Combinaison de CBC et CBC-MAC

En cours, nous avons vu que la combinaison "encrypt-then-MAC" d'un chiffrement IND-CPA avec un MAC SUF-CMA nous donnait un chiffrement authentifié IND-CCA, et par ailleurs impossible à contrefaire (il est donc impossible pour un adversaire de créer une nouvelle paire "message, tag" valide). Cependant, de mauvaises combinaisons peuvent aboutir à des attaques. Nous donnons ici l'exemple de CBC combiné avec ECBC-MAC en mode "encrypt-and-MAC", dans lequel :

- On chiffre le message avec CBC ;
- On appelle le MAC *sur le message clair* ;
- On renvoie les deux résultats.

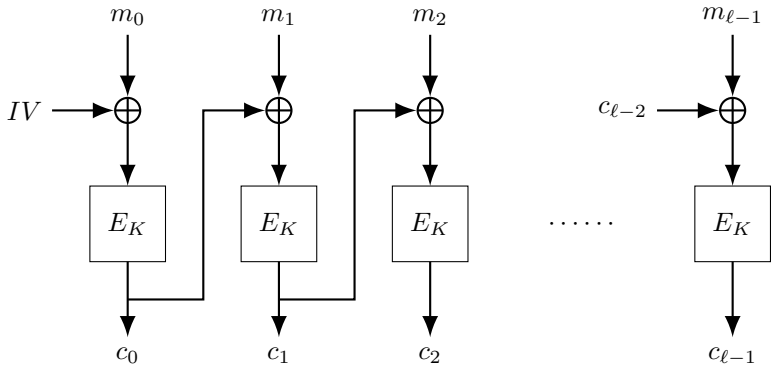

FIGURE 1 – Le mode CBC.

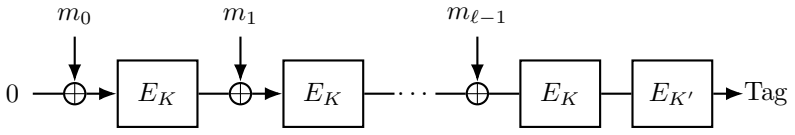

FIGURE 2 – ECBC-MAC.

Suppsons qu'Alice chiffre un message à deux blocs (P_1, P_2) avec la clé K et une IV aléatoire et l'envoie à Bob.

Question 1. *Donner l'expression du chiffré (C_1, C_2) et du tag T. Montrer que si $IV = 0$, $T = E_{K'}(C_2)$.*

Question 2. *En déduire qu'un attaquant Eve peut créer un nouveau triplet (C_1', C_2', T') valide.*

Question 3. *On suppose maintenant qu'on utilise encrypt-then-MAC. Donner la nouvelle expression du tag. Montrer qu'il y a toujours un problème avec cette construction ; en particulier qu'on peut casser la sécurité IND-CPA.*

Attaque par Oracle de remplissage sur CBC

On considère le mode CBC avec un chiffrement à bloc de n octets. Un texte en clair n'a pas forcément une longueur multiple de la taille des blocs. Pour contourner ce problème, on utilise un remplissage (*padding*) : la norme PKCS7 stipule que l'on remplit les octets restants par le **nombre d'octets nécessaires**. Par exemple, "Hello World" deviendra "Hello World`0x0505050505`" pour compléter en blocs de 8 octets, où `0x05` est la notation hexadécimale pour le nombre 5. Dans la suite le i-ème octet d'un bloc m pourra être noté $m[i]$.

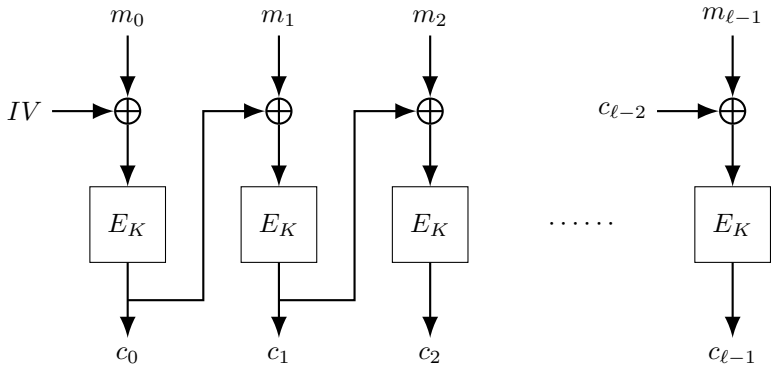

FIGURE 3 – Le mode CBC.

Soit Bob un serveur et Alice un client qui communiquent à l'aide de CBC, combiné avec un MAC sûr. L'adversaire Eve intercepte un message secret envoyé par Alice et communique ensuite avec le serveur Bob. Lorsque Eve (se faisant passer pour Alice) envoie un message à Bob, celui-ci effectue les opérations suivantes :

1. Déchiffrement du message avec CBC ;
2. Vérification que le padding est correct. Si ce n'est pas le cas, renvoyer "Erreur" à Alice ;
3. Vérification du tag. Si le tag est incorrect, renvoyer "Erreur" à Alice.

Question 4. *Montrer qu'en étudiant le temps de réponse de Bob, Eve peut utiliser le serveur comme un* oracle de padding $O^{padding}$ *qui prend en entrée un texte chiffré c et renvoie :*

- $\top$ *si le padding du texte clair correspondant est correct (c'est-à-dire s'il se termine par le bon nombre de symboles "*`0x0i`*")*
- $\bot$ *sinon*

Soit $C = (c_0, \ldots, c_{t-1})$ un texte chiffré intercepté. Nous nous concentrons d'abord sur le déchiffrement de c_{t-1}, le dernier bloc de C. Soit $C' = [r_0\| \ldots \|r_{n-1}], c_{t-1}$ un (faux) chiffré à deux blocs produit par Eve. Ici $[r_0\| \ldots \|r_{n-1}]$ est une concaténation de n octets r_i choisi par Eve, et c_{t-1} est un bloc complet récupéré sur le chiffré intercepté. Soit $(m'_0, m'_1) = \text{CBC}^{-1}(C')$ (le déchiffrement légitime par Bob).

Question 5. *Donner une relation entre $m'_1, c_{t-2}, [r_0\| \ldots \|r_{n-1}]$ et m_{t-1} le dernier bloc du texte en clair correspondant à c.*

Question 6. *On prend $r_0 = \ldots = r_{n-2} = 0$ et on fait varier r_{n-1}. Supposons qu'il existe une seule valeur de cet octet pour laquelle $O^{padding}(C') = \top$. Montrer qu'on peut en déduire le dernier octet de m_{t-1}.*

Question 7. *Montrer qu'on peut obtenir le dernier octet de m_{t-1} dans tous les cas en (au plus) 257 requêtes à $O^{padding}$.*

Question 8. *Montrer comment configurer $m'_1[n-1]$ à* `0x02` *et l'utiliser pour récupérer $m_{t-1}[n-2]$. En déduire comment récupérer m_{t-1} entièrement.*

Question 9. *Pouvons-nous récupérer tous les blocs du message ?*

On utilise maintenant une nouvelle fonction de padding. Étant donné un message M, si b est le dernier bit de M (0 ou 1), on complète M avec le complémentaire de b. Si M est vide, on renvoie un bloc de zéros. Si M se termine par un bloc complet, on ajoute un bloc de zéros. Ce padding est évidemment correct (et réversible, c'est-à-dire qu'on peut toujours revenir d'un message paddé au message initial).

Question 10. *L'attaque par oracle de padding s'applique-t-elle dans ce cas ? Pourquoi ?*

Remark 1. L'attaque de remplissage sur CBC s'est avérée très puissante en pratique, et a été la source de nombreuses vulnérabilités, notamment en combinaison avec une attaque de *downgrade* sur un protocole.

Réseaux de Feistel

Soient $G_i : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^\ell$ des familles de fonctions, et soit $d \geq 1$ un entier. Le réseau de Feistel de profondeur d associé à G_i est la famille de fonctions $F^{(d)} : \{0,1\}^{2\ell} \times \{0,1\}^{\ell d} \to \{0,1\}^{2\ell}$ définie comme suit :

$$F^{(d)}((K_i)_{i\in[1,d]}, x) : \begin{cases} 1: & L_0||R_0 \leftarrow x \\ 2: & \text{Pour } i \in [1,d] \text{ faire} \\ 3: & \quad L_i \leftarrow R_{i-1}, \quad R_i \leftarrow G_i(K_i, R_{i-1}) \oplus L_{i-1} \\ 4: & \text{Retourner } L_d||R_d \end{cases}$$

Question 11. *Dessinez une représentation d'un réseau de Feistel de profondeur 3.*

Question 12. *Montrez qu'un réseau de Feistel est inversible, même si les fonctions G_i ne le sont pas.*

Les réseaux de Feistel sont un moyen de construire une permutation inverse de manière efficace à partir d'un ensemble de fonctions pseudo-aléatoires. Dans le reste de cet exercice, nous supposons que $G_i(K, \cdot)$ est une famille de fonctions pseudo-aléatoires.

Question 13. *Rappeler (brièvement) la définition d'une PRF.*

Question 14. *Montrer que ni $F^{(1)}$ ni $F^{(2)}$ ne sont des PRF.*

Question 15. *On considère maintenant le jeu PRP joué par un adversaire PPT. On commence par remplacer les fonctions $G_i(K_i, \cdot)$ par de vraies fonctions aléatoires G'_i. Justifier que la probabilité que, pour deux requêtes différentes de l'adversaire, on ait une collision en R_1 (c'est-à-dire $R_1^i = R_1^j$), est négligeable.*

Question 16. *Justifier que $F^{(3)}$ est une PRF (la preuve complète étant très technique, on demande seulement une intuition ici).*

Question 17. *Trouvez une attaque qui distingue $F^{(3)}$ d'une permutation aléatoire, si l'on autorise accès à la permutation inverse (on montre ainsi que $F^{(3)}$ n'est pas une permutation pseudo-aléatoire « forte » , strong PRP – en revanche, c'est le cas de $F^{(4)}$).*