DCC–IME–USP

Routo Terada
Sala 204C Tel.: 3091 5918
e-mail rt@ime.usp.br

MAC 5723 - 336 - Criptografia

Primeiro Semestre de 2003

Exercício-Programa 1 Data de entrega no Panda: **27 de outubro de 2004.**

Observações

- Este exercício é para ser feito *individualmente*.
- Entregue no sistema Panda UM ÚNICO arquivo contendo os arquivos seguintes, eventualmente comprimidos:
  - um arquivo chamado LEIA.ME (em formato .txt) com:
    - seu nome completo, e número USP,
    - os nomes dos arquivos inclusos 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).
  - o arquivo MAKE,
  - os arquivos do programa-fonte necessariamente em *linguagem ANSI-C*,
  - o programa *compilado*, i.e.,

    incluir 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.
- A nota será diminuida de um ponto a cada dia "corrido" de atraso na entrega.

Função K128

Implementar a função criptográfica K128, com chave de 128 bits, e com entrada e saída de 128 bits. Você deve **deduzir** o algoritmo inverso do K128.

A Figura 1 a seguir mostra o fluxo geral do algoritmo K128. O número N de iterações (*rounds*) é variável, mas é recomendado $N = 16$.

Algoritmo K128 de N iterações

Entrada: 128 bits de texto legível $X = x_1x_2x_3...x_{128}$ e 128 bits de chave $K = k_1k_2k_3...k_{128}$

Saída: 128 bits de texto ilegível $Y = y_1y_2y_3...y_{128}$

1. Supor calculadas as $N + 8$ subchaves $K_0, K_1,...K_{N+7}$ a partir de K, cada K_j de 32 bits através do algoritmo abaixo.
2. Definir a metade esquerda $E = x_1x_2x_3...x_{64}$ e metade direita $D = x_{65}x_{66}x_{67}...x_{128}$
3. $E_0 \leftarrow E \oplus (K_N|K_{N+1})$ ($\oplus$ é o ou-exclusivo, e | denota concatenação)
4. $D_0 \leftarrow D \oplus (K_{N+2}|K_{N+3})$
5. $D_0 \leftarrow D_0 \oplus E_0$
6. **para** $j \leftarrow 1, 2,...\ N$ **faça** {
   (a) $E_j \leftarrow D_{j-1}; D_j \leftarrow E_{j-1} \oplus f(D_{j-1}, K_{j-1})$ (a função $f()$ especificada abaixo)
   (b) }
7. $E_N \leftarrow E_N \oplus D_N$
8. $D_N \leftarrow D_N \oplus (K_{N+4}|K_{N+5})$
9. $E_N \leftarrow E_N \oplus (K_{N+6}|K_{N+7})$
10. $Y \leftarrow (D_N, E_N)$ (Note que D_N ocupa a metade esquerda)

Definição de $f : \{0,1\}^{64} \times \{0,1\}^{32} \rightarrow \{0,1\}^{64}$

Para $d = 0$ ou 1, e T, U cada um de 16 bits, definir $S_d : \{0,1\}^{16} \times \{0,1\}^{16} \rightarrow \{0,1\}^{16}$ da seguinte forma:

$$S_d(T, U) = ROT2((T + U + d) \bmod 2^{16})$$

onde $ROT2$ denota rotação circular para a esquerda de 2 posições de bit.

Sendo V, W variáveis de 64 e 32 bits respectivamente, definir a função $f(V, W) = U_0U_1U_2U_3 = U$ de 64 bits como sendo o valor calculado por:

1. Denotando $V = V_0|V_1|V_2|V_3$ (cada V_j com 16 bits), e $W = W_0|W_1$ (cada W_j com 16 bits), calcular $t_1 \leftarrow (V_0 \oplus V_1) \oplus W_0$
2. Calcular $t_2 \leftarrow (V_2 \oplus V_3) \oplus W_1$
3. $U_1 \leftarrow S_1(t_1, t_2)$
4. $U_2 \leftarrow S_0(t_2, U_1)$
5. $U_0 \leftarrow S_0(V_0, U_1)$
6. $U_3 \leftarrow S_1(V_3, U_2)$

Algoritmo de geração de subchaves

Entrada: 128 bits de chave $K = k_1k_2k_3...k_{128}$

Saída: $N + 8$ subchaves $K_0, K_1,...K_{N+7}$, cada K_j de 32 bits

1. Seja $U^{-2} \leftarrow 0, U^{-1} \leftarrow k_1k_2...k_{64}, U^0 \leftarrow k_{65}k_{66}...k_{128}$
2. Sendo cada U_j de 16 bits, denote $U = U_0|U_1|U_2|U_3$
3. **para** $j = 1, 2, 3,...(N/2) + 4$ **faça** { /* supondo N par */
   (a) $U \leftarrow f_K(U^{j-2}, U^{j-1} \oplus U^{j-3})$ (a função $f_K()$ especificada abaixo)
   (b) $K_{2j-2} \leftarrow (U_0|U_1); K_{2j-1} \leftarrow (U_2|U_3); U^j \leftarrow U;$
   (c) }

Definição de $f_K : \{0,1\}^{64} \times \{0,1\}^{64} \rightarrow \{0,1\}^{64}$

Sendo V, W variáveis de 64 bits cada, definir a função $f_K(V, W) = U_0U_1U_2U_3 = U$ de 64 bits como sendo o valor calculado por:

1. Denotando $V = V_0|V_1|V_2|V_3$ (cada V_j com 16 bits), e $W = W_0|W_1|W_2|W_3$ (cada W_j com 16 bits), calcular $t_1 \leftarrow (V_0 \oplus V_1)$
2. Calcular $t_2 \leftarrow (V_2 \oplus V_3)$
3. $U_1 \leftarrow S_1(t_1, t_2 \oplus W_0)$
4. $U_2 \leftarrow S_0(t_2, U_1 \oplus W_1)$
5. $U_0 \leftarrow S_0(V_0, U_1 \oplus W_2)$
6. $U_3 \leftarrow S_1(V_3, U_2 \oplus W_3)$

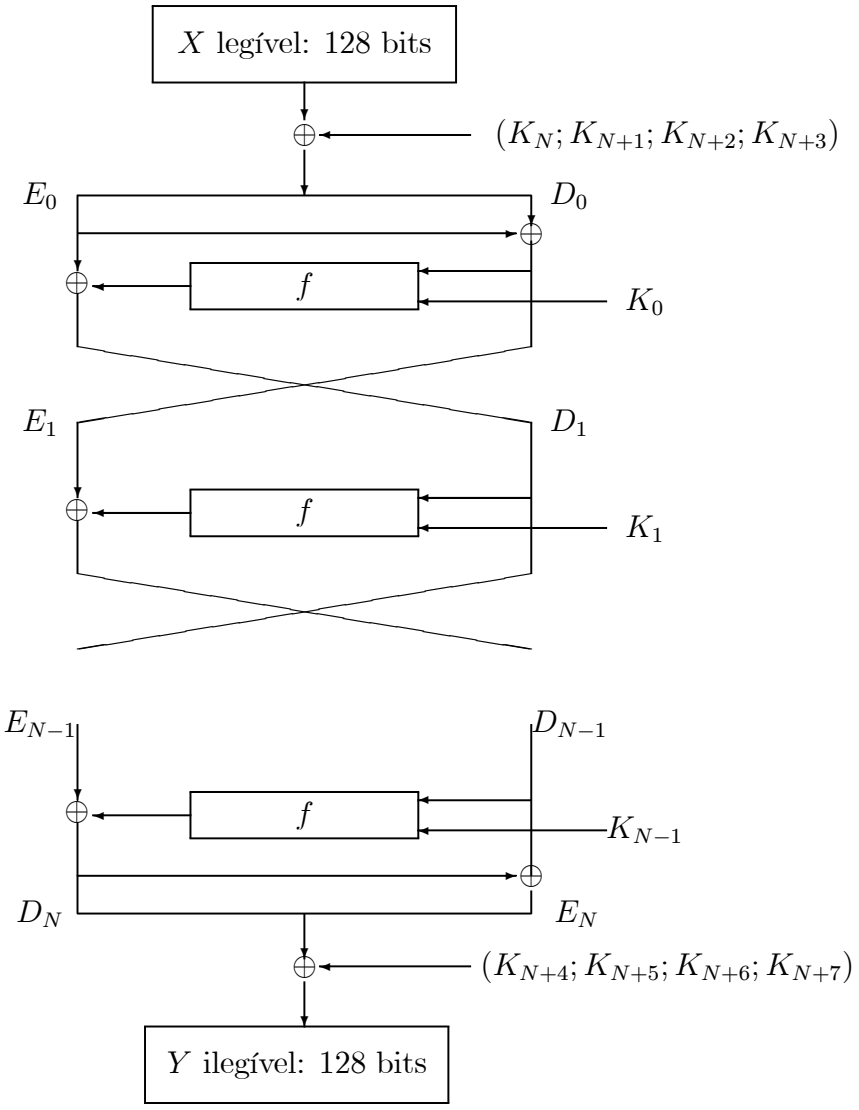

Figura 1: Algoritmo K128

Note que para o cálculo da inversa:

1. $E_N = D_{N-1} \oplus D_N \Rightarrow D_{N-1} = E_N \oplus D_N$
2. Seja $A = f_{K_{N-1}}(D_{N-1})$. Então, $E_{N-1} = D_N \oplus A$, pois, $D_N = E_{N-1} \oplus A$ e $A \oplus A = 0$.

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)

6. se deve efetuar os itens (1) e (2) descritos abaixo

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. A seguir deve apagar ou não o arquivo Entra (i.e, gravar brancos no lugar de Entra e deletar Entra).

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

1. No modo CBC, utilizar bits iguais a UM como Valor Inicial.

2. V. deve testar o programa com pelo menos dois arquivos Entra Por exemplo, o seu próprio programa-fonte. Teste não só com arquivos-texto como com arquivos binários; por exemplo, com algum código executável.

3. Se o último bloco a ser criptografado não possuir comprimento igual a 128 bits, completá-lo com bits iguais a UM.

4. Verifique se o arquivo decriptografado possui o mesmo comprimento que o arquivo original Entra.

O seu programa deve também efetuar os itens seguintes:

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

Para j=1, 2, ...,|VetEntra|, onde VetEntra 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 VetEntra de cada vez, obtendo em um outro vetor na memória chamado Y_j, tal que |VetEntra|=$|Y_j|$; isto é, VetEntra e Y_j só diferem no j-ésimo bit, mas são de igual comprimento. Por exemplo, se cada bloco fosse de 4 bits, VetEntra = 01010011... e Y_2 = 00010011... .

2. medir a distância de Hamming, **separadamente**, entre **cada** bloco de 128 bits de VetEntra criptografado e o correspondente bloco de 128 bits criptografado de Y_j, conforme o algoritmo K128 no modo CBC. Para 4 blocos de 128 bits, tem-se 4 medidas de distância, uma medida para cada bloco.

3. acumular estas medidas de distância de Hamming. Para 4 blocos de 128 bits, tem-se 4 somas cumulativas, sendo que há 128 valores na primeira soma ($j = 1, 2, 3..., 128$), 2*128 valores na segunda soma ($j = 1, 2, 3..., 128, 129..., 2 * 128$), 3*128 valores na terceira soma ($j = 1, 2, 3..., 2 * 128, 2 * 128 + 1..., 3 * 128$), e assim por diante.

No final 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 Y_j, conforme o algoritmo K128, no modo CBC. Para 4 blocos de 128 bits, tem-se 4, o programa deve imprimir 4 valores máximos, 4 mínimos, e 4 médias.

Item 2: Efetuar o Item 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.