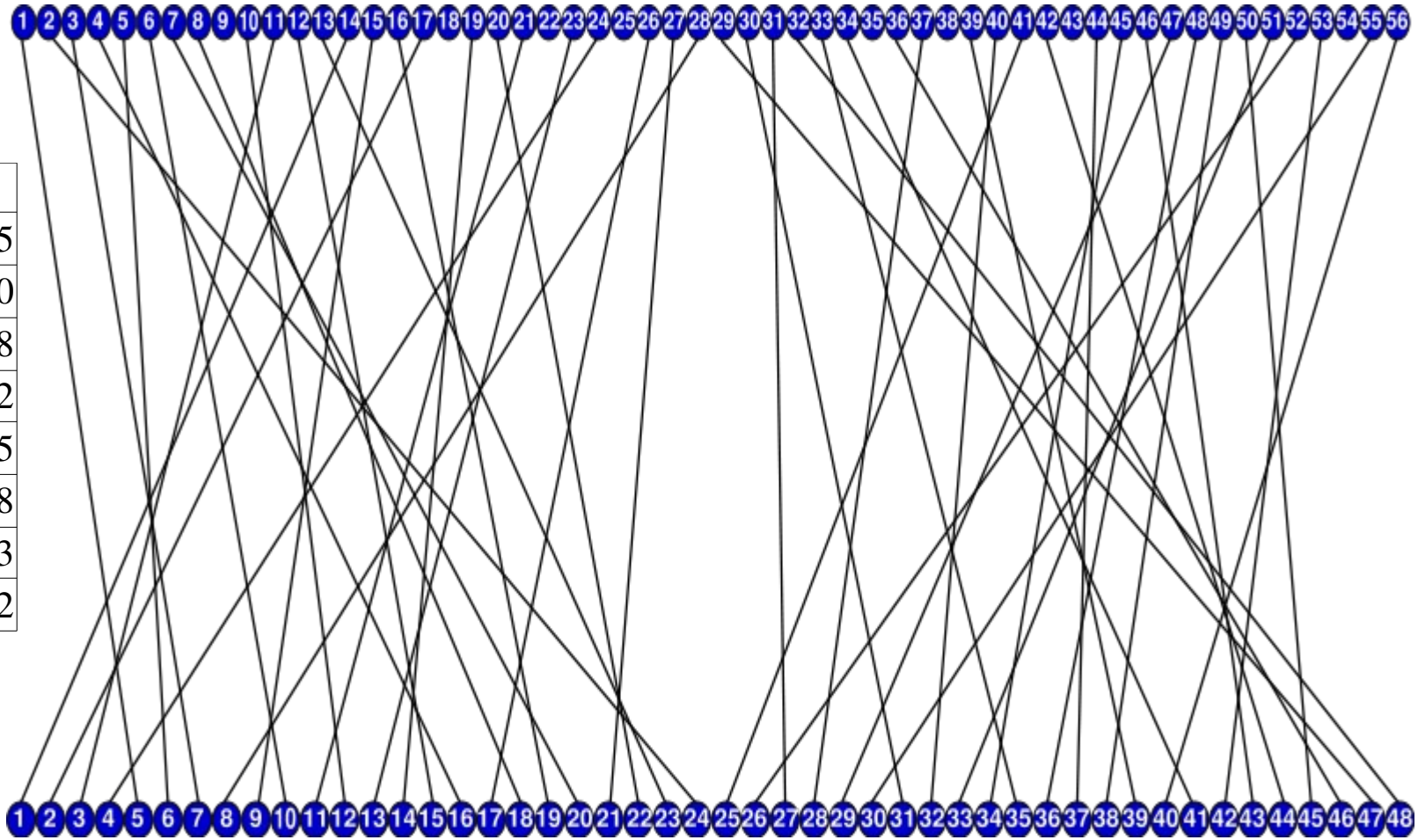
Si $k=k_1k_2...k_{64}$ alors :

- $C=k_{57}k_{49}...k_{36}$
- $D=k_{63}k_{55}...k_4$



DES: génération des clefs des rondes PC2

PC-2					
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



$PC2(b_1 b_2 ... b_{56}) = b_{14} b_{17} ... b_{32}$

DES: performances

- Des opérations simples:
  - La structure de Feistel est rapide
  - La dérivation des clefs est particulièrement simple et efficace
  - Les boîtes S se cablent facilement en matériel
- DES utilise des opérations simples facilement implémentables sur des puces spécialisées
  - XXXMb/s sur un XXX à XXX MHz
  - 1,5Gb/s sur une puce spécialisée bas de gamme
-

DES: sécurité

- Après 16 tours, résultat statistiquement plat
  - Aucune propriété du message source ne sera détectable
- Une légère modification du texte chiffré ou de la clef => de grosses modifications du texte chiffré
  - Bonné résistance aux attaques par analyse différentielles

DES: cryptanalyse

- Force brute : espace des clefs : 2^{56}
  - À la portée d'une PME (cf TD)
  - En 1998, l'EFF (Electronic Frontier Fondation) construit une machine
    - Craquant DES en 9 jours
    - Coût: $250 000
- Taille des blocs:
  - 64 est devenu court
  - On peut obtenir deux blocs chiffrés identiques ce qui laisse la porte ouverte à certaines attaques

DES: amélioration

- Idée d'amélioration :
  - Combiner plusieurs chiffrement DES de suite avec des clefs différentes
  - Le résultat n'est pas un DES
    - E(k1, E(k2, E(k3, M)) n'est pas égal à E(k4, M) (avec E: chiffrement DES et C: déchiffrement DES)
    - Sinon, pas d'augmentation de la complexité
  - Plusieurs tentatives :
    - Double DES : complexité #64 bits
    - Triple DES-EEE : 3 chiffrements avec des clefs différentes: E(k1, E(k2, E(k3, M))
    - Triple DES-EDE: E(k1, D(k2, E(k1, M)))
  - 3 fois plus lent que DES

DES: double DES

- Double DES:
  - E(k1, E(K2, M))
  - Atteint-on le niveau de sécurité d'un espace de clef de 56+56=112 bits. NON

Double DES: Attaque

1) On suppose que l'on connait un coupe clair M/chiffré C (un bloc d'un long message suffit)
2) On chiffre M avec toutes les clefs possibles : 2^{56}clefs possible
3) On trie les clefs : $56*2^{56}$ (n log n)
4) On déchiffre C avec toutes les clefs possible : 2^{56}clefs possible
   - On cherche la version déchiffrée avec la clef j dans l'ensemble de l'étape 2: au pire en log n donc au pire $56*2^{56}$
   - Si on trouve i une clef telle que E(i,M)=D(j,C) alors, on a trouvé le bon couple de clefs

– Complexité totale équivalente à celle d'un espace de clef de 64 bits

Triple DES:

- De nombreuses variantes possible
  - Certaines ont des faiblesses
  - DES-EEE : 3 clefs de 56 bits => 168 bits mais l'attaque « meet in the middle » rend la sécurité équivalente à un espace de clef de 112 bits
- La variante conseillée (FIPS 46-3) est DES-EDE:
  - 2 clefs
  - E(k1, D(k2, E(k1, M)))
- Espace de clef de 112 bits
  - Mais en tenant compte des attaques possibles, équivalent à un espace de clef de 80 bits
- Triple Des a été très utilisé avant l'arrivé d'AES pour palier les faiblesses de DES

AES:

- Dans une version ultérieure de ce document

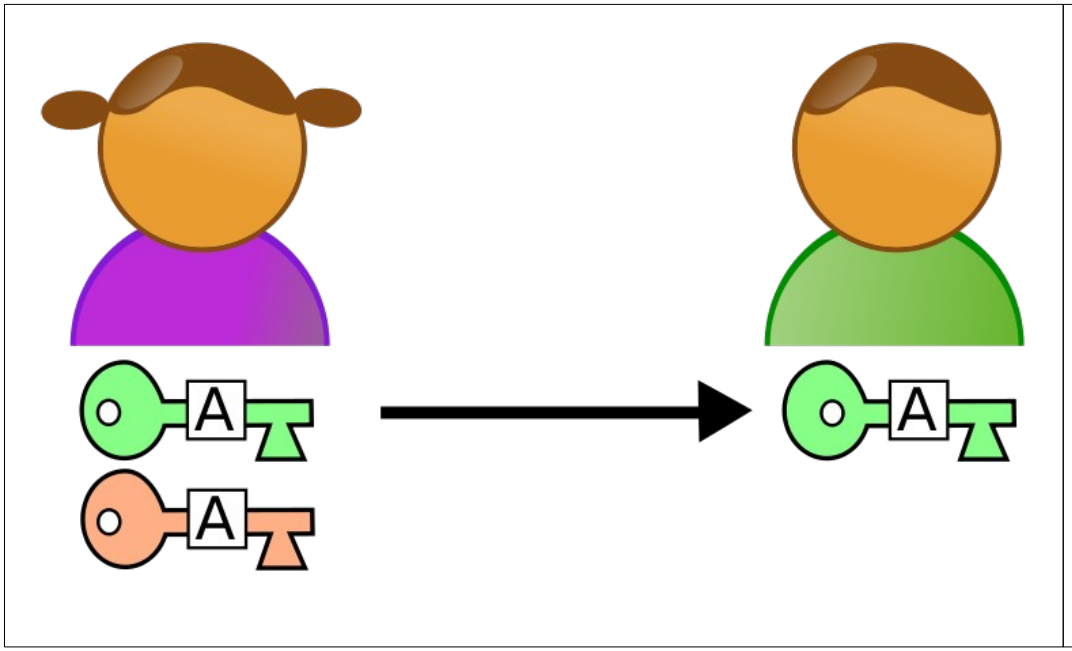
Chiffrement asymétrique

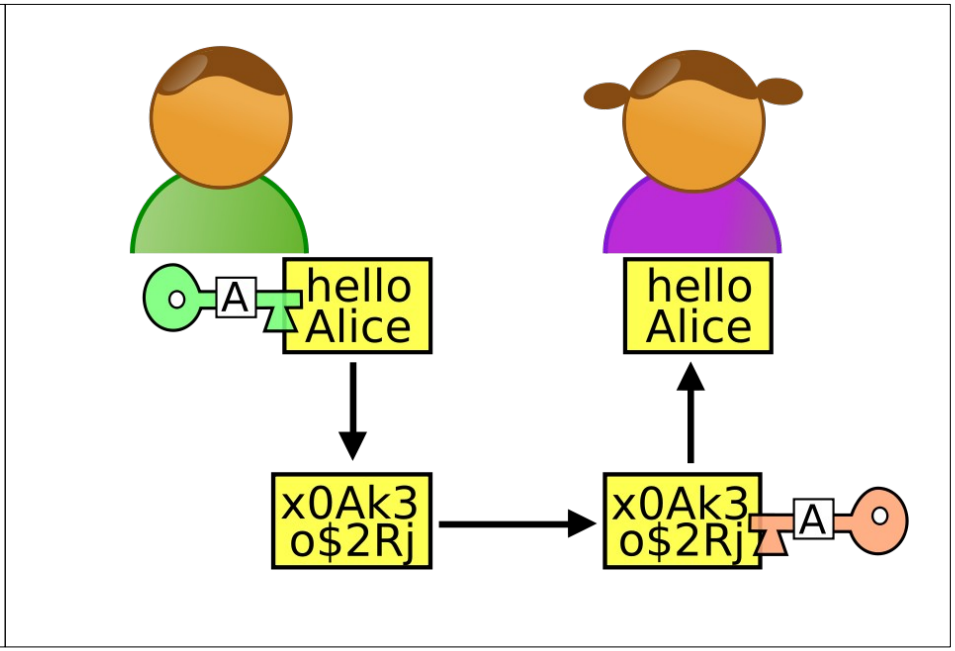
Chiffrement symétrique: nombre et diffusion des clefs

- Pour permettre à n personnes de communiquer,
  - il faut que chaque personne ait les clefs des (n-1) autres.
  - Au total: n(n-1) clefs
- Centrale de clefs:
  - Elle a la clef secrète de chaque utilisateur
  - Les utilisateurs lui envoient les messages
  - Elle les déchiffre et le rechiffre avec la clef privée du destinataire
  - Il faut n clefs
  - La centrale peut lire les messages de tout le monde et doit stocker les clefs de façon sûre.

Principe du chiffrement asymétrique

- La clef de chiffrement est publique
- La clef de déchiffrement est secrète et ne sert qu'au déchiffrement
- La clef publique
  - Ne permet pas de déchiffrer
  - Ne permet pas à un attaquant de trouver la clef privée





Chiffrement à clef publique

- Il est constitué de 3 algorithmes
  - L'algorithme de génération des clefs
    - (pk, sk)=KG(l) produit un couple clef privée, clef publique à partir d'un paramètre de sécurité l (souvent une saisie aléatoire)
  - L'algorithme de chiffrement
    - C=E(M, pk) utilise la clef publique pk
  - L'algorithme de déchiffrement
    - M=D(C, sk) utilise la clef privée sk
  - On a M=D(E(M, pk), sk)
  - On a **parfois** M=E(D(M, sk), pk). Les procédés qui vérifient cette seconde propriété peuvent servir de base à un procédé de signature

Le chiffrement RSA

- Soit deux grands nombres premiers impairs p et q
- n=pq
- Soit e un entier tel que
  - $1<e<\varphi(n)=(p-1)(q-1)$. $\varphi(n)$ est l'ordre du groupe multiplicatif (Z/nZ)*
  - PGCD(e, (p-1)(q-1))=1 (i.-e. e est premier avec $\varphi(n)$)
- Soit d tel que $d=e^{-1} \bmod \varphi(n)$ (d est l'inverse de e dans (Z/nZ)*) donc il existe u tel que $ed+u\varphi(n)=1$
- D'après le théorème d'Euler ($x^{\varphi(n)}=1 \bmod n$)
  - $(m^e)^d=m^{ed}=m^{1-\varphi(n)}=m \bmod n$

RSA:

- Informations publiques :
  - n=pq (p et q premiers)
  - e
- Informations privées:
  - $d=e^{-1} \bmod \varphi(n)$: clef secrète

- Génération de clefs: ((n,e), d)=KG(l)
- Chiffrement: $C=E(m)=m^e \bmod n$
- Déchiffrement: $m=D(C)=c^d=m^{ed}=m \pmod n$

Propriétés de RSA

- Propriété multiplicative
  - Le chiffré d'un produit est égal au produit des chiffrés
    - $(m_1 m_2)^e = m_1^e \, m_2^e = c_1 c_2 \bmod n$
  - Conséquence: attaque à chiffré choisi
    - Soit C un message chiffré (C=E(m), m est inconnu)
    - Supposons qu'on connaisse r et C' tels que
      - On sait déchiffrer C' en m'
      - $C' = r^e C$
    - Alors on sait déchiffrer C car m'=r.m

Propriétés de RSA

- Chiffrement et déchiffrement sont commutatifs
- $m=D(C(m))=m^{ed}=C(D(m))=m^{de}$
- Conséquence: RSA pourra être utilisé comme base d'un algorithme de signature:
  - On signe un document en chiffrant avec la clef privée
  - On le vérifie en déchiffrant avec la clef publique
- Pouvoir être déchiffré avec la clef publique est-il une preuve que ça a été chiffré avec une clef privée donnée ?

RSA: sélection des paramètres

- Choisir p et q de taille similaire pour éviter la factorisation par courbes elliptiques
- Choisir p et q au hasard pour éviter que p et q soit trop proches (sinon p et q de l'ordre de racine de n)
- Choisir p et q des nombres premiers forts
  - p-1 a un grand facteur premier (pour éviter l'algo p-1 de Pollard). grand=au moins 200bits
  - p+1 a un grand facteur premier (pour éviter l'algode Williams
  - r-1 a un grand facteur premier (attaques par cycles)
- Tiré de [Bresson]

Sécurité de RSA

- 2 méthodes d'attaque
  - Trouver d ou p et q : problème de factorisation
  - Trouver m à partir de m^e: extraction de racines e-ièmes
- En pratique, la factorisation est la seule méthode connue pour casser RSA
  - C'est un problème difficile
- Il n'a pas été prouvé que RSA était aussi solide que la factorisation
  - Apparition d'une attaque plus simple que la factorisation possible

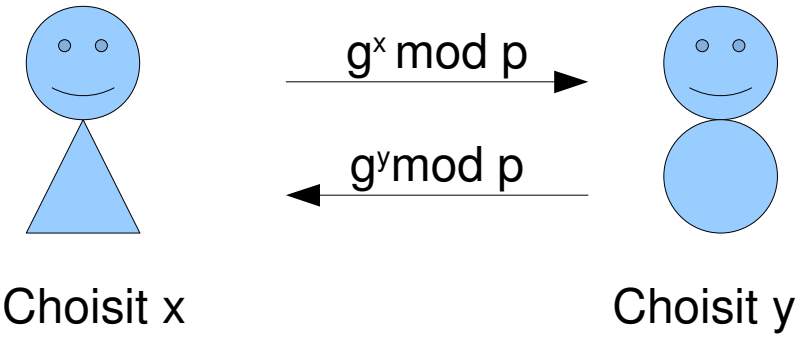
Diffie Hellman

- Années 1970: comment se passer de la transmission d'un secret partagé ?
- Réponses :
  - Chiffrement à clef publique: RSA 1977
  - Diffie Hellman 1976 : se mettre d'accord sur un secret partagé en n'échangeant que des informations publiques

Diffie Hellman: de l'importance la commutativité

- Exemple d'algorithme d'échange de clef :
  - Alice met un secret dans un coffre
  - Alice ferme le coffre avec un cadenas
  - Alice envoie le coffre à Bob
  - Bob ajoute au coffre un cadenas
  - Bob envoie le coffre à Alice
  - Alice retire son cadenas
  - Alice envoie le coffre à Bob
  - Bob retire son cadenas et lit le secret
- Le fait fondamental est que les cadenas peuvent être mis et retirés dans n'importe quel ordre
- Exemple incorrect: Bob met le coffre d'Alice dans un autre coffre fermé à clef par Bob

Diffie Hellman



- Valeurs publiques:
  - p un grand nombre premier
  - G un groupe multiplicatif de cardinal p-1
  - g un générateur de G
- Secret d'Alice: x, secret de Bob: y
- Secret commun: $K=g^{xy}=(g^x)^y=(g^y)^x$
- Attaque: trouver x connaissant g^x
  - Problème du logarithme discret
  - Pas d'attaque raisonnable si p est grand

Diffie Hellman

- L'algorithme ne prévoit pas d'authentification
  - Une attaque MIM est possible
    - La présenter avec le mécanisme des coffres
  - Une version améliorée sera vue en TD