DCC–IME–USP

Routo Terada
Sala 288A Tel.: 3091 6144
e-mail rt@ime.usp.br

MAC 5723 - 334 - Criptografia

Primeiro Semestre de 2003

Exercício-Programa 1 Data de entrega: até a aula de 28 de abril de 2003.

Observações

- Este exercício é para ser feito individualmente.
- Entregue um disquete com uma etiqueta identificadora contendo número USP, nome completo, data, e nome da disciplina, contendo os seguintes arquivos, eventualmente comprimidos, (e uma cópia backup de todos os arquivos em subdiretório chamado, por exemplo, BACKUP):
  - um arquivo chamado LEIA.ME (em formato .txt) com:
    - os nomes dos arquivos no disquete com uma breve descrição de cada arquivo,
    - uma descrição sucinta de como usar o programa executável, necessariamente na linha-de-comando, i.e., SEM interface gráfica,
    - qual computador (Intel, SUN, ou outro) e qual compilador C (gcc, TURBO-C, ou outro) e qual sistema operacional (LINUX, UNIX, MS-DOS, ou outro) foi usado,
    - instruções de como compilar o(s) arquivo(s) fonte(s), e o nome do arquivo MAKE se for o caso,
    - seu nome completo e número USP
  - os arquivos do programa-fonte necessariamente em linguagem ANSI-C,
  - o programa compilado, i.e., o código executável, (se não incluir, a nota será zero!!!!!)
  - se for o caso, alguns arquivos de entrada e saída usados nos testes: arquivos com os dados de entrada chamados ENT1, ENT2, etc., e arquivos com os dados de saída correspondentes, chamados SAI1, SAI2, etc.
- Coloque comentários em seu programa explicando o que cada etapa do programa significa! Isso será levado em conta na sua nota.
- Faça uma saída clara! Isso será levado em conta na sua nota.
- Não deixe para a última hora. Planeje investir 70 porcento do tempo total de dedicação em escrever o seu programa todo e simular o programa SEM computador (eliminando erros de lógica) ANTES de digitar e compilar no computador. Isso economiza muito tempo e energia.

[2] A nota será diminuida a cada dia de atraso na entrega.

Função K128

Implementar a função criptográ…ca K128, com chave de 128 bits, e com entrada e saída de 128 bits. Você deve deduzir o algoritmo inverso do K128. Na realidade a chave K de 128 bits serve de "semente " para derivar uma matriz pseudo-aleatória chamada SBoxes (como explicado abaixo), mas para este exercício teremos uma matriz …xa para padronizar os testes.

```
Entrada: 128 bits divididos em 4 blocos de 32 bits, E1,E2,D1,D2, nessa ordem,
         8 chaves secundarias de 32 bits em ChavesSecund[]
         e uma matriz SBoxes
Saida: 128 bits criptografados armazenados em E1,E2,D1,D2

E1,E2, D1,D2: int32; // cada variavel com 32 bits
Nrounds: integer; // valor ''default'' ee 16, mas pode ser 24 ou 32
//matriz SBoxes abaixo e' uma chave pre-gerada a partir de K;
//    cada valor entre 0 e (2^32)-1 (i.e., 32 bits)
SBoxes: ARRAY [1 .. Nrounds/8] OF ARRAY [0 .. 255] OF int32;
// 8 chaves auxiliares abaixo sao pre-geradas a partir de K tambem;
//   cada valor entre 0 e (2^32)-1 (32 bits)
ChavesSecund: ARRAY[1 .. 8] OF int32;
// Vetor abaixo e' o numero de posicoes de bits a serem rotacionados
PosicoesRotacao: ARRAY [1 .. 8] := [16,8,16,8,16,24,16,24];
IndLinhaNaSBoxes: integer; //indice de linha na matriz Sboxes

E1 := E1 XOR ChavesSecund[l]; // ou-exclusivo de 32 bits
E2 := E2 XOR ChavesSecund[2];
D1 := D1 XOR ChavesSecund[3];
D2 := D2 XOR ChavesSecund[4];
IndLinhaNaSBoxes := 1;
FOR round := 1 to Nrounds DO // Por definicao 'Nrounds' deve ser multiplo de 8
 Begin
 IndCol:= ''8 bits mais 'a direita de E1''; // int entre 0 e 255
 D1 := D1 XOR SBoxes[IndLinhaNaSBoxes][IndCol];
 E1 := RotacioneParaDireita(E1, PosicoesRotacao[1+(round-1) mod 8] );
 IndCol:= ''8 bits mais 'a direita de E2''; // int entre 0 e 255
 D2 := D2 XOR SBoxes[IndLinhaNaSBoxes][IndCol];
 E2 := RotacioneParaDireita(E2, PosicoesRotacao[1+(round-1) mod 8] );
 Trocar(E1,D2);
 Trocar(D1,E2);
 if (round mod 8 = 0) then IndLinhaNaSBoxes := IndLinhaNaSBoxes + 1;
 End;
E1 := E1 XOR ChavesSecund[5];
E2 := E2 XOR ChavesSecund[6];
```

```
D1 := D1 XOR ChavesSecund[7];
D2 := D2 XOR ChavesSecund[8];
```

1. Cada linha na matriz SBoxes é constituída de 256 elementos de 32 bits; cada elemento é constituído de 4 inteiros de 8 bits. Os 4 inteiros formam 4 colunas; cada uma dessas colunas é uma permutação dos inteiros de 0 a 255 (8 bits) calculada pelo algoritmo no Vol II do livro de Don Knuth, como veremos a seguir.
2. Na realidade a chave K de 128 bits serviria de "semente " para derivar a SBoxes, mas para este exercício teremos uma matriz ...xa pre'-gravada no web-site da disciplina, para padronizar os testes.
3. Neste algoritmo no livro de D. Knuth, descrito a seguir, tem-se no início um vetor SBox inicial, não necessariamente aleatório, chamado SBoxInicial, de 256 inteiros de 32 bits; este vetor é interpretado como uma matriz de 4 colunas e 256 linhas de inteiros de 8 bits chamada Sbox (como foi feito antes com uma linha de SBoxes). Cada coluna de SBox contém uma permutação dos inteiros de 0 a 255
4. Os inteiros na permutação em Sbox são trocados entre si de forma pseudo-aleatória resultando uma outra permutação.

```
FOR IndLinhaNaSboxes:= 1 to (Nrounds/8) DO // para cada linha de 256 colunas
 // A seguir, uma matriz SBoxInicial supostamente gerada por um algoritmo
 // tendo K como ''semente''. Este algoritmo de geracao devera' ser projetada
 // por voce, no caso de aplicacao real
 SBox:= SBoxInicial; // Sbox e' matriz de 4 colunas e 256 linhas de ints de 8 bits
                     // cada coluna e' uma permutacao dos ints de 0 a 255
 FOR IndCol := 0 to 3 DO // note 4 colunas de ints, cada coluna com 1 permutacao
   BEGIN
   FOR i := 0 to 255 DO // 256 linhas com uma permutacao de 8 bits
     BEGIN
     IndAleatorioLin := AleatorioEntre(i,255); // esta funcao retorna um inteiro
                                               // entre i e 255, inclusive
     TroqueInteiro(SBox[i,IndCol], SBox[IndAleatorioLin,IndCol]);
     END; // for i
   END; // for IndCol
SBoxes[IndLinhaNaSBoxes] := SBox; // define outra linha de SBoxes, 256 colunas
END; // for IndLinhaNaSBoxes
```

O seu programa deve perguntar ao usuário, na linha de comando:

1. se é para criptografar ou decriptografar; por exemplo: Digitar C para criptografar e D para decriptografar

2. qual o nome do arquivo de entrada Entra em disco; por exemplo: Digitar o nome do arquivo de entrada a seguir .....

3. qual o nome do arquivo de saída Sai em disco; por exemplo: Digitar o nome do arquivo de saída a seguir .....

4. (Geração da chave K) qual a chave A alfa-numérica, com pelo menos 8 caracteres, sendo A com pelo menos 2 letras e 2 algarismos decimais; se a chave A possuir menos que 16 caracteres (i.e., 16 bytes), a chave K de 128 bits deve ser derivada de A concatenando-se A com ela própria até completar 16 bytes (128 bits)

5. se deve apagar o arquivo Entra (i.e, gravar brancos no lugar de Entra e deletar Entra)

Geração de ChavesSecund[]

1. Os 4 primeiros elementos do vetor ChavesSecund[ ] são constituídos pelos 128 bits da chave K, sendo cada bloco de 32 bits contado da esquerda para a direita de K.

2. Os 4 elementos seguintes de ChavesSecund[ ] são constituídos pelos 128 bits da chave K rotacionados para a direita de 17 posições de bit.

O seu programa deve ler do disco o arquivo Entra, e deve gravar o arquivo Sai correspondente a Entra criptografado/decriptografado com a chave A, no modo CBC (Cipher Block Chaining), com blocos de 128 bits (se necessário, completar com brancos no ...m do último bloco). A seguir deve apagar ou não o arquivo Entra (i.e, gravar brancos no lugar de Entra e deletar Entra).

V. deve testar o programa com pelo menos dois arquivos Entra (que pode ser o seu próprio programa-fonte).

Veri...que se o arquivo decriptografado possui o mesmo comprimento que o arquivo original Entra.

Observação: Modo CBC consiste em encadear um bloco de 128 bits com o código do bloco anterior da maneira vista em aula.

Opcional 1:Medir a aleatoriedade da função da seguinte maneira.

Para j=1, 2, ...,jEntraj, onde Entra é um vetor lido de um arquivo de entrada com pelo menos 512 bits (i.e., pelo menos 4 blocos de 128 bits), fazer o seguinte:

1. alterar apenas na memória só o j-ésimo bit do vetor Entra de cada vez, obtendo em um outro vetor na memória chamado Yj, tal que jEntraj=jYjj; isto é, Entra e Yj só diferem no j-ésimo bit, mas são de igual comprimento.

2. medir a distância de Hamming entre cada bloco de 128 bits de Entra criptografado e o correspondente bloco de 128 bits criptografado de Yj, conforme o algoritmo K128, no modo CBC.

3. acumular estas distâncias de Hamming.

No …nal o programa deve imprimir uma tabela contendo os valores máximos, mínimos e médios das distâncias de Hamming entre cada bloco criptografado de 128 bits correspondente a cada bloco de Entra e Yj, conforme o algoritmo K128, no modo CBC.

Opcional 2: fazer o Opcional 1 uma outra vez, trocando a alteração do j-ésimo bit por alteração simultânea do j-ésimo e do (j+8)-ésimo bits. Isso detetaria uma provável compensação de bits na saída, devido a dois bytes consecutivos alterados na entrada.