Universidade da Beira Interior
Faculdade de Engenharia
Departamento de Informática

© Pedro R. M. Inácio (inacio@di.ubi.pt), 2018/19

Segurança Informática
Guia para Aula Laboratorial 3
Licenciatura em Engenharia Informática
Licenciatura em Informática Web
Licenciatura em Tecnologias e Sistemas da Informação

Computer Security
Guide for Laboratory Class 3
Degree in Computer Science and Engineering
Degree in Web Informatics
Degree in Information Technologies and Systems

Sumário
Estudo do modo de operação de uma cifra de chave simétrica contínua através da implementação da função de cifra usando um gerador de números pseudo-aleatórios inseguro. Simulação de um ataque de exploração da maneabilidade das cifras de chave simétricas contínuas.

Summary
Study of the way of functioning of a symmetric-key stream cipher via the implementation of the encryption function using an insecure pseudo-random number generator. Simulation of an attack that explores the maleability of symmetric-key stream ciphers.

Pré-requisitos:
Algumas das tarefas propostas a seguir requerem o uso de *software* para efetuar cálculos, o acesso a um sistema com compilador de programas escritos em linguagem de programação C e que disponibilize a ferramenta OpenSSL. Sugere-se, assim, o uso de uma distribuição comum de Linux, onde todas estas condições estarão provavelmente preenchidas.

1 Implementação de uma Cifra de Chave Simétrica Fraca

Implementation of a Weak Symmetric Key Cipher

Tarefa 1 *Task 1*

O guia laboratorial anterior sugeria a cifra de um ficheiro com uma cifra de chave simétrica contínua. Esta parte do guia propõe, precisamente, a implementação das funções de cifra e decifra deste tipo de cifra, mas recorrendo a um gerador de números pseudo-aleatórios inseguro.

Crie um programa em linguagem C que inicialize o gerador de número pseudo-aleatórios nativo ao C com a semente `123456789`, e imprima no ecrã o resultado das 1000 primeiras chamadas ao gerador. Para ajudar, fica um excerto de código que deverá incluir neste programa.

```
unsigned int ikey = 123456789;
srand(ikey);
int i;
for( i = 0; i < 1000; i++)
  printf("%u\n", (unsigned int) rand());
```

Q1.: Quantos bits têm os números resultantes da execução do programa?
☐ 2. ☐ 4. ☐ 8. ☐ 16. ☐ 31. ☑ 32.

Q2.: A sequência que o seu computador gerou é a mesma que gerou o computador do(a) seu(ua) colega?
☑ Sim, é, porque a sequência depende apenas do valor inicial (*seed*) que, neste caso, é igual nos dois casos.
☐ Não, não é.

Tarefa 2 *Task 2*

No programa que criou antes, implemente uma função que aceite, como parâmetros de entrada, duas *strings* e um valor inteiro, conforme se ilustra a seguir:

```
void encrypt(char * in, char * out, int ikey)
{
  ...
}
```

Para já, esta função deve abrir o ficheiro especificado por `in` para leitura, e copiar o seu conteúdo

para `out`, byte a byte. **Sugestão:** use o excerto de código incluido a seguir:

```
int b;
while (!feof(fIN)) {
  b = fgetc(fIN);
  printf("%x\n",b);
  fputc(b, fOUT);
}
```

No final, compile e teste o procedimento. Para isso, é necessário criar um ficheiro de texto (e.g., `plaintext.txt`). Coloque-lhe um poema de Fernando Pessoa:

> As armas e os barões assinalados,
> Que da ocidental praia Lusitana,
> Por mares nunca de antes navegados,
> Passaram ainda além da Taprobana,
> Em perigos e guerras esforçados,
> Mais do que prometia a força humana,
> E entre gente remota edificaram
> Novo Reino, que tanto sublimaram;
> (...)

Invoque o procedimento na `main()` com:

```
encrypt("plaintext.txt", "ciphertext.txt",
    123456789);
```

e, no final, verifique o que contém o ficheiro `ciphertext.txt`.

Tarefa 3 *Task 3*

Coloque o inicializador do gerador de números pseudo-aleatórios do C (`srand(int iSeed)`) dentro do procedimento de `encrypt` e faça o xor ($\oplus$) do *output* do gerador `rand()` com os bytes que ler do ficheiro de entrada antes de os escrever no ficheiro de saída. **Sugestão:** use a seguinte linha de código para conseguir o xor:

```
b = b ^ rand();
```

Compile e execute novamente o programa. Verifique o que está dentro do ficheiro `cifrado.txt`. **Q3.: Parece-lhe conteúdo cifrado?**

☑ Sim, parece.
☐ Não, consigo ler tudo...

Tarefa 4 *Task 4*

Nas cifras simétricas contínuas, a função de cifra é igual à função de decifra. **Q4.: Dado isto, como é que procederia para decifrar o ficheiro `ciphertext.txt` para o ficheiro `plaintext-2.txt`?**

☐ Tenho ainda de implementar a função de decifra com a operação contrária do xor.
☑ Só tenho de invocar a função da seguinte forma:

---

2 Maneabilidade

Maleability

Considere que queria enviar a mensagem `Enviar 2 euros para o Bob` ao seu banco. Esta mensagem é sensível, porque diz respeito ao seu dinheiro. Para a proteger, decidiu cifrá-la antes de a enviar. **Q5.: Parece-lhe bem?**

☑ Parece-me ótimo. Melhor era impossível.
☐ Hummm, o facto do Prof. estar a perguntar deixa-me desconfiado(a)...

Tarefa 5 *Task 5*

Crie o ficheiro `texto-limpo.txt` e coloque lá a mensagem referida em cima. De seguida, cifre o ficheiro com uma cifra por blocos:

```
$ openssl enc -aes128 -K
0123456789abcdef0123456789abcdef -in
texto-limpo.txt -out cifrado.aes -iv 0
```

Tarefa 6 *Task 6*

Use o programa incluído a seguir para alterar um só byte desse ficheiro (considere analisar o programa com calma):

```
#include<stdio.h>

void alterbyte(char * in, char * out, int
    ibytenumber){

  FILE *fIN, *fOUT;
  fIN  = fopen(in,  "r");
  fOUT = fopen(out, "w");
  if( (fIN == NULL) | (fOUT == NULL) ) {
    fprintf(stderr, "Problem with file\n");
    exit(1);
  }

  int b, i = 1;
  b = fgetc(fIN);
  while (!feof(fIN)) {
    if(i == ibytenumber)
      b = b ^ 0x01;
```

```
    fputc(b, fOUT);
    b = fgetc(fIN);
    i++;
  }
  fclose(fIN);
  fclose(fOUT);
}

int main(int argc, char **argv){
  alterbyte(argv[1],argv[2],atoi(argv[3]));
}
```

Invoque o programa da seguinte forma:

```
$ ./a.out cifrado.aes cifrado-2.aes 8
```

Depois volte a decifrar o ficheiro com:

```
$ openssl enc -d -aes128 -K
0123456789abcdef0123456789abcdef -in
cifrado-2.aes -out texto-limpo-2.txt -iv
0
```

Abra o ficheiro `texto-limpo-2.txt` e verifique o que lá tem dentro. **Q6.: Consegue ler a mensagem original?**

☐ Sim, claro que consigo... afinal, quase não alterei nada.

☑ Não... de maneira nenhuma. Que estranho!?

Q7.: Diria que, se alguém alterasse qualquer byte no ficheiro durante a sua transmissão, o recetor iria notar essa alteração?

☑ Sim, neste caso poder-se-ia dizer isso.

☐ Não, neste caso não se nota nada.

Q8.: Que tipo de cifra é a AES?

☐ Cifra de chave pública.

☐ Cifra de chave simétrica contínua.

☑ Cifra de chave simétrica por blocos.

Tarefa 7 *Task 7*

Apague os ficheiros `cifrado.aes`, `cifrado-2.aes` e `texto-limpo-2.txt` e recomece, desta feita com a cifra RC4, que já foi dito ser uma cifra simétrica continua. Cifre o ficheiro `texto-limpo.txt`:

```
$ openssl enc -rc4 -K 0123456789abcdef -in
texto-limpo.txt -out cifrado.rc4
```

Verifique o conteúdo do ficheiro cifrado com `$ cat cifrado.rc4`.

Tarefa 8 *Task 8*

Use o programa incluído antes para alterar o byte 8 do ficheiro via

```
$ ./a.out cifrado.rc4 cifrado-2.rc4 8
```

e volte a decifrá-lo usando:

```
$ openssl enc -rc4 -K 0123456789abcdef -in
cifrado-2.rc4 -out texto-limpo-2.txt
```

Q9.: Nota algo estranho no ficheiro decifrado?

☑ Sim, noto algo até meio (vá) assustador!

☑ Não... nada.

Q10.: Será que um atacante poderia mudar o texto-limpo sem o conhecer, alterando apenas o criptograma respetivo, e sem ser notado?

☑ Sim, pelos vistos sim.

☐ Não... nunca neste caso.

Q11.: De 0 a 5, em que 0 é muito má e 5 é muito boa, como classifica o facto de uma cifra ser maleável?

☑ 0. ☐ 1. ☐ 2. ☐ 3. ☐ 4. ☐ 5.

Q12.: Usaria uma cifra simétrica contínua para transmitir uma mensagem confidencial sobre um canal que está sujeito a escutas, mas não à alteração das mensagens?

☑ Sim, sem problema. ☐ Não, nem pensar.