Cryptologie : Cryptographie à clé secrète

Matthieu Amiguet

2006 – 2007

haute école arc ingénierie
neuchâtel berne jura saint-imier le locle porrentruy

Flux et blocs 2

- Cryptage par flux (ou *à la volée*) (*stream cipher*) : encrypte un caractère à la fois.
  - utilisation réduite de la mémoire
  - fonctionnement en “filtre”
  - peu de propagation d’erreur
  - pas ou peu d’algorithmes standard
- Cryptage par blocs (*block cipher*) : encrypte les caractères par groupes de taille fixe.
  - très bons algorithmes standard
  - meilleure “dispersion” des caractéristiques du message
  - problème du “bourrage”.

Cryptage de Vernam et masque jetable 3

- Vernam : $c_i = m_i \oplus k_i$
  - $\oplus$ désigne le XOR bit-à-bit ou l’addition modulo n

Masque jetable

- clé *aussi longue* que le message
- les éléments de la clé sont *aléatoires*
- La clé n’est utilisée qu’*une seule fois*

Sécurité du masque jetable 4

- Ce cryptage est incassable
  - Démontré par Claude Shannon en 1949
  - Limite *théorique* et non seulement pratique !
  - Dans les faits, en essayant de casser le cryptage, on pourra récupérer tous les message en clair de même longueur avec la même probabilité !
- Cette sécurité n’est même pas soumise à l’hypothèse $P \neq NP$!
- Ne protège par contre pas contre une attaque de type “Man in the middle”

Le masque jetable en pratique 5

- Malgré sa perfection théorique, le masque jetable pose de sérieux problèmes d'application pratique...
  - Générer une clé aléatoire
  - La transmettre (étant donné qu'elle ne pourra de toutes façons servir qu'une seule fois)
- Malgré ces problèmes, ce cryptage a été (est ?) utilisé par les services secrets de différents pays
  - USA et Grande-Bretagne durant la 2^e guerre mondiale
  - KGB, en tout cas jusque dans les années 60
  - "Hot line" entre la Maison Blanche et le Kremlin pendant la guerre froide
  - ...

Avenir du masque jetable 6

- De nombreux cryptages actuels se basent sur la complexité algorithmique de certaines opérations...
- ... ou sur un postulat d'algorithmique : $P \neq NP$
- Le masque jetable résisterait aussi bien à une augmentation significative de la puissance de calcul qu'à la démonstration que $P = NP$
- En particulier, le masque jetable serait probablement le seul cryptage connu à résister à l'avènement de l'ordinateur quantique...
- ... d'autant plus que la cryptographie quantique permet de transporter les clés de manière sûre !

Sinon quoi ? 7

- Pour de nombreuses utilisations, les contraintes posées par le masque jetable sont bien trop fortes
- On va donc essayer de s'en approcher en générant une clé *pseudo-aléatoire*
  - Il suffit donc d'une clé plus courte pour générer le "flux de clés" de ce masque affaibli...
  - Attention, ce flux de clés ne doit toujours être utilisé qu'une fois !
- Le flux de clé peut être indépendant du message à crypter (cryptage *synchrone*) ou être influencé par lui (cryptage *asynchrone*)

Qu'est-ce que c'est ? 8

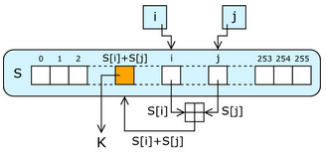
- RC4 est un algorithme de cryptage par flux développé en 1987 par Ron Rivest (RC4=Rivest Cipher 4)
- Le nom RC4 est une marque déposée des laboratoires RSA
- L'algorithme a été tenu secret...
- ... mais un algorithme compatible a été publié en 1994 sur la liste *Cypherpunks*
  - Parfois appelé ARCFOUR ou ARC4 ("Alledged RC4")
  - Obtenu par désassemblage ?
- RC4 est utilisé dans SSL, WEP, WPA, MS Office, ...
- Considéré comme moyennement sûr
  - À éviter dans les nouvelles applications !

RC4 – vue d'ensemble 9

- RC4 est caractérisé par un état interne S et deux pointeurs i et j
  - S est un tableau de 256 octets qui représente une permutation de $\{0, 1, \ldots, 255\}$
- S est d'abord initialisé en fonction de la clé
- Ensuite, S est utilisé pour générer un flux pseudo-aléatoire qui sera combiné par un XOR au flux du message
- Le flux généré est indépendant du message à crypter
  - cryptage synchrone

RC4 – génération du flux 10

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap(S[i],S[j])
    output S[(S[i] + S[j]) mod 256]
```



RC4 – initialisation 11

- L'initialisation est similaire, mais en "mélangeant" la clé avec S

```
for i from 0 to 255:
    S[i] := i

j := 0

for i from 0 to 255:
    j := (j + S[i] + key[i mod l]) mod 256
    swap(S[i],S[j])
```

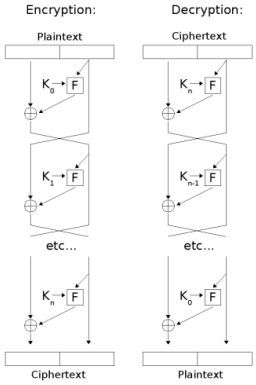

RC4 – conclusion 12

- Avantages
  - Algorithme très simple à comprendre et à implémenter
  - Très rapide en implémentation logicielle
- Désavantage majeur
  - Très facile à *mal* utiliser !
  - De nombreuses utilisations sont au fait peu sûres. . .

Data Encryption Standard (DES) 13

- Conçu par IBM en 1977
  - pour satisfaire la demande des banques
- Adopté par le ministère de la défense des États-Unis
- Implémentation matérielle possible.

Cryptages de Feistel – 1 14



Feistel Cipher

Cryptages de Feistel – 2 15

Cryptage itéré

Cryptage impliquant l'usage répété d'une fonction interne

Cryptage de Feistel

Cryptage itéré

- sur des blocs de la forme (L_i, R_i)
- $L_i = R_{i-1}$
- $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
- Les K_i sont déduits d'une clé K.

DES 16

- Cryptage de Feistel
- Blocs de 64 bits
- Clé de 56 bits (+8 bits de parité)
- 16 itérations
- Permutation initiale, inversée à la fin
- Détails : voir feuilles annexes !

DES – clés faibles 17

- Clé faible : $E_k(E_k(x)) = x$
  - DES en a 4
  - Pour une clé faible, E_k admet 2^{32} points fixes
- Clés semi-faibles : $E_{k_1}(E_{k_2}(x)) = x$
  - DES en a six paires
  - Pour une clé semi-faible, E_k admet 2^{32} points anti-fixes.

DES – propriétés 18

- Chaque bit de c dépend de chaque bit de m et de chaque bit de k
- Toute modification de 1 bit de m ou de k change chaque bit de c avec une proabilité de $\frac{1}{2}$
- La modification de c résulte en modifications imprévisibles de m
- La composition de deux DES n'est généralement pas un DES.

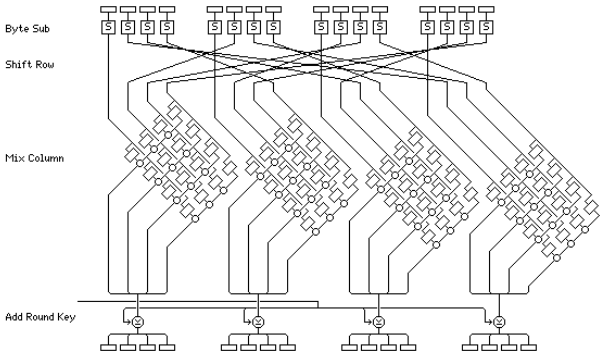
Aller plus loin 19

- Une clé de 56 bits, c'est bien peu...
- Solutions :
  - Triple-DES
  - AES
  - ...

Advanced Encryption Standard (AES) 20

- Algorithme de Rijndael
- Sélectionné en 2000 par le gouvernement américain comme nouveau standard
- Blocs de 128 bits
- Clés de 128, 192 ou 256 bits
- 9, 11 ou 13 itérations (suivant la taille de la clé).

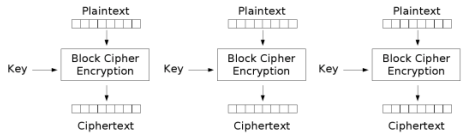
Une itération AES



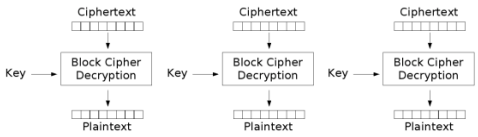
Enchaîner les blocs...

- DES (ou AES) nous dit comment crypter *un* bloc du message...
- ... mais que fait-on si le message fait plus d'un bloc ?
- La première idée qui vient à l'esprit est simplement de recommencer avec le bloc suivant
- Cette manière de faire est connue sous le nom de *Electronic Codebook*
  - Simple, mais assez faible
- La manière d'enchaîner le cryptage des blocs d'un cryptage par blocs s'appelle *mode d'opération*
- Le choix du mode d'opération est (en grande partie) indépendant du cryptage lui-même.

Electronic Codebook (ECB) – 1



Electronic Codebook (ECB) mode encryption

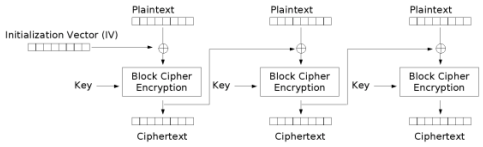


Electronic Codebook (ECB) mode decryption

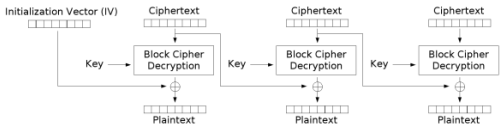
Electronic Codebook (ECB) – 2

- $c_j \leftarrow E_k(m_j)$
- $m_j \leftarrow E_k^{-1}(c_j)$
- Une erreur ne se propage que dans son bloc
- Blocs en clair identiques $\rightarrow$ blocs cryptés identiques
- Mauvaise résistance aux attaques actives
- Cryptage et decryptage peuvent être parallélisés.

Cipher Block Chaining (CBC) – 1 25



Cipher Block Chaining (CBC) mode encryption

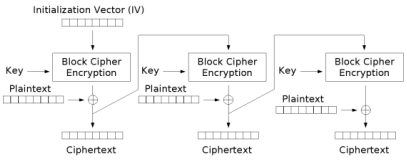


Cipher Block Chaining (CBC) mode decryption

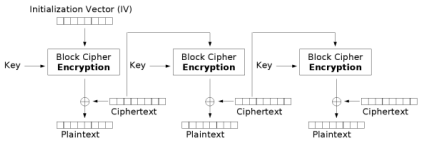
Cipher Block Chaining (CBC) – 2 26

- $c_j = E_k(c_{j-1} \oplus m_j)$, $c_0 = IV$
- $m_j = c_{j-1} \oplus E_k^{-1}(c_j)$
- Une erreur se propage dans son bloc et dans le suivant
- Blocs en clair identiques $\rightarrow$ blocs cryptés différents
- Résistance moyenne aux attaques actives
- Le cryptage ne peut pas être parallélisé, le décryptage peut l'être.

Cipher Feedback (CFB) – 1 27



Cipher Feedback (CFB) mode encryption

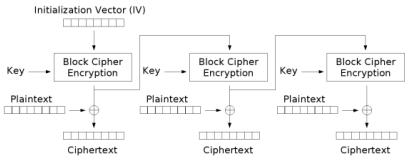


Cipher Feedback (CFB) mode decryption

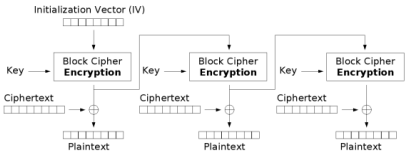
Cipher Feedback (CFB) – 2 28

- $c_j \leftarrow m_j \oplus E_k(c_{j-1})$, $c_0 \leftarrow IV$
- $m_j = c_j \oplus E_k(c_{j-1})$, $c_0 \leftarrow IV$.
- Cryptage *et* décryptage basés sur E_k
- Une erreur se propage dans son bloc et dans le suivant
- Blocs en clair identiques $\rightarrow$ blocs cryptés différents
- Résistance moyenne aux attaques actives
- Le cryptage ne peut pas être parallélisé, le décryptage peut l'être.

Output Feedback (OFB) – 1



Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

Output Feedback (OFB) – 2

- Cryptage
  - $I_0 \leftarrow IV$
  - $I_j \leftarrow E_k(I_{j-1})$
  - $c_j \leftarrow m_j \oplus I_j$
- Décryptage
  - Identique

OFB – propriétés

- Cryptage *et* décryptage basés sur E_k
- Aucune propagation d'erreur
- En cas de perte de données lors de la transmission, toute la fin du message est perdue
- Blocs en clair identiques $\rightarrow$ blocs cryptés différents
- Résistance moyenne aux attaques actives
- Le cryptage ne peut pas être parallélisé, le décryptage non plus
- Possibilité de calculer les t_i à l'avance.

Le problème du bourrage

- Si le message n'a pas comme longueur un multiple entier de la taille des blocs, que faire ?
- Première solution : introduire du bourrage
  - Augmentation de la taille du message
  - Problème de la distinction message/bourrage
- Deuxième solution : le dernier bloc est compté depuis la fin du message
  - Augmentation de la taille du message
  - Un peu plus complexe à mettre en place
- Troisième solution : Encrypter la fin par flux
  - Introduit un nouvel algorithme, donc potentiellement de nouvelles vulnérabilités...

Pour en savoir plus. . .

- Wikipedia : `http://wikipedia.org/`
- Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, *Handbook of Applied Cryptography*, 1996, téléchargeable sur `http://www.cacr.math.uwaterloo.ca/hac/`