Université de Tlemcen Faculté des Sciences Département d'Informatique	Module Sécurité Informatique L3 Année Universitaire 2017/2018 C. BEKARA

Examen de Rattrapage S2, durée 1h30, Documents et Téléphone non autorisés

NOM :	**PRÉNOM :**

Exercice 1 (12 pts) : Entourez **LA OU LES BONNES RÉPONSES/ BON CHOIX**

ATTENTION:
- Une **FAUSSE RÉPONSE/FAUX CHOIX** coché --> **-0.25**
- Si vous cochez la case **Autre:** Dans ce cas les choix précédents sont considérés comme faux), vous devez fournir **LA BONNE RÉPONSE**

Q1) Une fonction de hachage H ne doit pas être caractérisée par les propriétés suivantes:
a) Réversibilité
d) Facilité de déduire un message depuis son haché

Q2) Le chiffrement par bloc en mode CTR est caractérisé par
d) Si la valeur du compteur est répétée avec la même clé, ceci pourrait affaiblir la confidentialité du message

Q3) Une fonction d'intégrité HMAC est caractérisée par
b) Elle utilise des constantes prédéfinies

Q4) Le service de disponibilité permet de garantir que

e) Autre: La ressource (donnée, service, etc.) soit disponible/accessible au moment voulu

Q5) Soit H:$\{0,1\}^*$ -->$\{0,1\}^{\mathbf{128}}$ et F: $\{0,1\}^*$ -->$\{0,1\}^{\mathbf{192}}$ deux fonctions de hachage. Pour espérer trouver une collision où H(F(M1))=H(F(M2)) avec M1 différent de M2, il faut effectuer au maximum (au pire) un nombre d'opérations de hachage égale à :
e) Autre: $\mathbf{2^{129}+2}$

Q6) Un mot de passe peut servir à
a) Authentifier un utilisateur
b) Déchiffrer un message
c) Chiffrer un message

Q7) Soit H : $\{0,1\}^*$ --> $\{0,1\}^{\mathbf{96}}$ une fonction de hachage. Une collision sur H a été possible au bout d'un temps de **T** secondes. Sachant qu'une opération de hachage nécessite $\mathbf{2^{-15}}$ secondes, et que 2^{-24} **(1/2^{24})** de l'espace des valeurs hachés a été exploré avant de tomber sur la collision, la valeur de **T** est :
e) Autre:2^{57} + 2^{-15} secondes (ou 2^{57})

Q8) Un logiciel malveillant peut être invisible à un antivirus car
a) Sa signature n'est pas encore répertoriée dans la base virale b) Il est polymorphe
c) Il peut être caché par un root-kit

Q9) La fonction intégrité CBC-MAC possède une ou plusieurs des propriétés suivantes
c) En calculant le MAC sur deux messages identiques en utilisant la même (les mêmes) clé(s) on obtient le même code d'intégrité
d) La taille maximale du MAC généré dépend de la taille du bloc de chiffrement utilisé par CBC

Q10) La non-répudiation permet de garantir
a) Qu'une entité ne peut pas nier avoir envoyer un message
b) Qu'une entité ne peut pas nier avoir reçu un message

Q11) Dans une attaque par re-jeu (replay) un attaquant va
e) Autre : Intercepter un message envoyé à un instant t puis le ré-envoyer à un instant ultérieur t'>t

Q12) On a chiffré un message **M** en utilisant un chiffrement par flux (ex: RC4), où le message chiffré résultant **C** avait une taille de **44** octets (le VI n'étant pas inclus). La taille du message **M** est:
d) Autre : **44** octets

NOM :	PRÉNOM :

Exercice 2 (4 pts)

Une personne a chiffré un message clair **M** en un message **C** en utilisant le chiffrement en bloc avec le mode **ECB**, où la taille de la clé de chiffrement **K** utilisée est de **128** bits.

Q1) Un attaquant a remarqué qu'il existe **4** blocs identiques (ayant tous la même valeur) en **C**, que peut-il déduire**?**

R1) (1 pt)
il peut déduire que les 4 blocs en clairs dans M correspondant à ces 4 blocs chiffrés en C sont eux aussi identiques

Q2) Étant donné **C**, et sans connaître **K**, combien d'opérations de déchiffrement doit effectuer un attaquant au maximum (au pire) pour retrouver **M** (on notera qu'une seule opération de déchiffrement consiste à essayer de déchiffrer C en M)? (Donnez juste le résultat)

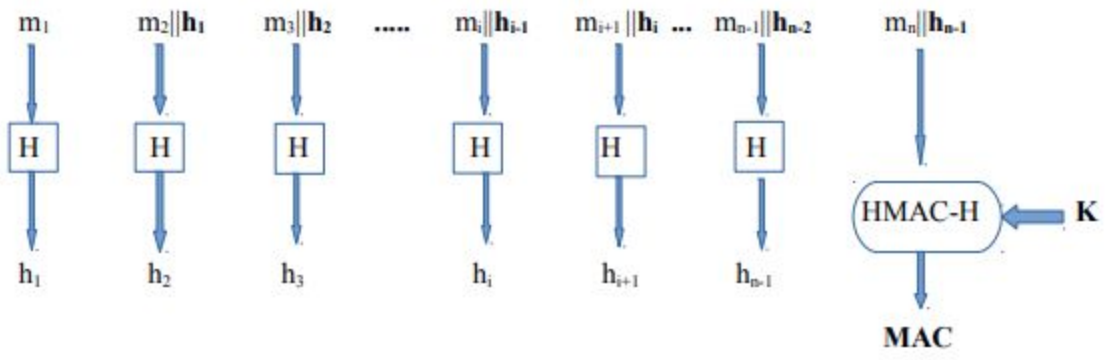
R2) $\mathbf{2^{128}}$ opérations de déchiffrement (**1** pt)

Q3) On suppose maintenant que l'attaquant possède certaines informations concernant **K** (mais ne connait toujours pas sa valeur): les **30** premiers bits de poids fort de **K** sont tous non-nuls, et les **10** derniers bits de poids faible de **K** sont tous nuls. Combien d'opérations de déchiffrement doit effectuer l'attaquant au maximum (au pire) pour retrouver **M** ? (Donnez juste le résultat)

R3) $\mathbf{2^{88}}$ opérations de déchiffrement (**2** pts)

Exercice 3:

Nous avons **n** messages m_1, m_2, ..., m_n de tailles quelconques que nous désirons garantir l'intégrité en générant un seul code d'intégrité MAC (au lieu de générer un MAC par message) en utilisant l'algorithme HASH_CONCAT_MAC décrit comme suit:



|| : concaténation / **H** : fonction de hachage / **HMAC-H**: fonction d'intégrité / **K** : clé secrète

Lors de la vérification de l'intégrité, le vérificateur dispose uniquement des **n** Messages (m_1, m_2, ..., m_{n-1},m_n) et le **MAC** généré, et utilise HASH_CONCAT_MAC pour s'assurer de l'intégrité des messages

Q1) Donner l'expression du MAC généré

R1) (**1** pt)
MAC= HMAC-H(K, $m_n||h_{n-1}$)
avec h_i=H($m_i||h_{i-1}$)
h_1=H(m_1)

Q2) Est-ce que la méthode décrite dans la Fig 1 permet de vérifier l'intégrité de chaque message **m_i** individuellement (utiliser uniquement **m_i**, **K** et **MAC** pour vérifier l'intégrité de **m_i**)? Justifiez

R2) (**1.5** pts)
Non, elle ne permet pas de vérifier l'intégrité de chaque m_i individuellement
En effet, pour pouvoir vérifier l'intégrité de **m_i**, on a besoin de calculer le haché **h_j** des autres messages **m_j** (**j=1 ... n**) ceci revient à dire que HASH_CONCAT_MAC permet de vérifier l'intégrité des messages m_1 ... m_n dans leur intégralité (tous à la fois) et non pas individuellement, donc soit le MAC est valide et dans ce cas on peut dire que l'intégrité de tous les messages est vérifiée, soit le MAC n'est pas valide, et dans ce cas on peut juste dire que au moins un **m_j** a été modifié mais on ne sait pas lequel

Q3) HASH_CONCAT_MAC possède une faiblesse qui, si exploitée par un attaquant, a comme conséquence que le vérificateur conclu que au moins un message a été modifié, bien qu'en réalité ni le MAC ni aucun des n messages n'ont été modifié. Expliquer cette faiblesse

R3) Il suffit pour un attaquant d'intercepter les n messages puis modifier au moins l'ordre de (permuter) deux messages **m_i** et **m_j** (j >i)
m_1, ... **m_i**, ...,**m_j**, ..., m_n, MAC → m_1, ..., **m_j**, ..., **m_i**, ..., m_n, MAC
Lors du calcul du MAC au niveau émetteur on avait: **h_i**=H(**m_i**||**h_{i-1}**) et **h_j**=H(**m_j**||h_{j-1})
Lors du calcul du MAC au niveau récepteur (après permutation des messages m_i et m_j) nous aurons
h_i=H(**m_j**||**h_{i-1}**) et **h_j**=H(**m_i**||h'_{j-1})
le hi et hj calculés au niveau du récepteur étant différents de ceux calculés au niveau de l'émetteur (aussi par propagation tous les **h_t** t>i), le récepteur va calculer un MAC différent de celui reçu (de celui calculé par l'émetteur) et concluera donc qu'au moins un message m_i a été modifié (**1.5** pts)