**M1 informatique – UE réseau et sécurité**
**Année universitaire :** 2021-2022
**Consignes :** documents interdits, calculatrices interdites, téléphones portables interdits.
**Durée :** 1h30

**Réseau et sécurité – Examen terminal – première session**

## Exercice 1 : Chiffrement de Vigenère (5 points)

**Question 1.1 (1 point) :** Décrivez le principe du chiffrement de *César* et sa faille.

**Question 1.2 (1 point) :** Expliquez en quoi le chiffrement de *Vigenère* surmonte la faille du chiffrement de *César* ?

**Question 1.3 (1 point) :** Supposons que nous connaissons la longueur *N* de la clé d'un chiffrement de Vigenère. Comment pouvons-nous casser un texte long chiffré avec cette clé ?

**Question 1.4 (2 points) :** Supposons que dans un texte chiffré avec Vigenère, avec une clé de longueur inconnue, on trouve les chaînes de caractères suivantes répétées au moins une fois (cf le tableau ci-dessous). Devinez les longueurs de clés possibles et expliquez votre raisonnement.

| Chaînes de caractères répétées | Nombre d'occurrences | Espaces entre les répétitions |
|---|---|---|
| CTX | 4 | 210, 75, 120 |
| BBT | 3 | 300, 455 |
| OPCTA | 3 | 56, 105 |
| ACH | 2 | 63 |

## Exercice 2 : Attaque *baby-step giant-step* en cryptographie sur courbes elliptiques (5 points)

L'attaque *baby-step giant-step* est une attaque sur les groupes cycliques, permettant, à partir d'un générateur *P* du groupe et d'un point *Q=k.P*, de calculer la valeur *k*. Dans cette attaque, on pose *m=ceil(sqrt(n))*, avec *n* l'ordre du groupe (*ceil* étant la partie entière supérieure et *sqrt* étant la racine carrée). Puis, on calcule tous les points *i.P* avec $0<=i<m$ et tous les points *Q-j.m.P* avec $0<=j<m$. Dès que l'on trouve un couple *(i,j)* tel que *i.P=Q-j.m.P*, on peut en déduire la valeur *k*.

**Question 2.1 (1 point) :** Prouvez qu'à la fin de l'attaque, la valeur recherchée est $k=i+j.m$.

**Question 2.2 (2.5 points) :** Calculez *i.P* pour $0<=i<m$, sachant que : $P=(5,1)$, $Q=(0,11)$, $n=19$, $a=2$ et $p=17$. Pour rappel, le point $C=A+B$ est défini ainsi (avec *A* et *B* différents de *O*) :

- si $A=B$, on a : $s=(3.x_A^2+a).((2.y_A)^{-1})\ [p]$, $x_C=s^2-2.x_A\ [p]$ et $y_C=s.(x_A-x_C)-y_A\ [p]$,
- si $A<>B$, on a : $s=(y_A-y_B).((x_A-x_B)^{-1})\ [p]$, $x_C=s^2-(x_A+x_B)\ [p]$ et $y_C=s.(x_A-x_C)-y^A\ [p]$.

**Remarque :** La table des inverses modulo 17 est donnée ci-dessous :

| *n* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n^{-1}$ *[17]* | 0 | 1 | 9 | 6 | 13 | 7 | 3 | 5 | 15 | 2 | 12 | 14 | 10 | 4 | 11 | 8 | 16 |

**Question 2.3 (0.5 point) :** Soit les points *Q-j.m.P* suivants : (0,11) pour *j=0,* (0,6) pour *j=1,* (6,3) pour *j=2,* (10,11) pour *j=3,* (13,10) pour *j=4*. Quelle est la valeur de *k* ?

**Question 2.4 (1 point) :** L'attaque *baby-step giant-step* est l'une des meilleures attaques connues actuellement sur les courbes elliptiques. Quelle est sa complexité (d'après la description faite ici) ? Quelle est la conséquence de cette complexité sur la taille des clés ?

**Exercice 3 : Fonctions de hachage incrémentales basées sur le chaînage de paires de blocs (10 points)**

Les fonctions de hachages incrémentales, non étudiées en cours, ont été introduites dans [1] dans l'objectif d'accélérer le calcul du haché de modifications de messages. A partir d'un message *M* avec un haché *h(M)* connu, l'objectif est de produire rapidement le haché *h(M*)* d'un message *M**, sachant que *M** est une légère modification de *M*. Le cas d'usage présenté dans [1] était la protection contre les virus : pour se propager, les virus modifient les fichiers de l'ordinateur infecté. Les fonctions de hachage sont utilisées pour vérifier l'intégrité des fichiers, et les fonctions incrémentales servent dans ce cas à recalculer les hachés rapidement à chaque fois que les fichiers de l'ordinateur sont modifiés (par un programme non malicieux).

**Question 3.1 (0.5 point) :** Expliquez en quoi les fonctions de hachage peuvent être utilisées pour vérifier l'intégrité des fichiers (de manière générale).

Le chaînage par paires de blocs [2] divise le message à hacher en *n* blocs, effectue le haché de chaque paire de blocs consécutifs, et combine ces hachés. Si *M[i]* désigne le *i*-ème bloc du message *M*, *h* une fonction de hachage, « || » la concaténation, et « + » l'opération XOR, alors le chaînage par paires de blocs produit le haché suivant :
*haché de M = h(M[1]||M[2]) + h(M[2]||M[3]) + … + h(M[n-1]||M[n])*.

**Question 3.2 (1 point) :** Montrez que si *M** est une extension d'un message *M* dont le haché *h(M)* est connu, c'est-à-dire que *M*=M||A*, alors on peut calculer le haché de *M** rapidement.

**Question 3.3 (0.5 point) :** Expliquez en quoi les fonctions de hachage non incrémentales nécessitent plus de calcul que les fonctions de hachage incrémentales ?

**Question 3.4 (1 point) :** Quel est l'intérêt d'un hachage par chaînage de blocs, comparé à un hachage des blocs du type : *haché-par-bloc de M = h(M[1]) + h(M[2]) + … + h(M[n])*.
**Indice :** Montrez que sur un hachage par blocs (sans chaînage), on peut facilement construire un message *M'* qui est en collision avec *M*, en réordonnant les blocs de *M*.

Plusieurs failles ont été identifiées dans la technique de chaînage de paires de blocs. La première faille [3] concerne les messages avec des blocs répétés : s'il existe dans *M* des blocs du type *A||B||A*, alors il existe une collision contenant *B||A||B*.

**Question 3.5 (1 point) :** Montrez qu'à partir d'un message *M* contenant *n* blocs (avec *n* grand) et contenant une séquence de trois blocs *A||B||A*, vous pouvez construire un message *M'* de *n* blocs ayant le même haché que *M*.

**Question 3.6 (1 point) :** Montrez qu'à partir d'un message *M* contenant *n* blocs (avec *n* grand) et contenant une séquence de cinq blocs *A||B||C||B||A*, vous pouvez construire un message *M'* de *n* blocs ayant le même haché que *M*.

**Question 3.7 (1 point) :** Montrez qu'à partir d'un message *M* contenant *n* blocs (avec *n* grand) et contenant une séquence de sept blocs *A||B||B||B||B||C*, vous pouvez construire un message *M'* de *n-2* blocs ayant le même haché que *M*.

La deuxième faille [3] est liée au mécanisme de *padding*. Cette faille fait l'hypothèse que tous les blocs de *M* sont distincts avant le *padding*, mais que le message *M* ne correspond pas à un nombre entier de blocs. Dans ce cas, le dernier bloc est complété par des 0. On supposera que deux blocs de taille différentes sont différents, même si les premières valeurs sont égales.

**Question 3.8 (1 point) :** Montrez qu'à partir d'un message *M* contenant un peu plus de deux blocs, vous pouvez construire un message *M'* de trois blocs ayant le même haché que *M*.

La troisième faille vise à construire à partir d'un message *M* connu, un message *M'* en collision avec *M*, tel que *M* contient un bloc *X* inséré entre les blocs *M[i]* et *M[i+1]*.

**Question 3.9 (1 point) :** Indiquez comment calculer le haché de *M'* à partir du haché de *M*.

**Question 3.10 (2 points) :** Proposez une solution pour empêcher les trois failles décrites précédemment.

[1] M. Bellare, O. Goldreich, S. Goldwasser. "Incremental cryptography: the case for hashing and signing". In Advances in cryptology – Crypto'94. Lecture notes in computer science, vol. 839. Germany: Springer-Verlag; 1994. p. 216—233.
[2] M. Bellare, O. Goldreich, S. Goldwasser. "Incremental cryptography and application to virus protection". In Proceedings of 27th ACM symposium on the theory of computing (STOC); 1995. p. 45—56.
[3] R. C.-W. Phan, D. Wagner. "Security considerations for incremental hash functions based on pair block chaining". In Computers & Security, volume 25. Elsevier; 2006. p. 131—136.