« L'arnaque au président » est une technique d'escroquerie qui consiste à usurper l'identité d'un patron afin de convaincre un responsable financier de transférer de manière illégitime des fonds. Il y a quatre ans, une entreprise des Deux-Sèvres en a été victime, la conduisant au bord de la fermeture.

Afin de se prémunir contre « l'arnaque au président » le directeur et les responsables financiers d'une entreprise décident de mettre en oeuvre une technique d'authentification des emails qu'ils s'échangent. Ce protocole d'email authentifié, décrit sur la figure 1, implémente le principe du HMAC et s'appuie donc sur un *secret partagé*, noté S.

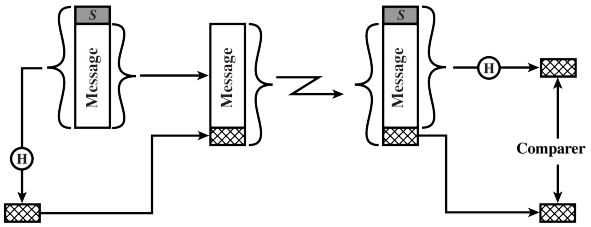

FIGURE 1 – Principe du protocole d'authentification d'un message par un HMAC

Au moment de l'envoi d'un email, un appendice h sera ajouté au courriel. Cet appendice est simplement le résumé MD5 de la concaténation du *corps* de l'email, noté c, et du secret S : $h = \text{MD5}(c\|S)$. À la réception d'un email avec un appendice h et un corps c, le résumé $h' = \text{MD5}(c\|S)$ est calculé puis comparé à l'appendice h reçu : si $h = h'$, l'email est jugé authentique (car il a été produit par une personne qui connait le secret S et qui a calculé le résumé de $c\|S$) ; sinon, une alerte est lancée, car le message est frauduleux.

Exercice A.1 Production d'un HMAC à la main Le fichier `email1.txt`, reproduit sur la figure 2, est disponible sur le site de l'UE « Cryptographie » dans le paquet `HMAC.zip`. Son contenu précis peut être visualisé sous Unix par la commande **`od -tc email1.txt`**. On y distingue, conformément à la RFC 2822, l'entête et le corps séparés par une *ligne vide*, c'est-à-dire une ligne contenant seulement le caractère « carriage return (CR) » (valeur ASCII 13, codé par `\r`) suivi du caractère « line feed (LF) » (valeur ASCII 10, codé par `\n`). Cette fin de ligne formée de deux caractères est usuelle dans les protocoles réseaux. L'entête comprend les champs habituels : `From`, `To`, `Subject`... ainsi qu'un champ non-standard : `X-Mailer`. Depuis les années 80 en effet (même si ce n'est plus recommandé) il est d'usage d'indiquer par un `X-` un champ non-standard, utilisé indépendamment du protocole sous-jacent.

```
Received: from 31.121.118.45 (EHLO serveur.fr)
  by mta1007.mail.ukl.yahoo.com with SMTP; Fri, 21 Sep 2019 21:31:16 +0000
Received: by serveur.fr (Postfix, from userid 106)
  id 3DF2F15A0CD; Fri, 21 Sep 2019 23:31:16 +0200 (CEST)
From: "Thomas Moore" <president@serveur.fr>
To: yvon.pigeon@yahoo.fr
Subject: Transfert d'argent urgent
Date: Fri, 21 Sep 2019 23:31:16 +0200
MIME-Version: 1.0
Content-Transfer-Encoding: 8bit
Content-Type: text/plain; charset=iso-8859-1
X-Mailer: Mozilla Thunderbird
Message-Id: <20190921212116.3DF2F16A0CD@serveur.fr>

Bonjour Yvon,
Transferez 100 000 euros sur le compte IBAN
FR7630056009271234567890182 des aujourd'hui.
Thomas
```

FIGURE 2 – Un exemple de courriel (RFC 2822)

Dans cet exercice et les suivants, le secret S est formé par 128 bits décrits par les 32 caractères hexadécimaux suivants : $S =$ **`c5dcb78732e1f3966647655229729843`**. Pour éviter une faute de frappe dans ce TP, vous pourrez utiliser le fait que S est en fait le résumé MD5 de la chaîne de caractères «**`Alain Turin`**», comme le montre l'exécution de la commande suivante :

```
$ echo -n "Alain Turin" | md5sum
c5dcb78732e1f3966647655229729843
```

Question 1. À l'aide de commandes Unix dans un terminal, calculez le résumé MD5 du *corps* de l'email fourni dans le fichier `email1.txt`. Notez ici le résultat : MD5$(c) =$ 70aa...

Vous pouvez vous appuyer ici sur le fait que l'entête comporte 13 lignes et le corps 4.

Question 2. À l'aide d'un éditeur de texte ou des commandes Unix appropriées, construisez un nouveau fichier constitué par le *corps* de cet email et une ligne supplémentaire qui ne contient que le secret S (sous la forme de 32 caractères hexadécimaux) sans CR+LF terminal. Puis calculez le résumé MD5 du fichier obtenu. Notez ici le résultat : $h =$ MD5$(c\|S) =$ 729...

Question 3. Modifiez le fichier `email1.txt` en ajoutant à la fin de son entête un nouveau champ, non-standard, formé par la chaîne de caractères `"X-AUTH: "` suivie des 32 caractères hexadécimaux du résultat h obtenu à la question précédente, avec naturellement un CR+LF de fin de ligne. Enregistrez sous le nom `email1-auth.txt` le fichier ainsi obtenu.

Exercice A.2 Calculs de HMAC (en C ou en Java) Cet exercice consiste à programmer (en C ou en Java, au choix) les manipulations effectuées à la main dans l'exercice précédent, sur un email quelconque. Les figures 3 et 4 illustrent comment il est possible de calculer, en C et en Java, le résumé MD5 d'un fichier en le lisant, de manière classique, morceau par morceau.

Question 1. Écrire un programme `cert` qui calcule et affiche le champ `"X-AUTH: `h`"` correspondant à un fichier email quelconque, précisé en paramètre. Vous devez en particulier programmer l'extraction du corps c de l'email indiqué en paramètre, afin de calculer le résumé $h =$ MD5$(c\|S)$.

Il s'agit ici essentiellement de lire un fichier texte ligne par ligne. En C, on recommandera d'utiliser **`getline()`** plutôt que **`fgets()`**. En Java, on pourra utiliser **`readLine()`** pour parcourir le fichier ligne par ligne et **`getBytes()`** pour obtenir les octets correspondants.

Question 2. Modifier ce programme afin qu'il insère *à la fin de l'entête* du fichier donné le champ obtenu à la question précédente et sauvegarde le nouvel email ainsi construit dans un nouveau fichier.

Question 3. Vérifier que le fichier obtenu lorsque l'on fournit en entrée le fichier `email1.txt` est identique au fichier `email1-auth.txt` obtenu lors du premier exercice.

Exercice A.3 Vérification du HMAC À la réception d'un tel email, il faut vérifier qu'il est authentique.

Question 1. Écrire un programme `check` qui vérifie si l'email indiqué en paramètre contient bien un champ `X-AUTH` valide et affiche un message d'alerte si ce n'est pas le cas. Ce programme doit parcourir une par une toutes les lignes de l'entête pour y détecter le champ `X-AUTH:` h, puis comparer l'appendice h indiqué au résumé $h' =$ MD5$(c\|S)$ où c désigne le corps de l'email reçu.

Question 2. Modifiez dans le fichier `email1-auth.txt` la valeur du champ `X-AUTH` (ou bien le contenu du corps) puis vérifiez qu'une alerte est bien lancée par le programme `check`.

Exercice A.4 RFC 2104 (facultatif) Dans la vie de tous les jours, l'appendice h n'est pas produit par la formule $h =$ MD5$(c\|S)$ mais d'une manière plus élaborée : suivant la RFC 2104,

$$h = \mathrm{HMAC}_S(c) = f\Big((S \oplus opad) \parallel f\big((S \oplus ipad) \parallel c\big)\Big)$$

où : - f désigne la fonction de hachage utilisée : MD5, SHA1, SHA256, etc.
- S désigne le secret partagé et c le message à authentifier, vus chacun comme une suite d'octets ;
- $\parallel$ désigne à nouveau la concaténation alors que $\oplus$ correspond au XOR bit-à-bit ;
- Les suites d'octets $ipad =$ **`0x36`**...**`0x36`** et $opad =$ **`0x5c`**...**`0x5c`** ont la même taille que S.

Modifiez les programmes `cert` et `check` selon ce nouveau mode de calcul en considérant désormais S comme une suite de 64 octets formés des 16 octets 0xc5, 0xdc, 0xb7, ..., 0x43 suivis de 48 octets nuls.

```c
#include <stdio.h>
#include <stdlib.h>
#include <openssl/md5.h>

int main(void) {
  char *nom_du_fichier = "butokuden.jpg";
  FILE *f = fopen (nom_du_fichier, "rb");
  if (f == NULL) {
    printf ("Le fichier %s ne peut pas être ouvert.\n", nom_du_fichier);
    exit(EXIT_FAILURE);
  }

  MD5_CTX contexte;
  MD5_Init(&contexte);

  unsigned char buffer[1024];
  int nb_octets_lus = fread(buffer, 1, sizeof(buffer), f);// Lecture du 1er morceau
  while (nb_octets_lus != 0) {
    MD5_Update (&contexte, buffer, nb_octets_lus);        // Digestion du morceau
    nb_octets_lus = fread (buffer, 1, sizeof(buffer), f); // Lecture du morceau suivant
  }
  fclose (f);

  unsigned char resume[MD5_DIGEST_LENGTH];
  MD5_Final (resume, &contexte);
  printf("Le résumé MD5 du fichier \"butokuden.jpg\" vaut: 0x");
  for(int i = 0; i < MD5_DIGEST_LENGTH; i++) printf("%02x", resume[i]);
  printf("\n");
  exit(EXIT_SUCCESS);
}
```

FIGURE 3 – Calcul du résumé MD5 d'un fichier en C (paquet `HMAC.zip`)

```java
import java.io.*;
import java.security.*;

public class Resume {
  public static void main(String[] args) throws Exception {
    File fichier = new File("butokuden.jpg");
    FileInputStream fis = new FileInputStream(fichier);

    MessageDigest hacheur = MessageDigest.getInstance("MD5");

    byte[] buffer = new byte[1024];
    int nbOctetsLus = fis.read(buffer);                  // Lecture du 1er morceau
    while (nbOctetsLus != -1) {
      hacheur.update(buffer, 0, nbOctetsLus);            // Digestion du morceau
      nbOctetsLus = fis.read(buffer);                    // Lecture du morceau suivant
    }
    fis.close();

    byte[] résumé = hacheur.digest();
    System.out.print("Le résumé MD5 du fichier \"butokuden.jpg\" vaut: 0x");
    for(byte k: résumé) System.out.printf("%02x", k);
    System.out.println();
  }
}
```

FIGURE 4 – Calcul du résumé MD5 d'un fichier en Java (paquet `HMAC.zip`)

```c
#include <stdio.h>
#include <stdlib.h>
#include <openssl/sha.h>

int main(void) {
  char *nom_du_fichier = "butokuden.jpg";
  FILE *f = fopen (nom_du_fichier, "rb");
  if (f == NULL) {
    printf ("Le fichier %s ne peut pas être ouvert.\n", nom_du_fichier);
    exit(EXIT_FAILURE);
  }

  SHA256_CTX contexte;
  SHA256_Init(&contexte);

  unsigned char buffer[1024];
  int nb_octets_lus = fread(buffer, 1, sizeof(buffer), f);// Lecture du 1er morceau
  while (nb_octets_lus != 0) {
    SHA256_Update (&contexte, buffer, nb_octets_lus);     // Digestion du morceau
    nb_octets_lus = fread (buffer, 1, sizeof(buffer), f); // Lecture du morceau suivant
  }
  fclose (f);

  unsigned char resume[SHA256_DIGEST_LENGTH];
  SHA256_Final (resume, &contexte);
  printf("Le résumé SHA256 du fichier \"butokuden.jpg\" vaut: 0x");
  for(int i = 0; i < SHA256_DIGEST_LENGTH; i++) printf("%02x", resume[i]);
  printf("\n");
  exit(EXIT_SUCCESS);
}
```

FIGURE 5 – Calcul du résumé SHA256 d'un fichier en C (paquet `HMAC.zip`)

```java
import java.io.*;
import java.security.*;

public class Resume {
  public static void main(String[] args) throws Exception {
    File fichier = new File("butokuden.jpg");
    FileInputStream fis = new FileInputStream(fichier);

    MessageDigest hacheur = MessageDigest.getInstance("SHA-256");

    byte[] buffer = new byte[1024];
    int nbOctetsLus = fis.read(buffer);                  // Lecture du 1er morceau
    while (nbOctetsLus != -1) {
      hacheur.update(buffer, 0, nbOctetsLus);            // Digestion du morceau
      nbOctetsLus = fis.read(buffer);                    // Lecture du morceau suivant
    }
    fis.close();

    byte[] résumé = hacheur.digest();
    System.out.print("Le résumé SHA-256 du fichier \"butokuden.jpg\" vaut: 0x");
    for(byte k: résumé) System.out.printf("%02x", k);
    System.out.println();
  }
}
```

FIGURE 6 – Calcul du résumé SHA256 d'un fichier en Java (paquet `HMAC.zip`)