Université de Yaoundé 2005–2006

Examen - Cryptographie & Securité Réseau

Sebastien.Varrette@imag.fr

(3 heures) Documents et calculatrices interdits. Les exercices sont indépendants et peuvent être traités dans n'importe quel ordre. On conseille de ne pas dépasser les indications horaires.

Exercice 1. [Chiffrement de Vigenère et RSA (1 h)]
On considère un système de chiffrement de Vigenère opérant sur l'alphabet $\{A, B, C, \ldots, Z, \_\}$ dont chaque symbole est désigné par un nombre compris entre 0 et 26 :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Etant donné une clé $K = K_1 K_2 \ldots K_t$ et un message clair $M = M_1 M_2 \ldots M_n$, le chiffré $C = C_1 C_2 \ldots C_n$ est donné par :

$$\forall i \in [1, n], \quad C_{i \mod t} = M_{i \mod t} + K_{i \mod t} \mod 27$$

Voici un texte français chiffré que l'agent SuperToto a intercepté pour vous :

```
HWQIO QVPIF TDIHT Y_WAF NGY_F COMVI CGEVZ CVIAF JDFZK
YLYHG YGEHR SHMMX CVHBF AJYKN ZIXHP ZHEQY YJRHT YWMUK
   YKPBY YGEHA G_DY_ YWDTF MHFZK YZYHX CISVI CHIVZ
```

1. Rappeler le principe général de cryptanalyse d'un chiffrement de Vigenère
2. Oscar et Eve travaillent aussi sur la cryptanalyse de ce texte qu'ils ont également intercepté. Oscar envoit à Eve la longueur de la clé qu'il a réussit à déterminer. Pour cela, il utilise la clé publique RSA de Eve $(n, e) = (35, 5)$. SuperToto intercepte ainsi le chiffré RSA de la longueur de la clé : il obtient 10. Quelle est la longueur de la clé ? (on pourra vérifier le résultat en le chiffrant pour s'assurer qu'on obtient bien 10)
3. Eve a réussi à déchiffrer la deuxième et la troisième clé de **chiffrement**. Elle les envoie à Oscar en utilisant la clé publique RSA de ce dernier $(n, e) = (65, 7)$. SuperToto intercepte ainsi le chiffré de K_1 (il obtient 48) et de K_2 (4). Quelles étaient les valeurs de K_1 et de K_2 ?
   Note : on pourra utiliser les résultats suivants :
   $48^2 \equiv 29 \mod 65$ et $48^5 \equiv 3 \mod 65$

_	18,35%
E	14,86%
S	6,97%
A	6,40%
N	6,23%
...	...

Table 1 – Répartition fréquentielle des symboles dans un texte français

4. Maintenant que vous disposez de la longueur de la clé, déchiffrer le texte. Préciser la clé utilisée pour le chiffrement **et** le déchiffrement (sous forme de chaîne de caractères). La répartition (en %) des symboles utilisés dans un texte en français est résumée dans le tableau 1.

Exercice 2. [D.E.S (20 min)]

1. Rappeler la signification des initiales D.E.S et A.E.S. Préciser les tailles de blocs et de clés utilisées pour chacun de ces algorithmes de chiffrement.
2. Rappeler l'algorithme général utilisé dans D.E.S (sans détailler la diversification des clés ou le calcul de la fonction f)
3. Le tableau 2 rappelle les boites-S de D.E.S. On suppose avoir en entrée de ces boîtes-S la chaine hexadécimale `15DA 058BE418`. Quelle est la chaîne hexadécimale de sortie des boîtes S ?

S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

TABLE 2 – Les boîtes S de DES

Exercice 3. [Fonctions de hachage (30 min)]

1. Rappeler la définition d'une fonction de hachage résistante aux collisions. Proposer une construction permettant de réaliser une telle fonction à partir d'une fonction de compression. Citer au moins deux exemples de fonctions de hachage sûres à l'heure actuelle.
2. On considère la fonction de hachage $H : \{0,1\}^t \longrightarrow \{0,1\}^m$. Soit q_k la probabilité de trouver au moins une collision dans H à partir de k messages $\{x_i\}_{1 \leq i \leq k}$ aléatoires. Montrer que

$$\forall 0 < \alpha < 1 : q_k > \alpha \Longrightarrow k > \sqrt{2n \ln\left(\frac{1}{1-\alpha}\right)} = \mathcal{O}(\sqrt{n})$$

   Note : on déterminera la probabilité $p_k = 1 - q_k$ et on admettra que $\forall x \in \mathbb{R}, e^{-x} \geq 1 - x$
   Comment s'appelle cette propriété ? Quelle conséquence peut-on en déduire pour le coût réel d'une attaque par recherche exhaustive (de clé ou de collision) ?

Exercice 4. [Certificats X.509 (15 min)]
Voici un extrait d'un certificat X.509 utilisé pour le transfert de notes depuis le centre d'examen de l'université de Yaoundé :

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=LU, ST=Luxembourg, L=Luxembourg, O=Universite du Luxembourg, \
                OU=Laboratoire LACS, CN=CA LACS/emailAddress=ca@lacs.uni.lu
        Validity
            Not Before: May  9 14:07:41 2005 GMT
            Not After : May  9 14:07:41 2006 GMT
        Subject: C=CM, ST=Centre, L=Yaounde, O=Universite de Yaounde I, OU=Master informatique, \
                CN=Centre d'examen/emailAddress=Emmanuel.Kamgnia@uycdc.uninet.cm
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (512 bit)
                Modulus (512 bit):
                    00:b5:72:3f:39:49:cb:a1:40:ec:33:ca:6f:14:99:
                    f8:94:76:1b:08:9e:9b:50:65:5a:73:cb:f3:f2:6e:
                    8a:62:6c:88:f7:52:6c:c6:32:87:9c:ad:27:4b:ed:
                    0b:5a:35:0f:e9:77:c1:58:54:c5:b0:9c:08:ea:98:
                    fb:22:75:a0:0f
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
[...]
    Signature Algorithm: md5WithRSAEncryption
[...]
```

1. Qui a délivré ce certificat ? Pour qui ?

2. Citer au moins un usage de ce certificat.
3. A l'heure actuelle, quelle taille de clé RSA est considérée comme sûre ?
4. En vous appuyant sur l'expertise que vous avez acquise dans ce cours, et au vu de ce certificat, avez-vous des remarques à formuler à propos de la sécurité du service d'examen de l'université utilisant ce certificat ?

Exercice 5. [Hacking & programmation (10 min + 45 min)]

1. On considère le programme suivant (fichier `test1.c`) :

```
#include <stdio.h>
#include <string.h>
#define MAX_SIZE 256
void f(char * s) {
    char buffer[5];
    strcpy(buffer, s);
    printf("Vous avez ecrit: %s\n",buffer);
}
int main() {
    char s[MAX_SIZE];
    printf("Entrer votre phrase: ");
    scanf("%s",s);
    f(s);
    return 0;
}
```

   (a) Quelle ligne de commande utiliseriez-vous sous Linux pour compiler ce programme avec `gcc` en levant tous les warnings possibles ?

   (b) Quelle manipulation effectuer pour obtenir le message d'erreur "`Segmentation fault`" ? En utilisant un dessin de l'état de la pile, Expliquer pourquoi il est possible d'obtenir un tel comportement.

   (c) Quelle modification apporter pour assurer un comportement cohérent ?

2. Un professeur qui ne connait rien à l'informatique compte utiliser un vieux programme d'un collegue pour calculer la note finale de l'étudiant pour un examen. Ce programme se divise en plusieurs fichiers : `f.c`, `f.h` et `main.c` qui sont fournis ici :

Listing 1 – Le fichier `f.c`

```
#include "f.h"
void f() {
    char tmp[56];
}
```

Listing 2 – Le fichier `f.h`

```
#ifndef __F_H
#define __F_H
void f();
#endif // __F_H
```

Listing 3 – Le fichier `main.c`

```
#include <stdio.h>
#include "f.h"
int main() {
    int i, tmp=0, note = 20;
    printf("*** Calcul de le note finale de l'étudiant ***\n");
    f();
    for (i=1; i<=4; i++) { // Il y a 4 exercices
        tmp = 0;
        printf("  Déduction à appliquer sur l'exercice %d: ",i);
        scanf("%d",&tmp);
        note -= tmp;
    }
    int note_min = 12;
    printf("-> Note finale de l'étudiant : %d - "
           "Moyenne à atteindre: %d\n", ((note < 0)?0:note),note_min);
    return 0;
}
```

(a) Le programme est fourni avec un `Makefile` qui configure l'outil `make`. Expliquer l'intérêt de cet outil et préciser le contenu du fichier `Makefile` permettant l'automatisation de la compilation.

(b) Un étudiant malicieux arrive à se connecter sur l'ordinateur du professeur avec le login `toto` (le login du professeur est `seb`). Les fichiers utilisés se trouvent dans le répertoire `computeNotes`. Voici le résultat de la commande `ls -ld computeNotes/` :

```
drwxr-xr-x   10 seb  seb  340 May  4 09:40 computeNotes/
```

Quelles informations en déduisez vous ? En particulier, l'utilisateur `toto` sera-t-il capable d'accéder aux fichiers contenus dans ce répertoire ? D'en créer des nouveaux ?

(c) Voici maintenant le résultat de la commande `ls -l computeNotes/` :

```
total 104
-rw-r--r--   1 seb  seb   7859 May  3 17:55 Makefile
-rw-r--r--   1 seb  seb   1100 May  3 17:51 README
-rwxr-xr-x   1 seb  seb  26388 May  3 17:58 computeNotes
-rw-rw-rw-   1 seb  seb     46 May  3 17:36 f.c
-rw-r--r--   1 seb  seb     56 May  3 17:35 f.h
-rw-r--r--   1 seb  seb    486 May  3 17:37 main.c
```

Quel fichier l'étudiant est il autorisé à modifier ?

(d) L'étudiant utilise `gdb` pour désassembler l'exécutable `computeNotes`. Voici le résultat du désassemblage de la fonction `f` :

```
(gdb) disassemble f
Dump of assembler code for function f:
0x08048410 <f+0>:       push   %ebp
0x08048411 <f+1>:       mov    %esp,%ebp
0x08048413 <f+3>:       sub    $0x48,%esp
0x08048416 <f+6>:       mov    %ebp,%esp
0x08048418 <f+8>:       pop    %ebp
0x08048419 <f+9>:       ret
```

A quoi servent les deux premières instructions ? Combien d'octets sont en fait réservés en mémoire pour stocker les 56 éléments du tableau `tmp` (déclaré à la ligne 3 de `f.c`) ?

(e) De façon similaire, l'étudiant obtient le désassemblage de la fonction `main` :

```
(gdb) disassemble main
Dump of assembler code for function main:
0x080483f4 <main+0>:    push   %ebp
0x080483f5 <main+1>:    mov    %esp,%ebp
0x080483f7 <main+3>:    sub    $0x28,%esp
0x080483fa <main+6>:    and    $0xfffffff0,%esp
0x080483fd <main+9>:    mov    $0x0,%eax
0x08048402 <main+14>:   sub    %eax,%esp
0x08048404 <main+16>:   movl   $0x0,0xfffffff8(%ebp)
0x0804840b <main+23>:   movl   $0x14,0xfffffff4(%ebp)
0x08048412 <main+30>:   movl   $0x8048600,(%esp)
0x08048419 <main+37>:   call   0x80482ec <printf@plt>
0x0804841e <main+42>:   call   0x80483d4 <f>
0x08048423 <main+47>:   movl   $0x1,0xfffffffc(%ebp)
0x0804842a <main+54>:   cmpl   $0x4,0xfffffffc(%ebp)
0x0804842e <main+58>:   jle    0x8048432 <main+62>
0x08048430 <main+60>:   jmp    0x804846e <main+122>
0x08048432 <main+62>:   movl   $0x0,0xfffffff8(%ebp)
0x08048439 <main+69>:   mov    0xfffffffc(%ebp),%eax
0x0804843c <main+72>:   mov    %eax,0x4(%esp)
0x08048440 <main+76>:   movl   $0x8048640,(%esp)
0x08048447 <main+83>:   call   0x80482ec <printf@plt>
0x0804844c <main+88>:   lea    0xfffffff8(%ebp),%eax
0x0804844f <main+91>:   mov    %eax,0x4(%esp)
0x08048453 <main+95>:   movl   $0x804866c,(%esp)
0x0804845a <main+102>:  call   0x80482cc <scanf@plt>
0x0804845f <main+107>:  mov    0xfffffff8(%ebp),%edx
0x08048462 <main+110>:  lea    0xfffffff4(%ebp),%eax
0x08048465 <main+113>:  sub    %edx,(%eax)
0x08048467 <main+115>:  lea    0xfffffffc(%ebp),%eax
0x0804846a <main+118>:  incl   (%eax)
0x0804846c <main+120>:  jmp    0x804842a <main+54>
0x0804846e <main+122>:  movl   $0xc,0xfffffff0(%ebp)
0x08048475 <main+129>:  mov    0xfffffff0(%ebp),%eax
0x08048478 <main+132>:  mov    %eax,0x8(%esp)
0x0804847c <main+136>:  mov    0xfffffff4(%ebp),%eax
0x0804847f <main+139>:  mov    %eax,0xffffffec(%ebp)
0x08048482 <main+142>:  cmpl   $0x0,0xffffffec(%ebp)
0x08048486 <main+146>:  jns    0x804848f <main+155>
0x08048488 <main+148>:  movl   $0x0,0xffffffec(%ebp)
0x0804848f <main+155>:  mov    0xffffffec(%ebp),%eax
0x08048492 <main+158>:  mov    %eax,0x4(%esp)
0x08048496 <main+162>:  movl   $0x8048680,(%esp)
0x0804849d <main+169>:  call   0x80482ec <printf@plt>
0x080484a2 <main+174>:  mov    $0x0,%eax
0x080484a7 <main+179>:  leave
0x08048490 <main+180>:  ret
```

On voit que l'instruction se trouvant à l'adresse `0x0804841e` correspond à l'appel de la fonction `f` (ligne 6 de `main.c`). Lors de cette instruction, quelle est la valeur de l'adresse de retour empilée sur la pile associée à la fonction `main` ?

(f) En sachant que 12 s'écrit `C` en héxadécimal, quelle adresse mémoire correspond à l'exécution de l'instruction de la ligne 13 de `main.c`. En déduire la distance (en nombre d'octets) séparant l'adresse de retour empilée à l'appel de `f` et celle de l'instruction de ligne 13 de `main.c`.

(g) Proposer la modification que l'étudiant doit opérer dans le fichier `f.c` pour que le programme ignore la boucle `for` à l'éxécution de sorte que la note finale de l'étudiant soit toujours 20.

(h) Quel moyen simple aurait permis au professeur d'éviter cette attaque ?