Cryptographie avancée

Anne Garcia-Sanchez

M2i M1 - CCI Avignon

15 octobre 2024

Chiffrements par blocs

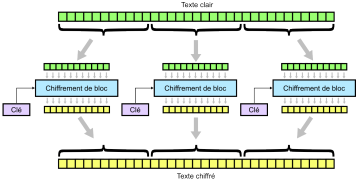
le message est découpé en blocs de longueur fixe.

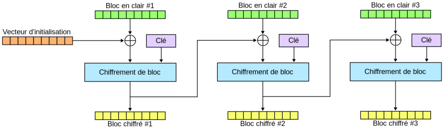
- DES
- AES

- modes opératoires: ECB, CBC, CTR, GCM

différentes façons d'enchaîner le chiffrement de plusieurs blocs

- rembourrage, padding





Modes opératoires

ECB: un bloc de message m est toujours chiffré de la même façon.
Exemple illustrant le problème: ECB penguin

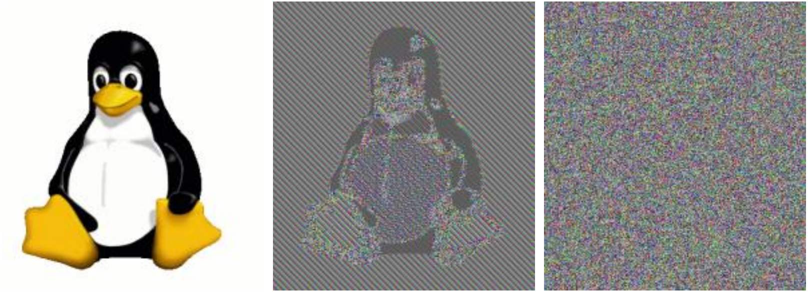
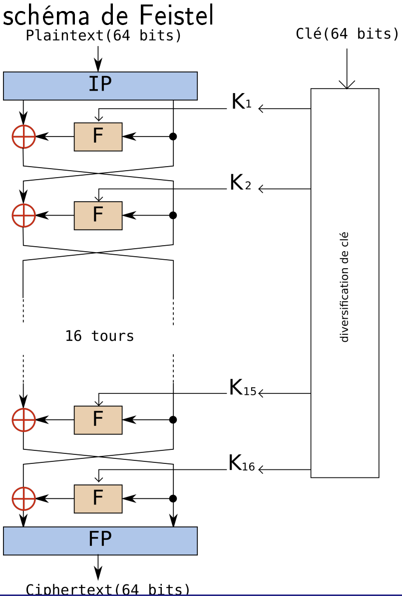


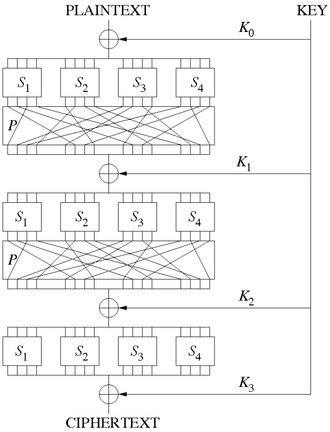
Figure: Image originale (Larry Ewing) et image chiffrée en mode ECB puis en mode CBC

Chiffrement DES



Chiffrement AES - Advanced Encryption Standard

- chiffrement de type réseau de substitutions-permutations
- blocs de 128 bits
- clés de 128, 192 ou 256 bits
- 10, 12 ou 14 tours



Confusion et diffusion

Claude Shannon: *Communication Theory of Secrecy Systems* - 1949

propriétés fondamentales pour assurer la sécurité d'un système de chiffrement

confusion: liens entre texte clair et texte chiffré trop compliqués pour être exploités par un attaquant

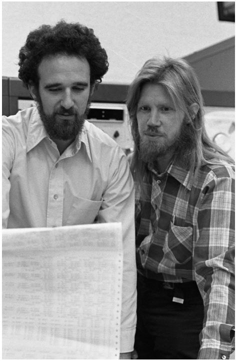
diffusion: chaque bit du texte clair et chaque bit de la clé influencent chaque bit du texte chiffré

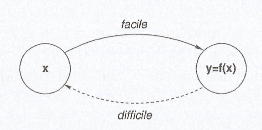
Limites du chiffrement symétrique

chiffrements sûrs, rapides

Problème: comment échanger la clé en toute sécurité?

Idée géniale 1

1976: Diffie et Hellman 

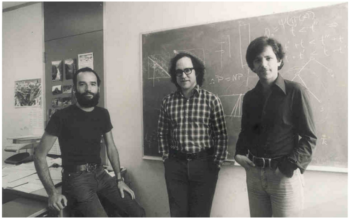


utilisation de fonctions à sens unique à trappe

clé publique permet de chiffrer mais pas de déchiffrer à moins de connaître la trappe - clé privée

Idée géniale 2

1978: Ronald Rivest, Adi Shamir et Leonard Adleman



premier exemple de fonction à sens unique (supposée) à trappe: fonction RSA

s'appuie sur problème de factorisation des entiers

notions: nombres premiers, puissance modulaire, inverse modulaire

Chiffrement RSA

clé publique: deux entiers **N** et **e**

clé privée: un entier **d**

- chiffrement de l'entier m: calcul du chiffré $c \equiv m^e \pmod{N}$ avec m et c positifs et inférieurs à N
- déchiffrement du chiffré : $m \equiv c^d \pmod{N}$

ordres de grandeur:
taille minimale de N: 2048 bits
e strictement supérieur à 2^{16} (= 65536) - exemple: 65537

Chiffrement RSA

ordres de grandeur

$2^{2048} =$
3231700607131100730071487668866995196044410266971548403213034542752465513886789089319720141152291346368871796092189801949411955915049092109508815238644828312063087736730099609175019775038965210679605763838406756827679221864261975616183809433847617047058164585203630504288757589154106580860755239912393038552191433338966834242068497478656456949485617603532632205807780565933102619270846031415025859286417711672594360371846185735759835115230164590440369761323328723122712568471082020972515710172693132346967854258065669793504599726835299863821552516638943733554360213543322960464531847860495214819355585361105959623065 6

Congruences

$a \equiv b \pmod{n}$ se lit «a est congru à b modulo n»

On peut passer de a à b en ajoutant ou retranchant un certain nombre de fois n

$$a = b + kn$$

exemples:

$53 \equiv 1 \pmod{26}$
$15 \equiv 3 \pmod{12}$

Exponentiation modulaire

Soient des entiers a et b et un entier non nul n, l'exponentiation modulaire est définie par :

$$c = a^b \pmod{n} \text{ avec } 0 \leq c < n$$

Exemple: $3^8 \pmod{7}$

$$\begin{aligned} 3^8 \pmod{7} &= 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3 \pmod{7} \\ &= 6561 \pmod{7} \\ &= 2 \pmod{7} \end{aligned}$$

Exponentiation rapide / binaire «square and multiply»

$$3^8 = 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3$$

rappels:

$$(a^m)^n = a^{m \times n} = (a^n)^m$$

$$a^{m+n} = a^m \times a^n$$

exposant: carrés successifs

$$\begin{aligned} 3^8 &= (3^4)^2 \\ &= ((3^2)^2)^2 \end{aligned}$$

Exponentiation rapide / binaire «square and multiply»

$3^{11} = 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3$

$$\begin{aligned} 3^{11} &= 3 \times 3^{10} \\ &= 3 \times (3^5)^2 \\ &= 3 \times (3 \times 3^4)^2 \\ &= 3 \times (3 \times (3^2)^2)^2 \end{aligned}$$

écriture binaire de 11: 1011

$11 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

Exponentiation rapide / binaire «square and multiply»

square - élever au carré: exposant décalé d'un bit vers la gauche

$$x^1.x^1 = x^{10_2} \qquad = x^2$$

$$x^{10}.x^{10} = x^{100_2} = x^2.x^2 \qquad = x^4$$

multiply - multiplier par la base: on ajoute 1 à l'exposant (binaire)

$$x.x^{100} = x^{101_2} \qquad = x^5$$

Exponentiation rapide / binaire «square and multiply»

$3^{11} = 3^{1011_2}$

idée: recréer l'exposant avec les opérations *square* et *multiply* avec le minimum d'étapes pour obtenir la valeur attendue

exposant	opération	calcul binaire	calcul décimal
1		3^1	
10	*square*	$3^1.3^1 = 3^{10_2}$	3^2
100	*square*	$3^{10}.3^{10} = 3^{100_2}$	3^4
101	*multiply*	$3.3^{100} = 3^{101_2}$	3^5
1010	*square*	$3^{101}.3^{101} = 3^{1010_2}$	3^{10}
1011	*multiply*	$3.3^{1010} = 3^{1011_2}$	3^{11}

Exponentiation rapide / binaire «square and multiply»

Algorithme d'exponentiation rapide:

Entrée: entiers a, b, n
avec l'écriture binaire de b: $b_m.2^m + ... + b_1.2^1 + b_0.2^0$
Sortie: $a^b \pmod{n}$

$result$ ← 1
pour i de m à 0 **faire**
$result = result^2$ modulo n
si $b_i = 1$ **alors**
result ← (result $\times$ a) modulo n
fin si
fin pour
Renvoyer $result$

attaques physiques sur l'algorithme «square and multiply»

attaques par observation

mesures physiques: temps, température, consommation de courant, rayonnement électromagnétique

image Laboratoire Haute Sécurité (LHS) d'INRIA Rennes

attaques physiques sur l'algorithme «square and multiply»

analyse simple de courant - *simple power analysis - SPA*

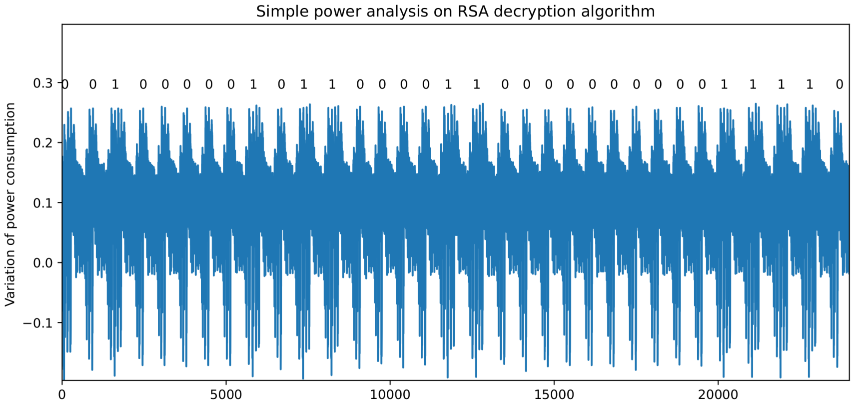


image IMT Atlantique

contre-mesures - exemple: ajout d'instructions factices