NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Arquitetura de Computadores 2023/24

TPC1

Este trabalho de casa consiste num exercício de programação **a ser realizado em grupo de no máximo dois alunos**. Pode esclarecer dúvidas gerais com colegas, mas a solução e a escrita do código devem ser estritamente realizadas pelos membros do grupo. Todas as resoluções serão comparadas de forma automática e os casos de plágio serão punidos de acordo com os regulamentos em vigor. **Caso use ferramentas como CoPilot ou o ChatGPT, deve incluir no código fonte um comentário a relatar esse uso.**

Data/ Hora limite para a entrega : dia 26/3 (3ª feira) às 21:00.
Modo de entrega: crie um ficheiro zip cujo nome é TPC1-XXXXX-YYYYY.zip em que XXXXX é o número do 1º autor e YYYYY é o número do 2º autor. Se o grupo só tiver um elemento o nome é TPC1-XXXXX-00000.zip. O ficheiro zip deve conter apenas dois ficheiros: o script Shell e o código fonte do programa C. O ficheiro zip deve ser **enviado por email** apenas pelo aluno com nº XXXXX para o email do Prof. **Kevin Gallagher (k.gallagher@fct.unl.pt)**

Exercício

A criptografia é atualmente a base fundamental da grande maioria das técnicas utilizadas para proteção de dados. Existem muitos esquemas criptográficos diferentes, desde os mais simples aos mais complexos, sendo os mais complexos os esquemas utilizados em cenários reais. Um dos esquemas de encriptação de dados mais simples é o chamado *one time pad* (OTP). Neste esquema, uma mensagem em claro (*plaintext:* nome dado aos dados originais, não cifrados) é cifrado originando uma mensagem cifrada (*ciphertext:* nome dado aos dados após a sua encriptação) utilizando uma *chave* (*key:* chave secreta utilizada durante a encriptação) que tem exatamente o mesmo comprimento que o *plaintext*, através de uma operação lógica XOR ou *exclusive or*. Suponhamos que o texto em claro tem S bytes e está guardado no vetor *P[0..S-1],* que a mensagem cifrada irá estar no vetor *C[0..S-1],* e a chave está num vetor *K[0..S-1],* será

$$C[i] = P[i] \text{ xor } K[i]$$
$$P[i] = C[i] \text{ xor } K[i]$$

Uma forma de gerar a chave adequada a este trabalho é usar o gerador de números pseudo-aleatórios que existe na biblioteca do C. As principais funções desta biblioteca são:

```
void srand( unsigned int seed)
int rand( )
```

A função *srand( )* recebe como argumento X um nº >= 0 e que funciona como semente do sequência gerada pela função rand( ), isto é, para um mesmo valor de X é gerada a mesma sequência de números . Quando se invoca a função *rand( )* é gerada o próximo elemento da sequência.

O trabalho prático consiste em criar dois programas:

1. Um programa em C que aceita como argumentos um ficheiro de entrada, um ficheiro de saída e um número (que funciona como semente / seed), e encripta o conteúdo do ficheiro de entrada usando o esquema OTP, através da operação XOR e de uma *key* pseudoaleatória gerada a partir da semente;

2. Um script em bash que encontra todos os ficheiros numa determinada diretoria e usa o programa descrito em 1. para os cifrar.

Programa em C

Especificamente, o programa em C receberá os seguintes argumentos:

- O *pathname* do ficheiro de entrada.
- O *pathname* do ficheiro de saída.
- Um inteiro a ser usado na função *srand*.

Utilizando os argumentos anteriormente descritos, o programa em C deve, para cada byte do ficheiro de entrada,

1. Obter o próximo elemento da sequência pseudo-aleatória
2. Fazer o XOR desse elemento com o byte
3. Escrever o byte que resulta de 2. no ficheiro de saída

Se o processo de encriptação for bem-sucedida, imprimir "Encryption of <nome do ficheiro de entrada> succeeded." caso contrário, imprimir "Encryption of <nome do ficheiro de entrada> failed.

Não se esqueça que os argumentos em C são guardados no vetor *argv* que é passado à função main

Script bash

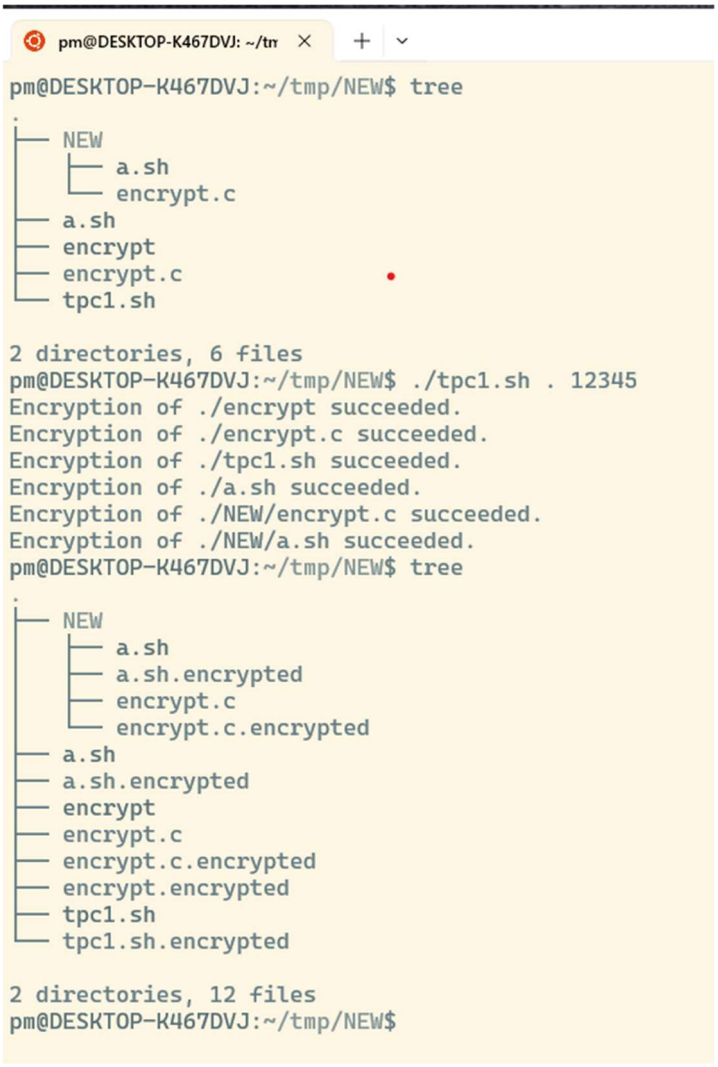
O script bash receberá os seguintes argumentos:

- Uma diretoria.
- Uma *seed*.

Utilizando os argumentos anteriormente descritos, o script *bash* deve:

- Encontrar todos os ficheiros na diretoria de input e suas subdiretorias
- Para cada ficheiro:
    - Ignorar os ficheiros com extensão ".encrypted".
    - Chamar programa C descrito anteriormente com os seguintes argumentos:
        - o nome do ficheiro original
        - o nome do ficheiro da saída que é o resultado de concatenar o nome do ficheiro original com a cadeia “.encrypted”
        - a seed que é o 2º argumento do script

A figura abaixo mostra um exemplo em que o script tpc1.sh invoca o programa *encrypt* na diretoria corrente com a semente 12345.


pm@DESKTOP-K467DVJ: ~/tm
pm@DESKTOP-K467DVJ:~/tmp/NEW$ tree
.
├── NEW
│   ├── a.sh
│   └── encrypt.c
├── a.sh
├── encrypt
├── encrypt.c
└── tpc1.sh
2 directories, 6 files
pm@DESKTOP-K467DVJ:~/tmp/NEW$ ./tpc1.sh . 12345
Encryption of ./encrypt succeeded.
Encryption of ./encrypt.c succeeded.
Encryption of ./tpc1.sh succeeded.
Encryption of ./a.sh succeeded.
Encryption of ./NEW/encrypt.c succeeded.
Encryption of ./NEW/a.sh succeeded.
pm@DESKTOP-K467DVJ:~/tmp/NEW$ tree
.
├── NEW
│   ├── a.sh
│   ├── a.sh.encrypted
│   ├── encrypt.c
│   └── encrypt.c.encrypted
├── a.sh
├── a.sh.encrypted
├── encrypt
├── encrypt.c
├── encrypt.c.encrypted
├── encrypt.encrypted
├── tpc1.sh
└── tpc1.sh.encrypted
2 directories, 12 files
pm@DESKTOP-K467DVJ:~/tmp/NEW$