UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ELETRÔNICA E TELECOMUNICAÇÕES

ESTUDO DE MÉTODOS DE CRIPTOGRAFIA DE DADOS UTILIZANDO IMAGENS

Mikhail Yasha Ramalho Gadelha

MANAUS
2010

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ELETRÔNICA E TELECOMUNICAÇÕES

ESTUDO DE MÉTODOS DE CRIPTOGRAFIA DE DADOS UTILIZANDO IMAGENS

Trabalho apresentado como requisito obrigatório para a obtenção do título de Engenheiro da Computação.

Mikhail Yasha Ramalho Gadelha

Orientadora: Profa. Dr. Marly Guimarães Fernandes Costa

MANAUS
2010

DEDICATÓRIA

Dedico essa obra à minha família, em especial minha mãe, Isley Maria da Conceição Ramalho Gomes, por estar sempre do meu lado, nos momentos de vitória e derrota, e cujo amor é incondicional. A meu pai, Aldamir Gadelha, por ter me acompanhado nessa jornada e à minha esposa, Loma Brito do Nascimento, por sua paciência, compreensão, ajuda e pelo seu amor.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me ajudar a conseguir todas as minha vitórias.

A meus amigos da Ufam, Jeferson Bentes, Evandro Teixeira, Nathan Almeida, Marcos Correia, Heitor Carlos, Wellyghan Júnior, Diego Aquino, Fredyson Costa, Jhony Braga, Rogério Ramos, Northon Farias, Lucas Marques,Victor Loyola, pela amizade e por agüentar junto comigo noites em claro pra terminar os trabalhos a tempo.

A meus amigos da Fucapi, Matheus Cavalcante, Alexsandro Sorato, Almino Júnior, Genival Caldeira, Marcus Brandt, Guilherme Juqueira, João Evangelista, Ronaldo Araújo, por sua amizade e companheirismo nesses 5 anos de faculdade.

Aos meus amigos brasileiros que conheci na Hungria, Maurício Andreis, Guilherme Becker, Victor Nicolau, Gabriel Brocki, Danillo Guilhon e Samir Salles, pela sua amizade e ajuda nos dias fora do Brasil.

Aos meus colegas e integrantes do projeto ZAGAIA, que contribuíram direta ou indiretamente para a realização desta obra.

Ao Prof. Kenny Vinente que nos últimos 2 anos vem me ajudado na formação acadêmica.

À minha orientadora, Prof.ª Marly Guimarães, pela paciência e suporte no desenvolvimento dessa monografia.

“The system must be practically,
if not mathematically, indecipherable”

Dr. Auguste Kerckhoffs

RESUMO

Em um mundo altamente conectado através dos mais diversos meios de comunicação, como o celular e a internet é extremamente relevante que as informações sigilosas, ao trafegarem por redes públicas, cheguem aos seus destinatários sem que sejam roubadas ou modificadas. Trata-se o presente projeto da proposição de um novo método de encriptação de dados, utilizando imagens, que se aproveita da distribuição aleatória dos valores de intensidade de pixels da imagem. Para validar o método, foram realizados testes comparativos com os principais algoritmos de encriptação utilizados atualmente, AES e RSA. Para tal, fez-se uso de mensagens aleatórias e textos na língua Portuguesa. O método proposto mostrou-se mais eficiente na encriptação da mensagem, uma vez que gera um arquivo encriptado diferente a cada vez que é utilizado, o que não acontece com o AES e o RSA, que sempre geram o mesmo arquivo encriptado. Por outro lado, mostrou-se mais lento que o AES e o RSA.

Palavras-chave: criptografia, imagens, AES, RSA.

ABSTRACT

In a highly connected world through several means of communication such as cell phones and internet, is extremely important that sensitive information, trafficking on public networks, reach their recipients without being stolen or modified. In this project is proposed a new method of data encryption, using images, which takes advantage of the random distribution of intensity values of image's pixels. To validate the method, comparative tests were made with the main encryption algorithms currently used, AES and RSA. To this end, it was made use of random messages and texts in Portuguese. The proposed method was more efficient to encrypt the message, since it generates a different file encrypted each time it is used, which does not happen with AES and RSA, which always generate the same encrypted file. On the other hand, was slower than AES and RSA.

Key-words: cryptography, images, AES, RSA.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE SIGLAS E SÍMBOLOS

AES	*Advanced Encryption Standard*
ASCII	*American Standard Code for Information Interchange*
CCD	*charge-coupled device*
COPACOBANA	*Cost-Optimized Parallel Code Breaker*
DES	*Data Encryption Standard*
EFF	*Electronic Frontier Foundation*
FIPS	*Federal Information Processing Standard*
IBM	*International Business Machines*
IEC	*International Electrotechnical Commission*
ISO	*International Organization for Standardization*
NIST	*National Institute of Standards and Technology*
RSA	*Ravest Shamir Adleman*
TDEA	*Triple Data Encryption Algorithm*

SUMÁRIO

1 INTRODUÇÃO

No decorrer da história da humanidade, o homem sempre precisou se comunicar de forma segura, seja mandando mensagens confidenciais para aliados durante a guerra através de redes de comunicações não seguras, como rádio, ou mais recentemente enviando relatórios confidenciais por email para um superior na empresa.

Existe uma necessidade constante de segurança das informações que trafegam no mundo atualmente, milhares de operações sigilosas precisam ocorrer de forma segura e a cada dia são desenvolvidas técnicas para escutar e modificar tais informações.

A criptografia surgiu para tentar evitar que informações sigilosas sejam recuperadas e alteradas por intrusos. A criptografia de informações abrange não somente a criptografia de arquivos, mas também a criptografia de canais de comunicação, quando existem informações importantes trafegando por canais pouco seguros como a *internet*.

Existem duas classes diferentes de métodos de criptografia: - o método de chave simétrica, utilizado desde o tempo da Roma antiga, que utiliza uma mesma chave para encriptar e decriptar uma mensagem (apesar do mesmo conceito, o algoritmo utilizado na Roma antiga é, nos dias de hoje, trivialmente quebrável) e ; – o método proposto em 1976 por pesquisadores do MIT que utiliza duas chaves diferentes, uma para a encriptação e outra para decriptação, chamado método de chave assimétrica.

No presente projeto será investigada a utilização de imagens como chave de criptografia de dados, sendo verificadas quais as implicações e os resultados dessa técnica. A principal motivação para a criação do algoritmo é utilizar a natureza aleatória que pode haver em uma imagem, utilizando-se desta para encriptar uma mensagem.

A técnica proposta é do tipo criptografia de chave simétrica, ou seja, a mesma chave (no caso, a imagem) é utilizada tanto na encriptação quanto na decriptação. Porém, a técnica será avaliada e os seus resultados comparados com as principais técnicas utilizadas atualmente para encriptação de informações, tanto de criptografia de chave simétrica quanto de chave assimétrica.

1.1 Objetivos do Trabalho

1.1.1 Objetivo Geral

Desenvolver uma proposta de utilização de imagens como chave de criptografia de dados.

1.1.2 Objetivos Específicos

1. Efetuar um levantamento bibliográfico sobre criptografia de dados.
2. Identificar métodos utilizados para avaliação de desempenho de técnicas de criptografia.
3. Investigar o potencial de utilização de imagens como chave de criptografia de dados.
4. Comparar o método proposto com os identificados no levantamento bibliográfico.

1.2 Metodologia

O desenvolvimento deste trabalho compreende as seguintes etapas:

- Pesquisa Bibliográfica: Nesta etapa serão estudados os conceitos fundamentais relacionados ao método sugerido;
- Desenvolvimento do algoritmo: Implementar algoritmos para criptografia de dados utilizando imagens.
- Testes de algoritmo: Realizar testes comparativos entre a implementação e os principais algoritmos de criptografia utilizados atualmente.
- Validação do algoritmo: Avaliar os resultados dos algoritmos e identificar os possíveis erros.

1.3 Estrutura do Trabalho

Este trabalho esta dividido em 6 capítulos, descritos a seguir:

O capítulo 1 apresenta a introdução, descrição do cenário atual, motivação da proposta do algoritmo assim como os objetivos e a metodologia empregada no desenvolvimento.

O capítulo 2 apresenta a forma como são representados textos e imagens em um computador, assim como o processo de aquisição e transformação de uma imagem analógica em digital.

O capítulo 3 apresenta um estudo sobre criptografia, seus princípios e objetivos, além de uma descrição dos modelos de sistemas criptográficos simétricos e assimétricos e seus principais algoritmos.

O capítulo 4 apresenta o método proposto de criptografia de chave simétrica utilizando imagens, da sua concepção à forma mais avançada.

O capítulo 5 apresenta os resultados obtidos e a comparação entre os principais algoritmos de criptografia e o método proposto.

O capítulo 6 apresenta as considerações finais do projeto assim como idéias para trabalhos futuros e melhorias no algoritmo proposto.

2 REPRESENTAÇÃO DE IMAGENS E TEXTOS EM UM COMPUTADOR

Um computador interpreta dados que são armazenados por representação numérica (uma seqüência de 1's e 0's). Nessa representação, a presença de um dado (ou informação) pode ser associada ao valor 1 e a ausência ao valor 0 [1,2].

Nesse item, discorre-se sobre as formas de representação de dados (texto e imagem).

2.1 Representação de Texto

Em um esforço para padronizar a representação de caracteres, a maioria dos fabricantes de computadores desenvolveram suas máquinas para utilizar um padrão de codificação. O mais utilizado atualmente é o padrão ASCII. ASCII significa *American Standard Code for Information Interchange* (Código Padrão Americano para o Intercâmbio de Informação). [3]

Na tabela ASCII, um valor numérico pequeno representa um valor, caractere ou pontuação ortográfica. Inicialmente a tabela ASCII possuía 128 valores, dos quais 33 eram caracteres não imprimíveis, ou seja, não eram letras, números ou pontuações, mas valores de controle, como o ASCII-10 (*Line-Feed*) utilizado para mover o cursor para linha seguinte sem mudar a coluna onde se encontra o cursor, o ASCII-11 (*Carriage Return*) utilizado para mover o cursor para a coluna zero enquanto continua na mesma linha, entre outros.

A tabela ASCII original podia representar somente o alfabeto inglês. Caracteres como cedilha (ç), letras com acentuação gráfica, entre outros, não podiam ser representados. Apartir de 1987, a norma ISO/IEC 8859, dividida em 12 partes, foi criada para suprir essa necessidade. A mais utilizada é a ISO/IEC 8859-1: *Part 1: Latin alphabet No. 1*, pois possui caracteres suficientes para representar, além do idioma inglês, a maioria dos idiomas latinos e do leste europeu. A tabela ASCII original associada à ISO 8859-1 possui valores entre 0 e 256 para representação de caracteres. O anexo A contém a tabela ASCII segundo a ISO/IEC 8859-1.

2.2 Representação de Imagens

Em um computador, as imagens digitais também são representadas utilizando números. Na verdade, uma imagem é representada por uma ou mais matrizes de números reais. Podemos definir a imagem como uma função bidimensional, $f(x, y)$, onde x e y são coordenadas espaciais, e a amplitude de f em qualquer par de coordenadas (x, y) é chamada intensidade ou nível de cinza naquele ponto. Quando x, y e a amplitude de f são valores finitos, quantidades discretas, a imagem é chamada de imagem digital. [4]

$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \cdots & f(0, N-1) \\ f(1, 0) & f(1, 1) & \cdots & f(1, N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1, 0) & f(M-1, 1) & \cdots & f(M-1, N-1) \end{bmatrix}.$$

Figura 1: Representação matricial de uma imagem.
Fonte: [4]

As imagens digitais são compostas por um número finito de elementos, e cada elemento possui um valor e coordenada na imagem digital. Esses elementos são chamados *picture elements* ou *pixels*. Assim, uma imagem digital é formada por uma matriz de MxN *pixels*.

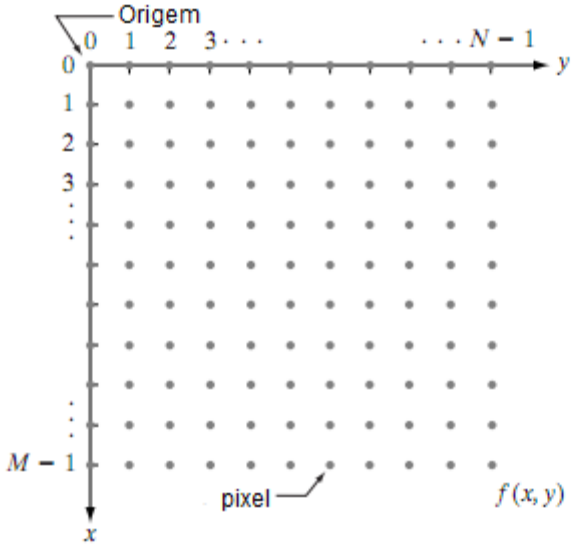


Figura 2: Representação de uma imagem digital.

2.2.1 Processo de Digitalização de Imagem

Para criar uma imagem digital, é necessário um processo para converter dados contínuos provenientes de um sensor de aquisição de imagem, como o CCD (*charge-coupled device*) utilizado em câmeras digitais, no formato digital. O processo possui três etapas: Aquisição, Amostragem e Quantização.

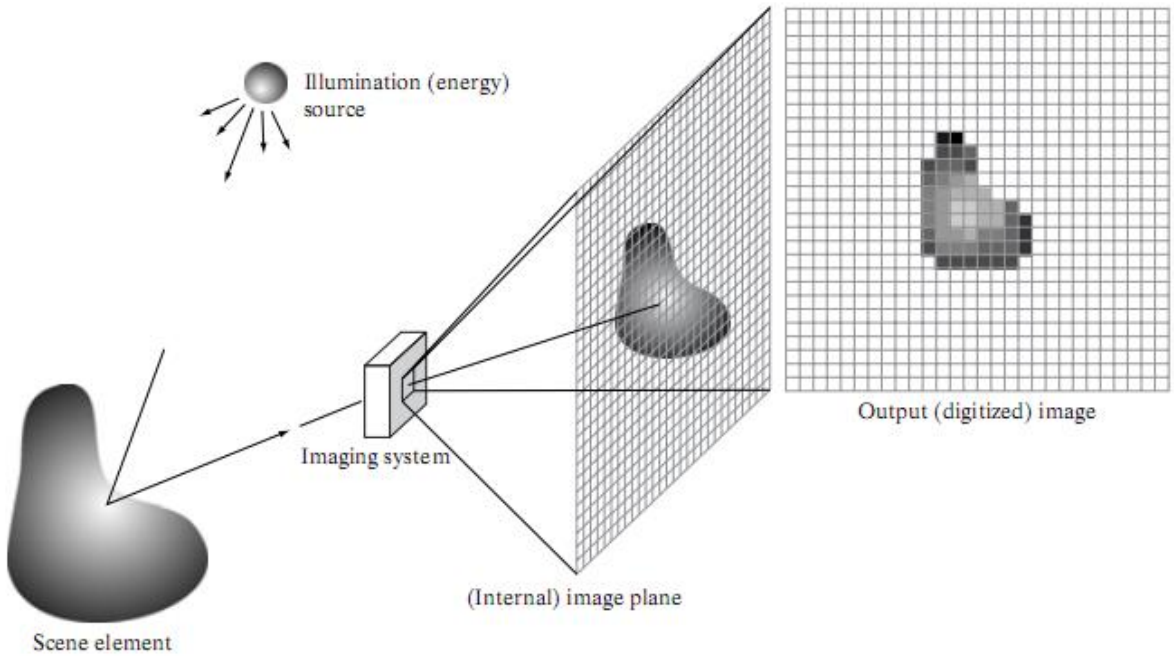


Figura 3: Processo de Digitalização de Imagem.
Fonte: [4]

Na **aquisição**, um conjunto de sensores tem por objetivo captar a energia refletida em um elemento. Em um CCD, uma matriz de sensores é utilizada para a captação dessa energia e terá como saída um nível de voltagem proporcional a energia incidente em cada sensor. Circuitos analógicos e digitais varrem essa saída e convertem em um sinal de vídeo, e envia para os circuitos de Amostragem. [4]

Na **amostragem**, o sinal de entrada proveniente da etapa anterior é um sinal contínuo tanto em amplitude quanto em coordenadas, conforme a Figura 4(b), que mostra o sinal de uma linha da imagem original (a alta variação é devido a ruído na imagem). No processo de amostragem, amostras igualmente espaçadas são retiradas do sinal contínuo, conforme a Figura 4(c). A localização de cada amostra é mostrada como a escala abaixo do sinal e os quadrados brancos representam o valor de amplitude da amostra.

Na última etapa do processo de digitalização, a **quantização**, cada amostra do sinal de entrada proveniente da amostragem precisa ser aproximada para o nível de cinza mais próximo representável pelo computador. Conforme podemos ver no lado direito Figura 4(c),

existe uma escala de 8 níveis de cinza e os valores de amplitude de cada amostra serão aproximados dependendo da proximidade do valor em relação à escala. A Figura 4(d) mostra o segmento inicial AB digitalizado.

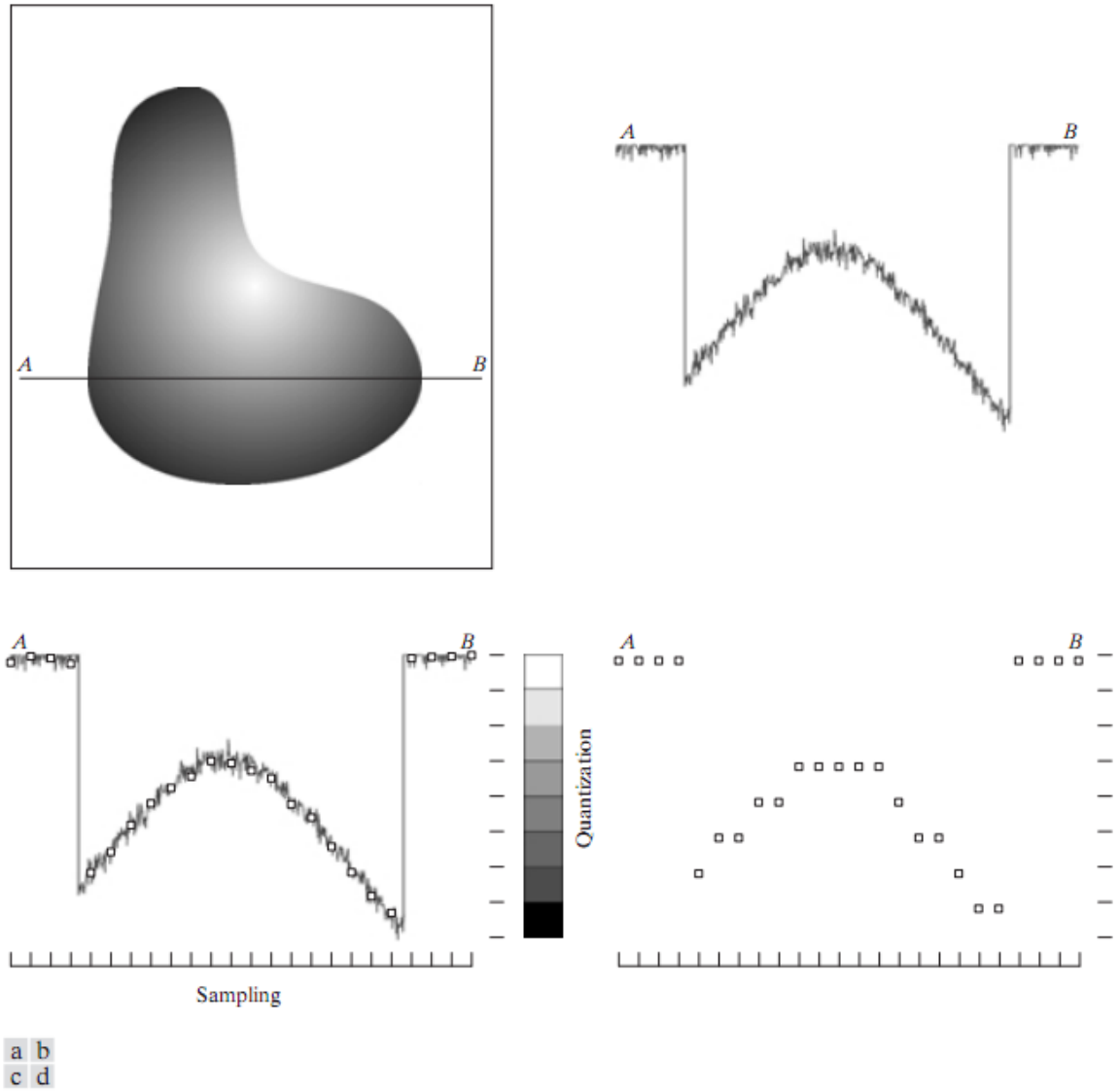


Figura 4: (a) Imagem analógica (b) Sinal analógico entre os pontos A e B (c) Amostragem (d) Quantização.
Fonte: [4]

2.2.2 Imagens Digitais

Após o fim do processo de digitalização, o resultado final é uma imagem digital, com valores de coordenadas e de intensidade de pixel discretas. Durante esse processo, foram necessárias decisões quanto aos valores de coordenadas (e, conseqüentemente, os valores do tamanho da imagem) e dos níveis de cinza discretos que cada pixel poderia assumir.

Segundo [4], não existem requerimentos quanto aos valores de coordenadas e resoluções, além do fato de serem valores positivos. Entretanto, considerando processamento,

armazenamento e amostragem no *hardware*, o número de níveis de cinza é geralmente um inteiro a uma potência de 2:

$$L = 2^k$$

O número de bits, *b*, necessários para representar uma imagem de tamanho $M\ x\ N$, é:

$$b = M\ x\ N\ x\ k$$

Considerando uma imagem quadrada $M = N$, então:

$$b = N^2\ x\ k$$

A Figura 5 mostra o número de bits necessários para representar imagens, variando o valor de *N* e *k*.

N/k	1 ($L = 2$)	2 ($L = 4$)	3 ($L = 8$)	4 ($L = 16$)	5 ($L = 32$)	6 ($L = 64$)	7 ($L = 128$)	8 ($L = 256$)
32	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192
64	4,096	8,192	12,288	16,384	20,480	24,576	28,672	32,768
128	16,384	32,768	49,152	65,536	81,920	98,304	114,688	131,072
256	65,536	131,072	196,608	262,144	327,680	393,216	458,752	524,288
512	262,144	524,288	786,432	1,048,576	1,310,720	1,572,864	1,835,008	2,097,152
1024	1,048,576	2,097,152	3,145,728	4,194,304	5,242,880	6,291,456	7,340,032	8,388,608
2048	4,194,304	8,388,608	12,582,912	16,777,216	20,971,520	25,165,824	29,369,128	33,554,432
4096	16,777,216	33,554,432	50,331,648	67,108,864	83,886,080	100,663,296	117,440,512	134,217,728
8192	67,108,864	134,217,728	201,326,592	268,435,456	335,544,320	402,653,184	469,762,048	536,870,912

Figura 5: Número de bits necessários para representar uma imagem.
Fonte: [4]

Geralmente são utilizados 256 níveis de cinza para representar uma imagem em escala de cinza. Logo:

$$L = [0, 255]$$

O valor 0 representa a ausência de luminosidade (preto total) e o valor 255 representa luminosidade máxima (branco total), e a escala intermediária representa os valores de níveis de cinza, como mostrado na Figura 6.

Figura 6: Imagem com 256 níveis de cinza.

3 CRIPTOGRAFIA

A palavra criptografia é formada a partir dos termos gregos *kryptos* (escondido, oculto) e *graphé* (grafia, escrita) e é a ciência que torna possível a comunicação segura entre dois agentes, sobre um canal aberto, convertendo informação legível em algo sem sentido, com a capacidade de ser recuperada ao estado original com o uso de processos inversos ou não. [5].

A criptografia é o estudo de técnicas matemáticas relacionada a aspectos de segurança da informação como confidencialidade, integridade de dados, autenticação de entidade e autenticação de origem de dados. [6]

Além da criptografia, existe também a criptoanálise, que é a arte de obter a informação escondida em um texto cifrado, sem que exista o acesso à chave secreta requerida para tal informação. Juntas, a criptografia e a criptoanálise formam a ciência chamada de Criptologia.

3.1 Objetivos da Criptografia

Muitos objetivos de segurança podem ser citados, porém existem quatro principais dos quais outros podem ser derivados [7]. São eles:

1. **Confidencialidade**: é um serviço utilizado para esconder o conteúdo da informação de todos que não tem autorização para tê-las. Sigilo é um termo sinônimo à confidencialidade e privacidade. Existem diversas abordagens para prover confidencialidade, desde proteção física a algoritmos matemáticos que tornem os dados ilegíveis.
2. **Integridade de dados**: é um serviço que identifica a alteração não autorizada de dados. Para assegurar a integridade de dados, a entidade deve possuir a possibilidade de detectar a manipulação dos dados por entidades não autorizadas. A manipulação de dados inclui inserções, remoções e substituições.
3. **Autenticação**: é um serviço relacionado à identificação. Essa função é aplicável tanto à entidade quanto informações. Duas entidades começando uma comunicação devem identificar uma à outra. A informação enviada através do canal deve ser autenticada quanto à origem, data de origem, conteúdo dos dados, horário enviado, etc. Por essa razão essa área da criptografia geralmente é subdividida em duas grandes áreas: Autenticação de Entidade e Autenticação de

origem de dados. Autenticação de origem de dados implica a integridade dos dados (se as informações forem modificadas, a sua origem foi alterada).

4. **Não-Repúdio**: é um serviço que previne uma entidade negar uma ação ou comprometimento passado. Quando uma disputa surge devido uma entidade negar que algumas ações foram feitas, uma maneira de resolver a situação é necessária. Por exemplo, uma entidade pode autorizar uma compra de propriedade por outra entidade e, mais tarde, negar que tal autorização foi dada. Um processo envolvendo uma terceira entidade confiável é necessário para resolver a disputa.

Um objetivo fundamental da criptografia é implementar adequadamente essas quatro áreas tanto na teoria quanto na prática. A criptografia trata de prevenção e detecção de fraudes e outras atividade maliciosas.

3.2 Princípio de Kerckhoff

Uma regra fundamental da criptografia é que se deve assumir que o analista especializado conhece o método utilizado na encriptação e decriptação [7]. A quantidade de esforço necessário para desenvolver, testar e instalar um novo algoritmo toda vez que o algoritmo antigo seja comprometido sempre dificultou a manutenção desse segredo.

Por isso a importância da chave secreta. A chave nada mais é do que um vetor de caracteres que, ao parametrizar um método de encriptação, torna a mensagem sigilosa para todos que não possuem tal chave. E, ao contrário do próprio algoritmo, a chave secreta pode ser trocada com mais freqüência do que o algoritmo, caso tenha sido comprometida.

A idéia de que o criptanalista conhece o algoritmo e que o segredo depende somente da chave é chamada Princípio de Kerckhoffs, citado por Dr. Auguste Kerckhoffs (1835, 1903), em 1883, em um artigo intitulado *La Cryptographie Militaire* (“Criptografia Militar”), junto a outros cinco princípios de desenvolvimento de criptografia práticos:

1. O sistema deve ser não inquebrável na teoria, deve ser inquebrável na prática.
2. Um sistema criptográfico deve ser seguro mesmo que tudo sobre o sistema criptográfico é de conhecimento público com exceção da chave. (Princípio de Kerckhoffs)
3. A chave deve ser lembrada sem a necessidade de anotações e deve ser facilmente trocada.
4. O texto cifrado deve poder ser enviado por telégrafo.

5. Os equipamentos e documentação necessária devem ser portáteis e operáveis por uma pessoa.
6. O sistema deve ser fácil, sem a necessidade de conhecimento sobre uma longa lista de regras.

3.3 Modelo de um Sistema Criptográfico de Chave Simétrica

O sistema da Figura 7 representa um Modelo de Sistema Criptográfico de Chave Simétrica, onde é utilizada uma mesma chave k, tanto na encriptação quanto na decriptação.

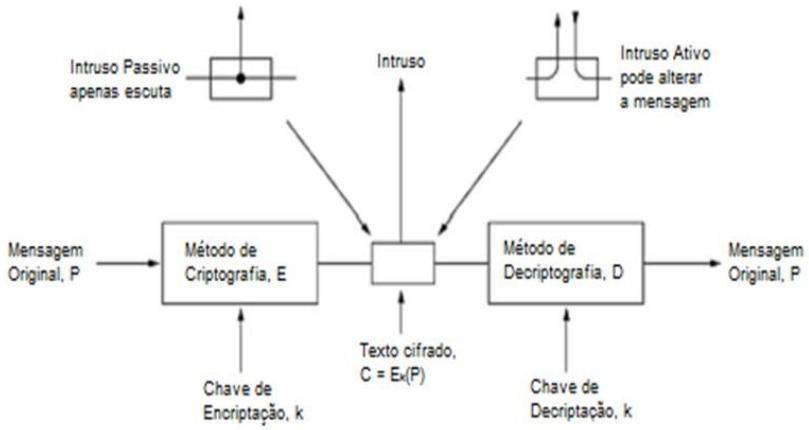


Figura 7: Sistema Criptográfico de Chave Simétrica

O sistema inicia quando se deseja enviar uma mensagem *P* sigilosa para um destino. Essa mensagem original é conhecida como texto simples. No bloco Método de Criptografia, E, essa mensagem é transformada por uma função que é parametrizada por uma chave *k*. A saída desse bloco é chamada texto cifrado *C*. Podemos definir então:

$$C = E_k(P) \tag{3.1}$$

O texto cifrado C é enviado por um canal que não possibilita a segurança da informação. Nesse momento podem existir dois tipos de intrusos tentando recuperar a informação: o intruso passivo e o intruso ativo.

O intruso passivo somente escuta o canal de comunicações e copia todo o conteúdo do texto cifrado. Entretanto, ele não compartilha a chave *k*, e terá certa dificuldade para decifrar o texto cifrado, dependendo do método empregado no primeiro bloco.

O intruso ativo também escuta o canal de comunicações, mas, diferentemente do intruso passivo, ele captura a mensagem, impedindo que a mesma chegue ao destinatário. O objetivo disso é tentar decifrar o texto cifrado e injetar ou modificar as informações contidas

no texto cifrado, encriptar essa nova mensagem e enviar ao seu destinatário como se fosse legítima.

No último bloco, Método de Decriptografia, D, é feito o processo inverso ao processo de criptografia, utilizando a mesma chave *k*, para recuperar a mensagem original *P*.

$$D_k(C) = D_k\big(E_k(P)\big) = P \tag{3.2}$$

Essa notação sugere que as funções $E()$ e $D()$ são simplesmente funções matemáticas, o que é verdade [6].

3.4 Métodos de Criptografia de Chave Simétrica

Existem basicamente duas técnicas de criptografia simétrica: a cifra de substituição e a cifra de transposição (ou permutação), também conhecidas como cifras de blocos. Ambos podem atuar *bit*-a-*bit* ou em blocos de dados.

3.4.1 Técnica de Substituição

Em um cifra de substituição, cada letra ou grupo de letras é substituído por outra letra ou grupo de letras, de modo a criar um disfarce. Uma das formas de cifra de substituição mais antiga é a chamada Código de César. Foi utilizada para enviar mensagens cifradas na Roma antiga por Júlio César. [6]

Código de César																									
P_i	a	b	c	d	e	f	g	h	I	j	k	l	M	n	o	p	q	r	s	t	u	v	w	x	y	z
E_i'	d	e	f	g	h	I	J	k	L	m	n	o	P	q	r	s	t	u	v	w	x	y	z	a	b	c

Tabela 1: Substituições efetuadas no Código de César.

No código de César, cada letra é substituída por uma letra três posições à frente no alfabeto. Generalizando a cifra de substituição temos:

$$E_i = P_i + k \; mod \; n$$

Onde:

E_i = a letra i encriptada

P_i = a letra i da mensagem original

k = deslocamento de posições no alfabeto

n = tamanho do alfabeto

Notamos então que o Código de César é uma cifra de substituição com valor de *k* = 3 e *n* = 26.

A primeira vista podemos pensar que a cifra de substituição é segura e que um intruso teria dificuldade de testar todas as $26! \approx 4\,x\,10^{26}$ possíveis combinações. Porém, técnicas de criptoanálise modernas podem descobrir a mensagem utilizando um texto de volume cifrado relativamente pequeno. Para tal, o criptanalista necessita saber o idioma em que a mensagem original foi escrita e faz uma análise de freqüência de ocorrência de letras, diagramas e trigramas da mensagem cifrada, que é igual à freqüência do idioma original.

Essa forma de cifra de substituição é chamada cifra de substituição monoalfabética. Além dessa existem também a cifra de substituição homófona e cifra de substituição polialfabética. [7]

3.4.1.1 Cifra de Substituição Homófona

Na cifra de substituição homófona, cada letra ou grupo de letras também é substituído por outra letra. A diferença é que cada letra possui mais de uma letra associada para a criptografia.

Cifra de Substituição Homófona																									
P_i	a	b	c	d	E	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
E_i'	g	A	v	0	S	2	r	D	R	4	o	Q	n	5	m	6	7	l	P	S	U	h	X	K	O	L
E_i''	p	1	B	t	C	x	3	q	Y	b	y	E	c	F	z	d	8	9	I	F	w	J	T	i	Z	u
E_i'''						a		M					N			G		e	H	V		k				

Tabela 2: Cifra de Substituição Homófona.

Esta técnica foi criada para dificultar a análise da freqüência de letras no texto cifrado, pois são atribuídas letras com menor freqüência de ocorrência a letras com maior freqüência de ocorrência, em uma tentativa de equalizar a distribuição de freqüências.

3.4.1.2 Cifra de Substituição Polialfabética

Na cifra de substituição polialfabética, múltiplos alfabetos são utilizados no processo criptográfico. O algoritmo mais conhecido dessa forma de encriptação é a Cifra de Vigenère, que utiliza a *tabula recta*, que é uma matriz retangular onde cada linha é feita deslocando a linha superior em uma posição para a esquerda, e uma chave que vai identificar qual linha será utilizada na criptografia.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figura 8: *Tabula Recta*
Fonte: [8]

Na cifra de Vigenère, uma chave é utilizada para identificar qual linha da tabela será utilizada para criptografar uma letra. Caso a chave seja menor que a mensagem, costuma-se repetir a chave até que chave e mensagem original possuam o mesmo tamanho. Por exemplo, supondo que se deseja criptografar o seguinte texto:

P = CIFRADEVIGENERE (“cifra de vigenere”)

E desejamos utilizar a chave secreta:

k = MONOGRAFIA

Notamos que a chave k é menor que a mensagem P a ser criptografada, então é necessário repetir a chave k até que possua o mesmo tamanho da mensagem original P:

k = MONOGRAFIAMONOG

Para criptografar cada letra é necessário verificar a letra correspondente à letra original na *tabula recta*, na linha indicada na chave. No exemplo, a letra C é criptografada utilizando a letra M da chave *k*, que na tabela corresponde à letra O, a letra I é criptografada utilizando a letra O da chave *k*, que na tabela corresponde à letra W, e assim por diante. No fim o texto cifrado seria:

E = OWSFMUEAQNQUEGK

Outros algoritmos de cifra de polialfabética funcionam de forma similar:

- Cifra de Gronsfeld: idêntica à de Vigenère, porém ao invés da *tabula recta*, utiliza 10 alfabetos e a chave é numérica.
- Cifra de Beaufort: utiliza também a *tabula recta*, porém o algoritmo criptografa a letra de forma diferente.

- *Autokey*: criptografa a mensagem misturando o texto na chave.
- *Running Key Cipher*: utiliza uma passagem de livro como uma longa chave para criptografar um texto.
- Enigma: implementação em hardware, utilizada pelas forças militares alemães durante a Segunda Guerra Mundial.

A cifra de substituição polialfabética apresenta uma maior dificuldade de quebra de código através da análise de freqüência, pois a freqüência de ocorrência é mascarada devido à utilização de vários alfabetos. Porém, quanto menor o tamanho da chave, menor a segurança do método, pois haverá maior repetição da chave durante a criptografia.

3.4.2 Cifra de Transposição

Na cifra de substituição, as letras são trocadas por outras letras para disfarçar as palavras, porém a ordem das letras na palavra continua a mesma. Na cifra de transposição, porém não há troca de letras, mas a reordenação delas. Além disso, a chave utilizada não contém nenhuma letra repetida.

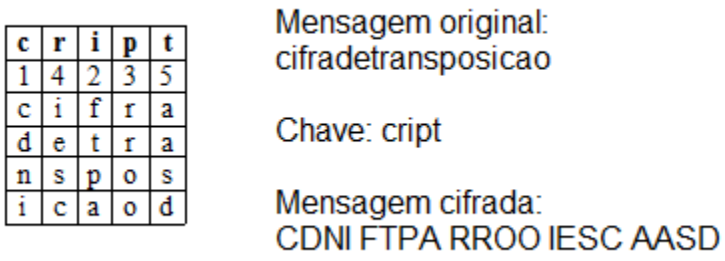


Figura 9: Exemplo de cifra de Transposição.

A Figura 9 mostra um exemplo de cifra de transposição. A chave utilizada na encriptação é “cript”. Na cifra de transposição, a chave tem dois objetivos: o primeiro é definir o número de colunas em que será escrito a mensagem original e a numeração das colunas. A numeração das colunas é feita crescentemente, iniciando com 1 na coluna que possui a letra da chave mais próxima do início do alfabeto, assim por diante, até que se tenha feito a seqüência numérica para todas as letras da chave.

Após a numeração das colunas, a mensagem original é escrita de tal forma que cada letra da mensagem ocupe uma célula da linha, caso a linha tenha sido totalmente preenchida e a mensagem não tenha chegado ao fim, continua-se a escrever a mensagem na próxima linha. Ao fim, caso a mensagem tenha acabado e células de uma linha ficaram vazias, preenchem-se

tais células com letras quaisquer. Na Figura 9, a mensagem não ocupou todos os espaços da última linha e adicionou uma letra d ao final.

Após o preenchimento da tabela, escrevem-se as colunas na ordem da numeração inicial. Espaçam-se igualmente os grupos de letras para tentar dificultar onde inicia e termina cada palavra da mensagem original.

Para decifrar a mensagem cifrada utilizando a chave basta fazer o processo inverso, numerando as colunas a partir da chave e escrevendo os grupos de letras em cada coluna e lendo a mensagem nas linhas.

A quebra de uma mensagem cifrada possui uma facilidade maior devido ao fato de que as letras continuam com a mesma freqüência de ocorrência do idioma original, pois só foram reordenadas.

3.4.3 Principais Algoritmos de Criptografia de Chave Simétrica

Algoritmos modernos utilizam as mesmas idéias básicas das cifras de substituição e transposição, porém a ênfase é diferente. Antigamente, os métodos de criptografia eram simples e a segurança estava contida no tamanho muito longo da chave, porém atualmente a idéia é diferente: os algoritmos atuais são feitos de forma a possuir chaves relativamente pequenas, porém tão complexos e emaranhados que um criptoanalista, nem mesmo com quantidades enormes de texto cifrado de sua própria escolha, seja capaz de quebrar o código.

Os algoritmos mais utilizados atualmente para transmissão de dados criptografados são os algoritmos DES e AES. Ambos são padrões norte-americanos definidos pela NIST (*National Institute of Standards and Technology*) através das FIPS (*Federal Information Processing Standard*) PUB 46-3 e PUB 197, para TDES e AES, respectivamente.

3.4.3.1 Data Encryption Standard (DES)

O algoritmo DES foi desenvolvido em 1976 pela IBM. Foi adotado como padrão de encriptação de dados pelos Estados Unidos e, posteriormente, em muitos lugares do mundo,. A forma original do DES não é mais segura devido ao aumento de poder computacional crescente no mundo. Dessa forma, em 1998, foi substituído pelo TDEA (*Triple Data Encryption Algorithm*), que aplica o DES três vezes em cada bloco de dados.

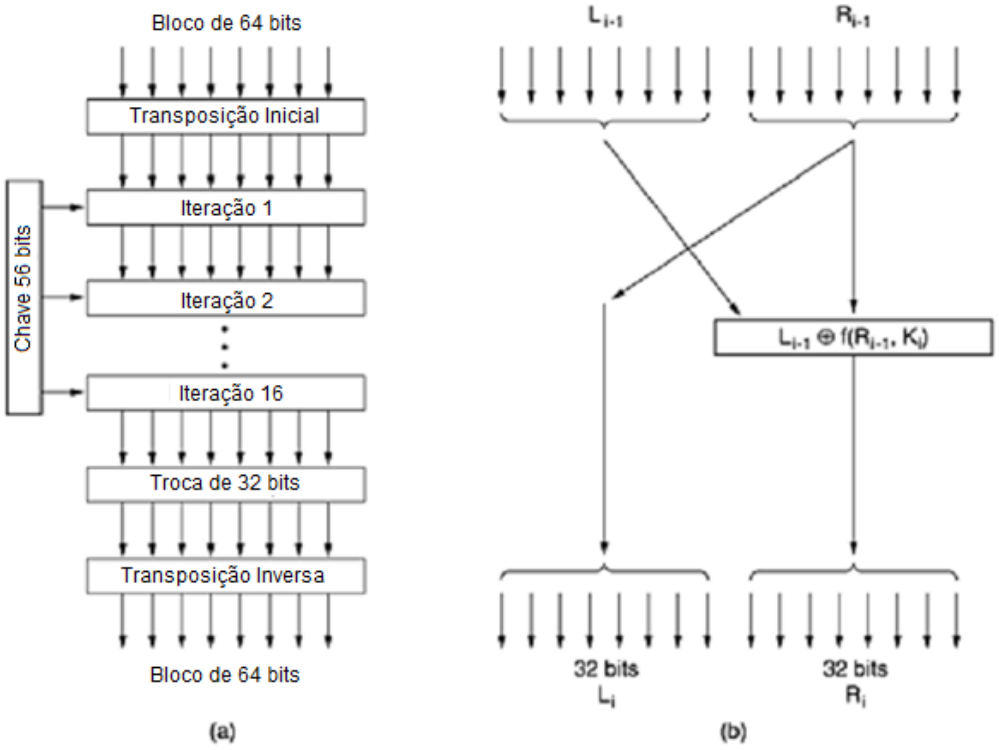


Figura 10: Algoritmo DES. (a) Passos do algoritmo. (b) Detalhe de uma iteração.

Na Figura 10(a), é mostrado o algoritmo do DES. O algoritmo inicialmente quebra uma mensagem em blocos de 64 bits e criptografa cada bloco separadamente. Ao final, o algoritmo junta todos os blocos.

O algoritmo é parametrizado por uma chave de 56 bits e possui 19 estágios distintos [6]. O passo inicial é uma transposição no bloco da mensagem original de 64 bits, e o último passo é o inverso da transposição. O penúltimo bloco faz uma troca dos 32 bits à esquerda pelos 32 bits à direita. Os 16 blocos intermediários são essencialmente idênticos, parametrizados pela chave. Na decriptação, somente aplica-se o algoritmo na ordem inversa.

Os 16 blocos intermediários são parametrizados por chaves diferentes, derivadas da chave original. A chave original de 56 bits é dividida em duas partes de 28 bits e cada parte é rotacionada à esquerda um número de bits que depende do número atual do bloco de iteração.

O funcionamento desses blocos intermediários é mostrado na Figura 10(b). Cada bloco usa duas entradas de 32 bits e produz duas saídas de 32 bits. Como mostrado na figura, a saída da direita é uma cópia da entrada da esquerda. A saída da direita é formada pelo resultado do OU-EXCLUSIVO *bit*-a-*bit* aplicado entre a entrada da esquerda e uma função f que tem como entrada, a entrada da direita e a chave rotacionada desse bloco. A complexidade do algoritmo reside no procedimento interno dessa função, chamada Função de Feistel.

A Função de Feistel possui quatro estágios: expansão, mistura da chave, substituição e permutação, como é mostrada na Figura 11. Na expansão, a entrada da direita de tamanho 32 bits é expandida, de acordo com uma transposição fixa e de uma regra de duplicação, em 48 bits. Esses 48 bits passam por um processo de OU-EXCLUSIVO com a chave específica do bloco. Esse estágio é chamado de mistura de chave.

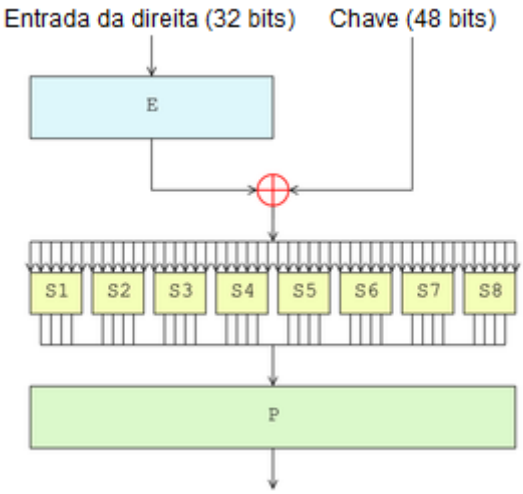


Figura 11: Função de Feistel

No estágio de substituição, esses 48 bits são divididos em 8 grupos de 6 bits, que passam por um processo de substituição nas caixas S1 a S8 mostradas na Figura 11. Cada caixa mapeia um grupo de 6 bits em grupos de 4 bits. É utilizado um método de transformação não-linear no mapeamento que proporciona a segurança do algoritmo[7]. No último estágio, de permutação, esses 8 grupos de 4 bits passam por uma permutação fixa.

3.4.3.2 Triple Data Encryption Algorithm (TDEA)

Em 1979, a IBM percebeu que o tamanho da chave utilizada no DES era muito pequeno e criou uma forma de aumentá-lo utilizando criptografia tripla [6]. O processo foi incorporado ao padrão DES para aumentar a segurança do processo, em 1999 [9]. O TDEA pode ser visto na Figura 12.

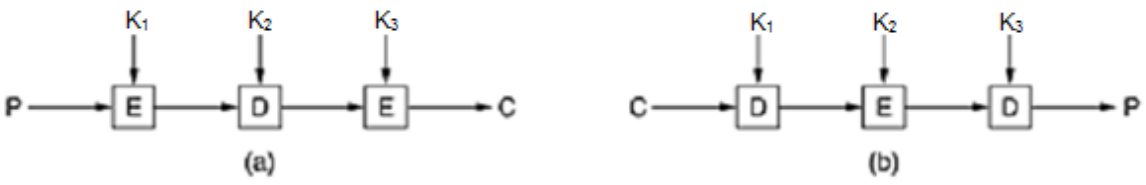


Figura 12: TDEA (a) Encriptação (b) Decriptação.

No TDEA são utilizadas três chaves (K_1, K_2, K_3) pra criptografar e decriptografar um bloco de 64 bits. O algoritmo foi proposto de tal forma que tanto o processo de encriptação quanto de decriptação tivessem compatibilidade com o padrão DES de criptografia única.

O algoritmo pode ser aplicado de três formas:

1. $K_1 \neq K_2 \neq K_3$: criptografia mais segura contra ataques, pois possui chave de tamanho total de $3\ x\ 56 = 168$ bits. Não é muito utilizado, pois cria um *overhead* desnecessário de gerenciar e transportar a chave, com pouco ganho real [6].
2. $K_1 \neq K_2$, $K_1 = K_3$: menos segura que a anterior, porém mais segura que o DES de criptografia única, com uma chave de tamanho total $2\ x\ 56 = 112$ bits.
3. $K_1 = K_2 = K_3$: utilizado para compatibilidade com dispositivos que utilizem DES de criptografia única.

3.4.3.3 Advanced Encryption Standard (AES)

Com o passar do tempo, o DES, mesmo com o TDEA, tornou-se relativamente fraco frente a nova geração de computação paralela. Em 1998, o computador *EFF DES Cracker*, custando em torno de US$250.000, quebrou uma mensagem cifrada em DES em 56 horas [10]. Em 2006, o computador COPACOBANA, quebrou uma mensagem cifrada em DES em 6,4 dias, com um custo de US$10.000 [11]. O sucessor do COPACOBANA, RIVYERA, diminui o tempo para menos de um dia [12].

Em 1997, o NIST patrocinou um concurso de criptografia que iria eleger o novo padrão de criptografia a ser utilizado pelas agências norte-americanas. Foram feitas 15 propostas e organizadas conferências públicas para apresentação e análise de cada algoritmo a procura de falhas. Em outubro de 2000, o algoritmo Rijndael (de Joan Daemen e Vicent Rijmen) foi escolhido para ser o próximo padrão de encriptação[13], denominado AES [6]. O algoritmo Rijndael é mostrado na Figura 13.

O algoritmo aceita chaves de tamanho 128, 192 ou 256 *bits*. O tamanho da chave define também o número de rodadas, N_r, da Figura 13, a qual varia de 10 para chave de 128 *bits* até 14 para chave de 256 *bits*. No entanto, diferente do DES, todas as operações são feitas utilizando *bytes*, possibilitando implementações eficientes, tanto em hardware quanto em software [6].

O algoritmo codifica blocos de 16 *bytes*, e durante cada rodada o estado atual da mensagem cifrada é armazenado em uma matriz de *bytes* chamado estado, cujo tamanho depende do tamanho da chave, sendo 4 x 4 *bytes* para chave de tamanho 128 bits.

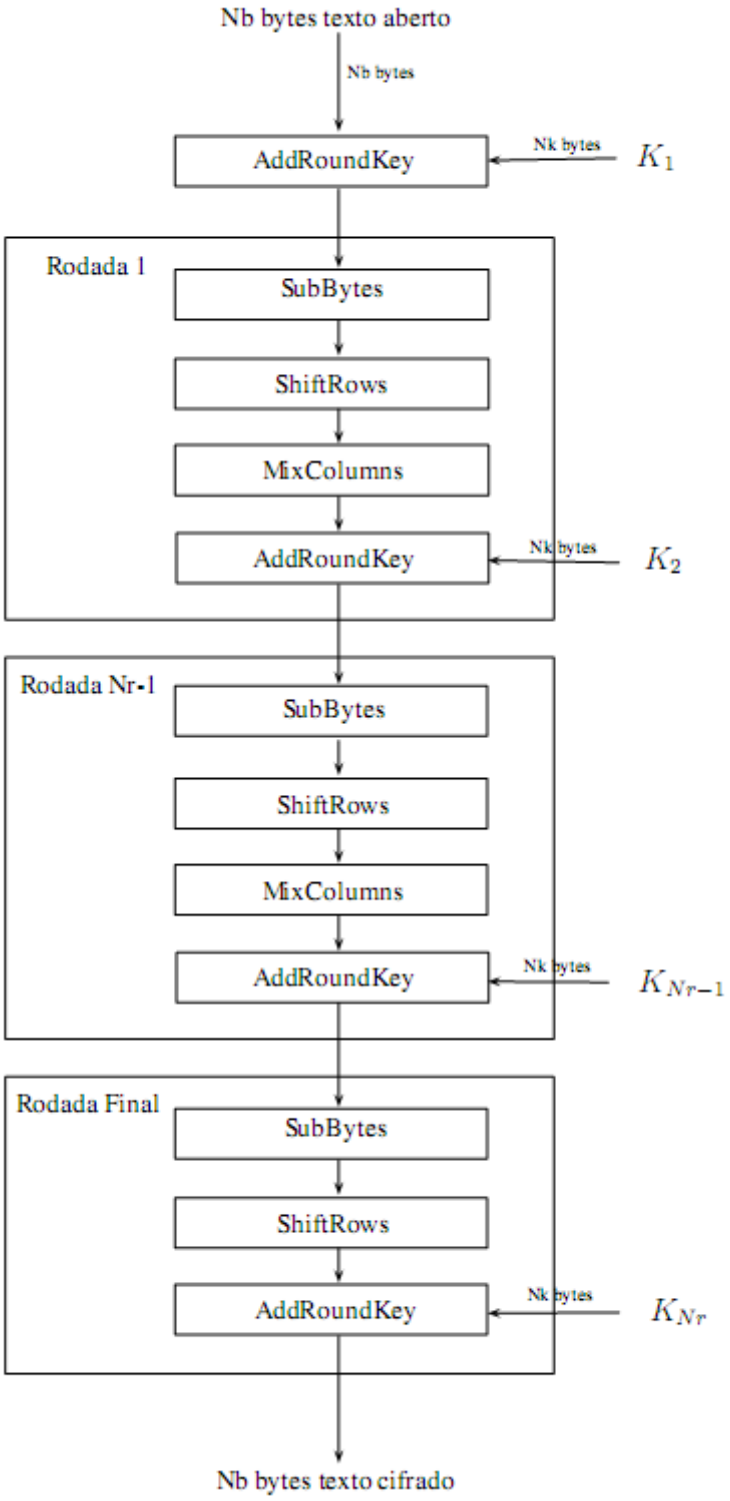


Figura 13: Fluxograma do algoritmo Rijndael.
Fonte: [14]

O processo de encriptação inicia expandindo a chave em N_r+1 vetores do mesmo tamanho que o estado. Um será utilizado no início do cálculo e os outros serão utilizados durantes as N_r rodadas (um por rodada). Em seguida, o texto é copiado para a matriz estado, preenchendo as colunas seqüencialmente. Além disso, antes de começar as rodadas, é feita uma operação OU EXCLUSIVO entre a matriz estado e a primeira chave expandida. Esse processo é chamado AddRoundKey e também utilizado ao final de cada rodada.

As rodadas seguintes são dividas em quatro etapas: SubBytes, ShiftRows, MixColumns e AddRoundKey. Na última rodada, porém, a operação MixColumns não é realizada. Na etapa SubBytes, mostrada na Figura 14, uma substituição não-linear irreversível opera em cada *byte* da matriz estado. Cada byte é usado como índice para um bloco de substituição, a fim de substituir o seu valor pelo conteúdo dessa entrada na cifra de substituição [6]. Basicamente, é uma cifra de substituição monoalfabética.

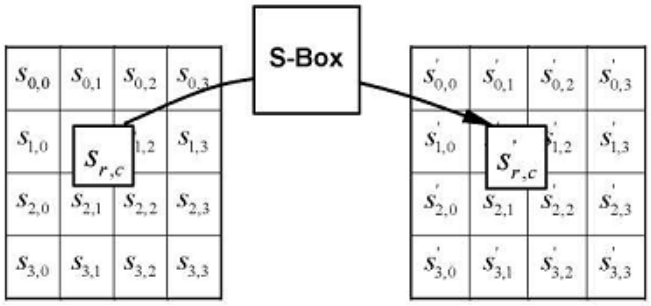


Figura 14: Operação SubBytes.
Fonte: [5]

Na etapa ShiftRows, cada linha da matriz estado é girada para esquerda, um número de vezes que representa a linha na matriz, ou seja, a linha 0 é girada 0 *byte* para a esquerda (não se altera), a linha 1 é girada 1 *byte* para esquerda e assim por diante, conforme a Figura 15.

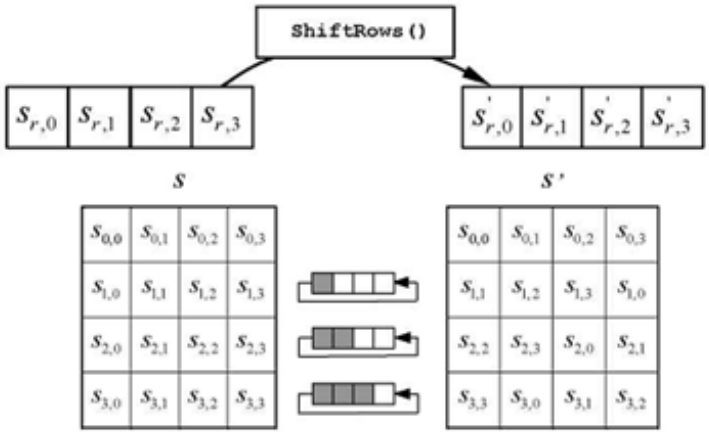


Figura 15: Operação ShiftRows.
Fonte: [5]

Na etapa MixColumns, mostrada na Figura 16, cada coluna da matriz estado é misturada independentemente. Essa mistura é realizada pela multiplicação da matriz estado com uma matriz constante. Embora isso possa parecer um *overhead* no processo de encriptação, existe um algoritmo que permite calcular cada elemento da nova coluna utilizando duas pesquisas em *lookup tables* e três operações XOR [6].

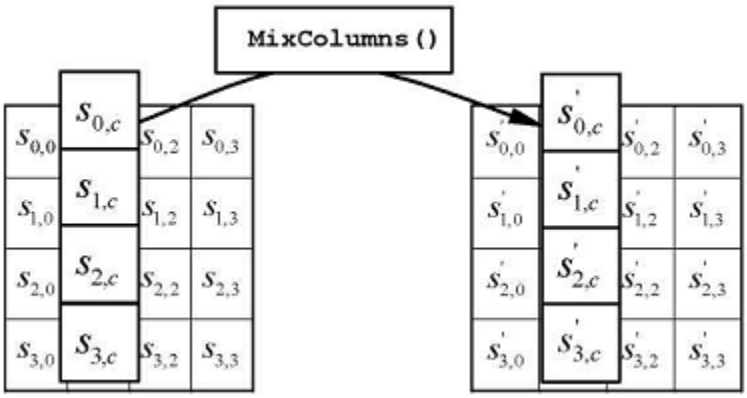


Figura 16: Operação MixColumns.
Fonte: [5]

Na última etapa, AddRoundKey, é efetuada a mesma operação que ocorreu ante do início das rodadas porém com a chave expandida correspondente da rodada, conforme pode ser visto na Figura 17.

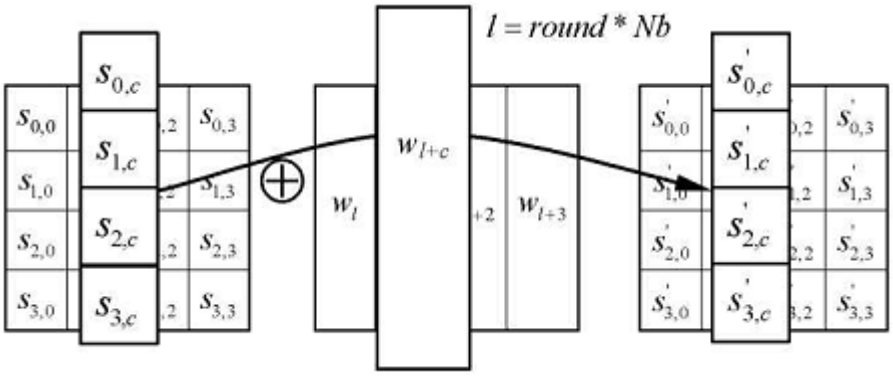


Figura 17: Operação AddRoundKey.
Fonte: [5]

O Rijndael foi desenvolvido para prover ao usuário tanto velocidade quanto segurança. Em termos de velocidade, um boa implementação em software em uma máquina de 2 GHz pode alcançar uma taxa de encriptação de 700 Mbps [6]. E, em termos de segurança, mesmo que seja desenvolvida uma máquina com um bilhão de processadores paralelos, cada um avaliando uma chave por picossegundo, levaria 10^{10} anos para pesquisar o espaço total de chaves [6].

3.5 Modelo de um Sistema Criptográfico de Chave Assimétrica

O modelo de criptografia assimétrica foi proposta em 1976, por dois pesquisadores da Universidade de Harvard, Diffie e Hellman. A sua proposta baseava-se no fato de que por mais sólido que fosse um método de encriptação, a distribuição de chaves tornava o sistema fraco, pois caso um intruso roubasse a chave, o sistema se tornaria inútil [6].

Em sua proposta inicial, um sistema de criptografia assimétrico deveria atender aos seguintes requisitos:

1. $D\big(E(P)\big) = P$
2. É extremamente difícil deduzir *D* a partir de *E*.
3. *E* não pode ser decifrado por um ataque de texto simples escolhido.

O primeiro requisito diz que se aplicado o processo de decriptação *D* em um texto cifrado *E(P)* deve-se obter o texto original. O segundo é auto-explicativo. O terceiro é importante, pois o intruso pode testar o processo de encriptação indefinidamente utilizando a chave pública. Um sistema criptográfico de chave assimétrica é mostrado na Figura 18.

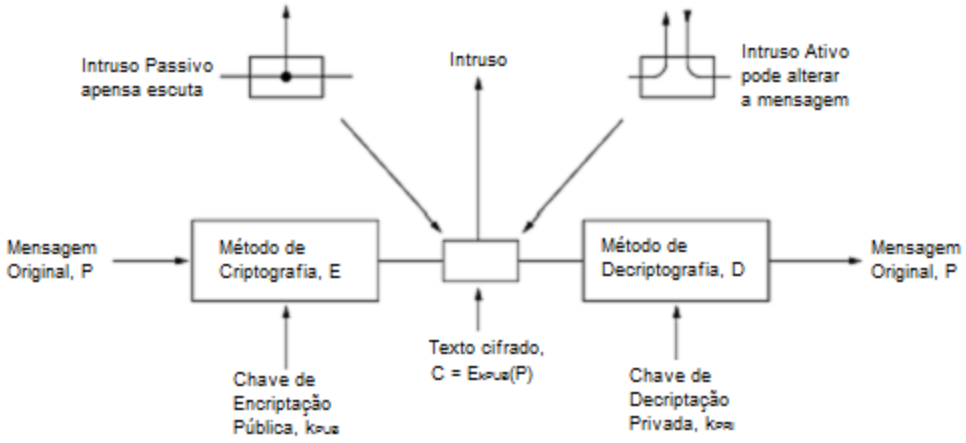


Figura 18: Sistema Criptográfico de Chave Assimétrica.

O sistema criptográfico de chave assimétrica é semelhante ao sistema criptográfico simétrico no sentido de que uma mensagem é criptografada por um processo de encriptação, passa por um meio onde pode ser recuperada e/ou alterada por um intruso e recebida e decriptada por outro usuário.

A diferença entre os dois sistemas reside no fato de que duas chaves são utilizadas no sistema, uma pública e uma privada. A chave pública, como o próprio nome diz, é de conhecimento público, qualquer usuário pode ter acesso à chave. A chave privada é possuída somente por um usuário. A chave pública e a privada são relacionadas entre si de tal forma

que uma mensagem encriptada utilizando a chave pública só pode ser decriptada utilizando a chave privada.

3.6 Métodos de Criptografia de Chave Assimétrica

Existem diversos algoritmos de criptografia assimétrica conhecidos, dos quais os mais importantes são [7]:

- RSA: considerado o mais seguro, foi proposto em 1978 por três pesquisadores do MIT. Baseia-se no problema da fatoração de números muito grandes.
- ElGamal: proposto em 1985 por Taher Elgamal. Baseia-se no problema do logaritmo discreto.
- Rabin: proposto em 1979 por Michael O. Rabin. Assim com o RSA baseia-se no problema da fatoração de números muito grandes.
- McEliece: proposto em 1978 por Robert McEliece. Baseia-se na dificuldade de decodificar códigos lineares.

3.6.1 RSA

O algoritmo mais utilizado na atualidade para encriptação de informações foi proposto em 1978, por três pesquisadores do MIT, Ron Rivest, Adi Shamir e Leonard Adleman. O nome RSA deriva da primeira letra do sobrenome dos seus criadores.

Desde que foi proposto, sobreviveu a todas as tentativas de quebra e é considerado um algoritmo bastante forte, porém sua principal desvantagem é exigir uma chave de pelo menos 1024 *bits*, em comparação aos 128 *bits* dos algoritmos simétricos.

O algoritmo funciona da seguinte forma [6]:

1. Escolha dois números primos muito grandes, p e q (geralmente, 1024 *bits*).
2. Calcule $n = p \times q$ e $z = (p - 1) \times (q - 1)$
3. Escolha um número d tal que z e d sejam primos entre si.
4. Encontre e de forma que $(e \times d) \bmod z = 1$

Com esses parâmetros calculados, o algoritmo inicia dividindo a mensagem original em blocos, de tal forma que bloco de mensagem P fique no intervalo $0 \leq P < n$. Isso pode ser feito agrupando-se cada bloco de mensagem em k bits, onde k é o maior inteiro para $2^k < n$.

A partir disso, para encriptar a mensagem P, calcule $C = P^e \ mod \ n$ e para decriptar, calcule $P = C^d \ mod \ n$. A chave pública é o número *e* e a chave privada é o número *d*. Uma outra vantagem do método, além da segurança, é que, por exemplo, p e $q \approx 2^{512}$, então $n \approx 2^{1024}$, logo cada bloco poderia encriptar até 1024 *bits*, ou 128 caracteres de 8 *bits*, em comparação aos 8 *bits* do DES e aos 16 *bits* do AES [6].

Em termos de segurança, considerando o melhor algoritmo conhecido de fatoração e um computador com tempo de instrução de 1μs, o mesmo levaria 10^{25} anos para fatorar um número de 500 *bits*.

4. CRIPTOGRAFIA UTILIZANDO IMAGENS

O método proposto baseia-se nas seguintes premissas:

1. Uma imagem digital geralmente possui 256 valores possíveis que descrevem a intensidade de um pixel em cada coordenada da imagem, com valores variando entre 0 (preto total) e 255 (branco total).
2. Assim também é a tabela ASCII, como vista no anexo A, a qual é utilizada para representar todos os caracteres imprimíveis, possuindo valores que variam entre 0 e 255.

Vale lembrar que nem todos os valores da tabela ASCII são imprimíveis por uma impressora. Os valores entre 0 e 32 da tabela ASCII são valores de controle, utilizados como bits de sinalização, como o ASCII-10 (*Line-Feed*) utilizado para mover o cursor para linha seguinte sem mudar a coluna onde se encontra o cursor, o ASCII-11 (*Carriage Return*), utilizado para mover o cursor para a coluna zero enquanto continua na mesma linha, entre outros.

4.1. Método de Criptografia utilizando Imagens

Partindo dessa semelhança entre o nível de intensidade de cinza de um pixel em uma imagem e o conjunto de caracteres da tabela ASCII, o método propõe que o processo de encriptação de um vetor de caracteres consista no mapeamento do caractere em uma determinada posição randômica na imagem, que possua o valor de intensidade de pixel igual ao valor do caractere na tabela ASCII.

4.1.1. Processo de Encriptação

O algoritmo é descrito abaixo:

Encriptação

1. Ler mensagem original *msg* de um arquivo
2. Ler imagem-chave *img*
3. Inicia mensagem cifrada *msgCript* nula
4. **Para** cada caractere *c* na *msg* **faça**
    a. Retorne uma lista *list* de par de coordenadas *(x,y)* onde *img(x,y) = c*
    b. Escolha aleatoriamente uma posição *k* na *list*
    c. Adicione ao fim de *msgCript* o par de coordenadas de *list(k)*

5. **Fim para**
6. Salve *msgCript* em um arquivo

Algoritmo 1: Encriptação.

No processo de encriptação, o Algoritmo 1 necessita da imagem que será a chave para decriptar. Chamaremos essa imagem de imagem-chave.

A partir do vetor de caracteres que serão criptografados, o algoritmo irá encriptar os caracteres um a um. Para cada caractere, o algoritmo irá selecionar um par de coordenadas aleatórias, que possua o valor de intensidade de pixel igual ao valor correspondente do caractere de entrada na tabela ASCII.

Após selecionar o par de coordenadas equivalente ao caractere, esse par é adicionado ao vetor de saída. Ao final, o vetor de saída, ou seja, o vetor de caracteres encriptados será um conjunto de pares de coordenadas.

Por exemplo, vamos supor que para um vetor de entrada que possua os caracteres "ab", desejamos criptografar utilizando a Figura 19.

Figura 19: Imagem de teste.

Durante o processo de encriptação, o algoritmo verifica os valores de cada caractere, como podemos ver na tabela no Anexo A, os valores que representam os caracteres "a" e "b" são 97 e 98, respectivamente.

A partir desses valores, o algoritmo irá procurar na imagem todas as coordenadas que possuem valor de intensidade de pixel igual a 97 (para encriptar “a”) e 98 (para encriptar “b”), e escolher aleatoriamente um par de coordenadas para representar cada um dos caracteres.

Vamos supor que as coordenadas escolhidas estejam sendo mostradas na Figura 20 em vermelho.

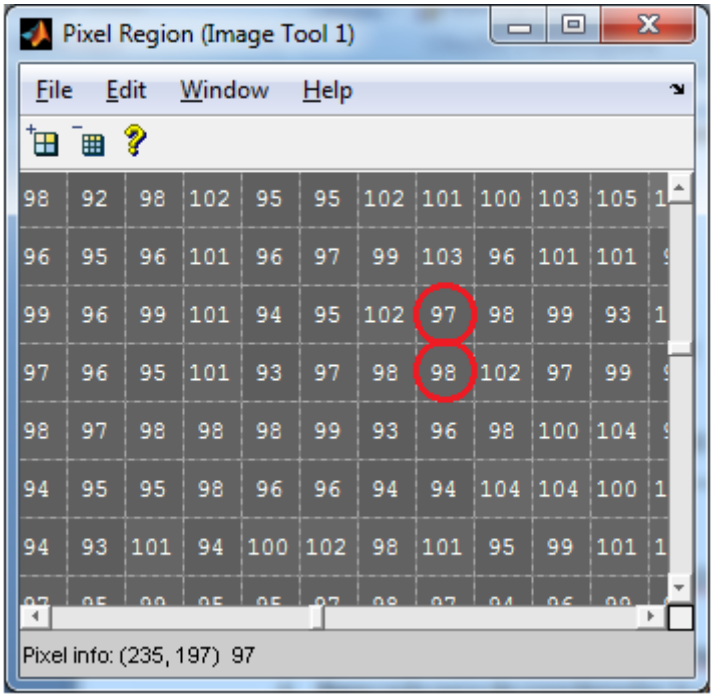


Figura 20: Coordenadas escolhidas aleatoriamente para encriptar a mensagem "ab", as coordenadas do círculo superior são mostradas no canto.

Logo, o vetor final encriptado vai ser o par de coordenadas do círculo superior e o par de coordenadas do círculo inferior. O processo é mostrado abaixo.

Vetor de Entrada = [a b]		
Representação tipo char	a	b
Representação tipo inteiro	97	98
Representação coordenadas	[235 197]	[235 198]
Vetor de Saída = [235 197 235 198]		

4.1.2. Processo de Decriptação

O algoritmo é descrito abaixo:

Decriptação

1. Ler mensagem cifrada *msgCript*
2. Ler imagem-chave *img*
3. Inicia mensagem decriptada *msgDecript* nula
4. **Para** cada par de coordenadas *(x,y)* na *msgCript* **faça**

    a. Retorne o valor do pixel *p* na coordenada *img(x,y)*
    b. Adicione ao fim de *msgDecript* o valor *p*
5. **Fim para**
6. Salve *msgDecript* em um arquivo

Algoritmo 2: Decriptação.

Para o processo de decriptação, o Algoritmo 2 necessita da mesma imagem-chave utilizada para encriptar o arquivo original.

A partir do vetor de caracteres criptografados, ou seja, do vetor de coordenadas, o algoritmo irá decriptar cada par de coordenadas uma a uma. Para cada par de coordenadas, o algoritmo irá procurar na imagem-chave o valor correspondente do pixel para cada par de coordenadas.

Após retornar esse valor de pixel, ele será adicionado ao final do vetor de saída, ou seja, o vetor decriptado. Ao final, o vetor decriptado tem que ser igual ao vetor original antes do processo de encriptação.

Como exemplo, tentaremos decriptar a mensagem encriptada anteriormente utilizando a Figura 19.

O algoritmo irá carregar as coordenadas salvas no arquivo encriptado e, para cada par de coordenadas, irá procurar na imagem o valor correspondente do pixel.

Vetor de Entrada = [235 197 235 198]		
Coordenadas encontradas	[235 197]	[235 198]
Valor de intensidade de pixel das coordenadas	97	98
Representação tipo char	a	b
Vetor de Saída = [a b]		

Ao final do processo, caso a imagem-chave utilizada para decriptar seja a mesma utilizada para encriptar, o vetor decriptado deve ser o mesmo que o vetor original.

4.2. Vantagens e desvantagens do método

4.2.1. Alta variabilidade do Arquivo Encriptado

Como descrito no algoritmo de encriptação, para cada caractere lido do vetor de entrada, é retornado uma lista de par de coordenadas na imagem que possuam valor de intensidade de pixel igual ao valor do caractere.

A vantagem disso é que, ao escolher aleatoriamente uma posição na lista, o vetor final encriptado será sempre diferente, o que garante que uma mesma mensagem original terá diferentes versões encriptadas.

Como exemplo, supondo que gostaríamos de encriptar uma mensagem de 10 caracteres utilizando a Figura 21.

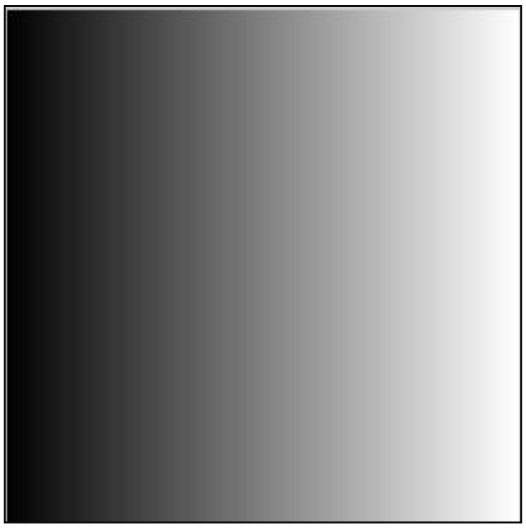


Figura 21: Imagem de dimensões 256x256 com distribuição uniforme de intensidade de pixels.

A Figura 21 possui dimensões 256x256 pixels com todos os valores de intensidade de pixel com o mesmo número de ocorrências, como podemos ver pelo seu histograma na Figura 22. Histograma é o gráfico das freqüências de ocorrências dos caracteres em uma imagem ou texto.

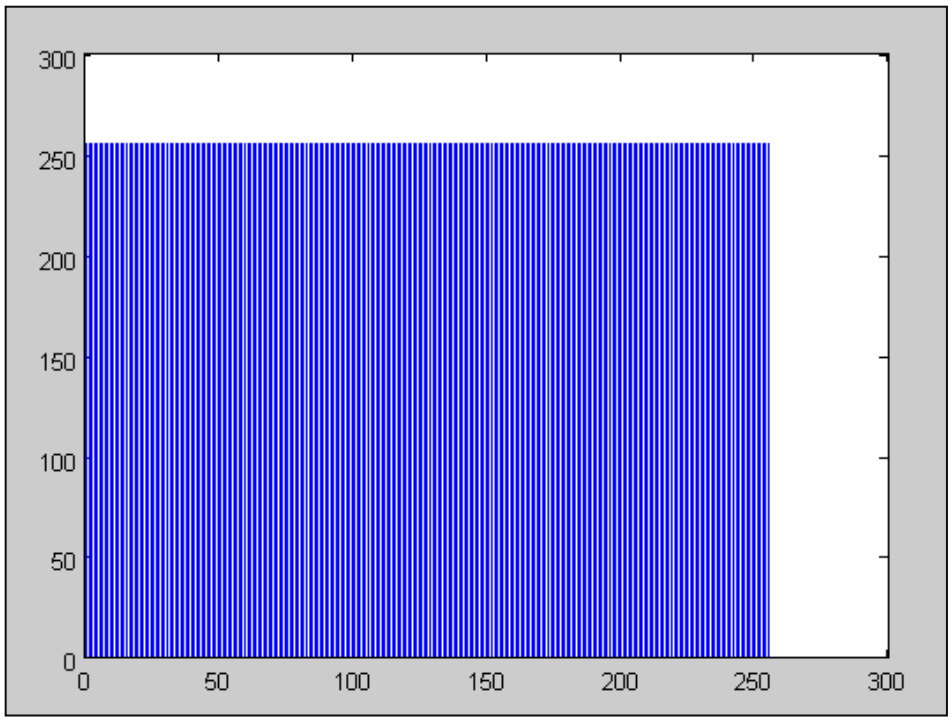


Figura 22: Histograma da Figura 21.

Para uma mensagem com 10 caracteres, o número de possíveis mensagens encriptadas é igual a:

$$N_E = \prod_{i=0}^{k-1} Oc(x,i)$$

Onde:

N_E = Número de possíveis mensagens encriptadas.

k = Tamanho da mensagem.

Oc(x,i) = Número de ocorrências do valor *x* na imagem *i*.

Como todos os caracteres possuem o mesmo número de ocorrências na imagem:

$$N_E = \prod_{i=0}^{9} 256 = 256^{10} = 2^{80} = 1{,}208\, x\, 10^{24}\ mensagens$$

E mesmo com todas essas possíveis mensagens encriptadas, a Figura 21 seria capaz de decriptá-las corretamente.

Notamos também que, qualquer imagem pode ser utilizada para decriptar a mensagem encriptada, porém a resposta só será igual se para cada par de coordenadas do vetor encriptado o valor de intensidade de pixel for igual ao da imagem original.

Suponhamos uma mensagem com 10 caracteres encriptada com a Figura 21. E, após a encriptação, desejamos decriptar utilizando a Figura 23, que nada mais é que a Figura 21 com cada linha deslocada pelo seu número (linha 1 possui todos os pixels deslocados de 1 pixel, linha 2 possui todos os pixels deslocados de 2 pixels, etc.).

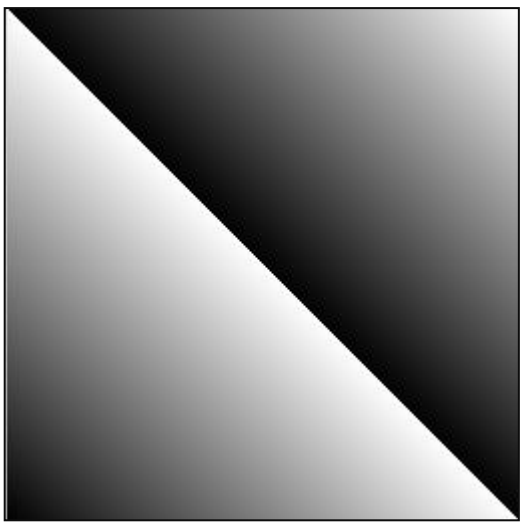


Figura 23: Imagem de dimensões 256x256 com distribuição uniforme de intensidade de pixels.

Como a Figura 23 é a Figura 21 deslocada, podemos afirmar que o histograma é o mesmo da Figura 22, porém somente uma linha é comum às duas imagens, a última linha.

Para que a mensagem encriptada anteriormente seja corretamente decriptada, é necessário que no processo de encriptação tenha selecionado, para cada caractere, as coordenadas da última linha da Figura 21, a probabilidade de isso acontecer é:

$$p(x) = \frac{1}{1{,}208\ x\ 10^{24}}$$

Na verdade, a probabilidade dessa mensagem ocorrer é igual à probabilidade de qualquer outra combinação de coordenadas. Apesar da probabilidade de decriptar um caractere é de:

$$p(x) = \frac{1}{256}$$

Logo, as chances de decriptar a mensagem corretamente é igual à probabilidade de encriptação, por que no processo de decriptação não existe nenhuma aleatoriedade.

4.2.2. Tamanho do Arquivo Encriptado

Se por um lado, o fato de utilizar uma imagem é vantajoso, devido a robustez do algoritmo, por outro lado, notamos um aumento do arquivo encriptado em relação ao arquivo original, em média, o arquivo encriptado é oito vezes maior que original.

Isso acontece porque durante o processo de encriptação, a cada byte de informação lida (um caractere, um valor entre 0 e 255), serão escolhidos um par de coordenadas. Caso esse par de coordenadas seja inteiro, de tamanho 4 bytes, o tamanho final do arquivo seria 8 vezes maior que o original (4 bytes para a posição x e 4 bytes para a posição y).

Para minimizar esse fato podemos utilizar menos bytes para representar a informação, como podemos ver na Tabela 3, o valor inteiro (int) pode representar valores extremamente altos, valores que dificilmente serão alcançados pelas coordenadas de uma imagem.

Tipo	**bits (bytes)**	**Intervalo de valores**	
		Inferior	**Superior**
Int	32 (4)	-2147483648	2147483647
unsigned int	32 (4)	0	4294967295
short int	16 (2)	-32768	32767
unsigned short int	16 (2)	0	65535
n-bit int	n (n/8)	-2^{n-1}	$2^{n-1}-1$
unsigned n-bit int	n (n/8)	0	$2^{n}-1$

Tabela 3: Valores representados pelos diferentes tipos de inteiros.

Outro ponto que podemos notar que é que o tipo inteiro representa também valores negativos, que nunca serão representados por coordenadas de imagem, que possuem valores maiores que 0.

Logo, podemos modificar os algoritmos de encriptografia e decriptografia de forma que, a partir da imagem-chave, seja escolhido o menor número possível de bits para representar os valores de coordenadas. Por exemplo, caso a imagem possua dimensões até 511 pixels, o algoritmo deverá escolher um inteiro sem sinal de 9 bits, como podemos perceber na Tabela 4.

Tipo	Intervalo de valores	
	Inferior	Superior
unsigned 09-bits int	0	511
unsigned 10-bits int	0	1023
unsigned 11-bits int	0	2047
unsigned 12-bits int	0	4095
unsigned 13-bits int	0	8191
unsigned 14-bits int	0	16383
unsigned 15-bits int	0	32767

Tabela 4: Valores de inteiros sem sinal de tamanho entre 9 e 15 bits.

A vantagem dessa escolha inteligente do algoritmo são duas: o arquivo criptografado terá um tamanho reduzido pois um caractere de tamanho 8 bits, será criptografado em 18 bits (9 bits para cada coordenada).

A outra vantagem é que caso um atacante tente decriptar a mensagem, sem que saiba as dimensões da imagem-chave, terá dificuldade em determinar as diferentes coordenadas.

Inserindo essas modificações no Algoritmo 1 e Algoritmo 2, teríamos para a encriptação:

Encriptação com coordenadas utilizando n-bits

1. Ler mensagem original *msg* de um arquivo
2. Ler imagem-chave *img*
3. Calcule o número de bits *n* máximos necessários para representar as coordenadas de *img*
4. Inicia mensagem cifrada *msgCript* nula com inteiro sem sinal de precisão *n*
5. **Para** cada caractere *c* na *msg* **faça**
    a. Retorne uma lista *list* de par de coordenadas *(x,y)* onde *img(x,y) = c*
    b. Escolha aleatoriamente uma posição *k* na *list*
    c. Adicione ao fim de *msgCript* o par de coordenadas de *list(k)*
6. **Fim para**
7. Salve *msgCript* em um arquivo utilizando *n* bits de precisão

Algoritmo 3: Encriptação com coordenadas utilizando n-bits sem sinal.

E para decriptação, temos:

Decriptação com coordenadas utilizando n-bits

1. Ler imagem-chave *img*
2. Calcule o número de bits *n* máximos necessários para representar as coordenadas de *img*
3. Ler mensagem cifrada *msgCript* utilizando *n*-bits para cada coordenada
4. Inicia mensagem decriptada *msgDecript* nula
5. **Para** cada par de coordenadas *(x,y)* na *msgCript* **faça**
    a. Retorne o valor do pixel *p* na coordenada *img(x,y)*
    b. Adicione ao fim de *msgDecript* o valor *p*
6. **Fim para**
7. Salve *msgDecript* em um arquivo

Algoritmo 4: Decriptação com coordenadas utilizando n-bits sem sinal.

Notamos que não são necessárias grandes alterações no algoritmo, a mudança ocorrerá basicamente no momento de escrita do arquivo criptografado (Algoritmo 3) e na leitura do arquivo decriptado (Algoritmo 4).

Com essas alterações o tamanho do arquivo encriptado diminui drasticamente, como será mostrado na próxima seção.

4.2.3. Tamanho da Imagem-chave

Em comparação aos algoritmos atuais, uma chave que tenha tamanho de uma imagem, mesmo que considerada pequena, geralmente terá tamanho em *Kilo-Bytes*.

Como exemplo, podemos citar a Figura 21, que possui dimensões 256x256, em escala de cinza, salva em jpeg utiliza 8KB.

Porém, ao tentarmos diminuir o tamanho da imagem, diminuímos o número de coordenadas possíveis para cada caractere, diminuindo assim a aleatoriedade do arquivo criptografado. Apesar disso, ao diminuirmos a imagem-chave, dependendo do número de coordenadas que representem o valor desejado e da aleatoriedade do algoritmo, podemos ainda ter um grande número de possíveis vetores encriptados.

Figura 24: Imagem de dimensões 40x64 com distribuição uniforme de intensidade de pixels.

Suponhamos que desejamos encriptar uma mensagem aleatória com 10 caracteres utilizando a Figura 24.

A Figura 24 possui uma distribuição uniforme de intensidade de pixel igual a 10, ou seja, cada valor de intensidade de pixel ocorre 10 vezes na imagem, logo, se a mensagem é aleatória temos:

$$N_E = \prod_{i=0}^{9} 10 = 1\ x\ 10^{10}\ mensagens$$

Ou seja, mesmo uma imagem com tamanho reduzido ainda pode gerar um grande número de arquivos encriptados diferentes.

Quanto ao tamanho da image-chave, a Figura 24, codificada em jpeg, o tamanho é de 809 bytes, que comparada ao tamanho de uma chave do processo de criptografia de Chave Pública, utiliza uma chave de 1024 bits. O tamanho da Figura 24 é 6,32 vezes maior que uma chave utilizada no algoritmo RSA e 50,56 vezes maior que uma chave utilizada no algoritmo AES.

4.2.4. Falta de Coordenadas para Representar Valores

Caso o Algoritmo 3 encontre um valor na mensagem que não possua um par de coordenadas que represente um pixel de mesmo valor de intensidade na imagem, ele irá parar de funcionar. Isso pode acontecer em diversas situações, mesmo com imagens como a Figura 25.

Figura 25: Imagem de teste *bridge*.

O histograma da Figura 25 é mostrado na Figura 26.

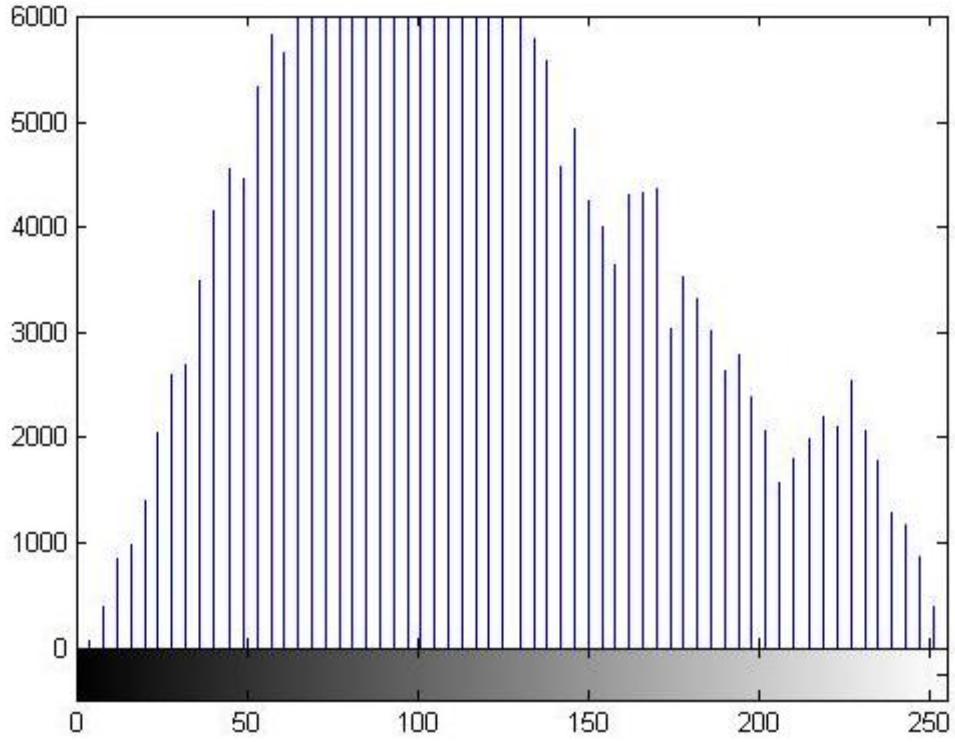


Figura 26: Histograma da Figura 25.

Como pode ser visto, apesar de haver valores de pixels em toda a faixa de valores, existem espaços vazios entre cada feixe, que não possuem representações valores na imagem.

A solução proposta para esse problema é a inserção de *flags* e informações de distância para o pixel de intensidade mais próxima do desejado existente. A inserção dessas *flags* e informações irão funcionar da seguinte forma:

- Ao iniciar o algoritmo, será verificado se existe pelo menos um par de coordenadas para representar o caractere, caso haja o algoritmo inicia com valor 0, caso não haja o algoritmo irá iniciar com valor 1.
- Caso um par de coordenada tenha sido encontrado, será iniciado o algoritmo com valor 0 e será feito uma nova verificação para o próximo pixel, caso haja um par correspondente, somente adiciona ao texto cifrado. Isso deve ocorrer até encontrar um pixel que não pode ser representado por um par de coordenadas. Quando isso acontecer deve inserir uma *flag* de parada, que é definida como a maior dimensão da imagem + 1.
- Ao inserir uma *flag* de parada, será inserida uma *flag* 1 para sinalizar que não foi encontrada um par de coordenadas para o valor do caractere. Em seguida, o algoritmo deve procurar o próximo valor mais próximo que pode ser representado,

dentro do intervalo [0,255]. Após encontrar o par de coordenada que representa a intensidade de pixel mais próxima, deve ser inserido um sinal para representar se ele é menor ou maior que o valor de intensidade encontrado e quantas unidades ele é maior ou menor. Por fim, novamente a *flag* de parada, o valor 0 para representar a coordenada encontrada e o par de coordenadas.

Como exemplo, vamos supor que desejamos encriptar a seqüência [a d h], que são equivalente à [97 100 104] em valor numérico. Vamos supor também que não exista nenhum pixel na imagem-chave que possua valor de intensidade igual no intervalo [103,107]. Supondo pares de coordenadas quaisquer, o algoritmo irá encriptar a mensagem da seguinte forma (as *flags* utilizadas estão em vermelho):

Vetor de Entrada = [a d h]			
Representação tipo char	a	d	h
Representação tipo inteiro	97	100	104
Representação coordenadas	[243 412]	[283 398]	?
Vetor de Saída = [0 243 412 283 398 513 1 45 2 513 0 274 117]			

Como pode ser visto, o algoritmo inseriu a *flag* 0 e começou a preencher a mensagem cifrada até encontrar o valor que não possuía representação. Nesse momento foi inserida a *flag* de parada (considerando uma imagem de dimensões 512x512, a *flag* será 513), o valor 1 para representar que não foi encontrado o valor desejado.

Foi selecionado valor 45 para representar que o pixel mais próximo encontrado é menor que o atual e 43 para representar que o pixel mais próximo é maior que o atual (esses valores foram escolhidos por representar o sinal aritmético "-" e "+" na tabela ASCII, respectivamente). Inserido o valor 45 para representar que o pixel é menor, inseri-se o quanto ele é menor (2 unidades no exemplo, levando o mais próximo pixel encontrado foi o 102).

Por fim, a *flag* de parada, o valor 0 e as coordenadas do pixel mais próximo encontrado. Caso o pixel sem representação fosse o primeiro da sequência, o vetor de saída ficaria da seguinte forma (com as *flags* em vermelho):

Vetor de Saída = [1 45 2 513 0 274 117 243 412 283 398]

O novo processo de encriptação é mostrado no Algoritmo 5.

Encriptação com procura de pixels próximos

1. Ler mensagem original *msg* de um arquivo
2. Ler imagem-chave *img*
3. Calcule o número de bits *n* máximos necessários para representar as coordenadas de *img*

4. Calcule *flag* de parada *escape* como a maior dimensão da imagem + 1
5. Inicia mensagem cifrada *msgCript* nula com inteiro sem sinal de precisão *n*
6. **Para** cada caractere *c* na *msg* **faça**
    a. Retorne uma lista *list* de par de coordenadas *(x,y)* onde *img(x,y)* = *c*
    b. **Se** existir o par coordenadas
        i. Escolha aleatoriamente uma posição *k* na *list*
        ii. *encryptedChar* = par de coordenadas de *list(k)*
    c. **Senão**
        i. Retorne o vetor contendo *temp* = [1 (43/45) *dist escape* 0]
        ii. Retorne uma lista *list* de par de coordenadas *(x,y)* onde *img(x,y)* = *c (+/-) dist*
        iii. Escolha aleatoriamente uma posição *k* na *list*
        iv. *encryptedChar* = *temp* concatenado com *list(k)*
    d. FimSe
    e. **Se** o tamanho de *encryptedChar* == 2 e *msgCript* está vazia
        i. *msgCript* = 0 concatenado com ele mesmo.
    f. FimSe
    g. **Se** o tamanho de *encryptedChar* > 2 e *msgCript* não está vazia
        i. *msgCript* = *escape* concatenado com *encryptedChar*
    h. FimSe
    i. Adicione ao fim de *msgCript* a variável *encryptedChar*
7. **Fim para**
8. Salve *msgCript* em um arquivo utilizando *n* bits de precisão

Algoritmo 5: Encriptação com procura de pixels próximos.

O processo de decriptação deve ser atualizado também para entender a nova forma de encriptação, como mostrado no Algoritmo 6:

Decriptação com procura de pixels próximos

1. Ler imagem-chave *img*
2. Calcule o número de bits *n* máximos necessários para representar as coordenadas de *img*
3. Ler mensagem cifrada *msgCript* utilizando *n*-bits para cada coordenada
4. Calcule *flag* de parada *escape* como a maior dimensão da imagem + 1
5. Inicia mensagem decriptada *msgDecript* nula
6. Inicia *hasOp* como *false*
7. Inicia i = 1
8. **Enquanto** i < tamanho da mensagem encriptada **faça**
    a. **Se** msgCript(i) == 0
        i. i = i + 1
        ii. **Enquanto** *msgCript*(i) ≠ *escape* **faça**
            1. *x* = *msgCript*(i)
            2. *y* = *msgCript*(i+1)
            3. *decryptedChar* = valor do pixel *p* na coordenada *img(x,y)*
            4. **Se** *hasOp == true*
                a. *decryptedChar = decryptedChar + corr*
                b. hasOp = false

5. **FimSe**
6. *msgDecript* = *decryptedChar* concatenado com *msgDecript*
7. i = i + 2
8. **Se** i > tamanho de *msgCript*
    a. *Break*
9. **FimSe**

b. **SenãoSe** msgCript(i) == 1
    i. i = i + 1
    ii. *hasOp* = *true*
    iii. **Se** *msgCript*(i) == 43
        1. *corr* = msgCript(i+1)*(-1)
    iv. **SenãoSe** *msgCript*(i) == 45
        1. *corr* = msgCript(i+1)
    v. **FimSe**
    vi. i = i + 2
c. **FimSe**
d. i = i + 1

9. **FimEnquanto**
10. Salve *msgDecript* em um arquivo

Algoritmo 6: Decriptação com procura de pixels próximos.

Outra vantagem de inserir *flags* no algoritmo é que, aliado ao fato de o número de bits utilizados para salvar cada coordenada varia, dá ao intruso infinitas combinações de possíveis tamanhos de imagens-chave, dificultando a quebra da mensagem cifrada.

Além disso, o sistema de *flags* utilizados foi pensado de forma a facilitar um processo de compressão chamado compressão aritmética, que será mais bem comentado nos trabalhos futuros.

5 EXPERIMENTOS E RESULTADOS

Todos os algoritmos apresentados nesse capítulo foram implementados utilizando o programa MATLAB®. O principal motivo da utilização da ferramenta MATLAB® deve-se ao fato de, apesar de ser mais lento que implementações em linguagens como C ou C++, facilita o trabalho com imagens dos mais diversos formatos, pois o mesmo faz o *parser* dos cabeçalhos automaticamente para o usuário.

O MATLAB® (*Matrix Laboratory*) é uma linguagem de alto nível e ambiente de desenvolvimento interativo de alto desempenho voltado para o cálculo numérico. Ele foi desenvolvido para trabalhar com matriz, o que facilita o desenvolvimento de aplicações que trabalhem com imagens.

Os experimentos a seguir foram divididos em 3 grupos. No primeiro grupo de experimentos, foram geradas 50 mensagens randômicas de tamanho variável entre 9000 e 10000 caracteres para serem encriptadas, com valores entre 0 e 255. No segundo grupo de experimentos, foi utilizado um texto real, o texto Gênesis da Bíblia Cristã. O texto possui 187071 caracteres, entre o ASCII-10 e ASCII-252. No último grupo foram feitos testes comparativos em o método proposto e os algoritmos AES e RSA.

No primeiro e segundo grupo de experimento foram utilizadas 10 imagens de um banco de imagem público. As imagens utilizadas podem ser vistas no anexo B. Para os testes foi utilizado um computador com Intel® Core 2 Duo 2.20 GHz e 4GB de memória RAM, Windows 7 Home Premium 32 bits.

As especificações das imagens podem ser vistas na Tabela 5.

Imagem-chave (formato)	Dimensões (*pixels*)	Tamanho (*bytes*)
Primeira (aerial.pgm)	512x512	257K
Segunda (boats.pgm)	576x720	406K
Terceira (bridge.pgm)	512x512	257K
Quarta (D108.pgm)	640x640	401K
Quinta (f16.pgm)	512x512	257K
Sexta (girl.pgm)	576x720	406K
Sétima (Lena.jpg)	512x512	43K
Oitava (peppers.pgm)	512x512	257K
Nona (pp1209.pgm)	512x512	257K
Décima (zelda.pgm)	576x720	406K

Tabela 5: Especificações das imagens utilizadas para teste.

5.1 Grupo de Experimento 1

Os gráficos a serem vistos estão divididos em uma matriz 4x2, onde cada célula da matriz representa um resultado:

- Célula (1x1): Tamanho da mensagem original.
- Célula (1x2): Tamanho da mensagem encriptada.
- Célula (2x1): Tamanho da mensagem decriptada (deve ser igual à célula (1x1)).
- Célula (2x2): Razão entre o tamanho da mensagem encriptada e a mensagem original.
- Célula (3x1): Entropia da mensagem encriptada. É um número que representa o número de bits teórico ideal para representar a informação, em comparação aos 9 bits utilizados nos testes, devido ao tamanho da imagem.
- Célula (3x2): Erro de decriptação. Deve ser uma linha constante em zero.
- Célula (4x1): Tempo de encriptação.
- Célula (4x2): Tempo de decriptação.

A seguir os resultados obtidos. Vale notar que para todos os testes realizados, não houve erro de decriptação.

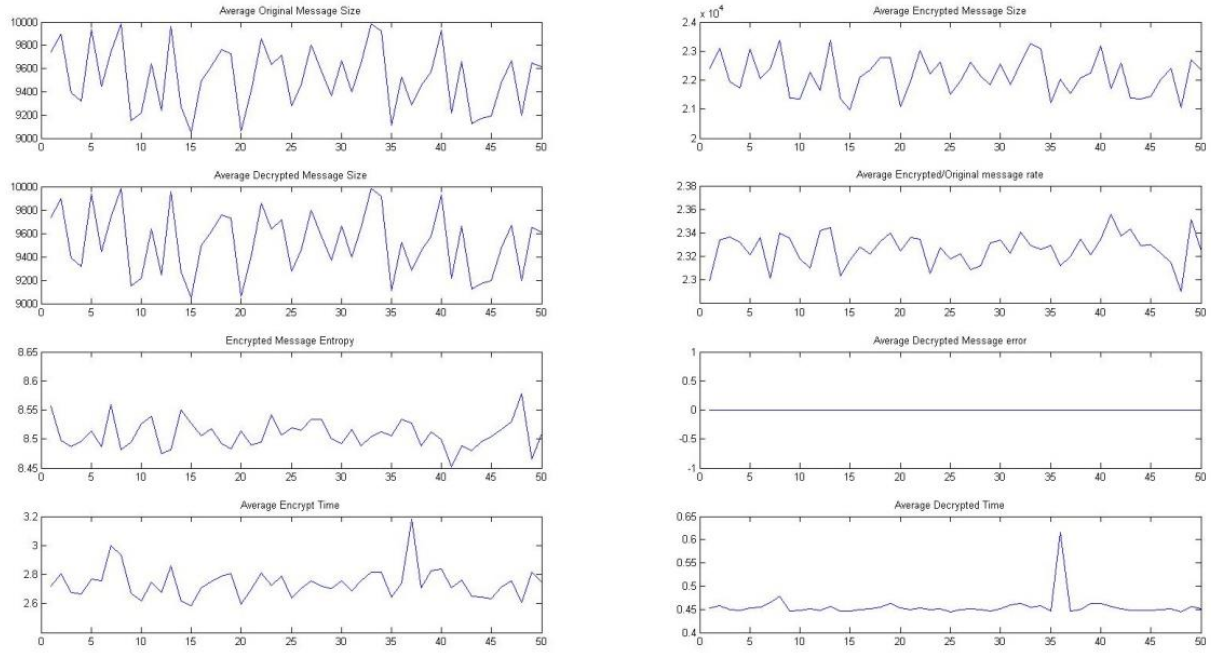


Figura 27: Resultados obtidos com a primeira imagem de teste.

Os resultados obtidos com a primeira imagem de teste (aerial.pgm), são mostrados na Figura 27. A razão entre a mensagem encriptada e a mensagem original ficou em média em 2.33. O tempo de encriptação médio ficou em 2.8 segundos e o tempo de decriptação ficou em

0.5 segundos. O gráfico mostra também que a entropia média é de 8.5 bits e estão sendo utilizados 9 bits para representar um número, ou seja, próximo do ideal.

Pela análise dos resultados, pode-se concluir que o tempo de encriptação e decriptação estão aceitáveis.

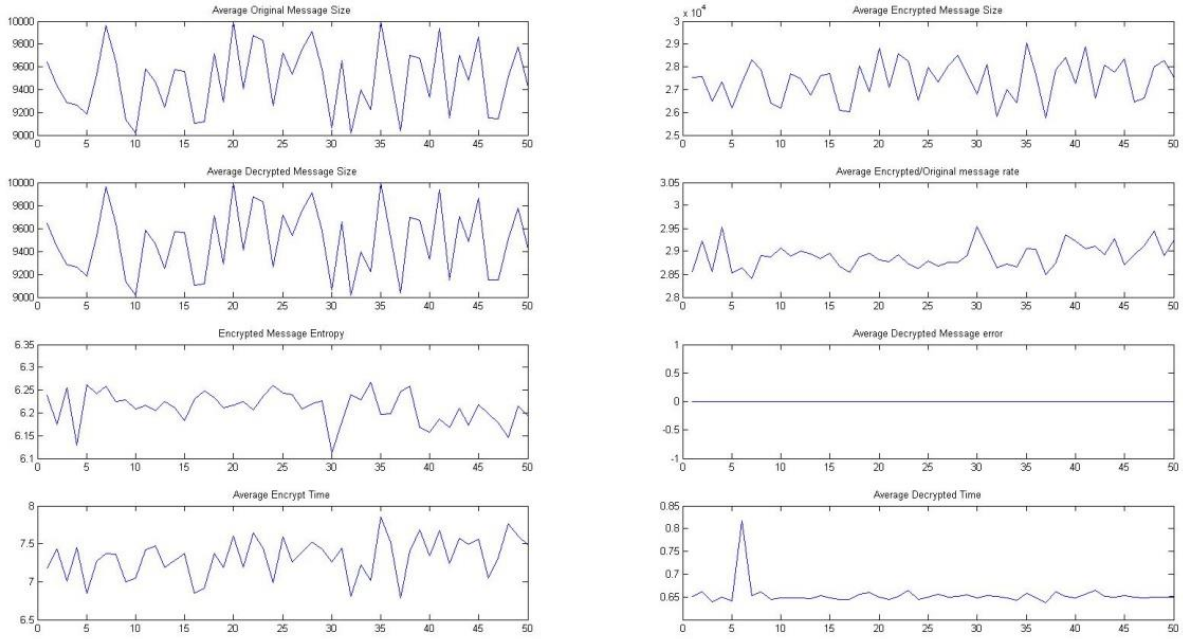


Figura 28: Resultados obtidos com a segunda imagem de teste.

Os resultados obtidos com a segunda imagem de teste (boats.pgm) são mostrados na Figura 28. A razão entre a mensagem encriptada e a mensagem original ficou em média em 2.9. O tempo de encriptação médio ficou em 7.5 segundos e o tempo de decriptação ficou em 0.7 segundos. O gráfico mostra também que a entropia média é de 6.15 bits e estão sendo utilizados 9 bits para representar um número.

Nota-se um aumento no tempo de encriptação, que se deve principalmente ao fato de que a Figura 42 (Anexo B) não possui valores de intensidade de *pixel* acima de 218, como pode ser visto na Figura 43(Anexo B). Isso leva o algoritmo a procurar diversas vezes pelo pixel mais próximo representável.

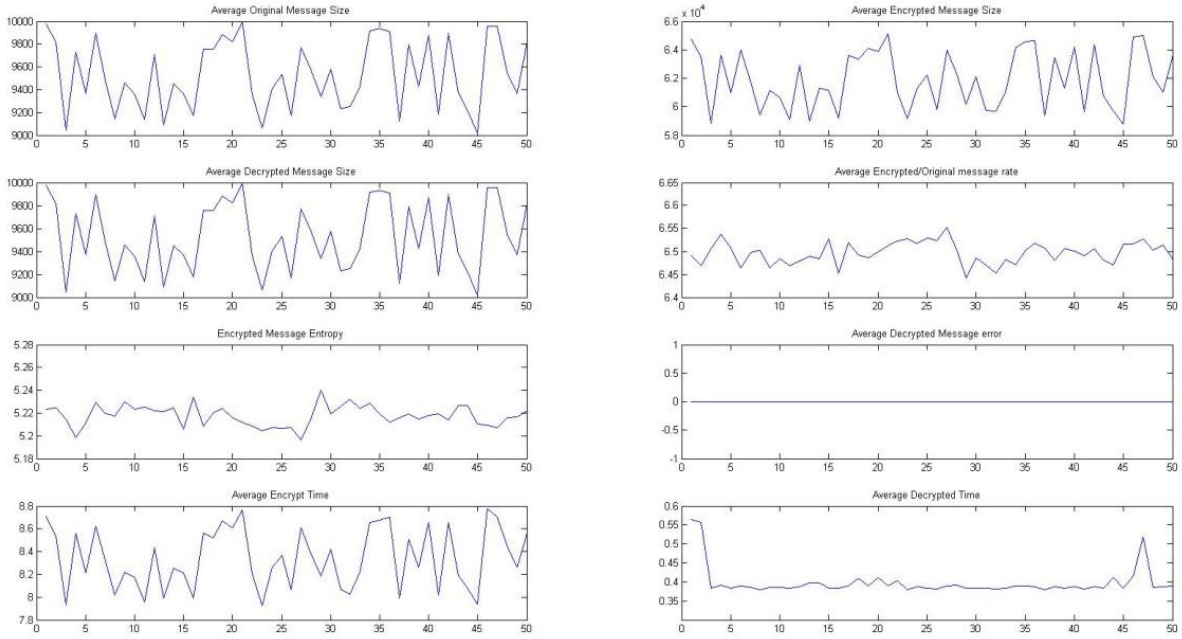


Figura 29: Resultados obtidos com a terceira imagem de teste.

Os resultados obtidos a partir da terceira imagem de teste (bridge.pgm) são mostrados na Figura 29. A razão entre a mensagem encriptada e a mensagem original ficou em média em 6.5. O tempo de encriptação médio ficou em 8.3 segundos e o tempo de decriptação ficou em 0.45 segundos. O gráfico nos mostra também que a entropia média é de 5.22 bits e estão sendo utilizados 9 bits para representar um número.

Nota-se um aumento do tempo de encriptação e do tamanho do arquivo encriptado. Isso se deve ao fato de que o histograma mostrado na Figura 45 (Anexo B) mostra linhas espaçados, ou seja, na maior parte do tempo o algoritmo deve procurar o pixel mais próximo representável, o que aumenta bastante o tamanho do arquivo, devido à inserção de *flags*. Essa repetição diminui a entropia da mensagem cifrada.

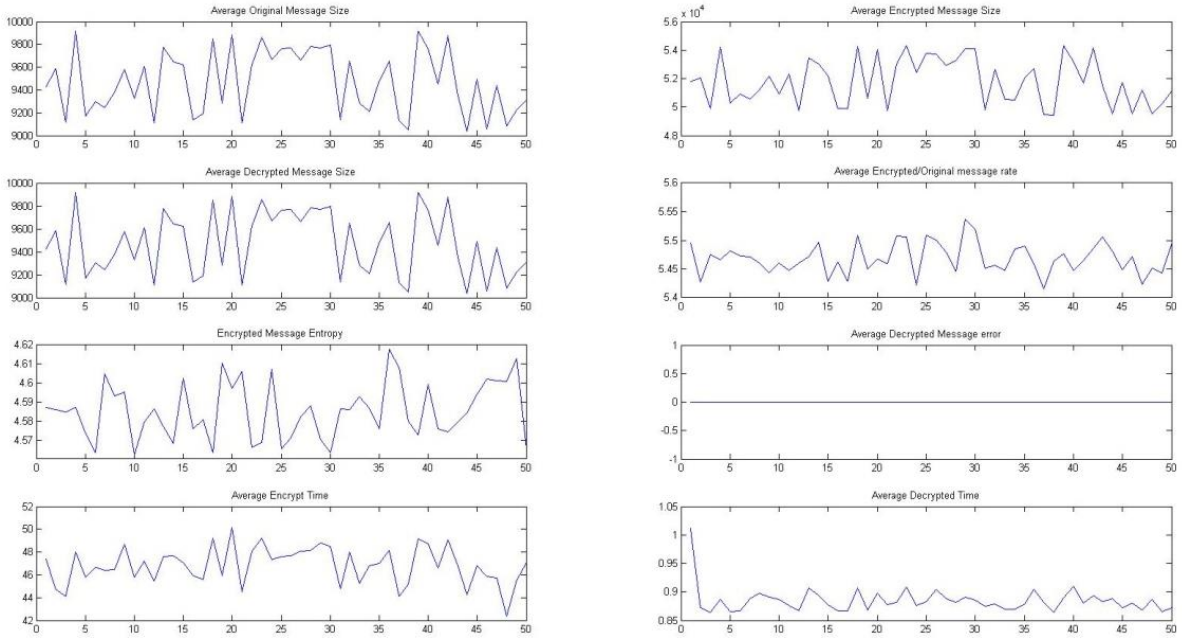


Figura 30: Resultados obtidos com a quarta imagem de teste.

Os resultados obtidos com a da quarta imagem de teste (D108.pgm), são mostrados na Figura 30. A razão entre a mensagem encriptada e a mensagem original ficou em média em 5.45. O tempo de encriptação médio ficou em 46 segundos e o tempo de decriptação ficou em 0.85 segundos. O gráfico nos mostra também que a entropia média é de 4.59 bits e estão sendo utilizados 9 bits para representar um número.

Nota-se um tempo de encriptação maior, assim como aumento do arquivo encriptado e baixa entropia. Isso pode ser explicado analisando o histograma da imagem-chave, mostrado na Figura 47 (Anexo B). Assim como a imagem anterior, o histograma desta imagem possui muitas lacunas, o que torna o algoritmo mais lento, pois este procura por pixels próximos representáveis. Essa repetição diminui a entropia da mensagem cifrada.

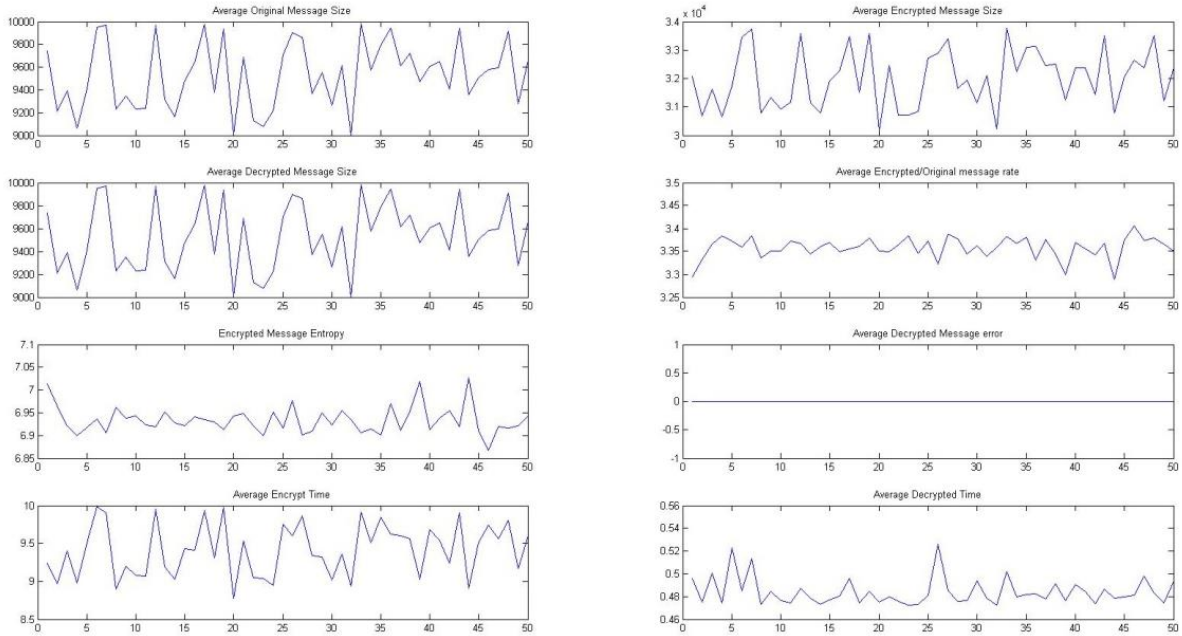


Figura 31: Resultados obtidos com a quinta imagem de teste.

Os resultados obtidos com a quinta imagem de teste (f16.pgm) são mostrados na Figura 31. A razão entre a mensagem encriptada e a mensagem original ficou em média em 3.35. O tempo de encriptação médio ficou em 9.5 segundos e o tempo de decriptação ficou em 0.48 segundos. O gráfico nos mostra também que a entropia média é de 6.95 bits e estão sendo utilizados 9 bits para representar um número, ou seja, próximo do ideal.

Notam-se bons resultados na criptografia utilizando a f16.pgm. O tempo de encriptação é maior, possivelmente devido ao fato de não haver valores de intensidade de pixel menor que 35 e maior que 228.

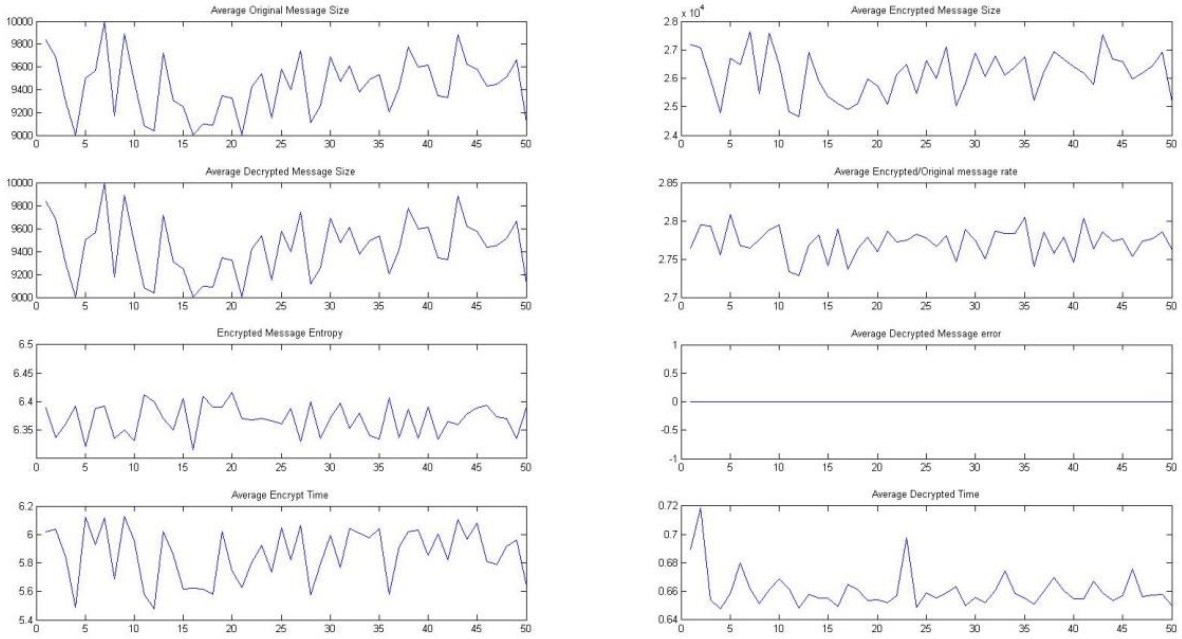


Figura 32: Resultados obtidos com a sexta imagem de teste.

Os resultados obtidos com a sexta imagem de teste (girl.pgm), são mostrados na Figura 32. A razão entre a mensagem encriptada e a mensagem original ficou em média em 2.75. O tempo de encriptação médio ficou em 5.8 segundos e o tempo de decriptação ficou em 0.66 segundos. O gráfico nos mostra também que a entropia média é de 6.35 bits e estão sendo utilizados 9 bits para representar um número, ou seja, próximo do ideal.

Resultados aceitáveis utilizando na criptografia a girl.pgm. O tempo de encriptação é um pouco elevado, pois a imagem não possui valores de intensidade de pixel maiores que 220.

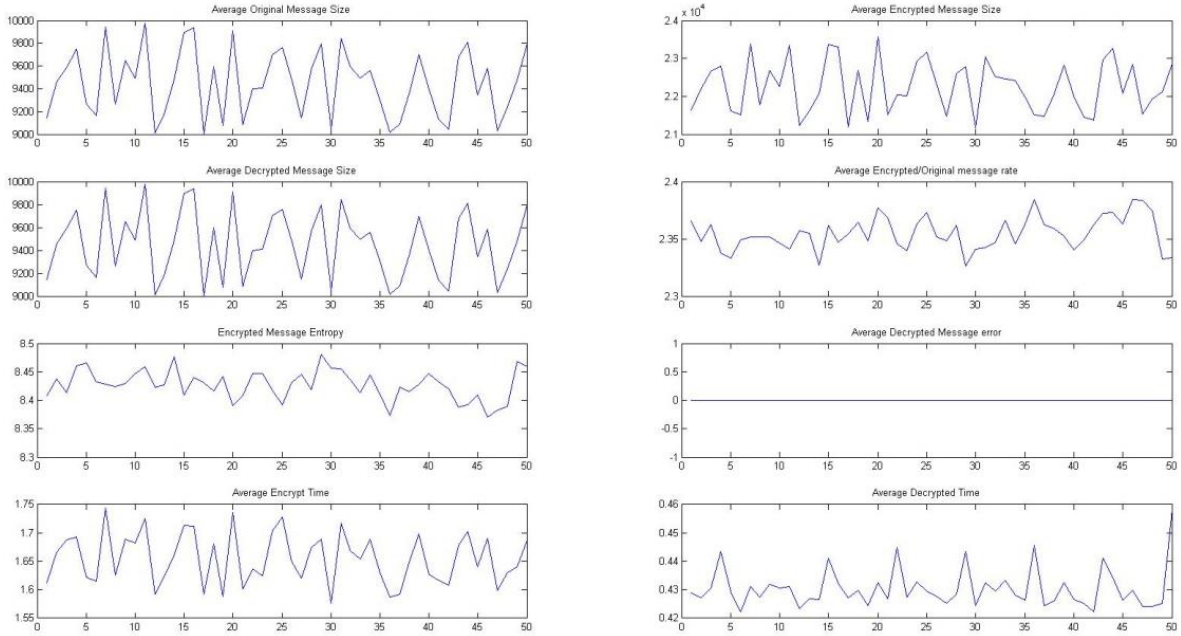


Figura 33: Resultados obtidos com a sétima imagem de teste.

Os resultados obtidos com a sétima imagem de teste (lena.jpg), são mostrados na Figura 33. A razão entre a mensagem encriptada e a mensagem original ficou em média em 2.35. O tempo de encriptação médio ficou em 1.65 segundos e o tempo de decriptação ficou em 0.43 segundos. O gráfico nos mostra também que a entropia média é de 8.45 bits e estão sendo utilizados 9 bits para representar um número, ou seja, próximo do ideal.

Resultados aceitáveis utilizando na criptografia a lenga.jpg. Apesar da imagem não possuir valores de intensidade de pixel acima de 235, não houve grande impacto no tempo de encriptação. Para esse conjunto de imagens, essa imagem em particular apresentou os melhores resultados obtidos, possivelmente por ter valores de intensidade de pixel em quase toda faixa entre 0 e 255.

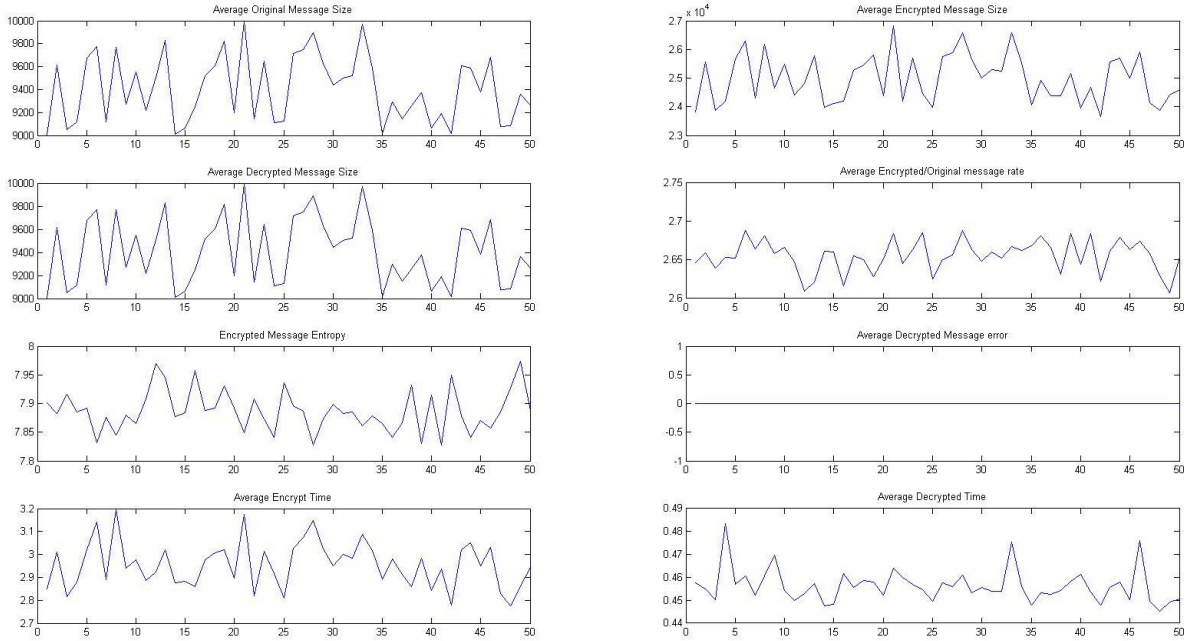


Figura 34: Resultados obtidos com a oitava imagem de teste.

Os resultados obtidos com a oitava imagem de teste (peppers.pgm), são mostrados na Figura 34. A razão entre a mensagem encriptada e a mensagem original ficou em média em 2.65. O tempo de encriptação médio ficou em 3.0 segundos e o tempo de decriptação ficou em 0.46 segundos. O gráfico nos mostra também que a entropia média é de 7.9 bits e estão sendo utilizados 9 bits para representar um número, ou seja, próximo do ideal.

Resultados aceitáveis utilizando na criptografia a Figura 54 (Anexo B). O tempo de encriptação é um pouco elevado, pois a imagem não possui valores de intensidade de pixel maiores que 224.

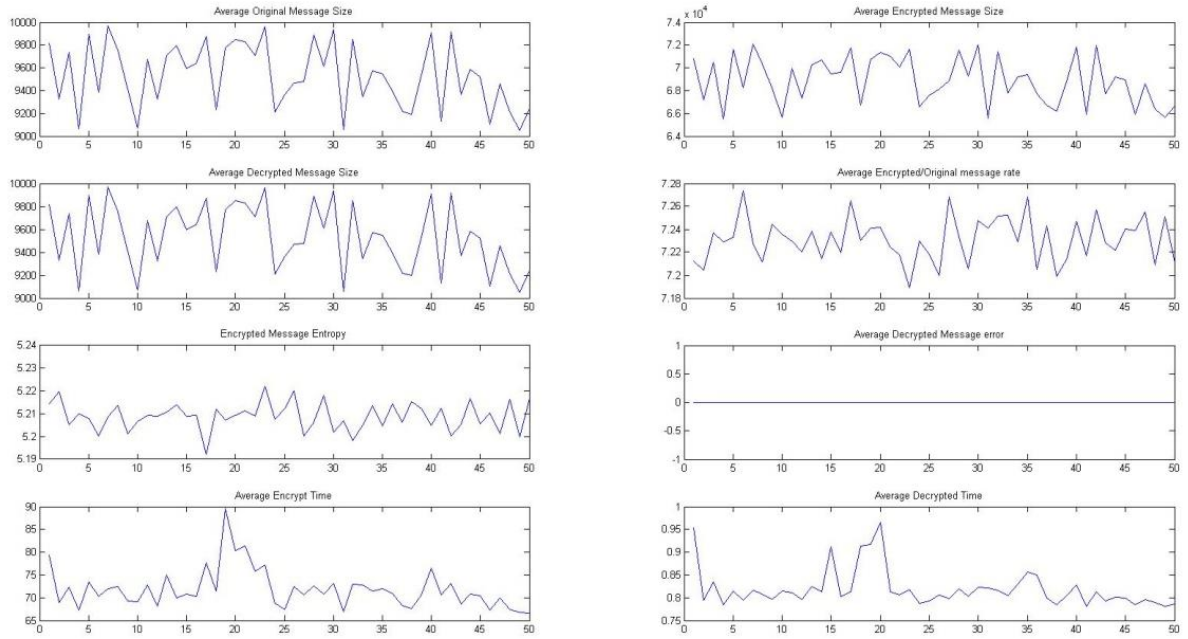


Figura 35: Resultados obtidos com a nona imagem de teste.

Os resultados obtidos com a nona imagem de teste (pp1209.pgm), são mostrados na Figura 35. A razão entre a mensagem encriptada e a mensagem original ficou em média em 7.22. O tempo de encriptação médio ficou em 75 segundos e o tempo de decriptação ficou em 0.85 segundos. O gráfico nos mostra também que a entropia média é de 5.21 bits e estão sendo utilizados 9 bits para representar um número.

Nota-se um aumento no tempo de encriptação, assim como o tamanho do arquivo encriptado. Isso pode ser explicado analisando a Figura 57 (Anexo B). O histograma possui poucos feixes e que são bastante espaçados, ou seja, na maior parte do tempo o algoritmo está procurando o pixel mais próximo representável.

Essa alta repetição de níveis de cinza diminui a entropia da mensagem cifrada, acarretando vantagens e desvantagens. A vantagem é que uma compressão de dados vai ser mais efetiva nessa mensagem, porém, por mais efetiva que a compressão seja,dificilmente a mensagem comprimida será próximo do ideal.

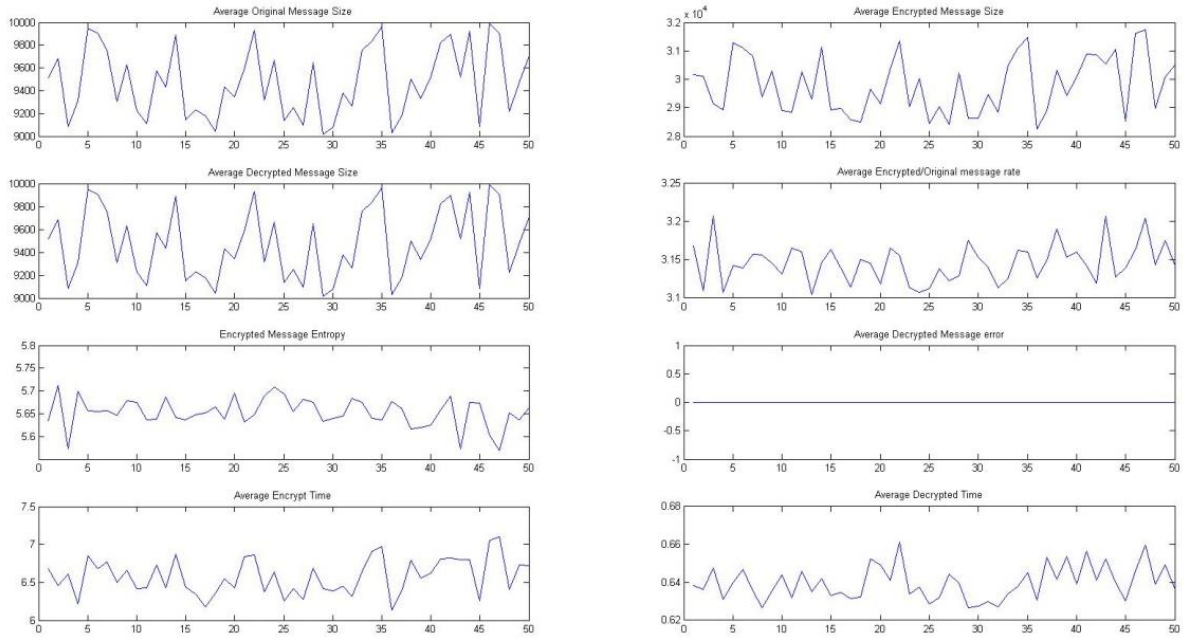


Figura 36: Resultados obtidos com a décima imagem de teste.

Os resultados obtidos com a décima imagem de teste (zelda.pgm), são mostrados na Figura 36. A razão entre a mensagem encriptada e a mensagem original ficou em média em 3.15. O tempo de encriptação médio ficou em 6.5 segundos e o tempo de decriptação ficou em 0.64 segundos. O gráfico mostra também que a entropia média é de 5.65 bits e estão sendo utilizados 9 bits para representar um número.

Nota-se um aumento no tempo de encriptação, possivelmente devido ao fato da imagem não apresentar valores de intensidade de pixel maior do que 205. Nota-se também

que, apesar de que o tamanho da mensagem cifrada ser relativamente pequeno, sua entropia é baixa, comparada às mensagens cifradas do mesmo tamanho.

5.2 Grupo de Experimento 2

No grupo de experimento 2, foram utilizadas as mesmas imagens-chave para encriptar uma mensagem. A diferença é que a mensagem a ser encriptada é um texto em português, e não randômico. Essa diferença aparece no histograma da mensagem original, pois ao invés de possuir um histograma que tende à uniformidade, como o randômico, as freqüências de certas letras na língua portuguesa tendem a ser maior que outras. Além disso, em um texto real, os caracteres não imprimíveis não aparecem no texto, com exceção do ASCII-10 e ASCII-11 (*line feed* e *carriage return*, respectivamente). O texto utilizado foi o capítulo Gênesis da Bíblia Cristã e seu histograma pode ser visto na Figura 37.

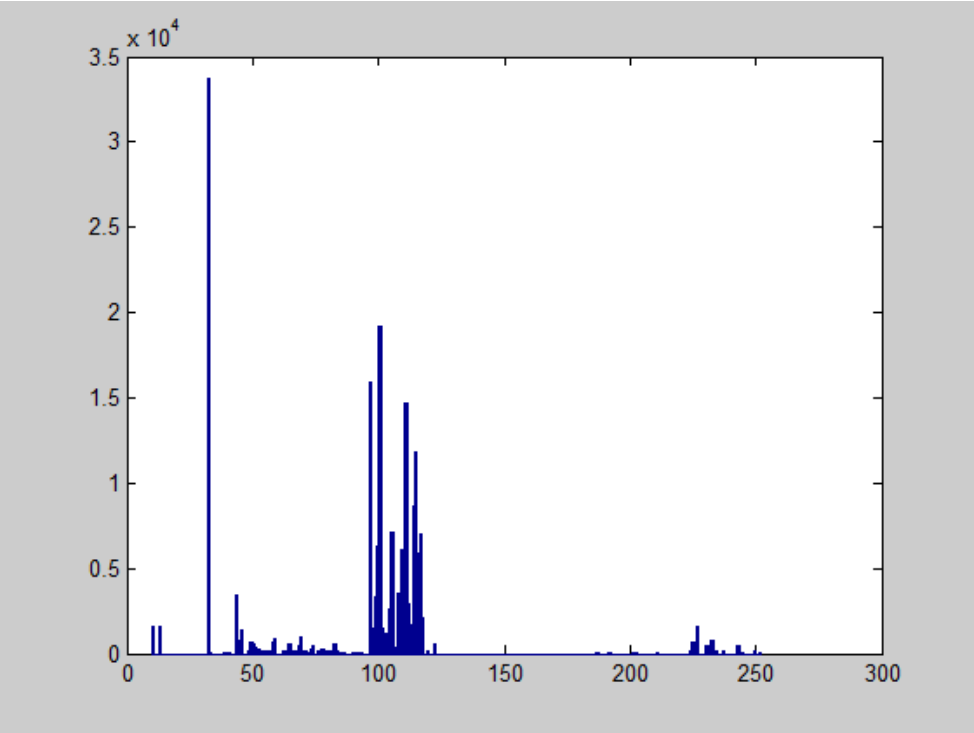


Figura 37: Histograma do capítulo Gênesis da Bíblia Cristã.

Como podemos notar na Figura 37, o caractere que mais se repete é o ASCII-32, o espaço entre palavras. Em seguida, notamos uma concentração na área entre 97 e 122, que são as letras de “a” a “z”, minúsculas. Outros caracteres também aparecem, mas com menor freqüência do que os citados. A quantidade de caracteres contidos no Gênesis é de 187071. Os resultados da encriptação com cada imagem do Gênesis podem ser visto na Tabela 6.

Imagem-chave	Tamanho da mensagem encriptada (*bytes*)	Razão mensagem original/ encriptada	Tempo de encriptação (s)	Tempo de decriptação (s)	Tempo Total (s)
aerial	383941	2.05	78.80	8.21	87.83
boats	402739	2.15	112.06	8.58	121.49
bridge	913517	4.88	317.86	11.21	331.30
D108	698795	3.73	763.47	10.26	775.47
f16	610279	3.26	199.06	9.43	210.02
girl	401689	2.14	100.93	8.51	110.29
lena	378583	2.02	62.89	9.33	73.03
peppers	388057	2.07	68.94	8.35	78.14
pp1209	1251431	6.98	1418.35	14.34	1435.70
zelda	402757	2.15	90.34	8.49	99.68

Tabela 6: Resultados obtidos na encriptação do Gênesis.

Para um texto real, que possui freqüências diferentes para cada caractere, um alto número de espaços entre palavras e poucos caracteres de controle, a encriptação utilizando as mesmas imagens-chave do grupo de experimento anterior, mostrou resultados bem semelhantes.

Notamos que sempre há um aumento do tamanho da mensagem encriptada em relação à mensagem original de, no mínimo, duas vezes. Nos piores casos, onde o histograma da imagem mostra muitos espaços vazios, ou seja, a imagem não possui certos valores de intensidade de pixel, são gerados dois problemas: o primeiro é que o algoritmo demora muito para encriptar, pois na maioria das vezes ele precisa procurar o pixel mais próximo representável. O segundo problema é que, pelo fato do algoritmo inserir *flags* e informações para representar o valor de intensidade de pixel faltante, o tamanho da mensagem encriptada aumenta bastante. Como pode ser visto na Tabela 6, o arquivo encriptado aumentou de 183KB para 1.49MB, quase 7 vezes o tamanho original. Além disso, o tempo de encriptação foi alto, comparado às outras imagens, demorando quase 24min para encriptar toda a informação.

5.3 Grupo de Experimento 3

No grupo de experimento 3 foram feitos testes comparativos entre o método proposto de encriptação de dados e o algoritmo AES e RSA, de criptografia de chave simétrica e criptografia de chave assimétrica, respectivamente. Os algoritmos utilizados para encriptação AES e RSA estão disponíveis online, em [15] e [16], respectivamente.

Para os experimentos, a mensagem utilizada foi o Gênesis da Bíblia Cristã, assim como no grupo de experimentos 2. Para a comparação de resultados, foi escolhido o melhor resultado do método proposto, encriptação com a sétima imagem-chave (Figura 52).

Os seguintes resultados foram obtidos:

Algoritmo	Tamanho da mensagem encriptada (*bytes*)	Razão mensagem original/ encriptada	Tempo de encriptação (s)	Tempo de decriptação (s)	Tempo Total (s)
AES	187071	1	4.58	5.41	9.66
RSA	187071	1	6.64	4.38	11.37
Método Proposto	378583	2.02	62.89	9.33	73.03

Tabela 7: Comparativo entre o método proposto, AES e RSA.

A grande vantagem dos métodos atuais sobre o proposto sem dúvida é o tempo de encriptação e o tamanho da mensagem encriptada. Os algoritmos são rápidos, pois são constituídos basicamente de cifras de substituição e transposição não-lineares, e não precisam tomar decisões aleatórias sobre tabelas nem procurar por *pixels* próximos representáveis como o algoritmo proposto.

Porém, o método proposto possui uma enorme vantagem sobre os algoritmos comparados. Independente do número de vezes que os algoritmos AES e RSA sejam executados tendo como entrada o texto de teste, o resultado, a mensagem encriptada será sempre a mesma, enquanto utilizando o método proposto, o resultado será diferente para cada encriptação, não afetando o processo de decriptação que irá sempre decriptar a mesma mensagem.

6 DISCUSSÃO E CONCLUSÕES

O objetivo deste trabalho foi desenvolver um método de encriptação que utiliza uma imagem como chave para encriptação e avaliar seu desempenho frente aos principais algoritmos atuais de encriptação. Para alcançar esse objetivo foi desenvolvido um algoritmo que se aproveita da natureza aleatória dos valores de intensidade de pixels na imagem para encriptar uma mensagem.

Foram utilizadas 10 imagens para testes, cada uma com características diferentes. Umas possuem valores de níveis de cinza distribuídos em quase toda faixa entre 0 e 255 (imagens com alto contraste) e outras que mostravam poucos picos espaçados no histograma (vide anexo B).

Foram feitos testes utilizando tanto mensagens aleatórias geradas pelo programa de testes, que gerava mensagens com valores entre 0 e 255 e mensagens de texto reais, como o Gênesis da Bíblia Cristã. Esses testes foram necessários para validar o método frente aos algoritmos atuais.

Na comparação com os principais algoritmos atuais de criptografia de chave simétrica e assimétrica, AES e RSA, respectivamente, o método proposto mostrou poucos resultados satisfatórios quando levado em consideração o tamanho da mensagem encriptada e o tempo na encriptação. Porém, o método se mostrou mais eficiente na forma de esconder mensagens, pois, diferentemente dos algoritmos atuais que sempre geram a mesma saída para uma entrada, o método proposto gera diferente saídas para uma mesma entrada, aumentando a dificuldade da quebra da mensagem encriptada.

O algoritmo ainda pode ser melhorado para melhorar seus dois principais pontos fracos: tempo de encriptação e tamanho de mensagem encriptada.

Como se pode notar nos resultados, a maior causa da demora da encriptação deve-se ao fato de que, quando o algoritmo encontra um valor que não possui equivalente de valor de intensidade de *pixel* na imagem, ele procura o mais próximo *pixel* representável. Essa procura diminui muito a eficiência do algoritmo. Uma possível solução para esse problema é, antes do processo de encriptação de caracteres iniciar, iniciar também, para os pixels faltantes na imagem, uma tabela dos *pixels* mais próximos, ou uma lista de distâncias para quaisquer pixels da imagem. Dessa forma, ao invés de procurar um valor de pixel próximo representável na imagem, o algoritmo deve apenas ler os valores dessa tabela.

Quanto ao tamanho da mensagem encriptada, pode usar uma forma de compressão. Os testes mostraram que a entropia das mensagens encriptadas, ou seja, o número de bits necessários para representar a informação está acima do ideal. Um compressor por entropia iria diminuir o tamanho da mensagem, aproximando-o do ideal, diminuindo assim o tamanho do arquivo. A própria estrutura de *flags* possibilita a implementação de um compressor aritmético de dois níveis. Isso traz vantagens e desvantagens, a vantagem é que a compressão serviria como uma segurança a mais na mensagem encriptada, pois um intruso não teria acesso direto à mensagem encriptada. A principal desvantagem é o *overhead* computacional necessário para comprimir os dados.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] LIBERTY, J.; HORVATH, D. B. **Sams Teach Yourself C++ for Linux in 21 Days**. 2ª Edição. ed. Indiana: Sams Publishing, 2000.

[2] GOODRICH, M. T.; TAMASSIA, R. **Projeto de Algoritmos**. Porto Alegre: Artmed, 2004.

[3] DEITEL, H. M.; DEITEL, P. J. **C++ How To Program**. 3ª Edição. ed. New Jersey: Prentice-Hall, 2001.

[4] GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 2° Edição. ed. New Jersey: Prentice Hall, 2002.

[5] CAMPOS, A. A. N. **Algoritmo de Criptografia AES em Hardware, Utilizando Dispositivo de Lógica Programável (FPGA) e Linguagem de Descrição de Hardware (VHDL)**. Itajubá - MG: [s.n.], 2008.

[6] TANENBAUM, A. S. **Redes de Computadores**. 3 Edição. ed. Manaus: Campus, 2002.

[7] MENEZES, A.; OORSCHOT, P. V.; VANSTONE, S. **Handbook of Applied Criptography**. 1ª Edição. ed. [S.l.]: CRC Press, 2001.

[8] **Braingle**. Disponivel em: <http://www.braingle.com/>. Acesso em: 2010 Julho 29.

[9] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **Data Encryption Standard**, 25 outubro 1999.

[10] **Electronic Frontier Foundation**. Disponivel em: <http://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_des_faq.html>. Acesso em: 7 agosto 2010.

[11] **COPACOBANA**. Disponivel em: <http://www.copacobana.org/>. Acesso em: 7 agosto 2010.

[12] **SciEngines**. Disponivel em: <http://www.sciengines.com/joomla/index.php>. Acesso em: 7 agosto 2010.

[13] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **Advanced Encryption Standard**, 26 novembro 2001.

[14] SOUZA, R. D. A.; OLIVEIRA, F. B. D. O Padrão de Criptografia Simétrica AES, Petrópolis, 5 outubro 2007.

[15] **Matlab Central**.

Disponivel em: <http://www.mathworks.com/matlabcentral/fileexchange/9083-rapidly-encrypt-and-decrypt-using-rsa>. Acesso em: 2010 agosto 7.

[16] **Matlab Central**.

Disponivel em: <http://www.mathworks.com/matlabcentral/fileexchange/8925-rapid-aes-data-encryption-and-decryption>. Acesso em: 2010 agosto 9.

[17] **ASCII Table - ASCII and Unicode Characters**. Disponivel em: <http://ascii-table.com/>. Acesso em: 2010 agosto 8.

[18] ISO/IEC. **ISO/IEC 8859-1: Latin Alphabet No. 1**, 1997.

ANEXO A

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

Figura 38: Tabela ASCII original.
Fonte: [15]

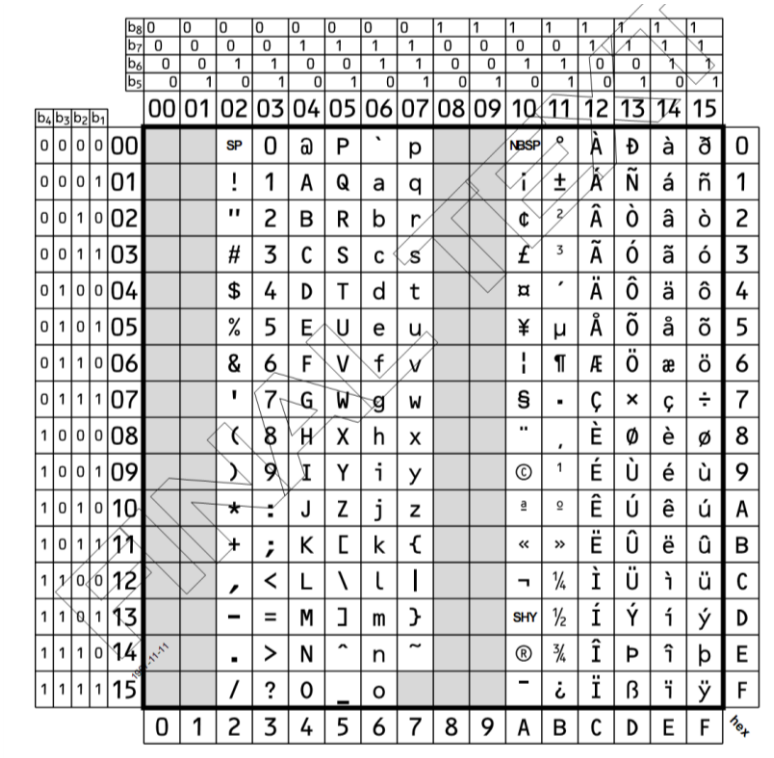


Figura 39: ISO/IEC 8859-1.
Fonte: [16]

ANEXO B

No anexo B estão todas as imagens utilizadas nos experimentos. Assim como seus respectivos histogramas.

Figura 40: Primeira imagem-chave utilizada nos experimentos. (aerial.pgm)

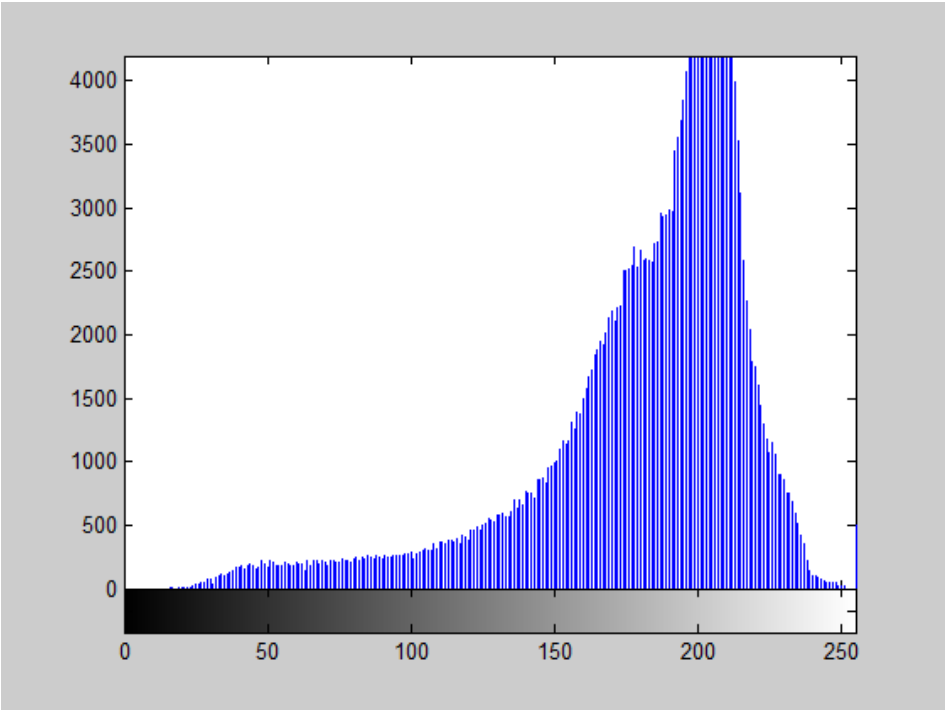


Figura 41: Histograma da imagem aerial.pgm

Figura 42: Segunda imagem-chave utilizada nos experimentos. (boats.pgm)

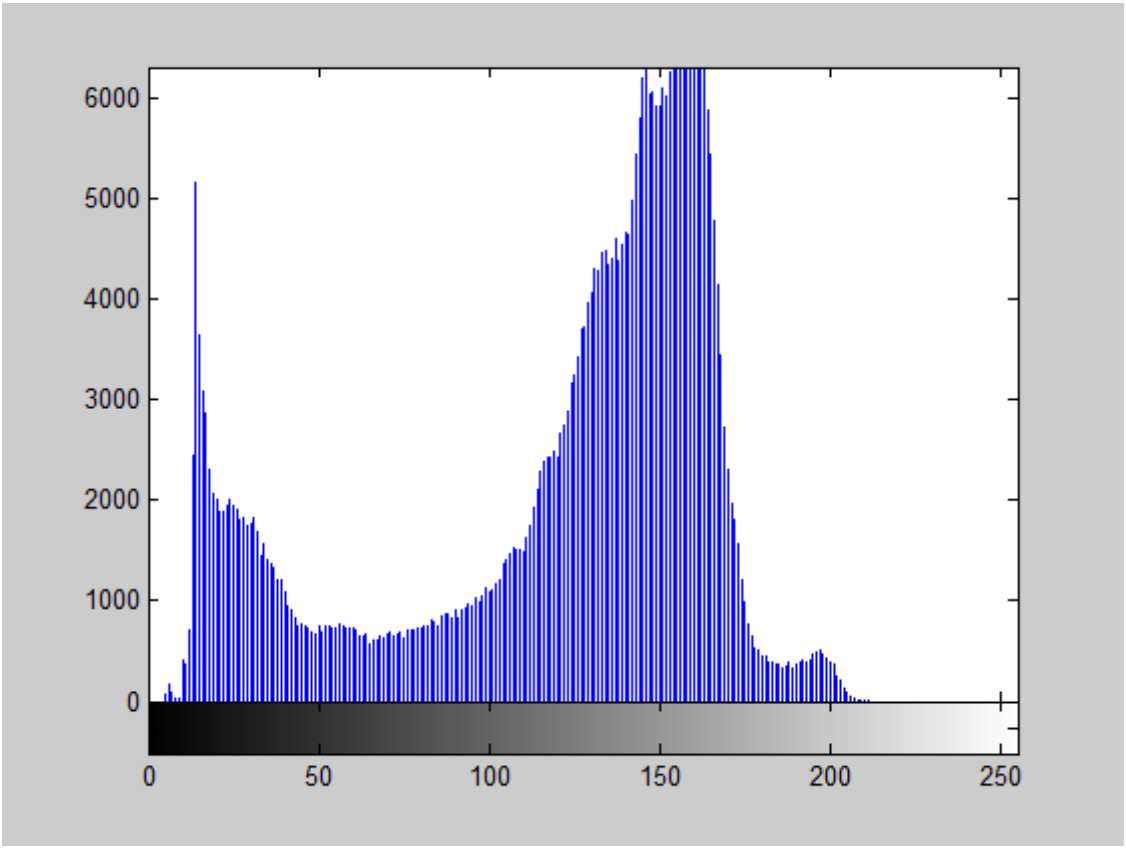


Figura 43: Histograma da imagem boats.pgm

Figura 44: Terceira imagem-chave utilizada nos experimentos. (bridge.pgm)

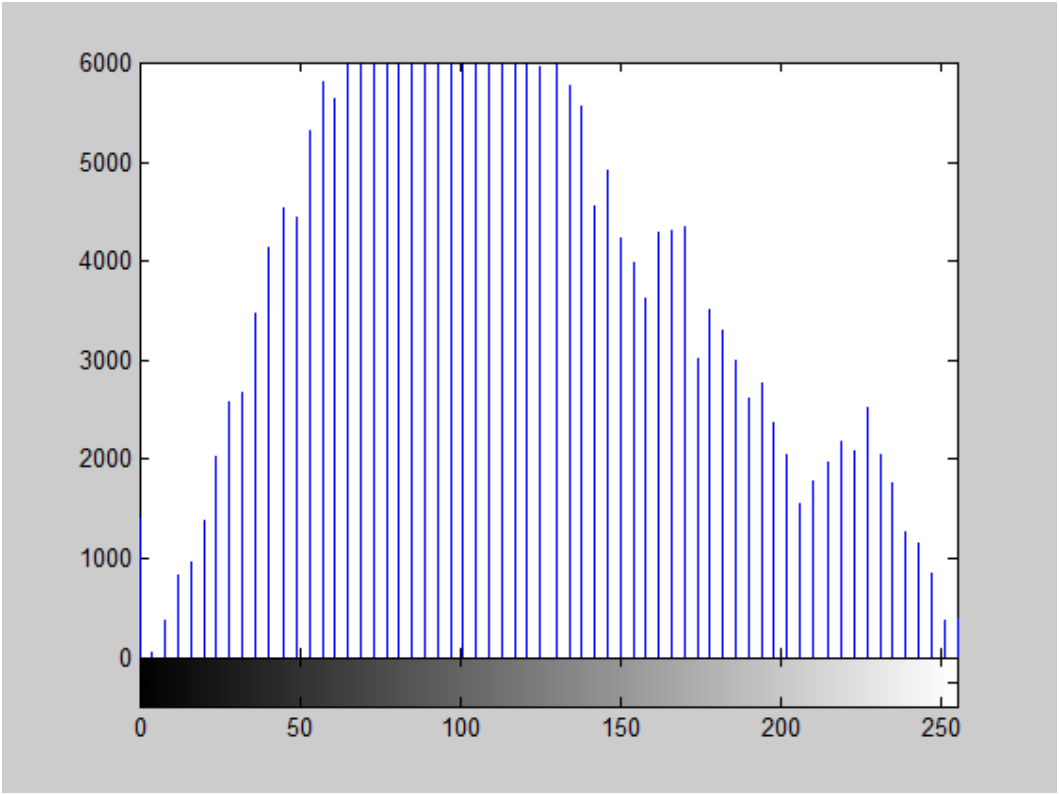


Figura 45: Histograma da imagem bridge.pgm

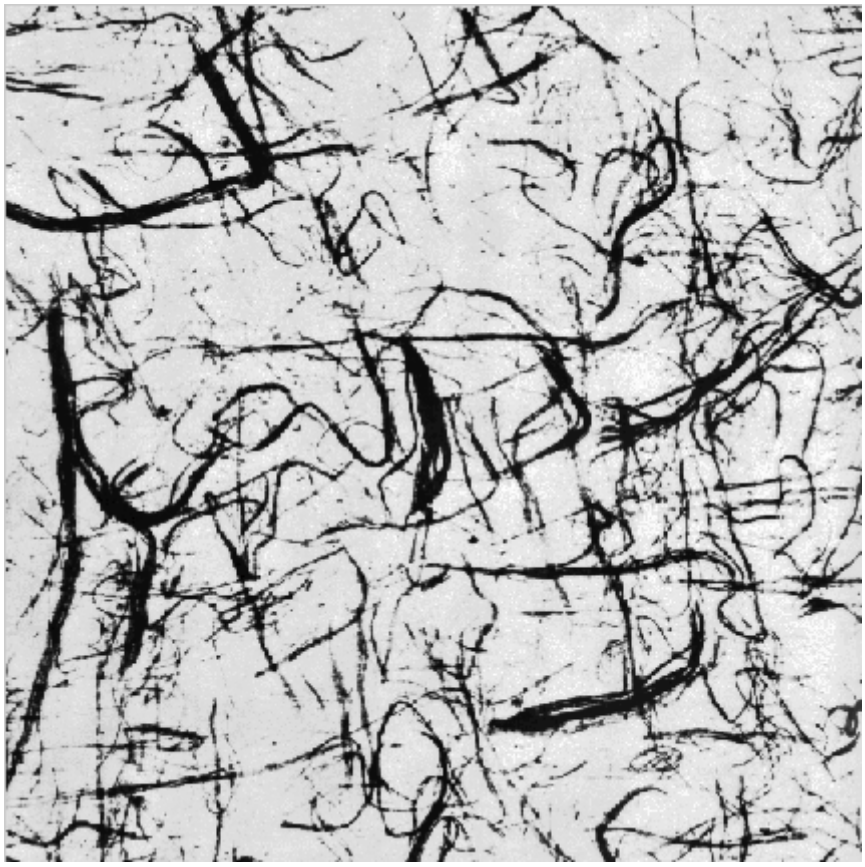


Figura 46: Quarta imagem-chave utilizada nos experimentos. (D108.pgm)

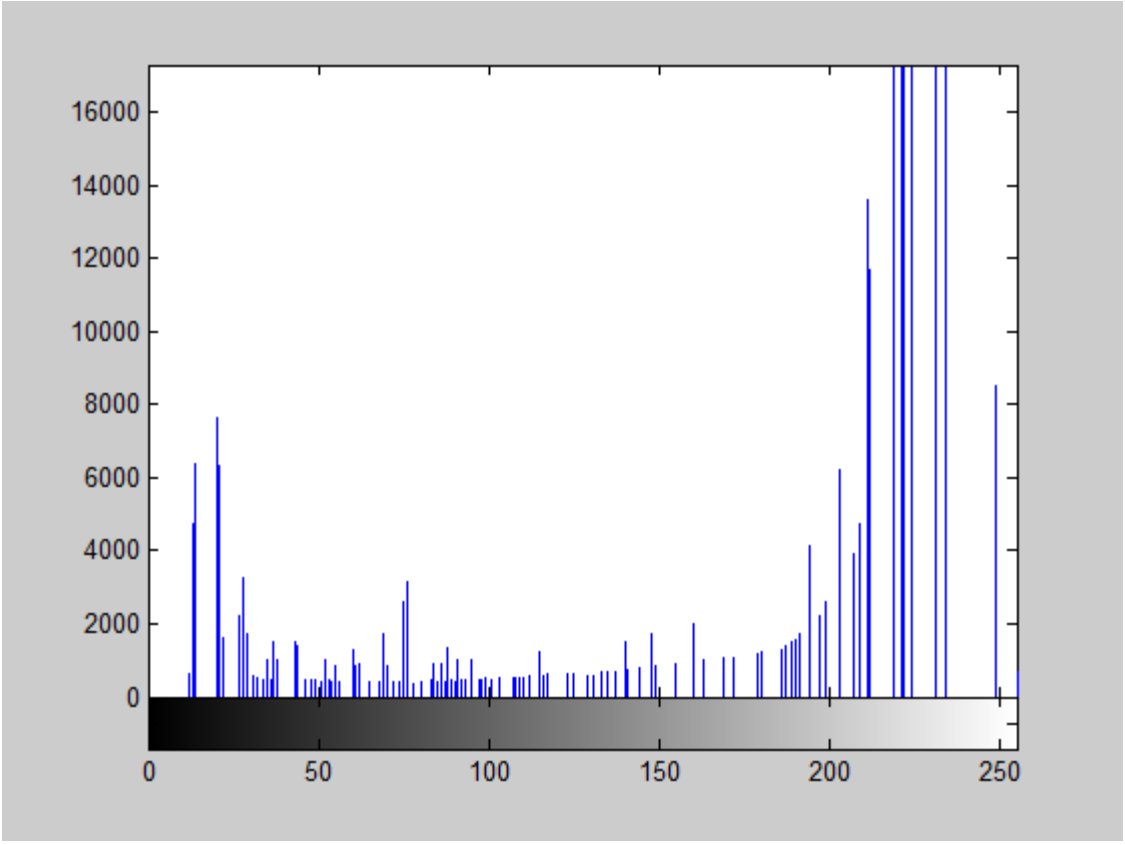


Figura 47: Histograma da imagem D108.pgm

Figura 48: Quinta imagem-chave utilizada nos experimentos. (f16.pgm)

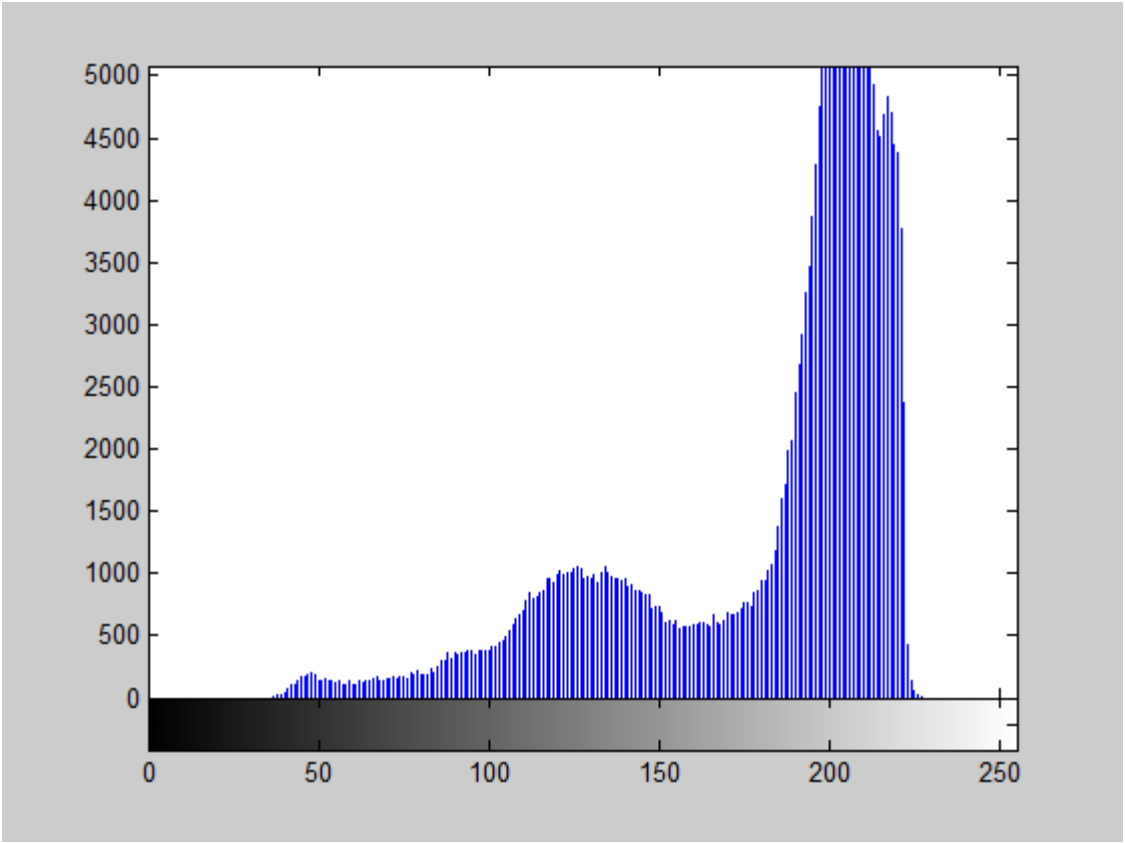


Figura 49: Histograma da imagem f16.pgm

Figura 50: Sexta imagem-chave utilizada nos experimentos. (girl.pgm)

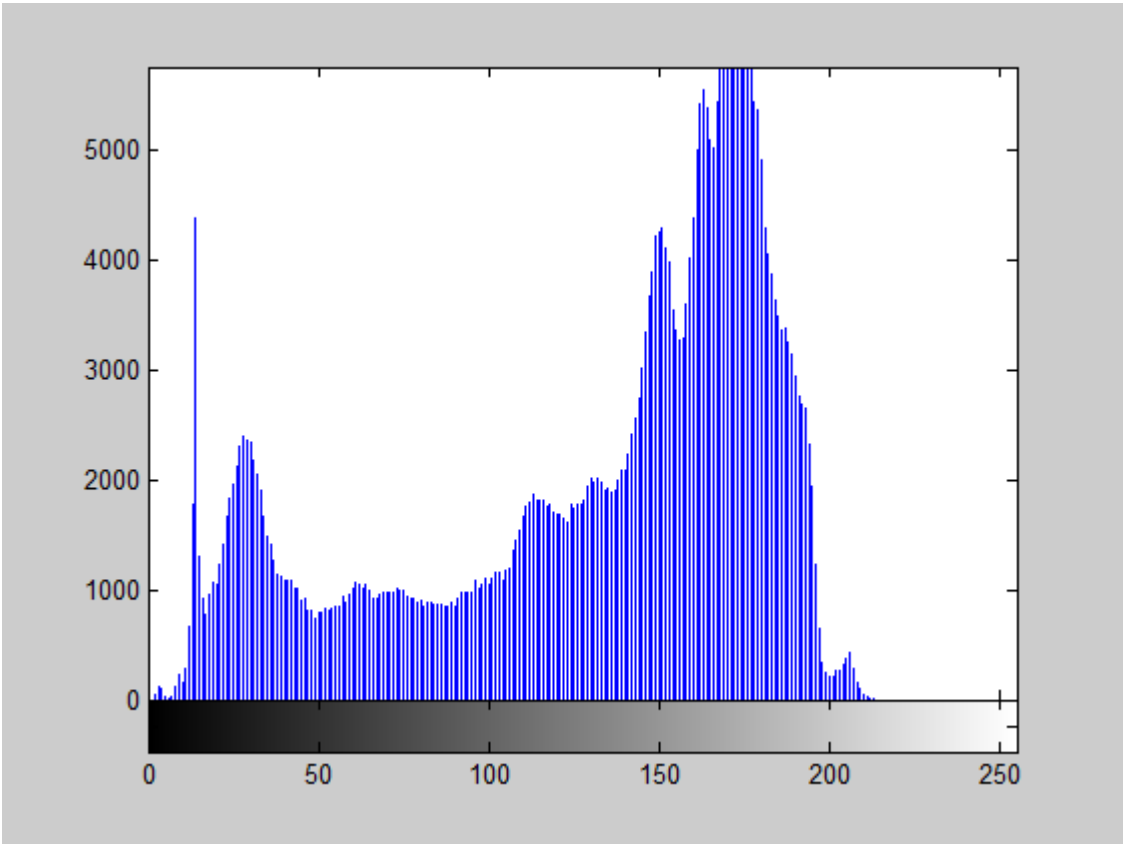


Figura 51: Histograma da imagem girl.pgm

Figura 52: Sétima imagem-chave utilizada nos experimentos. (lena.jpg)

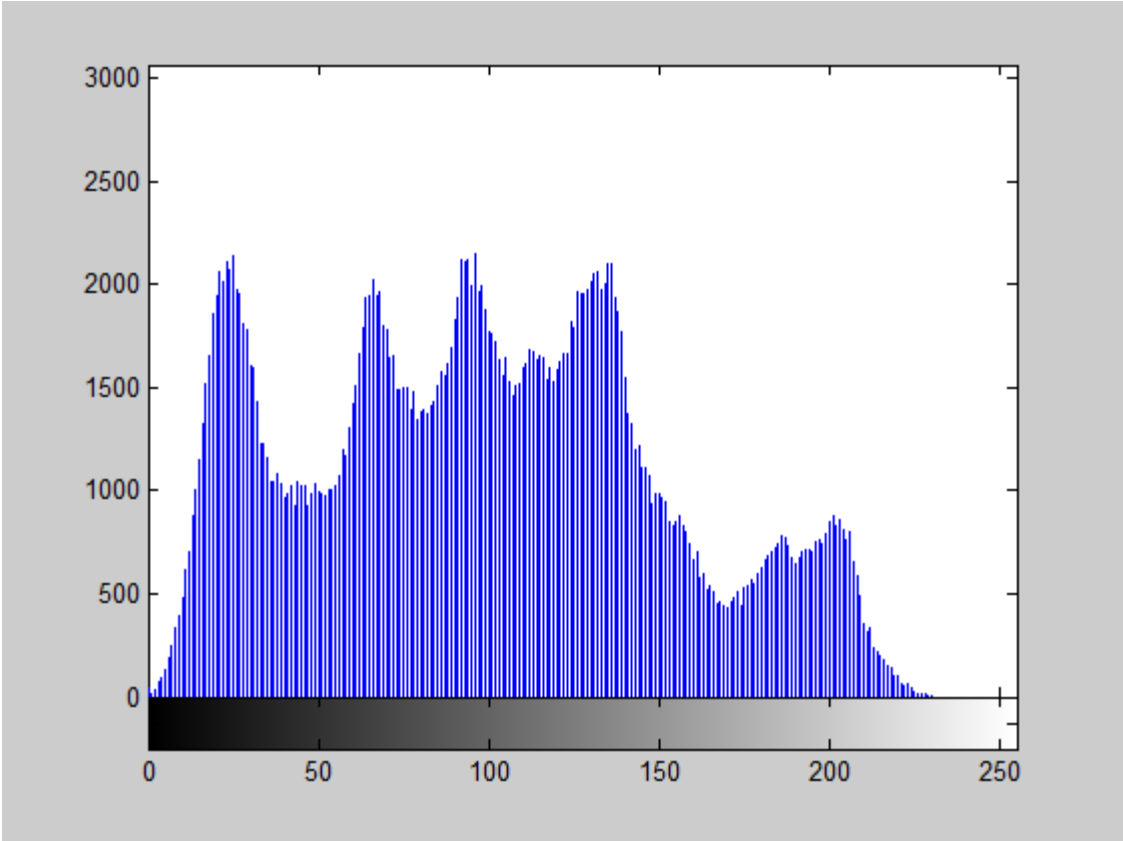


Figura 53: Histograma da imagem lena.jpg

Figura 54: Oitava imagem-chave utilizada nos experimentos. (peppers.pgm)

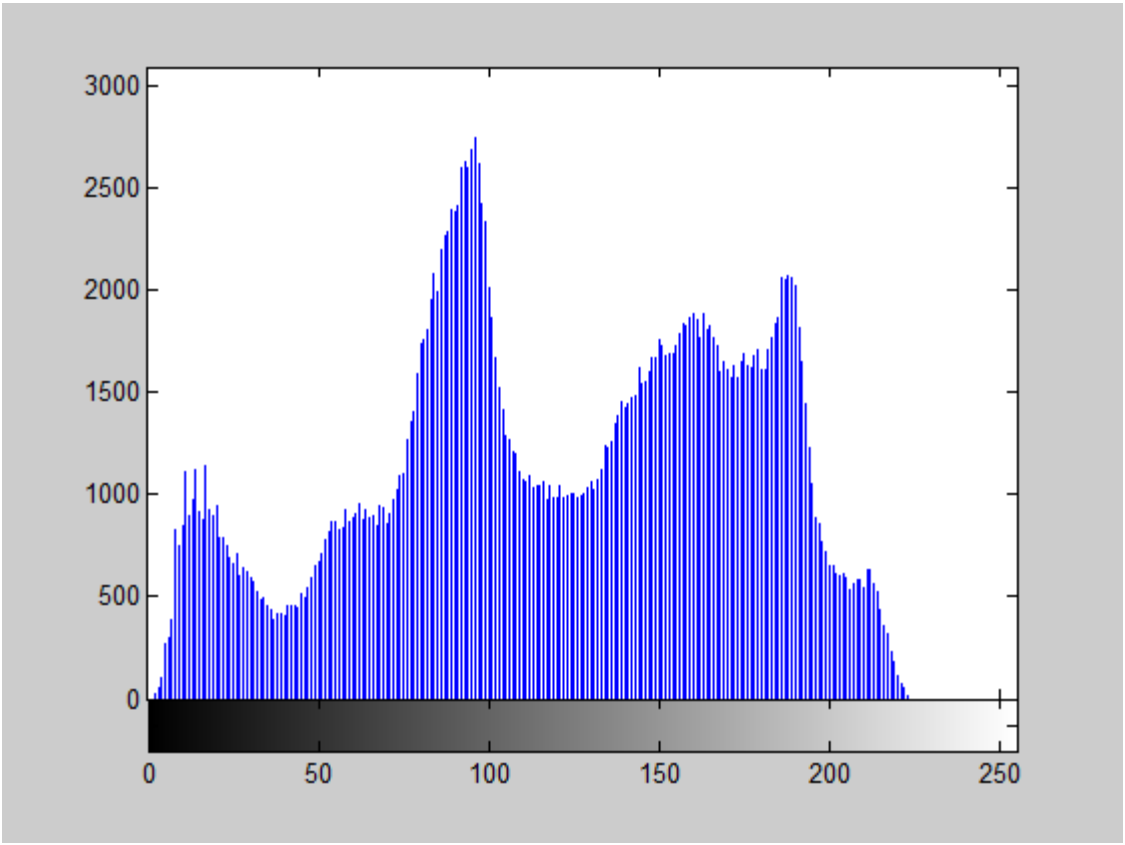
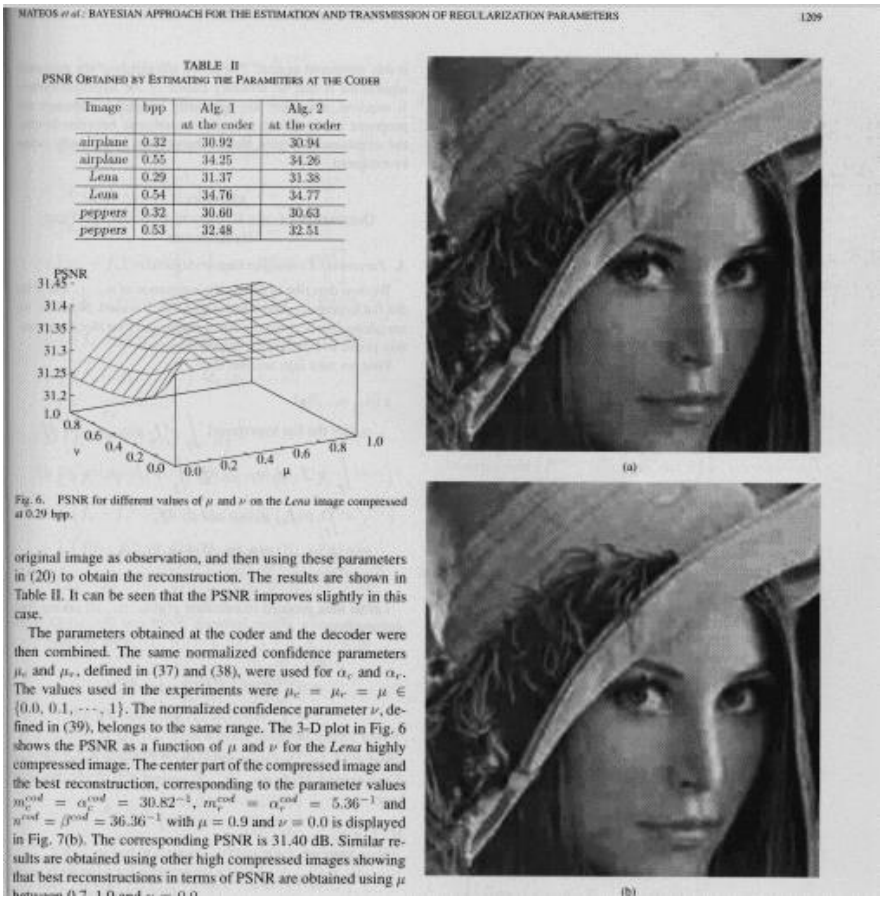


Figura 55: Histograma da imagem peppers.pgm



MATEOS *et al.*: BAYESIAN APPROACH FOR THE ESTIMATION AND TRANSMISSION OF REGULARIZATION PARAMETERS 1209

TABLE II
PSNR OBTAINED BY ESTIMATING THE PARAMETERS AT THE CODER

Image	bpp	Alg. 1 at the coder	Alg. 2 at the coder
airplane	0.32	30.92	30.94
airplane	0.55	34.25	34.26
Lena	0.29	31.37	31.38
Lena	0.54	34.76	34.77
peppers	0.32	30.60	30.63
peppers	0.53	32.48	32.51

Fig. 6. PSNR for different values of μ and ν on the *Lena* image compressed at 0.29 bpp.

original image as observation, and then using these parameters in (20) to obtain the reconstruction. The results are shown in Table II. It can be seen that the PSNR improves slightly in this case.

The parameters obtained at the coder and the decoder were then combined. The same normalized confidence parameters μ_c and μ_r, defined in (37) and (38), were used for α_c and α_r. The values used in the experiments were $\mu_c = \mu_r = \mu \in \{0.0, 0.1, \cdots, 1\}$. The normalized confidence parameter ν, defined in (39), belongs to the same range. The 3-D plot in Fig. 6 shows the PSNR as a function of μ and ν for the *Lena* highly compressed image. The center part of the compressed image and the best reconstruction, corresponding to the parameter values $m_c^{cod} = \alpha_c^{cod} = 30.82^{-1}$, $m_r^{cod} = \alpha_r^{cod} = 5.36^{-1}$ and $n^{cod} = \beta^{cod} = 36.36^{-1}$ with $\mu = 0.9$ and $\nu = 0.0$ is displayed in Fig. 7(b). The corresponding PSNR is 31.40 dB. Similar results are obtained using other high compressed images showing that best reconstructions in terms of PSNR are obtained using μ

(a)

(b)

Figura 56: Nona imagem-chave utilizada nos experimentos. (pp1209.pgm)

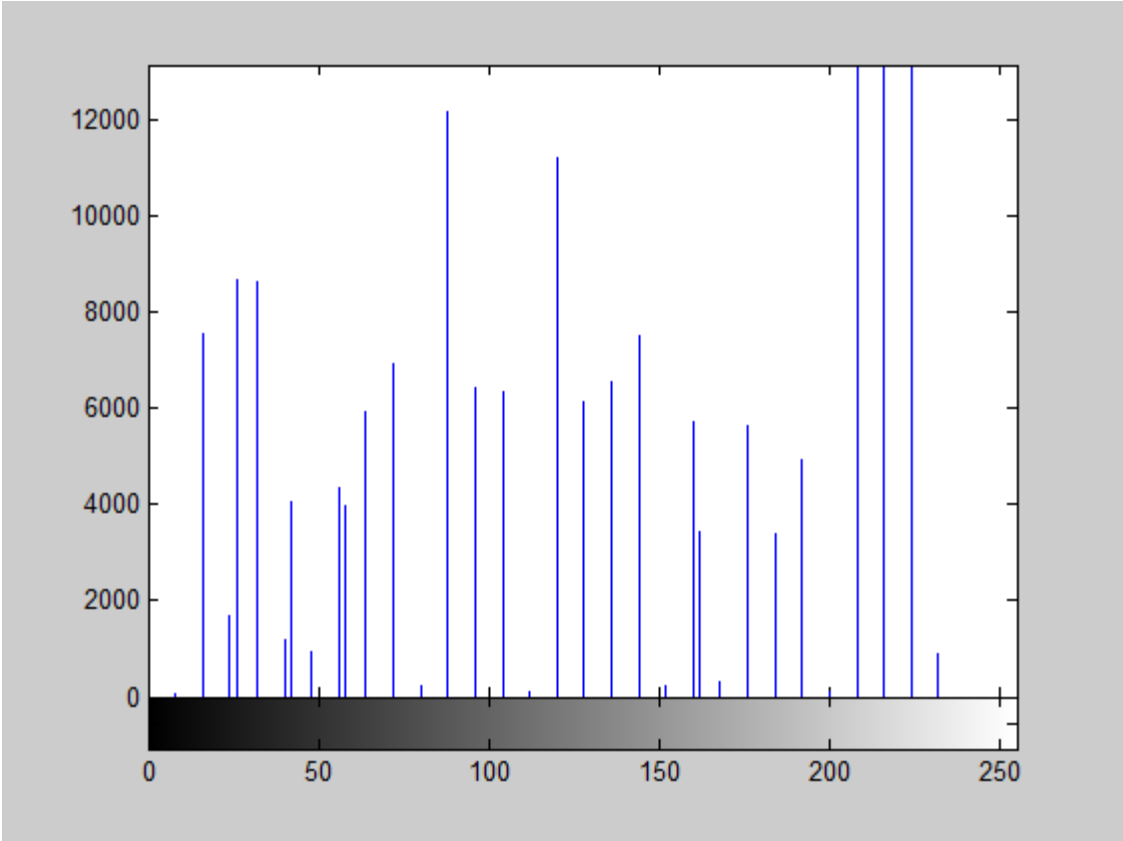


Figura 57: Histograma da imagem pp1209.pgm

Figura 58: Décima imagem-chave utilizada nos experimentos. (zelda.pgm)

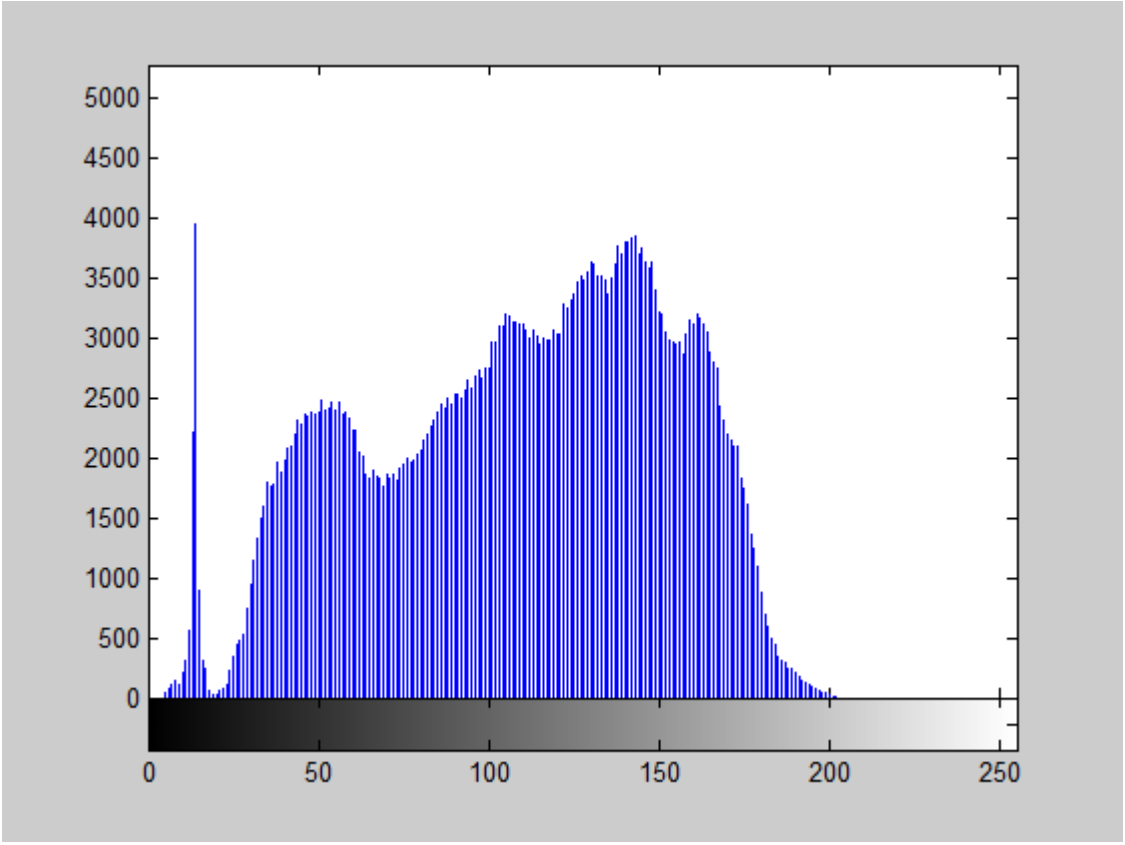


Figura 59: Histograma da imagem zelda.pgm