*« 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. L'an dernier encore, 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 $S$.

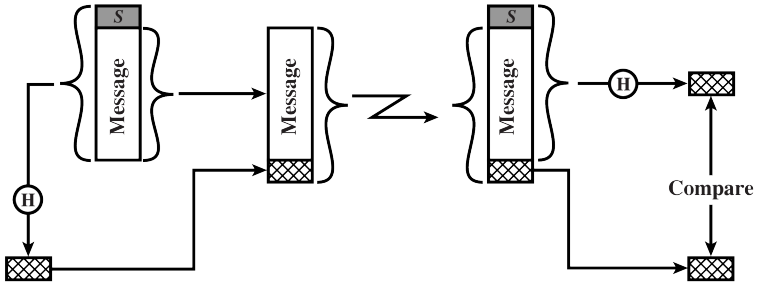


FIGURE 1 – Principe du protocole d'échange d'un message authentifié

Au moment de l'envoi d'un email, un appendice $h$ sera ajouté au courrier. 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 authentique (car il a été produit par une personne connaissant le secret $S$ et le corps $c$) ; 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 « Pratique de la cryptographie » dans le paquet `HMAC.tgz`. 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) suivi du caractère « line feed (LF) » (valeur ASCII 10). Cette fin de ligne en deux caractères est standard 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 2012 21:31:16 +0000
Received: by serveur.fr (Postfix, from userid 106)
  id 3DF2F15A0CD; Fri, 21 Sep 2012 23:31:16 +0200 (CEST)
From: "Thomas Moore" <president@serveur.fr>
To: marcel.pigeon@yahoo.fr
Subject: Transfert d'argent urgent
Date: Fri, 21 Sep 2012 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: <20120921212116.3DF2F16A0CD@serveur.fr>

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

FIGURE 2 – Un exemple d'email (RFC 2822)

Dans cet exercice et les suivants, le secret $S$ est formé par les 128 bits suivant :

$$S = \texttt{0xc5dcb78732e1f3966647655229729843}.$$

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. En utilisant un site Internet spécialisé ou, c'est plus sûr, une commande dans un terminal, calculez le résumé MD5 du *corps* de l'email donné dans le fichier `email1.txt`.
Notez ici le résultat : $h = \text{MD5}(c) =$6F0...

Question 2. Construisez un fichier formé par le corps de cet email et une ligne supplémentaire qui ne contient que le secret $S$ suivi par un CRLF terminal. Puis calculez le résumé MD5 du fichier obtenu.
Notez ici le résultat : $h = \text{MD5}(c\|S) =$E46D...

Question 3. Modifiez le fichier `email1.txt` en ajoutant à la fin de l'entête un nouveau champ, non-standard, formé par `X-UdC_authentique:` suivi du résultat $h$ de la question précédente et d'un CRLF terminal. Enregistrez le fichier obtenu sous le nom `email1-auth.txt`.

**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. Les figures 3 et 4 illustrent comment il est possible de calculer, en C et en Java, le résumé MD5 d'un fichier nommé `butokuden.jpg`. Ces deux programmes et ce fichier JPG sont disponibles dans l'archive `HMAC.tgz`.

Question 1. Écrire un programme `cert` qui calcule et affiche le champ `X-UdC_authentique:` $h$ correspondant au fichier `email1.txt`. Vous devez en particulier programmer l'extraction du corps $c$ du fichier email donné, afin de calculer ensuite le résumé $h = \text{MD5}(c\|S)$.

Question 2. Modifier ce programme afin qu'il insère *à la fin de l'entête* du fichier `email1.txt` ce nouveau champ et sauvegarde le résultat obtenu sous le nom `email1-secure.txt`.

Question 3. Vérifier que les fichiers `email1-auth.txt` et `email1-secure.txt` sont identiques.

**Exercice A.3 Vérification du HMAC** Il s'agit maintenant d'écrire un programme qui vérifie qu'un email est authentique.

Question 1. Écrire un programme `check` qui vérifie si le fichier `email1-secure.txt` contient bien un champ `X-UdC_authentique` 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-UdC_authentique:` $h$, puis extraire l'appendice $h$ et enfin le comparer au résumé $h' = \text{MD5}(c\|S)$ où $c$ désigne le corps de l'email.

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

**Exercice A.4 Calcul d'un HMAC conforme à la RFC 2104** Dans la vie de tous les jours, l'appendice $h$ n'est pas produit par la formule $h = \text{MD5}(c\|S)$ mais de manière un peu plus élaborée. En effet, selon la RFC 2104,

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

où :

- $f$ désigne la fonction de hâchage utilisée : MD5, SHA1, etc.
- $S$ désigne le secret partagé (de la taille des résumés produits par $f$) et $c$ le message à authentifier.
- Les deux suites d'octets $ipad =$ **`0x363636...3636`** et $opad =$ **`0x5c5c5c...5c5c`** sont également de la taille des résumés produits par $f$, et donc de la même taille que $S$.
- $\parallel$ désigne à nouveau la concaténation alors que $\oplus$ correspond au XOR bit-à-bit.

Question 1. Corrigez les programmes `cert` et `check` selon ce nouveau mode de calcul.

Question 2. Organisez votre code afin que le calcul d'un HMAC et celui de l'extraction du corps d'un email n'apparaissent qu'une seule fois dans l'ensemble de votre travail, soit en le rassemblant dans un seul programme, soit en développant une librairie.

```c
#include <stdio.h>
#include <openssl/md5.h>
int main() {
  int i;
  unsigned char resume_md5[MD5_DIGEST_LENGTH];
  unsigned char buffer[1024];
  int nb_octets_lus;
  char *nom_du_fichier="butokuden.jpg";

  FILE *fichier = fopen (nom_du_fichier, "rb");                    // On ouvre le fichier
  if (fichier == NULL) {
    printf ("Le fichier %s ne peut pas être ouvert.\n", nom_du_fichier);
    return 0;
  }
  MD5_CTX contexte;
  MD5_Init (&contexte);                          // On initialise la fonction de hachage
  nb_octets_lus = fread (buffer, 1, sizeof(buffer), fichier);   // Lecture d'un morceau
  while (nb_octets_lus != 0) {
    MD5_Update (&contexte, buffer, nb_octets_lus);               // Digestion du morceau
    nb_octets_lus = fread (buffer, 1, sizeof(buffer), fichier); // Lecture d'un morceau
  }
  fclose (fichier);
  MD5_Final (resume_md5, &contexte);                          // On finalise la digestion
  printf("Le résumé MD5 du fichier \"butokuden.jpg\" vaut: 0x");
  for(i = 0; i < MD5_DIGEST_LENGTH; i++) printf("%02x", resume_md5[i]);
  printf("\n");
  return 0;
}
```

FIGURE 3 – Calcul du résumé MD5 d'un fichier en C

```java
import java.io.*;
import java.security.*;
public class Resume {
  public static void main(String[] args) {
    byte[] buffer, resume;
    MessageDigest fonction_de_hachage;
    File fichier = new File("butokuden.jpg");

    try {
      FileInputStream fis = new FileInputStream(fichier);       // On ouvre le fichier
      fonction_de_hachage = MessageDigest.getInstance("MD5");
      buffer = new byte[1024];
      int nbOctetsLus = fis.read(buffer);                // Lecture du premier morceau
      while (nbOctetsLus != -1) {
        fonction_de_hachage.update(buffer, 0, nbOctetsLus);    // Digestion du morceau
        nbOctetsLus = fis.read(buffer);                  // Lecture du morceau suivant
      }
      fis.close();
      resume = fonction_de_hachage.digest();               // On finalise la digestion
      System.out.print("Le résumé MD5 du fichier \"butokuden.jpg\" vaut: 0x");
      for(byte octet: resume)
        System.out.print(String.format("%02X", octet));
      // On affiche le résumé en hexadécimal
      System.out.println();
    } catch (Exception e) { e.printStackTrace(); }
  }
}
```

FIGURE 4 – Calcul du résumé MD5 d'un fichier en Java

```c
#include <stdio.h>
#include <openssl/md5.h>
int main()
{
  int i;
  unsigned char resume_md5[MD5_DIGEST_LENGTH];
  unsigned char message[] = "Alain Turin";
  printf("Message à hacher: \"%s\" \n", message);

  MD5_CTX contexte;
  MD5_Init (&contexte); // Initialisation de la fonction de hachage
  MD5_Update (&contexte, message, sizeof(message) -1 );
  MD5_Final (resume_md5, &contexte);

  printf("Le résumé MD5 de cette chaîne est: 0x");
  for(i = 0; i < MD5_DIGEST_LENGTH; i++) printf("%02x", resume_md5[i]);
  printf("\n");
}

/*
> ./resumes_chaine
Message à hacher: "Alain Turin"
Le résumé MD5 de cette chaîne est: 0xc5dcb78732e1f3966647655229729843
> echo -n "Alain Turin" | md5sum
c5dcb78732e1f3966647655229729843
>
*/
```

FIGURE 5 – Calcul du résumé MD5 d'une chaîne de caractères en C

```java
import java.io.*;
import java.security.*;
public class ResumeChaine {
  public static void main(String[] args) {
    byte[] buffer, resume;
    MessageDigest fonction_de_hachage;
    String message= "Alain Turin";
    try {
      System.out.println("Message à hacher: \"" + message +"\"");
      buffer = message.getBytes();          // On récupère les octets de la chaîne
      fonction_de_hachage = MessageDigest.getInstance("MD5");     // On charge MD5
      resume = fonction_de_hachage.digest(buffer);         // On calcule le résumé
      System.out.print("Le résumé MD5 de cette chaîne est: 0x");
      for(byte octet: resume)
        System.out.print(String.format("%02X", octet));
      // On affiche le résumé en hexadécimal
      System.out.println();
    } catch (Exception e) { e.printStackTrace(); }
}

/*
> javac ResumeChaine.java
> java ResumeChaine
Message à hacher: "Alain Turin"
Le résumé MD5 de cette chaîne est: 0xC5DCB78732E1F3966647655229729843
>
*/
```

FIGURE 6 – Calcul du résumé MD5 d'une chaîne de caractères en Java