Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

DCC003 – Algoritmos e Estruturas de Dados I

Criptografia e Est3g4n0gr4f14

Prof. João Guilherme Maia de Menezes
Monitor: Sérgio Henrique Barbosa Marques

3 de maio de 2018

1 Descrição

A palavra Criptografia deriva do grego, Cryptos, que significa oculto e consiste simplesmente em tornar secreto o significado de uma mensagem. Uma das técnicas mais modernas de criptografia é o RSA. O RSA é considerado um dos algoritmos de encriptação mais seguros, já que mandou por terra diversas tentativas de quebrá-lo. Foi também o primeiro algoritmo a possibilitar criptografia e assinatura digital, e uma das grandes inovações em criptografia de chave pública.

Por outro lado, Esteganografia (do grego "escrita escondida") é o estudo e uso das técnicas para ocultar a existência de uma mensagem dentro de outra, uma forma de segurança por obscurantismo [1]. Em outras palavras, esteganografia é o ramo particular da criptologia que consiste em fazer com que uma forma escrita seja camuflada em outra a fim de mascarar o seu verdadeiro sentido.

É importante frisar a diferença entre criptografia e esteganografia. Enquanto a primeira oculta o significado da mensagem, a segunda oculta a existência da mensagem.

Um exemplo básico de técnica moderna de esteganografia é a alteração do bit menos significativo de cada pixel de uma imagem colorida de forma que ele corresponda a um bit da mensagem. Essa técnica, apesar de não ser ideal, pouco afeta o resultado final de visualização da imagem.

A esteganografia possui algumas aplicações práticas interessantes. Ela é uma das técnicas utilizadas para implementar mecanismos de verificação de direitos autorais em imagens e outras mídias [4]. Além disso, pode ser utilizada para a divulgação de mensagens sem o conhecimento da existência dessas mensagens por parte de outros interessados [2].

O principal objetivo deste trabalho é a **implementação de um programa para codificar e decodificar uma mensagem dentro de uma imagem utilizando técnicas de criptografia e esteganografia**.

2 Sistema de codificação

2.1 RSA

O RSA é provavelmente o algoritmo de cifragem mais usado no mundo. Ele consiste numa cifra de chave pública: qualquer pessoa pode cifrar mensagens usando a chave pública; entretanto, é necessário ter a chave privada para decifrar mensagens, e ter a chave pública não ajuda a obter a chave privada. Criptografia de chave-pública é o que possibilita uma Internet segura. Antes dela, chaves tinham que ser cuidadosamente compartilhadas entre pessoas que queriam se comunicar,

geralmente através de meios não-eletrônicos. Hoje, o RSA é utilizado rotineiramente para se trocar chaves sem permitir que alguém espionando o canal de comunicação entenda o que foi enviado. Acredita-se que o RSA é muito seguro, com base na premissa bem aceita de que não existe um algoritmo que fatore números grandes de forma eficiente; derivar a chave privada a partir da chave pública parece ser tão difícil quanto realizar fatoração.

A seguir, uma descrição dos passos do algoritmo de RSA para encriptação.

2.1.1 Geração da chave

1. Escolha dois números primos distintos p e q;
2. Faça $n = p \times q$;
3. "Quebre"o código da mensagem em blocos menores do que n;
4. Calcule $\phi(n)$, o *totiente* de n. Ele é o número de inteiros positivos menores que n que são primos entre si com n. Para o produto de dois primos p e q o *totiente* é fácil de calcular: $\phi(n) = (p - 1)(q - 1)$. Perceba que para se calcular $\phi(n)$ é necessário que se tenha p e q.
5. Escolha um inteiro e de forma que $1 < e < \phi(n)$ e e e $\phi(n)$ sejam primos entre si. Ou seja, o máximo divisor comum de e e $\phi(n)$ é 1.
6. Calcule a chave d como o inverso multiplicativo de e módulo o *totiente*: $e^{-1} \bmod \phi(n)$. Esse é um valor d tal que $1 \equiv e \bmod \phi(n)$, que pode ser achado com o Algoritmo de Euclides estendido.

A chave pública é o par (n,e) e a chave privada é o par (n,d).

2.1.2 Cifragem

Uma mensagem s é transformada numa mensagem cifrada c através da fórmula: $c = s^e \bmod n$. Observe que isso pode ser feito usando-se apenas os valores públicos n e e.

2.1.3 Decifragem

Uma mensagem cifrada c é decifrada para o text simples s através da seguinte fórmula: $s = c^d \bmod n$.

Onde:

- s = Mensagem a ser encriptada
- c = Mensagem encriptada
- e = Expoente de cifragem
- d = Expoente de decifragem
- n = Módulo formado pelos dois primos p e q
- $\phi(n)$ Função totiente de Euler

A	E	D	S
65	69	68	83

Tabela 1: Tabela com valores ASCII para a palavra "AEDS"

Observe que não é possível realizar esse cálculo apenas com a chave pública.

Para ilustrar de forma mais direta o algoritmo, considere a seguinte mensagem a ser encriptada:

`AEDS`

Utilizando a tabela ASCII, temos os seguintes valores abaixo:

Assim sendo, a mensagem a ser encriptada fica sendo na seguinte forma: 65696883. Vamos tomar $p = 29$ e $q = 31$, assim, $n = 29 \times 31 = 899$.
Calculando o totiente:

$\phi(n) = (29 - 1) \times (31 - 1) = 840$

Portanto, dividindo a mensagem em blocos de até 840, temos os seguintes blocos: $656, 96, 88, 3$

Usaremos o co-primo e como sendo 11, ele é o primeiro co-primo com 840.

Encriptando o primeiro bloco, aplicamos a função de encriptação:

$$c = m^e \ (mod\ n) \rightarrow 656^{11} mod 899 = 831$$

Fazendo para os outros blocos, recebemos o resultado da encriptação sendo: $831, 602, 378, 44$, juntando todos os blocos, a mensagem final é 83160237844.

Desencriptando a mensagem, precisamos encontrar o inverso modular de e com $\phi(n)$, assim:

$$d = e^{-1} \ (mod\ \phi(n)) \rightarrow 11^{-1}\ mod(840) = 611$$

Logo, para desencriptar nossa mensagem, fazemos as operações acima:

$$m = c^d \ (mod\ n) \rightarrow 831^{611}\ mod\ 899 = 656$$

E assim em diante, até recuperar a mensagem original.

2.1.4 Por que isso funciona?

Se cifrarmos e decifrarmos um texto simples s, obtemos uma nova mensagem $s' = (s^e \bmod n)^d \bmod n$. Para que a criptografia funcione, é necessário que $s = s'$. Não é tão difícil observar que isso é verdadeiro se nos basearmos num resultado famoso da Teoria dos Números.

Pelas propriedades da aritmética modular, podemos tirar mod n para fora: $s' = (s^e)^d \bmod n = s^{ed} \bmod n$. Recorde que e e d são escolhidos como inversos multiplicativos módulo $\phi(n)$, então $s^{ed} = s^{1+k\phi(n)} = \text{s}.s^{k\phi(n)}$ para algum inteiro k. Acontece que a sequência $s^0, s^1, s^2, s^3, ...$, tomada módulo n, sempre se repete com período $\phi(n)$. Assim, $s^{\phi(n)} \equiv s^0 \equiv 1 \bmod n$, então $\text{s}.s^{k\phi(n)} \equiv s.(s^{\phi(n)})^k \equiv s.1^k \equiv s \bmod n$, como desejado.

Por exemplo, $\phi(10) = \phi(2.5) = 1.4 = 4$ e $3^4 = 81 \equiv 1 \equiv 2401 = 7^4 \bmod 10$. Podemos então usar 3 e 7 como e e d, já que 3. $7 = 21 \equiv 1 \bmod \phi(10)$. Aplicando isso à mensagem 8, nós a cifrамos como $8^3 = 512 \equiv 2 \bmod 10$. Fazendo a operação inversa, $2^7 = 128 \equiv 8 \bmod 10$. Funciona!

2.2 Esteganografia

Considere o valor dos caracteres em binário da palavra "Wikipedia": `W` (01110111), `i` (01101001), `k` (01101011), `i` (01101001), `p` (01110000), `e` (01100101), `d` (01100100), `i` (01101001), `a` (01100001). Na forma apresentada, a palavra "Wikipedia"é representada utilizando 72 bits. Neste trabalho, iremos utilizar 3 pixels da imagem para codificar cada caracter. Logo para codificar a palavra "Wikipedia"(que possui tamanho 9), precisamos de 27 pixels no total.

Cada pixel em uma imagem colorida possui 3 valores de intensidade, um para cada canal de cor separadamente: Red, Green, Blue (RGB). Assim um pixel com valores 255 0 0, representa a mistura das intensidades de cor nos canais Red=255, Green=0 e Blue=0.

Para codificar os caracteres de uma frase, iremos utilizar o bit menos significativo de cada canal de cor dos 3 primeiros pixels (como cada pixel possui 3 canais de cores, teremos 9 bits no total, onde o último bit, referente ao canal Blue do terceiro pixel, não será utilizado). A codificação é feita na ordem dos canais, assim a letra 'W'(0111 0111) por exemplo será codificada bit a bit onde o primeiro pixel recebe os três primeiros bits (**011**1 0111), o segundo pixel recebe três bits a partir do quarto (011**1 01**11) e o terceiro pixel recebe os últimos dois bits (0111 01**11**).

Exemplo: para armazenar a letra 'W'(0111 0111), precisamos codificar cada bit de W separadamente. Considerando o seguintes valores do primeiro pixel da imagem: 254 20 37, onde Red = 254(1111 1110) Green = 20(0001 0100) Blue = 37(0010 0101). Temos que o primeiro bit do caracter 'W' é 0 (**0**111 0111), se o bit menos significativo do primeiro pixel (Red = 1111 111**0**), for 0, o valor é mantido, caso contrário é trocado para 0.

Caso o bit a ser armazenado tenha valor 1, por exemplo o segundo bit do caractere 'W'(0**1**11 0111), a mesma regra é usada (porém agora analisando o segundo canal de cor: Green= 0001 010**0**). Como o bit menos significativo do segundo canal é 0, seu valor deverá ser invertido para coincidir com o segundo bit do caractere 'W' a ser representado.

Assim, de uma forma geral: é comparado o bit a ser codificado com o bit menos significativo de um dos canais de cores de um pixel, caso forem iguais, o valor é mantido, caso contrário é trocado. Note que o bit menos significativo é o que indica a paridade de um número, ou seja, valores pares são terminados em '0' e valores ímpares terminam em '1'.

Exemplo dos primeiros três bits de W codificados em um pixel:

Pixel original: 254 20 37

Primeiro bit 'W'(0111 0111), Red = 254(1111 1110) Mantém
Segundo bit 'W'(0111 0111) Green = 20(0001 0100) Troca
Terceiro bit 'W'(0111 0111) Blue = 37(0010 0101). Mantém

Pixel codificado: 254 21 37

3 Sistema de decodificação

Para extrair a informação da imagem basta fazer o processo inverso. Leia o valor de cor de cada pixel e armazene **apenas** o bit **menos significativo**. Para cada caractere a ser decodificado, é necessário extrair os primeiros 8 valores dos canais de cor, montando assim a sua forma em 8 bits.

O sistema de decodificação irá parar quando encontrar o caractere que delimita o final da mensagem.

Exemplo de decodificação do caracter 'W' (011 101 11):

Pixels codificados: 254 21 37 , 21 30 237 , 71 55 20*

Tabela 2: Exemplo de Decodificação

Primeiro bit	254	=	1111 1110	W[0] =	0
Segundo bit	21	=	0001 0101	W[1] =	1
Terceiro bit	37	=	0010 0101	W[2] =	1
Quarto bit	21	=	0001 0101	W[3] =	1
Quinto bit	30	=	0001 1110	W[4] =	0
Sexto bit	237	=	1110 1101	W[5] =	1
Setimo bit	71	=	0100 0111	W[6] =	1
Oitavo	55	=	0011 0111	W[7] =	1
W = 0111 0111					

*O ultimo valor não será utilizado

4 Formato da entrada e da saída

Neste trabalho prático, você deverá implementar dois programas: Codificador e Decodificador.

4.1 Codificador

O codificador irá receber cinco argumentos como **entrada**, **nesta ordem**:

1. Uma imagem no formato .ppm (ASCII,P3);
2. A mensagem que será codificada;
3. O nome da imagem de saída que contém a mensagem escondida;
4. O valor do primo p;
5. O valor do segundo primo q.

Exemplo de entrada:

```
./codificador image.ppm "Minha caneta sumiu." image_encoded.ppm 7 5
```

Onde: (1) image.ppm (2) "Minha caneta sumiu."(3) image_encoded.ppm (4) 7 (5) 5

As **saídas do codificador** serão uma imagem de mesmo tamanho, visivelmente similar a imagem original, com o nome fornecido no terceiro argumento (no exemplo: 'image_encoded.ppm'), e com a mensagem de entrada, após encriptada usando RSA, inserida dentro dela. A segunda saída consiste de um arquivo de texto chamado `'private.txt'`, que vai conter a chave privada (o par (n, d)), com o valor de n na primeira linha, e d na segunda.

IMPORTANTE: Após inserir a mensagem encriptada, coloque um delimitador de fim de mensagem. Como a RSA só retorna números, pode ser qualquer caractere não numérico (sugestão: '.' ou '*').

4.2 Decodificador

O decodificador irá receber como **entrada, nessa ordem**:

1. A imagem ja codificada anteriormente no formato .ppm (ASCII,P3)
2. o caractere delimitador de fim da mensagem (nesse exemplo, o ponto final '.')
3. O arquivo com a chave privada, para poder desencriptar a mensagem.

Exemplo de entrada:

```
./decodificador image_encoded.ppm '.' private.txt
```

Onde: (1) image_encoded.ppm (2) '.' (3) private.txt

A saída do decodificador será a frase codificada anteriormente, impressa na tela no seguinte formato: Exemplo de saida:

```
Message decoded:
Minha caneta sumiu.
```

5 Formato de imagem .ppm

Nesse trabalho prático, as imagens de entrada e saída deverão estar no formato *.ppm* (ASCII, P3) com representação de 8 bits por canal de cor (valores podem ir de 0 a 255). A imagem codificada resultante deverá então ter apenas pixels entre 0 e 255. Mais detalhes sobre esse formato de representação de imagens podem ser encontrados em [3].

6 Entrega do código

O TP será dividido em duas entregas:

- A primeira entrega deve ser feita com o módulo codificador e decodificador apenas criptografando e descriptografando a mensagem usando RSA. Uma pequena modificação deve ser feita no programa codificador, em vez de retornar uma imagem, o programa deve imprimir na tela, o valor da mensagem criptografada.
- A segunda entrega consiste em adicionar a parte de esteganografia aos módulos codificadores e decodificadores. Adaptando a saída do módulo codificador para não imprimir nada na tela, e em vez disso, retornar apenas a imagem resultado com a mensagem dentro dela.

O código fonte e a documentação devem ser entregues em um arquivo compactado no **Moodle**. Submissões onde os programas não seguem as especificações de parâmetros ou não compilarem não serão corrigidos. O trabalho pode ser feito em qualquer editor (Code::Blocks, Visual C++, TextPad, etc).

7 Documentação

A documentação deve ser entregue com o Zip junto ao código. Trabalhos que forem entregues no **Moodle** mas sem a documentação não serão corrigidos. O texto da documentação deve ser breve, de forma que o corretor possa entender o que foi feito no código sem ter que entender linha a linha dos arquivos. A documentação deve conter os seguintes itens:

- Sumário do problema a ser tratado.
- Uma descrição sucinta do algoritmo e procedimentos utilizados.
- Decisões de implementação que porventura estejam omissos na especificação.

8 Avaliação

Sua solução será avaliada de acordo com a corretude das saídas geradas e também com o estilo do código. Um programa correto compila sem erros e advertências, e executa de acordo com os requerimentos elencados neste documento. Um programa com bom estilo é claro, conciso e de fácil leitura. Utilize nomes de variáveis fáceis de serem relacionados e identação apropriada. Seu código deve conter comentários que sejam necessários para explicar como o seu programa funciona, mas sem explicar coisas que sejam óbvias.

Referências

[1] Esteganografia. https://pt.wikipedia.org/wiki/Esteganografia.

[2] Esteganografia (a arte de escrever mensagens ocultas). http://www.newtoncbraga.com.br/index.php/eletronica/artigos-diversos/7049-esteganografia-a-arte-de-escrever-mensagens-ocultas-art1175.

[3] Ppm format specification - netpbm. http://netpbm.sourceforge.net/doc/ppm.html.

[4] Eduardo Pagani Julio, Wagner Gaspar Brazil, and Célio Vinicius Neves. Esteganografia e suas aplicações.