Une école de l'IMT

Implémentation matérielle et logicielle des algorithmes de cryptographie

Guillaume Duc
guillaume.duc@telecom-paristech.fr
2018–2019

Objectifs du cours

- Rappeler les bases d'électronique numérique nécessaires à la compréhension du cours sur les SCA et l'injection de fautes
- Présenter les différentes implémentations matérielles et logicielles des algorithmes de chiffrement

Plan

Plan

Électronique numérique

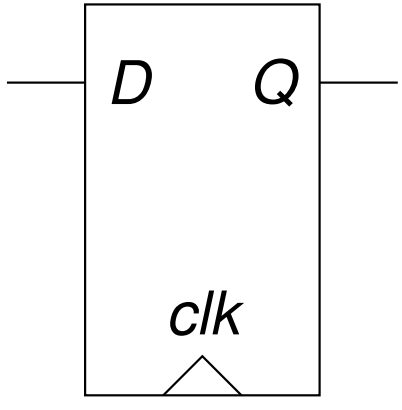
Deux types de logique utilisés dans un circuit

- Logique combinatoire
  - Fonctions ne dépendant pas du temps (les mêmes entrées produisent toujours les mêmes sorties)
  - Portes logiques de base (NON, ET, OU...) combinées en fonctions plus complexes (multiplexeur, décodeur, additionneur...)
  - Permet de faire du calcul (au sens large)
- Logique séquentielle synchrone
  - Fonctions dépendant du temps (mémorisation)
  - Bascules D, registres
  - Permet le séquencement des opérations (contrôle)

La bascule D (*flipflop, dff*)

Élément de base de la logique séquentielle

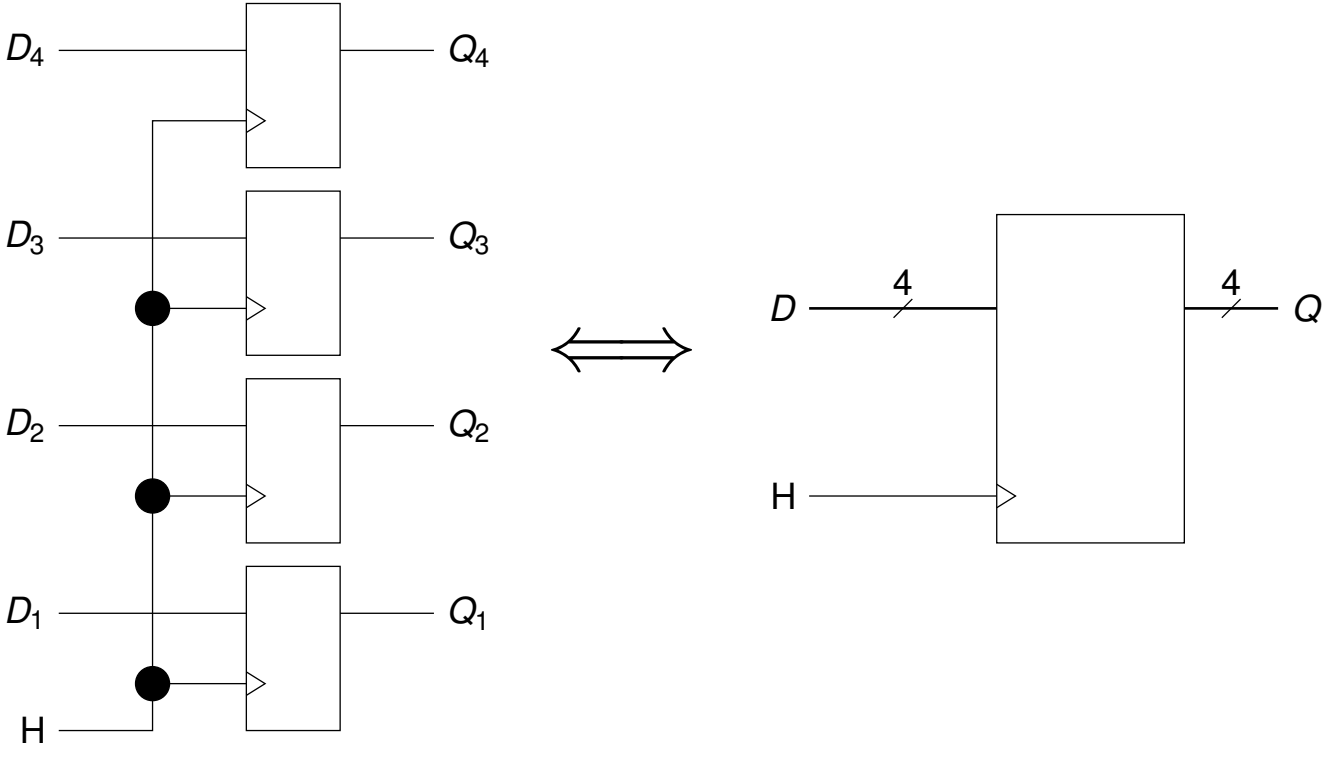
- Fonctionnement
  - À chaque front montant de l'horloge **clk** (passage de 0 $\rightarrow$ 1) l'entrée **D** est copiée (échantillonnée) sur la sortie **Q**
  - Entre deux fronts d'horloge, la sortie **Q** ne change pas (mémorisation)
- Table de vérité



D	clk	Q	
0	$\uparrow$	0	copie de D sur Q
1	$\uparrow$	1	copie de D sur Q
$\times$	0	Q	Q conserve sa valeur
$\times$	1	Q	Q conserve sa valeur
$\times$	$\downarrow$	Q	Q conserve sa valeur

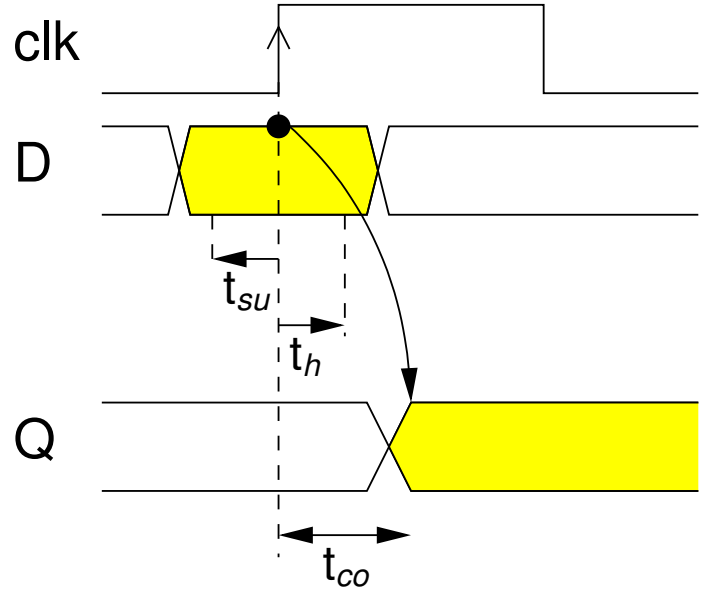
Le registre

- Un registre est un ensemble de bascules D fonctionnant en parallèle
  - Exemple un registre 4 bits



La bascule D

Contraintes temporelles

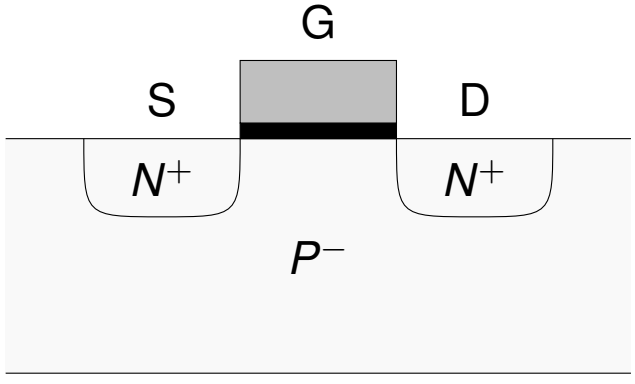


- La donnée doit être stable autour du front d'horloge
  - Elle doit avoir atteint sa valeur t_{su} avant le front d'horloge (temps de pré-positionnement, *setup*)
  - Elle doit être maintenue t_h après le front d'horloge (temps de maintient, *hold*)
- La copie de l'entrée sur la sortie se fait avec un retard de t_{co} (temps de propagation, *clock to output*)

Logique CMOS

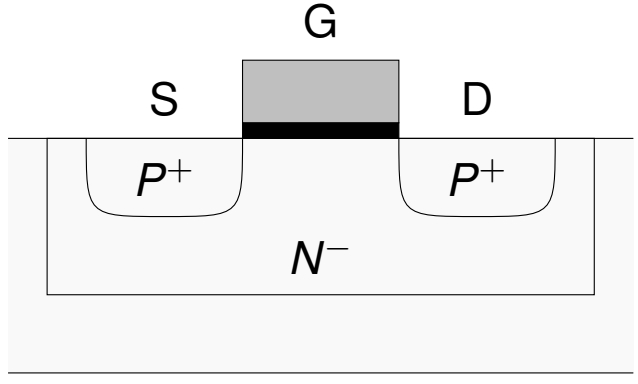
- La très vaste majorité des circuits électroniques numériques est réalisée en logique CMOS (*Complementary Metal Oxide Semiconductor*)
- Une source d'alimentation V_{dd} (5 V, 3,3 V...)
- Par *convention*, on établit un lien entre une tension et un niveau logique correspondant : $0 \equiv V_{ss}$, $1 \equiv V_{dd}$

Logique CMOS



Transistor nMOS

- Canal N
- Passant si $V_{gs} > V_T$

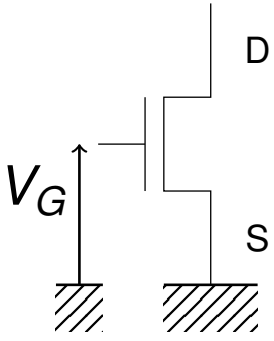


Transistor pMOS

- Canal P
- Passant si $V_{gs} < -|V_T|$

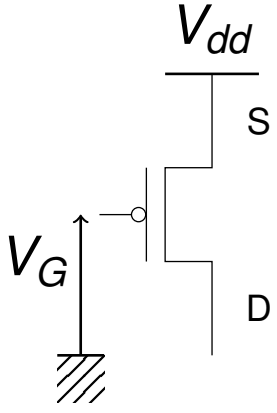
Le transistor MOS

Deux interrupteurs électroniques



Transistor nMOS

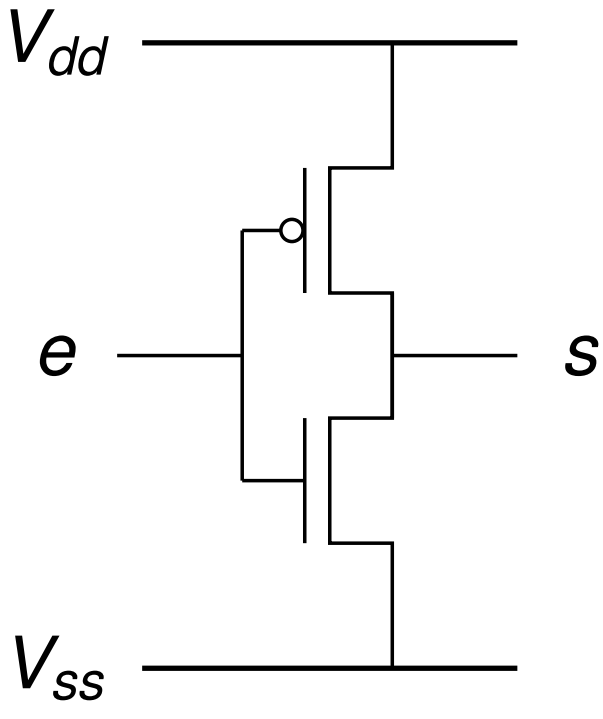
- $V_G = V_{ss}$
  $\Rightarrow$ interrupteur ouvert
- $V_G = V_{dd}$
  $\Rightarrow$ interrupteur fermé



Transistor pMOS

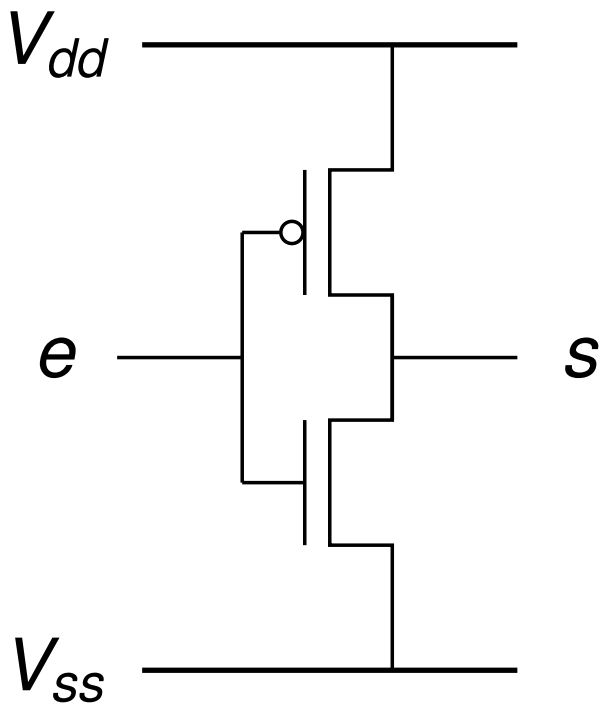
- $V_G = V_{ss}$
  $\Rightarrow$ interrupteur fermé
- $V_G = V_{dd}$
  $\Rightarrow$ interrupteur ouvert

Inverseur CMOS



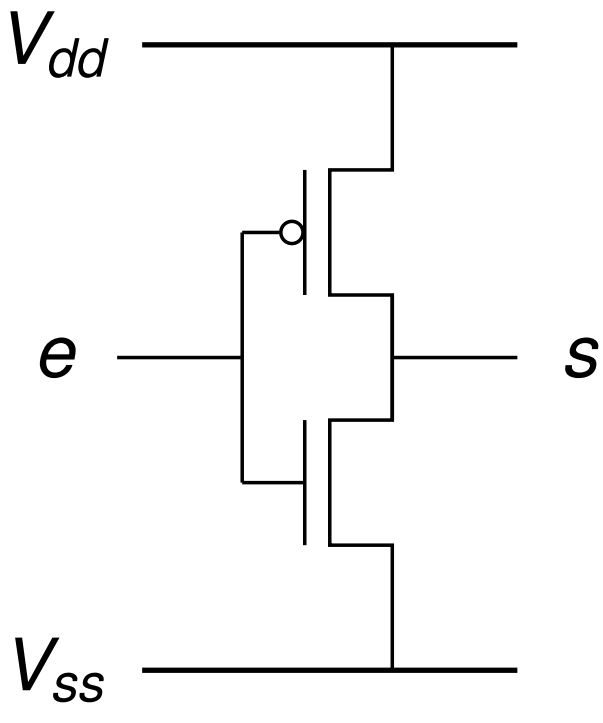
Inverseur CMOS

- Entrée logique $e = 0$
  - $\rightarrow$ $V_e = 0$
    - $\rightarrow$ nMOS bloqué
    - $\rightarrow$ pMOS passant
  - $\rightarrow$ $V_s = V_{dd}$
- $\rightarrow$ Sortie logique $s = 1$



Inverseur CMOS

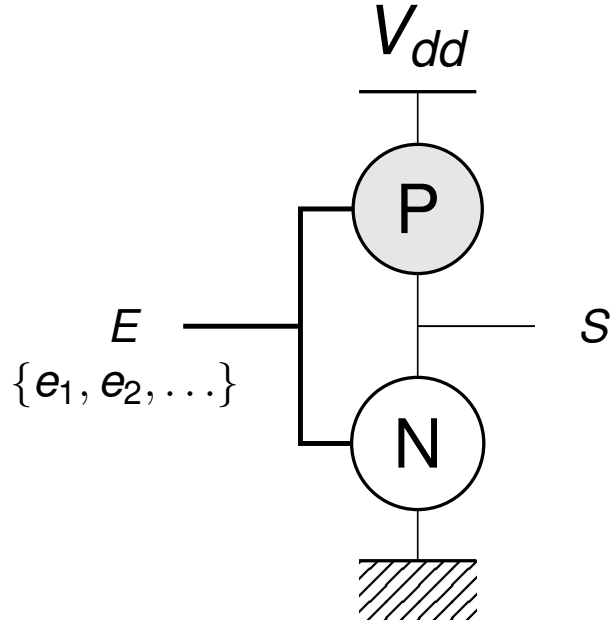
- Entrée logique $e = 0$
  - $\rightarrow$ $V_e = 0$
    - $\rightarrow$ nMOS bloqué
    - $\rightarrow$ pMOS passant
  - $\rightarrow$ $V_s = V_{dd}$
- $\rightarrow$ Sortie logique $s = 1$
- Entrée logique $e = 1$
  - $\rightarrow$ $V_e = V_{dd}$
    - $\rightarrow$ nMOS passant
    - $\rightarrow$ pMOS bloqué
  - $\rightarrow$ $V_s = 0$
- $\rightarrow$ Sortie logique $s = 0$



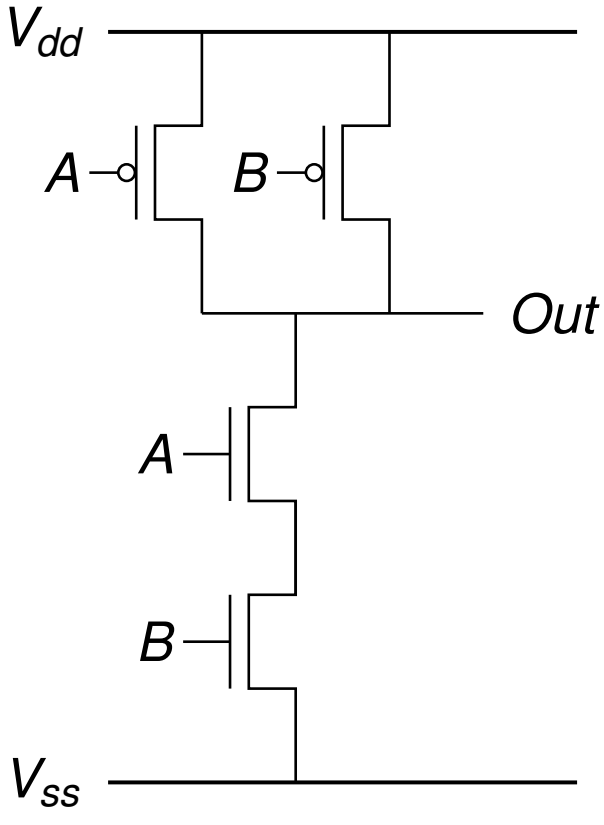
Généralisation à une porte complexe

Une porte avec les entrées $\{e_1, e_2, \ldots\}$ et la sortie S

- Deux réseaux duaux :
  - nMOS : permet la mise à 0
  - pMOS : permet la mise à 1
- Les deux réseau ne doivent jamais être passants en même temps
- Pour que S soit une fonction logique :
  - Si N est passant P bloqué
  - Si P est passant N bloqué



La porte non-et (NAND)



CMOS & Consommation

- Lorsque la sortie d'une porte ne change pas d'état, pas de consommation $\rightsquigarrow$ *consommation statique nulle* (ou quasiment)
- Lorsque la sortie d'une porte change d'état (passage $0 \rightarrow 1$ ou $1 \rightarrow 0$), la porte consomme $\rightsquigarrow$ *consommation dynamique non nulle*
  - Cette consommation est liée au chargement/déchargement des capacités parasites connectées en sortie de la porte (fils et grilles des transistors connectés)
- Donc la consommation d'un circuit est liée au nombre de sorties de portes changeant d'état
- Cette propriété peut être exploitée pour retrouver des secrets manipulés par le circuit (voir cours sur les SCA)

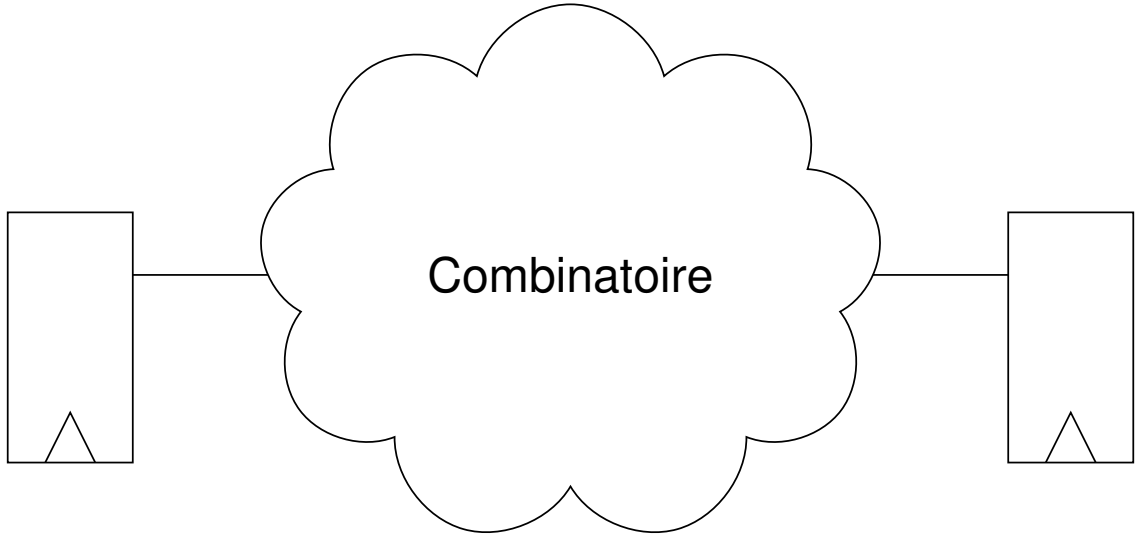
CMOS & Temps de propagation

- Lorsque les entrées d'une porte logique changent, il faut un certain temps pour que sa sortie reflète le nouvel état des entrées
- Ce temps est appelé *temps de propagation*
- Il est également lié au chargement/déchargement des capacités parasites en sortie des portes
- Pendant ce temps, la sortie peut être fausse (vis-à-vis de la fonction réalisée et de l'état actuel des entrées) et ne doit donc pas être prise en compte

Lien combinatoire/séquentielle

- Pour pouvoir utiliser un bloc de logique combinatoire, il est donc nécessaire de maintenir stables ses entrées suffisamment longtemps
- De plus ces entrées sont souvent le résultats d'un autre bloc combinatoire
- Les différents blocs combinatoires d'un circuit sont donc encadrés par des registres (bascules D) qui échantillonnent le résultat d'un bloc et le présentent de façon stable (mémorisation) en entrée du bloc suivant

Fréquence de fonctionnement d'un circuit



- Parties combinatoires encadrées par des registres
- Pour assurer un bon fonctionnement $T_{clk} > t_{co} + t_{crit} + t_{su}$ où t_{crit} est le temps de propagation dans le plus long chemin combinatoire (chemin critique)
- Si cette contrainte n'est pas respectée $\rightsquigarrow$ fautes

Évolution du temps de propagation

- Pour un circuit fixe, le temps de propagation sur le chemin critique augmente (et donc la fréquence maximale de fonctionnement du circuit diminue) avec :
  - L'augmentation de la température
  - La diminution de la tension d'alimentation
- Une partie des attaques par injection de faute vise à obtenir une violation du chemin critique, soit en augmentant la fréquence de fonctionnement, soit en augmentant la durée du chemin critique

Plan

Cryptologie

- La science du secret
- Se divise en deux domaines
  - *Cryptographie* qui vise à protéger des messages
  - *Cryptanalyse* qui vise à analyser (casser) des messages protégés
- Les algorithmes de cryptographie vont être utilisés pour garantir différentes propriétés de sécurité (principalement confidentialité, intégrité, authentification) sur les informations sensibles d'un système pendant leur stockage, leur traitement et leur transmission

Cryptographie

Taxonomie des algorithmes

- Algorithmes sans clé
  - Fonction de hachage (MD5, SHA1, SHA2, SHA3...)
  - Générateur de nombres aléatoires
- Algorithmes à clé symétrique
  - Chiffrement
    - Par bloc (DES, AES...)
    - Par flux/flot (RC4...)
  - Fonction de hachage à clé ou MAC (*Message Authentication Code*)
- Algorithmes à clé publique (asymétrique)
  - Chiffrement (RSA...)
  - Signature (RSA, DSA...)

Plan

Plan

Chiffrement

Généralités

$$E : \{0,1\}^n \times \{0,1\}^{k_e} \to \{0,1\}^m$$
$$P, K_e \mapsto C = E_{K_e}(P)$$
$$D : \{0,1\}^m \times \{0,1\}^{k_d} \to \{0,1\}^n$$
$$C, K_d \mapsto P = D_{K_d}(C)$$

- P est le *message en clair* (*plaintext*)
- C est le *message chiffré* (*ciphertext*)
- K_e est le secret (*clé*) utilisé pour le chiffrement
- K_d est le secret (clé) utilisé pour le déchiffrement
- E est la *fonction de chiffrement*
- D est la *fonction de déchiffrement* (notée aussi E^{-1})
- Si $K_e = K_d$ on parle de cryptographie/chiffrement *symétrique* ($K = K_e$ est appelée *clé secrète*)
- Si $K_e \neq K_d$ on parle de cryptographie/chiffrement *asymétrique* (K_e est appelée clé publique et K_d est appelée *clé privée*)

Chiffrement

Généralités

- Pour un bon algorithme de chiffrement, étant donné C, il doit être impossible pour un adversaire ne connaissant pas la clé K_d de retrouver P tel que $P = D_{K_d}(C)$
- De même, connaissant $P_1, \ldots, P_N$ et $C_1, \ldots, C_N$ tels que $P_i = D_{K_d}(C_i)$, il doit être impossible pour un adversaire de retrouver K_d
- *Principe de Kerckhoffs* [4, 5] : la sécurité ne repose pas sur le fait que E et D soient secrets (les algorithmes sont publics ou ont vocation à le devenir)
- Les algorithmes de chiffrement sont principalement utilisés pour garantir la confidentialité d'une information

Chiffrement

Exemple (cryptographie symétrique)

- Alice et Bernard veulent échanger des messages confidentiels sur un canal de communication non sécurisé
  - Alice et Bernard disposent tout les deux d'un secret partagé K avant le début de la communication
  - Alice chiffre son message m avec la clé K : $c = E_K(m)$ et envoie c à Bernard
  - Bernard déchiffre c : $E_K^{-1}(c) = E_K^{-1}(E_K(m)) = m$
  - Ève qui espionne la communication récupère c mais ne peut pas retrouver m car elle ne connaît pas la clé K
- Inconvénient : nécessité d'un secret partagé (problèmes : établissement et nombre de secret à gérer)
- Avantage : les algorithmes de chiffrement symétriques sont souvent rapides

Chiffrement

Exemple (cryptographie asymétrique)

- Alice et Bernard veulent échanger des messages confidentiels sur un canal de communication non sécurisé
  - Alice possède la clé publique K_e^B de Bernard et seul ce dernier possède la clé privée correspondante K_d^B
  - Alice chiffre son message m : $c = E_{K_e^B}(m)$ et envoie c à Bernard
  - Bernard déchiffre c : $D_{K_d^B}(c) = m$
  - Ève qui espionne la communication récupère c mais ne peut pas retrouver m car elle ne connaît pas la clé K_d^B
- Avantage : pas de secrets partagés
- Inconvénients : les algorithmes de chiffrement asymétriques sont souvent lents, comment prouver que l'on possède bien la clé publique de son correspondant et pas celle d'un adversaire (*man-in-the-middle*) ?

Taxonomie des algorithmes de chiffrement

- Algorithmes de chiffrement symétriques
  - Algorithmes de *chiffrement par bloc* (*block cipher*, exemples : AES, DES)
  - Algorithmes de *chiffrement par flot* (*stream cipher*, exemple : RC4)
- Algorithmes de chiffrement asymétriques (exemple : RSA)

Algorithme de chiffrement par bloc

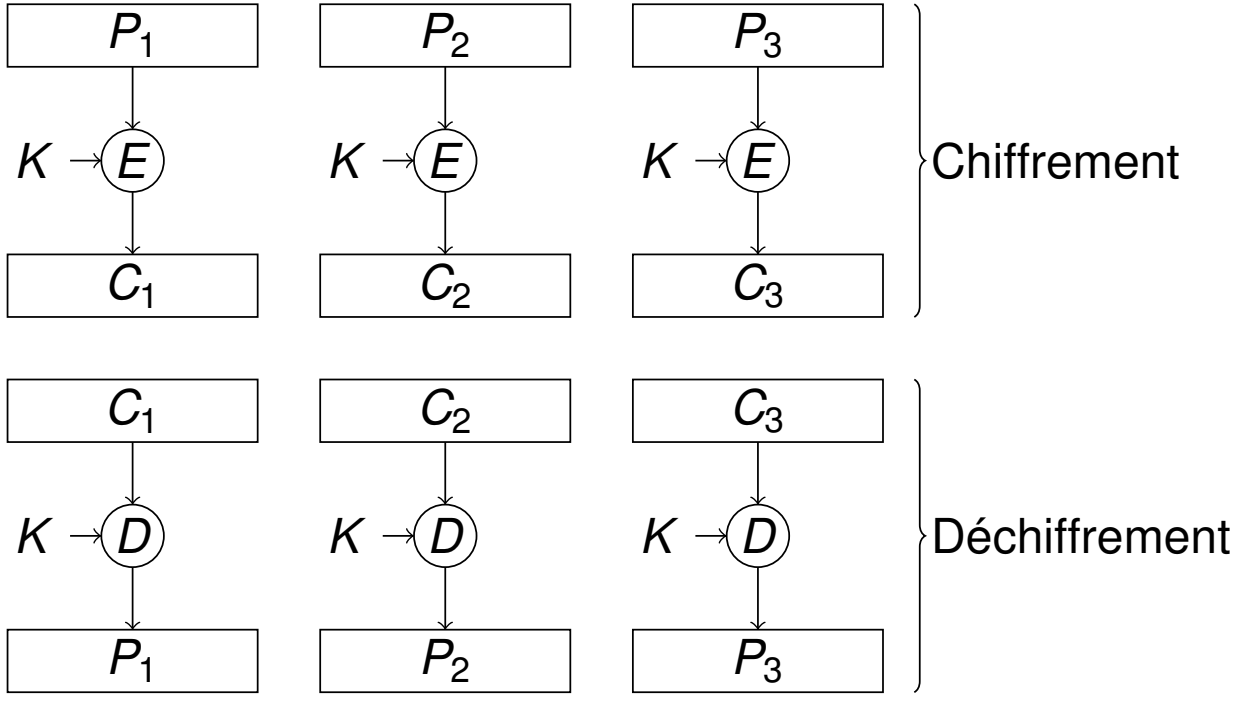
Block cipher

$$E : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$$
$$P, K \mapsto C = E_K(P)$$

- n est la taille d'un bloc (en général 64 ou 128 bits)
- k la taille de la clé (en général 128 ou 256 bits)
- Si on veut chiffrer des messages dont la taille est différente de n, il faut utiliser un *mode d'opération*

Modes d'opération usuels

ECB — *Electronic CodeBook*



Modes d'opération usuels

ECB — *Electronic CodeBook*

Image originale

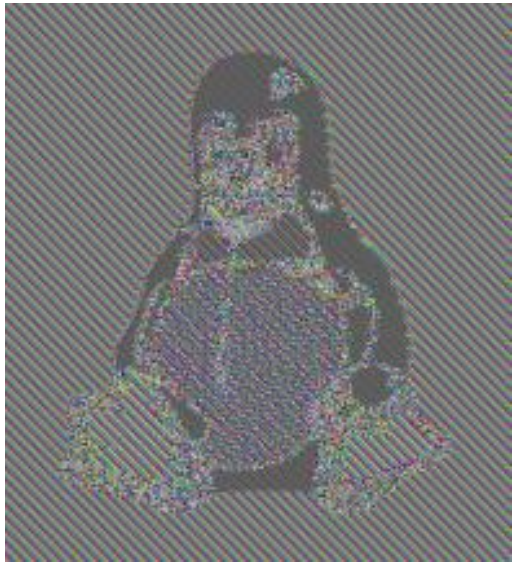


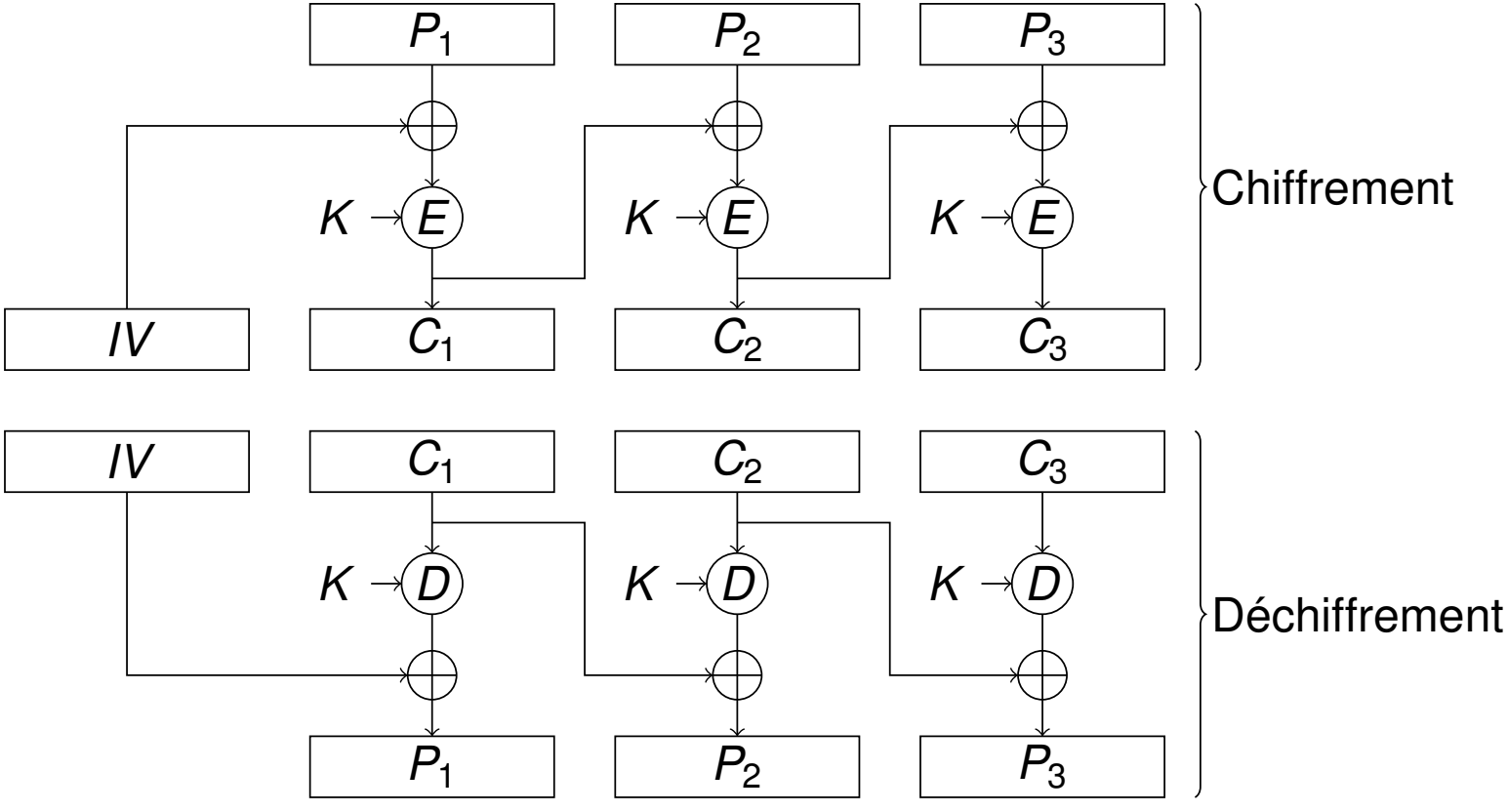
Image chiffrée avec ECB

- Problème de sécurité avec ce mode (identification de motifs)

Crédits images : Larry Ewing, Lunkwill

Modes d'opération usuels

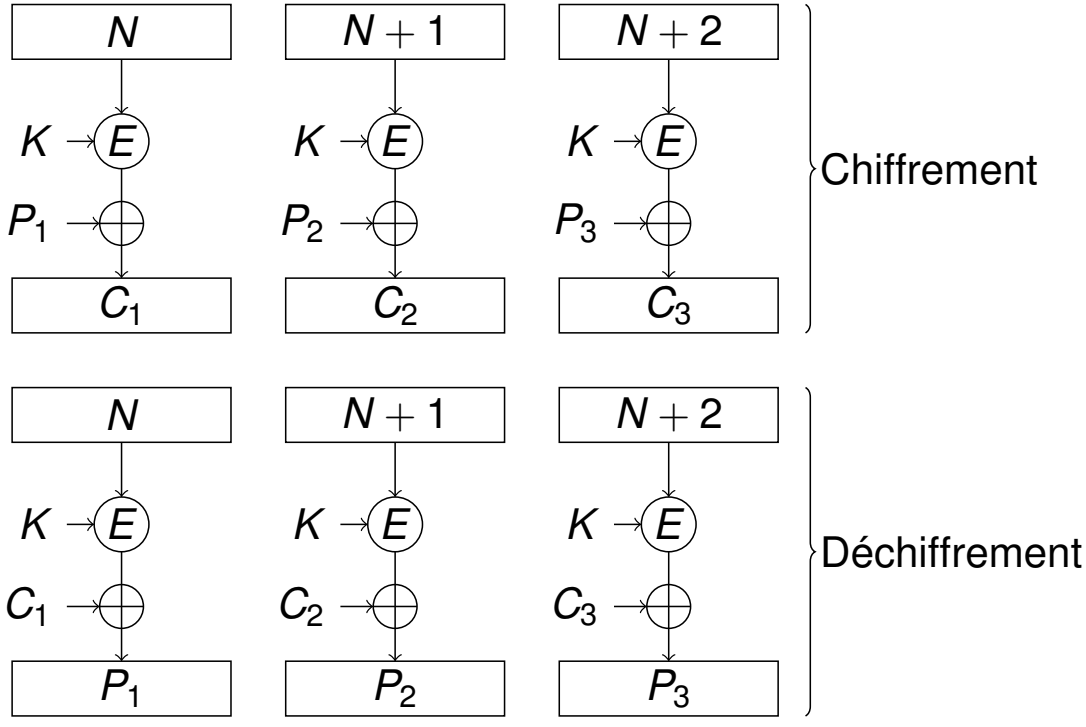
CBC — *Cipher Block Chaining*



- IV (*Initialization Vector*) est généré aléatoirement pour chaque message chiffré et transmis (en clair)

Modes d'opération usuels

CTR — *Counter Mode*



- N est un nombre aléatoire, transmis en clair avec le message
- On ne doit jamais chiffrer deux blocs avec le même compteur

Choix d'un modes d'opération

- Sécurité
  - ECB est déconseillé (identification de motifs)
  - CBC est vulnérable lorsqu'une collision survient (après en moyenne $2^{n/2}$ où n est la taille d'un bloc en bits)
  - Mode compteur : ne jamais chiffrer deux blocs avec le même compteur
- Performances
  - ECB et compteur : entièrement parallélisables
  - CBC : déchiffrement parallélisable uniquement

Blocs de base d'un algo de chiffrement par bloc

- Selon Shannon [9], un algorithme de chiffrement est basé sur deux propriétés : la *diffusion* et la *confusion*
- Confusion : rendre la relation entre la clé symétrique et le texte chiffré la plus complexe possible
- Diffusion : la redondance statistique sur le texte en clair doit être dissipée dans les statistiques du texte chiffré (les statistiques du texte chiffré doivent donner le moins d'informations possible sur le texte clair)
  - L'inversion d'un bit en entrée doit changer chaque bit en sortie avec une probabilité $1/2$

Blocs de base d'un algo de chiffrement par bloc

- La confusion est généralement réalisée par des tables de substitution (*Substitution Box (S-Box)*) qui permettent d'introduire de la non linéarité dans les calculs
- La diffusion est réalisée de diverses façons
  - DES : Permutations
  - AES : Opérations ShiftRows et MixColumns

Tables de substitution (*S-Box*)

- Fonction logique *n* bits vers *m* bits fortement non linéaire
  - DES : 8 fonctions différentes 6 bits vers 4 bits
  - AES : 1 fonction 8 bits vers 8 bits
- Implémentation logicielle
  - Table en mémoire
- Implémentation matérielle
  - Logique (portes ou LUT sur un FPGA)
  - Table en ROM

Permutations

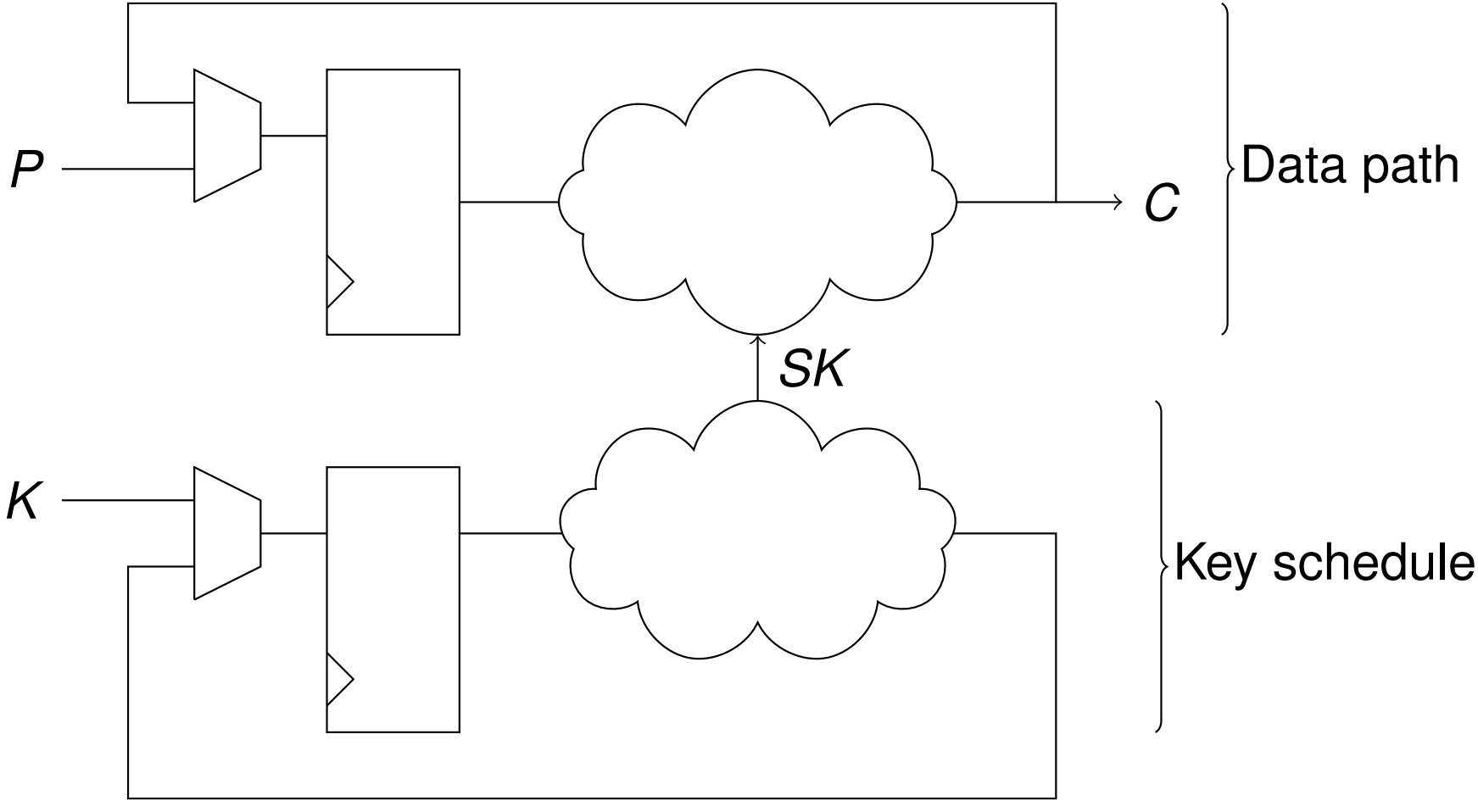
- Permutations des bits à l'intérieur d'un mot (ex. permutations de DES)
  - Implémentation matérielle triviale
  - Implémentation logicielle beaucoup plus coûteuse (complexité en $\mathcal{O}(n)$ où n est la taille (en bit) du mot) car les processeurs généralistes ne disposent pas d'instructions pour la permutation de bits
- Permutations de mots (ex. ShiftRows d'AES)
  - Implémentation triviale en matériel ou en logiciel (adressage)

Schéma général

- Les algorithmes classiques de chiffrement par bloc présentent une structure itérative
- Opération de base (appelée tour / ronde / *round*) répétée un certain nombre de fois (16 pour DES, 10, 12 ou 14 pour AES)
- Chaque ronde utilise une sous-clé dérivée de la clé principale grâce à un algorithme baptisé *key schedule*
- Structure idéale pour une implémentation matérielle itérative

Implémentation matérielle itérative

Schéma général



Implémentation matérielle itérative

Problèmes

- Chemin critique souvent long (génération de la sous-clé en parallèle avec la première partie du chemin de données puis addition de la sous-clé et seconde partie du chemin de données)
  - Possibilité, en ajoutant un cycle, de calculer en parallèle le calcul de la ronde n et celui de la sous-clé nécessaire à la ronde $n+1$, et donc de réduire le chemin critique
- Nécessite d'appliquer les boites S en parallèle (duplication du matériel, obligatoire pour DES puisque les boites sont différentes, mais pour AES, duplication 16 fois de la boite S)
  - Pour AES, possibilité de traiter colonne par colonne et ainsi réduire l'utilisation matérielle (voir plus loin)

Implémentation matérielle itérative

Problèmes

- Fuite d'information importante à des moment précis (fronts d'horloge) lors de la mise à jour des registres d'état (clé et données)
- Ces registres contiennent des variables internes de l'algorithme qui ne sont pas censées être accessibles par l'adversaire
- Voir cours sur les SCA

Implémentation matérielle pipelinée

- Moyennant la duplication du matériel nécessaire pour faire le calcul d'une ronde, il est possible de pipeliner l'algorithme
- Le pipeline ne réduit pas le temps nécessaire (latence) pour chiffrer un bloc mais augmente le débit (nombre de blocs chiffrés pendant une période de temps donnée)
- Cependant, certains modes d'opération se prêtent mal à la parallélisation (comme par exemple le mode CBC en chiffrement)

Implémentation logicielle pipelinée

- Utiliser les instructions SIMD (*Single Instruction Multiple Data*, comme AVX, MMX, SSE, etc.) pour pipeliner de façon logicielle [2]
- Découpage des entrées (plusieurs blocs à chiffrer) en sous-blocs et traitement en parallèle de ces sous-blocs grâce aux opérateurs SIMD
- Possibilité de découper jusqu'à une granularité du bit (si les opérateurs SIMD le supportent) et ainsi résolution du problème des permutations de bits par simple adressage

Plan

Introduction

- Standardisé en 1977 par le *National Bureau of Standards* américain (FIPS PUB 46)
- Basé sur l'algorithme Lucifer proposé par IBM et modifié par la NSA (*National Security Agency*)
- Traite des blocs de 64 bits
- Clés de 56 bits (et non 64 !)
- Schéma de Feistel à 16 rondes

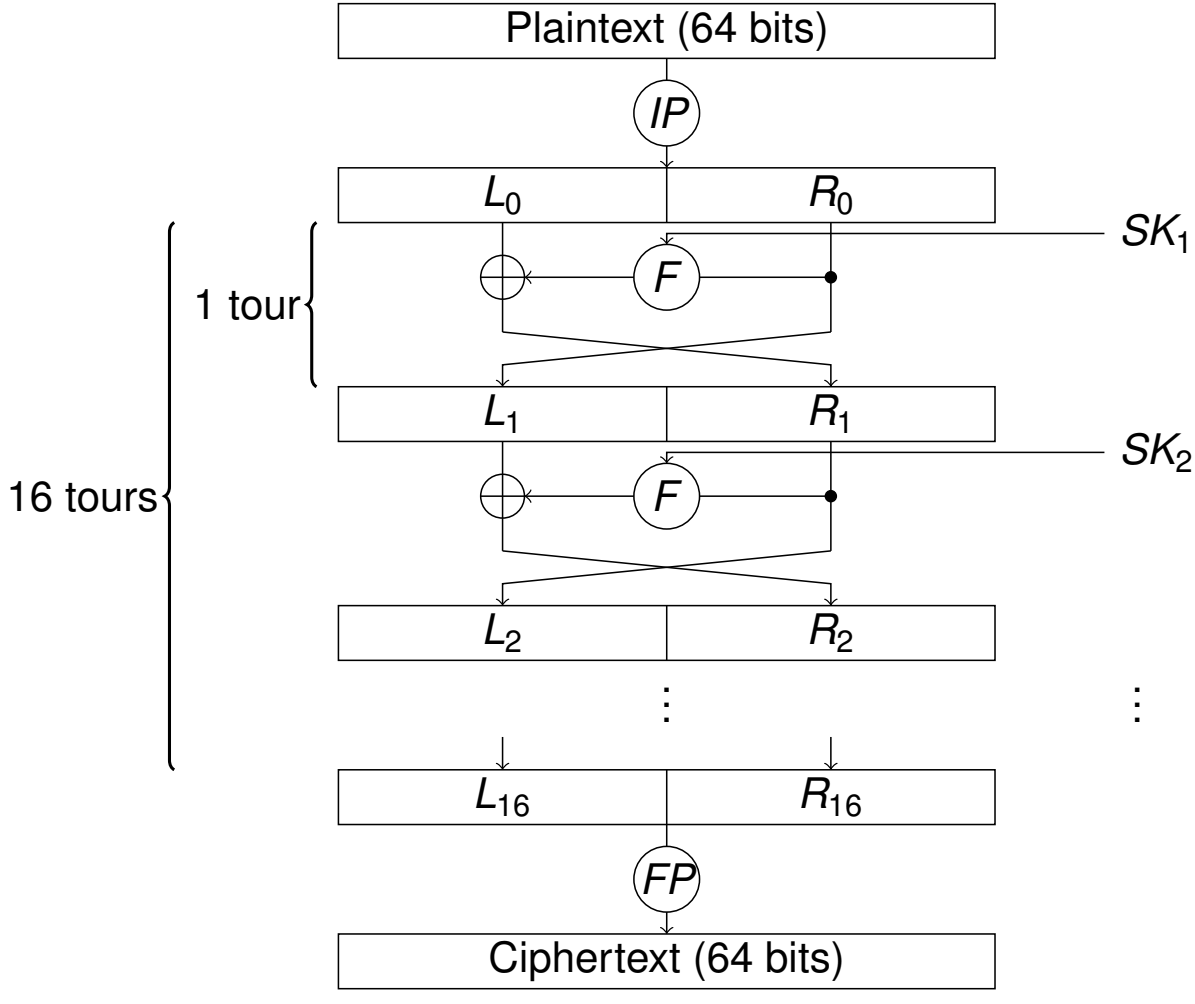
Sécurité (1/2)

- 4 clés faibles et 12 clés semi-faibles à éviter
  - Clé faible : $E_K(E_K(M)) = M$
  - Clés semi-faibles : $E_{K_1}(E_{k_2}(M)) = M$
- Cryptanalyse différentielle : résistant (grâce aux modifications apportées aux *S-Box*) [1]
- Cryptanalyse linéaire : nécessite 2^{43} clairs connus [7] $\rightsquigarrow$ Impraticable
- Force brute : taille de la clé trop faible (56 bits) pour les puissances de calcul actuelles $\rightsquigarrow$ Machine RIVYERA [8] (128 FPGA) teste toutes les clés (2^{56}) en moins de 24 heures

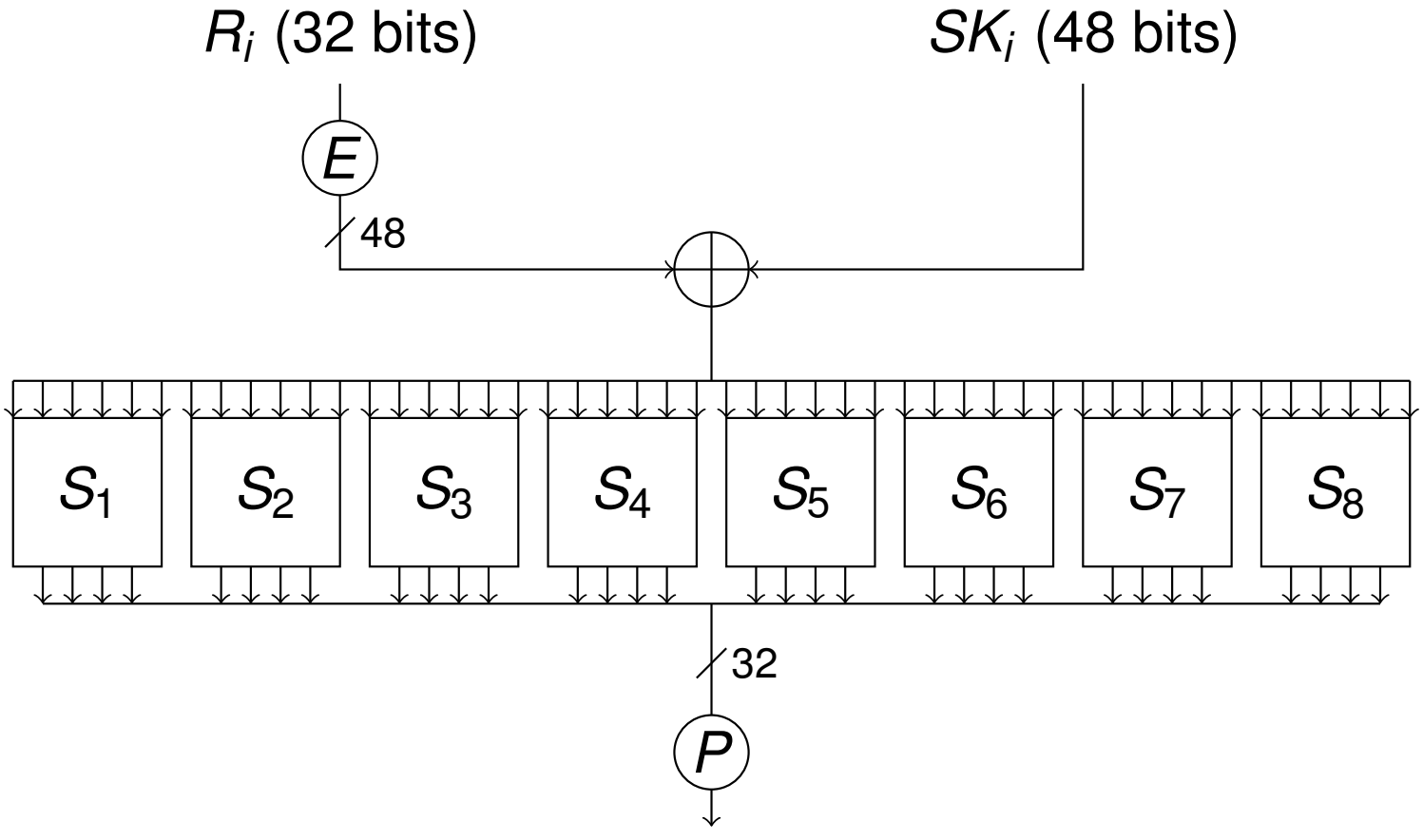
Sécurité (2/2)

- DES n'est plus utilisé qu'en mode EDE (Triple DES) : $C = E_{K_3}(E_{K_2}^{-1}(E_{K_1}(P)))$
  - Clé de 168 bits (3×56) mais sécurité effective de 112 bits (attaque *meet-in-the-middle*)
  - Toujours utilisé, particulièrement dans les applications bancaires (ex. standard EMV)

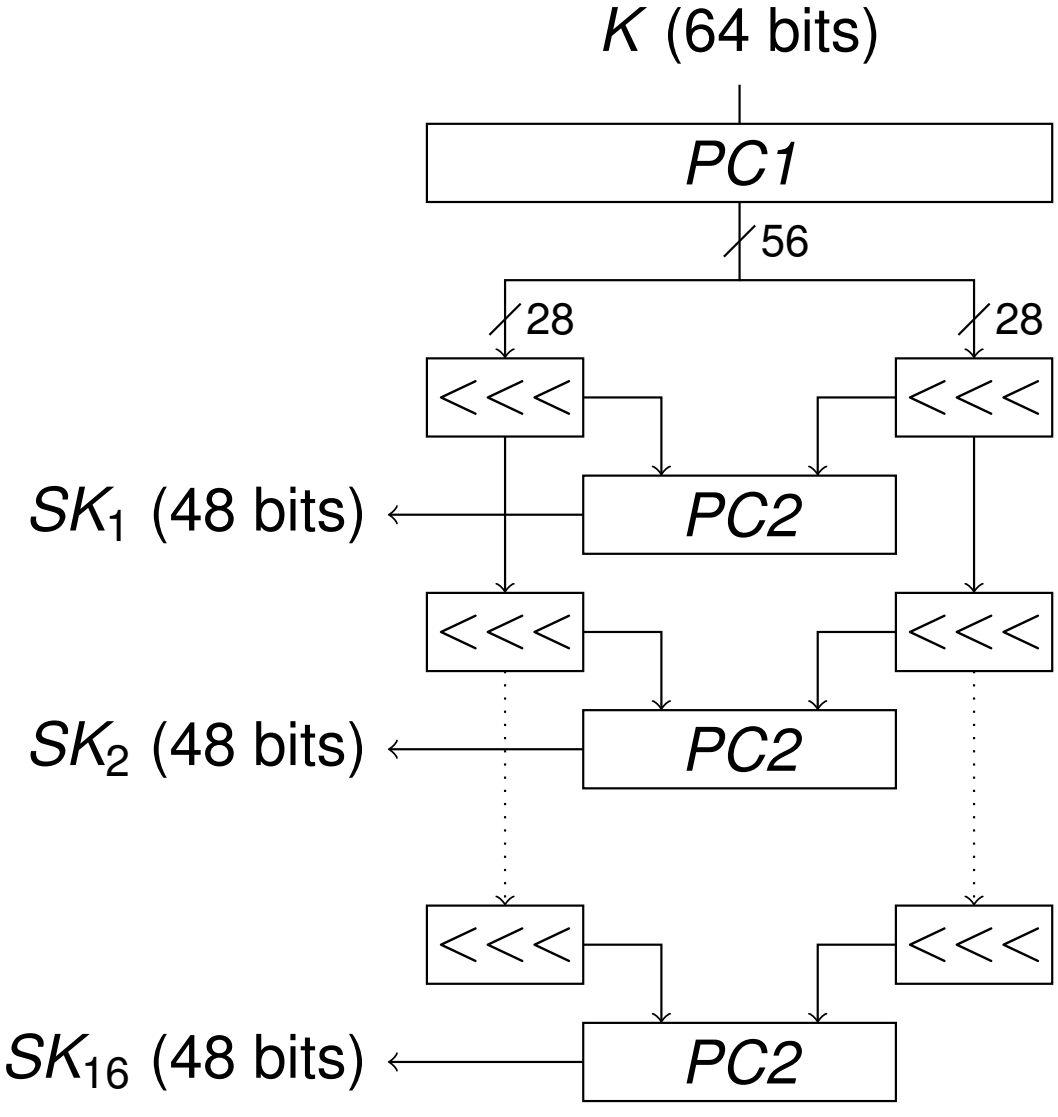
Schéma de Feistel



Fonction de Feistel (DES)



Key Scheduling (DES)



Implémentation logicielle

- Algorithme de *Key Schedule* complexe mais réalisé peu souvent (on chiffre beaucoup de données avec la même clé donc on peut calculer une fois les sous-clés puis les conserver pour les chiffrements ultérieurs)
- Partie la plus coûteuse : les permutations, notamment au sein de la fonction de tour (réalisées 16 fois)
- Optimisation : combiner la permutation *P* avec les *S-Box*
  - Au lieu de 8 *S-Box* 6 vers 4 bits, on utilise 8 tables 6 bits vers 32 bits (les 4 bits du résultat de la *S-Box* sont positionnés au “bon” endroit dans le mot de 32 bits et les 28 autres bits sont à 0)
  - Il suffit alors simplement de faire un OU bit-à-bit entre les 8 mots de 32 bits issus de la consultation des tables

Implémentation logicielle

- La fonction d'expansion (E), du fait de sa structure, est assez simple à calculer
- En effet, 6 des 8 entrées des S-Box (sur 6 bits) sont simplement 6 bits consécutifs de R_i XORé avec 6 bits consécutifs de la sous-clé (donc un décalage suffit)
- Seul les deux blocs de 6 bits extrêmes nécessitent une opération supplémentaire (récupération d'un bit à l'extrémité opposée de R_i)

Plan

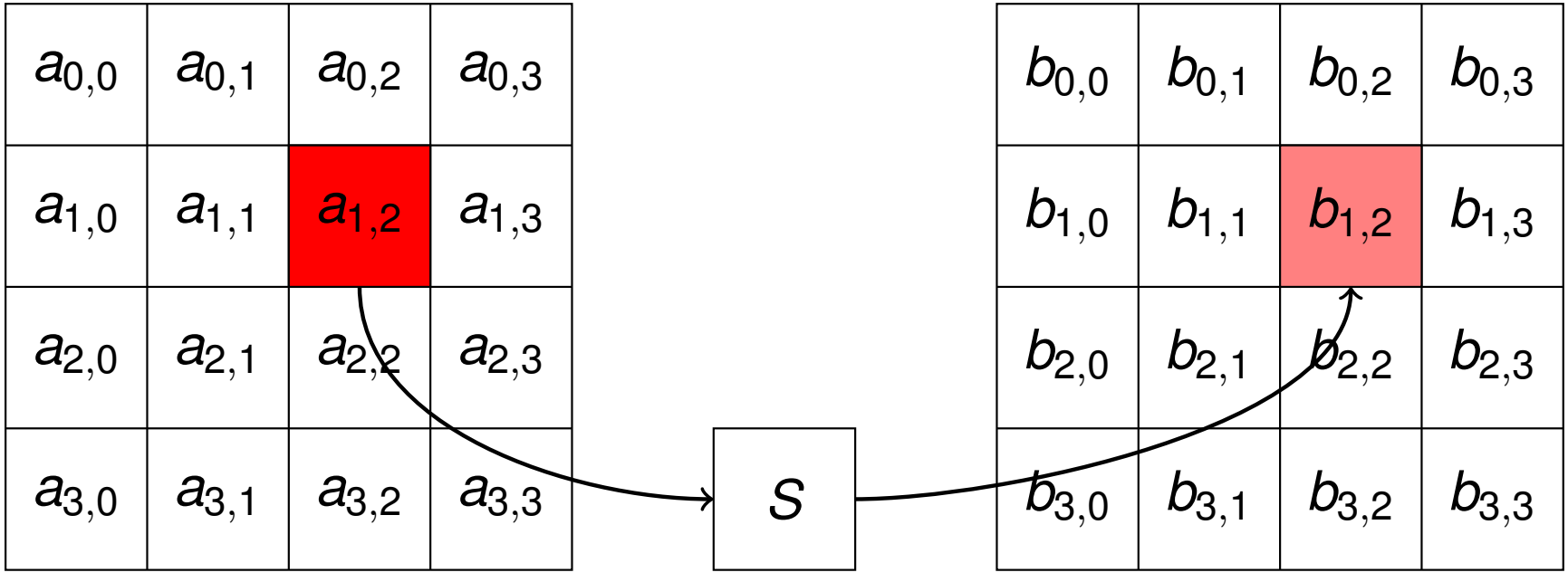
Introduction

- Standardisé par le NIST (*National Institute of Standards and Technology*) en 2001 après un concours international
- Algorithme original : Rijndael de Joan DAEMEN et Vincent RIJMEN (Belgique)
- Traite des blocs de 128 bits (16 octets)
- Clés de 128, 192 ou 256 bits
- 10, 12 ou 14 tours en fonction de la taille de la clé
- Architecture SPN (*Substitution Permutation Network*)

Opérations

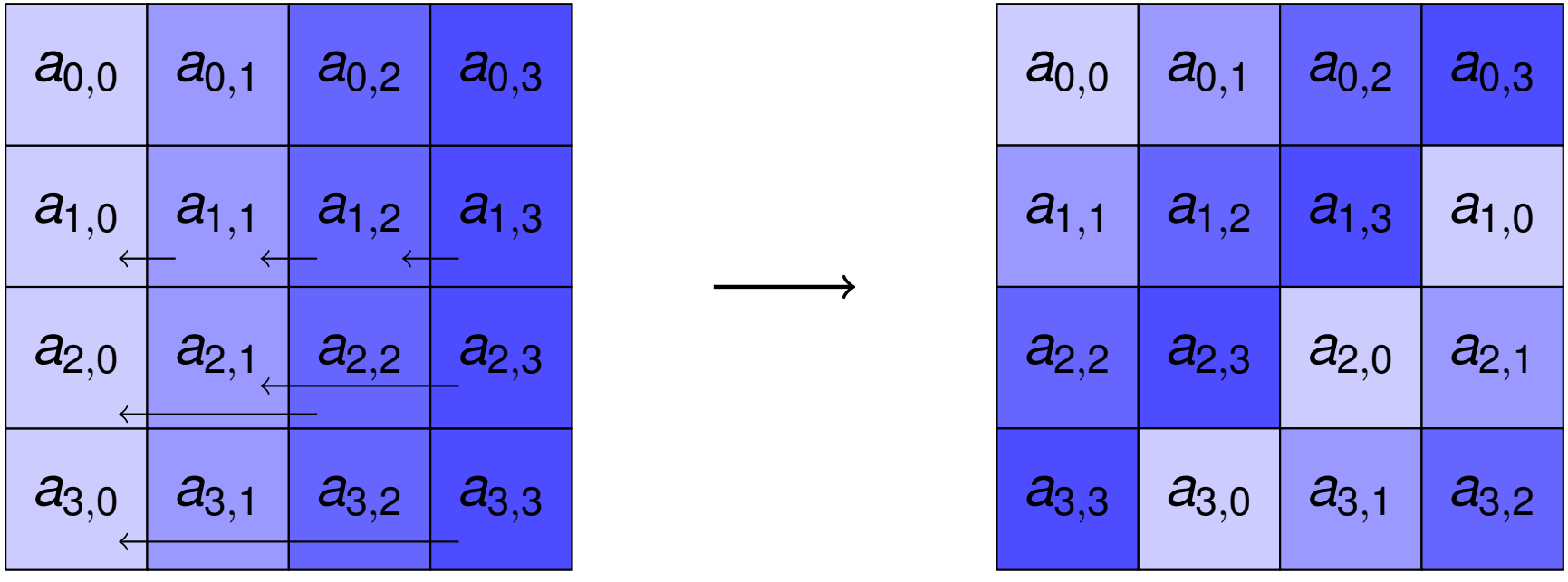
- Tour initial
  - AddRoundKey
- Tours suivants
  - SubBytes
  - ShiftRows
  - MixColumns
  - AddRoundKey
- Tour final
  - SubBytes
  - ShiftRows
  - AddRoundKey

Opération SubBytes

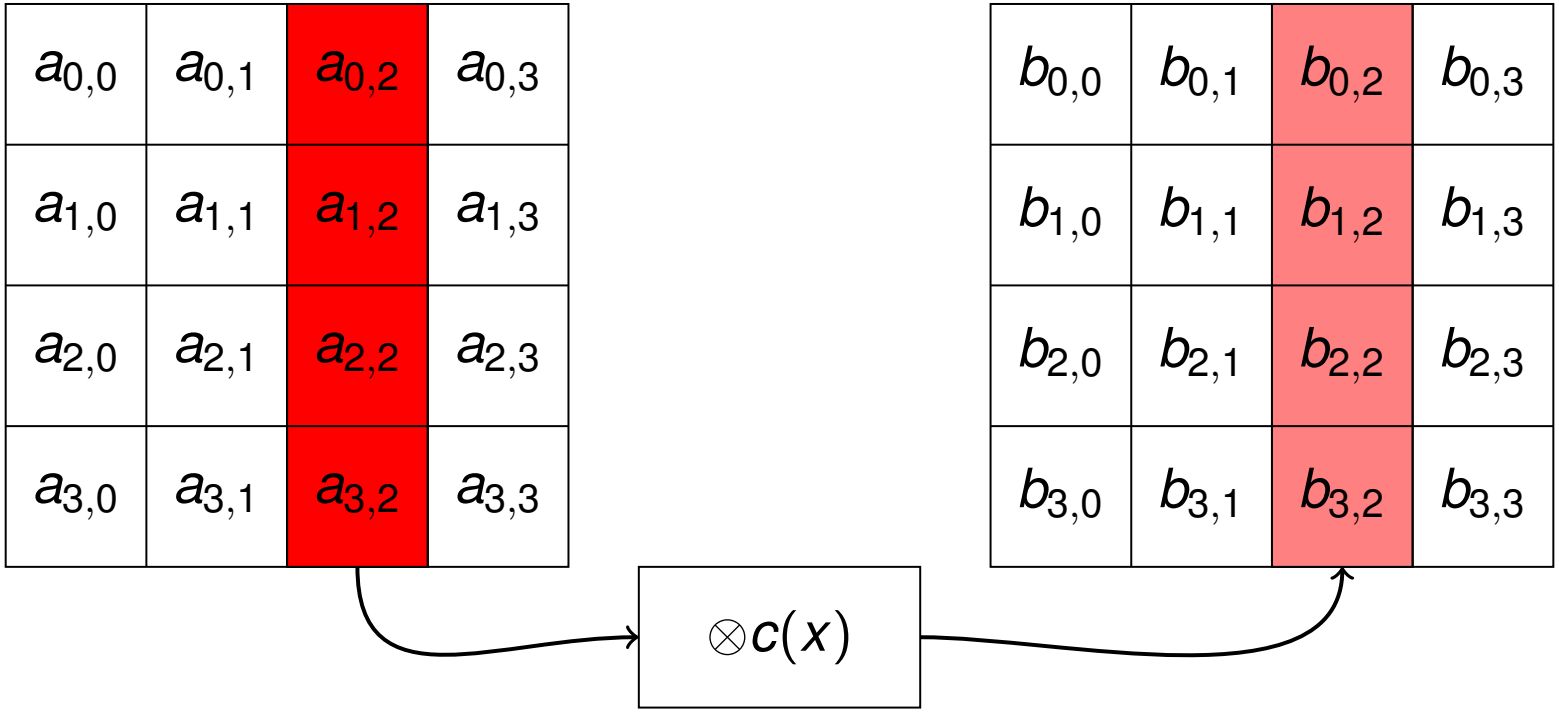


- $b_{i,j} = S(a_{i,j})$

Opération ShiftRows

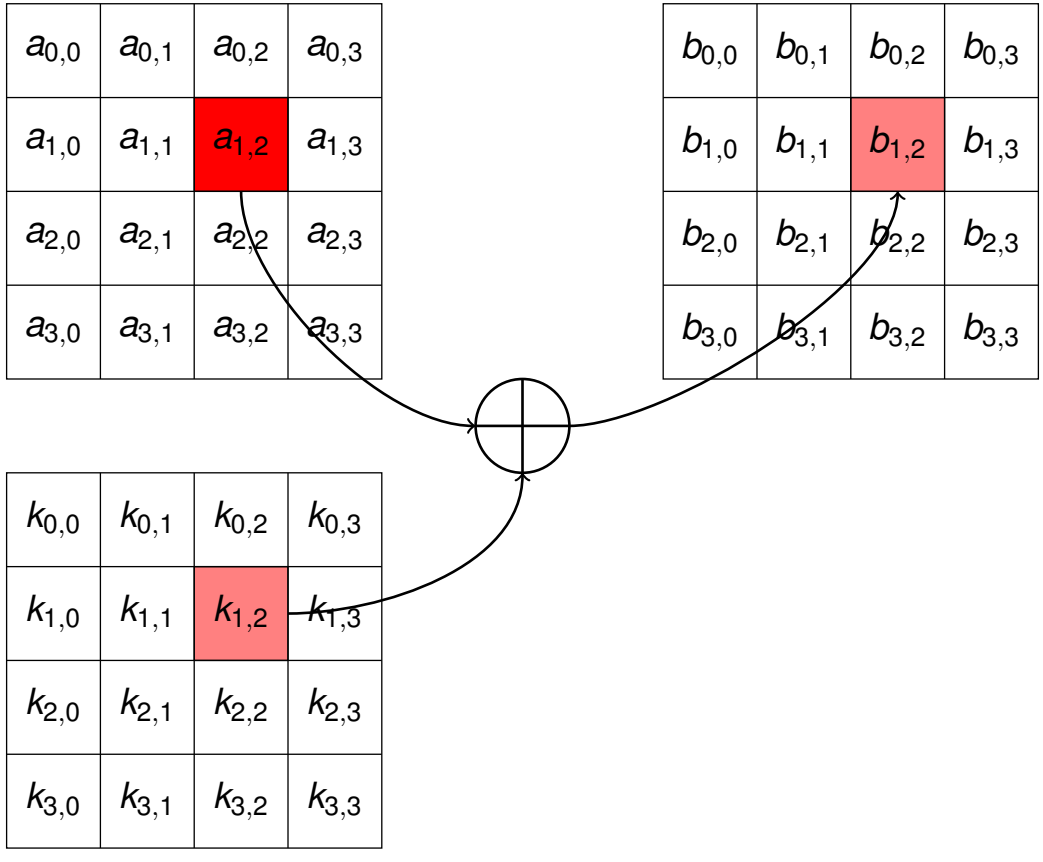


Opération MixColumns



- Multiplication de chaque colonne par un polynôme (dans un corps particulier)
- Permet la diffusion

Opération AddRoundKey



- $b_{i,j} = a_{i,j} \oplus k_{i,j}$

Implémentation matérielle

Remarques

- Possibilité de traiter colonne par colonne (nécessite 4 fois plus de cycles mais limite la duplication du matériel des opérations SubBytes et MixColumns)

Implémentation logicielle

AES instruction set

- Introduit par Intel en 2008 (AES-NI (New Instructions), aussi disponible chez AMD)
- Permet d'accélérer le calcul logiciel d'AES en proposant les instructions suivantes [3]
  - `AESENC` Effectue une ronde de d'AES (ShiftRows, SubBytes, MixColumns & AddRoundKey)
  - `AESENCLAST` Effectue la dernière ronde d'AES (ShiftRows, SubBytes & AddRoundKey)
  - `AESDEC` Effectue une ronde de déchiffrement d'AES (InvShiftRows, InvSubBytes, InvMixColumns & AddRoundKey)
  - `AESDECLAST` Effectue la dernière ronde de déchiffrement (InvShiftRows, InvSubBytes & AddRoundKey)
  - `AESKEYGENASSIST` Effectue une partie de la génération des sous-clés
  - `AESIMC` Effectue une partie de la préparation des sous-clés pour le déchiffrement

Plan

Plan

Introduction

- Inventé par Ron RIVEST, Adi SHAMIR et Leonard ADLEMAN en 1977
- Permet de faire du chiffrement asymétrique et des signatures numériques
- Sécurité basée sur la difficulté de factorisation de grands nombres (une clé RSA de 768 bits (soit 232 chiffres décimaux) a été cassée (factorisée) en 2009 [6])

Clés

- Choisir deux nombres premiers p et q (grands)
- $n = pq$ (module public, la taille de la “clé” est la taille de n)
- Calculer $\phi(n) = (p-1)(q-1)$ (indicatrice d’Euleur en n)
- Choisir e tel que $1 < e < \phi(n)$ et tel que e et $\phi(n)$ sont premiers entre eux, c’est l’exposant public (en général, $e = 2^{16} + 1 = 65537$)
- Déterminer d tel que $ed \equiv 1 \pmod{\phi(n)}$ (en utilisant par exemple l’algorithme d’Euclide étendu)
- La clé publique est constituée de n et e et la clé privée de n et d

Chiffrement / Déchiffrement

- Chiffrement : $c \equiv m^e \pmod{n}$
- Déchiffrement : $m \equiv c^d \pmod{n}$
- En pratique, on effectue rarement que cette opération (problème de sécurité) mais on rajoute notamment du bourrage (*padding*)

Preuve

- Petit théorème de Fermat : si p est premier et a n'est pas multiple de p, on a $a^{(p-1)} \equiv 1 \pmod{p}$
- $ed \equiv 1 \pmod{(p-1)(q-1)}$ donc $ed - 1 = h(p-1)(q-1)$
- Si $m \equiv 0 \pmod{p}$, $m^{ed} \equiv 0 \equiv m \pmod{p}$
- Si $m \not\equiv 0 \pmod{p}$, $m^{ed} = m^{(ed-1)}m = m^{h(p-1)(q-1)}m = (m^{p-1})^{h(q-1)}m \equiv 1^{h(q-1)}m \equiv m \pmod{p}$
- De même, on montre que $m^{ed} \equiv m \pmod{q}$
- D'où $(m^e)^d \equiv m \pmod{pq}$

Exponentiation modulaire

Algorithme square and multiply (Left to Right)

Inputs : M , K
$R = 1$;
for $i = |K| - 1$; $i \geq 0$; $i - -$ **do**
  $R = R^2 \pmod{n}$;
  if $K_i == 1$ **then**
    $R = R \times M \pmod{n}$;
  end if
end for
Return $R = M^K \pmod{n}$;

Exponentiation modulaire

Algorithme square and multiply (Left to Right)

- Algorithme simple
- Mais vulnérable à des attaques SPA et timings
  - En fonction de la valeur de chacun des bits de la clé, une opération (mise au carré) ou deux opérations (mise au carré + multiplication) sont effectuées
  - Ces deux opérations ont un profil de consommation différent
  - Donc en observant simplement la consommation, on obtient directement chacun des bits de la clé...

Exponentiation modulaire

Algorithme Montgomery ladder exponentiation

Inputs : M , K
$R = 1$; $S = M$;
for $i = |K| - 1$; $i \geq 0$; $i - -$ **do**
  if $K_i == 1$ **then**
    $R = R \times S$; $S = S^2$;
  else
    $S = S \times R$; $R = R^2$;
  end if
end for
Return $R = M^K$;

Plan

Conclusion

- Le choix de la méthode d'implémentation de l'algorithme n'impacte pas uniquement les performances mais également les vulnérabilités de l'algorithme face aux attaques physiques
- L'algorithme ne fait pas tout, la façon dont il est utilisé (mode d'opération, protocole, génération des clés, etc.) est également critique d'un point de vue de la sécurité globale

Plan

Références

Références I

[1] Eli Biham and Adi Shamir.
Differential cryptanalysis of the full 16-round DES.
In *Advances in Cryptology — CRYPTO' 92*, volume 740 of *Lecture Notes in Computer Science*, pages 487–496. Springer Berlin Heidelberg, 1993.

[2] Johannes Götzfried and Tilo Müller.
Fast software encryption with simd.
In *Proceedings of the Third European Workshop on System Security (EUROSEC'13)*, April 2013.

[3] Shay Gueron.
Intel advanced encryption standard (aes) new instructions set.
White Paper, September 2012.
`http://download-software.intel.com/sites/default/files/article/165683/aes-wp-2012-09-22-v01.pdf`.

[4] Auguste Kerckhoffs.
La cryptographie militaire.
Journal des sciences militaires, IX :5–38, January 1883.

[5] Auguste Kerckhoffs.
La cryptographie militaire.
Journal des sciences militaires, IX :161–191, February 1883.

[6] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen Lenstra, Emmanuel Thomé, Joppe Bos, Pierrick Gaudry, Alexander Kruppa, Peter Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev, and Paul Zimmermann.
Factorization of a 768-bit RSA modulus.
Cryptology ePrint Archive, Report 2010/006, 2010.
`http://eprint.iacr.org/2010/006`.

Références II

[7] Mitsuru Matsui.
Linear cryptanalysis method for DES cipher.
In *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer Berlin Heidelberg, 1994.

[8] SciEngines.
Break des in less than a single day.
Press Release, 2009.
`http://www.sciengines.com/company/news-a-events/74-des-in-1-day.html`.

[9] Claude E. Shannon.
Communication theory of secrecy systems.
Bell System Technical Journal, 28(4) :656–715, 1949.