SEGURANÇA EM REDES DE COMUNICAÇÃO

Projecto de Criptografia Aplicada:

Desenvolvimento de um programa de cifra baseado num PRNG

Docente:

Rui Miguel Soares Silva

Discentes:

José Afonso Esteves Janeiro Nº2467
Miguel Bilro Murta Soares Nº2863
Emanuel dos Santos José Nº4121

Ano Lectivo: 2007/2008
1º semestre

Índice

Introdução e Objectivos

Este projecto tem como objectivo o desenvolvimento de um programa que cifre um ficheiro de entrada guardando a sua encriptação num outro ficheiro de texto, assim como a sua desencriptação também num outro ficheiro de texto. Desenvolvido em linguagem C com recurso à biblioteca GMP – GNU Multiple Precision Arithmetic Library.

O programa desenvolvido recorre a um gerador de números pseudo aleatórios (PRNG) validado através da bateria de testes estatísticos Diehard para encriptar e desencriptar ficheiros.

Um gerador de números pseudo aleatórios (PRNG) é um algoritmo que gera uma sequência de números que têm propriedades aproximadas às propriedades dos números aleatórios.

Diehard é uma bateria de testes estatísticos que analisa dados com um grau aprofundado de detalhe.

O ficheiro de entrada será cifrado em blocos de 8, 16, 32, 64, 128, 256 ou 512 bits, conforme desejado pelo utilizador.
De referir que o texto introduzido no ficheiro plaintext.txt não cifra em blocos de 8 bits apenas, mas pensamos que a causa desta situação é o tamanho do ficheiro de texto incluído neste projecto, pois funcionou com outro ficheiro de texto por nós experimentado.
Na introdução de quaisquer outros valores o programa exibe uma mensagem informando que o programa só aceita os valores acima descritos.

Sintaxe Da Linha De Comandos:

O programa é invocado com a seguinte sintaxe:

<NomeDoPrograma.exe> <c|d> <LCPRNG .cfg> <NomeFicheiroSaida> <NomeFicheiroEntrada> <bloco>

Sendo que:

Para encriptar:

<NomeDoPrograma.exe> <c> <LCPRNG .cfg> <FicheiroAEncriptar> <FicheiroPlaintext> <bloco>

Para desencriptar:

<NomeDoPrograma.exe> <d> <LCPRNG .cfg> < FicheiroEncriptado> < FicheiroADesencriptar > <bloco>

NomeDoPrograma.exe :

Programa de cifra executável criado para encriptação e desencriptação de dados.

LCPRNG.cfg :

Gerador de números pseudo aleatórios

FicheiroSaida :

- Na encriptação é o ficheiro de texto que se pretende criar com a mensagem cifrada (Ciphertext)
- Na desencriptação é o ficheiro de texto que se pretende desencriptar (Plaintext)

FicheiroEntrada :

- Na encriptação é o ficheiro de texto com a mensagem original (Plaintext)
- Na desencriptação é o ficheiro de texto com a mensagem encriptada (Ciphertext)

Bloco :

Inteiro múltiplo de oito (8) que corresponde ao valor de bits do bloco parametrizado. Este parâmetro aceita os valores 8, 16, 32, 64, 128, 256 e 512 bits.

Nota: O projecto inclui um ficheiro de texto denominado plainText.txt para experimentar o programa.
Caso não estejam criados, o programa gera automaticamente os ficheiros de dados encriptados e desencriptados.

Testes

Utilizámos o ficheiro dhin.bin de tamanho 12MB constituídos pelas chaves geradas pelo gerador de números pseudo aleatórios PRNG e testadas nos quinze testes da bateria de testes estatísticos Diehard. No entanto pela falta do conhecimento dos números correctos para efectuar estes testes os seus resultados foram insucessos.

Estrutura De Dados

A estrutura de dados utilizada foi o *array* do tipo *char*. Durante a execução do programa, é nos arrays do tamanho que utilizador especifica através do último parâmetro passado pela linha de comandos, que são guardados os blocos de bits do plain text, das chaves e do criptograma.

Plaintext, Chaves e Criptograma

O programa divide o ficheiro de entrada em blocos de 8, 16, 32, 64, 128, 256 e 512 bits através da condição existente no ciclo *while* e coloca estes blocos de bits no *array plainTextBin*. A condição do ciclo while também verifica o final do ficheiro (EOF), para que o número de chaves geradas não ultrapassem o tamanho do ficheiro.

```
while( fread(plainTextBin, sizeof(unsigned int), 1, plainText) != 0 )
```

O programa gera as chaves para a encriptação e desencriptação dos dados através do gerador linear congruencial (LCG) através da expressão $X_{n+1} = (ax_n + c) \bmod m$. O código para efectuar esta expressão é o seguinte:

```
mpz_mul(temp1, Xn, a);
mpz_add(Xn1, temp1, c);
mpz_mod(Xn1, Xn1, m);
mpz_set(Xn, Xn1);
```

Depois a chave é colocada em binário no array *chaveBinAux* através da função

```
mpz_get_str(chaveBinAux, 2, Xn);
```

De seguida a primeira posição do array chaveBin é preenchida com o valor zero através do seguinte ciclo for:

```
for(i = 0; i = ((blockSize - 2) - csize); i++)
{
      chaveBin[i]='0';
}
```

Depois o resto do *array* é preenchido com os valores existentes no array *chaveBinAux* através do seguinte ciclo for:

```
for(; i <= (blockSize - 2); i++)
{
      chaveBin[i] = chaveBinAux[i];
}
```

A última posição do *array chaveBin* é preenchida com o valor null:

```
chaveBin[i] = '\0';
```

Ficando assim o *array chaveBin* com a chave.

Para encriptar o plain text ou para desencriptar o criptograma é feito um XOR bit a bit entre a chave e o plain text, ou entre a chave e o criptograma. Para isso é utilizado o seguinte ciclo for:

O programa encripta ou desencripta através de um XOR entre a chave e o plaintext ou entre o ficheiro encriptado e o ficheiro a desencriptar. Depois o ficheiro encriptado ou desencritado é colocado no *array cripto*. Este processo é feito através do seguinte código:

```
for(i = 0; i <= blockSize - 2; i++)
{
	cripto[i] = chaveBin[i]^plainTextBin[i];
}
```

Por fim o conteúdo do array cripto é passado para o ficheiro *fpout* através da seguinte função:

```
fwrite(&cripto, sizeof(unsigned int), 1, fpout);
```