Centro Universitário de Brasília

MARCO ANTONIO DA SILVA SOARES

MARCA D'ÁGUA DE AUTENTICAÇÃO PARA IMAGENS TIFF RGB

Brasília/DF, Junho de 2006

MARCO ANTONIO DA SILVA SOARES

MARCA D'ÁGUA DE AUTENTICAÇÃO PARA IMAGENS TIFF RGB

Monografia apresentada ao Centro Universitário de Brasília – UniCEUB como um dos pré-requisitos para obtenção do título de Bacharel em Engenharia da Computação.

Profº Orientador: MC Aderlon M. Queiroz

Brasília/DF, Junho de 2006

MARCO ANTONIO DA SILVA SOARES

MARCA D'ÁGUA DE AUTENTICAÇÃO PARA IMAGENS TIFF RGB

Monografia apresentada ao Centro Universitário de Brasília – UniCEUB como um dos pré-requisitos para obtenção do título de Bacharel em Engenharia da Computação.

MEMBROS DA BANCA EXAMINADORA

MEMBROS DA BANCA	**ASSINATURA**
1. COORDENADOR DO CURSO Prof º. Abiezer Amarília Fernandes	
2. PROFESSOR ORIENTADOR Prof º. Aderlon M. Queiroz	
3. PROFESSOR EXAMINADOR Prof º. Fabiano Mariath D'Oliveira	
4. PROFESSOR EXAMINADOR Prof º. Miguel Archanjo Bacellar	
MENÇÃO FINAL:	

Brasília/DF, 5 de Julho de 2006

RESUMO

Atualmente preocupa-se muito em preservar o direito de propriedade de imagens digitais. Uma forma de garantir a autenticidade e integridade dessas imagens é a utilização de marcas d'água de autenticação. Marca d'água de autenticação é uma imagem visualmente imperceptível quando embutida em outra imagem, e ao ser extraída carrega consigo informações acerca da imagem marcada, em relação a propriedade e a possíveis alterações. Este trabalho apresenta a implementação de um aplicativo desenvolvido no ambiente e linguagem MATLAB® para inserção de marca d'água de autenticação em imagens no formato TIFF RGB 24 *bits* sem compressão. O aplicativo ainda apresenta recursos comparativos através de histogramas e dados estatísticos a fim de se verificar a preservação da qualidade da imagem após a inserção da marca d'água de autenticação.

Palavras-chave: Imagem digital, marca d'água, processamento digital de imagem, criptografia.

ABSTRACT

Nowadays, preserving the copyright of the digital images is something that worries a lot. One way to guarantee the authenticity and the integrity of these images is the use of authentication watermarks. Authentication watermark is an imperceptible image visually when embedded in another image. This image after being extracted carries the information concerning the marked image, which is related to the property and the possible alterations. This work presents the implementation of a program developed in the MATLAB® environment and language for insertion of authentication watermark in no compressed TIFF RGB 24 bits images. The program also presents comparative resources through histograms and statistical data in order to verify if the image quality has been preserved after the insertion of the authentication watermark.

Keywords: Digital image, watermarking, digital image processing, cryptography.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

BIT	*Binary Digital* (Menor unidade de informação binária)
BMP	*Bitmap* (Mapa de Bits)
CMY	*Cyan, Magenta, Yellow* (Ciano, Magenta, Amarelo)
DCT	*Discrete Cosine Transform* (Transformada Discreta do Cosseno)
DFT	*Discrete Fourier Transform* (Transformada Discreta de Fourier)
HLS	*Hue, Lightness, Saturation* (Matiz, Luminosidade, Saturação)
HSV	*Hue, Saturation, Value* (Matiz, Saturação, Luminância)
JPEG	*Joint Photographic Experts Group* (Formato de Imagem)
LSB	*Least Significant Bit* (Bit menos significativo)
LZW	*Lempzel-Ziv-Welch* (Algoritmo de compressão de imagem)
PIXEL	*Picture Element* (Elemento da imagem)
RGB	*Red, Green, Blue* (Vermelho, Verde, Azul)
RLE	*Run-Length Enconding* (Algoritmo de compressão de imagem)
TIFF	*Tagged Image File Format* (Formato de Imagem)

1. INTRODUÇÃO

1.1. Tema e delimitação do tema

Marca d'água digital[1]: utilização de marca d'água digital para garantir a autenticidade e integridade de imagens digitais 2-D estáticas.

1.2. Problema de pesquisa

É possível inserir uma marca d'água digital em uma imagem colorida RGB de formato TIFF (sem compressão) preservando sua qualidade?

1.3. Justificativas

As limitações do projeto[2] anterior, no qual a inserção da marca d'água digital era feita somente em imagens de tons de cinza no formato Bitmap, levou a elaboração deste projeto, com a proposta de estender às suas aplicações, possibilitando também, a utilização de imagens coloridas RGB no formato TIFF (sem compressão), ou seja, um formato de imagem que é bem mais utilizado e de melhor qualidade, e por esse motivo, necessite de um recurso que garanta a integridade e a autenticidade da imagem.

[1] Sinal portador de informação visualmente imperceptível, embutido numa imagem digital. (BARRETO, 2003).
[2] Monografia de Projeto Final de autoria de Cezário Rodrigues da Costa Neto, orientado pelo Prof. MC Aderlon M. Queiroz, realizada em 2004 na Faculdade de Ciências Exatas e Tecnologia (FAET) do Centro Universitário de Brasília (UniCEUB), para obtenção do título de Bacharel em Engenharia da Computação.

1.4. Objetivos

1.4.1. Geral

Dar continuidade ao projeto *Processamento Digital de Imagens – Implementação de Watermarking*, estendendo suas aplicações para um formato de imagem mais complexo, colorido e de melhor qualidade.

1.4.2. Objetivos específicos

- Implementar o software aplicativo, para que seja capaz de inserir marcas d'água digitais em imagens de formato TIFF sem compressão;
- Utilizar além de imagens em tons de cinza, imagens coloridas RGB-24 *bits*;
- Realizar a inserção da marca d'água digital em até três LSB's de cada *pixel*, sem que a imagem hospedeira[3] sofra degradação.

1.5. Limitações da pesquisa

No projeto apresentado, algumas limitações apareceram. A primeira delas foi o fato do formato TIFF ser bem mais complexo que o Bitmap, devido a isso não foi possível estender as aplicações do projeto a outros formatos com qualidade similar, como o JPEG ou até mesmo o TIFF com

[3] Também chamada de imagem marcada, é aquela que contém a marca d'água digital embutida.

compressão. O segundo aspecto que limitou a pesquisa foi o método escolhido para a realização da inserção da marca d'água digital, no caso o LSB, que a embute diretamente nos *pixels*, ou seja, no domínio do espaço. E para a utilização de outros formatos citados de imagens seria necessário o uso de métodos no domínio da freqüência, utilizando-se as transformadas DFT ou DCT para, diferentemente do outro, embutir a marca d'água nos coeficientes alterados.

1.6. Estrutura do trabalho

Após este capítulo introdutório, o Capítulo 2 expõe a metodologia de pesquisa utilizada na execução deste trabalho. O Capítulo 3 apresenta uma revisão da literatura sobre os temas: processamento de imagem digital; marca d'água digital; formato TIFF; criptologia e assinaturas digitais. No Capítulo 4 apresenta o protótipo desenvolvido. No Capítulo 5 são apresentados os testes, simulações e resultados obtidos. E por fim, expõe as considerações finais e conclusões do trabalho, e também as recomendações para trabalhos futuros.

2. METODOLOGIA

Segundo Lúcia Santaella (2001, p. 132) "método de pesquisa científica não é outra coisa do que a elaboração, consciente e organizada, dos diversos procedimentos que nos orientam para realizar o ato reflexivo [...]".

E Antonio Carlos Gil (1999, p. 19) define que "a observação constitui, sem dúvida, importante fonte de conhecimento" relacionando este conhecimento com o significado etimológico da palavra ciência.

No caso do projeto proposto a metodologia utilizada na sua elaboração foi um estudo comparativo que "procede pela investigação de [...] fenômenos ou fatos, com vistas a ressaltar as diferenças e similaridades entre eles" (GIL, 1999, p. 34), procurando-se identificar essas particularidades entre a imagem original e a imagem marcada.

A abordagem feita foi a quantitativa, considerando-se que tudo pode ser quantificável, o que significa traduzir em números informações para classificá-las e analisá-las, utilizando-se recursos e técnicas estatísticas como média, desvio-padrão e coeficiente de correlação.

Além disso, foi feito um levantamento bibliográfico sobre as teorias referentes ao tema delimitado, pois, de acordo com Gil (1999, p. 36), essas teorias "desempenham importante papel metodológico na pesquisa" e citando Popper ele afirma que as teorias são "redes estendidas para capturar o que chamamos 'o mundo', para racionalizá-lo, explicá-lo e dominá-lo" (in GIL, 1999, p. 36). Partiu-se então de dados e informações

já elaboradas e publicadas, dando suporte a uma análise mais profunda para resolução do problema proposto.

A pesquisa também teve um caráter experimental ao se determinar um objeto de estudo, selecionando-se as variáveis que seriam capazes de influenciá-lo e definindo-se as formas de controle e de observação dos efeitos que estas variáveis produziriam neste objeto, com a realização de testes e simulações da implementação proposta.

Para finalizar, Santaella (2001, p. 139 e 140) afirma que

> motivação principal das pesquisas aplicadas, por seu lado, está na sua contribuição para resolver um problema. Para tal, ela aplicará conhecimentos já disponíveis, mas das aplicações podem resultar não apenas a resolução do problema que a motivou, mas também a ampliação da compreensão que se tem do problema, ou ainda a sugestão de novas questões a serem investigadas. […] O conhecimento está em um *continuum* cuja origem e cujo fim serão eternamente desconhecidos.

3. FUNDAMENTAÇÃO TEÓRICA

3.1. Percepção visual

As características da percepção visual, isto é, características psicofísicas do sistema visual humano, são fundamentais no desenvolvimento de sistemas que envolvem processamento digital de imagens. Principalmente nos critérios de fidelidade da imagem processada.

Devemos compreender então, a importância da percepção visual humana no desenvolvimento de algoritmos de processamento digital de imagens. Luiz Lourenço Filho (1999) coloca que

> é interessante ressaltar que vários estudos tentam cada vez mais desvendar o funcionamento do Sistema Óptico Ocular, tanto a medicina pelo seu aspecto fisiológico como outras áreas que procuram simular este Modelo. Na Computação, podemos citar a Computação Gráfica no processo de produção de imagens que serão interpretadas por este Sistema, também a área de Processamento de Imagens que necessita da simulação deste Sistema para obter estas imagens (câmeras de vídeo, satélites, etc.) e após interpretá-las.

Porém, sabemos que o olho humano é constituído por um sistema muito complexo e segundo Luiz Lourenço Filho (1999),

> ainda não existe nenhum modelo genérico de percepção visual passível de ser aplicado na prática. O nosso conhecimento de como funcionam os mecanismos de percepção visual nos animais tampouco é suficiente para que possamos aplicar algum mecanismo de 'engenharia reversa', utilizando, por exemplo, técnicas de redes neurais para modelar ou imitar a percepção visual biológica.

Mas sabemos que o funcionamento de uma câmera fotográfica pode ser considerado um modelo aproximado do olho humano, onde a luz

passa através da pupila, que é controlada pela íris e cristalino é responsável pelo foco da luz sobre a superfície da retina.

O olho humano é um sensor de sinais de radiação eletromagnética, isto é, ele é sensível à radiação eletromagnética que varia de 400nm (violeta) a 700nm (vermelho), apresentando um pico de sensibilidade ao redor de 555nm.

Na retina encontram-se dois tipos de receptores visuais, os cones e os bastonetes. Eles são células nervosas que recebem os estímulos luminosos, sendo que, em um olho normal pode ter aproximadamente 6,5 milhões de cones e 130 milhões de bastonetes.

Os dois tipos de fotoreceptores têm funções distintas. Os bastonetes são sensíveis à luz muito fraca, sendo responsável pela visão noturna. Já os cones são sensíveis a níveis mais altos de iluminação, sendo assim, responsáveis pela visão diurna e também pela distinção das cores e detalhes.

Segundo Nelson Mascarenhas e Flávio Velasco (1989, p. 1.9) a acuidade visual pode ser definida como

> a habilidade do sistema visual em detectar bordas agudas. A sensibilidade do ser humano em distinguir variações muitas lentas ou muito rápidas de tom de cinza é melhor do que sua capacidade em distinguir variações médias. Isso pode ser caracterizado por uma resposta em freqüência do sistema visual que tem um pico na ordem de 6 a 10 ciclos por grau.

Eles ainda destacam a existência de três tipos básicos de cones na retina, que correspondem a picos de absorção de luz nas regiões do azul, verde e vermelho.

> A existência dos três tipos de cones oferece uma base fisiológica para a teoria tricromática[4] da visão humana. Embora o olho humano seja capaz de discriminar simultaneamente apenas a área de 20 ou 30 tons de cinza, sua habilidade em discriminar cores é muito maior.

3.2. Processamento digital de imagens

A evolução tecnológica dos computadores contribuiu muito no avanço da área de processamento digital de imagens. Isso porque, as aplicações dessa área geralmente requerem muito do processamento e da memória dos computadores. Com os computadores modernos cada vez mais rápidos e com mais disponibilidade de memória, os sistemas de processamento digital de imagens se tornaram mais eficientes e baratos. Com isso muitas áreas, como pesquisa espacial, medicina, meteorologia, dentre outros, vêm utilizando sistemas de processamento digital de imagens.

O termo processar, de acordo com Kenneth Castleman (1979, p. 7), "é o ato de sujeitar algo a um processo", e que um processo

> é uma série das ações ou das operações que conduzem a um resultado desejado. Assim uma série das ações ou das operações é executada em cima de um objeto para alterar sua forma de uma maneira desejada.

Castleman (1979, p. 8) ainda afirma que, "o termo processamento digital de imagem, entretanto, é usado largamente para processamento e análise".

[4] Teoria da percepção cromática da visão desenvolvida por Thomas Young em 1801, em que define a percepção cerebral a partir de 3 cores fundamentais (vermelho, verde e azul).

Segundo Nelson Mascarenhas e Flávio Velasco (1989, p. 1.5) "por processamento digital de imagens entendem-se a análise e a manipulação de imagens por computador".

Eles afirmam também que "a finalidade deste processamento é extrair informações de imagens e transformar a imagem (por exemplo, aumentar o contraste, realçar bordas) de tal modo que a informação seja mais facilmente discernível por um analista humano".

No processo de análise de imagens, segundo Mascarenhas e Velasco,

> a entrada do processamento é uma imagem, enquanto a saída é uma "descrição" (não-pictória) da imagem. Este processo pode ser entendido como de "redução de dados" em que se diminui o volume de dados mantendo o conteúdo de informação relevante para uma dada aplicação.

Já na manipulação de imagens em geral, Mascarenhas e Velasco, afirmam que "tanto as entradas quanto as saídas são imagens".

Eles destacam duas classes de transformações de imagens: as transformações radiométricas onde "os valores de níveis de cinza dos pontos da imagem são alterados, sem modificação da geometria"; e as transformações geométricas, onde "a geometria da imagem é alterada, mantendo-se o máximo possível os valores dos níveis de cinza".

3.3. Imagem digital

Segundo Jonas Gomes e Luiz Velho (1994, p. 131), a imagem

> é o resultado de estímulos luminosos produzidos por um suporte bidimensional. Essa é a percepção da imagem no universo físico no qual habitamos, seja ela o resultado de

> um processo intermediário, como por exemplo uma fotografia, ou ultimamente através da projeção do mundo tridimensional na retina do olho humano.

Uma imagem pode ser definida matematicamente. Segundo Nelson Mascarenhas e Flávio Velasco (1989, p. 1.1)

> pode-se definir imagem como uma função I(x,y), bidimensional, definida, numa certa região. Para a maioria das imagens a região de definição é um subconjunto limitado do plano, e os valores assumidos pela função são números reais limitados e não-negativos.

Mas por comodidade, eles afirmam que "a imagem é definida num retângulo [0, r] x [0, s], e os valores tomados estão contidos no intervalo [0, t]. Ao valor I(x,y) da imagem no ponto (x,y) dá-se o nome de nível de cinza".

Já Paulo Sérgio Barreto (2003, p. 22) define uma imagem matematicamente como: "uma imagem de dimensões *m* x *n*, *c* cores e *k* níveis é uma função $A: Z_n \text{ x } Z_m \Rightarrow Z_2^{c^k}$ ".

Ele destaca que, "o número *c* de cores é geralmente igual a três, mas em certos casos pode ser substancialmente expandido pela inclusão de faixas não visuais do espectro eletromagnético, como ocorre em imagens meteorológicas e astronômicas".

De acordo com Mascarenhas e Velasco (1989, p 1.1), uma imagem digital ou discreta

> é aquela onde a função só é definida numa grade regular de pontos de forma (m.dx, n.dy); dx e dy são os intervalos nas direções x e y; e (m,n) são inteiros nos intervalos [0, m-1] e [0, n-1]. Os valores assumidos nestes pontos são também múltiplos de uma dada quantidade dz, ou seja, da forma k.dz, onde k está no intervalo [0, k-1].

Uma imagem digital pode ser obtida também de uma imagem não digital ou contínua através dos processos de amostragem e de quantização, onde, segundo eles, "o processo de amostragem consiste em discretizar o domínio de definição da imagem, ou seja, escolher os valores dx e dy da grade", e o processo de quantização, "consiste em escolher um valor múltiplo de *dz* para a imagem em cada ponto (m.dx, n.dy)".

Eles definem ainda que no trabalho com imagens digitais,

> "é de costume ignorar os intervalos de discretização *dx*, *dy* e *dz*. Assim uma imagem digital pode ser vista como uma matriz de pontos com N linhas e M colunas, cada ponto pertencente ao intervalo [0, k-1]. É usual, também, que N, M e K sejam potências de 2.

Quando o número de níveis de k é igual a 2, isto é, tem-se somente dois níveis (0 e 1), a imagem é chamada binária. Mascarenhas e Velasco destacam que

> imagens binárias têm grande importância, pois são fáceis de ser obtidas, ocupam menos espaço em computador e podem ser manipuladas através de operadores lógicos que, em geral, estão disponíveis diretamente nas instruções dos computadores.

3.3.1. Os elementos da imagem

Uma imagem é constituída, essencialmente, das coordenadas dos *pixels* e da informação de cor de cada *pixel*. Para Gomes e Velho (1994, p. 137)

> esses dois elementos estão diretamente relacionados com a resolução espacial e resolução de cor da imagem. O número de componentes do *pixel* é a dimensão do espaço de cor utilizado. Dessa forma, cada *pixel* em uma imagem monocromática tem uma única correspondente.

3.3.1.1. Pixel

A abreviatura pixel de *picture element* quer dizer "componente da imagem". O *pixel* é o menor elemento de uma imagem digital e contém informações tais como, luminosidade e cor. Estas informações são representadas com um número inteiro positivo (de 0 a 255) que indicam a intensidade na escala de tons de cinza, para imagens monocromáticas, ou na escala da cor associada (RGB), para imagens coloridas. Então, o número armazenado contém as instruções para que cada pixel possa ser reproduzido com as informações de cor e brilho específicas.

Os pixels têm o formato quadrado e são posicionados lado a lado, formando a matriz da imagem.

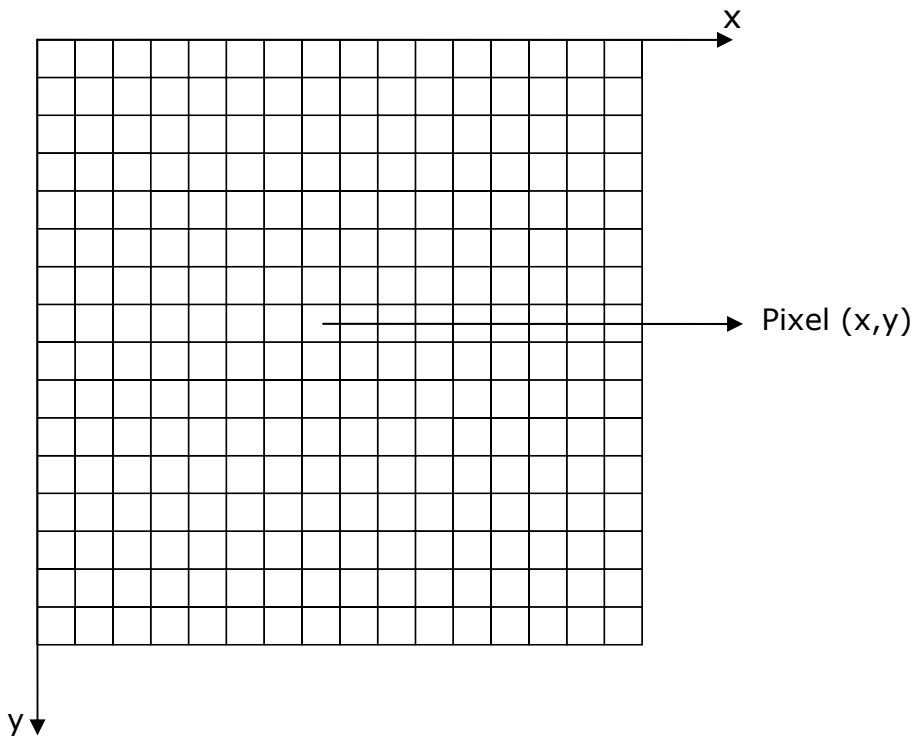


Figura 3.1 – Representação matricial de uma imagem monocromática

Fonte: [FALCÃO, 2003]

Porém, o *pixel* pode ser pensado como uma unidade lógica e não física. O tamanho de um *pixel* depende de como é configurada a resolução de um monitor. Se a resolução for configurada para o máximo, o tamanho físico do pixel será igual ao tamanho físico do *dot pitch*[5].

3.3.1.2. Resolução

A resolução indica a qualidade de uma imagem, isto é, a qualidade depende fundamentalmente da quantidade de informação contida na imagem e do grau de detalhes desta informação que é perceptível pelo olho humano. Esses parâmetros estão associados aos conceitos de resolução espacial e profundidade da imagem.

A resolução espacial é representada pela quantidade N x M *pixels* de uma imagem, por exemplo, em uma imagem de 20cm em *x* e 15cm em *y,* temos N=150 x M=200, portanto a resolução espacial da imagem é 150x200 *pixels*.

A profundidade da imagem representa a quantidade de *bits* por *pixel* que a imagem possui. Por exemplo, em uma imagem em tons de cinza, cada *pixel* pode ter associado um valor de cinza entre 0 e 255, que requer no máximo 8 *bits* para ser armazenado na memória do computador, sendo assim, a profundidade dessa imagem é de 8 *bits* por *pixel*. Já em imagens coloridas, cada *pixel* possui um valor associado a cada uma das

[5] *Dot pitch* é uma medida estabelecida pelos fabricantes de monitores de vídeo. É uma limitação de *hardware* da ordem de centésimos de milímetros (0.28mm), que define a menor distância física entre dois pontos. Quanto menor esta medida, maior a quantidade de *pixels* na horizontal e na vertical e melhor a qualidade das imagens que podem ser geradas em resolução máxima.

três cores (RGB), ou seja, 8 *bits* para cada cor, dessa forma, a profundidade dessa imagem é de 24 *bits* por *pixel*.

3.3.1.3. Cor

As cores primárias, vermelho, verde e azul, são utilizadas para produzir as outras cores, isto é, a partir de uma combinação das cores primárias, ou da composição de duas combinações, são formadas as demais cores. Agma Traina e Maria Cristina de Oliveira (2006, p 96) destaca que "não existe um conjunto finito de cores primárias que produza todas as cores visíveis, mas sabe-se que uma grande parte delas pode ser produzida a partir de 3 primárias".

Para descrever as diferentes características da cor que são percebidas pelo olho humano, existem sistemas distintos, sendo que cada um abrange um aspecto diferente relacionado à cor. Os principais sistemas de cores são: RGB, CMY, HSV e HLS, sendo que eles podem ser aditivos ou subtrativos.

Traina e Oliveira (2006, p 96) definem os modelos aditivos e subtrativos da seguinte forma

> Nos modelos aditivos (por exemplo, RGB e XYZ), as intensidades das cores primárias são adicionadas para produzir outras cores. […] Pode-se pensar que o branco é a mistura das intensidades máximas das 3 cores primárias aditivas (vermelha, verde e azul). […] Nos modelos subtrativos (por exemplo, o CMY), as cores são geradas subtraindo-se o comprimento da onda dominante da luz branca, por isso, a cor resultante correspondente à luz que é refletida. […] Pode-se pensar que o preto é a combinação das 3 cores subtrativas (ciano, magenta e amarelo). A quantidade de preto numa cor é indicada pela diferença

> entre o branco e a intensidade máxima das 3 cores primárias aditivas. E, da mesma forma, a quantidade de branco numa cor é indicada pela diferença entre o preto e a intensidade mínima das 3 cores primárias aditivas.

A figura 2 mostra, a combinação das três cores primárias aditivas com intensidades máximas formando a cor branca e também, a cada combinação de duas cores primárias é gerada uma cor subtrativa.

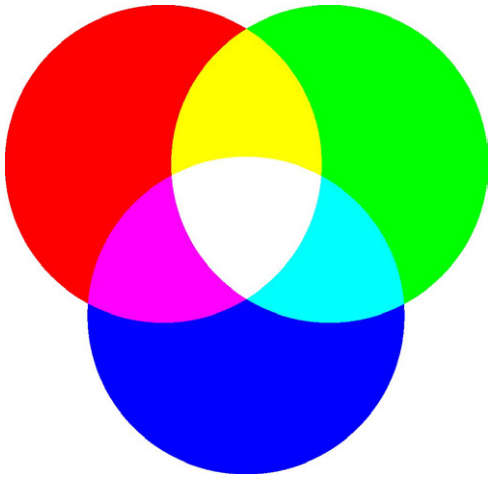


Figura 3.2 – Mistura das cores aditivas RGB

Fonte: [TRAINA; OLIVEIRA, 2006]

Na figura 3 pode ser observado a combinação das cores subtrativas.

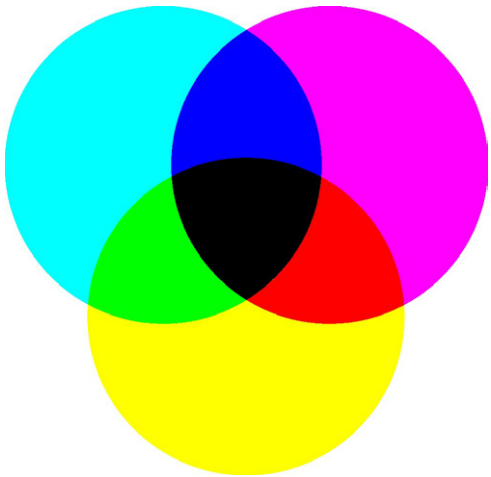


Figura 3.3 – Mistura das cores subtrativas CMY

Fonte: [TRAINA; OLIVEIRA, 2006]

O modelo RGB (*Red, Green, Blue*) é o sistema padrão que foi definido em 1931, pela Comissão Internacional de Iluminação – CIE[6]. É um modelo aditivo de cores primárias, baseado na teoria tricromática.

3.4. Formato TIFF

O TIFF (*Tagged Image File Format*) é um formato de arquivo *raster*[7] para imagens digitais criado pela *Aldus Corporation* e controlado pela *Adobe Systems Inc*. É um formato padronizado que permite o armazenamento e intercâmbio de informações gráficas em formato *raster*, podendo ser utilizado em várias plataformas como o PC (IBM), o Macintosh e estações de trabalho UNIX.

As principais vantagens do formato TIFF são:

- a sua estrutura é conhecida por uma larga variedade de aplicações;
- independe da arquitetura de um computador, sistemas operacionais e plataformas gráficas;
- podem ser ajustados às características de um scanner, monitor ou impressora.

As principais desvantagens do TIFF é o fato de que o usuário pode fornecer novos atributos (*tags*) em uma imagem, impossibilitando a interpretação correta dos dados por outro sistema de leitura de imagens,

[6] CIE (Comissão Internacional de Iluminação) é o órgão responsável pela padronização na área de fotometria e colorimetria.

[7] *Raster* ou bitmap – tipo de imagem onde a informação gráfica é descrita como um conjunto de *pixels*, normalmente organizados da maneira como são visto no monitor e que cada *pixel* tem uma localização específica e uma cor atribuída a ele.

além também de requerer um grande esforço computacional para decodificação dos dados.

O TIFF inclui esquemas de compressão que permite aos usuários escolherem o que mais se adapta às suas aplicações. Podendo ser comprimido com RLE, LZW ou JPEG.

A estrutura de um arquivo de formato TIFF é definida por uma seqüência de 8 *bits* (1 *byte*), onde os *bytes* são numerados de 0 a N. Este arquivo pode atingir 232 *bytes* de comprimento. Ele possui um arquivo *header* e um ou mais diretórios de arquivo de imagem (IFD) que contém os dados de imagem.

O *header* contém 8 *bytes* com as seguintes informações:

- *Bytes* 0-1: um código que contém a ordem do *byte*. MM (arquitetura Motorola) onde os *bytes* são armazenados em ordem de maior para menor significância ou II (arquitetura Intel) onde os *bytes* são armazenados na ordem inversa, isto é, de menor para maior significância.
- *Bytes* 2-3: um número código identificando o arquivo como TIFF;
- *Bytes* 4-7: um ponteiro para o diretório de arquivo de imagem. O IFD possui uma série de ponteiros que indicam o início do arquivo, tipo e comprimento dos dados.

3.5. Marca d’água digital

O avanço tecnológico da computação e a evolução na área de processamento digital de imagens contribuíram para uma crescente demanda pela proteção aos direitos da propriedade de imagens digitais.

A solução encontrada foi a de esconder informações dentro de outras informações, através da utilização de marcas d’águas digitais.

As marcas d’águas digitais surgiram a partir da esteganografia, uma técnica utilizada para esconder mensagens dentro de conteúdos, para que fossem enviadas sem serem percebidas.

Hae Yong Kim (2004, p. 127) define que “uma marca d’água é um sinal portador de informação embutido no dado digital que pode ser extraído mais tarde para fazer alguma asserção sobre o dado hospedeiro”.

Outra definição feita por Paulo Sérgio Barreto (2003, p. 24)

> Uma marca d’água digital é um sinal portador de informação visualmente imperceptível, embutido numa imagem digital. A imagem que contém a marca d’água é dita imagem marcada ou hospedeira.

Portanto, o principal objetivo na utilização de marcas d’águas digitais é a manutenção da integridade e autenticidade das imagens digitais. Pois segundo Barreto (2003) as marcas d’águas “são capazes não só de detectar, mas também de localizar alterações numa imagem marcada com uma resolução previamente estabelecida”.

Barreto também destaca algumas propriedades acerca da marca d’água:

- Armazenada na própria imagem;

- Visualmente imperceptível quando inserida;
- Visualmente significativa quando extraída;
- Irreproduzível por terceiros não autorizados;
- Capaz de localizar alterações maliciosas na imagem hospedeira com resolução suficiente;
- Publicamente verificável;
- Indelével por manipulação não autorizada;
- Resistente a certas operações de processamento de imagens (como mudar o nível de compressão);
- Aplicável a formatos com e sem perdas, e a imagens binárias, em níveis de cinza e coloridas;
- Eficiente em tempo de processamento e espaço de armazenamento.

Porém, desenvolver um algoritmo que atenda todas essas propriedades é muito difícil na prática.

> A maioria dos algoritmos de marca d'água concentra-se apenas em alguns poucos destes requisitos, e portanto, são aplicáveis numa gama restrita de circunstâncias. Outra dificuldade reside no fato que os requisitos acima podem ser contraditórios, no sentido de que algumas operações em imagens são ora exigidas, ora proibidas. (BARRETO, 2003, p. 25)

As marcas d'águas digitais são classificadas, segundo a dificuldade em removê-las do hospedeiro, em robustas e frágeis.

Uma marca d'água diz-se robusta,

> se sua remoção de uma imagem marcada deteriora a qualidade da imagem resultante a ponto de destruir seu conteúdo visual. Mais precisamente, a correlação entre uma imagem marcada e uma marca d'água robusta nela inserida

> permanecerá detectável mesmo após um processamento digital, enquanto a imagem resultante do processamento continuar visualmente reconhecível e identificável com a imagem original. Por esse motivo, marcas d'água robustas são normalmente utilizadas para a verificação de propriedade ou de *copyright* de imagens. (BARRETO, 2003, p. 26)

Ele também destaca que uma marca d'água frágil

> pode ser removida sem afetar substancialmente o aspecto visual da imagem resultante. Contudo, é possível construir marcas d'água frágeis cuja remoção sempre pode ser detectada. Esta propriedade torna tais marcas d'água úteis para fins de autenticação e atestação de integridade de imagens. Em outras palavras, uma marca d'água frágil fornece uma garantia de que a imagem marcada não seja despercebidamente editada ou adulterada, e também de que efetivamente provém da origem declarada ou assumida. Neste sentido, o termo "frágil" é infeliz para qualificar esses algoritmos, sendo mantido apenas por motivos históricos. (BARRETO, 2003, p. 26)

3.5.1. Métodos

Os métodos utilizados para esconder informações em imagens digitais podem ser divididos em três categorias:

- Métodos no Domínio Espacial (*Spatial-domain methods*): Engloba todos os métodos de esteganografia em que as informações são embutidas diretamente no *pixel* da imagem hospedeira.
- Métodos no Domínio da Freqüência (*Frequency-domain methods*): Nesta categoria, a imagem hospedeira é inicialmente transformada para o domínio da freqüência. Em seguida, usando o DFT (*Discrete Fourier Transform*) ou o DCT

(*Discrete-Cosine transform*) ou o *Spread Spectrum* etc, a informação é embutida nos coeficientes alterados.

- Mascaragem da Percepção (*Perceptual Masking Systems*): Este grupo explora as propriedades do sistema visual humano para camuflar de forma eficiente a informação. Antes de ser efetuado o processo de esteganografia, é realizada uma avaliação da imagem hospedeira para determinar quais são as áreas mais significantes para a percepção visual humana. As informações decorrentes da análise definirão que regiões serão utilizadas para realizar a inserção da informação.

3.5.2. Marca D'água de Wong

Em 1998, Ping Wong desenvolveu um algoritmo para obtenção de marcas d'água digitais. O esquema proposto consiste, basicamente, em dividir uma imagem em blocos e assinar cada bloco independentemente. A marca d'água digital é inserida nos LSB's da imagem, ou seja, no *bit* menos significativo de cada *pixel.*

3.5.3. LSB – *Least Significant Bit*

O método LSB (*Least Significant Bit*), isto é, *bit* menos significativo, é o mais comum utilizado para armazenar informações em imagens digitais. Consiste em inserir a informação no *bit* menos significativo de cada *pixel*, ou seja, nas imagens em níveis de cinza de 8 *bits*, é possível

inserir um *bit* em cada *pixel*, e nas imagens RGB de 24 *bits*, é possível inserir três *bits* em cada *pixel*.

3.5.4. HBC – *Hash Block Chaining*

O HBC (*Hash Block Chaining*), que significa encadeamento de blocos de *hash*, foi proposto por Barreto (2003) e Kim (2004), como uma forma de resolver as fraquezas criptoanalíticas do algoritmo de Wong. O método consiste em alimentar a função de *hashing H* com os blocos vizinhos, dessa forma, se um bloco for alterado, a verificação da assinatura irá falhar em todos os blocos dependentes.

3.6. Criptologia

Criptologia é uma palavra de origem grega, kryptós (escondido, oculto) + *lógos* (estudo, ciência), ou seja, é uma ciência que estuda os códigos. A criptologia é dividida ainda em duas áreas: a criptografia e a criptoanálise.

A criptografia, também de origem grega, *kryptós* (escondido, oculto) + *grápho* (grafia, escrita), é

> a arte ou a ciência de escrever em cifra ou em código; em outras palavras, é um conjunto de técnicas que permitem tornar incompreensível uma mensagem originalmente escrita com clareza, de forma a permitir normalmente que apenas o destinatário a decifre e compreenda. (CLÁUDIO LUCCHESI, 1986).

Mas para que o destinatário possa decifrar e compreender a mensagem, é necessário que ele possua uma chave, isto é, uma informação secreta que seja capaz de decifrar a mensagem.

Porém, terceiros podem interceptar a mensagem criptografada, e sem possuir a chave, podem obter a mensagem original ou até mesmo a chave, através da criptoanálise.

A criptoanálise, que em grego é *kryptós* + análysis, ou seja, escondido + decomposição, é "a arte ou a ciência de determinar a chave ou decifrar mensagens sem conhecer a chave" (LUCCHESI, 1986).

A criptografia computacional é utilizada para garantir:

- Confidencialidade: impede que pessoas não autorizadas tenham acesso ao conteúdo da informação;
- Integridade: garante que o conteúdo da informação não foi alterado;
- Autenticidade: garante a identidade de quem produziu a informação;
- Não-repudiação: impede que alguém negue o envio ou recepção de uma determinada informação.

Os sistemas criptográficos são classificados de acordo com a forma como são realizadas a codificação e a decodificação, ou seja, podem ser baseados em chave simétrica ou baseados em chave assimétrica.

3.6.1. Criptografia simétrica

A criptografia simétrica é baseada no uso de uma mesma chave tanto para cifrar como para decifrar a mensagem trocada entre o remetente e o destinatário. Essa chave deve ser conhecida somente por eles, porém o algoritmo criptográfico pode ser público.

Portanto, para que a chave seja de conhecimento somente do remetente e do destinatário, é necessário encontrar um meio seguro para compartilharem a mesma chave criptográfica. Quanto a mensagem, uma vez que ela já foi criptografada e cifrada por uma chave, ela pode ser transmitida por qualquer meio, pois caso ela seja interceptada, não seja facilmente decifrada sem o conhecimento da chave.

A figura 3.4 ilustra um modelo de funcionamento de criptografia baseada em chave simétrica.

Os principais algoritmos simétricos são:

- DES (*Data Encryption Standard*) – é o algoritmo simétrico mais disseminado no mundo. Foi criado pela IBM em 1977, com um tamanho de chave de 56 *bits,* considerado pequeno, pois já foi quebrado por "força bruta" em 1997 em um desafio lançado na Internet.
- 3DES – Uma variação do DES, utiliza 3 ciframentos em seqüência, empregando chaves com tamanho de 112 ou 168 *bits*. É seguro, porém muito lento para ser um algoritmo padrão.

- IDEA (*International Data Encryption Algorithm*) – Criado em 1991, segue as mesmas idéias do DES, porém com execução mais rápida.
- AES *(Advanced Encryption Standard)* – É o padrão atual para ciframento recomendado pelo NIST (*National Institute of Standards and Technology*). Trabalha com chaves de 128, 192 e 256 *bits*.

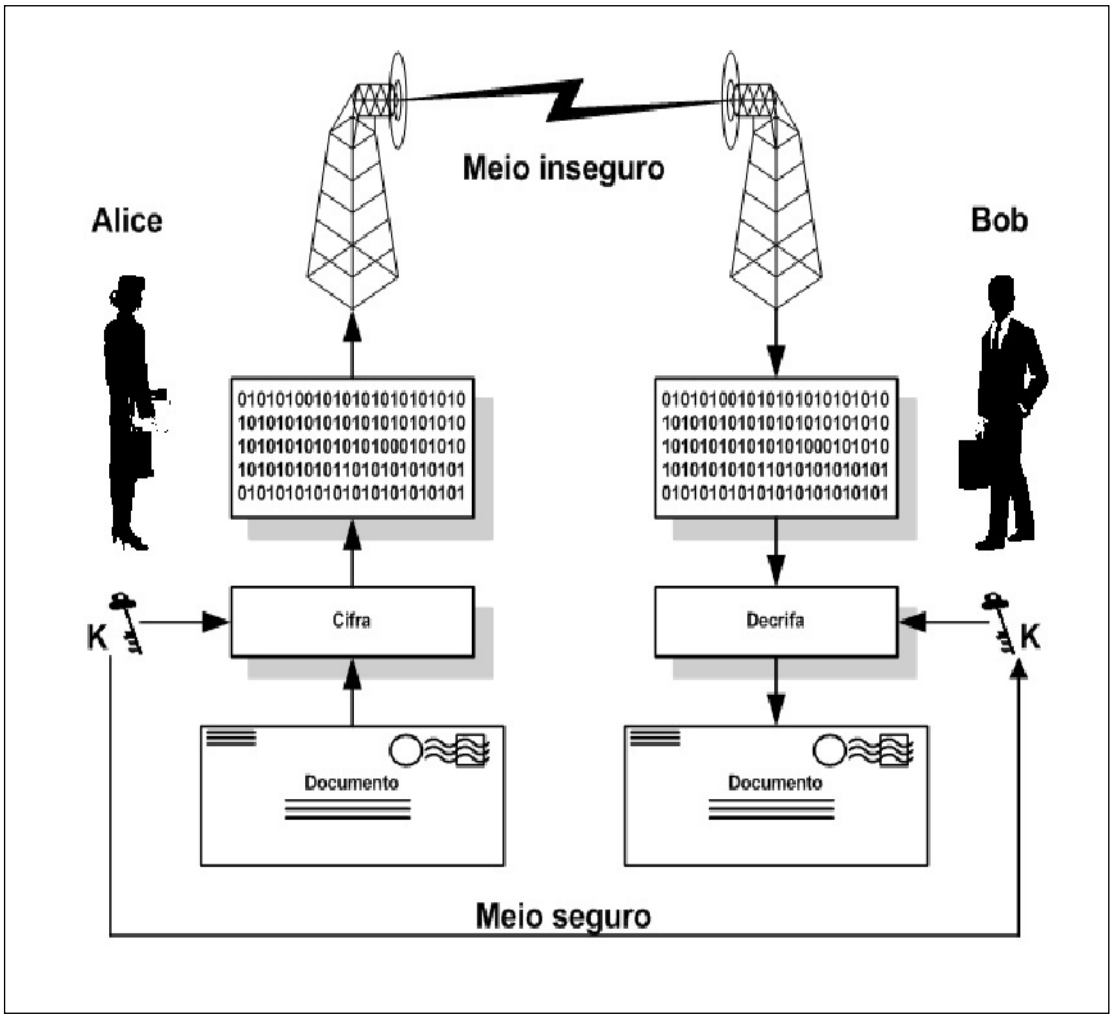


Figura 3.4 – Modelo de criptografia baseada em chave simétrica

Fonte: [BORCHARDT, 2002]

A criptografia simétrica apresenta algumas vantagens e desvantagens quando comparados a outros sistemas.

Segundo Borchardt, as principais vantagens dos sistemas criptográficos simétricos são:

> muito rápidos, permitindo a cifragem rápida de grandes volumes de dados. As chaves são relativamente pequenas bem como relativamente simples, podendo gerar cifradores muito robustos.

Como desvantagens ele destaca que

> a comunicação é ponto-a-ponto e cada par de pontos comunicantes precisa manter uma chave secreta própria, desconhecida dos demais pontos. A gerência de chaves pode ser complicada em redes onde existam muitos pontos. Recomenda-se a troca freqüente das chaves. Não se garante a identidade da origem.

3.6.2. Criptografia assimétrica

A criptografia assimétrica surgiu da necessidade de resolver os problemas do sistema simétrico. Ela é baseada no conceito de par de chaves, sendo uma chave privada e outra pública. Qualquer uma das chaves é utilizada para cifrar uma mensagem e a outra para decifrá-la. As mensagens cifradas com uma das chaves do par só podem ser decifradas com a outra chave correspondente. A chave privada deve ser mantida secreta, enquanto a chave pública disponível livremente para qualquer interessado. A figura 3.5 apresenta um modelo assimétrico.

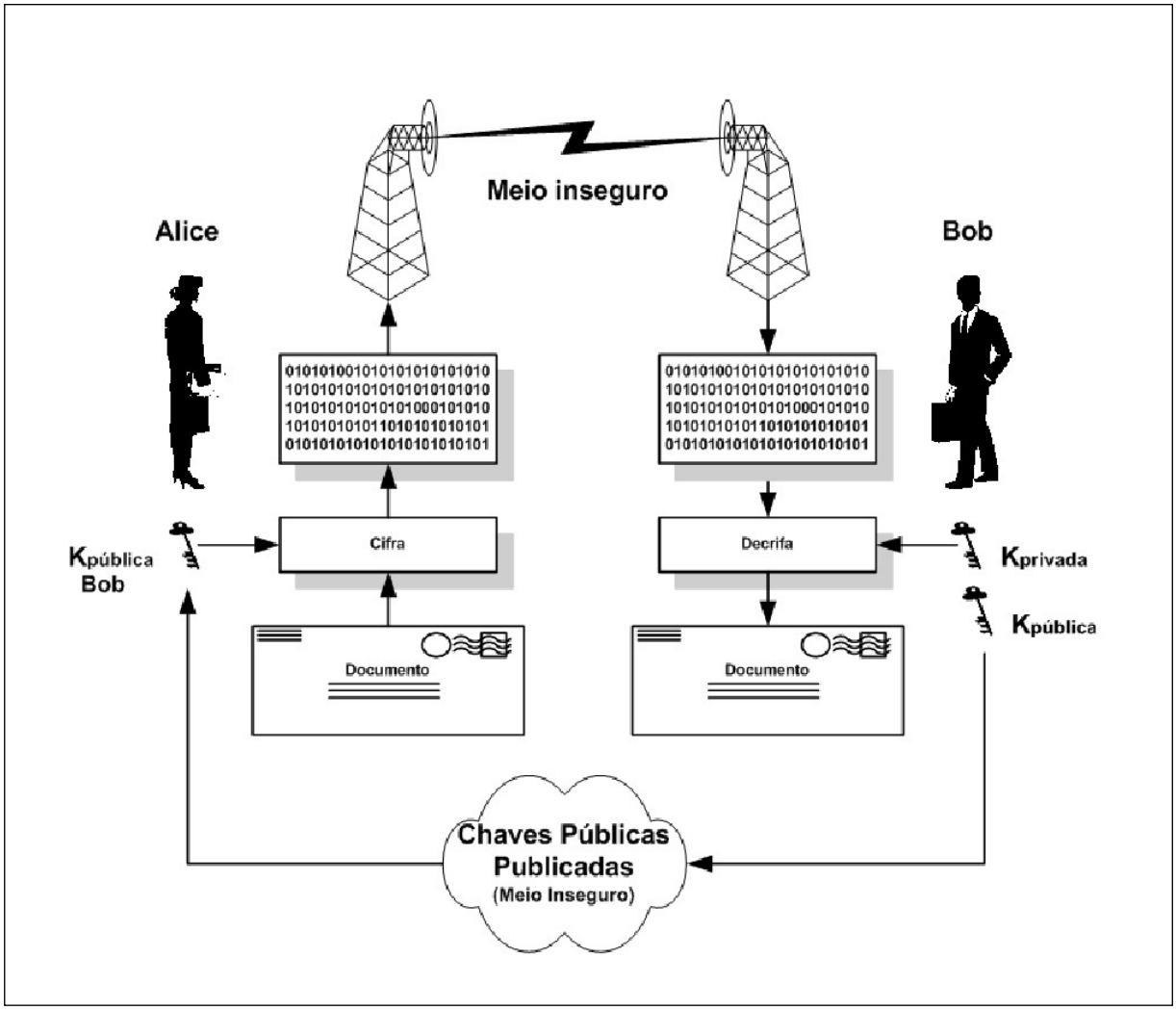


Figura 3.5 – Modelo de criptografia baseada em chave assimétrica

Fonte: [BORCHARDT, 2002]

Os principais algoritmos criptográficos assimétricos são:

- RSA (*Rivest, Shamir e Adleman*) – Foi criado pelo grupo Ron Rivest, Adi Shamir e Len Adleman, em 1977 no MIT. É, atualmente, o algoritmo de chave pública mais amplamente utilizado, além de ser uma das mais poderosas formas de criptografia de chave pública conhecidas até o momento. O RSA utiliza números primos.

- ElGamal – Esse algoritmo envolve a manipulação matemática de grandes quantidades numéricas, porém sua matemática difere da utilizada no RSA. Sua segurança é baseada na dificuldade de se calcular valores na ordem de centenas de *bits*.

Os sistemas criptográficos assimétricos apresentam algumas vantagens quando comparados a outros sistemas. Sendo elas:

> somente a chave privada precisa ser secreta, com isto a gerência de chaves é mais simples. Dependendo do modelo usado o par de chaves pode permanecer o mesmo por longos períodos. Os mecanismos de assinatura digital são relativamente eficientes. O número de chaves é pequeno se comparado ao modelo do sistema simétrico. (BORCHARDT, 2002)

Como principais desvantagens, Borchardt destaca que

> São de execução lenta, o que os torna ineficientes para grandes volumes de dados. O tamanho das chaves é grande para oferecer a mesma dificuldade computacional quando comparado ao sistema simétrico.

3.6.3. Resumos digitais

Resumo digital é definido por Borchardt como

> 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.

Esse recurso criptográfico, também conhecido como função de *Hash,* é normalmente utilizado para a detecção de modificações em dados, pois apresenta as seguintes propriedades

> "mão única", ou seja, dada uma entrada deve ser facilmente computado o valor de resumo, mas o cálculo no sentido oposto deve ser muito difícil, ou mesmo impossível. Para uso em criptografia elas também devem apresentar a propriedade de que uma pequena perturbação na entrada gera uma grande mudança na saída. Também devem apresentar resistência à colisão, ou seja, dadas duas entradas diferentes, suas saídas devem ser distintas. (BORCHARDT, 2002)

Os principais algoritmos de resumos digitais são:

- MD5 (*Message Digest 5*) – Inventada por Ron Rivest, é uma função de espalhamento unidirecional que produz um valor *hash* de 128 *bits*, para uma mensagem de entrada de tamanho arbitrário. Seu algoritmo é público, e foi projetado para ser rápido, simples e seguro.
- SHA-1 (*Secure Hash Algorithm*) – Inventada pela NSA (*National Security Agency*), também é uma função de espalhamento unidirecional, porém gera um valor hash de 160 bits, a partir de um tamanho arbitrário de mensagem, por esse motivo, torna-se mais seguro do que o MD5.

3.6.4. Assinatura digital

Assinatura digital é um processo criptográfico utilizado para garantir a autenticidade da origem e a integridade de uma determinada mensagem. Segundo Borchardt,

> assinar digitalmente um documento consiste em usar um algoritmo criptográfico fazendo uso de uma chave secreta, de tal forma que se possa verificar se o documento recebido pelo destino não foi adulterado após sua criação, garantindo assim a autenticidade da origem e a integridade do documento.

O modelo básico de funcionamento da assinatura digital é ilustrado na figura 3.6, no qual Alice quer enviar uma mensagem assinada digitalmente para Bob. Primeiro Alice teria que publicar sua chave pública em algum lugar acessível a Bob. Depois, para criar e assinatura digital e cifrar a mensagem, Alice aplicaria uma função de *hash* sobre a mensagem, dessa forma obtendo o resumo digital desta mensagem, que em seguida seria cifrado com sua chave privada. Este resumo cifrado juntamente com a mensagem original formariam o corpo da mensagem assinada, que seria enviada para Bob por algum meio. Bob retiraria do corpo da mensagem assinada, a mensagem original e sobre ela aplicaria a mesma função de *hash* utilizado por Alice, obtendo assim o valor atual do resumo digital da mensagem. Depois, Bob utilizando-se da chave pública de Alice, decifraria o resumo assinado que está no corpo da mensagem assinada e obteria o valor do resumo da mensagem enviada por Alice. Através da comparação dos valores de ambos os resumos, o atual e o da Alice, Bob pode saber se a mensagem foi adulterada ou não, isto é, se forem iguais a mensagem é a mesma, ou seja, está íntegra e pode-se assegurar que a mensagem foi gerada por Alice.

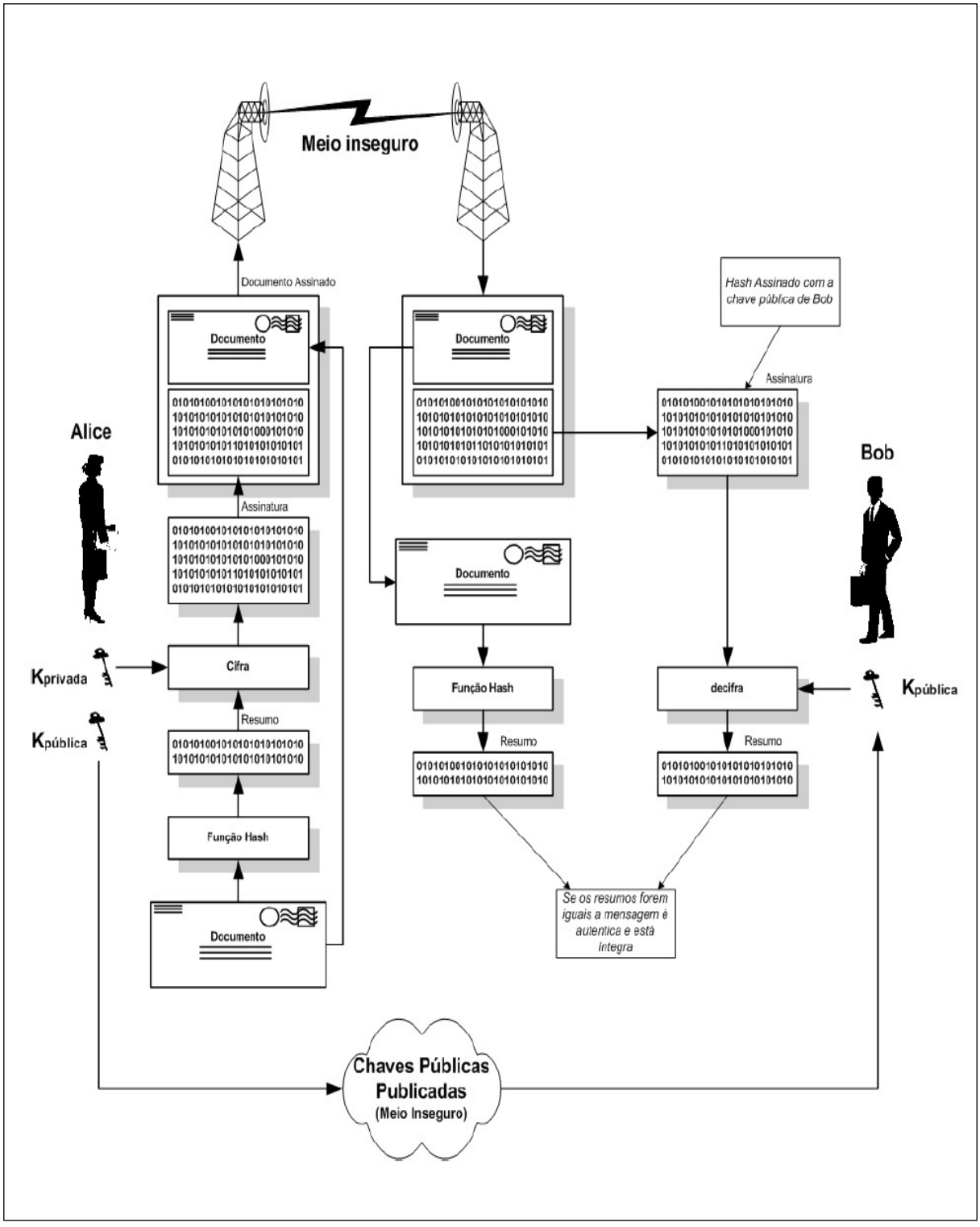


Figura 3.6 – Modelo de assinatura digital

Fonte: [BORCHARDT, 2002]

3.6.5. Cifras por deslocamento

A cifra por deslocamento é um recurso criptográfico utilizado como chave. Esse método consiste em rearranjar posicionalmente os elementos de uma mensagem, isto é, ele não substitui os elementos por outros, ele apenas desloca suas posições a partir de um valor pré-determinado.

As figuras 3.1 e 3.2, ilustram o procedimento utilizado na cifra por deslocamento. Neste exemplo, a mensagem original (figura 3.1) tem 10 elementos e o valor da cifra é 4, ou seja, a mensagem será cifrada por um deslocamento de 4 posições (figura 3.2).

Posição	1	2	3	4	5	6	7	8	9	10
Elemento	1	1	0	1	0	1	1	0	1	0

Figura 3.1 – Informação Original

Fonte: Autor

Posição	1	2	3	4	5	6	7	8	9	10
Elemento	1	0	1	0	1	1	0	1	0	1

Figura 3.2 – Informação cifrada por deslocamento de 4 posições

Fonte: Autor

Para obtenção da informação original, é necessário realizar outro deslocamento, mas neste caso, o valor da cifra será a diferença do tamanho da informação pelo valor da cifra utilizado no processo de cifragem da mensagem.

3.6.5.1. Função de Deslocamento – *Shift*

A função *Shift* foi desenvolvida por Shane Crockett em 1998. O seu funcionamento é bem simples, mas bem eficiente e rápido para a aplicação no projeto. A partir dos seguintes dados: um vetor *v* qualquer e um valor inteiro *n,* no máximo do tamanho desse vetor, a função deslocam os elementos de *v* em *n* posições.

4. PROTÓTIPO

4.1. Ferramenta utilizada

A implementação do projeto foi realizada com a utilização do *software* MATLAB®, versão 6.5, *Release* 13.

O MATLAB® além de ser uma plataforma de desenvolvimento, é também uma linguagem de programação e relativamente de fácil utilização. Através do seu conjunto de ferramentas para as mais diversas aplicações, o MATLAB® apresenta um ambiente rico para a visualização de dados graças a sua poderosa capacidade gráfica.

4.2. Métodos e recursos utilizados

Para o desenvolvimento da implementação do projeto, foram utilizados os seguintes métodos e recursos:

- Algoritmo de Wong (WONG, 1998);
- HBC – *Hash Block Chaining* (BARRETO, 2003 e KIM 2004);
- LSB – *Least Significant Bit;*
- Função de Deslocamento *Shift*[8] (SHANE CROCKETT, 1998);
- Função de *Hash* MD5[9] – *Message Digest* 5 (RONALD RIVEST, 1991)

[8] Função shift.m, disponível em http://www.c3.lanl.gov/wibarf/shane.demo/webfiles/matlabfiles.html
[9] Função MD5, disponível em http://fourmilab.ch/md5

4.3. Marca proposta

4.3.1. Inclusão da marca d'água digital

O processo de inserção da marca d'água digital, em cada plano RGB, de uma imagem TIFF RGB de 24 *bits* pode ser descrito simplificadamente da seguinte forma:

- Passo 1: Escolha uma imagem I TIFF RGB de 24 *bits* a ser marcada, com $\boldsymbol{N} \times M$ *pixels*. Zerar os LSB's de I obtendo I^*;
- Passo 2: Particione 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 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 LSB's de I^*_t, obtendo o bloco marcado. O resultado será a imagem marcada I'.

O fluxograma da figura 4.1 ilustra o processo de inclusão da marca d'água digital.

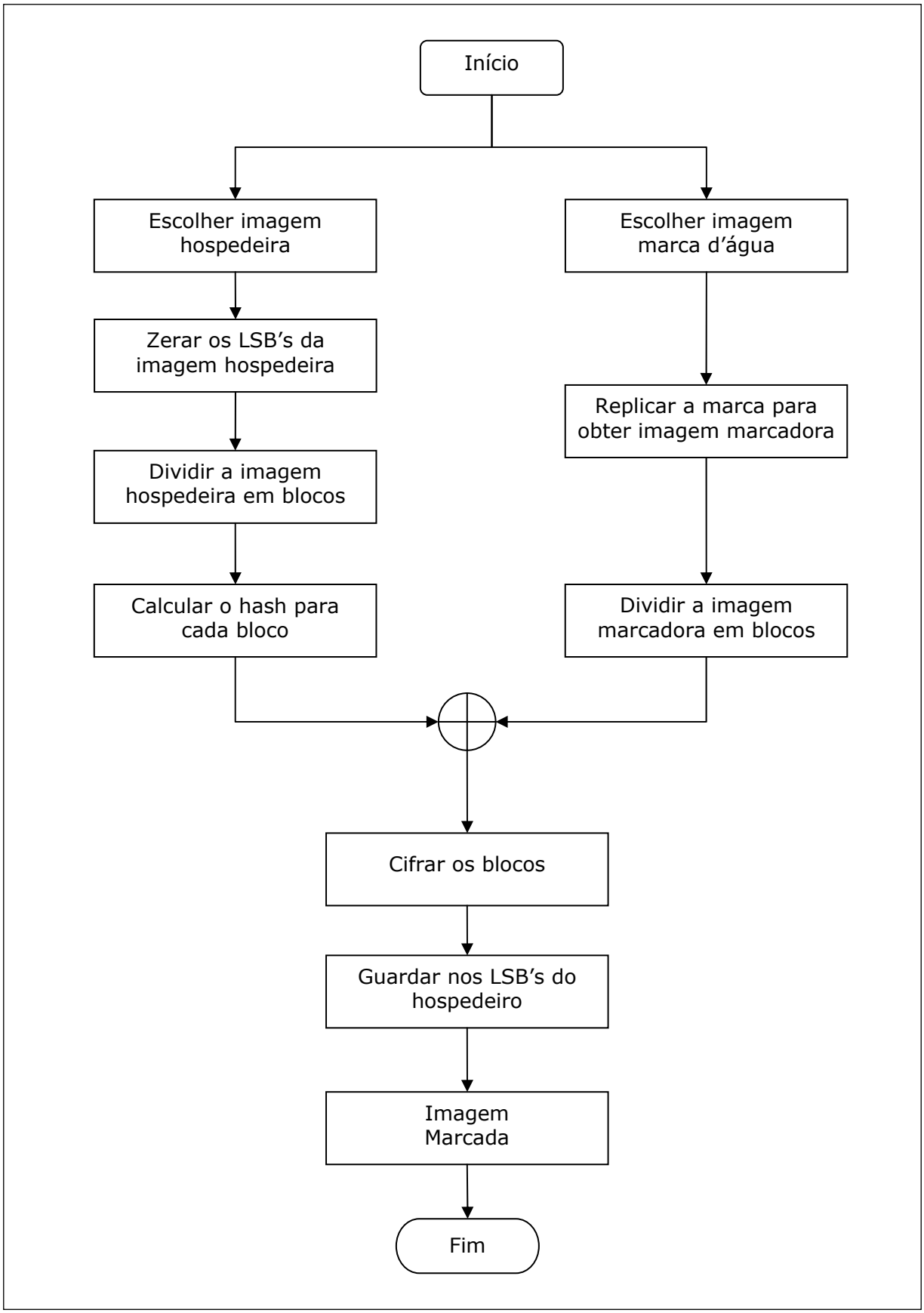


Figura 4.1 – Fluxograma da inserção da marca d'água digital

Fonte: Autor

4.3.2. Verificação da imagem marcada

O processo de verificação da imagem marcada, em cada plano RGB, basicamente é da seguinte forma:

- Passo 1: Escolha uma imagem I' TIFF RGB de 24 *bits* marcada pelo processo de inserção da marca d'água digital, descrito anteriormente em 4.3.1, com $\boldsymbol{N} \times M$ *pixels*. Particione I' em t blocos I'_t, da mesma forma do processo de inserção.
- Passo 2: Pegar o bloco t e decifrá-lo obtendo D_t;
- Passo 3: Zere os LSB's do hospedeiro, obtendo I^*_t;
- Passo 4: Utilize a mesma função de *hash* e calcule o resultado;
- Passo 5: Calcule o ou-exclusivo de H_t com D_t, obtendo o bloco de checagem C_t;
- Passo 6: Salve o bloco C_t em uma nova imagem I_v. A imagem será verificada, se em toda imagem I_v puder ser vista claramente a imagem B replicada. Caso contrário, a imagem marcada foi alterada e um ruído será mostrado no bloco t.

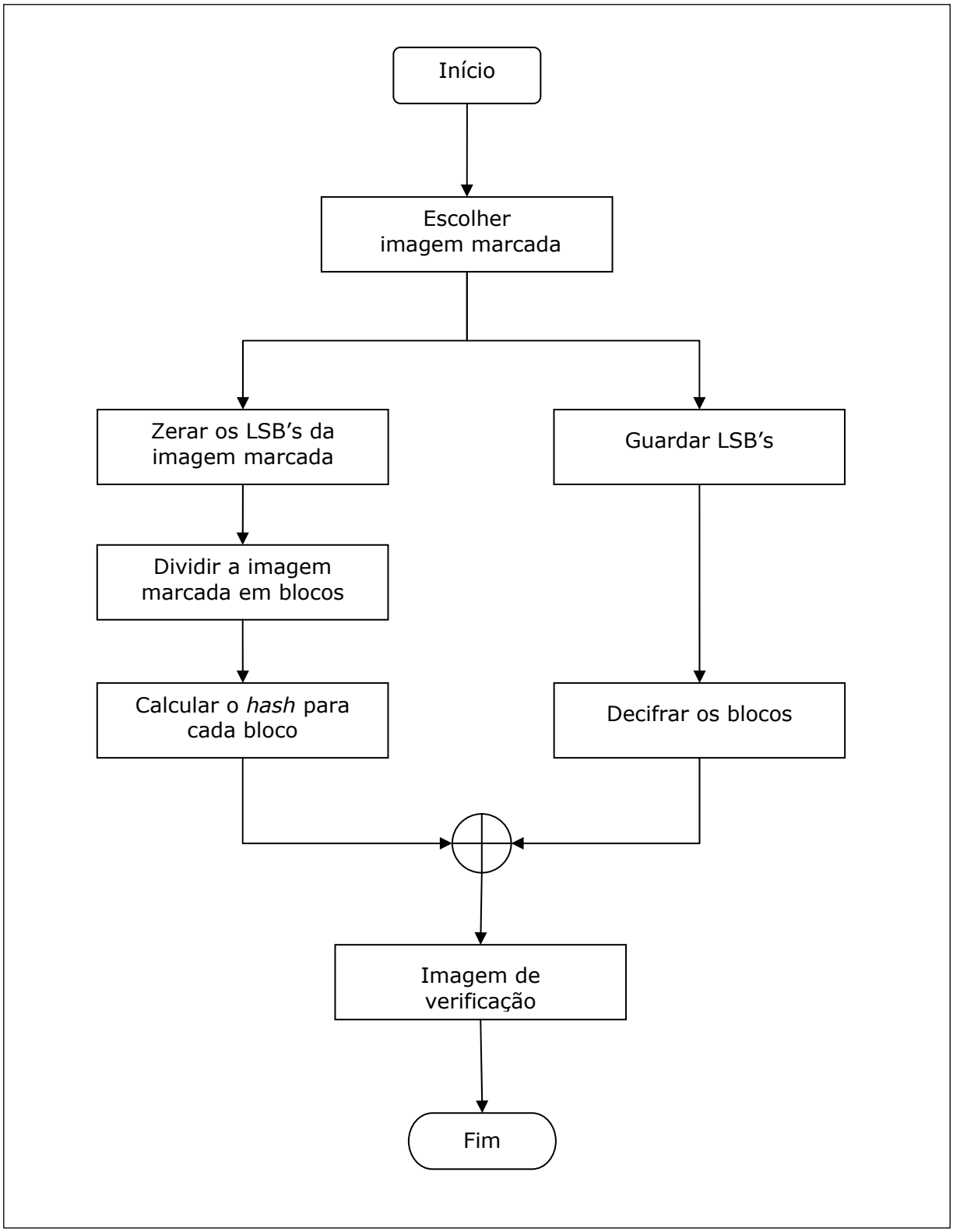


Figura 4.2 – Fluxograma de verificação da imagem marcada

Fonte: Autor

4.4. Interface gráfica

A interface gráfica do protótipo foi totalmente construída explorando os recursos *Handle Graphics*[10] do MATLAB®, a fim de criar um ambiente funcional de fácil operação, além de apresentar os resultados obtidos de uma forma bem clara.

A principal característica desse ambiente é que a execução das funções acontece de forma transparente para o usuário, isto é, como se não estivesse no MATLAB®, pois, uma vez chamada a função principal, surge uma tela, contendo botões, que acionam as funções.

4.4.1. Telas do protótipo

Chamando a função principal *mda.m* do protótipo, através da janela *Command*[11] do MATLAB®, surge uma tela (figura 4.3) com três opções. Cada uma dessas opções altera a tela principal, de forma a surgirem outras opções.

A figura 4.4 mostra a tela do processo de inclusão da marca d'água digital. Na figura 4.5, pode ser visto a tela do processo de verificação da imagem marcada. E a figura 4.6 apresenta a tela de simulação de ataque.

[10] Handle Graphics – coleção de rotinas gráficas de baixo nível que cumprem a tarefa de gerar gráficos no MATLAB®.
[11] Command – janela do MATLAB® na qual são enviados os comandos de execução do programa.

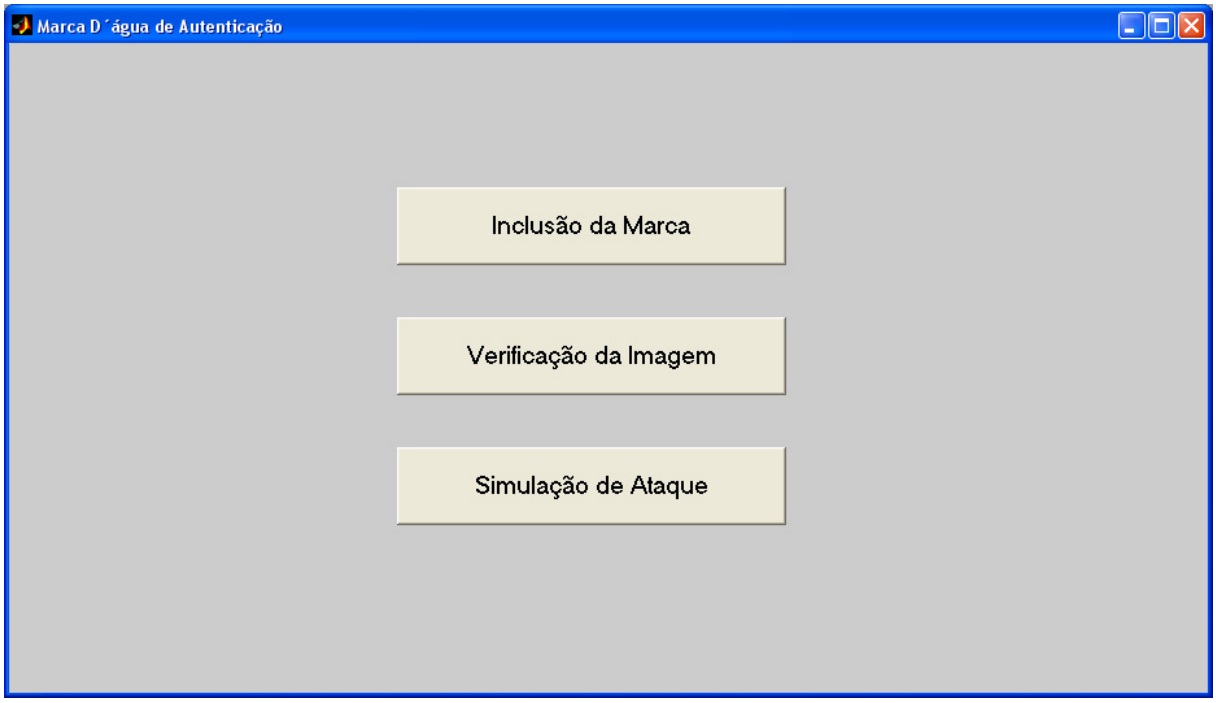


Figura 4.3 – Tela principal do protótipo

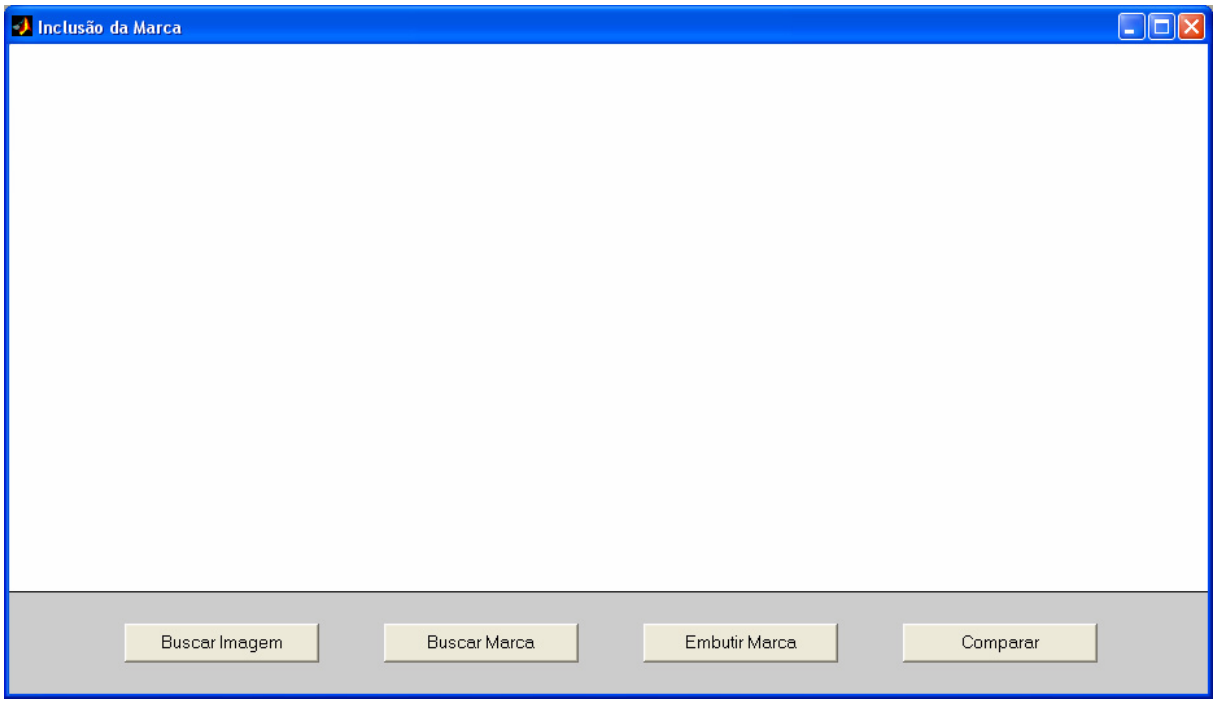


Figura 4.4 – Tela do processo de inclusão da marca d'água

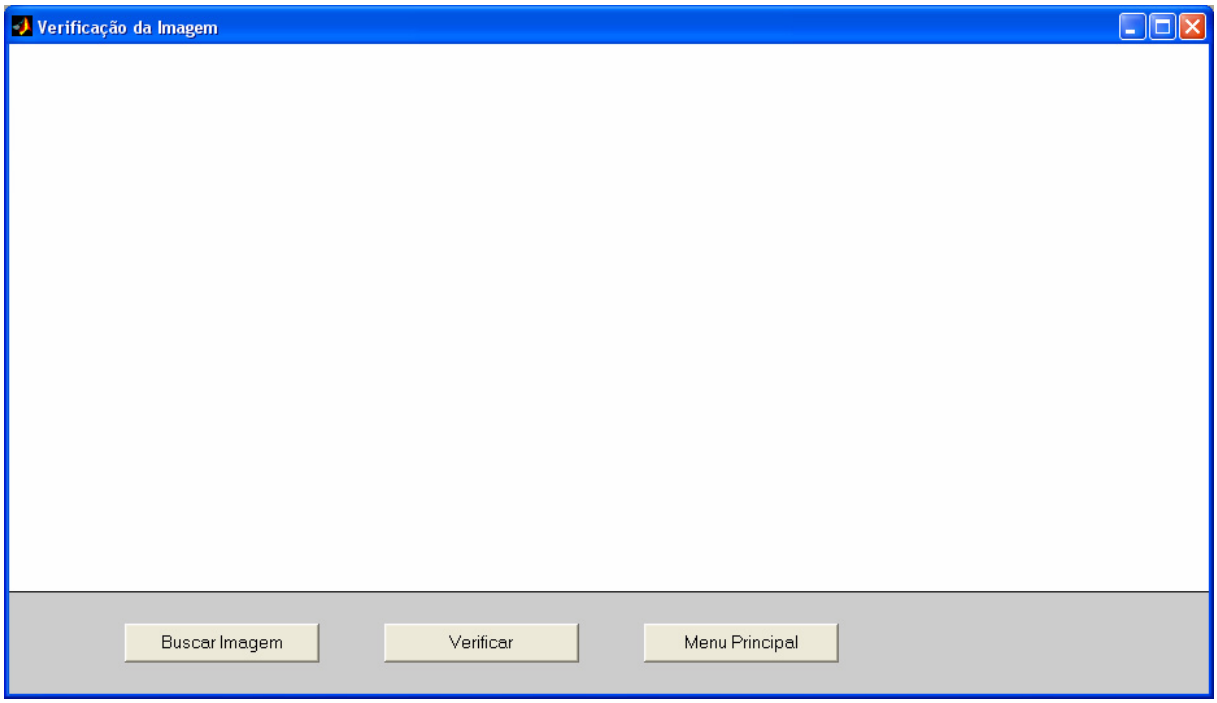


Figura 4.5 – Tela do processo de verificação da imagem

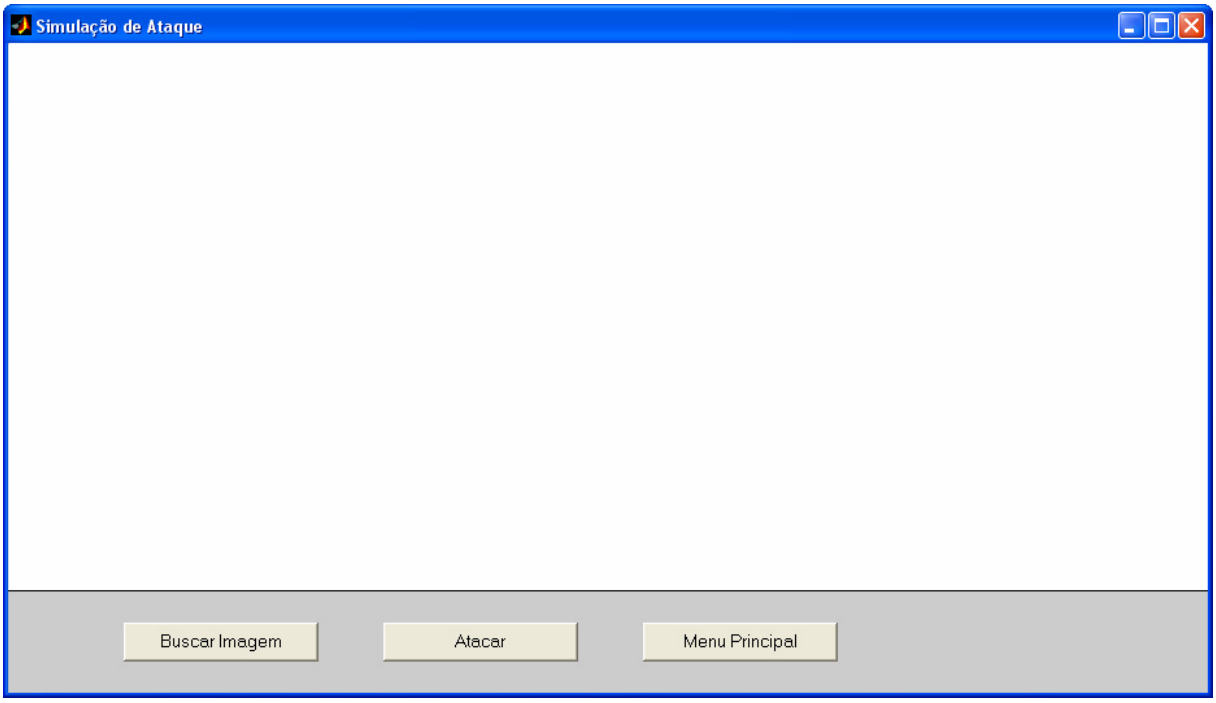


Figura 4.6 – Tela do processo de simulação de ataque

Durante a execução do programa, surgem outras telas temporárias sobre a tela principal ativa, como por exemplo, a tela do teclado para entrada da chave (figura 4.7) na qual aguarda o usuário entrar com o valor da cifra. Outra tela temporária, chamada processamento (figura 4.8), surge durante a execução dos processos de inclusão da marca d'água e verificação da imagem, indicando o progresso da execução.

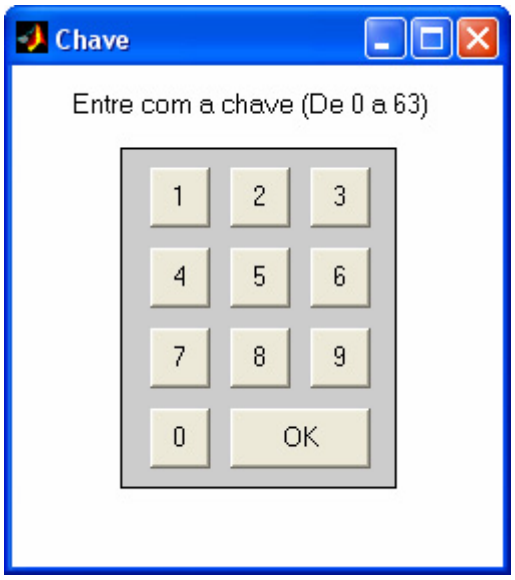


Figura 4.7 – Tela do teclado para entrada da chave

Figura 4.8 – Tela do progresso da execução

5. TESTES E SIMULAÇÕES

5.1. Imagens testadas

Todas as imagens utilizadas nos testes e simulações do protótipo são cortesias da USC – *University of Southern Califórnia*. O *Signal & Image Processing Institute* da USC possui um banco de dados de imagens e através da Internet disponibiliza essas imagens para realização de testes de processamento digital de imagem.

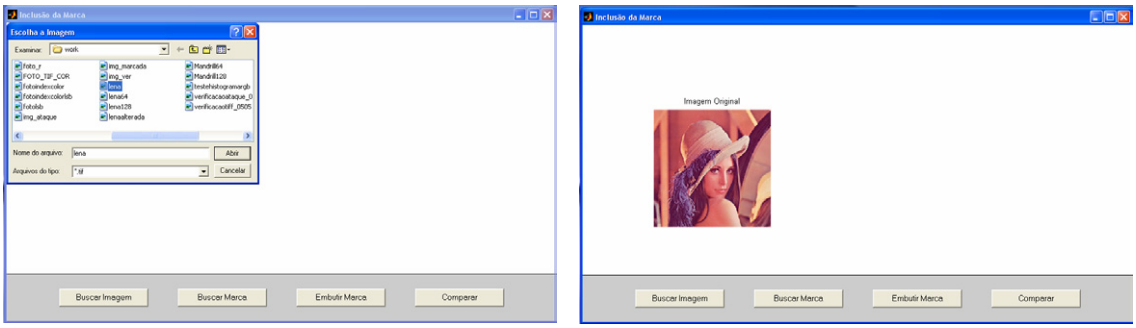
5.2. Inclusão da marca d'água digital

O processo de inclusão da marca d'água digital é dividido em quatro etapas:

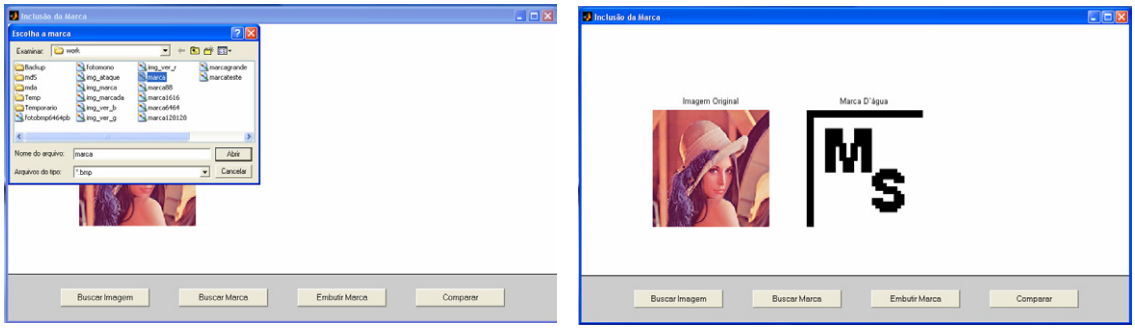
- Escolha da imagem hospedeira;
- Escolha da marca d'água digital;
- Entrada do valor da cifra;
- Inclusão da marca d'água digital.

A figura 5.1 ilustra as quatro etapas do processo de inclusão da marca d'água digital.

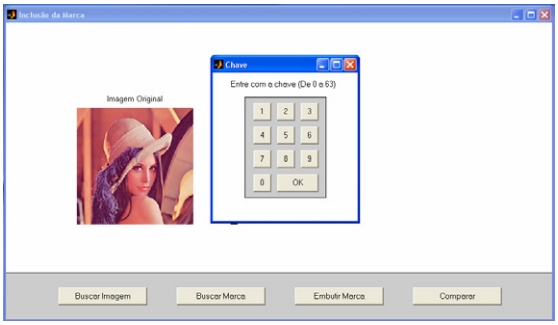
1ª Etapa – Escolha da imagem hospedeira



2ª Etapa – Escolha da marca d'água



3ª Etapa – Entrada do valor da cifra



4ª Etapa – Inclusão da marca d'água

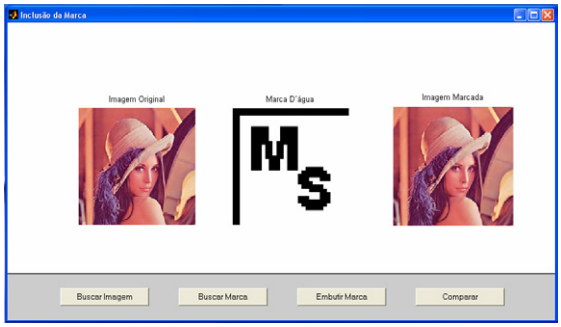


Figura 5.1 – Etapas do processo de inclusão

5.3. Comparação das imagens

Após a inclusão da marca d'água digital é possível através do botão "Comparar", apresentar na tela as duas imagens: original e marcada, lado a lado, conforme pode ser observado na figura 5.2.

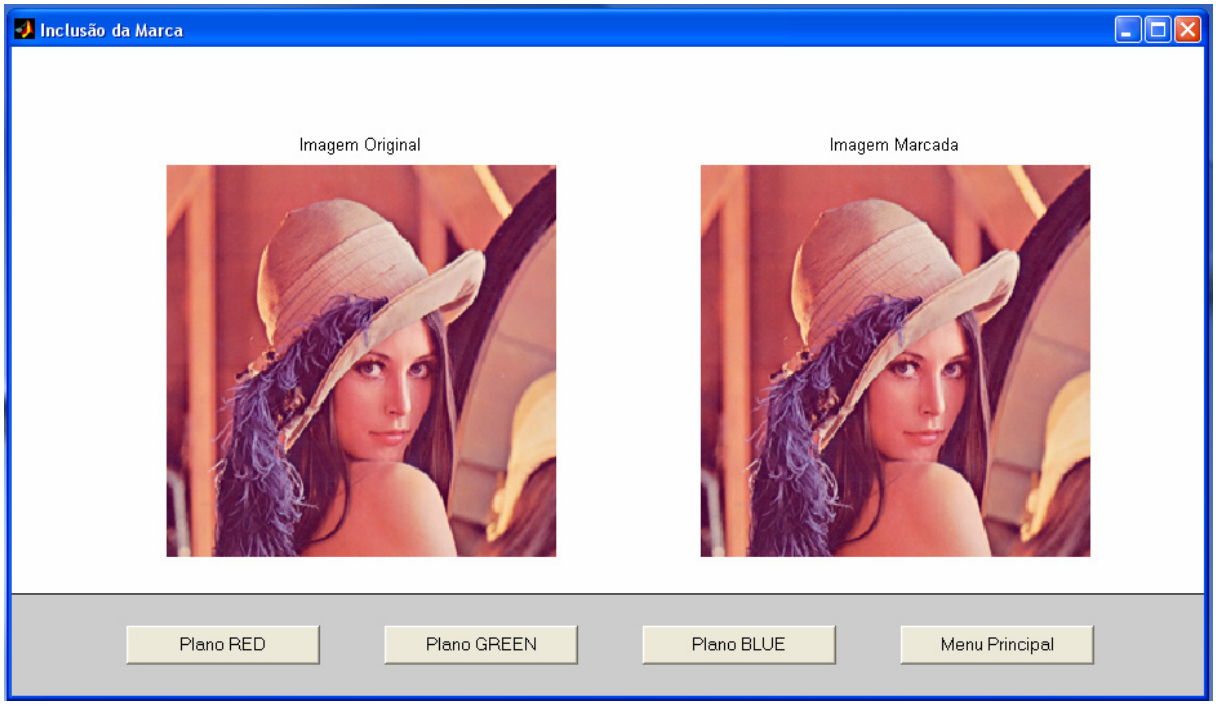


Figura 5.2 – Comparação das imagens

5.3.1. Histograma e estatística

O protótipo também apresenta os histogramas de cada plano RGB das duas imagens, juntamente com os dados estatísticos, a fim de possibilitar uma avaliação comparativa entre a imagem original e a imagem marcada.

As figuras 5.3, 5.4 e 5.5 apresentam a comparação dos três planos RGB, respectivamente.

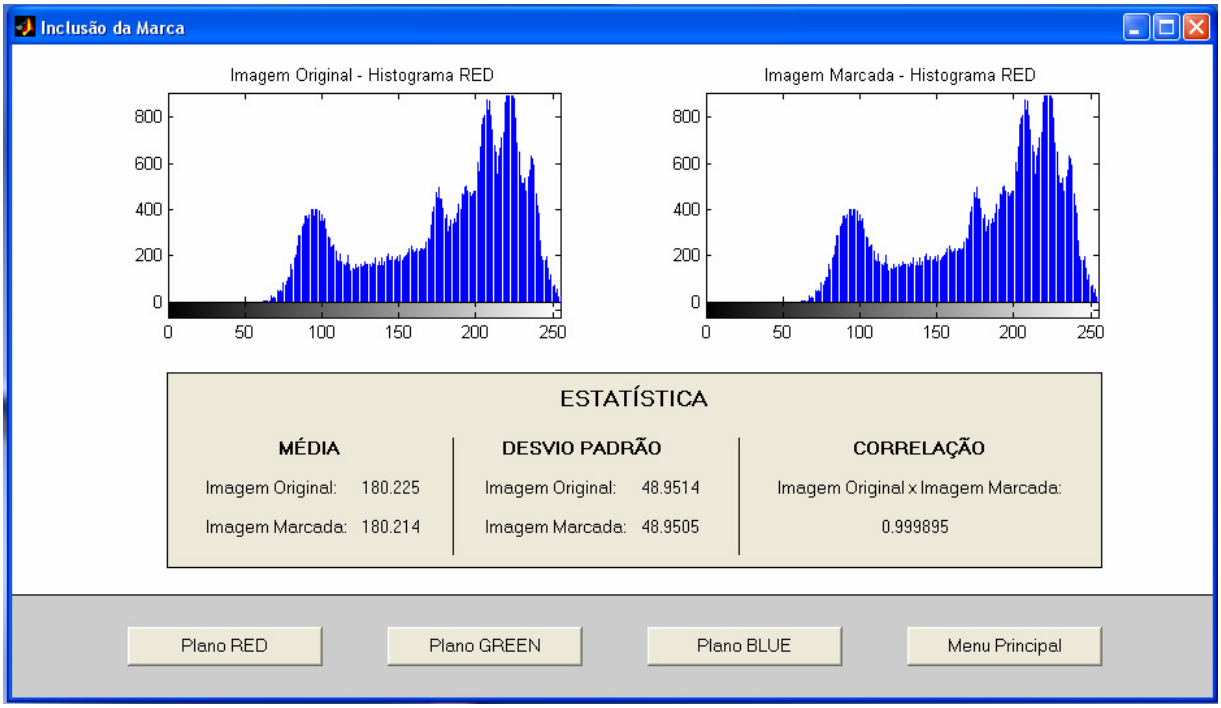


Figura 5.3 – Comparação do plano RED

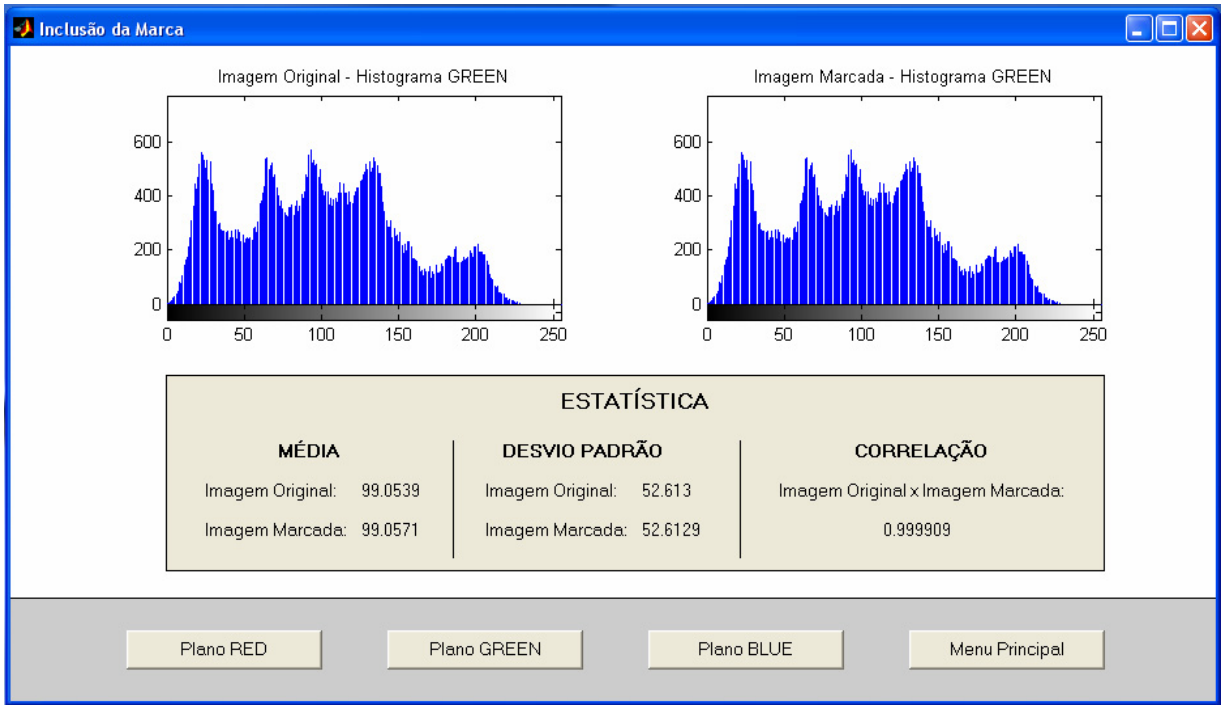


Figura 5.4 – Comparação do plano GREEN

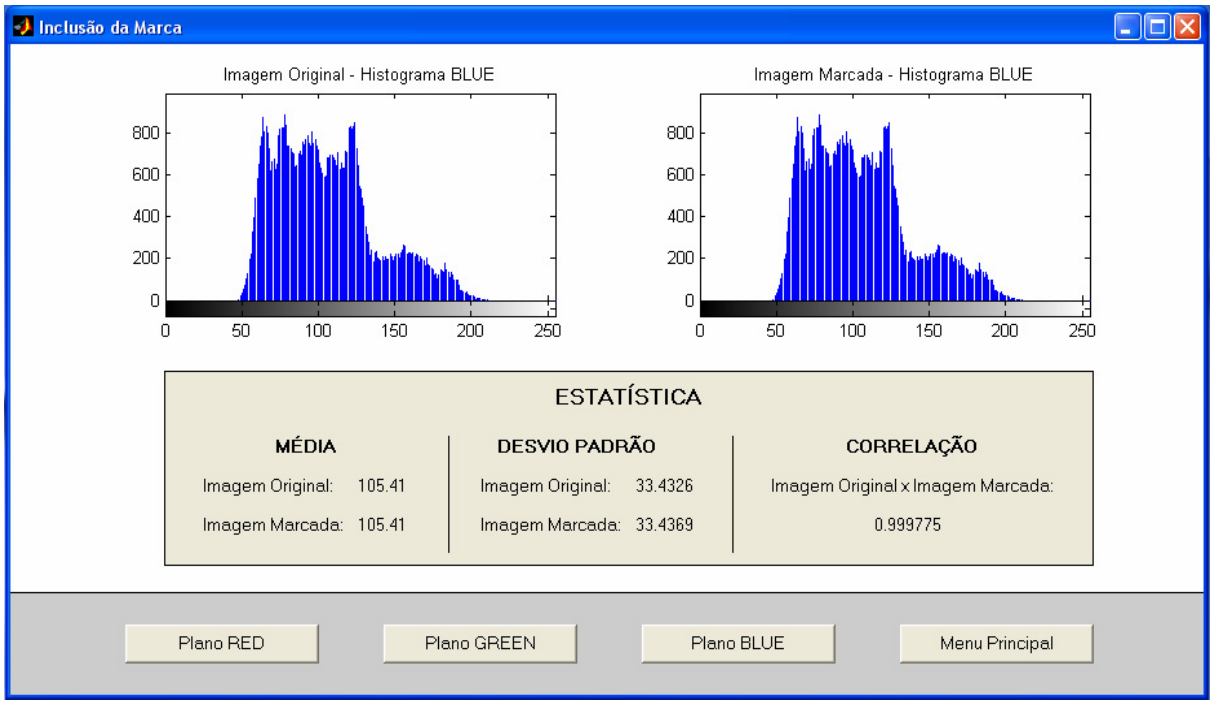


Figura 5.5 – Comparação do plano BLUE

Baseado nos resultados apresentados pode-se observar que nos três planos RGB, tanto os histogramas quanto os dados estatísticos (média e desvio padrão) das duas imagens comparadas, são bem semelhantes. Outro dado importante é o coeficiente de correlação, entre a imagem original e a imagem marcada, que apresentou valores em próximo de 1, assim constatando a semelhança entre as duas imagens.

Tabela 5.1 – Comparação estatística do plano RED

Imagem	Média	Desvio Padrão	Correlação
Original	180,225	48,9514	0,999895
Marcada	180,214	48,9505	

Fonte: Autor

Tabela 5.2 – Comparação estatística do plano GREEN

Imagem	Média	Desvio Padrão	Correlação
Original	99,0539	52,6130	0,999909
Marcada	99,0571	52,6129	

Fonte: Autor

Tabela 5.3 – Comparação estatística do plano BLUE

Imagem	Média	Desvio Padrão	Correlação
Original	105,41	33,4326	0,999775
Marcada	105,41	33,4369	

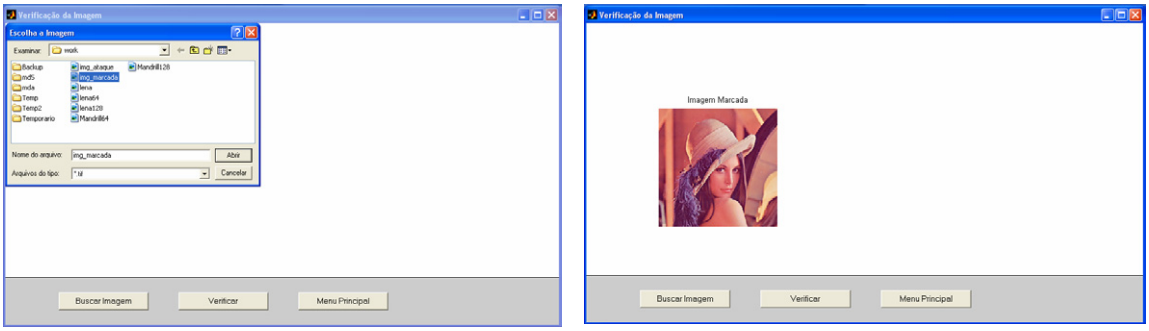
Fonte: Autor

5.4. Verificação da imagem marcada

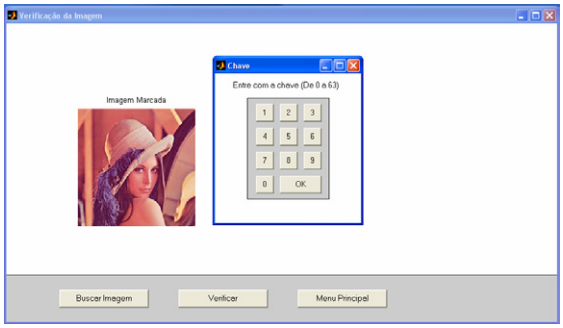
O processo de verificação da imagem marcada é dividido em três etapas (conforme figura 5.6):

- Buscar uma imagem que contém a marca d'água;
- Entrar com o valor da cifra correspondente ao valor utilizado no processo de inclusão da marca d'água;
- Extrair a imagem marcadora para verificar a autenticidade e integridade da imagem.

1ª Etapa – Buscar a imagem marcada



2ª Etapa – Entrar com o valor da cifra



3ª Etapa – Extrair a imagem marcadora

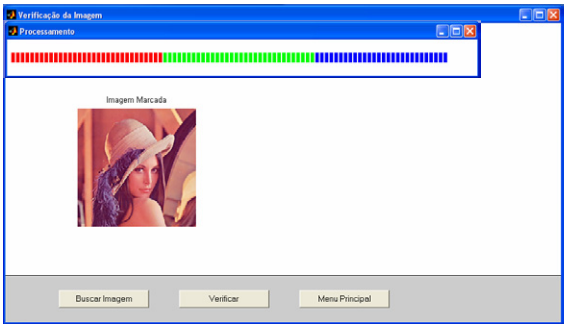


Figura 5.6 – Etapas do processo de verificação

Para simulação da verificação da imagem serão apresentadas três situações:

- Verificação com sucesso;
- Verificação com cifra inválida;
- Verificação de imagem alterada.

5.4.1. Verificação com sucesso

Uma verificação com sucesso ocorre quando a imagem marcada não sofreu nenhuma alteração e o valor utilizado da cifra no processo de verificação foi correspondente ao valor utilizado na inclusão da marca d'água.

A figura 5.6 apresenta o resultado da verificação com sucesso.

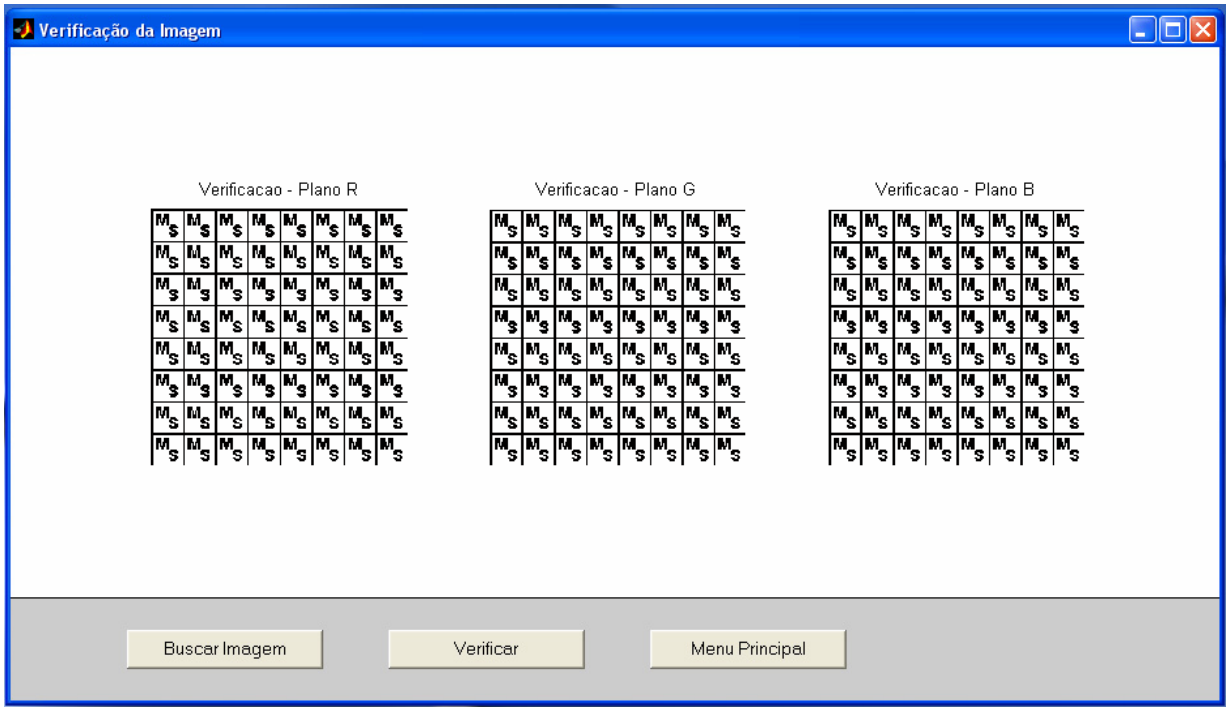


Figura 5.6 – Verificação com sucesso

5.4.2. Verificação com cifra inválida

Caso o valor da cifra seja inválido, isto é, não corresponda ao valor utilizado no processo de inclusão, não será possível visualizar a imagem extraída com a marca, pois, em todos os blocos da imagem, a verificação da assinatura digital será inválida, e dessa forma, a imagem extraída apresentará um ruído aleatório, conforme pode ser observado na figura 5.7.

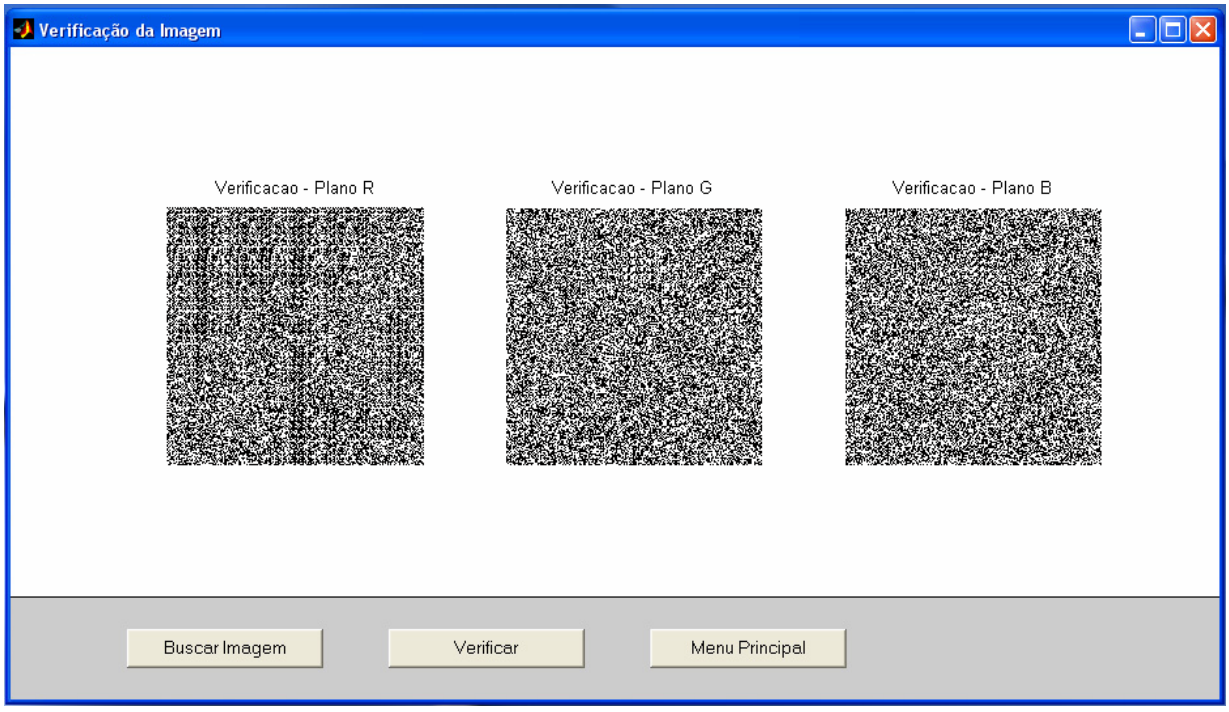


Figura 5.7 – Verificação com cifra inválida

5.4.3. Verificação de imagem alterada

Na verificação de uma imagem marcada que foi alterada, será gerado um ruído aleatório nos blocos que possuem *pixels* modificados e também nos blocos vizinhos. Essa situação está ilustrada na figura 5.8.

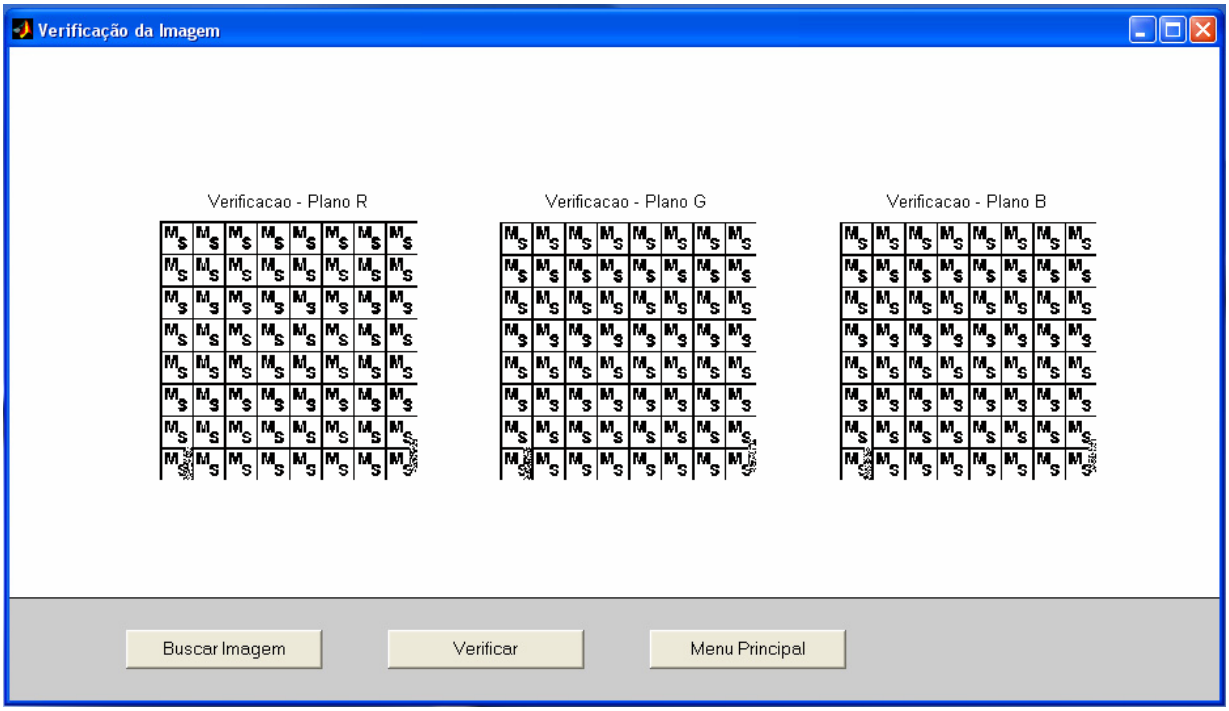


Figura 5.8 – Verificação de imagem alterada

5.5. Simulação de ataque

Essa função do protótipo realiza uma simulação de um ataque em uma imagem marcada, denominado ataque "recortar-e-colar", no qual um bloco é recortado e colado em outra parte da imagem.

A figura 5.9 ilustra um exemplo de simulação de ataque, no qual um quadrado de 32x32 *pixels* da imagem foi recortado e colado em outra posição.

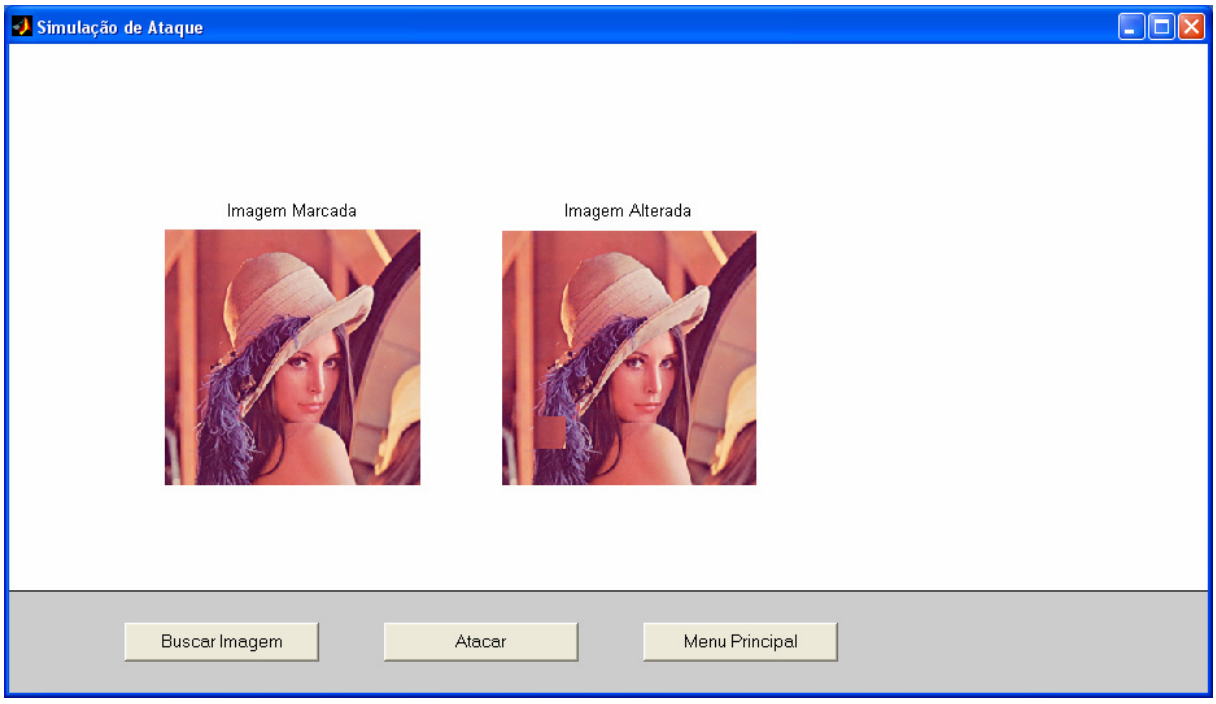


Figura 5.9 – Simulação de ataque

Se essa imagem alterada, passar pelo processo de verificação será gerado um ruído aleatório nos blocos correspondentes aos *pixels* alterados da imagem, conforme pode ser observado na figura 5.10.

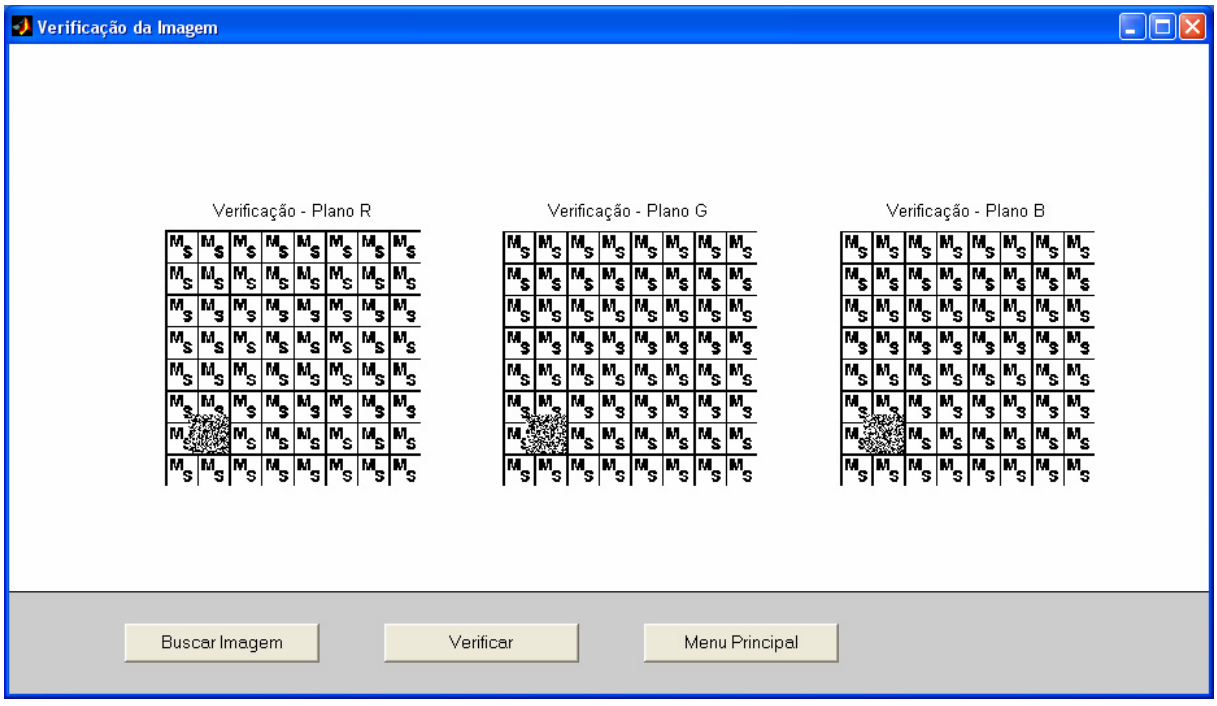


Figura 5.10 – Verificação de imagem alterada

6. CONCLUSÃO

No presente estudo realizado a respeito de marca d'água de autenticação, concluiu-se que a implementação, com o formato TIFF, foi operacionalmente viável, da mesma forma que no projeto anterior o formato BMP atingiu os objetivos propostos.

Além disso, a inserção da marca d'água de autenticação nos três planos RGB da imagem, também se mostrou possível, pois, através da análise dos dados estatísticos obtidos nos testes e simulações, pode-se verificar que não houve degradação da imagem.

Também verificou-se que o algoritmo desenvolvido foi eficiente na manutenção da integridade da imagem, detectando possíveis alterações nos três planos RGB.

Deste modo, pode-se destacar que o desenvolvimento do projeto foi bem sucedido, pois, os objetivos pré-determinados foram alcançados satisfatoriamente.

6.1. Trabalhos futuros

Nas pesquisas para o desenvolvimento deste trabalho, foi possível verificar que, a cada dia surgem novas técnicas e métodos para criação de uma marca d'água de autenticação eficiente e segura. Não se esgotando assim, as várias possibilidades de estudo com marca d'água de autenticação.

Desta forma, apresentam-se a seguir algumas sugestões para trabalhos futuros:

- Implementar o protótipo em outra linguagem de desenvolvimento (C, Builder, Java, etc);
- Estender a aplicação para o TIFF com compressão ou para o formato JPEG, através da utilização do método no domínio da freqüência.
- Utilizar outros recursos criptográficos, a fim de aumentar a segurança.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ADOBE.com. *TIFF Revision 6.0 Specification*. Disponível em: http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf. Último acesso em 20/05/2006.

BARRETO, Paulo S. L. M. *Criptografia Robusta e Marcas 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.* Curitiba: Universidade Católica do Paraná, 2000.

CASTLEMAN, Kenneth R. *Digital Image Processing*. New Jersey: Prentice-Hall, Inc. Eglewood Cliffs, 1979.

CHAPMAN, Stephen J. *Programação em MATLAB® para Engenheiros*. São Paulo: Pioneira Thomson Learning, 2003.

FALCÃO, Alexandre Xavier. *Fundamentos de Processamento de Imagem Digital.* Campinas: UNICAMP, 2003.

FILHO, Luiz Lourenço de Mello. *Percepção Visual Humana*. 1999. Disponível em: http://www.parconsult.com.br/uff.visao.htm. Último acesso em: 08/05/2006.

GIL, Antonio Carlos. *Métodos e Técnicas de Pesquisa Social*. 5. ed. São Paulo: Atlas, 1999.

GOMES, Jonas; VELHO, Luiz. *Computação Gráfica: Imagem*. Rio de Janeiro: IMPA/SBM, 1994.

HANSELMAN, Duane; LITTLEFIELD, Bruce. *MATLAB® 6. Curso Completo*. São Paulo: Prentice Hall, 2003.

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

LUCCHESI, Cláudio Leonardo. *Introdução à Criptografia Computacional*. Campinas: Papirus/Unicamp, 1986.

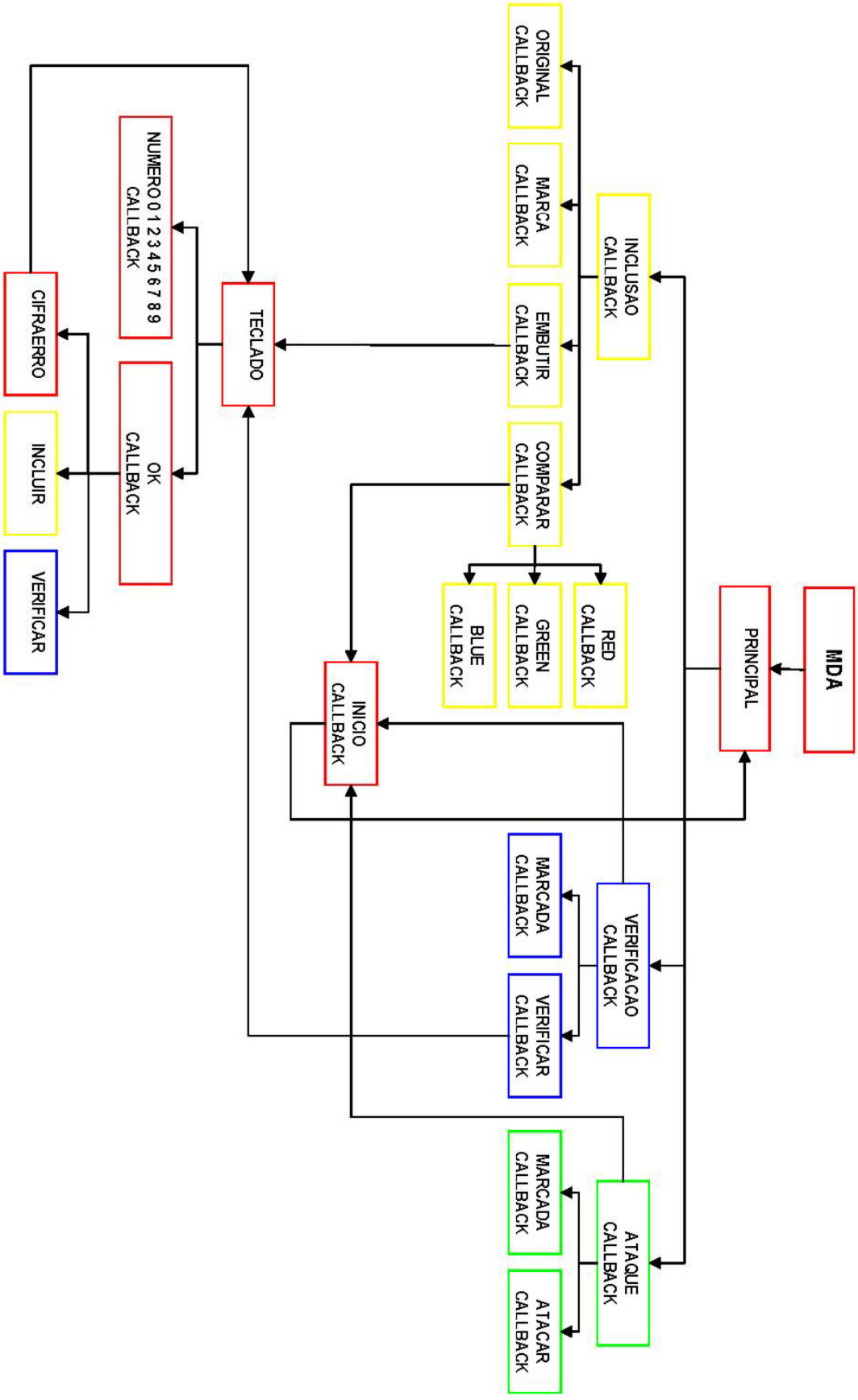
MASCARENHAS, D. A; VELASCO, Flávio R. D. *Processamento Digital de Imagens*. Termas de Rio Hondo, Argentina: IV Escola Brasileiro-Argentina de Informática – Universidad Católica de Santiago Del Estero, 1989.

SANTAELLA, Lucia. *Comunicação e Pesquisa*: *Projetos para Mestrado e Doutorado*. São Paulo: Hacker Editores, 2001.

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. Último acesso em: 15/05/2006.

USC - University of Southern California. *The USC-SIPI Image Database*. Disponível em: http://sipi.usc.edu/database. Último acesso em: 30/05/2006.

APÊNDICE A – Fluxograma das funções implementadas



APÊNDICE B – Funções de inicialização

mda.m

```
% ========================================================================
% Centro Universitario de Brasilia - UniCEUB
% Faculdade de Ciencias Exatas e Tecnologia - FAET
% Curso de Engenharia da Computaçao
% Disciplina: Projeto Final - 1° Semestre de 2006
% Prof. Orientador: Aderlon Queiroz
% Aluno: Marco Antonio da Silva Soares
% ========================================================================
% Aplicativo MDA - Marca D'agua de Autenticaçao
% ========================================================================
% Funçoes Implementadas:
% inicio.m               Desenha a tela principal
% principal.m            Cria o menu principal
% inicio_callback.m      Retorna ao menu principal
% inclusao_callback.m    Chama as funçoes do processo de inclusao
% original_callback.m    Escolher o hospedeiro (imagem TIFF)
% marca_callback.m       Escolher a marca (imagem binaria)
% embutir_callback.m     Chama a funçao do teclado
% teclado.m              Desenha o teclado
% numero0_callback.m     Atribui o valor 0 a variavel da cifra
% numero1_callback.m     Atribui o valor 1 a variavel da cifra
% numero2_callback.m     Atribui o valor 2 a variavel da cifra
% numero3_callback.m     Atribui o valor 3 a variavel da cifra
% numero4_callback.m     Atribui o valor 4 a variavel da cifra
% numero5_callback.m     Atribui o valor 5 a variavel da cifra
% numero6_callback.m     Atribui o valor 6 a variavel da cifra
% numero7_callback.m     Atribui o valor 7 a variavel da cifra
% numero8_callback.m     Atribui o valor 8 a variavel da cifra
% numero9_callback.m     Atribui o valor 9 a variavel da cifra
% ok_callback.m          Verifica se o valor da cifra e valido
%                        e atribui a variavel chave
% cifraerro.m            Atribui o valor 0 a variavel chave e
%                        retorna a funçao teclado
% incluir.m              Realiza o processo de inclusao da marca d'agua
%                        e salva a imagem marcada como img_marcada.tif
% comparar_callback.m    Mostra na tela as duas imagens original e marcada
%                        e chama as funçoes de comparaçao de planos RGB
% red_callback.m         Mostra na tela os histogramas e os dados
%                        estatisticos do plano RED das duas imagens
%                        (original e marcada)
% green_callback.m       Mostra na tela os histogramas e os dados
%                        estatisticos do plano GREEN das duas imagens
%                        (original e marcada)
% blue_callback.m        Mostra na tela os histogramas e os dados
%                        estatisticos do plano BLUE das duas imagens
%                        (original e marcada)
% verificacao_callback.m Chama as funçoes do processo de verificaçao
% marcada_callback.m     Escolher a imagem marcada no processo de inclusao
% verificar_callback.m   Chama a funçao do teclado
% verificar.m            Realiza o processo de verificacao da imagem
%                        e salva as imagens de verificaçao de cada plano
%                        RGB como img_ver_r.bmp, img_ver_g.bmp e
%                        img_ver_b.bmp e apresenta na tela
% ataque_callback.m      Chama as funçoes do processo de simulaçao
%                        de ataque
```

```
% atacar_callback.m      Realiza a simulaçao do ataque e salva a imagem
%                        como img_ataque.tif e apresenta na tela
% ========================================================================
% Chama a funçao inicio
%
clear all;
clc;
inicio;
```

inicio.m

```
function inicio
    global cor
    cor=[0.8 0.8 0.8];
    figure('Position',[50 134 924 500],'Color',cor,'menubar','none',
    'NumberTitle','off','Name','Marca D´água de Autenticação');
    principal;
```

inicio_callback.m

```
function inicio_callback
    global hosp8
    global marca2
    principal;
```

principal.m

```
function principal
    global cor
    clf;
    set(figure(1),'Color',cor,'Name','Marca D´água de Autenticação');
    uicontrol('Style','pushbutton','String','Inclusão da Marca',
    'Enable','on','Fontsize',14,'Position',[300 330 300 60],
    'callback','inclusao_callback');
    uicontrol('Style','pushbutton','String','Verificação da Imagem',
    'Enable','on','Fontsize',14,'Position',[300 230 300 60],
    'callback','verificacao_callback');
    uicontrol('Style','pushbutton','String','Simulação de Ataque',
    'Enable','on','Fontsize',14,'Position',[300 130 300 60],
    'callback','ataque_callback');
```

APÊNDICE C – Funções do processo de inclusão da marca d'água

inclusão_callback.m

```
function inclusao_callback
    global cor
    global hosp8
    global marca2
    global chave
    clf;
    set(figure(1),'Color','white','Name','Inclusão da Marca');
    uicontrol('Style','frame','BackgroundColor',cor,
    'Position',[0 0 926 80]);
    uicontrol('Style','pushbutton','String','Buscar Imagem',
    'Enable','on','Fontsize',10,'Position',[90 25 150 30],
    'callback','original_callback');
    uicontrol('Style','pushbutton','String','Buscar Marca',
    'Enable','on','Fontsize',10,'Position',[290 25 150 30],
    'callback','marca_callback');
    uicontrol('Style','pushbutton','String','Embutir Marca',
    'Enable','on','Fontsize',10,'Position',[490 25 150 30],
    'callback','embutir_callback');
    uicontrol('Style','pushbutton','String','Comparar','Enable','on',
    'Fontsize',10,'Position',[690 25 150 30],
    'callback','comparar_callback');
```

original_callback.m

```
function original_callback
    global hosp8
    [a_hosp, h_dir]=uigetfile('*.tif', 'Escolha a Imagem');
    hosp8=imread([h_dir, a_hosp]);
    subplot(1,3,1), imshow(hosp8), title('Imagem Original');
```

marca_callback.m

```
function marca_callback
    global marca2
    [a_marca, m_dir]=uigetfile('*.bmp', 'Escolha a marca');
    marca2=imread([m_dir, a_marca]);
    subplot(1,3,2), imshow(marca2), title('Marca D´água');
```

embutir_callback.m

```
function embutir_callback
    global hosp8
    global marca2
    global cr
    global chave
    global md
    global p
    cr = '0';
    md = 1;
    teclado;
```

incluir.m

```
function incluir
    global hosp8
    global marca2
    global chave
    global figmarcada
    global p;
    global pg;

% ========================================================================
% Parametros da Criptografia
% ========================================================================
%
chave=floor(chave);
%
% ========================================================================
% Parametros do Hospedeiro
% ========================================================================
%
hosp=double(hosp8);
%
% Pega o tamanho do hospedeiro
%
[M,N,O]=size(hosp);
%
% Seta tamanho dos blocos de marcaçao
%
I=8;
J=8;
%
% ========================================================================
% Monta a marca d'agua do tamanho do hospedeiro
% ========================================================================
%
% Le o bitmap que servira como marca
%
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);
```

```
imwrite(figmarca,'img_marca.bmp','bmp');
%
% ========================================================================
% Zera os LSB's do hospedeiro
% ========================================================================
%
% Variavel hosp sem lsb : hosp_
%
for z=1:O,
    for u=1:M,
        for v=1:N,
            if mod(hosp(u,v,z),2)==1,
                hosp_(u,v,z)=hosp(u,v,z)-1;
            else
                hosp_(u,v,z)=hosp(u,v,z);
            end
        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
% ========================================================================
%
p=0;
for z=1:O,
for x=0:I:M-1,
    f=(M+8)*3;
        if z==1 & x==0
        figure('Position',[53 545 f
60],'Color','white','menubar','none','NumberTitle','off','Name','Processame
nto');
        end
        if z==1
        p=8+p
        figure(2);
        uicontrol('Style','frame','Position',[p 23 6
16],'BackgroundColor','red','ForegroundColor','red');
        drawnow;
        elseif z==2
        p=8+p
        figure(2);
        uicontrol('Style','frame','Position',[p 23 6
16],'BackgroundColor','green','ForegroundColor','green');
        drawnow;
        elseif z==3
        if x==M-8
            close(2)
        else
            p=8+p
        figure(2);
        uicontrol('Style','frame','Position',[p 23 6
16],'BackgroundColor','blue','ForegroundColor','blue');
        drawnow;
        end
        end
    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),z);
        %
        % Primeiro bloco (usado na ultima dependencia)
        %
        if (x==0 & y==0)
            bloco1=blocoh;
        end
        %
        % 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),z);
        else
            blocoh2=hosp_((1+x):(I+x),(1+y+8):(J+y+8),z);
        end
        blocohdep((1:8),(1:8),z)=blocoh;
        blocohdep((1:8),(9:16),z)=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
        % (Formado ASCII)
        %
        dos('c:\matlab6p5p1\work\md5\md5 -oc:\matlab6p5p1\work\blocoh.hsh
c:\matlab6p5p1\work\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);
%
% 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 informaçao
        %
        Dmxh = double(mxh);
        Dblocoh = double(blocoh);
        blocoht = Dmxh + Dblocoh;
        for v=1:I,
            for u=1:J,
                figmarcada(v+x,u+y,z)=blocoht(v,u);
            end
        end
    end
end
end
% ========================================================================
% Salva a imagem marcada
% ========================================================================
%
figmarcada=uint8(figmarcada);
imwrite(figmarcada,'img_marcada.tif','tif');
subplot(1,3,3), imshow(figmarcada), title('Imagem Marcada');
```

APÊNDICE D – Funções do teclado para entrada da chave

teclado.m

```
function teclado
    global cor
    global hosp8
    global marca2
    global cr
    global chave
    global md
    global p
    figure('Position',[400 300 245 250],'Color','white','menubar','none',
    'NumberTitle','off','Name','Chave');
    uicontrol('Style','frame','BackgroundColor',cor,
    'Position',[55 40 138 170]);
    uicontrol('Style','text','BackgroundColor','white','String',
    'Entre com a chave (De 0 a 63)','Fontsize',10,
    'Position',[0 210 240 30]);
    uicontrol('Style','pushbutton','String','1','Enable','on',
    'Fontsize',10,'Position',[70 170 30 30],'callback','numero1_callback');
    uicontrol('Style','pushbutton','String','2','Enable','on',
    'Fontsize',10,'Position',[110 170 30 0],'callback','numero2_callback');
    uicontrol('Style','pushbutton','String','3','Enable','on',
    'Fontsize',10,'Position',[150 170 30 0],'callback','numero3_callback');
    uicontrol('Style','pushbutton','String','4','Enable','on',
    'Fontsize',10,'Position',[70 130 30 30],'callback','numero4_callback');
    uicontrol('Style','pushbutton','String','5','Enable','on',
    'Fontsize',10,'Position',[110 130 30 0],'callback','numero5_callback');
    uicontrol('Style','pushbutton','String','6','Enable','on',
    'Fontsize',10,'Position',[150 130 30 0],'callback','numero6_callback');
    uicontrol('Style','pushbutton','String','7','Enable','on',
    'Fontsize',10,'Position',[70 90 30 30],'callback','numero7_callback');
    uicontrol('Style','pushbutton','String','8','Enable','on',
    'Fontsize',10,'Position',[110 90 30 30],'callback','numero8_callback');
    uicontrol('Style','pushbutton','String','9','Enable','on',
    'Fontsize',10,'Position',[150 90 30 30],'callback','numero9_callback');
    uicontrol('Style','pushbutton','String','0','Enable','on',
    'Fontsize',10,'Position',[70 50 30 30],'callback','numero0_callback');
    uicontrol('Style','pushbutton','String','OK','Enable','on',
    'Fontsize',10,'Position',[110 50 70 30],'callback','ok_callback');
```

numero0_callback.m

```
function numero0_callback
    global cr
    r='0';
    cr=strcat(cr,r);
```

numero1_callback.m

```
function numero1_callback
    global cr
    r='1';
    cr=strcat(cr,r);
```

numero2_callback.m

```
function numero2_callback
    global cr
    r='2';
    cr=strcat(cr,r);
```

numero3_callback.m

```
function numero3_callback
    global cr
    r='3';
    cr=strcat(cr,r);
```

numero4_callback.m

```
function numero4_callback
    global cr
    r='4';
    cr=strcat(cr,r);
```

numero5_callback.m

```
function numero5_callback
    global cr
    r='5';
    cr=strcat(cr,r);
```

numero6_callback.m

```
function numero6_callback
    global cr
    r='6';
    cr=strcat(cr,r);
```

numero7_callback.m

```
function numero7_callback
    global cr
    r='7';
    cr=strcat(cr,r);
```

numero8_callback.m

```
function numero8_callback
    global cr
    r='8';
    cr=strcat(cr,r);
```

numero9_callback.m

```
function numero9_callback
    global cr
    r='9';
    cr=strcat(cr,r);
```

ok_callback.m

```
function ok_callback
    global cr
    global chave
    global hosp8
    global marca2
    global figmarcada
    global md
    global p
    global pg
    chave=str2double(cr);
    if chave > 63
       close(2);
       cifraerro;
       figure(2);
       uicontrol('Style','frame','BackgroundColor','red',
        'Position',[0 0 247 35]);
        uicontrol('Style','text','FontWeight','bold','Fontsize',10,'String',
        'Chave Inválida! (Maior que 63)','BackgroundColor','red',
        'Position',[0 0 247 26]);
    else
        close(2);
        drawnow;
         if md==1
             incluir;
         else
             verificar;
        end
    end
```

cifraerro.m

```
function cifraerro
    global cor
    global hosp8
    global marca2
    global cr
    global chave
    global md
    global p
    cr = '0';
    teclado;
```

APÊNDICE E – Funções do processo de comparação

comparar_callback.m

```
function comparar_callback
    global cor
    global hosp8
    global figmarcada
    uicontrol('Style','frame','BackgroundColor',cor,
    'Position',[0 0 926 80]);
    uicontrol('Style','pushbutton','String','Plano RED','Enable','on',
    'Fontsize',10,'Position',[90 25 150 30],'callback','red_callback');
    uicontrol('Style','pushbutton','String','Plano GREEN','Enable','on',
    'Fontsize',10,'Position',[290 25 150 30],'callback','green_callback');
    uicontrol('Style','pushbutton','String','Plano BLUE','Enable','on',
    'Fontsize',10,'Position',[490 25 150 30],'callback','blue_callback');
    uicontrol('Style','pushbutton','String','Menu Principal','Enable','on',
    'Fontsize',10,'Position',[690 25 150 30],'callback','inicio_callback');
    subplot(1,2,1), imshow(hosp8), title('Imagem Original');
    subplot(1,2,2), imshow(figmarcada), title('Imagem Marcada');
```

red_callback.m

```
function red_callback
    global cor
    global hosp8
    global figmarcada
    subplot(2,2,1), imhist(hosp8(:,:,1)),
    title('Imagem Original - Histograma RED');
    subplot(2,2,2), imhist(hosp8(:,:,1)),
    title('Imagem Marcada - Histograma RED');
    md_ori=mean2(hosp8(:,:,1));
    dv_ori=std2(hosp8(:,:,1));
    md_mrc=mean2(figmarcada(:,:,1));
    dv_mrc=std2(figmarcada(:,:,1));
    crl=corr2(hosp8(:,:,1),figmarcada(:,:,1));
    uicontrol('Style','frame','Position',[120 100 720 150]);
    uicontrol('Style','text','String','ESTATÍSTICA','FontWeight','bold',
    'Fontsize',12,'Position',[380 220 200 20]);
    uicontrol('Style','text','String','MÉDIA','FontWeight','bold',
    'Fontsize',10,'Position',[130 180 200 20]);
    uicontrol('Style','text','String','Imagem Original:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[150 150 160 20]);
    uicontrol('Style','text','String','Imagem Marcada:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[150 120 120 20]);
    uicontrol('Style','text','String',md_ori,
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[270 150 70 20]);
    uicontrol('Style','text','String',md_mrc,
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[270 120 70 20]);
    uicontrol('Style','text','String','DESVIO PADRÃO',
    'FontWeight','bold','Fontsize',10,'Position',[360 180 160 20]);
    uicontrol('Style','text','String','Imagem Original:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[365 150 160 20]);
```

```
    uicontrol('Style','text','String','Imagem Marcada:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[365 120 200 20]);
    uicontrol('Style','text','String',dv_ori,
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[485 150 70 20]);
    uicontrol('Style','text','String',dv_mrc,
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[485 120 70 20]);
    uicontrol('Style','text','String','CORRELAÇÃO',
    'FontWeight','bold','Fontsize',10,
    'Position',[620 180 160 20]);
    uicontrol('Style','text','String','Imagem Original x Imagem Marcada:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[590 150 230 20]);
    uicontrol('Style','text','String',crl,'HorizontalAlignment','left',
    'Fontsize',12,'Position',[670 120 70 20]);
    uicontrol('Style','frame','Position',[340 110 1 90]);
    uicontrol('Style','frame','Position',[560 110 1 90]);
```

green_callback.m

```
function green_callback
    global hosp8
    global figmarcada
    subplot(2,2,1), imhist(hosp8(:,:,2)),
    title('Imagem Original - Histograma GREEN');
    subplot(2,2,2), imhist(hosp8(:,:,2)),
    title('Imagem Marcada - Histograma GREEN');
    md_ori=mean2(hosp8(:,:,2));
    dv_ori=std2(hosp8(:,:,2));
    md_mrc=mean2(figmarcada(:,:,2));
    dv_mrc=std2(figmarcada(:,:,2));
    crl=corr2(hosp8(:,:,2),figmarcada(:,:,2));
    uicontrol('Style','frame','Position',[120 100 720 150]);
    uicontrol('Style','text','String','ESTATÍSTICA','FontWeight','bold',
    'Fontsize',12,'Position',[380 220 200 20]);
    uicontrol('Style','text','String','MÉDIA','FontWeight','bold',
    'Fontsize',10,'Position',[130 180 200 20]);
    uicontrol('Style','text','String','Imagem Original:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[150 150 160 20]);
    uicontrol('Style','text','String','Imagem Marcada:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[150 120 120 20]);
    uicontrol('Style','text','String',md_ori,
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[270 150 70 20]);
    uicontrol('Style','text','String',md_mrc,
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[270 120 70 20]);
    uicontrol('Style','text','String','DESVIO PADRÃO','FontWeight','bold',
    'Fontsize',10,'Position',[360 180 160 20]);
    uicontrol('Style','text','String','Imagem Original:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[365 150 160 20]);
    uicontrol('Style','text','String','Imagem Marcada:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[365 120 200 20]);
```

```
    uicontrol('Style','text','String',dv_ori,
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[485 150 70 20]);
    uicontrol('Style','text','String',dv_mrc,
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[485 120 70 20]);
    uicontrol('Style','text','String','CORRELAÇÃO',
    'FontWeight','bold','Fontsize',10,'Position',[620 180 160 20]);
    uicontrol('Style','text','String','Imagem Original x Imagem Marcada:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[590 150 230 20]);
    uicontrol('Style','text','String',crl,
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[670 120 70 20]);
    uicontrol('Style','frame','Position',[340 110 1 90]);
    uicontrol('Style','frame','Position',[560 110 1 90]);
```

blue_callback.m

```
function blue_callback
    global hosp8
    global figmarcada
    subplot(2,2,1), imhist(hosp8(:,:,3)),
    title('Imagem Original - Histograma BLUE');
    subplot(2,2,2), imhist(hosp8(:,:,3)),
    title('Imagem Marcada - Histograma BLUE');
    md_ori=mean2(hosp8(:,:,3));
    dv_ori=std2(hosp8(:,:,3));
    md_mrc=mean2(figmarcada(:,:,3));
    dv_mrc=std2(figmarcada(:,:,3));
    crl=corr2(hosp8(:,:,3),figmarcada(:,:,3));
    uicontrol('Style','frame','Position',[120 100 720 150]);
    uicontrol('Style','text','String','ESTATÍSTICA','FontWeight','bold',
    'Fontsize',12,'Position',[380 220 200 20]);
    uicontrol('Style','text','String','MÉDIA','FontWeight','bold',
    'Fontsize',10,'Position',[130 180 200 20]);
    uicontrol('Style','text','String','Imagem Original:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[150 150 160 20]);
    uicontrol('Style','text','String','Imagem Marcada:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[150 120 120 20]);
    uicontrol('Style','text','String',md_ori,
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[270 150 70 20]);
    uicontrol('Style','text','String',md_mrc,
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[270 120 70 20]);
    uicontrol('Style','text','String','DESVIO PADRÃO',
    'FontWeight','bold','Fontsize',10,'Position',[360 180 160 20]);
    uicontrol('Style','text','String','Imagem Original:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[365 150 160 20]);
    uicontrol('Style','text','String','Imagem Marcada:',
    'HorizontalAlignment','left','Fontsize',12,
    'Position',[365 120 200 20]);
    uicontrol('Style','text','String',dv_ori,
    'HorizontalAlignment','left','Fontsize',12,'Position',[485 150 70 20]);
    uicontrol('Style','text','String',dv_mrc,
    'HorizontalAlignment','left','Fontsize',12,
```

```
'Position',[485 120 70 20]);
uicontrol('Style','text','String','CORRELAÇÃO',
'FontWeight','bold','Fontsize',10,'Position',[620 180 160 20]);
uicontrol('Style','text','String','Imagem Original x Imagem Marcada:',
'HorizontalAlignment','left','Fontsize',12,
'Position',[590 150 230 20]);
uicontrol('Style','text','String',crl,
'HorizontalAlignment','left','Fontsize',12,
'Position',[670 120 70 20]);
uicontrol('Style','frame','Position',[340 110 1 90]);
uicontrol('Style','frame','Position',[560 110 1 90]);
```

APÊNDICE F – Funções do processo de verificação

verificacao_callback.m

```
function verificacao_callback
    global cor
    global hosp8
    global marca2
    clf;
    set(figure(1),'Color','white','Name','Verificação da Imagem');
    uicontrol('Style','frame','BackgroundColor',cor,
    'Position',[0 0 926 80]);
    uicontrol('Style','pushbutton','String','Buscar Imagem','Enable','on',
    'Fontsize',10,'Position',[90 25 150 30],
    'callback','marcada_callback');
    uicontrol('Style','pushbutton','String','Verificar','Enable','on',
    'Fontsize',10,'Position',[290 25 150 30],
    'callback','verificar_callback');
    uicontrol('Style','pushbutton','String','Menu Principal','Enable','on',
    'Fontsize',10,'Position',[490 25 150 30],
    'callback','inicio_callback');
```

marcada_callback.m

```
function marcada_callback
    global hosp8
    [a_hosp, h_dir]=uigetfile('*.tif', 'Escolha a Imagem');
    hosp8=imread([h_dir, a_hosp]);
    subplot(1,3,1), imshow(hosp8), title('Imagem Marcada');
```

verificar_callback.m

```
function verificar_callback
    global hosp8
    global marca2
    global cr
    global chave
    global md
    cr = '0';
    md = 2;
    teclado;
```

verificar.m

```
function verificar
    global hosp8
    global marca2
    global chave
    global figmarcada
    global p
```

```
% ========================================================================
% Parametros da Criptografia
% ========================================================================
%
chave=floor(chave);
%
% ========================================================================
% Parametros do Hospedeiro
% ========================================================================
%
hosp=double(hosp8);
%
% Pega o tamanho do hospedeiro
%
[M,N,O]=size(hosp);
%
% Seta tamanho dos blocos de marcacao
%
I=8;
J=8;
%
% ========================================================================
% Zera os LSB's da imagem marcada // Guarda LSB em outra matriz
% ========================================================================
%
% Variavel hosp sem lsb : hosp_
% Variavel de lsb's : lsb
%
for z=1:O,
    for u=1:M,
        for v=1:N,
            if mod(hosp(u,v,z),2)==1,
                hosp_(u,v,z)=hosp(u,v,z)-1;
                lsb(u,v,z) = 1;
            else
                hosp_(u,v,z)=hosp(u,v,z);
                lsb(u,v,z) = 0;
            end
        end
    end
end
%
% ========================================================================
% Varre as imagens hosp_
% Particiona a imagem em blocos de IxJ
% Calcula hash do bloco para verificaçao
% ========================================================================
%
p=0;
for z=1:O,
for x=0:I:M-1,
        f=(M+8)*3;
        if z==1 & x==0
        figure('Position',[53 545 f
60],'Color','white','menubar','none','NumberTitle','off','Name','Processame
nto');
        end
        if z==1
        p=8+p
        figure(2);
```

```
        uicontrol('Style','frame','Position',[p 23 6
16],'BackgroundColor','red','ForegroundColor','red');
        drawnow;
        elseif z==2
        p=8+p
        figure(2);
        uicontrol('Style','frame','Position',[p 23 6
16],'BackgroundColor','green','ForegroundColor','green');
        drawnow;
        elseif z==3
        if x==M-8
            close(2)
        else
            p=8+p
        figure(2);
        uicontrol('Style','frame','Position',[p 23 6
16],'BackgroundColor','blue','ForegroundColor','blue');
        drawnow;
        end
        end
    for y=0:J:N-1,
        %
        % Cria bloco IxJ de hosp_
        %
        blocoh=hosp_((1+x):(I+x),(1+y):(J+y),z);
        %
        % Cria bloco IxJ de lsb
        %
        blocolsb=lsb((1+x):(I+x),(1+y):(J+y),z);
        %
        % Primeiro bloco (usado na ultima dependencia)
        %
        if (x==0 & y==0)
            bloco1=blocoh;
        end
        %
        % 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) , z);
        else
            blocoh2=hosp_((1+x) : (I+x) , (1+y+8) : (J+y+8) , z);
        end
        blocohdep((1:8) , (1:8) , z)=blocoh;
        blocohdep((1:8) , (9:16) , z)=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
        % (Formado ASCII)
        %
        dos('c:\matlab6p5p1\work\md5\md5 -oc:\matlab6p5p1\work\blocoh.hsh
c:\matlab6p5p1\work\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);
        %
        % 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 bloco h (mhash)
        %
        mxh = xor(blocolsb, mhash);

        % Insere no bloco de hosp_ a informaçao
        %
        Dmxh = double(mxh);
        for v=1:I,
            for u=1:J,
                if z==1
                    figver_r(v+x,u+y)=Dmxh(v,u);
                elseif z==2
                    figver_g(v+x,u+y)=Dmxh(v,u);
                else
                    figver_b(v+x,u+y)=Dmxh(v,u);
                end
            end
        end
    end
end
end
```

```
% ========================================================================
% Salva a imagem marcada
% ========================================================================
%
figver_r=logical(figver_r);
imwrite(figver_r,'img_ver_r.bmp','bmp');
figver_g=logical(figver_g);
imwrite(figver_g,'img_ver_g.bmp','bmp');
figver_b=logical(figver_b);
imwrite(figver_b,'img_ver_b.bmp','bmp');
%
ver_r=imread('img_ver_r.bmp');
ver_g=imread('img_ver_g.bmp');
ver_b=imread('img_ver_b.bmp');

subplot(1,3,1), imshow(ver_r), title('Verificação - Plano R');
subplot(1,3,2), imshow(ver_g), title('Verificação - Plano G');
subplot(1,3,3), imshow(ver_b), title('Verificação - Plano B');
```

APÊNDICE G – Funções do processo de simulação de ataque

ataque_callback.m

```
function ataque_callback
    global cor
    global hosp8
    clf;
    set(figure(1),'Color','white','Name','Simulação de Ataque');
    uicontrol('Style','frame','BackgroundColor',cor,
    'Position',[0 0 926 80]);
    uicontrol('Style','pushbutton','String','Buscar Imagem','Enable','on',
    'Fontsize',10,'Position',[90 25 150 30],'callback','marcada_callback');
    uicontrol('Style','pushbutton','String','Atacar','Enable','on',
    'Fontsize',10,'Position',[290 25 150 30],
    'callback','atacar_callback');
    uicontrol('Style','pushbutton','String','Menu Principal','Enable','on',
    'Fontsize',10,'Position',[490 25 150 30],
    'callback','inicio_callback');
```

atacar_callback.m

```
function atacar_callback
    global hosp8

% ========================================================================
% Pega um bloco da imagem (Neste caso, 16x16 - os primeiros pixels num
% quadrado de 32)
% ========================================================================
%
imgsaida = hosp8;
%
for z=1:3
    bloco = hosp8((1:16),(1:16),z);
    %
    % Insere este mesmo bloco em outra posiçao
    % Aqui, no quadrado limitado pelos pontos (1,17) e (16,32)
    %
    imgsaida((1:16),(17:32),z) = bloco;
end
    imgsaida = uint8(imgsaida);
%
% Salva imagem alterada em img_ataque.bmp
%

imwrite(imgsaida,'img_ataque.tif','tif');
subplot(1,3,2), imShow(imgsaida), title('Imagem Alterada');
```