Licence Informatique

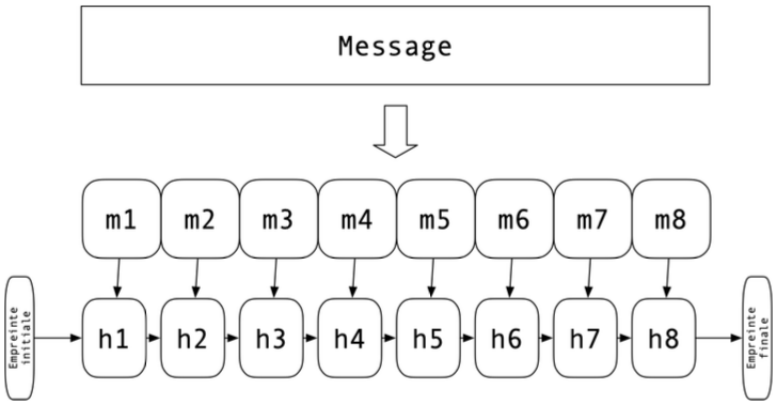
Introduction à la Cryptographie

TP3

1 Fonction de Hachage : adaptation de Toy Tetragraph Hash (TTH)

On considère une fonction de hachage inspirée de TTH que l'on nommera TTH_{64}^5 qui travaille sur des valeurs numériques modulo 64 (i.e $[0, ..., 63]$) et dont l'empreinte résultante est constituée de 5 valeurs. Par ailleurs, TTH_{64}^5 travaille sur une partition en blocs de 25 valeurs. TTH_{64}^5 considère un message clair M sous forme binaire. Dans l'hypothèse où la taille de M n'est pas multiple de 25, le bloc incomplet résultant de cette situation est complété par la valeur 32 puis autant de valeurs 0 que nécessaire pour le compléter. On parle dans ce cas de *padding* ou de *bourrage* de M.

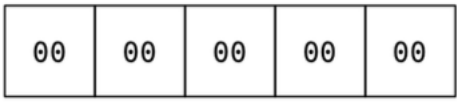
Le schéma ci-dessous vous donne le schéma général d'un chiffrement (ici un hachage) par blocs.



On considèrera à titre d'exemple que M est l'expression d'un message clair ré-encodé sur des valeurs entières non signées comprises entre 0 et 63. On pose :

$$M = \{00, 06, 08, 35, 17, 28, 24, 56, 62, 07, 12, 16, 20, 05, 33, 43, 35, 27, 12, 60, \\ 25, 23, 18, 01, 45, 56, 12, 34, 21, 20, 02, 10, 22, 20, 17, 34, 01\} \quad (1)$$

On considère également que l'empreinte est initialement un vecteur nul d'ordre 5, soit :



Les différentes étapes de $TTH_{64}^{5}(M)$ sont les suivantes :

A) Padding de M

Aprés la phase de *padding*, on a :

$$\begin{aligned} M_{+padding} = \{&00, 06, 08, 35, 17, 28, 24, 56, 62, 07, 12, 16, 20, \\ &05, 33, 43, 35, 27, 12, 60, 25, 23, 18, 01, 45, 56, 12, 34, 21, \\ &20, 02, 10, 22, 20, 17, 34, 01, 32, 00, 00, 00, 00, 00, 00, 00, \\ &00, 00, 00, 00, 00, \} \end{aligned} \tag{2}$$

B) Arrangement matriciel de $M_{+padding}$

$M_{+padding}$ est divisé en blocs successifs de 25 valeurs qui sont réparties dans des matrices 5×5 en suivant le sens de lecture de la gauche vers la droite et ligne par ligne. A partir de notre exemple, on obtient les matrices suivantes :

00	06	08	35	17
28	24	56	62	07
12	16	20	05	33
43	35	27	12	60
25	23	18	01	45

Bloc 1

56	12	34	21	20
02	10	22	20	17
34	01	32	00	00
00	00	00	00	00
00	00	00	00	00

Bloc 2

C) Calcul de l'empreinte pour le bloc en cours - étape 1

A l'empreinte courante est ajoutée la somme de chaque colonne en regard du bloc en cours. Ce cumul se fait modulo 64. Pour notre exemple, on a :

00	06	08	35	17
28	24	56	62	07
12	16	20	05	33
43	35	27	12	60
25	23	18	01	45
↓	↓	↓	↓	↓
44	40	01	51	34

D) Calcul de l'empreinte pour le bloc en cours - étape 2

Les lignes 1 à 4 du bloc en cours sont décalées circulairement vers la gauche de son numéro d'indice. Rmq : l'indiçage commence à 1. Sur la dernière ligne, l'ordre est inversée. Pour notre premier bloc dans l'exemple on a :

17	00	06	08	35
62	07	28	24	56
20	05	33	12	16
35	27	12	60	43
45	01	18	23	25

E) Calcul de l'empreinte pour le bloc en cours -

étape 3

A l'instar de l'étape C), à l'empreinte courante est ajoutée la somme de chaque colonne en regard du bloc en cours. Ce cumul se fait modulo 64. Pour notre exemple, on a :

17	00	06	08	35
62	07	28	24	56
20	05	33	12	16
35	27	12	60	43
45	01	18	23	25
↓	↓	↓	↓	↓
31	16	34	50	17

F) Passage au bloc suivant

S'il reste des blocs à traiter, aller à étape C) avec le bloc suivant. Pour notre exemple, on traite le bloc 2 avec l'empreinte en cours obtenue precedemment :

31	16	34	50	17

1.1 Collisionnement de TTH_{64}^5 et preuve de travail

- En se basant sur le paradoxe des anniversaires, donnez la fonction estimant la probabilité de collisionnement de TTH_{64}^5 avec une probabilité p.
- Quel sera l'effort à fournir pour générer un M tel que $TTH_{64}^5(M)$ débute par les valeurs 1 puis 2 puis 3 ?

1.2 Implémentation de TTH_{64}^5

Implémentez, dans le langage de votre choix - si possible en langage C - TTH_{64}^5

1.3 Collision de TTH_{64}^5

En vous basant sur l'algorithme vu en cours, tentez de collisionner TTH_{64}^5. Autrement dit, fournissez, à l'aide de votre implémentation de la question précédente, 2 messages M_1 et M_2 t.q. $TTH_{64}^5(M_1) = TTH_{64}^5(M_2)$ avec $M_1 \neq M_2$. Quelle est la longueur du cycle ? Dans un deuxième temps, implémentez la version sans mémoire de l'algorithme.