UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence Informatique
Info0603 – 2021/2022

Travaux dirigés et TP n° 6

Fonctions de hachage

Exercice 1 (Évaluation de la résistance aux collisions)
Un peu de mathématiques :

— Exprimer la probabilité $p(n)$ que n personnes aient chacune une date d'anniversaire différente en notant $N = 365$.

— Expliquer en quoi cette formule a une forme très peu utile dans la mise au point d'une méthode d'évaluation de clé de hachage.

— Retrouver, dans votre mémoire, une formule permettant de remplacer $(1-x)$ par une formule plus pratique au calcul lorsque x est très proche de 0.

— En déduire que $p(n) \approx \prod_{i=1}^{n} e^{-\frac{k}{n}}$.

— Retrouver dans votre mémoire la formule, très utile en informatique, permettant de calculer 1+2+...n en seulement quelques calculs.

— En déduire que $p(n) \approx e^{-\frac{n.(n-1)}{2.N}}$ et exprimer n puis N en fonction des autres paramètres.

— Proposer un moyen d'évaluer de combien varie chacun de ces paramètres lors qu'une des autres varie d'un pas donné.

— On souhaite travailler avec une probabilité de collision inférieure à 1%. En déduire le nombre de donnée hachable sur 8,16,32,64 et 128 bits. Construire le graphique correspondant.

Expliquer pourquoi connaitre l'espace d'une clé de hachage ne suffit pas à déterminer sa réelle résistance aux collisions. Proposer un indicateur et une méthode pour ce faire.

— Pour ce faire, construire une fonction qui génère des données aléatoires applique la fonction de hachage jusqu'à détecter une collision. La fonction renvoie alors le nombre n de données sans collisions.

— Construire une fonction applique la fonction de hachage sur des données réelles (à déterminer) jusqu'à détecter une collision. La fonction renvoie alors le nombre n de données sans collisions.

— En déduire la taille N des clés pour laquelle la probabilité de collision est de 1%.

Exercice 2 (Tests de fonctions de hachage)

— Les fonctions de hachage héritent-elles aussi de CodeurCA sans bien-sûr surcharger binDecode : Expliquer pourquoi.

— Proposer quelques fonctions de hachage (Une méthode binCode qui retourne un Binaire603 de taille fixe) et évaluer "à vue de nez" leurs qualités.

— pour chacune de ces fonctions proposer une méthode permettant, à partir d'un Binaire603, d'en déduire un autre ayant la même image par la fonction de hachage.

— Proposer une méthode renvoyant, à partir d'un Binaire603, un Binaire603 dont il serait l'image.

— Proposer un test aux collisions : est-il universel ?

— Proposer une méthode de test sur les mots en français.

Exercice 3 (Résistances !)
Montrer, par contra-posée, que la résistance aux collisions implique la résistance à la seconde pré- image qui implique la résistance à la pré-image.

Exercice 4 (Quelques codes vérificateur)

Le numéro de carte de crédit est son identifiant unique structuré selon la norme ISO/IEC 7812, qui se décompose ainsi :

les six premiers chiffres constituent le numéro d ?émetteur désigné par IIN pour Issuer Identification Number. Le premier chiffre identifie le type de carte : 3 pour les cartes American Express, 4 pour les cartes Visa, 5 pour les cartes Mastercard. Les chiffres suivants (9 à 12 chiffres) constituent l ?identifiant de la carte chez l ?émetteur et sont attribués par l ?émetteur lui-même. Les banques françaises utilisent exactement 9 chiffres comme identifiant dans la très grande majorité des cas. Le dernier chiffre est une clé de contrôle permettant de vérifier que le numéro de carte est conforme à la norme. Le code complet est vérifié selon l'algorithme de Luhn :

1. L'algorithme multiplie par deux un chiffre sur deux, en commençant par l'avant dernier et en se déplaçant de droite à gauche. Si le double d'un chiffre est plus grand que neuf (par exemple $28 = 16$), alors il faut le ramener à un chiffre entre 1 et 9 en prenant son reste dans la division euclidienne par 9. Pour cela, il y a 2 manières de faire (pour un résultat identique) :
   — Soit on additionne les chiffres composant le double. Dans l'exemple du chiffre 8, $28 = 16$, puis on additionne les chiffres $1 + 6 = 7$. Soit on soustrait 9 au double. Avec le même exemple, $16?9 = 7$.
2. La somme de tous les chiffres obtenus est effectuée.
3. Le résultat est divisé par 10. Si le reste de la division est égal à zéro, alors le nombre original est valide.

1. Programmer l'algorithme de Luhm
2. La somme de la clé de contrôle du numéro de Sécurité Sociale (NIR) (mais aussi du RIB et de l'IBAN) se calcule sinsi :(13 premiers chiffres du NIR)%97 est égale à 97.

Exercice 5 (Fonction de hachage à partir d'un CodeurCA)

1. Programmer une fonction de Hachage recevant en paramètre un CodeurCA et hachant selon la construction de Davies-Meyer
2. Faire de même selon une construction de Miyaguchi-Prenel.
3. Vérifier la qualité de ces constructions avec le "test" des bijections ainsi que celui des anniversaires.

Exercice 6 (Une mauvaise fonction de hachage)

Soit $f : {F_2}^m \to {F_2}^m$ une fonction quelconque. Soit H la fonction de hachage résultat de l'itération de l'itéreration de la fonction $\begin{array}{ccccc} g & : & {F_2}^{2m} & \to & {F_2}^m \\ & & x = (x_l, x_r) & \mapsto & f(x_r \oplus x_l) \end{array}$

1. Vérifier la qualité de cette fonction avec le "test" des bijections ainsi que celui des anniversaires.
2. Montrer que H n'est pas résistante à la seconde pré-image.

Exercice 7 (Hachage par le logarithme discret Voir Codage par B.Martin p234)

Soit la fonction h définie pour p premier et $q = \frac{p-1}{2}$ premier aussi,

α et β deux éléments primitifs de F_p (c'est à dire d'ordre p) par :

$$\begin{array}{ccccc} h & : & F_q F_p & \to & F_p^* \\ & & (x_1, x_2) & \mapsto & \alpha^{x_1} \beta^{x_2} \end{array}$$

— Déterminer le premier couple (p,q) pour p>40.

— Implémenter cette fonction et vérifier ses qualités de cette fonction avec le "test" des bijections ainsi que celui des anniversaires.

Exercice 8 (AES 1)

Facultatif : Programmer la fonction de hachage AES-1