ISEC - Session 2
20 juin 2019

Énoncé

Notes manuscrites et documents du cours autorisés à l'exclusion de toute autre document
L'utilisation de tout matériel électronique (en dehors d'une montre non connectée) est interdite

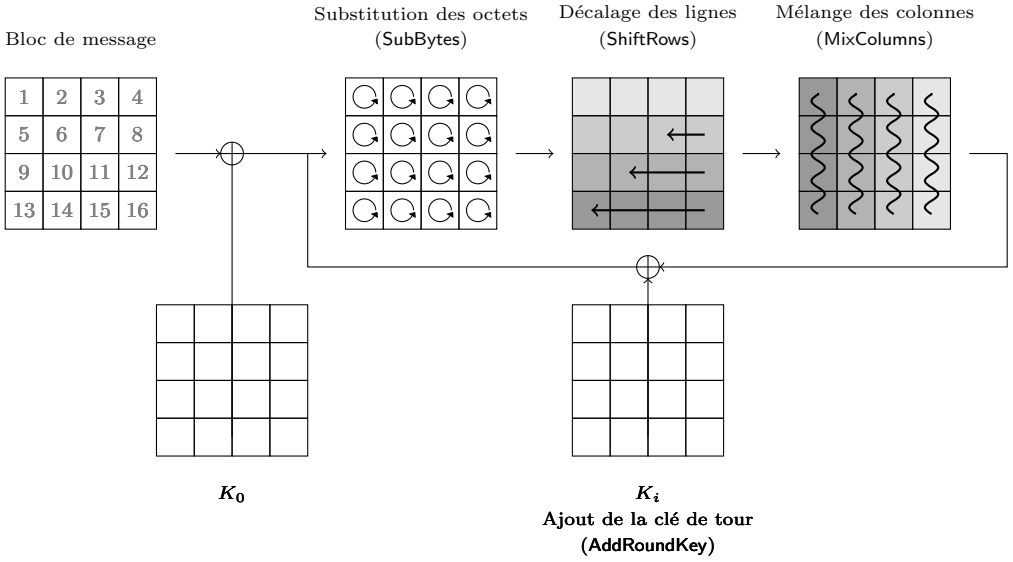
Les exercices sont indépendants. Une rédaction claire et concise sera appréciée. Toute affirmation devra être justifiée. Une question non résolue n'empêche pas de faire les suivantes (dans ce cas indiquez clairement que vous admettez le(s) résultat(s) de la question non faite).

Exercice 1 : Chiffrement AES

Nous considérons le chiffrement AES vu en cours (avec une clé de 128 bits pour des blocs de 128 bits). Le chiffrement découpe chaque bloc de 128 bits en 16 octets et chaque bloc est représenté sous forme d'une matrice carrée 4×4 où les octets sont numérotés de 1 à 16 de gauche à droite et de haut en bas.

Ce bloc est initialement le bloc de message clair à chiffrer auquel on additionne bit à bit une clé de tour initiale K_0. Le chiffrement AES applique ensuite de façon répétée quatre procédures appelées SubBytes, ShiftRows, MixColumns et AddRoundKey sur la matrice carrée 4×4 appelée *matrice état* :

- la procédure « substitution des octets » (SubBytes) applique sur chacun des seize octets de l'état interne une S-boîte ;
- la procédure « décalage de lignes » (ShiftRows) effectue des rotations vers la gauche sur les lignes de la matrice état ;
- la procédure « mélange des colonnes » (MixColumns) combine les 4 octets de chaque colonne de la matrice état en appliquant une transformation linéaire ;
- la procédure « ajout de la clé de tour » (AddRoundKey) combine l'état interne avec une clé différente pour chaque tour.



La composition de ces quatre opérations est appelée un tour et le chiffrement AES pour une clé de 128 bits est composé de 10 tours. La procédure MixColumns est omise lors du dernier tour. Les clés de tour K_i sont des blocs de 128 bits différents à chaque tour.

1.a] Expliquer pourquoi la procédure MixColumns est omise lors du dernier tour.

1.b] Dans le chiffrement AES, la procédure ShiftRows est conservée lors du dernier tour. Dire si elle pourrait être omise lors du dernier tour (si c'est le cas, justifier la sécurité et si ce n'est pas le cas, proposer une attaque lorsqu'elle est omise).

1.c] Considérons maintenant une variante de l'AES où l'opération AddRoundKey est omise à tous les tours (mais les trois autres opérations sont effectuées normalement à chaque tour). Donner une attaque à un clair connu (i.e. en connaissant un bloc de clair et le bloc de chiffré correspondant) qui permet de retrouver K_0.

1.d] Considérons maintenant une variante de l'AES où l'opération MixColumns est omise à tous les tours (mais les trois autres opérations sont effectuées normalement à chaque tour). En considérant le chiffrement de deux messages différents dont les premières lignes sont égales (i.e. les octets 1,2,3 et 4), montrer que cette variante n'assure pas la sécurité sémantique.

1.e] Considérons maintenant une variante de l'AES où l'opération ShitRows est omise à tous les tours (mais les trois autres opérations sont effectuées normalement à chaque tour). Montrer que cette variante n'assure pas la sécurité sémantique.

1.f] Considérons enfin une variante de l'AES où l'opération SubBytes est omise à tous les tours (mais les trois autres opérations sont effectuées normalement à chaque tour). Montrer que cette variante n'assure pas la sécurité sémantique.

Exercice 2 : Signature RSA

Nous considérons un entier RSA $N = pq$ qui est le produit de deux nombres premiers p et q distincts impairs et dont valeur de la fonction indicatrice d'Euler est égale à $\varphi(N) = (p-1)(q-1)$. L'objectif de l'exercice est de proposer une variante plus efficace des schémas de signature basée sur RSA.

Soient a et b deux entiers premiers entre eux, inférieurs à N et premiers avec $\varphi(N)$.

2.a] Soit $m \in (\mathbb{Z}/N\mathbb{Z})^*$. Montrer que si l'on dispose d'une racine a-ième de $(m^b \bmod N)$ (i.e. un entier $\sigma \in \{1, \ldots, N-1\}$ tel que $\sigma^a = m^b \bmod N$), alors il est possible d'obtenir en temps polynomial (en $\log(N)$) une racine a-ième de m (i.e. un entier $\sigma^* \in \{1, \ldots, N-1\}$ tel que $\sigma^{*a} = m \bmod N$) **sans connaitre la factorisation de N (ou la valeur de $\varphi(N)$)**.

Indication : On pourra utiliser le théorème de Bézout qui assure l'existence de deux entiers relatifs u et v (inférieurs à N en valeur absolue) tels que $au + bv = 1$ et considérer l'entier $(m^u \cdot \sigma^v \bmod N)$.

2.b] Soient $m_1, m_2 \in (\mathbb{Z}/N\mathbb{Z})^*$ et notons $c = ab$. En considérant l'entier $m = m_1^b \cdot m_2^a \bmod N$, montrer que si l'on dispose d'une racine c-ième de m (i.e. un entier $\sigma \in \{1, \ldots, N-1\}$ tel que $\sigma^c = m \bmod N$), alors il est possible d'obtenir en temps polynomial (en $\log(N)$) une racine a-ième de m_1 et une racine b-ième de m_2 (i.e. deux entiers $\sigma_1, \sigma_2 \in \{1, \ldots, N-1\}$ tel que ${\sigma_1}^a = m_1 \bmod N$ et ${\sigma_2}^b = m_2 \bmod N$) **sans connaitre la factorisation de N (ou la valeur de $\varphi(N)$)**.

2.c] Donner un algorithme polynomial (en $\log(N)$) pour calculer une racine c-ième de m en connaissant la factorisation de N (ou la valeur de $\varphi(N)$).

2.d] En utilisant les questions précédentes, proposer un moyen d'accélérer la signature basée sur RSA de sorte qu'un signataire puisse authentifier deux messages pour essentiellement le coût de signer un message avec la signature RSA classique. Le nouveau schéma utilisera deux exposants publics différents et la signature indiquera (par un bit par exemple) quel exposant a été utilisé pour la signature.