UNICEUB - CENTRO UNIVERSITÁRIO DE BRASÍLIA
FAET – FACULDADE DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE ENGENHARIA DA COMPUTAÇÃO

PROCESSAMENTO DIGITAL DE IMAGENS

IMPLEMENTAÇÃO DE WATERMARKING

Aluno: Cezário Rodrigues da Costa Neto – R.A.: 2001566-1
Orientador: Prof. MSc. Aderlon M. Queiroz

BRASÍLIA-DF
2004

UNICEUB - CENTRO UNIVERSITÁRIO DE BRASÍLIA
FAET – FACULDADE DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE ENGENHARIA DA COMPUTAÇÃO
PROJETO FINAL

PROCESSAMENTO DIGITAL DE IMAGENS

IMPLEMENTAÇÃO DE WATERMARKING

Monografia, sob a orientação do Prof. MSc. Aderlon Marcelino Queiroz avaliado por Banca Examinadora do Curso de Engenharia da Computação da Faculdade de Ciências Exatas e Tecnologia - FAET do Centro Universitário de Brasília - UniCEUB e constituiu requisito para obtenção do título de Bacharel em Engenharia da Computação.

Brasília-DF, dezembro de 2004.

“Há duas coisas infinitas: o universo e a tolice dos homens.”

Albert Einstein

Agradecimentos

A Deus, pela vida, saúde e tudo mais por Ele consentido para nossa felicidade.

Aos pais, pelo apoio durante a árdua jornada nos estudos e confiança depositada ao longo dos anos.

Aos meus irmãos, cuja juventude deram-me inspiração, esperança e ânimo em momentos difíceis.

Aos mestres e professores, em especial ao professor MSc. Aderlon M. Queiroz pela sua orientação nesta pesquisa.

Ao caro amigo Antonio Victor, que não mediu esforços para a revisão gramatical.

Ao primo e amigo Charles e sua esposa Sara, pelo auxílio na língua estrangeira.

Aos colegas e companheiros pelo auxílio e intercâmbio de idéias ao longo dos anos de estudo.

A todos aqueles que de alguma forma contribuíram com o sucesso não apenas deste trabalho, mas que tiveram fundamental importância em toda a minha vida acadêmica e profissional.

Resumo

O tema a ser abordado neste projeto é sobre marcas d'água digitais. Marca d'água ou *watermark* é um sinal portador de informação que acrescentada ao dado digital (imagem, áudio, vídeo etc.) pode ser extraído posteriormente com o intuito de passar informações sobre o dado. O objetivo deste é criar um software que seja capaz de incluir marcas d'água digitais frágeis em imagens 2D, estáticas, 256 tons de cinza, com a finalidade de autenticar, identificar e verificar possíveis alterações nos conteúdos destes dados.

As atividades desenvolvidas ao longo do projeto foram: pesquisa e estudo sobre marcas d'água digitais, matemática estatística e computação gráfica; criação de marca d'água personalizada; estudo de algoritmos criptográficos; ajuste de algoritmos de leitura e inserção de marca d'água; pesquisa e preparação de imagens para realização dos testes.

Palavras-chave: Processamento de imagem, marca d'água, *watermarking*, criptografia, identificação, autenticação, direitos autorais, propriedade intelectual.

Abstract

The topic of the present paper is the digital watermarking which is a signal containing information that added to the digital data (image, audio, video etc) which can be extracted lately with purpose of passing information about the data. The objective of this project is to make up a software that is capable of including fragile digital watermarking in 2D images, static, 256 tons of gray to authenticate, identify and verify possible changes in the contents of this data.

The activities developed throughout the paper were the following: Research and the study of digital watermarking, statistics and computer graphics; creation of personal watermark; the study of cryptographic algorithms; adjust of reading algorithms and insertion of watermark; research and preparation of the images to make the tests.

Keywords: Image processing, watermarking, identification, authentication, copyrights, intellectual property.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1 – Introdução

1.1 Motivação

As questões dos direitos autorais e de propriedade intelectual sempre causaram polêmicas. A pirataria e a distribuição ilegal de sons e imagens não são problemas recentes. Com o advento da Internet e das redes globalizadas de comunicação, o acesso à informação foi aumentado exponencialmente e, com estes benefícios, vieram também outros problemas. A distribuição de produtos que possuem direitos autorais aumentou na mesma proporção do acesso à informação. Os primeiros a reivindicar os direitos foram as indústrias fonográficas. Produtores de filmes e grandes estúdios também estão entrando nas batalhas judiciais que envolvem sistemas de busca ponto a ponto e seus desenvolvedores.

Em relação ao direito autoral de imagens, os detentores de suas propriedades estão na busca de soluções que evitem sua distribuição não autorizada. Ainda temos problemas no que tange a identificação e autenticação da imagem, ou seja, quem produziu determinado dado. Outro fato, não menos importante e que está gerando bastante polêmica, é a alteração e adulteração de imagens. Os softwares de edição de imagens estão cada vez mais robustos e permitem alterações cada vez mais precisas. Nos tópicos de identificação, autenticação e alteração de imagens é que será embasado este projeto.

1.2 Objetivos

Desenvolver um software aplicativo, que seja capaz de incluir marca d'água de autenticação frágeis, em imagens 2D estáticas, meio-tom (256 tons de cinza), no formato *bitmap*, utilizando de chaves criptográficas de modo a obter a autenticação do dado, identificação do autor e a verificação de possíveis alterações posteriores à assinatura do dado digital.

1.3 Descrição dos Capítulos

Será descrita brevemente a composição do trabalho em cada capítulo.

Capítulo 2: Define o termo marca d'água digital, suas características e aplicações.

Capítulo 3: Introduz conceitos de processamento digital de imagens e estrutura do *bitmap*.

Capítulo 4: Apresenta tópicos de criptologia, exemplos de criptossistemas e assinaturas digitais.

Capítulo 5: Exemplifica o protótipo, quais os recursos e técnicas utilizadas na sua criação.

Capítulo 6: Mostra os resultados obtidos na realização das simulações e testes.

Capítulo 7: Faz as considerações finais do trabalho e propõe trabalhos futuros.

Capítulo 2 – A Marca d'Água

2.1 Histórico

Existem duas formas básicas de se esconder uma informação: criptografia e a esteganografia. Comumente, podem-se confundir os termos. Entretanto, eles possuem significados distintos. Criptografia é o estudo dos métodos de formas de enviar mensagens cifradas em que apenas os receptores poderão traduzi-las. Já a esteganografia é a técnica utilizada para o envio de mensagens de forma que elas possam ser levadas escondidas dentro de outros conteúdos, sem que sejam percebidas, até seu destino final.

Ao longo dos milênios, antigas civilizações já se utilizavam de artifícios da esteganografia com a intenção de enviar mensagens com diversos propósitos (MORAES, 2004). Com o passar do tempo, foram surgindo variações nas formas esteganográficas e uma delas, que será o objeto de estudo deste trabalho, é a marca d'água.

A marca d'água é uma informação que é embutida numa mensagem, que deverá passar alguma informação sobre esta mensagem. Técnicas para encaixe de marcas d'água são conhecidas há mais de 2000 anos. As primeiras marcas d'água surgiram da necessidade de autenticar documentos. Os papéis eram compostos com diferentes tipos de fibra (um para o papel em si, outro para mostra da marca d'água). Após a prensagem e secagem, o resultado era o papel com a marca d'água, que era modelada de acordo com o interesse do proprietário.

Atualmente, as marcas d'água estão presentes em vários lugares. Papéis timbrados, selos, papel moeda e, inclusive, dados digitais possuem seus mecanismos de identificação através de marcas d'água. A tabela 2.1 mostra as principais características da marca d'água.

Tabela 2.1 – Marca d'água digital X Esteganografia

CARACTERÍSTICA	MARCA D'ÁGUA	ESTEGANOGRAFIA
Quantidade de dados incluídos	Pequenas quantidades	Tanto quanto possível
Facilidade em detectar	De acordo com a necessidade	Muito difícil.
Facilidade em remover	Importante que não seja removido	Importante que não seja removido
Objetivo do atacante	Remover a marca	Detectar a informação
Objetivo do usuário	Embutir assinatura para que possa ser comprovada a propriedade	Esconder a informação para que não possa ser descoberta
Uso na atualidade	Proteção de direitos de propriedade	Espionagem industrial, terrorismo

2.2 Conceito

De acordo com MOHANTY (1999, com adaptações), marca d'água "é o processo que encaixa um dado chamado marca d'água ou uma assinatura digital dentro de um objeto multimídia, onde tal marca pode ser detectada ou extraída posteriormente para fazer alguma asserção sobre o objeto".

Resumidamente, KIM (2004), define: "uma marca d'água é um sinal portador de informação, visualmente imperceptível, embutido numa imagem digital que pode ser extraído mais tarde para fazer asserção sobre o dado hospedeiro".

Uma marca d'água é melhor descrita pela comparação com uma marca d'água tradicional. Marcas d'água tradicionais são adicionadas a alguns tipos de papel para oferecer uma prova de autenticidade. Elas são imperceptíveis, exceto quanto o papel é levado à luz. Analogamente, marcas d'água digitais são adicionadas a dados que podem ser verificados por um computador, mas imperceptíveis ao olho humano. A marca d'água leva uma mensagem com informações sobre o criador, distribuidor ou o proprietário do dado.

A partir daqui, não havendo possibilidade confusão, utilizar-se-á apenas a palavra marca para referenciar a marca d'água.

2.3 Princípios Básicos

2.3.1 Descrição Formal de uma Marca d'Água

O princípio básico de funcionamento de uma marca pode ser descrito em dois subsistemas: Codificação (*C*) e decodificação (*D*). Os elementos participantes destes processos são o dado original (*O*), também chamado de hospedeiro, a marca d'água (*W*), o dado marcado ($O^{'}$) e uma chave (*K*), que será opcional.

A codificação pode ser representada pelo seguinte esquema:

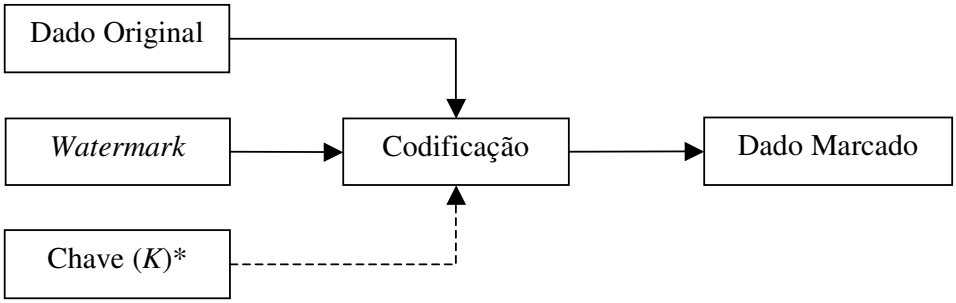

Figura 2.1 – Codificação

E ainda pode ser descrita pela equação:

$$C(O,W,K^{*}) = O^{'} \tag{2.1}$$

Onde

C: Codificação

W: Marca d'água

K^{*}: Chave (uso facultativo)

$O^{'}$: Dado Marcado

Já a decodificação, representa-se assim:

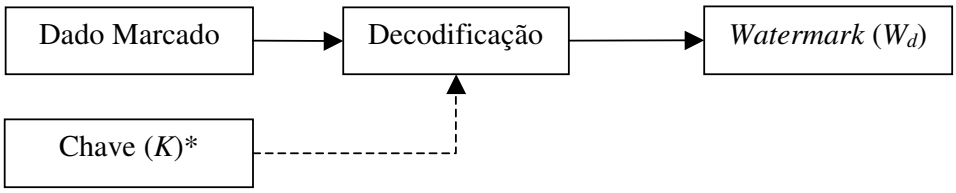

Figura 2.2 – Decodificação

Descrita pela equação:

$$D(O', K^*) = W_d \tag{2.2}$$

Onde

D: Decodificação

O': Dado Marcado

K^*: Chave (uso facultativo)

W_d: Marca d'água extraída

Neste caso, se a chave foi utilizada na codificação, ela será um elemento necessário no processo de decodificação.

Realizadas as etapas de codificação/decodificação, devem-se comparar as marcas (W e W_d) e verificar se houve ou não manipulações no dado.

2.3.2 Classificação da Marca d'Água

As marcas d'água e as técnicas de encaixe das marcas podem ser dividias em várias categorias e de várias formas.

Podem-se classificar as marcas, conforme o tipo do hospedeiro. Os hospedeiros podem ser: texto, imagem, áudio e vídeo. Neste trabalho, tem-se o encaixe de marcas em imagens digitais, em duas dimensões e estáticas.

Marcas podem ser inseridas conforme o domínio. Existem métodos diferentes para embutir marca d'água em imagens digitais. Uma delas é no domínio do espaço e a outra no domínio da freqüência. Neste trabalho, as marcas serão encaixadas no domínio do espaço.

Podem ser classificadas, ainda, de acordo com sua perceptibilidade visual. Existem marcas visíveis e invisíveis. Aqui, a abordagem será para marcas invisíveis.

Ainda, existem as classificações quanto à dificuldade em removê-las. De acordo com KIM (2004), as marcas podem ser divididas em robustas, semifrágeis e frágeis. A seguir, uma breve descrição destas características:

• Marcas Robustas: São projetadas para resistirem à maioria dos procedimentos de manipulação de imagens. A informação embutida numa imagem através de uma marca robusta deve ser possível de ser extraída mesmo que a imagem hospedeira sofresse rotação, mudança de escala, mudança de brilho/contraste, compactação com perdas com diferentes níveis de compressão, corte das bordas (*cropping*) etc. Uma boa marca d'água robusta deve ser impossível de ser removida a não ser que a qualidade da imagem resultante deteriorasse a ponto de destruir o seu conteúdo visual. Isto é, a correlação entre uma imagem marcada e a marca robusta nela inserida deveria permanecer detectável mesmo após um processamento digital, enquanto a imagem resultante do processamento continuasse visualmente reconhecível e identificável como a imagem original. (KIM, 2004)

• Marcas Frágeis: são facilmente removíveis e corrompidas por qualquer processamento na imagem. Este tipo de marca é útil para se checar a integridade e a autenticidade da imagem, pois possibilita detectar alterações na imagem. Em outras palavras, uma marca frágil fornece uma garantia de que a imagem marcada não seja despercebidamente editada ou adulterada. Neste sentido, o termo "frágil" é infeliz para qualificar esses algoritmos, sendo mantido por razões históricas. Talvez o termo mais apropriado seja "marca d'água de autenticação". (KIM, 2004)

• Marca Semifrágil: Também serve para autenticar imagens. Só que estas procuram distinguir as alterações que modificam uma imagem substancialmente daquelas que não modificam o conteúdo visual da imagem. Uma marca semifrágil normalmente extrai algumas características da imagem que permanecem invariantes através das operações "permitidas" e as insere de volta na imagem, de forma que a alteração de uma dessas características possa ser detectada. (KIM, 2004)

A robustez aqui aplicada será do tipo frágil.

Existe também uma classificação quanto à forma de recuperação da marca d'água. O termo utilizado para expressar esta classificação é o vocábulo "cego". Por exemplo, uma marca "cega" é aquela cuja recuperação poder ser realizada sem a necessidade do dado original, ao passo que uma marca "não-cega" deve possuir, obrigatoriamente, o dado original, pois a marca será recuperada com a comparação entre ambos os dados. A marca possuirá a característica de ser "cega", ou seja, não necessita do dado original, para que possam ser feitas asserções sobre o dado.

Existem, ainda, diversos outros tipos de classificações das marca d'água, como por exemplo: Pública ou privada, onde todos os usuários ou apenas o proprietário poderá extrair a Marca; Invertibilidade, Reversibilidade entre inúmeras outras, que assim como estas citadas neste parágrafo não serão objetos de estudo neste trabalho.

2.4 Aplicações

2.4.1 Proteção dos Direitos Autorais

Para proteger a propriedade intelectual, o proprietário do dado pode encaixar uma marca d'água que contenha informações de *copyright* no dado. Esta aplicação pode ser uma ferramenta útil para determinar os direitos de propriedade numa disputa judicial. É provavelmente a maior motivação que está disseminando o uso de marcas d'água digitais. (HAMPIHOLI, 2004, com adaptações)

A proteção dos direitos possuídos pelo criador, ou proprietário legítimo, de uma parte ou de todo o trabalho é rodeado de muitos aspectos especiais, incluindo a proteção dos direitos autorais e direitos morais, como por exemplo a garantia de que a integridade do trabalho será respeitada e não será violada a crença no proprietário/criador. Devido à grande variedade de situações encontradas em aplicações práticas, para um grande número de objetivos possíveis de serem perseguidos pelos sistemas de proteção intelectual, que existem diferentes legislações em diferentes países, é impossível (e está fora do escopo do trabalho) dar um tratamento unificado de marcas utilizadas para proteção dos direitos autorais. (BARNI; BARTOLINI, 2004, com adaptações)

Esta é a mais clássica situação de uso da marca d'água: o autor de um trabalho deseja provar que ele é o único proprietário. Para tanto, assim que ele criar um trabalho, também deverá encaixar uma marca que individualiza seu produto. Contudo, é sabido que a marca pode não ser válida judicialmente, a não ser que o proprietário mostre o esquema de encaixe da marca no dado que ele está reivindicando. Mas a partir do momento que ele abre seu esquema de codificação, perde-se toda a segurança oferecida pelo método utilizado. (BARNI; BARTOLINI, 2004, com adaptações)

Um algoritmo a ser usado para verificação de posse legal deve possuir, sobretudo, um bom esquema de segurança, pois se sabe que falsificadores estão sempre interessados em

remover a marca, possivelmente por meios computacionais de força bruta. Além do mais, marcas privadas são preferíveis, pois elas são naturalmente seguras (Apenas os detentores da propriedade podem removê-las). (BARNI; BARTOLINI, 2004, com adaptações)

2.4.2 Autenticação

Um dos efeitos da disponibilidade de muitas ferramentas robustas de processamento de sinais, e possivelmente utilizadas para modificar o visual ou o conteúdo digital de documentos sem deixar traços perceptíveis da modificação, é a perda de credibilidade do dado digital. Para solucionar tal problema, é necessário que o proprietário tome medidas preventivas que garantam a autenticação dos sinais gravados em formato digital. (BARNI; BARTOLINI, 2004, com adaptações)

Isso pode ser resolvido com a aplicação de uma marca d'água. Imagine-se a seguinte situação: José é fotógrafo e distribui as imagens por ele produzidas para imprensa. José poderá incluir uma marca que identifique sua propriedade nas suas imagens e distribuí-las aos clientes. Os clientes também podem realizar o procedimento inverso. Através da marca, eles podem confirmar a origem das fotos. (BARNI; BARTOLINI, 2004, com adaptações)

2.4.3 Detecção de Alterações

Em dados marcados, o proprietário tem condições de saber quais regiões do dado original foram alteradas para a produção do dado marcado. Caso ocorra algum processamento de sinal no dado marcado, é possível saber onde foram realizadas tais modificações. (BARNI; BARTOLINI, 2004, com adaptações).

Neste caso, pode-se verificar se um dado marcado foi alterado ou não, bastando apenas observar, no processo de extração da marca, em quais pontos foram realizados os ataques. Falhas podem ser visualmente percebidas na verificação. (BARNI; BARTOLINI, 2004, com adaptações)

Capítulo 3 – Processamento Digital de Imagens

3.1 Percepção Visual Humana

O sistema óptico ocular é formado por um complexo esquema fisiológico, o qual permite interpretar não somente a sensação de cor, mas também a profundidade, textura, movimento etc. É formado basicamente pelos globos oculares (olhos), nervos ópticos, corpos geniculares laterais e as áreas de visão. (FILHO, 1999)

A parte fisiológica responsável pela tradução dos raios luminosos em impulsos (sinais elétricos) a serem interpretados pelo cérebro é denominada globos oculares (receptores das imagens), que são formados pela córnea, íris, retina etc. A retina, por sua vez, é uma membrana ocular interna, em que estão as células nervosas (bastonetes e cones) que recebem os estímulos luminosos, e onde se projetam as imagens produzidas pelo sistema óptico ocular. (FILHO, 1999)

O conceito de cor é extremamente importante e indispensável não só para a computação gráfica, pois com o seu uso pode-se explorar um dos principais sentidos, a visão. Os seres humanos podem detectar cerca de 100 níveis de intensidade de luz. O desafio do sistema digital é reproduzir esta resposta, convertendo a informação recebida em um número apropriado de cores. Sabemos que todas as cores que o olho humano pode perceber são combinações de três cores primárias: Vermelho/Verde/Azul ou *Red/Green/Blue* (Padrão RGB). (FILHO, 1999)

O que ocorre na prática é que, para resolvermos uma determinada tarefa de interpretação de imagens, utilizamos um conjunto de algoritmos bastante específicos, que são respectivamente responsáveis por realizar subtarefas bastante limitadas dentro do processo de interpretação dessa imagem. Esses algoritmos são divididos em grupos, como filtros de contraste, detectores de bordas de objetos, segmentadores de imagens em regiões, classificadores de texturas e assim por diante. Comumente, resolvemos um problema encaixando um conjunto desses algoritmos um atrás do outro para chegarmos a um resultado que só funcionará para um conjunto de imagens com características muito específicas, deixando de funcionar para todas as outras. (FILHO, 1999)

3.2 Processamento Gráfico

O processamento gráfico (PG) está ligado a qualquer forma de manipulação de informações gráficas através do uso de um computador. (CASACURTA *et al*., 1998)

Trata-se do estudo, geração e manipulação gráfica através dos computadores. O processamento gráfico está subdividido em processamento de imagens e computação gráfica, envolvendo a síntese, tratamento, análise e visualização de imagens, bem como a sua animação. (CASACURTA *et al.*, 1998)

3.2.1 Processamento de Imagens

O processamento de imagens envolve as técnicas de transformação de imagens, em que tanto a imagem original quanto a imagem resultado apresentam-se sob uma representação visual (geralmente matricial). Estas transformações visam melhorar as características visuais da imagem (aumentar contraste, foco, ou mesmo diminuir ruídos e/ou distorções). (TRAINA; OLIVEIRA, 2004)

O processamento de imagens irá manipular as imagens, modificando-as ou identificando os seus elementos componentes. O processamento de imagens possui inúmeras aplicações, tais como:

- Tratamento e melhoria de imagens a fim de permitir a visualização de algum detalhe específico de interesse do usuário. Possui inúmeras aplicações, como por exemplo: medicina (tomografia, ultra-sons etc), geologia, sensoriamento remoto (imagens de satélite), meteorologia, entre outros.

- Reconhecimento e classificação de objetos presentes em uma imagem. Aplicável em sistemas de segurança (impressões digitais, reconhecimento de face), interpretação automática de textos, visão artificial, robótica, exploração. (CASACURTA *et al.*, 1998)

3.3 Imagem

A imagem digital pode ser definida matematicamente como uma função bidimensional *A(x, y)* em uma certa região do plano. (TRAINA; OLIVEIRA, 2004)

$$A : [0, r] \times [0, s] \rightarrow [0, t] \tag{3.1}$$

Assim a imagem é definida num retângulo [0,*r*]×[0,*s*], e os valores tomados estão contidos no intervalo [0,*t*]. (TRAINA; OLIVEIRA, 2004)

Imagens são geralmente definidas como uma grande coleção de dados, que estão frequentemente dispostos em forma de uma matriz ou alguma outra estrutura de dados discreta. (TRAINA; OLIVEIRA, 2004)

Uma imagem é representada através de um conjunto de valores, onde cada valor é um número que descreve os atributos de um *pixel* na imagem. Usualmente estes números são inteiros positivos, que representam a intensidade na escala de tons de cinza de um ponto da imagem, ou a cor associada ao ponto. Quando se utiliza escalas de tons de cinza, usualmente associa-se a tonalidade de cinza mais escura, o preto, ao nível zero da escala de cinza, e o mais claro, o branco, ao maior valor permitido na escala. Ou seja, se cada *pixel* é representado em um sistema de 8 *bits*, o preto é associado ao valor zero, e o branco ao valor 255. (TRAINA; OLIVEIRA, 2004)

As dimensões do conjunto de valores que especifica a imagem são chamadas de largura e altura da imagem, e o número de *bits* associado com cada *pixel* da matriz é chamado de profundidade da imagem. (TRAINA; OLIVEIRA, 2004)

Uma determinada imagem possuirá também uma "resolução" associada a ela, que é o número de elementos que esta imagem possui na horizontal e na vertical. Cada elemento da imagem possuirá uma localização, que é definida pela suas coordenadas. (CASACURTA *et al.*, 1998)

A alteração na apresentação dos *pixels* permite modificar a imagem, alterando seu tamanho, intensidade e cor. Para realizar esta modificação é necessário conhecer cada elemento da imagem. Uma maneira de se referir aos *pixels* é considerar a imagem como uma matriz. Observe a ilustração a seguir.

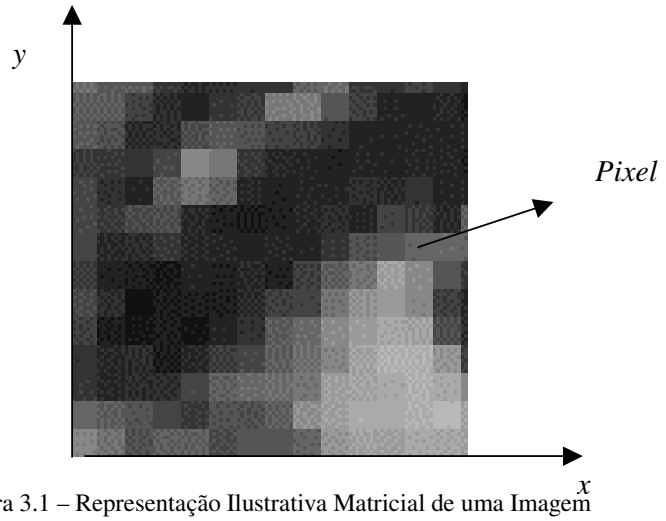

Figura 3.1 – Representação Ilustrativa Matricial de uma Imagem

3.3.1 Atributos

Uma imagem é uma matriz de pontos ou *pixels*, com uma determinada resolução horizontal (eixo X) e vertical (eixo Y), onde para cada ponto desta matriz temos uma cor associada. A cor pode ser obtida de forma direta ou através de uma tabela de acesso indireto. (CASACURTA *et al.*, 1998).

A cor é uma sensação relacionada especificamente à percepção visual dos seres. O homem é um dos poucos animais que possui a sensação de distinguir várias tonalidades de cor.

A noção de "COR" pode ser definida através da "*tri-stimulus theory*", que será explicada de uma maneira simplificada. O ser humano possui em seu sistema visual três tipos de sensores capazes de identificar três faixas diferentes de "espectros de energia". Estas faixas correspondem às tonalidades de Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*). Logo, o ser humano vê na realidade a combinação resultante da mistura destas três cores básicas. (CASACURTA *et al.*, 1998)

O sistema de cores utilizado nos computadores é usualmente o sistema RGB (*Red-Green-Blue*), onde o que fazemos é controlar a intensidade da geração destas três cores básicas. Ao definirmos uma determinada cor em um computador, o que estamos especificando na realidade é a intensidade (valor associado) aos emissores R, G e B. Por meio de testes realizados com o ser humano chegou-se à conclusão de que a utilização de 256 variações diferentes de intensidade em cada uma das cores básicas é capaz de gerar um

número de cores superior à capacidade visual do ser humano, ou seja, fica praticamente impossível de distinguir entre duas cores "vizinhas". (CASACURTA *et al.*, 1998)

No sistema RGB, o valor (0, 0, 0) equivale a cor preta com intensidade zero nas três componentes. O valor (255, 255, 255) equivale a cor branca onde as três componentes estão presentes com a sua intensidade máxima. As diferentes combinações entre RGB serão capazes de gerar qualquer tipo de cor, sendo que se as três componentes tiverem sempre valores exatamente iguais teremos definida uma escala de tons de cinza do preto ao branco, é a chamada "*gray scale*" ou tons de cinza. (CASACURTA *et al.*, 1998)

Existem outros padrões para composição de cores, como por exemplo: CMY (*Cyan, Magenta, Yellow*), HLS (*Hue, Lightness, Saturation*) etc.

3.4. Armazenamento de Imagens e Estrutura do Bitmap

Ao manipular imagens, deve-se prestar muita atenção no que tange ao seu armazenamento. Existe uma necessidade de se criar padrões de armazenamento de imagens, para que possamos realizar o intercâmbio de imagens entre diferentes sistemas e adaptados a situações específicas. Outro fato, não menos importante, é a compressão das imagens, pois já é sabido que as imagens ocupam muito espaço em memória.

3.4.1 Formatos de Arquivos

Existem diferentes formatos de arquivos para o armazenamento de imagens, uma vez que temos várias classes diferentes de representações de imagens. O armazenamento da imagem envolve basicamente três elementos principais: a forma como a imagem está representada, o tipo de compactação empregado e o cabeçalho contendo as informações acerca desta imagem (resolução, quantidade de bits, classe da imagem, compactação etc). Um mesmo tipo de arquivo pode até permitir o armazenamento de diferentes classes de imagens e também permitir a utilização de vários métodos de compactação. (CASACURTA *et al.*, 1998)

Existem vários tipos de formatos para armazenamento de imagens. Entre eles pode-se citar o BMP (*Windows*), PCX, GIF, TIFF, JPG ou JPEG (formato comprimido) etc.

3.4.2. O *Bitmap*

O formato de armazenamento de imagem que será abordado neste projeto é um padrão de codificação da Microsoft ® para arquivamento de imagens no sistema operacional Windows®. Por ser um formato nativo, é utilizado em diversos locais dentro do sistema operacional, que vão de papéis de parede, ícones, ponteiros de mouse etc.

Os arquivos BMP podem ser classificados conforme a quantidade de *bits* para representar 1 *pixel (bit/pixel*); existindo versões de *1 bit/pixel* (2^1=2 cores), 4 *bit/pixel* (2^4=16 cores), 8 *bit/pixel* (2^8=256 cores), 24 *bit/pixel* (com até 2^{24}, ou seja, mais de 16 milhões de cores) e mais recentemente 32 bits (*true color* com até 2^{32}, com mais de 4 bilhões de cores). Convém lembrar que o ser humano não é capaz de distinguir visualmente mais que 16 milhões de cores. (OLIVEIRA, 2000)

A partir de agora, e quando não houver perigo de confusão, *Bitmap* será interpretado como o formato de armazenamento de arquivo no padrão *Bitmap*.

3.4.3 Estrutura geral do *Bitmap*

Cada arquivo BMP é composto por um cabeçalho de arquivo (*header*), por uma área de informação do cabeçalho de mapa de *bits* (*bitmap-information header*), uma tabela de cores (*color table*) e uma seqüência de *bits* (*bitmap bits*) que definem o conteúdo da imagem. A seguir, uma breve descrição do que contêm estas áreas.

- Cabeçalho de arquivo

Contém a assinatura BM (um dado posicionado no início do arquivo para indicar que é uma imagem BMP) e informações sobre o tamanho e *lay-out* do arquivo BMP (disposição dos dados dentro do arquivo).

- Cabeçalho de *mapa de bits*

Contém as informações da imagem contida no arquivo. Define as dimensões, tipo de compressão (se houver) e informações sobre as cores da imagem.

- Paleta ou mapa de cores (opcional)

Somente estará presente em arquivos de imagens que usam 16 ou 256 cores (4 e 8 *bits/pixel*). Nas demais, em seu lugar, vem diretamente a parte seguinte: área de dados da imagem.

- Área de dados da imagem contida no arquivo

Dados que permitem a exibição da imagem propriamente dita, ou seja, *pixels* a serem exibidos. Podem ser com ou sem compressão.

Para realização do protótipo, serão utilizados só os *bits* da área de dados da imagem.

Capítulo 4 – Criptologia

4.1 Histórico e Conceituação

Palavra de origem grega, criptologia quer dizer estudo (*logos*) do oculto (*criptos*). Como definição atual, a criptologia é, segundo MORAES (2004), como sendo "o estudo de códigos e cifras (não necessariamente secretas)".

A criptologia ainda é dividida em duas áreas de estudos que se complementam: a criptoanálise e a criptografia. A primeira pode ser descrita como sendo a ciência das "quebras" de códigos, com o interesse de decifrar a informação, sem o conhecimento das chaves, ao passo que a segunda, de acordo com MORAES (2004) em sua definição mais básica como "Arte de escrever com chave secreta ou de modo enigmático". Atualmente, diz-se que a criptografia deixou de ser uma "arte", para ser um conjunto de técnicas que oferecem proteção a mecanismos de acesso e a integridade de dados, bem como ferramentas de avaliação destas técnicas.

Sabe-se, no entanto, que a criptografia não é um estudo da atualidade. Historiadores afirmam desde a Antiguidade, os homens já possuíam formas de cifrar importantes informações. No Egito antigo e no Império Romano, existem citações históricas de "esquemas" para esconder informações confidenciais (dados sobre tesouros, caminhos, informações de guerra etc.). Existem relatos que dão conta de que o imperador Júlio César, rei de romano da Antiguidade, utilizava-se de substituições no alfabeto para o envio de mensagens aos seus exércitos. Funcionava extremamente bem, até que descobriram como decifrá-lo, tornando-se, portanto, obsoleto. (MORAES, 2004)

4.2 Encriptação e Decriptação

A encriptação é definida como o processo que transforma os dados de uma forma que o torna inelegível, a não ser que se possua o conhecimento adequado. Deve garantir a privacidade da informação, não deixando entendimento a qualquer um que não seja seu destino.

Já a decriptação, segundo MORAES (2004), "o reverso da encriptação é a transformação de dados encriptados novamente em uma forma inteligível". Dessa forma, temos aqui um inter-relacionamento entre encriptação - decriptação, ou seja, primeiramente o proprietário codifica a mensagem e/ou dado, trafega-os por diversos canais (que podem ser interceptados) e os seus usuários os decodificam seguramente.

As rotinas de encriptação e decriptação necessitam de um dado secreto, que atuará como chave. Mecanismos em que a cifragem usa a mesma chave para a decifragem são chamados de simétricos ou de chave privada, ao passo que os que utilizam de chaves distintas para realização dos processos de encriptação e decriptação são denominados assimétricos ou de chave pública.

Nas palavras de MORAES (2004), "técnicas de criptografia podem ser usadas como um meio efetivo de proteção de informações suscetíveis a ataques, estejam elas armazenadas em um computador ou sendo transmitidas pela rede".

Adiante, diagrama simplificado de um sistema criptográfico.

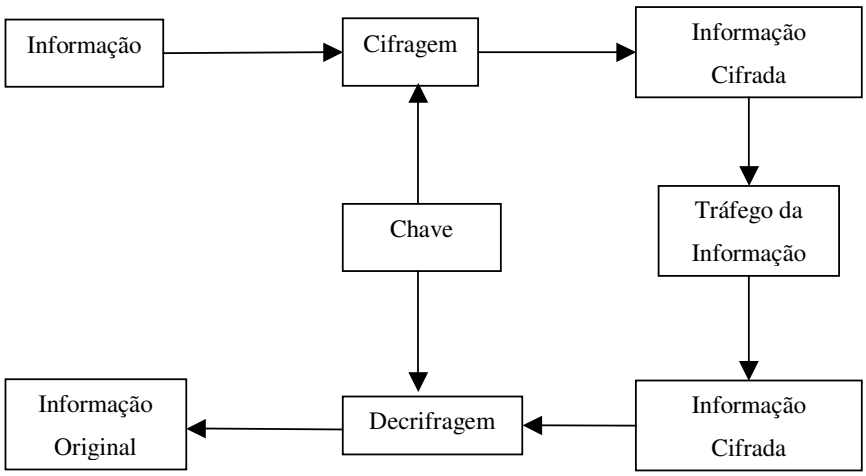

Figura 4.1 – Esquema de Codificação, Envio e Decodificação de uma Informação.

É extremamente importante ressaltar que a segurança da criptografia não reside nos algoritmos de encriptação e decriptação, mas sim no sigilo da(s) chave(s) utilizada(s) no sistema. Apenas usuários autorizados deverão ter o conhecimento das delas.

4.3 Requisitos de Segurança

Para que a informação seja trafegada de forma segura, reduzindo riscos de acessos não autorizados, uso indevido, roubo de informações e inúmeros outros problemas, deve-se dar atenção aos seguintes requisitos. A saber:

- Autenticidade: Está associada à identificação correta dos participantes. De acordo com MORAES (2004), "o serviço de autenticação em um sistema deve assegurar ao receptor que a mensagem é realmente procedente da origem informada em seu conteúdo".

- Confidencialidade: Segundo BENITS (2003), é dar "garantia de que as informações armazenadas em um sistema de computação ou transmitidas via rede de computadores seja acessada ou manipulada somente pelos usuários devidamente autorizados". É importante dar atenção não apenas no dado como um todo, mas também nas partes que o compõe.

- Integridade: Consiste em proteger a informação contra modificações não autorizadas. Deve fornecer a garantia de que a informação ou o dado transmitido seja entregue ao destinatário de forma íntegra, sem violações, assim como foi expedido pelo emissor.

- Não-Repudiação ou Irretratabilidade: Garante que nenhum dos envolvidos no sistema criptográfico (emissor ou receptor) negue a transmissão, recepção ou posse do dado. Este requisito só é garantido na criptografia assimétrica.

4.4 Formas de Ataques

Segundo BENITS (2003), "a segurança de uma mensagem foi comprometida quando um criptanalista consegue quebrar um texto criptografado, (isto é, quando consegue transformar o texto criptografado de volta ao texto legível correspondente) ou quando consegue quebrar a chave (isto é, descobre a chave que foi utilizada para criptografar a mensagem, no caso de criptografia simétrica, ou a chave de decriptografia, no caso de criptografia assimétrica)".

Entre as inúmeras formas de ataques existentes, podemos citar:

- Ataque do texto cifrado: o atacante tem muitas mensagens cifradas, mas desconhece as mensagens originais e as chaves que foram usadas. Aqui ele vai tentar deduzir a chave utilizada.

- Ataque do texto conhecido: o atacante tem a sua disposição uma grande quantidade de mensagens cifradas e também as mensagens originais equivalentes. Sua tarefa é deduzir as chaves.

- Ataque da chave escolhida: o atacante pode testar o sistema com diversas chaves diferentes, ou pode convencer diversos usuários legítimos do sistema a utilizarem determinadas chaves em mensagens por ele enviadas. Neste último caso, a finalidade imediata seria de decifrar as mensagens cifradas com essas chaves.

- Ataque cortar e colar: o atacante utiliza-se de mensagens cifradas com uma mesma chave para combinar trechos e, assim, gerar uma mensagem.

- Força bruta ou exaustão – Neste tipo de ataque, o interesse está em desvendar a chave usada pela codificação de uma mensagem através do método de tentativa e erro.

4.5 Algoritmos

Nas palavras de CASTRO *et al*. (2004), "os algoritmos de criptografia são técnicas formais, baseadas em expressões matemáticas combinadas à lógica de programação, utilizadas para cifrar informações, utilizando algum valor ou chave para este processo".

Existem duas classes de sistemas criptográficos, conforme dito anteriormente. Uma utiliza a chave privada (ou simétrica) e outra utiliza chave pública (assimétrica). Segundo BORCHARDT (2002), podem ainda "existir sistemas criptográficos baseados somente em algoritmo, que neste caso deve ser secreto, ou seja, conhecido somente entre as partes".

4.5.1 Cifras por Deslocamento

A idéia da cifra por deslocamento é manter os mesmos caracteres do texto plano, apenas rearranjando-os através da mudança posicional dos seus caracteres. As tabelas a seguir ilustram resumidamente este processo.

Tabela 4.1 – Informação Original

Posição Original	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Caracter	A	B	C	D	E	F	G	H	I	J

Tabela 4.2 – Informação Cifrada pelo Deslocamento de 4 Posições

Posição Original	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Caracter	G	H	I	J	A	B	C	D	E	F

Para obtenção da informação original, é necessário realizar outro deslocamento, neste caso, no valor diferença do tamanho da informação pelo valor inicialmente utilizado no ciframento.

4.5.2 Simétricos

Um sistema criptográfico simétrico é aquele que utiliza a mesma chave para cifrar e decifrar uma mensagem. É conhecido também como criptografia por chave única ou criptografia tradicional.

No momento em que duas entidades desejam trocar mensagens encriptadas, será necessário, primeiramente, combinar de forma segura uma mesma chave. Depois de combinada a chave a ser utilizada, admitamos que Alice (origem) pode enviar para José (destino) mensagens que são cifradas com a chave K (acordada de forma segura), codificando-as em novo conjunto de bits que podem ser transmitidos por qualquer meio, que José pode obter os dados originais das mensagens, bastando, para tanto, efetuar a decriptação com a mesma chave K.

Como exemplos de algoritmos simétricos, temos:

• DES (*Data Encryption Standard*): Criptografia de bloco criada pela IBM e endossada pelo governo dos Estados Unidos em 1977. O DES é um codificador composto que cifra blocos de 64 bits (8 caracteres) em blocos de 64 bits, para isso se utiliza de uma chave composta por 56 bits, com 8 bits de paridade totalizando 64 bits. (CASTRO *et al*., 2004)

• IDEA (*International Data Encryption Algorithm*), desenvolvido na década de 80 por Xuejia Lai e James Massey da ASCOM Tech AG da Suíça, em Zurique. Ele foi projetado para ser facilmente programado, é forte e resistente a muitas formas de criptoanálise. O seu método baseia-se na utilização de uma chave de 128 bits, onde blocos de texto de 64 bits da mensagem de entrada são alterados em uma seqüência de interações, produzindo blocos de saída. Sua chave de 128 bits é suficiente para resistir à maior parte dos ataques. O IDEA é usado pelo popular programa PGP, para encriptar arquivos e correio eletrônico. (MORAES, 2004)

• RC6: Projetado por Ron Rivest. A última versão de uma série de cifradores (RC2, RC3, RC4, RC5) desenvolvidos por ele. Voltado para criptografia de e-mail corporativo.

As vantagens dos algoritmos simétricos são muitas, como: simplicidade e rapidez, permitindo a cifragem rápida de grandes volumes de dados. Possuem chaves pequenas.

Entretanto, existem desvantagens tais como: Irretratabilidade, pois tanto o emissor quanto o receptor possuem a mesma chave, logo, não se sabe quem codificou e quem decodificou; para cada par de entidades (emissor/receptor), tem-se uma chave, não podendo, portanto, ser reaproveitada nas cifras de mensagens para um terceiro usuário; segurança da chave (é necessário um meio seguro para a combinação das chaves).

4.5.3 Assimétricos

Em 1976, Whitfield Diffie e Martin Hellman apresentaram o conceito de criptografia por chave pública. Nos sistemas assimétricos, cada pessoa possui um par de chaves, uma denominada chave pública e a outra, chave privada. A chave pública tem seu

conhecimento liberado, ao passo que a chave privada deve ser mantida em segredo pelo proprietário.

Assim, não é mais necessária a troca de informações confidenciais entre as partes envolvidas, sendo que todas as comunicações envolverão apenas a chave pública, não sendo necessária, portanto, a troca de chaves secretas por nenhuma das partes. Também, o sistema não exige credibilidade dos meios de transmissão envolvidos.

Dessa forma, qualquer um dos possuidores da chave pública poderá usá-la para enviar uma mensagem. Porém, a mesma informação só pode ser lida (decriptada) com o uso da chave privada associada, de exclusividade do proprietário.

Nas palavras de MAIA (1999), "o sistema pode ser simplificadamente assim explicado: José e todos os que desejam comunicar-se de modo seguro geram uma chave de ciframento e sua correspondente chave de deciframento. Ele mantém secreta a chave de deciframento; esta é chamada de sua chave privada. Ele torna pública a chave de ciframento: esta é chamada de sua chave pública".

A chave pública realmente condiz com seu nome. Qualquer pessoa pode obter uma cópia dela. José até mesmo encoraja a isto, enviando-a para seus amigos ou publicando-a em boletins. Assim, Kevin (usuário malicioso) não tem nenhuma dificuldade em obtê-la. Quando Alice deseja enviar uma mensagem a José, precisa primeiro encontrar a chave pública dele. Feito isto, ela cifra sua mensagem utilizando a chave pública de José, despachando-a em seguida. Quando José recebe a mensagem, ele a decifra facilmente com sua chave privada. Kevin, que interceptou a mensagem em trânsito, não conhece a chave privada de José, embora conheça sua chave pública. Mas este conhecimento não o ajuda a decifrar a mensagem. Mesmo Alice, que foi quem cifrou a mensagem com a chave pública de José, não pode decifrá-la agora.

Assim com o sistema simétrico, os algoritmos assimétricos possuem vantagens e desvantagens. A saber:

- Vantagens: Apenas a chave privada deve ser secreta, facilitando, portanto, a organização das chaves; a confidencialidade da informação é garantida, enquanto a chave privada estiver segura; a quantidade de chaves é pequena, quando comparada ao modelo do sistema simétrico.

- Desvantagens: Têm a execução mais lenta, tornando-se ineficientes para grandes volumes de dados.

Entre os diversos algoritmos existentes, pode-se citar:

• RSA (Rivest, Shamir e Adleman): Talvez, o mais popular algoritmo de chave pública. Este algoritmo foi desenvolvido por um grupo de pesquisadores: Ronald Rivest do MIT, Adi Shamir do Weizmann Institute de Israel e Leonard Adleman da University of Southern California, sendo patenteado pelo MIT em 1978. Nas palavras de SILVA (1998), "a segurança do RSA está baseada na dificuldade de fatorar grandes números: as chaves são calculadas matematicamente combinando dois números primos de grande tamanho. Mesmo conhecendo-se o produto desses números primos (que faz parte da chave pública divulgada), a segurança do algoritmo é garantida pela complexidade de fatorar esse produto e se obterem os valores secretos.".

• ElGamal: Segundo MAIA (1999), é o "algoritmo de chave pública utilizado para gerenciamento de chaves. Sua matemática difere da utilizada no RSA, mas também é um sistema comutativo. O algoritmo envolve a manipulação matemática de grandes quantidades numéricas. Sua segurança advém de algo denominado problema do logaritmo discreto. Assim, o ElGamal obtém sua segurança da dificuldade de se calcular logaritmos discretos em um corpo finito, o que lembra bastante o problema da fatoração."

4.6 Resumo Digital - Funções de *Hash*

De acordo com BORCHARDT (2002), resumo digital é "o resultado da aplicação de determinadas funções matemáticas facilmente calculáveis que mapeiam uma cadeia de *bits* de qualquer tamanho em um número determinado fixo de *bits*, que é o tamanho do resumo".

Um resumo digital de "mão única" ou unidirecional é aquele que através de uma entrada, deve gerar um *hash* (resultado da função) facilmente. Entretanto, a operação inversa deve ser muito difícil ou praticamente impossível de ser realizada.

Outro detalhe importante é quanto à geração de seus resultados. O resumo deve ser único para cada mensagem ou informação, fazendo com que duas mensagens distintas não possuam o mesmo valor de *hash*.

Entre os algoritmos de *hash* existentes, pode-se citar:

• MD5 (*Message Digest* 5): É uma função de espalhamento unidirecional inventada por Ron Rivest (MIT), que também trabalha para a *RSA Data Security*. Nas palavras de MAIA (1999), "este algoritmo produz um valor *hash* de 128 *bits*, para uma mensagem de entrada de tamanho arbitrário. Foi inicialmente proposto em 1991, após alguns ataques de criptoanálise terem sidos descobertos contra a função *hashing* prévia de Rivest, a MD4. O algoritmo foi projetado para ser rápido, simples e seguro. Seus detalhes são públicos, e têm sido analisados pela comunidade de criptografia.".

• SHA (*Secure Hash Algorithm*): É uma função de espalhamento unidirecional inventada pela NSA (*National Security Agency*) e gera um valor *hash* de 160 *bits*, a partir de um tamanho arbitrário de mensagem. De acordo com MAIA (1999), "o funcionamento interno do SHA-1 é muito parecido com o observado no MD4 (precursor do MD5), indicando que os estudiosos da NSA basearam-se no MD4 e fizeram melhorias em sua segurança.". Como tem resultado um valor *hash* de 160 bits, torna-se, portanto, um pouco mais seguro.

4.7 Assinaturas Digitais

Na definição de BENITS (2003), "a assinatura digital é o processo de criptografar uma mensagem (ou um código associado univocamente a ela) com uma chave particular. Desta forma, garantimos a autenticidade da origem, pois somente quem possui a chave particular usada na criptografia poderá ter produzido o código gerado. Neste caso, a chave pública do usuário que criptografou a mensagem é utilizada para verificação da assinatura".

Dessa forma, pode-se observar que a assinatura digital só será possível na criptografia assimétrica ou de chave pública, pois através das chaves é possível saber quem emitiu a informação.

Conforme XÉXEO (2000), "a assinatura digital permite que qualquer pessoa que leia uma mensagem se certifique de que ela realmente foi assinada pelo originador da assinatura (garantia de origem) e de que a mensagem não foi modificada (garantia de integridade). Além disso, assinaturas digitais não podem ser repudiadas, isto é, o originador

da assinatura não pode, após assiná-la, deixar de cumprir as obrigações determinadas pela assinatura utilizando a afirmação de que a assinatura teria sido falsificada.".

Suponha que Alice queira enviar uma mensagem assinada digitalmente para José. Primeiramente, ela deverá disponibilizar sua chave pública para José. Em seguida, aplica-se uma função de resumo sobre a mensagem e, como resultado, obtém-se o resumo digital da mensagem que Alice cifrará com sua chave privada. O resumo cifrado juntamente com a mensagem original formam o corpo da mensagem digitalmente assinada na qual será enviada a José. Ao receber, ele retira da mensagem assinada a mensagem original e sobre ela aplica a mesma função de resumo que Alice utilizou inicialmente, obtendo dessa forma, o valor atual do resumo digital recuperado na mensagem. Utilizando-se da chave pública de Alice, José decifra o resumo assinado que está no corpo da mensagem assinada e obtém o valor do resumo da mensagem enviada por Alice. Por fim, José compara os valores dos resumos (o que veio na mensagem e o gerado por ele, ao recebê-la), e assim fazer asserções sobre a mensagem, como por exemplo, se foi adulterada ou não.

Como exemplo de esquemas de assinatura digital, pode-se citar:

- RSA

Criptossistema desenvolvido por Ron Rivest, Adi Shamir e Leonard Adleman em 1977. A segurança está baseada na dificuldade de se determinar fatores primos de um número inteiro muito grande, que são os que gerarão as chaves.

Algumas características estão presentes neste algoritmo, como por exemplo: dificuldade de fatorar números primos grandes (a quebra por força bruta atualmente é inviável); já é maduro e bastante estudado; permite sigilo; provê confidencialidade, autenticação, não-repúdio e integridade (essenciais para assinaturas digitais).

- DSA

O DSA (*Digital Signature Assimetric*) é um algoritmo de chave pública, proposto pelo órgão norte-americano NIST (*National Institute of Standards and Technology*) para uso no padrão DSS (*Digital Signature Standard*). Como seu nome indica, foi desenvolvido para assinatura digital. Logo, não pode ser usado para cifragem. Segundo SILVA (1998), "sua segurança é obtida do problema de logaritmo discreto. As matemáticas são muito diferentes do RSA, mas a segurança é semelhante para chaves semelhantes em tamanho."

Características como sigilo, autenticação, não-repúdio, dificuldade de quebra do algoritmo, dentre outras, estão presentes no DSA, como mecanismos de proteção da assinatura e garantia da inviolabilidade.

Capítulo 5 – Protótipo

5.1 Ferramenta Utilizada

Para criação e realização da inclusão da marca nas imagens, foi utilizado um software de computação algébrica, o Matlab® . A versão utilizada para implementação do projeto foi a 6.5, *Release* 13. O computador, um Intel Pentium IV de 2.4 GHz, 256MB de memória RAM e 40 GB de disco rígido.

O Matlab® é um ambiente de desenvolvimento com uma linguagem bastante simples e um ambiente computacional de fácil utilização. Fornece um grande conjunto de ferramentas (as *toolboxes*) que contém inúmeras funções matemáticas já implementadas, facilitando diversas operações como: manipulação de *bits*, matrizes e imagens, operações que serão exaustivamente aplicadas na implementação dos algoritmos.

Apesar da facilidade de uso da ferramenta, da vasta documentação e boa quantidade de exemplos de código, existe um custo a ser pago pelos seus usuários: o tempo de processamento.

Com relação a outras linguagens de desenvolvimento, o tempo de processamento no Matlab® é um pouco maior, pois se trata de um ambiente interpretado, e não compilado.

Sendo assim, o protótipo implementado não terá interesse comercial, mas apenas fins acadêmicos e de pesquisa.

5.2 Métodos e Recursos Utilizados

O algoritmo de inclusão e verificação de marca d'água proposto é baseado no esquema de WONG (1998) e o HBC (*Hash Block Chaining*), proposto por BARRETO (2003) e KIM(2004).

5.3 Marca Proposta

5.3.1 Inclusão da Marca

A inserção da marca proposta numa imagem em níveis de cinza pode ser simplificadamente descrita:

• Passo 1: Escolha uma imagem *I* em níveis de cinza a ser marcada, com N × M *pixels*. Zerar os *bits* menos significativos (LSBs) do hospedeiro obtendo I^*.

• Passo 2: Divida a imagem I^* em t blocos de 8×8 *pixels*.

• Passo 3: Calcule o *hash* para cada bloco I_t^*

• Passo 4: Escolha uma imagem-logotipo binária *B* a ser utilizada como marca d'água. Replique a imagem *B* de modo a obter uma imagem com as mesmas dimensões de *I*. Para cada bloco I_t^*, existirá um bloco B_t correspondente.

• Passo 5: Calcule o ou-exclusivo (XOR) de H_t com B_t, obtendo a impressão digital.

• Passo 6: Cifre, gerando assim a assinatura digital S_t do bloco *t*.

• Passo 7: Insira S_t nos LSBs de I_t^*, obtendo o bloco marcado. O resultado será a imagem marcada *I'*.

As figuras 5.1a e 5.1b ilustra o processo de inclusão da marca.

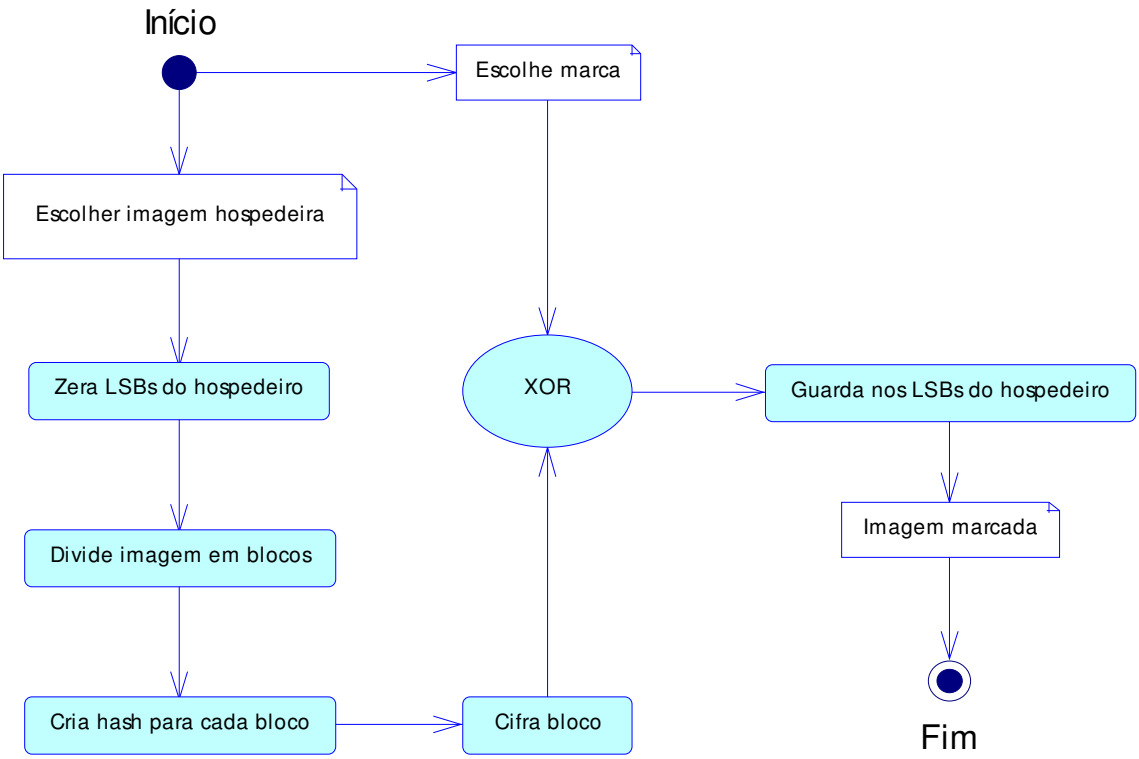

Figura 5.1a – Fluxograma - Inserção da Marca d'Água

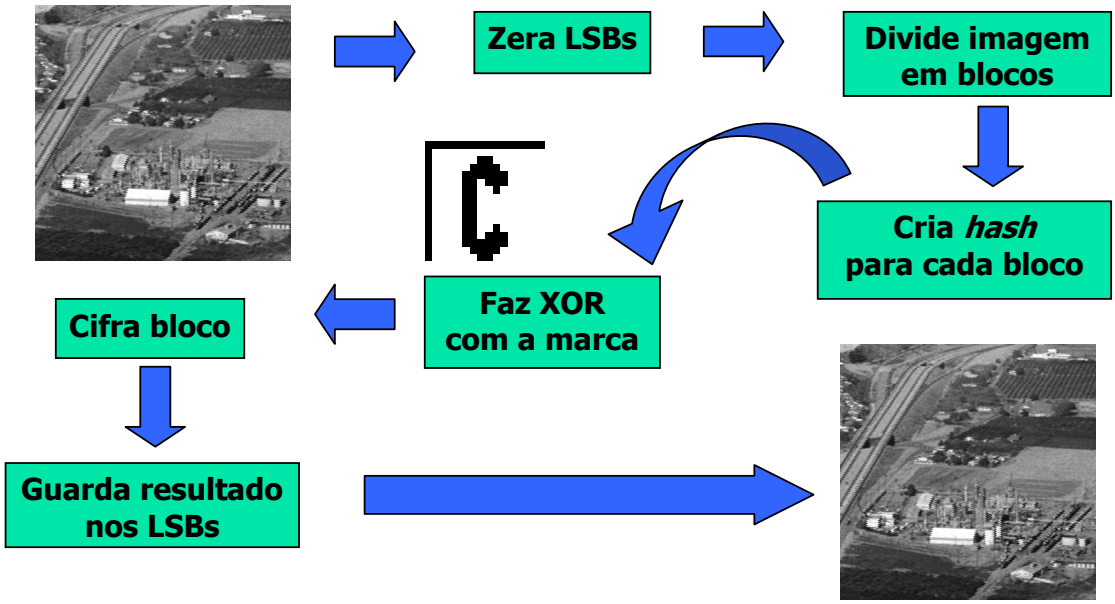

Figura 5.1b – Diagrama do Processo de Inserção da Marca d'Água

5.3.2 Verificação da Marca

O processo de verificação da marca d'água é:

• Passo 1: Seja *I'* uma imagem $N \times M$ em níveis de cinza com uma marca d'água inserida pelo processo descrito em 5.3.1. Particione *I'* em *t* blocos $I_t^{'}$, como na inserção.

• Passo 2: Pegar o bloco *t* e decifra-lo obtendo D_t.

• Passo 3: Zere os *bits* menos significativos (LSBs) do hospedeiro obtendo $I_t^{'*}$.

• Passo 4: Utilize a mesma função de *hash* e calcule o resultado.

• Passo 5: Calcule o ou-exclusivo (XOR) de H_t com D_t, obtendo o bloco de checagem *Ct*.

• Passo 6: Salvar o bloco C_t numa nova imagem I_v. A marca será verificada se em toda I_v puder ser visto claramente a imagem *B* replicada. Caso contrário, a imagem marcada foi alterada e um ruído será mostrado no bloco *t*.

As figuras 5.2a e 5.2b mostra esquematicamente o processo realizado na verificação da marca d'água.

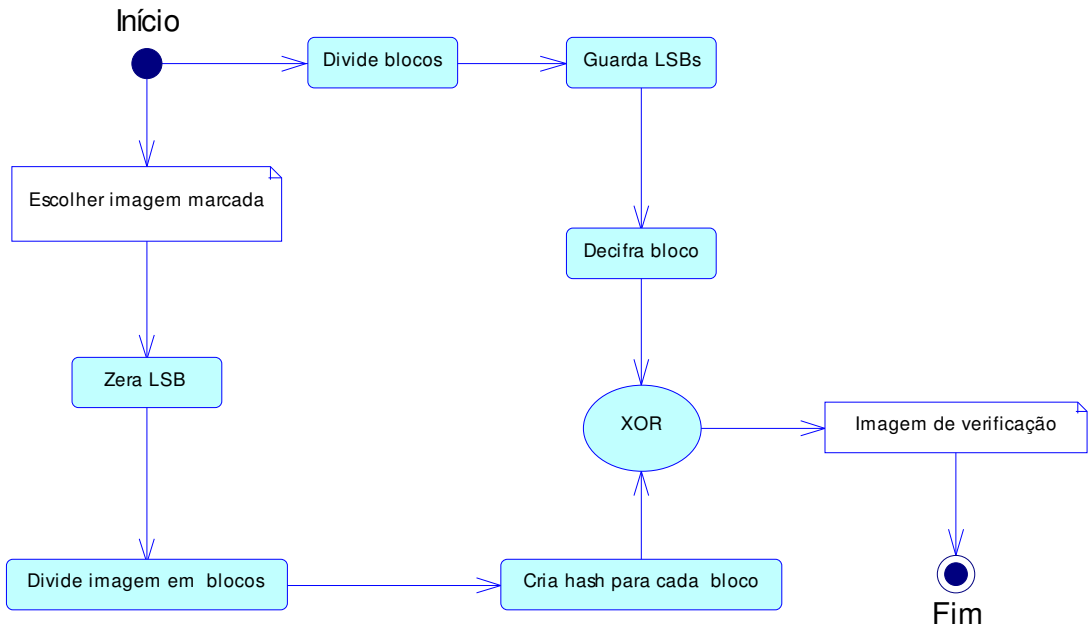

Figura 5.2a – Fluxograma – Verificação da Marca d'Água

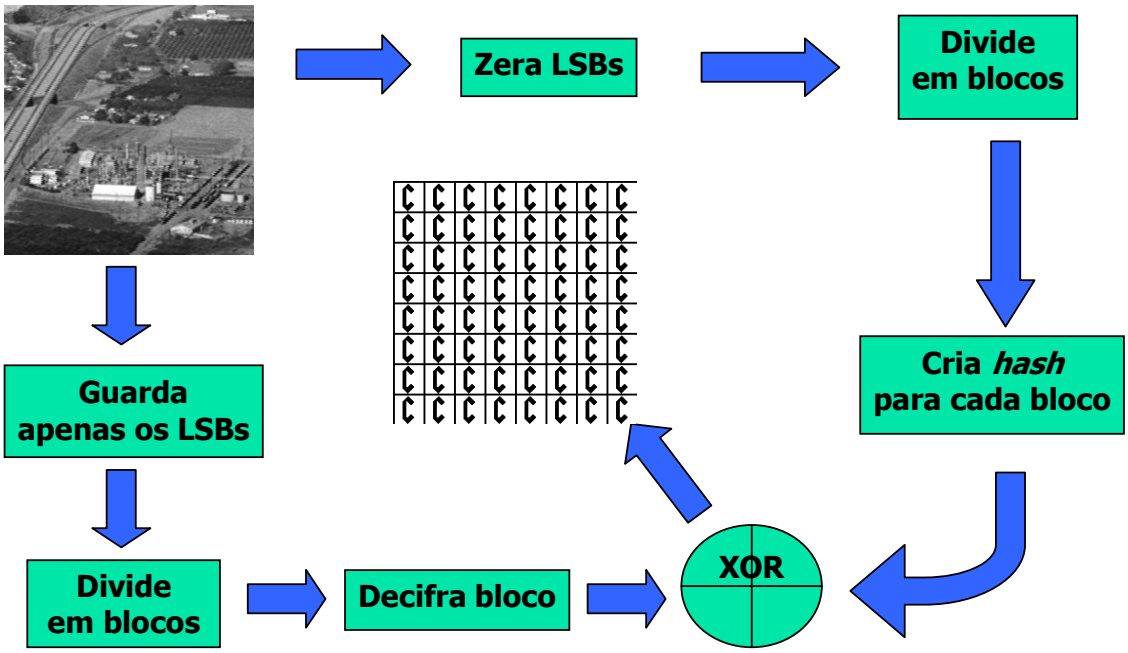

Figura 5.2b – Diagrama do Processo de Verificação da Marca d'Água

5.3 *Hash* MD5 e Crifa de Deslocamento

Para autenticar a imagem digitalmente, será utilizada a função de *hash* MD5 e a Cifra de Deslocamento, representada pela função *shift*.

Estes recursos criptográficos foram escolhidos pela quantidade e informação existentes, que por sinal já existem há bastante tempo e estão bem consolidadas.

5.4 *Bitmap* e *Bit* Menos Significativo (LSB)

Sabe-se que cada *pixel* no *bitmap* padrão com 256 tons de cinza, corresponde a 8 *bits*. No *bit* menos significativo é que serão inseridas as informações da marca. Segundo JASCONE (2003), "o método *Last Significant Bit* (LSB) é o mais comum utilizado para armazenar informação em imagens digitais. Consiste em utilizar o bit menos significativo de cada *pixel* (ou de cada cor) da imagem, para ocultar a mensagem.".

"Embora as técnicas de LSB consigam esconder os dados aos olhos humanos, elas podem ser facilmente destruídas computacionalmente utilizando algoritmos de compressão com perdas de dados. Estes algoritmos selecionam apenas as partes mais significativas do

objeto, ou seja, os *bits* menos significativos têm uma chance bem menor de serem selecionados. Por exemplo, se a mensagem for escondida em uma imagem no formato *bitmap*, e esta for convertida para um formato JPEG, a informação é perdida." (JASCONE, 2003)

5.6 Robustez e Segurança

Por ser uma marca frágil, toda e qualquer alteração realizada na imagem será percebida no algoritmo de verificação. Ataques do tipo cortar e colar, ampliação, redução, rotação, compressão, utilização de filtros ou qualquer outro retornará um ruído na imagem de verificação.

Caso a chave esteja incorreta, também será impossível recuperar a marca. A imagem de verificação também estará preenchida com um ruído, que não permite sua validação.

Outro fato importante é que se pelo menos um *bit* for alterado, todo o bloco é marcado como inválido. Isto ocorre porque as assinaturas são geradas por bloco.

Deve-se ressaltar que a marca resiste a um tipo de ataque cortar e colar na qual a marca de WONG (1998) não é capaz de identificar. Na marca de WONG (1998), caso seja copiado um bloco válido de uma imagem na região exata de seu início e fim (tamanho utilizado na inserção da marca), o mesmo poderá ser colado em outra região da imagem ou até mesmo em outra imagem, obtendo sucesso no ataque. Na marca implementada, tal ataque é identificado com uma alteração e por conseguinte, invalida a marca na região modificada.

Capítulo 6 – Teste e simulações

As imagens para realização dos testes e simulações são cortesias da University of Southern California.

6.1 Inserção da Marca

Na inclusão da marca, são necessários três elementos: o valor da cifra, uma imagem digital de 256 tons de cinza, a quem chamaremos de hospedeiro e uma imagem de marca (necessariamente binária), que será a identificação.

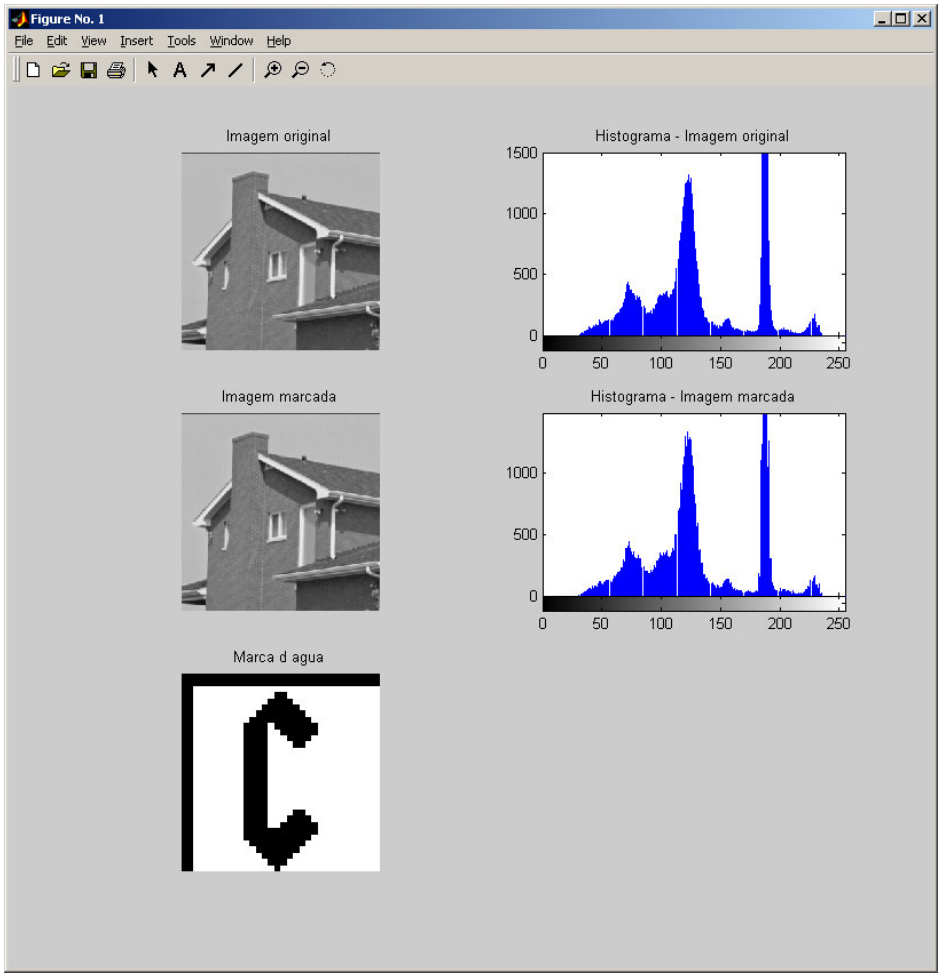

Figura 6.1 – Inclusão da Marca

Observa-se que a imagem não sofreu alteração visual perceptível. Percebe-se também que os histogramas das imagens (original e marcada) também são muito parecidos.

Dessa forma, vê-se que a inclusão da marca não causa prejuízo visual na imagem, podendo as imagens marcadas ser distribuídas para utilização em qualquer fim.

6.1.1 Análise Quantitativa

Foram realizados testes de inclusão de marcas em várias imagens. As semelhanças entre as imagens original e marcada não são apenas visuais. Outros elementos quantitativos indicam a proximidade entre ambas no que diz respeito ao tamanho do arquivo, valores para média, desvio padrão e níveis de cinza presente.

Tabela 6.1 – Resultados Comparativos entre Dado Original e Dado Marcado

ARQUIVO	**DADO ORIGINAL**				**DADO MARCADO**			
	Média*	Desvio Padrão*	Tamanho (em *bytes*)	Tons Cinza	Média*	Desvio Padrão*	Tamanho (em *bytes*)	Tons Cinza
casag.bmp	137,57	46,13	66.614	224	137,56	46,13	22.614	223
elaine.bmp	136,36	46,06	263.222	228	136,35	46,06	263.222	228
caminhao.bmp	107,11	27,06	263.224	144	107,17	27,15	263.222	172

* Resultados colhidos da ferramenta Adobe Photoshop

Dessa forma, chega-se a conclusão que além do mínimo prejuízo visual, a degradação matemática da imagem criada após a inclusão da marca também é ínfima. Portanto, as imagens podem ser utilizadas para quaisquer fim e aplicações.

6.2 Verificação da Marca

O algoritmo consiste em verificar a imagem que foi previamente marcada. Como parâmetros necessários para verificação da marca, temos que indicar qual a imagem marcada e chave pública correspondente à chave da privada utilizada na marcação. Caso exista alguma modificação na imagem marcada, será gerado um ruído aleatório nos blocos que possuem *pixels* modificados, com relação à imagem original.

Para simulação e verificação dos resultados, serão mostradas adiante três situações: Verificação com sucesso, onde não houve nenhum tipo de processamento na imagem marcada; verificação em região alterada, onde será realizada uma alteração uma parte da imagem e por fim a verificação com chave incorreta.

Vale lembrar que nos processos de verificação da marca, não há a necessidade de análises quantitativas das imagens produzidas, pelo fato de ser gerada apenas uma imagem binária de verificação, que se torna sem valor quando utilizada fora do contexto das marcas digitais.

6.2.1 Verificação com Sucesso

Neste caso, não houve nenhuma alteração na marca e a chave está correta. O resultado, portanto, é a marca nítida. Observe a figura 6.2.

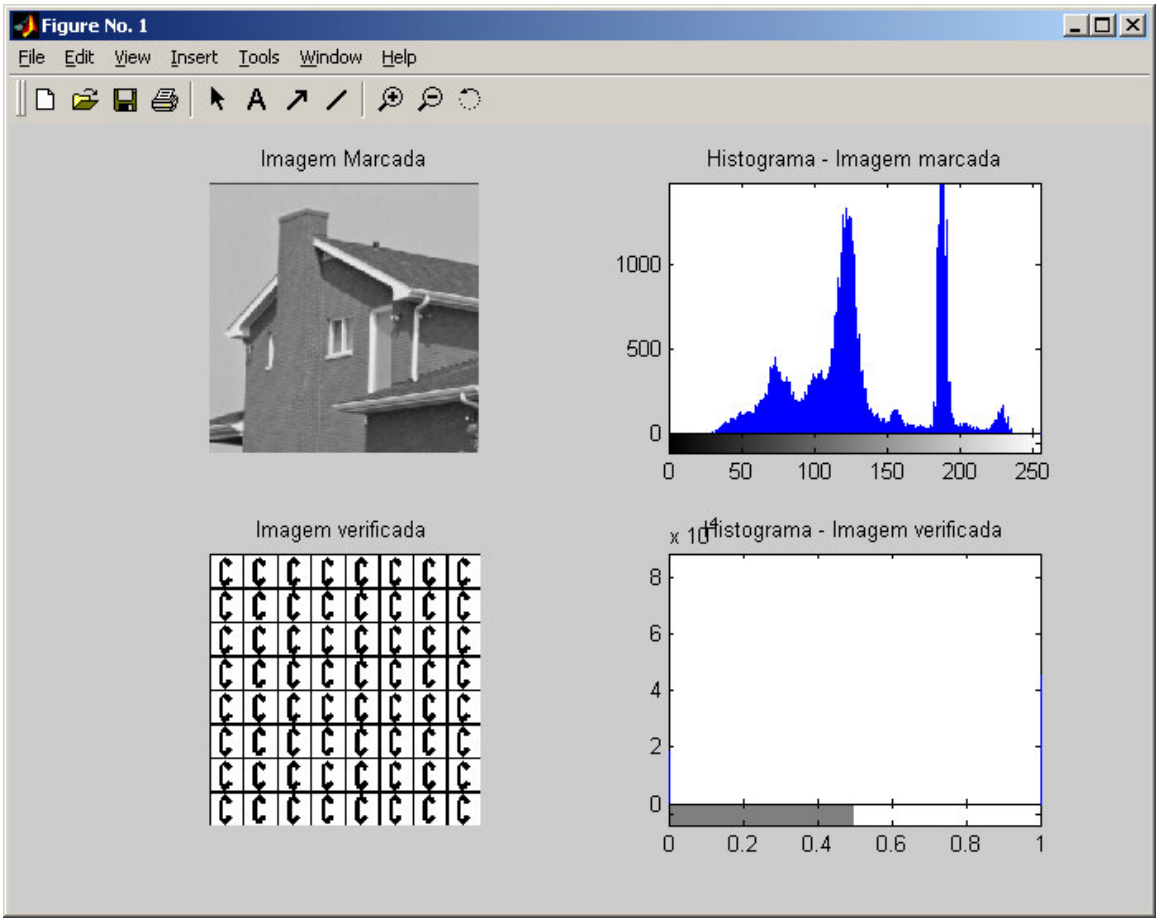

Figura 6.2 – Verificação com Sucesso

6.2.2 Verificação em Região Alterada

Nesta simulação, podemos identificar claramente as regiões onde foram alterados os blocos. O algoritmo retorna o ruído nos locais que foram modificados após a inclusão da marca (blocos onde foram alterados *pixels*, com relação imagem originalmente marcada).

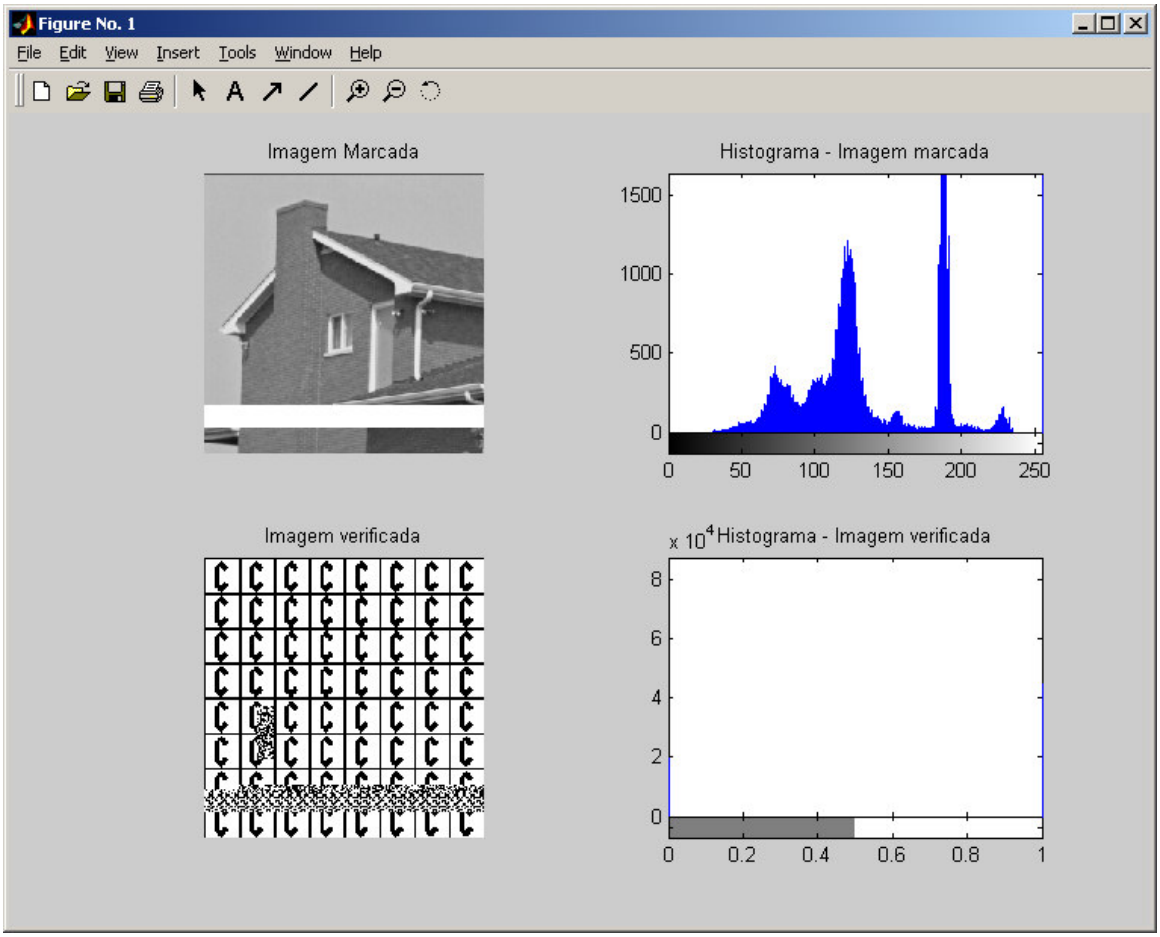

Figura 6.3 – Verificação em Região Alterada

6.2.3 Verificação com Chave Incorreta

Aqui, a situação é mais crítica, pois com a chave inválida fica praticamente impossível recuperar a marca. O retorno de verificação é o mesmo quando a imagem é comprimida, rotacionada ou se nela for realizada qualquer operação que mude características em todos os blocos. Observar a figura 6.4.

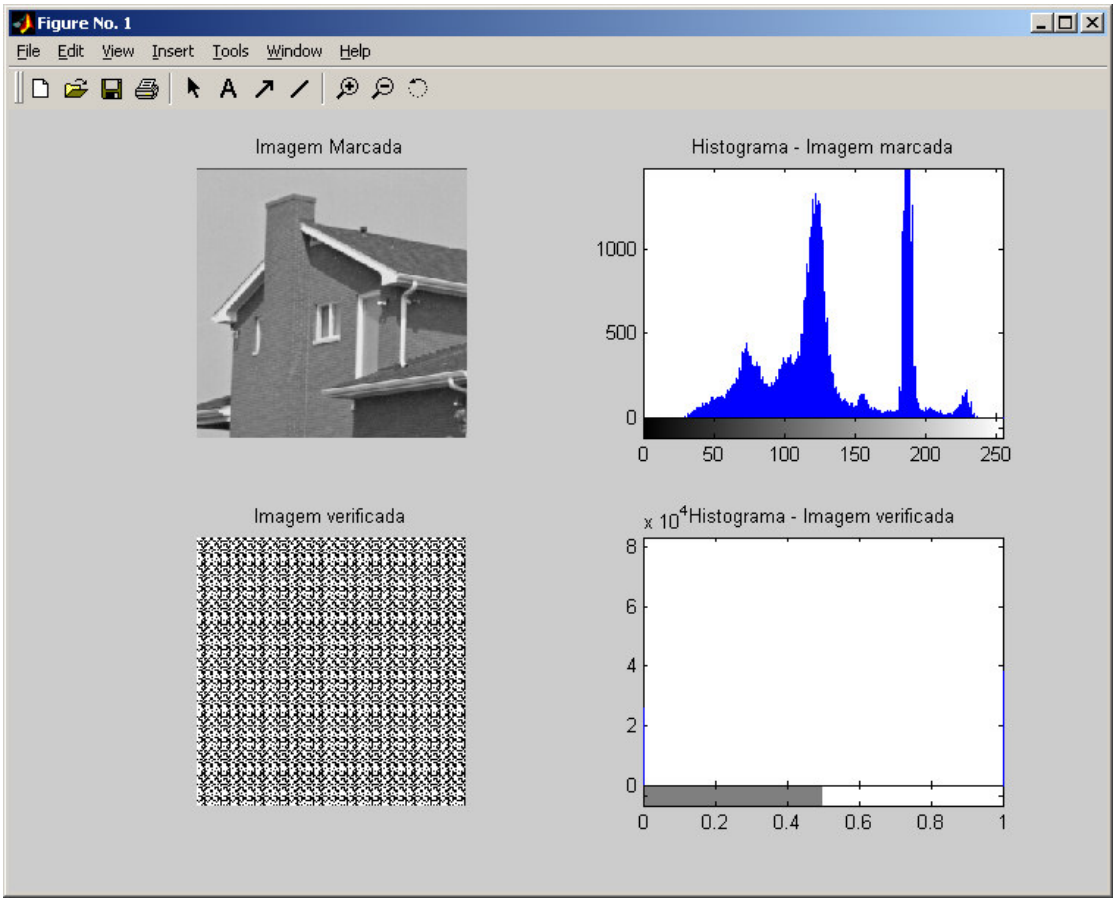

Figura 6.4 – Verificação com Chave Inválida

7. Conclusão

Apesar dos crescentes esforços e pesquisas na área de marcas d'água digitais, ainda não se chegou a uma marca que satisfaça a todas as suas necessidades de robustez e segurança, sem diminuir consideravelmente a qualidade do hospedeiro ou aumentar significativamente o tamanho final do dado marcado.

Outro fato que prejudica os usuários que utilizam o *watermarking* é que ainda não existe uma marca extremamente robusta, que resista a ataques do tipo conversão analógico-digital e vice-versa, como, por exemplo, imprimir uma imagem com uma marca digital invisível embutida e, em seguida, utilizar um *scanner* para obtê-la novamente no computador (porém, sem a marca).

Com relação à criptografia, vemos que, a cada ano, as máquinas estão mais robustas e novos ataques criptoanalíticos são realizados. Para tanto, é importante estar atento aos esquemas de codificação e, sempre que possível, utilizar aqueles que oferecem maior segurança.

Mesmo com todas as adversidades, novas técnicas e esquemas de codificação têm surgido a cada dia, dificultando mais a vida dos piratas reais e virtuais, que obtêm lucro sobre o trabalho árduo dos verdadeiros proprietários desses dados.

Dentro das situações imagináveis na simulação, o protótipo mostrou-se seguro e eficaz nos testes realizados. Pode resistir a um ataque elaborado cortar e colar. Tal mérito deve-se ao recurso de encadeamento de blocos de *hash*.

Observando quantitativamente as imagens que foram geradas, percebe-se que as alterações existentes são mínimas e leva a afirmar que não há prejuízo para o hospedeiro da marca d'água.

O algoritmo implementado neste trabalho pode ser adaptado e produzido comercialmente, desde que seja explorada um pouco mais a segurança, buscando fragilidades, caso existam, no processo de inclusão da marca d'água.

No âmbito deste projeto, os principais quesitos que foram almejados na sua produção foram satisfatoriamente realizados. Os resultados esperados na teoria foram confirmados nas simulações realizadas no protótipo.

7.1 Trabalhos futuros

Como sugestão para trabalhos futuros:

- Implementar o protótipo numa linguagem de desenvolvimento comercial (C Builder, Delphi etc);

- Estender a inserção da marca para outros tipos de imagens (coloridas e/ou outros formatos);

- Utilizar mais *bits* do hospedeiro de maneira a embutir mais informação na imagem.

- Enrobustecer a marca para que ela resista a ataques tais como compressão, rotação e/ou mudanças de escala;

- Utilizar outro esquema de assinatura digital que possua comprimentos menores.

Referências Bibliográficas

ARNOLD, Michael; SCHMUCKER, Martin e WOLTHUSEN, Stephen D. *Techniques and Applications of Digital Watermarking and Content Protection*. Estados Unidos: Artech House, Inc., 2003.

BARNI, Mauro; BARTOLINI, Franco. *Watermarking Systems Engineering. Enabling Digital Assets Security and Other Applications*. Estados Unidos: Marcel Deker, Inc, 2004.

BARRETO, Paulo S. L. M. *Criptografia Robusta e marca d'água Frágeis: Construção e Análise e Algoritmos para Localizar Alterações em Imagens Digitais*. São Paulo: Escola Politécnica da Universidade de São Paulo, 2003.

BORCHARDT, Mauro A. *Uma Arquitetura para a Autenticação Dinâmica de Arquivos*. Paraná: Curitiba, Universidade Católica do Paraná, 2000.

CACHIN, Christian. *An Information-Theoretic Model for Steganography*. Estados Unidos: MIT Laboratory from Computer Science, Massachusetts, 1998.

CASACURTA, Alexandre; OSÓRIO, Fernando; FIGUEROA, Franz; MUSSE, Soraia R. *Computação Gráfica – Introdução*. Rio Grande do Sul: UNISINOS, 1998.

CASTRO, Alexandre R.; LACERDA, Guilherme S. e ZAVALIK, Claudimir. *Um Estudo sobre Criptologia*. URCAMP: 2004.

COLE, Eric. *Hiding in Plain Sight: Steganography and the Art of Covert Communication*. Indiana: Wiley Publishing, Inc., 2003.

DIGIMARC. *Digimarc – Watermarking Guide*. Estados Unidos: Digimarc Corporation, 1999.

FILHO, Luiz Lourenço de Mello. *Percepção Visual Humana*. 1999. Disponível em: http://www.parconsult.com.br/uff/visao.htm. Último acesso em: 03/11/2004.

HAMPIHOLI, Vallabha. *Multiresolution Watermark Based on Wavelet Transform for Digital Images*. Cranes Software International Limited: 2004. Disponível em: http://www.cranessoftware.com/press/articles/?tid=300008&qid=300008. Último acesso em: 13/11/2004.

HARTUNG, Frank; KUTTER, Martin. *Multimedia Watermarking Techniques*. IEEE: Proceedings of the IEEE, vol. 87, nº 7, july, 1999.

HINZ, Marco Antônio Mielke. *Um Estudo Descritivo de Novos Algortimos de Criptografia*. Rio Grande do Sul: 2000.

JASCONE , Fábio Luis Tavares. *Protótipo de Software para Ocultar Texto Criptografado em Imagens Digitais*. Santa Catarina: Blumenau, 2003.

JÄHNE, Bernd. *Digital Image Processing*. Alemanha: Springer, 5ª ed., 2000.

KATZENBEISSER, Stefan; PETITCOLAS, Fabien A. P. *Information Hiding Techniques for Steganography and Digital Watermarking*. Estados Unidos: Artech House, Inc., 2000.

KIM, Hae Yong. *Projeto de Operadores pela Aprendizagem, Difusão Anisotrópica e Marca d'água de Autenticação*. São Paulo: Universidade de São Paulo, 2004.

MAIA, Luiz Paulo. *Criptografia e Certificação Digital*. São Paulo: Fundação Getúlio Vargas, 1999.

MARTIN, Christopher G. *Digital Watermarking Techniques*. Rochester Institute Technology: 2000.

MATHWORKS, The. *Image Processing Toolbox – For Use with Matlab®*. Estados Unidos: The Mathworks, Inc., 3ª versão, 2002.

MORAES, Rosane França de. *Construção de um Ambiente Web com Ferramentas para Estudo de Algoritmos de Criptografia Através do Matlab*. Rio de Janeiro: Universidade Federal do Rio de Janeiro, 2004.

MOHANTY, Saraju P. *Watermarking of Digital Images.* Índia: Bangalore, 1999.

NATARAJAN, Balas. *Robust Public Key Watermarking of Digital Images*. Estados Unidos: Hewlett Packard Company, 1997.

OLIVEIRA, Marcelo V.. *Formato de Arquivo: BMP*. Rio de Janeiro: Universidade Federal Fluminense, Niterói, 2000. Disponível em: http://arcserve.ic.uff.br/~aconci/CV.html. Último acesso em: 03/11/2004.

PETITCOLAS, Fabien A. P. *Watermarking Schemes Evaluation*. Estados Unidos: Microsoft Research, 2004.

PICARD, Justin; ROBERT, Arnaud. *On the Public Key Watermarking Issue.* Suíça: PublicMark Inc., 2001.

PRATT, William K. *Digital Image Processing*. Estados Unidos: Califórina, John Wiley & Sons, Inc., 3ª ed., 2000.

NIKOLAIDIS, N.; PITTAS, I. e VOYATZIS, G. *Digital Watermarking: An Overview.* Grécia: Universidade de Tessalonique, 1998.

SILVA, Rogério G. *Segurança de Redes de Computadores*. Rio de Janeiro: CEFET-RJ, 1998. Disponível em: http://www.cefetrio.hpg.ig.com.br/ciencia_e_educacao/8/trabalhos/seguranca2/seguranca.htm. Último acesso em: 10/11/2004.

TRAINA, Agma Juci Machado; OLIVEIRA, Maria Cristina Ferreira de. *Apostila de Computação Gráfica.* São Paulo: Universidade de São Paulo, 2004. Disponível em: http://gbdi.icmc.usp.br/documentacao/apostilas/cg. Último acesso em: 23/11/2004.

VILELA, Henrique S. *Criptografia*. Rio Grande do Sul: Universidade Católica de Pelotas. Pelotas, 2003.

XEXÉO, Geraldo. *Autenticação de Documentos Digitais por Sistemas Criptográficos de Chave Pública*. Rio de Janeiro: UFRJ, 2000. Disponível em: http://www.cos.ufrj.br/~xexeo/talk/assina~1/assi1.htm. Último acesso em: 13/11/2004.

WALKER, John. *Command Line Message Digest Utility*. Fourmilab Home Page: 2003. Disponível em: http://www.fourmilab.ch/md5/. Último acesso em 22/09/2004.

WONG, Ping W. *A Public Key Watermark for Image Verification and Authentication*. Estados Unidos: Hewlett Packard Company. 1998.

Glossário

Bit: Menor unidade de informação. Pode ser 0 (zero) ou 1 (um).

Bitmap: Mapa de *bits*. A organização destes *bits* formam a imagem.

Byte: Conjunto de 8 *bits.*

Cifra: Conjunto de procedimentos e conjunto de símbolos (letras, nomes, sinais, etc) usados para substituir as letras de uma mensagem para encriptá-la.

CMY (*Cyan, Magenta, Yellow)*: Modelo de cores baseado no ciano, magenta e amarelo.

DES (*Data Encryption Standard*): Algoritmo simétrico de 64 *bits.*

Digital Watermark: Veja marca d'água.

DSA (*Digital Signature Assimetric*): Algoritmo apenas para assinatura digital. É baseado nos padrões do DSS.

DSS (*Digital Signature Standard*): Padrão do governo norte-americano para assinaturas digitais.

GB: Giga byte.

GHz: Abreviatura para Giga Hetz.

Grayscale: Escala de cinza.

Hash: Funções matemáticas que mapeiam uma cadeia de *bits* de qualquer tamanho em um número determinado fixo de *bits*

HBC (*Hash Block Chaining*): Esquema de inclusão de marcas d'água proposto por BARRETO (2003) e KIM (2004) baseado no encadeamento de blocos.

HLS (*Hue, Lightness and Saturation*): Modelo de cores baseado na matiz, brilho e saturação.

IDEA (*International Data Encryption Algorithm*): Algoritmo simétrico de 128 *bits*.

LSB (*Least Significant Bits*): *Bits* menos significativos.

Marca d'água digital: Sinal portador de informação que é embutido num hospedeiro e pode ser extraído posteriormente para se fazer asserção sobre o mesmo.

MB: Mega Byte.

MD5 (*Message Digest* 5): Função que gera um *hash* de 128 *bits.*

NIST (*National Institute of Standards and Technology*): Órgão norte-americano de tecnologia.

Pixel: Elemento ou ponto de uma imagem matricial que define uma cor.

RGB: Abreviatura para *Red* (vermelho), *Green* (verde) e *Blue* (Azul).

RSA (Rivest, Shamir e Adleman): Algortimo de chave pública mais conhecido. Possui chaves de comprimentos variáveis. Associado a uma função de *hash*, pode se tornar um algoritmo de assinatura digital.

SHA (*Secure Hash Algorithm*): Função *hash* de 160 *bits.*

XOR: Função ou-exclusivo. Exemplo: O XOR de dois bits iguais é 0 (zero). Para dois bits diferentes, o resultado é 1 (um).

Watermarking: Processo de inclusão e verificação marcas d'água digitais.

Anexos

Anexo A – Algoritmos

A.1 Inclusão da Marca

Início do código.

```
%============================================================
% Centro Universitario de Brasilia - UNICEUB
% Faculdade de Ciencias Exatas e Tecnologia - FAET
% Curso de Engenharia da Computacao
% Discp: Projeto Final  -   10 Semestre
% Prof. Orientador: Aderlon Queiroz
% Aluno: Cezario R. da C. Neto
%============================================================
%
% Algoritmo de inclusao da marca d'agua
%
% Arquivo de saida (marcado) salvo na pasta c:\proj\matlab\projeto
% com o nome img_marcada.bmp
%============================================================
clear all;
clc;
%============================================================
% Busca Arquivos
%============================================================
%
[a_hosp, h_dir]=uigetfile('*.*', 'Escolha o hospedeiro');
[a_marca, m_dir]=uigetfile('*.*', 'Escolha a marca');

% Le hospedeiro
hosp8=imread([h_dir, a_hosp]);
% Le marca
marca2=imread([m_dir, a_marca]);

% Verifica se os arquivos sao validos
if ~isa(hosp8, 'uint8')
    error('O hospedeiro deve ser uma imagem com 256 níveis de cinza.');
end

if ~isa(marca2, 'logical')
    error('A marca deve ser uma imagem binaria (preto e branco).');
end

%============================================================
% Parametros da Criptografia
%============================================================
%
% Entra com a chave de criframento - Deslocamento de 64
Chave=input('Entre com a chave (Escolha um numero de 1 a 63): ');
Chave=floor(Chave);

%============================================================
```

```
% Parametros do hospedeiro
%=============================================================
%
%hosp=imread('casag.bmp');
hosp=double(hosp8);
%Pega o tamanho do hospedeiro
[M,N]=size(hosp);

%Seta tamanho dos blocos de de marcacao
I=8;
J=8;

%=============================================================
% Monta a marca d'agua do tamanho do hospedeiro
%=============================================================
%
% Le o bitmap que servira como marca
%marca=imread('marca32.bmp');
marca=single(marca2);

% Pega o tamanho original da marca
[r,s]= size(marca);

% Cria matriz para marca

for x=0:r:M-1,
    for y=0:s:N-1,

        %Cria imagem marcadora (De M x N pixels)
        for v=1:r,
            for u=1:s,
                figmarca(v+x,u+y)=marca(v,u);
            end
        end

    end
end

% Salva imagem marcadora
%figmarca=logical(figmarca);
%imShow(figmarca)
%imwrite(figmarca,'test.bmp','bmp');

%=============================================================
% Zera os LSB's do hospedeiro
%=============================================================
% Variavel hosp sem lsb :  hosp_

for u=1:M,
    for v=1:N,
      if mod(hosp(u,v),2)==1,
          hosp_(u,v)=hosp(u,v)-1;
      else
          hosp_(u,v)=hosp(u,v);
      end
    end
end

%=============================================================
```

```
% Varre as imagens figmarca e hosp
% Particiona a imagem em blocos de IxJ
%   1 - Para calculo do hash do bloco
%   2 - XOR com o bloco de figmarca
%=============================================================
for x=0:I:M-1,
    for y=0:J:N-1,

        % Cria bloco IxJ de figmarca
        blocom=figmarca((1+x):(I+x),(1+y):(J+y));

        % Cria bloco IxJ de hosp_
        blocoh=hosp_((1+x):(I+x),(1+y):(J+y));

        % Primeiro bloco (usado na ultima dependencia)
        if (x==0 & y==0)
            bloco1=blocoh;
        end
        %blocoh=double(blocoh);

        % Cria proximo bloco de IxJ de hosp_ (Fazer a dependencia por
        % bloco)
        if (y==N-8 & x==M-8)
            blocoh2=bloco1;
        elseif y==N-8;
            blocoh2=hosp_((x+8+1):(I+x+8),(1):(J));
        else
            blocoh2=hosp_((1+x):(I+x),(1+y+8):(J+y+8));
        end

        blocohdep((1:8),(1:8))=blocoh;
        blocohdep((1:8),(9:16))=blocoh2;

        % Salva o blocoh no arquivo bloco.bin
        blocohdep=uint8(blocohdep);
        % Cria/abre o blocoh.bin. Pega o handle
        fbin=fopen('blocoh.bin','wb');
        % Escreve no arquivo as informacoes de blocoh
        fwrite(fbin,blocohdep,'int8');
        % Fecha o arquivo blocoh.bin
        fclose(fbin);

        % Calcula o hash de bloco.bin e coloca no arquivo blocoh.hsh
(Formato ASCII)
        dos('c:\proj\matlab\projeto\md5 -oc:\proj\matlab\projeto\blocoh.hsh
c:\proj\matlab\projeto\blocoh.bin');

        % Abre o blocoh.hsh para leitura
        fhsh=fopen('blocoh.hsh','rb');
        % Separa apenas a string de hash (ASCII)
        hash=fread(fhsh,32,'uint8');
        % Fecha blocoh.hsh
        fclose(fhsh);

        % Conversor ASCII->Binario
        for u=1:length(hash),
            if hash(u)<65,
                hash(u)=hash(u)-48;
            else
```

```
            hash(u)=hash(u)-55;
        end
    end
    hash=dec2bin(hash);

    % Como os blocos sao de IxJ pega os primeiros valores ate I*J (Se
I=8 e
    % J=8, pega os 64 primeiros bits
    hash=hash(1:I*J);

    %ciph=hash;
    % Coloca o hash na forma de matriz
    mhash=zeros(I,J);
    count=1;
    for v=1:I,
        for u=1:J,
            % Conversao de string para numero
            mhash(v,u)=str2num(hash(count));
            % Incremento do acumulador
            count=count+1;
        end
    end

    % Faz o XOR de blocom com hash de blocoh (mhash)
    mxh = xor(blocom, mhash);

    %=============================================================
    % CRIPTOGRAFIA
    %=============================================================
    %
    % Transforma matriz da marca para linha
    for v=1:I,
        for u=1:J,

            % Conversao de string para numero
            if (v==1 & u==1)
                lmxh=num2str(mxh(v,u));
            else
                lmxh=strcat(lmxh, num2str(mxh(v,u)));
            end

        end
    end

    % Cifra a marca
    mcifra=shift(lmxh,Chave);

    % Volta linha da marca para forma de matriz
    count=1;
    for v=1:I,
        for u=1:J,
            % Conversao de string para numero
            mxh(v,u)=str2num(mcifra(count));
            % Incremento do acumulador
            count=count+1;
        end
    end

    %Insere no bloco de hosp_ a informacao
```

```
        Dmxh = double(mxh);
        Dblocoh = double(blocoh);
        blocoht = Dmxh + Dblocoh;

        for v=1:I,
            for u=1:J,
                figmarcada(v+x,u+y)=blocoht(v,u);
            end
        end


    end
end


%=============================================================
% Salva a imagem marcada
%=============================================================
imShow(figmarcada,256);
imwrite(figmarcada+1,gray,'img_marcada.bmp','bmp');

%=============================================================
% Abre imagem marcada e mostra histogramas
%=============================================================
marcada = imread('img_marcada.bmp');

subplot(3, 2, 1), imshow(hosp8), title('Imagem original');
subplot(3, 2, 2), imhist(hosp8), title('Histograma - Imagem original');
subplot(3, 2, 3), imshow(marcada), title('Imagem marcada');
subplot(3, 2, 4), imhist(marcada), title('Histograma - Imagem marcada');
subplot(3, 2, 5), imshow(marca2), title('Marca d agua');

%=============================================================
```

Fim de código

A.2 Verificação da Marca

Início de código.

```
%============================================================
% Centro Universitario de Brasilia - UNICEUB
% Faculdade de Ciencias Exatas e Tecnologia - FAET
% Curso de Engenharia da Computacao
% Discp: Projeto Final  -   10 Semestre
% Prof. Orientador: Aderlon Queiroz
% Aluno: Cezario R. da C. Neto
%============================================================
%
% Algoritmo de verificacao da marca d'agua
%
%============================================================
clear all;
clc;
%============================================================
% Busca Arquivos
%============================================================
[a_hosp, h_dir]=uigetfile('*.*', 'Escolha o hospedeiro');

% Le hospedeiro
hosp8=imread([h_dir, a_hosp]);

% Verifica se os arquivos sao validos
if ~isa(hosp8, 'uint8')
    error('O hospedeiro deve ser uma imagem com 256 níveis de cinza.');
end

%============================================================
% Parametros da Criptografia
%============================================================
%
% Entra com a chave de criframento - Deslocamento de 64
Chave=input('Entre com a chave (Escolha um numero de 1 a 63): ');
Chave=floor(Chave);


%============================================================
% Parametros do hospedeiro
%============================================================
%
hosp=double(hosp8);

% Le o bitmap que servira como marca
%marca=imread('test.bmp');
%marca=single(marca);

% Pega o tamanho do hospedeiro
[M,N]=size(hosp);

% Seta tamanho dos blocos de marcacao
I=8;
```

```
J=8;

%===============================================================
% Zera os LSB's da imagem marcada // Guarda LSB noutra matriz
%===============================================================
% Variavel hosp sem lsb :  hosp_
% Variavel de lsb's : lsb

for u=1:M,
    for v=1:N,
      if mod(hosp(u,v),2)==1,
          hosp_(u,v)=hosp(u,v)-1;
          lsb(u,v) = 1;
      else
          hosp_(u,v)=hosp(u,v);
          lsb(u,v)=0;
      end
    end
end

%===============================================================
% Varre as imagens hosp_
% Particiona a imagem em blocos de IxJ
%   Calcula hash do bloco para verificacao
%===============================================================
for x=0:I:M-1,
    for y=0:J:N-1,

        % Cria bloco IxJ de hosp_
        blocoh=hosp_((1+x):(I+x),(1+y):(J+y));
        %blocoh=double(blocoh);
        % Cria bloco IxJ de lsb
        blocolsb=lsb((1+x):(I+x),(1+y):(J+y));

        % Primeiro bloco (usado na ultima dependencia)
        if (x==0 & y==0)
            bloco1=blocoh;
        end
        %blocoh=double(blocoh);

        % Cria proximo bloco de IxJ de hosp_ (Fazer a dependencia por
        % bloco)
        if (y==N-8 & x==M-8)
            blocoh2=bloco1;
        elseif y==N-8
            blocoh2=hosp_((x+8+1):(I+x+8),(1):(J));
        else
            blocoh2=hosp_((1+x):(I+x),(1+y+8):(J+y+8));
        end

        blocohdep((1:8),(1:8))=blocoh;
        blocohdep((1:8),(9:16))=blocoh2;

        % Salva o blocoh no arquivo bloco.bin
        blocohdep=uint8(blocohdep);
        % Cria/abre o blocoh.bin. Pega o handle
        fbin=fopen('blocoh.bin','wb');
        % Escreve no arquivo as informacoes de blocoh
        fwrite(fbin,blocohdep,'int8');
```

```
        % Fecha o arquivo blocoh.bin
        fclose(fbin);

        % Calcula o hash de bloco.bin e coloca no arquivo blocoh.hsh
(Formato ASCII)
        dos('c:\proj\matlab\projeto\md5 -oc:\proj\matlab\projeto\blocoh.hsh
c:\proj\matlab\projeto\blocoh.bin');

        % Abre o blocoh.hsh para leitura
        fhsh=fopen('blocoh.hsh','rb');
        % Separa apenas a string de hash (ASCII)
        hash=fread(fhsh,32,'uint8');
        % Fecha blocoh.hsh
        fclose(fhsh);

        % Conversor ASCII->Binario
        for u=1:length(hash),
            if hash(u)<65,
                hash(u)=hash(u)-48;
            else
                hash(u)=hash(u)-55;
            end
        end
        hash=dec2bin(hash);

        % Como os blocos sao de IxJ pega os primeiros valores ate I*J (Se
I=8 e
        % J=8, pega os 64 primeiros bits
        hash=hash(1:I*J);

        %ciph=hash;
        % Coloca o hash na forma de matriz
        mhash=zeros(I,J);
        count=1;
        for v=1:I,
            for u=1:J,
                % Conversao de string para numero
                mhash(v,u)=str2num(hash(count));
                % Incremento do acumulador
                count=count+1;
            end
        end


        %================================================================
        % CRIPTOGRAFIA
        %================================================================
        %
        % Transforma matriz da marca para linha
        for v=1:I,
            for u=1:J,

                % Conversao de string para numero
                if (v==1 & u==1)
                    llsb=num2str(blocolsb(v,u));
                else
                    llsb=strcat(llsb, num2str(blocolsb(v,u)));
                end
```

```
            end
        end

        % Cifra a marca
        mcifra=shift(llsb,Chave);

        % Volta linha da marca para forma de matriz
        count=1;
        for v=1:I,
            for u=1:J,
                % Conversao de string para numero
                blocolsb(v,u)=str2num(mcifra(count));
                % Incremento do acumulador
                count=count+1;
            end
        end

        % Faz o XOR de blocom com hash de blocoh (mhash)
        mxh = xor(blocolsb, mhash);

        %Insere no bloco de hosp_ a informacao
        Dmxh = double(mxh);

        for v=1:I,
            for u=1:J,
                figver(v+x,u+y)=Dmxh(v,u);
            end
        end


    end
end


%============================================================
% Salva a imagem marcada
%============================================================
figver=logical(figver);
imShow(figver)
imwrite(figver,'img_ver.bmp','bmp');

ver=imread('img_ver.bmp');
%hosp8=imread('img_ataque.bmp');
subplot(2, 2, 1), imshow(hosp8), title('Imagem Marcada');
subplot(2, 2, 2), imhist(hosp8), title('Histograma - Imagem marcada');
subplot(2, 2, 3), imshow(ver), title('Imagem verificada');
subplot(2, 2, 4), imhist(ver), title('Histograma - Imagem verificada');

%============================================================
```

Fim de código.

A.3 Algoritmo de Ataque

Início de código.

```
%============================================================
% Centro Universitario de Brasilia - UNICEUB
% Faculdade de Ciencias Exatas e Tecnologia - FAET
% Curso de Engenharia da Computacao
% Discp: Projeto Final  -   10 Semestre
% Prof. Orientador: Aderlon Queiroz
% Aluno: Cezario R. da C. Neto
%============================================================
%
% Algoritmo de ataque a uma figura marcada
%
%============================================================
%
% Com uma figura ja marcada, recortar um bloco de fiel (8x8, 16x16, 32x32
% etc) e inserir em outra parte de mesma imagem
%
%============================================================
clear all;
clc;
%============================================================
% Busca Arquivos
%============================================================
%
[a_hosp, h_dir]=uigetfile('*.*', 'Escolha a imagem a ser atacada');

% Le imagem
hosp=imread([h_dir, a_hosp]);

% Verifica se os arquivos sao validos
if ~isa(hosp, 'uint8')
    error('O hospedeiro deve ser uma imagem com 256 níveis de cinza.');
end

% Pega um bloco da imagem (Neste caso, 32x32 - os primeiros pixels num
quadrado de 32)
bloco = hosp((1:32),(1:32));

% Insere este mesmo bloco noutra posiçao (Aqui, no quadrado limitado pelos
pontos
% (97,97) e (128,128) )
imgsaida = hosp;
imgsaida((97:128),(97:128)) = bloco;
imgsaida = double(imgsaida);

% Salva imagem alterada em img_ataque.bmp
imShow(imgsaida,256);
imwrite(imgsaida+1,gray,'img_ataque.bmp','bmp');

%============================================================
```

Fim de código.

A.4 Função de Deslocamento - *Shift*

A função de deslocamento utilizada na implementação do protótipo pode ser encontrada em http://www.c3.lanl.gov/wibarf/shane.demo/webfiles/matlabfiles.html, com o nome de shift.m.

Início de código.

```
%============================================================
function [g] = shift(d,y)
%SHIFT    Circular shifts left a 1-D vector
% X = SHIFT(A, N) will shift the vector A to the left by N number of
elements.
% The shifted elements are wrapped back around to the end of the vector.

%Shane Crockett  - 11JUN98
%Developed for use in "straightening out" hyberbolic chirp

temp = d(1:y);
g = [d((y+1):length(d)) temp];


%x = length(d) - 1;
%for c = y:x;
%   g(1,c - (y-1)) = d(1,1+c);
%end
%g = [g d(1:y)]


%============================================================
```

Fim de Código.

A.5 Função de *Hash* MD5

Uma função de resumo de mensagem ou função de *hash* é uma assinatura digital compacta para uma cadeia de tamanho arbitrário de uma informação binária. Um algoritmo de resumo ideal nunca poderá gerar a mesma assinatura para dois tipos diferentes de entrada, mas a realização de tal perfeição teórica requer um resumo tão grande quanto o tamanho da informação de entrada. Algoritmos de *hash* têm muito em comum com técnicas de encriptação, mas com finalidades distintas. Uma das aplicações é a verificação de dados que não foram alterados desde quando a assinatura foi publicada. (WALKER, 2003, com adaptações)

A função de *hash* utilizada no protótipo pode ser encontrada no seguinte sítio: http://www.fourmilab.ch/md5/, com o nome de MD5.zip.

Anexo B – Tabelas de Estrutura do Bitmap

Tabela B.1 – Detalhes do Cabeçalho de Arquivo BMP

CAMPO	BYTES	DESCRIÇÃO
BfType	2	Assinatura do arquivo: os caracteres ASCII "BM" ou (42 4D)h. É a identificação de ser realmente BMP.
BfSize	4	Tamanho do arquivo em *bytes*.
BfReser1	2	Campo reservado 1; deve ser ZERO
BfReser2	2	Campo reservado 2; deve ser ZERO
BfOffSetBits	4	Especifica o deslocamento, em *bytes*, para o início da área de dados da imagem, a partir do início deste cabeçalho. - Se a imagem usa paleta, este campo tem tamanho = 14 + 40 + (4 x Número de cores) - Se a imagem for *true color*, este campo tem tamanho = 14 + 40 = 54
Total de *Bytes*	14	

Tabela B.2 – Detalhes do Cabeçalho de Mapa de *Bits* – Informações da Imagem

CAMPO	BYTES	DESCRIÇÃO
BiSize	4	Tamanho deste cabeçalho (40 bytes). Sempre (28)h
BiWidth	4	Largura da imagem em *pixels*
BiHeight	4	Altura da imagem em *pixels*
BiPlanes	2	Número de planos de imagem. Sempre 1
BiBitCount	2	Quantidade de *bits* por *pixel* (1,4,8,24,32) Este campo indica, indiretamente, ainda o número máximo de cores, que é 2 *bits* por *pixel*
BiCompress	4	Compressão usada. Pode ser: 0 = BI_RGB - sem compressão 1 = BI_RLE8 - compressão RLE 8 *bits* 2 = BI_RLE4 - compressão RLE 4 *bits*
BiSizeImag	4	Tamanho da imagem (dados) em *bytes* - Se arquivo sem compressão, este campo pode ser ZERO - Se imagem em *true color*, será tamanho do arquivo (Bfsize) menos deslocamento (BfOffSetBits)
BiXPPMeter	4	Resolução horizontal em *pixels* por metro
BiYPPMeter	4	Resolução vertical em *pixels* por metro
BiClrUsed	4	Número de cores usadas na imagem. Quando ZERO indica o uso do máximo de cores possível pela quantidade de *bits* por *pixel*, que é 2 *bits* por *pixels*
BiClrImpor	4	Número de cores importantes (realmente usadas) na imagem Por exemplo, das 256 cores, apenas 200 são efetivamente usadas Se todas são importantes pode ser ZERO. É útil quando for exibir uma imagem em 1 dispositivo que suporte menos cores que a imagem possui
Total de *Bytes*	40	

Tabela B.3 – Detalhes da Paleta ou Tabela de Cores.

CAMPO	BYTES	DESCRIÇÃO
Blue	1	Intensidade de azul. De 0 a 255
Green	1	Intensidade de verde. De 0 a 255
Red	1	Intensidade de vermelho. De 0 a 255
Reservado	1	Campo reservado deve ser ZERO sempre
Total de *Bytes*	4*	*Quando for utilizada, o tamanho total será de 4 X o número de cores (16 ou 256)

Anexo C – Esquemas Estudados

C.1 Marca de Wong

WONG (1998), propôs uma outra marca d'água de autenticação, baseada em criptografia de chave pública, tornando-se o primeiro trabalho de marca de autenticação de chave pública. De acordo com KIM (2004), "o esquema de Wong consiste, basicamente, em dividir uma imagem em blocos e assinar cada bloco independentemente. Assim, é possível localizar o bloco onde a imagem foi alterada. Quanto menor o tamanho dos blocos, melhor a resolução de localização da alteração. A marca d'água é inserida nos *bits* menos significativos (LSB - *least significant bits*) da imagem. Assim, nas imagens em níveis de cinza, é possível inserir um *bit* em cada *pixel*. Para uma imagem com 256 níveis de cinza (8 *bits* por *pixel*), a alteração dos LSBs é visualmente imperceptível, pois equivale a somar ou subtrair um dos níveis de cinza.

A inserção da marca numa imagem em níveis de cinza, usando o esquema de Wong, pode ser resumida como segue.

Inserção da marca de Wong (KIM, 2004):

• Passo 1: Seja I uma imagem em níveis de cinza a ser marcada, com $N \times M$ *pixels*. Particione I em n blocos I_t $(0 \leq t < n)$ de 8×8 *pixels* (no máximo, os blocos nas bordas podem ser menores). Cada bloco I_t será marcado independentemente.

• Passo 2: Seja B uma imagem-logotipo binária a ser utilizada como marca d'água. Esta imagem é replicada periodicamente ou redimensionada para obter uma imagem suficientemente grande para cobrir I. Para cada bloco I_t, existe um bloco binário correspondente B_t.

• Passo 3: Seja o bloco obtido de I_t^* zerando o bit menos significativo de todos os *pixels*. Usando uma função *hash* $H_t = H(M, N, I_t^*)$ criptograficamente segura, calcule a impressão digital. Aqui, M e N entram na função *hash* para detectar cortes das bordas da imagem (*cropping*).

• Passo 4: Calcule o ou-exclusivo (XOR) de H_t com B_t, obtendo a impressão digital marca d'água .

• Passo 5: Criptografe com a chave privada, gerando assim a assinatura digital S_t do bloco *t*.

• Passo 6: Insira S_t nos LSBs de , obtendo o bloco marcado .

Extração da marca de Wong (KIM, 2004):

O algoritmo de verificação da marca d'água correspondente é direto:

• Passo 1: Seja *I'* uma imagem $N \times M$ em níveis de cinza com marca d'água inserida. Particione *I'* em *n* blocos $I_t^{'}$, como na inserção.

• Passo 2: Seja o I_t^{*} bloco obtido de $I_t^{'}$ limpando os LSBs de todos os *pixels*. Usando a mesma função *hash* escolhida para a inserção, calcule a impressão digital tal $H_t = H(M, N, I_t^{*})$.

• Passo 3: Retire os LSBs de $I_t^{'}$ e decriptografe o resultado usando a chave pública, obtendo o bloco decriptografado D_t.

• Passo 4: Calcule o ou-exclusivo (XOR) de H_t com D_t, obtendo o bloco de checagem *Ct*.

• Passo 5: Se C_t e B_t (o bloco t da imagem-logotipo) forem iguais, a marca d'água está verificada. Caso contrário, a imagem marcada foi alterada no bloco *t*.

C.2 *Hash Block Chaining* – HBC

Nos trabalhos de BARRETO (2003) e KIM (2004) são apresentadas algumas fraquezas criptoanalíticas na marca de WONG (1998) e então eles propõem uma nova marca, denominada *Hash Block Chaining*, que em sua tradução significaria encadeamento de blocos de *hash*.

A solução proposta para aumentar a segurança consiste em "alimentar a função de *hashing H* com os blocos vizinhos de I_t^{*}, além do próprio bloco I_t^{*} (veja a figura C.1). Neste caso, se um bloco $I_t^{'}$ for alterado, a verificação da assinatura irá falhar em todos aqueles blocos que dependem de $I_t^{'}$, além do próprio bloco $I_t^{'}$. Portanto, um número tão pequeno

quanto possível de dependências é desejável para uma localização acurada da alteração na imagem. Idealmente, uma única dependência por bloco.”. (BARRETO, 2003 e KIM, 2004)

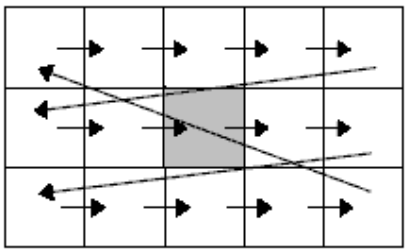

Figura C.1 – Uma Dependência por Bloco – *Raster* (KIM, 2004)