Trabalho Nº 1 - Encriptação com Chave Simétrica

1. Introdução

Os algoritmos de encriptação de chave simétrica constituíram durante largos anos a base das comunicações seguras.

Como exemplo desse tipo de algoritmo, salienta-se o DES (*Data Encryption System*) que foi largamente utilizado até que o facto de utilizar uma chave de 56 bits levou a que sua decifra fosse efectuada de um simples ataque de força bruta. Foi então substituído pelo AES, *Advanced Encryption Standard*, um algoritmo do mesmo tipo mas utilizando uma chave cujo comprimento pode ir até 256 bits.

Um outro algoritmo intermédio, o TEA (*Tiny Encription Algorithm*) tem características semelhantes ao DES, utiliza uma chave de 128 bits, e tem a grande vantagem de ser implementado de forma simples em algumas linhas de código.

Todos estes algoritmos têm em comum o facto de utilizarem uma mesma chave para encriptação e desencriptação, e de funcionarem em modo bloco (*Block Ciphers*), encriptando blocos de bits de 64 bits (DES e TEA) ou 128 bits (AES).

2. Trabalho a efectuar

Com este trabalho iremos utilizar o algoritmo TEA para realizar uma aplicação capaz de encriptar e desencriptar ficheiros.

Para tal iremos utilizar a implementação em C deste algoritmo que é a seguinte:

```
void encrypt(unsigned int k[], unsigned int text[]) {
  unsigned int y = text[0], z = text[1];
  unsigned int a = k[0], b = k[1], c = k[2], d = k[3];
  unsigned int delta = 0x9e3779b9, sum = 0; int n;

  for (n = 0; n < 32; n++) {
    sum += delta;
    y += ((z << 4) + a) ^ (z+sum) ^ ((z >> 5)+b);
    z += ((y << 4) + c) ^ (y+sum) ^ ((y >> 5)+d);
  }
  text[0] = y;  text[1] = z;
}

void decrypt(unsigned int k[], unsigned int text[]) {
  unsigned int y = text[0], z = text[1];
  unsigned int a = k[0], b = k[1], c = k[2], d = k[3];
  unsigned int delta = 0x9e3779b9;
  unsigned sum = delta << 5; int n;

  for (n = 0; n < 32; n++) {
    z -= ((y << 4) + c) ^ (y+sum) ^ ((y >> 5) + d);
    y -= ((z << 4) + a) ^ (z+sum) ^ ((z >> 5) + b);
    sum -= delta;
  }
  text[0] = y; text[1] = z;
}
```

As duas rotinas recebem como argumentos:

Um ponteiro para um bloco de dados de 128 bits contendo a chave de encriptação, armazenado em quatro inteiros sucessivos:

```
unsigned int k[4];
```

Um ponteiro para um bloco de dados de 64 bits contendo os dados a encriptar ou desencriptar, armazenado em dois inteiros sucessivos:

```
unsigned int text[2];
```

Cada rotina escreve directamente no bloco de dados fornecido em argumento.

O programa a realizar deverá realizar as funcionalidades seguintes:

1. Receber na linha de argumentos o nome do ficheiro a origem e o nome do ficheiro destino, assim como a operação a realizar (*encript*, *decript*)

2. Gerar uma chave de 128 bits para encriptação, que deve ser guardada para permitir a desencriptação do ficheiro (sugestão: utilizar a rotina MD5 da biblioteca **openssl** que gera um *Digest* de 128 bits a partir de uma *string* de caracteres de qualquer tamanho).

3. Ler o ficheiro de origem e dividi-lo em blocos de 64 bits. Se o conteúdo do ficheiro não for múltiplo exacto de 64 bits, o bloco deverá ser preenchido (*padded*) com espaços até perfazer esse valor.

4. Invocar a rotina *encrypt* ou *decrypt* segundo a operação pretendida, dando como argumentos sucessivamente todos os blocos de 64 bits que constituem o ficheiro e a chave gerada.

5. Escrever os dados resultantes (encriptados ou desencriptados) à medida que são calculados no ficheiro de destino.

Para validar o bom funcionamento do programa, deverá encriptar um ficheiro, voltar a desencriptá-lo e compará-lo com o ficheiro original. Os ficheiros só poderão diferir no tamanho (e nunca por mais de 7 bytes), no caso de ter havido *padding* de um bloco.

O programa **`md5.c`**, a cujo código pode aceder em

http://netlab.ulusofona.pt/cr/praticas

apresenta a forma como pode ser utilizada a função de Digest MD5 para gerar uma chave de 128 bits a partir de uma *string* lida da consola.

Se quiser, pode realizar o programa em Java, estando o código das respectivas classes disponível no site acima referido.

3. Prazo de Entrega do Trabalho

O prazo para a entrega dos trabalhos é de duas semanas a contar da aula em que o enunciado foi apresentado. Não serão aceites trabalhos fora do prazo. A entrega deverá ser feita por e-mail num ficheiro zip (ou rar) contendo um relatório em PDF e as listagens dos programas realizados, obedecendo OBRIGATORIAMENTE ao seguinte formato:

`a123456-trabalho-N.zip`

4. Referências

Tiny Encryption Algorithm: www.simonshepherd.supanet.com/tea.htm
(Cuidado com o Código Java apresentado neste site que está errado!)

The Open Source toolkit for SSL/TLS: www.openssl.org

Exemplos de Código C: www.java2s.com/Code/C/CatalogC.htm

Referências das páginas do site do Laboratório: http://netlab.ulusofona.pt/

- *The C Book*