Les modes opératoires de la cryptographie symétrique

Jang Schiltz

Base de connaissances Ξ, © 2003

Table des matières

1 Introduction

Les algorithmes de chiffrement par blocs ne traitent pas les textes bit par bit, mais manipulent des blocs de texte - par exemple de 64 bits pour le DES ou de 128 bits pour le AES. Il n'est pas suffisant de mettre n'importe comment un algorithme de chiffrement par blocs dans un programme. On peut au contraire utiliser ces algorithmes de diverses manières suivant ses besoins spécifiques et ces divers modes de chiffrement ont également des propriétés de sécurité spécifiques.

Les principaux modes opératoires du DES sont définis dans les standards FIPS 81 du 2 décembre 1980 et aussi dans ANSI X3.106-1983. Ils peuvent être utilisés également pour les autres procédures de chiffrement à clé secrète, quitte à adapter éventuellement la taille des blocs utilisés.

Tous les modes opératoires impliquent bien sûr qu'il faut, si nécessaire, commencer par rallonger le texte en clair de façon à ce que sa longueur devienne divisible par la longueur des blocs de chiffrement. Il n'y a pas de règle générale pour effectuer cette opération, la bonne démarche à suivre dépend des applications. Le standard FIPS 81 préconise de remplir les blocs tout simplement par des bits choisis de façon aléatoire ou pseudo-aléatoire si on utilise le mode ECB. Dans le cas où l'intégrité des messages ne pose pas problème, le choix du rembourrage, en anglais "padding", ne semble pas avoir beaucoup d'importance. Pour des systèmes qui contrôlent l'intégrité des messages (comme c'est souvent le cas pour des systèmes en ligne), il faut par contre faire attention afin d'éviter des attaques comme on va le voir plus loin.

Dans tous les cas, il faut toujours marquer les fichiers avec padding pour que le destinataire sache que les derniers bits du dernier bloc ne font pas partie du texte en clair.

Les notations utilisées dans la description des modes opératoires sont les suivantes :

$M[n]$ Le n-ième bloc du texte en clair. La taille du bloc dépend du mode opératoire et du procédé de chiffrement.

$C[n]$ Le n-ième bloc du texte chiffré. Là aussi, la taille du bloc dépend du mode opératoire et du procédé de chiffrement.

$e(M)$ La fonction de chiffrement (Dans le cas du DES, elle agit sur des blocs de 64 bits, en utilisant les sous-clés générées à partir d'une clé de 56 bits). Dans le cas où

plusieurs clés peuvent intervenir, on note $e_K(M)$ pour montrer qu'on utilise la clé K pour chiffrer.

$d(M)$ La fonction de déchiffrement (Dans le cas du DES, elle agit sur des blocs de 64 bits, en utilisant les mêmes sous-clés que la fonction e, sauf qu'elles sont appliquées en ordre inverse). Dans le cas où plusieurs clés peuvent intervenir, on note $d_K(M)$ pour montrer qu'on utilise la clé K pour déchiffrer.

VI Un vecteur d'initialisation de la même taille que les blocs de texte.

$I[n]$ La n-ième valeur d'un registre de décalage utilisé dans certains modes. Il a la même taille que les blocs de texte.

$MSB_k(m)$ (de l'anglais "most significant bits") Les k bits les plus significatifs de m.

$\oplus$ L'opérateur XOR (ou exclusif), c'est-à-dire l'addition bit par bit modulo 2.

$\ll$ L'opérateur de décalage vers la gauche ($X \ll k$ veut dire que les bits de la variable X sont décalés de k positions vers la gauche et que les anciens k premiers bits sont perdus).

$\gg$ L'opérateur de décalage vers la droite ($X \gg k$ veut dire que les bits de la variable X sont décalés de k positions vers la droite et que les anciencs k derniers bits sont perdus).

$|$ L'opérateur de concaténation, c'est-à-dire le regroupement de 2 messages en un seul.

2 Le carnet de codage électronique (ECB)

2.1 Description du mode opératoire

Le carnet de codage électronique, en anglais "Electronic Code Book" (ECB), est le mode opératoire le plus simple. Il consiste à chiffrer chaque bloc de texte en clair en un bloc de texte chiffré.

Son nom provient du fait qu'un texte en clair donné est transformé à chaque fois dans le même texte chiffré à condition d'utiliser toujours la même clé pour l'encoder. On pourrait donc théoriquement construire un carnet comportant tous les textes en clair et les textes chiffrés correspondants pour une clé donné. Mais cela fait par exemple 2^{64} paires de blocs par clé dans le cas du DES et il existe 2^{56} clés différentes. Ce n'est donc pas faisable pour les algorithmes de chiffrement dont les clés ont une longueur suffisante.

Pour chiffrer un texte en mode ECB, on effectue donc les opérations suivantes :

$$C[n] = e(M[n]), \text{ pour } n \geq 1.$$

Pour déchiffrer un texte en mode ECB, on effectue les opérations suivantes :

$$M[n] = d(C[n]), \text{ pour } n \geq 1.$$

Exemple 2.1 *Supposons que comme algorithme de chiffrement on utilise la permutation simple σ définie sur des blocs de 4 bits par*

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix},$$

ce qui veut dire que le premier bit de chaque bloc de texte en clair va être remplacé par son deuxième, le deuxième par le troisième, le troisième par le quatrième et le quatrième par le premier.

Considérons le texte en clair $M = 101100010100101$. Puisque sa longueur de bits n'est pas égale à un multiple de 4, on le rallonge en ajoutant un 0 à la fin et on obtient ainsi les 4 blocs suivants :

$$M[1] = 1011, \;\; M[2] = 0001, \;\; M[3] = 0100, \;\; M[4] = 1010.$$

En chiffrant ces 4 blocs avec l'algorithme décrit ci-dessus, on obtient les 4 blocs de texte chiffrés

$$e([1]) = 0111, \;\; e(M[2]) = 0010, \;\; e(M[3]) = 1000, \;\; e(M[4]) = 0101,$$

donc le texte chiffré $C = 0111001010000101$.

Le carnet de codage électronique est le mode opératoire le plus simple à utiliser. L'avantage est que chaque bloc de texte en clair est chiffré indépendamment des autres ; on peut donc chiffrer les blocs dans n'importe quel ordre. Cela est important pour des fichiers chiffrés auxquels on accède aléatoirement, comme une base de données. En effet, si les enregistrements d'une base de données ont comme longueur un nombre entier de blocs de texte, le chiffrement avec le mode ECB permet de déchiffrer n'importe quel enregistrement indépendamment des autres, respectivement d'ajouter ou de retirer des enregistrements quelque part au milieu de la base de données.

2.2 Considérations de sécurité

Il y a cependant plusieurs problèmes de sécurité avec ce mode de chiffrement.

Le premier problème est que si on utilise deux fois le même texte en clair et la même clé de chiffrement, le résultat du chiffrement sera identique. Des régularités dans le texte en clair se reproduisent dans le texte chiffré. Le texte chiffré permet donc d'obtenir des informations sur le texte en clair, même si on n'arrive pas à le déchiffrer.

Le deuxième problème est que si le cryptanalyste a le texte en clair et le texte chiffré de plusieurs messages, il peut commencer à construire un carnet de codage sans connaître la clé. Dans la plupart des situations du monde réel, des fragments de messages ont tendances à se répéter, surtout au début et à la fin des messages, où apparaissent souvent des en-têtes et des conclusions bien définies qui contiennent des informations contenant l'expéditeur, le destinataire, la date, etc. De même, des messages engendrés par ordinateur ont souvent une structure régulière.

Le troisième problème est qu'un attaquant peut changer le texte chiffré en ajoutant simplement des blocs chiffrés par la même procédure de chiffrement et la même clé ou en échangeant tout simplement quelques-uns des blocs chiffrés. Si des formats de messages standards sont utilisés (par exemple pour des opérations de transfert d'argent entre deux banques), l'adversaire peut même arriver à modifier les messages chiffrés sans connaître la clef, ni même les détails de l'algorithme de chiffrement utilisé.

Pour toutes ces raisons, le mode ECB n'est pas à recommander.

2.3 Propagation d'erreurs

Si pendant la transmission des données, il y a des erreurs qui apparaissent dans un bloc, le texte en clair obtenu par déchiffrement de ce bloc sera incorrect. En effet, si on utilise un algorithme de chiffrement sûr, chaque bit d'un texte chiffré provient d'une transformation complexe de tous les bits du bloc de texte en clair correspondant et de tous les bits de la clé. Ainsi une erreur d'un bit dans le texte chiffré ou dans la clé utilisée implique qu'en moyenne 50 pour cent du texte en clair récupéré sont faux.

L'erreur ne se propage cependant pas. Puisque le chiffrement respectivement le déchiffrement d'un bloc se fait indépendamment des autres blocs, une erreur dans un bloc n'a pas d'incidence sur les autres blocs.

Le même raisonnement s'applique si un bloc complet de données est perdu. Cela non plus n'affecte pas les autres blocs.

Si la démarcation entre deux blocs qui se suivent est perdue, les données vont être déchiffrées incorrectement. Mais, dès que l'émetteur et le récepteur sont de nouveau bien synchronisés, le déchiffrement redevient correct.

3 Le mode de chiffrement avec chaînage de blocs (CBC)

3.1 Description du mode opératoire

Le mode de chiffrement avec chaînage de blocs, en anglais "Cipher Block Chaining" (CBC), est un des modes les plus populaires.

Il offre une solution à la plupart des problèmes du mode ECB. Le chaînage utilise une méthode de rétroaction comme le résultat du chiffrement du bloc précédent est réutilisé pour le chiffrement du bloc courant. Plus précisément, l'opérateur binaire XOR est appliqué entre le bloc actuel de texte en clair et le bloc précédent de texte chiffré et on applique ensuite l'algorithme de chiffrement au résultat de cette opération.

Pour le tout premier bloc, un bloc ayant un contenu aléatoire, appelé vecteur d'initialisation (initialization vector), est généré et utilisé pour l'application de l'opération XOR.

Ainsi, chaque bloc chiffré ne dépend non seulement du bloc de texte en clair correspondant, mais également de tous les blocs chiffrés qui le précèdent.

Pour chiffrer un texte en mode CBC, on effectue par conséquent les opérations suivantes :

$$C[1] = e(M[1] \oplus VI)$$
$$C[n] = e(M[n] \oplus C[n-1]), \quad \text{pour } n > 1.$$

Pour déchiffrer un bloc de texte chiffré en mode CBC, on applique l'algorithme de déchiffrement au bloc chiffré et puis on le combine par ou exclusif avec le bloc chiffré précédent, respectivement avec le vecteur d'initialisation, dans le cas du premier bloc.

On effectue donc les opérations suivantes :

$$M[1] = d(C[1]) \oplus VI$$
$$M[n] = d(C[n]) \oplus C[n-1], \quad \text{pour } n > 1.$$

Exemple 3.1 *Supposons de nouveau qu'on utilise la permutation simple*

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$$

comme algorithme de chiffrement et qu'on veuille chiffrer le même texte en clair $M = 101100010100101$ *qui produit les* 4 *blocs*

$$M[1] = 1011, \;\; M[2] = 0001, \;\; M[3] = 0100, \;\; M[4] = 1010.$$

Choisissons le vecteur d'initialisation

$$VI = 1010$$

et chiffrons en utilisant le mode CBC. Nous obtenons

$$C[1] = e(M[1] \oplus VI) = e(0001) = 0010$$
$$C[2] = e(M[2] \oplus C[1]) = e(0011) = 0110$$
$$C[3] = e(M[3] \oplus C[2]) = e(0010) = 0100$$
$$C[4] = e(M[4] \oplus C[3]) = e(1110) = 1101,$$

donc le texte chiffré $C = 0010011001001101$.

Pour déchiffrer, nous effectuons les opérations suivantes :

$$M[1] = d(C[1]) \oplus VI = 1010 \oplus 1010 = 1011$$
$$M[2] = d(C[2]) \oplus C[1] = 0010 \oplus 0011 = 0001$$
$$M[3] = d(C[3]) \oplus C[2] = 0110 \oplus 0010 = 0100$$
$$M[4] = d(C[4]) \oplus C[3] = 0100 \oplus 1110 = 1010,$$

ce qui nous permet de retrouver le texte en clair original.

3.2 Considérations de sécurité

Le rôle du vecteur d'initialisation est d'empêcher que si deux textes en clair débutent de la même façon, les textes chiffrés correspondants le font également.

Il n'est pas obligatoire de le choisir aléatoirement ; il peut aussi être un numéro d'ordre qui est augmenté après chaque message. La seule chose importante est qu'il doit être différent pour chaque message chiffré avec la même clé.

Grâce au vecteur d'initialisation, même un bloc de texte en clair identique donnera un message chiffré différent. Il n'est plus possible pour un espion d'apprendre la moindre information concernant le texte en clair à partir du texte chiffré.

Il n'est pas nécessaire non plus de tenir secret le vecteur initialisation. Généralement on le transmet en clair avec le texte chiffré.

3.3 Propagation d'erreurs

Une erreur d'un simple bit dans le texte chiffré, par exemple dû à un bruitage de la liaison ou d'un disfonctionnement du dispositif de stockage, affecte un bloc et un bit du texte en clair reconstruit. En effet, le bloc contenant l'erreur est complètement brouillé et de plus il apparaît une erreur d'un bit dans le bloc suivant au même endroit que le bit erroné. Les autres blocs ne sont pas affectés par l'erreur ; on dit que le mode CBC est auto-récupérant.

Le mode de chiffrement avec chaînage de blocs ne récupère par contre pas du tout les erreurs de synchronisation. Si un bit est ajouté ou perdu dans le flot de texte chiffré, alors tous les blocs qui suivent sont décalés d'un bit et deviennent indéchiffrables. A cause de ce phénomène d'amplification d'erreur, tout cryptosystème qui utilise le mode CBC doit s'assurer que la structure des blocs reste intacte, soit par encadrement soit en stockant les données dans des paquets de plusieurs blocs. Généralement, on ajoute un système de détection d'erreurs et des méthodes de récupération à la ligne de communication qu'on applique après le chiffrement et avant le déchiffrement.

Fin 2001, Serge Vaudenay a trouvé une attaque contre l'implémentation de certains systèmes qui utilisent le mode CBC pour chiffrer des messages préformatés et qui testent après le déchiffrement si le texte en clair récupéré est bien du bon type. En exploitant le protocole de communication de tels systèmes, on peut, sous certaines conditions assez restrictives, déchiffrer les messages chiffrés, sans connaître la clé de chiffrement. Martin Vuagnoux a publié en février 2003 un programme qui permettait d'attaquer l'OpenSSL en mode CBC.

On peut cependant éviter cette attaque, ou bien en appliquant une fonction de hachage au texte rembourré avant de le chiffrer ou bien en remplaçant le mode CBC par un mode qui garantit l'authenticité et la confidentialité des messages, comme le mode OCB (les nouvelles versions 0.9.6i et 0.9.7 de OpenSSL ne sont d'ailleurs plus vulnérables par cette attaque).

4 Le mode de chiffrement à rétroaction (CFB)

4.1 Description du mode opératoire

Avec les modes EBC ou CBC, le chiffrement ne peut pas commencer avant qu'un bloc complet de données n'ait été reçu, ce qui peut poser un problème dans certaines applications réseaux, où les données à traiter se présentent sous forme de paquets de la taille de l'octet.

Le mode de chiffrement à rétroaction, en anglais "Cipher Feedback"(CFB), permet de chiffrer les données par unités plus petites que la taille de bloc. On appelle mode de chiffrement à rétroaction à k bits le mode CFB qui chiffre les données par unités de k bits. Souvent on chiffre un caractère ASCII à la fois, c'est-à-dire $k = 8$, mais k peut en principe prendre n'importe quelle valeur, pourvu qu'elle ne dépasse pas la taille d'un bloc.

En mode CFB, on manipule un registre de décalage de la taille d'un bloc de texte en clair. Pour commencer, le registre de décalage est rempli par un vecteur d'initialisation. On chiffre ensuite le registre de décalage avec l'algorithme de chiffrement utilisé et on combine les k bits les plus significatifs, c'est-à-dire les k bits les plus à gauche du résultat par un ou exclusif avec les k premiers bits du texte en clair pour obtenir ainsi les k premiers bits du texte chiffré qu'on peut alors transmettre. On place ce bloc de texte chiffré dans les k bits les plus à droite du registre de décalage et on décale tous les autres bits du registre de décalage de k positions vers la gauche en coupant ses k premiers bits. On continue alors de chiffrer le reste du texte en clair de la même manière.

Pour chiffrer un texte en mode CFB à k bits, on effectue donc les opérations suivantes (ici $M[\cdot]$ et $C[\cdot]$ sont des blocs de k bits) :

$$\begin{array}{ll} I[1] = VI & \\ I[n] = (I[n-1] \ll k) | C[n-1], & \text{pour } n > 1 \\ C[n] = M[n] \oplus MSB_k\big(e(I[n])\big), & \text{pour } n \geq 1. \end{array}$$

En mode CFB, le déchiffrement se fait en appliquant de nouveau la fonction de chiffrement de l'algorithme au même registre de décalage que pour le chiffrement et en ajoutant les k premiers bits de ce résultat au bloc précédent du texte chiffré. On effectue donc les opérations suivantes :

$$\begin{array}{ll} I[1] = VI & \\ I[n] = (I[n-1] \ll k) | C[n-1], & \text{pour } n > 1 \\ M[n] = C[n] \oplus MSB_k\big(e(I[n])\big), & \text{pour } n \geq 1. \end{array}$$

Exemple 4.1 *Supposons de nouveau qu'on utilise la permutation simple*

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$$

comme algorithme de chiffrement. Utilisons un mode CFB à 3 *bits. On obtient alors les* 5 *blocs de texte en clair suivants :*

$$M[1] = 101, \;\; M[2] = 100, \;\; M[3] = 010, \;\; M[4] = 100, \;\; M[5] = 101.$$

Choisissons le vecteur d'initialisation

$$VI = 1010.$$

Nous obtenons

$$\begin{array}{lll} I[1] = 1010 & e(I[1]) = 0101 & C[1] = 101 \oplus 010 = 111 \\ I[2] = 0111 & e(I[2]) = 1110 & C[2] = 100 \oplus 111 = 011 \\ I[3] = 1011 & e(I[3]) = 0111 & C[3] = 010 \oplus 011 = 001 \\ I[4] = 1001 & e(I[4]) = 0011 & C[4] = 100 \oplus 001 = 101 \\ I[5] = 1101 & e(I[5]) = 1011 & C[5] = 101 \oplus 101 = 000, \end{array}$$

donc le texte chiffré $C = 111011001101000$.

Pour déchiffrer, nous effectuons les opérations suivantes :

$$\begin{array}{lll} I[1] = 1010 & e(I[1]) = 0101 & M[1] = 111 \oplus 010 = 101 \\ I[2] = 0111 & e(I[2]) = 1110 & M[2] = 011 \oplus 111 = 100 \\ I[3] = 1011 & e(I[3]) = 0111 & M[3] = 001 \oplus 011 = 010 \\ I[4] = 1001 & e(I[4]) = 0011 & M[4] = 101 \oplus 001 = 100 \\ I[5] = 1101 & e(I[5]) = 1011 & M[5] = 000 \oplus 101 = 101 \end{array}$$

et nous retrouvons ainsi le texte en clair $M = 101100010100101$.

4.2 Considérations de sécurité

Le mode CFB offre une grande sécurité.

Le seul problème est que le texte en clair est seulement soumis à un XOR. Si le texte en clair est connu, un texte en clair complètement différent peut être substitué en inversant les bits du texte chiffré au même endroit où on inverse les bits du texte en clair (bit-flipping attack).

4.3 Propagation d'erreurs

Une erreur dans le texte chiffré d'un seul bit provoque une erreur d'un seul bit dans le bloc de texte en clair correspondant. Puis, elle entre dans le registre à décalage, où elle embrouille tout jusqu'elle en ressorte. Après cela, le système se récupère et les blocs suivants de texte en clair sont déchiffrés correctement. Si on utilise le DES en mode CFB à 8 bits, par exemple, 9 octets de texte sont embrouillés par une erreur d'un seul bit.

Le mode CFB est également autoréparateur pour ce qui est des erreurs de synchronisation. L'erreur entre dans le registre à décalage, embrouille un bloc de données, puis ressort à l'autre bout après quoi tout fonctionne à nouveau correctement.

5 Le mode de rétroaction de sortie (OFB)

5.1 Description du mode opératoire

Le mode de rétroaction de sortie, en anglais "Output Feedback" (OFB), ressemble beaucoup au mode CFB, sauf que ce sont les k premiers bits du résultat du chiffrement

du registre de décalage qui sont ajoutés à droite du registre de décalage et non les k premiers bits du texte chiffré.

Pour commencer, le registre de décalage est rempli par un vecteur d'initialisation. On chiffre ensuite le registre de décalage avec l'algorithme de chiffrement utilisé et on combine les k bits les plus significatifs, c'est-à-dire les k bits les plus à gauche du résultat par un ou exclusif avec les k premiers bits du texte en clair pour obtenir ainsi les k premiers bits du texte chiffré qu'on peut alors transmettre.

On place ensuite les k premiers bits du résultat du chiffrement du registre de décalage dans les k bits les plus à droite du registre de décalage et on décale tous les autres bits du registre de décalage de k positions vers la gauche en coupant ses k premiers bits. On continue alors de chiffrer le reste du texte en clair de la même manière.

Pour chiffrer un texte en mode OFB à k bits, on effectue donc les opérations suivantes (ici $M[\cdot]$ et $C[\cdot]$ sont des blocs de k bits) :

$$\begin{array}{ll} I[1] = VI & \\ I[n] = (I[n-1] \ll k) | MSB_k\big(e(I[n-1]\big), & \text{pour } n > 1 \\ C[n] = M[n] \oplus MSB_k\big(e(I[n])\big), & \text{pour } n \geq 1. \end{array}$$

Cette méthode est parfois appelée rétroaction interne, parce que le mécanisme de rétroaction est indépendant à la fois des flots de texte en clair et de texte chiffré.

En mode OFB, le déchiffrement se fait en appliquant de nouveau la fonction de chiffrement de l'algorithme au même registre de décalage que pour le chiffrement et en ajoutant les k premiers bits de ce résultat au bloc précédent du texte chiffré. On effectue donc les opérations suivantes :

$$\begin{array}{ll} I[1] = VI & \\ I[n] = (I[n-1] \ll k) | MSB_k\big(e(I[n-1]\big), & \text{pour } n > 1 \\ M[n] = C[n] \oplus MSB_k\big(e(I[n])\big), & \text{pour } n \geq 1. \end{array}$$

Par construction, le chiffrement des blocs, c'est-à-dire le travail lourd, peut donc se faire avant qu'on connaisse le texte en clair. Ceci permet de chiffrer respectivement déchiffrer plus rapidement en utilisant le mode OFB.

Exemple 5.1 *Supposons de nouveau qu'on utilise la permutation simple*

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$$

comme algorithme de chiffrement. Utilisons un mode OFB à 3 bits. On obtient alors les 5 blocs de texte en clair suivants :

$$M[1] = 101, \;\; M[2] = 100, \;\; M[3] = 010, \;\; M[4] = 100, \;\; M[5] = 101.$$

Choisissons le vecteur d'initialisation

$$VI = 1010.$$

Nous obtenons

$$\begin{array}{lll} I[1] = 1010 & e(I[1]) = 0101 & C[1] = 101 \oplus 010 = 111 \\ I[2] = 0010 & e(I[2]) = 0101 & C[2] = 100 \oplus 010 = 110 \\ I[3] = 0010 & e(I[3]) = 0101 & C[3] = 010 \oplus 010 = 000 \\ I[4] = 0010 & e(I[4]) = 0101 & C[4] = 100 \oplus 010 = 110 \\ I[5] = 0010 & e(I[5]) = 0101 & C[5] = 101 \oplus 010 = 111, \end{array}$$

donc le texte chiffré $C = 111110000110111$.

Pour déchiffrer, nous effectuons les opérations suivantes :

$$\begin{array}{lll} I[1] = 1010 & e(I[1]) = 0101 & M[1] = 111 \oplus 010 = 101 \\ I[2] = 0010 & e(I[2]) = 0101 & M[2] = 110 \oplus 010 = 100 \\ I[3] = 0010 & e(I[3]) = 0101 & M[3] = 000 \oplus 010 = 010 \\ I[4] = 0010 & e(I[4]) = 0101 & M[4] = 110 \oplus 010 = 100 \\ I[5] = 0010 & e(I[5]) = 0101 & M[5] = 111 \oplus 010 = 101 \end{array}$$

et on retrouve ainsi le texte en clair $M = 101100010100101$.

5.2 Considérations de sécurité

L'exemple précédent montre qu'il peut arriver qu'à partir d'un certain moment le registre de décalage devient constant, ce qui engendre de graves problèmes de sécurité.

Pour éviter ce problème, il faut toujours que la taille de rétroaction soit égale à la taille des blocs de texte que manipule l'algorithme utilisé. Pour le DES, par exemple, il faut donc seulement utiliser le mode OFB à 64 bits.

Par ailleurs, comme dans le mode CFB, le texte en clair est seulement soumis à un XOR. Le mode OFB est donc également vulnérable à une "bit-flipping attack".

5.3 Propagation d'erreurs

L'avantage du mode OFB est qu'il n'y a pas d'amplification d'erreur. Une erreur d'un seul bit dans le texte chiffré occasionne une erreur d'un seul bit dans le texte en clair récupéré.

La perte de synchronisation ne peut par contre pas être récupérée. Si les contenus des registres à décalage de chiffrement respectivement de déchiffrement ne sont pas identiques, alors le texte en clair récupéré sera complètement embrouillé. Un système qui utilise le mode OFB doit donc avoir un mécanisme de détection de perte de synchronisation et de resynchronisation.

6 Les modes opératoires du Triple-DES

A part les quatre modes opératoires classiques présentés ci-dessus, il existe encore trois modes spéciaux réservés à l'algorithme de chiffrement Triple-DES. Ce sont en fait des variantes du CBC, CFB et OFB qui permettent de travailler plus rapidement en faisant effectuer une partie des calculs en parallèle.

Comme dans la plupart des modes opératoires standards, les modes opératoires du Triple-DES nécessitent des vecteurs d'initialisations VI. Or, pour optimiser le Triple-DES, on travaille généralement avec trois processeurs différents. Le premier et le troisième processus font le cryptage DES avec les clés K_1 respectivement K_3 et le deuxième le décryptage DES avec la clé K_2. Il faut donc disposer de trois vecteurs d'initialisation notés VI_1, VI_2 et VI_3.

On les génère aléatoirement, ou bien on utilise une valeur VI d'un compteur qui augmente ou diminue à chaque occurrence de façon à garantir de ne jamais avoir deux fois le même vecteur d'initialisation pendant la durée de vie de la clé cryptographique utilisée. Dans ce cas, on obtient les trois vecteurs par les manipulations suivantes (ici on donne les vecteurs en notation hexadécimale) :

- $VI_1 = VI$;
- $VI_2 \equiv VI_1 + R_1 \pmod{2^{64}}$, avec $R_1 =$(5555555555555555) ;
- $VI_3 \equiv VI_1 + R_2 \pmod{2^{64}}$, avec $R_2 =$(aaaaaaaaaaaaaaaa).

6.1 Le mode de chiffrement avec chaînage de blocs interlacé (TCBC-I)

Afin d'augmenter la performance du mode CBC, on répartit le message en clair en trois sous-flots différents dont on interlace alors le cryptage et le décryptage avec l'aide de trois processeurs.

La division du message en clair en trois sous-flots de données se fait de la façon suivante :

Pour $i = 1, ..., n$, on pose

$$\begin{array}{l} j_i \equiv i \pmod 3 \\ \text{Si } j_i = 0, \text{alors } j_i = 3 \\ h_i = 1 + \frac{i - j_i}{3} \\ M[j_i, h_i] = M[i]. \end{array}$$

Cela revient tout simplement à réindexer les blocs du texte en clair. En effet, le texte en clair $M = (M[1], M[2], ..., M[n])$ s'écrit maintenant

$$M = (M[1,1], M[2,1], M[3,1], ..., M[1,h], M[2,h], M[3,h], ..., M[j_n, h_n]).$$

Pour chiffrer un texte en mode TCBC-I, on effectue alors les opérations suivantes pour chacun des 3 sous-flots de données, c'est-à-dire, pour $j = 1, 2, 3$:

$$\begin{array}{l} C[j,0] = VI_j \\ C[j,h] = e_{K_3}\Big(d_{K_2}\big(e_{K_1}(M[j,h] \oplus C[j,h-1])\big)\Big), \quad \text{pour } h = 1, ..., n_h. \end{array}$$

De cette façon, on obtient trois flots de texte chiffré qu'il faut alors réindexer par la transformation inverse de celle explicitée ci-dessus afin d'obtenir le texte chiffré final.

Pour déchiffrer un texte chiffré en mode TCBC-I, on scinde d'abord le texte chiffré en trois sous-flots de données en utilisant la réindexisation vue ci-dessus, puis on effectue

les opérations suivantes pour chacun des 3 flots de données, c'est-à-dire, pour $j = 1, 2, 3$:

$$C[j,0] = VI_j$$
$$P[j,h] = d_{K_1}\Big(e_{K_2}\big(d_{K_3}(C[j,h] \oplus C[j,h-1])\big)\Big), \quad \text{pour } h = 1, ..., n_h.$$

De cette façon, on obtient trois flots de texte en clair qu'il faut alors réindexer par la transformation inverse de celle explicitée ci-dessus afin d'obtenir le texte en clair final.

Dans le mode TCBC-I, une erreur d'un bit dans un bloc de texte chiffré $C[j, h-1]$ va affecter deux blocs de texte en clair déchiffrés, à savoir les blocs $M[j, h-1]$ et $M[j, h]$. Dans le bloc $M[j, h-1]$ en moyenne 50 pour cent des bits seront mal déchiffrés, alors que dans le bloc $M[j, h]$ seulement le bit correspondant à l'erreur dans le texte chiffré sera faux.

Remarquons que les blocs $M[j, h-1]$ et $M[j, h]$ se suivent dans leur sous-flot de données, mais qu'il y a deux blocs entre eux dans le texte en clair recomposé.

Tout comme le mode CBC ordinaire, le mode TCBC-I n'est pas capable de récupérer les erreurs de synchronisation. Si un bit est ajouté ou perdu dans le flot de texte chiffré, alors tous les blocs qui suivent sont décalés d'un bit et deviennent indéchiffrables.

6.2 Le mode de chiffrement à rétroaction avec utilisation d'une pipeline (TCFB-P)

Pour chiffrer un texte en mode TCFB-P à k bits, on effectue les opérations suivantes (ici $M[\cdot]$ et $C[\cdot]$ sont des blocs de k bits) :

$$\begin{array}{ll} I[n-1] = VI_n, & \text{pour } n = 1, 2, 3 \\ I[n-1] = (I[n-2] \ll k) | C[n-3], & \text{pour } n \geq 4 \\ C[n] = M[n] \oplus MSB_k\big(e(I[n])\big), & \text{pour tout } n \geq 1. \end{array}$$

Remarquons que dans les formules ci-dessus e désigne l'algorithme de chiffrement du Triple-DES, donc $e(m) = e_{K_3}\Big(d_{K_2}\big(e_{K_1}(m)\big)\Big)$, où e_K et d_K désignent les fonctions de chiffrement respectivement de déchiffrement du DES avec la clé K. Cette procédure est organisée en pipeline en utilisant trois processeurs différents pour les 3 fonctions de chiffrement et de déchiffrement du DES.

Pour déchiffrer un texte en mode TCFB-P à k bits, on effectue les opérations suivantes :

$$\begin{array}{ll} I[n-1] = VI_n, & \text{pour } n = 1, 2, 3 \\ I[n-1] = (I[n-2] \ll k) | C[n-3], & \text{pour } n \geq 4 \\ M[n] = C[n] \oplus MSB_k\big(e(I[n])\big), & \text{pour tout } n \geq 1. \end{array}$$

Dans le mode TCFB-P une erreur d'un bit dans un bloc de texte chiffré $C[n]$ affecte le déchiffrement des $64/k$ blocs qui suivent $C[n+2]$. Après cela, la faute sera sortie du registre de décalage et n'aura plus d'effet sur le déchiffrement des blocs restants. Le bloc $P[n]$ comportera un bit faux au même endroit que l'erreur originale, tandis que les $64/k$ blocs à partir de $P[n+3]$ seront complètement embrouillés.

Tout comme le mode TCBC-I ordinaire, le mode TCFB-P n'est pas capable de récupérer les erreurs de synchronisation. Si un bit est ajouté ou perdu dans le flot de texte chiffré, alors tous les blocs qui suivent sont décalés d'un bit et deviennent indéchiffrables.

6.3 Le mode de rétroaction de sortie interlacé (TOFB-I)

Contrairement au mode de chiffrement avec chaînage de blocs interlacé, le mode de rétroaction de sortie interlacé ne nécessite pas de scinder le flot de données en trois sous-flots. Ceci est dû au fait que dans ce mode, on utilise une pipeline et qu'en fait, pour la rétroaction de sortie, les modes interlacés et avec pipeline sont identiques.

De plus, seul le mode avec chaînage à $k = 64$ bits est défini. Pour chiffrer un texte en mode TOFB-I, on effectue donc les opérations suivantes :

$$\begin{array}{ll} I[n-1] = VI_n, & \text{pour } n = 1, 2, 3 \\ I[n-1] = e(I[n-4]), & \text{pour } n \geq 4 \\ C[n] = M[n] \oplus e(I[n-1]), & \text{pour tout } n \geq 1. \end{array}$$

Pour déchiffrer un texte en mode TOFB-I, on effectue les opérations suivantes :

$$\begin{array}{ll} I[n-1] = VI_n, & \text{pour } n = 1, 2, 3 \\ I[n-1] = e(I[n-4]), & \text{pour } n \geq 4 \\ M[n] = C[n] \oplus e(I[n-1]), & \text{pour tout } n \geq 1. \end{array}$$

Il n'y a pas de propagation d'erreur dans le mode TOFB-I. Une erreur d'un bit dans un bloc de texte chiffré entraîne une erreur dans le même bit du bloc correspondant de texte en clair et tout le reste du texte en clair reste inaffecté.

Ce mode ne permet par contre pas de récupérer les erreurs de synchronisation.

7 D'autres modes opératoires (non standardisés)

A côté des 4 modes standardisés, on peut trouver un tas d'autres propositions dans la littérature. Les plus courants sont sommairement décrits ci-dessous.

7.1 Le mode OFB avec compteur

Au lieu d'utiliser la sortie du chiffrement pour remplir le registre à décalage dans le mode OFB, on peut utiliser un compteur qui est augmenté d'une certaine valeur après le chiffrement de chaque bloc. Les propriétés de synchronisation et de propagation d'erreurs sont les mêmes que pour le mode OFB.

7.2 Le mode de chaînage de blocs (BC)

Pour utiliser un algorithme de chiffrement par blocs en mode de chaînage de blocs, en anglais "Block Chaining" (BC), il suffit de combiner par ou exclusif les blocs du texte en

clair avec la somme modulo 2 de tous les blocs de texte chiffré précédents. Tout comme pour le mode CBC, un vecteur d'initialisation démarre le processus.

Pour chiffrer un texte en mode BC, on effectue donc les opérations suivantes :

$$\begin{aligned} &C[1] = e(M[1] \oplus VI) \\ &I[1] = (0...0) \\ &I[n] = I[n-1] \oplus C[n-1], \quad \text{pour } n > 1 \\ &C[n] = e(M[n] \oplus I[n]), \quad \text{pour } n > 1. \end{aligned}$$

Pour déchiffrer un texte en mode BC, on effectue les opérations :

$$\begin{aligned} &M[1] = d(C[1]) \oplus VI \\ &I[1] = (0...0) \\ &I[n] = I[n-1] \oplus C[n-1], \quad \text{pour } n > 1 \\ &M[n] = d(C[n]) \oplus I[n], \quad \text{pour } n > 1. \end{aligned}$$

Tout comme pour le mode CBC, le processus de rétroaction du mode BC a la propriété d'amplifier les erreurs dans le texte en clair. Le défaut principal du mode BC est que le déchiffrement d'un bloc dépend de tous les blocs précédents. La moindre erreur dans le déchiffrement falsifie donc toute la suite du travail.

7.3 Le mode de chaînage de blocs avec propagation (PCBC)

Le mode de chaînage de blocs avec propagation, en anglais "Propagating Cipher Bloc Chaining" (PCBC) est similaire au mode CBC, sauf que le bloc de texte chiffré précédent et le bloc de texte en clair précédent sont tous les deux combinés par ou exclusif avec le bloc de texte en clair courant.

Pour chiffrer un texte en mode PCBC, on effectue donc les opérations suivantes :

$$\begin{aligned} &C[1] = e(M[1] \oplus VI) \\ &C[n] = e(M[n] \oplus C[n-1] \oplus M[n-1]), \quad \text{pour } n > 1. \end{aligned}$$

Pour déchiffrer un texte en mode PCBC, on effectue les opérations :

$$\begin{aligned} &M[1] = d(C[1]) \oplus VI \\ &M[n] = d(C[n]) \oplus C[n-1] \oplus M[n-1] \quad \text{, si } n > 1. \end{aligned}$$

Le mode PCBC était utilisé dans Kerberos version 4 pour effectuer à la fois le chiffrement et le contrôle d'intégrité. En effet, en mode PCBC, une erreur dans le texte chiffré occasionne un déchiffrement incorrect de tous les blocs qui suivent. Cela implique qu'un bloc standard à la fin du message permet de s'assurer de l'intégrité de tout le message.

Malheureusement, le mode PCBC n'est pas sûr. En effet, échanger deux blocs de texte chiffré occasionne un déchiffrement incorrect des deux blocs de texte en clair correspondants, mais à cause de l'addition modulo 2 avec le texte en clair et le texte chiffré, ces erreurs se compensent et s'annulent. Le système de vérification d'intégrité peut donc être amené à accepter des messages partiellement embrouillés. Pour cette raison, le mode PCBC a été remplacé par le mode CBC dans Kerberos version 5.

7.4 Le carnet de codage avec décalage (OCB)

Le carnet de codage avec décalage, en anglais "Offset Codebook Mode" (OCB), proposé fin 2001, est un mode de chiffrement facilement parallélisable qui garantit en même temps l'authenticité et l'intégrité des messages. De plus, le mode OCB permet de chiffrer des flots continues de données tout comme les modes CFB et OFB. Le carnet de codage avec décalage utilise un vecteur d'initialisation VI, choisi par l'encrypteur et envoyé en clair avec le message. Comme toujours, ce vecteur doit être différent pour chaque chiffrement avec la même clé.

Soit m le nombre de blocs de texte en clair à chiffrer. Notons $\otimes$ la multiplication de deux vecteurs dans $GF(2^n)$ et γ_n le code de Gray canonique (Pour une définition détaillé de ces concepts, voir l'article "OCB : A Block-Cipher Mode of operation for Efficient Authenticated Encryption" de P.Rogaway, M. Bellare, J. Black et T. Krovetz. La fonction $\text{len}(M)$ donne la longueur du texte M.

Pour chiffrer un texte en mode OCB à k bits, on effectue les opérations suivantes :

$$
\begin{array}{ll}
L = e(0, ..., 0) & \\
R = e(VI \oplus L) & \\
I[n] = \gamma_n \otimes L \oplus R, & \text{pour } n = 1, ..., m \\
C[n] = e(M[n] \oplus I[n]) \oplus I[n], & \text{pour } n = 1, ..., m-1 \\
X[m] = \text{len}(M[m]) \oplus L \otimes x^{-1} \oplus I[m] & \\
C[m] = e(X[m]) \oplus M[m] & \\
T = MSB_k\Big(e\big(M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]0^* \oplus e(X[m]) \oplus I[m]\big)\Big) &
\end{array}
$$

(ici $C[m]0^*$ veut dire qu'on ajoute des 0 à droite dans $C[m]$, de façon à obtenir un bloc complet).

Puis, on envoie le message crypté concaténé au vecteur contrôle T.

Pour déchiffrer un texte en mode OCB à k bits, on effectue les opération suivantes :

$$
\begin{array}{ll}
L = e(0, ..., 0) & \\
R = e(VI \oplus L) & \\
I[n] = \gamma_n \otimes L \oplus R, & \text{pour } n = 1, ..., m \\
M[n] = d(C[n] \oplus I[n]) \oplus I[n], & \text{pour } n = 1, ..., m-1 \\
X[m] = \text{len}(M[m]) \oplus L \otimes x^{-1} \oplus I[m] & \\
M[m] = e(X[m]) \oplus C[m] & \\
T' = MSB_k\Big(e\big(M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]0^* \oplus e(X[m]) \oplus I[m]\big)\Big). &
\end{array}
$$

Si $T = T'$ on accepte le texte en clair obtenu, sinon on le rejette.

Un des avantages principaux de ce mode est qu'il est capable de chiffrer un texte de n'importe quelle longueur sans avoir besoin de techniques de remplissage de blocs. Par ailleurs, ce mode a été conçu de façon à ce qu'on puisse prouver sa sécurité dans le sens du schéma cryptologique de Bellare et Rogaway.

7.5 Quelques modes rarement utilisés

Il existe d'autres modes qui ne sont cependant pas couramment utilisés. Le mode de chaînage de texte en clair, en anglais "Plaintext Block Chaining" (PBC) fonctionne comme le mode CBC, sauf que le bloc de texte en clair à chiffrer est combiné par ou exclusif avec le bloc de texte en clair précédent au lieu du texte chiffré.

Le mode de chiffrement à rétroaction du texte en clair, "Plaintext Feedback" (PF) est comme le mode CFB, sauf que c'est le texte en clair qui remplit le registre à décalage et non le texte chiffré.

Mentionnons également les modes de chiffrement par blocs avec chaînage des différences entre blocs de texte en clair, en anglais "Cipher Block Chaining of Plaintext Difference" (CBCPD) et de rétroaction de la sortie avec fonction non linéaire, en anglais "Output Feedback with a Non-Linear Function" (OFBNLF).

8 Choix d'un mode opératoire

Le mode ECB est le plus facile et le plus rapide à utiliser. Compte tenu des problèmes de sécurité, on l'utilise cependant uniquement pour coder des fichiers aléatoires peu volumineux, comme par exemple des clés.

Les modes CBC, CFB et OFB sont tous les trois couramment utilisés, suivant les besoins spécifiques. Le mode CBC est juste un peu plus lent que le mode ECB, la seule différence résidant dans un registre mémoire de la taille d'un bloc et une opération de ou exclusif par bloc. L'augmentation du niveau de sécurité est par contre significative et bien qu'il puisse y avoir des erreurs de bit dans les données stockées, il y a rarement des erreurs de synchronisation. Pour une réalisation logicielle, c'est presque toujours le meilleur choix.

Les modes OFB et CFB sont tous les deux significativement plus lents que le mode CBC et sont surtout utilisés s'il faut chiffrer les données en continu, sans pouvoir toujours attendre d'être en présence d'un bloc complet. Le mode CFB est généralement le mode de choix pour chiffrer en continu des flots de caractères quand chaque caractère doit être traité individuellement, comme c'est le cas pour le lien entre un terminal et un ordinateur central. Le mode OFB est généralement utilisé pour les systèmes synchrones à grande vitesse où la propagation d'erreurs est inacceptable.

Remarquons que les 4 modes opératoires classiques garantissent la confidentialité des messages, mais pas l'intégrité. Si on a besoin également d'intégrité, il faut utiliser un protocole approprié ou chiffrer le message avec une procédure telle le mode OCB qui a été spécialement conçu pour garantir l'intégrité.

Un de ces quatre modes de base est adéquat pour presque n'importe quelle application et il est fortement recommandé de rester à l'écart des autres modes qui sont moins bien étudiés et qui de toute façon n'ont pas de meilleures propriétés pour la propagation d'erreurs et pour les mécanismes de récupération d'erreurs.