SERGIO VICENTE DENSER PAMBOUKIAN

MARCAS D'ÁGUA DE AUTENTICAÇÃO PARA IMAGENS BINÁRIAS: MARCAS REVERSÍVEIS E MARCAS PARA O PADRÃO JBIG2

São Paulo
2007

SERGIO VICENTE DENSER PAMBOUKIAN

MARCAS D'ÁGUA DE AUTENTICAÇÃO PARA IMAGENS BINÁRIAS: MARCAS REVERSÍVEIS E MARCAS PARA O PADRÃO JBIG2

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia.

São Paulo
2007

SERGIO VICENTE DENSER PAMBOUKIAN

MARCAS D'ÁGUA DE AUTENTICAÇÃO PARA IMAGENS BINÁRIAS: MARCAS REVERSÍVEIS E MARCAS PARA O PADRÃO JBIG2

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia.

Área de Concentração:
Sistemas Eletrônicos

Orientador: Prof. Livre-Docente
Hae Yong Kim

São Paulo
2007

DEDICATÓRIA

À minha amada esposa Mari, por todo amor, incentivo e companheirismo.

AGRADECIMENTOS

Ao Prof. Dr. Hae Yong Kim, meu orientador e amigo, por ter me recebido tão bem e por todo o incentivo, estímulo e apoio recebidos.

Aos meus pais, Íris e Vicente, por toda a educação, orientação e amor durante toda a vida.

À minha querida esposa Mari, pelo amor, ajuda, incentivo, paciência e compreensão, não apenas durante o desenvolvimento deste trabalho, mas por toda a vida.

Ao meu irmão Celso, por todo apoio e incentivo.

Ao Rubens, Sueli, Rodrigo e Alexandre por toda a força, carinho e compreensão.

Aos professores Ricardo L. de Queiroz e Paulo S. L. M. Barreto, com os quais tive oportunidade de realizar publicações relacionadas ao tema desta tese.

Aos professores João José Neto, Maria Alice G. V. Ferreira e Hae Yong Kim que tornaram a fase de obtenção de créditos para o Doutorado muito prazerosa e produtiva.

Ao mestre e amigo Orlando Monezi Jr. que me iniciou na carreira acadêmica e que sempre será um exemplo de dedicação.

Aos amigos Edson A. R. Barros, Lincoln C. Zamboni, Melanie L. Grinkraut, Angela H. Tchemra, Daniel B. Barrios, Daniel A. Alves, Osvaldo R. T. Hu, Vilar R. de Figueiredo e tantos outros que me ajudaram com suas idéias e sugestões sobre a redação e formatação desta tese, principalmente na fase final do trabalho.

Ao amigo Elie Chadarevian pela ajuda e incentivo no início deste caminho.

Aos colegas da Universidade Presbiteriana Mackenzie que me incentivaram a buscar esta titulação.

A toda a minha família por todo amor, incentivo e carinho.

A todos que colaboraram de forma direta ou indireta para que este trabalho fosse possível.

RESUMO

Esteganografia é uma técnica utilizada para ocultar uma informação secreta dentro de outro tipo de informação sem perda de qualidade da informação hospedeira e com o objetivo de extrair a informação posteriormente. Esteganografia reversível permite a exata restauração (sem perda) do sinal hospedeiro original após a extração da informação oculta. Várias técnicas reversíveis têm sido desenvolvidas, mas nenhuma delas parece ser apropriada para imagens binárias. Uma técnica de marca d'água faz uso de técnicas esteganográficas para inserir informação em uma imagem hospedeira, com o intuito de fazer uma asserção sobre a imagem no futuro. Uma marca d'água de autenticação (AWT) insere uma informação oculta na imagem com a intenção de detectar qualquer alteração acidental ou maliciosa na imagem. Uma AWT normalmente usa criptografia de chave secreta ou chave pública para computar a assinatura de autenticação da imagem, inserindo-a na própria imagem. JBIG2 é um padrão internacional para compressão de imagens binárias (com ou sem perda). Ele decompõe a imagem em várias regiões (texto, meio-tom e genérica) e codifica cada região usando o método mais apropriado. A criação de AWTs seguras para imagens binárias comprimidas é um importante problema prático. Porém, parece que nenhuma AWT para JBIG2 já foi proposta. Este trabalho propõe algumas técnicas esteganográficas para arquivos JBIG2. Então, estas técnicas são usadas para criar AWTs para imagens codificadas como JBIG2. As imagens marcadas são visualmente agradáveis, sem ruídos do tipo sal-e-pimenta. Este trabalho também propõe uma técnica esteganográfica reversível para imagens binárias. A técnica proposta seleciona um conjunto de pixels de baixa visibilidade e utiliza o algoritmo de Golomb para comprimir as previsões desses pixels. Então, a informação comprimida e a informação a ser oculta são inseridas na imagem. Imagens marcadas com a técnica proposta possuem excelente qualidade visual, pois apenas pixels de baixa visibilidade são modificados. Então, a técnica proposta é utilizada para autenticar imagens binárias e documentos de maneira reversível.

Palavras-chave: Processamento de imagens. Esteganografia. Marcas d'água de autenticação. Marcas d'água reversíveis. Imagens binárias. JBIG2.

ABSTRACT

Data-hiding is a technique to hide secret information inside another group of information data, without loss of quality of the host information, and the means to extract the secret information afterwards. Reversible data-hiding enable the exact restoration (lossless) of the original host signal after extracting the embedded information. Several reversible data hiding techniques have been developed, but none of them seems to be appropriate for binary images. A watermarking technique makes use of data-hiding to insert some information into the host image, in order to make a posterior assertion about the image. An authentication watermarking technique (AWT) inserts hidden data into an image in order to detect any accidental or malicious alteration to the image. AWT normally makes use of secret- or public-key cryptographic ciphers to compute the authentication signature of the image, and inserts it into the image itself. JBIG2 is an international standard for compressing bi-level images in both lossy and lossless modes. JBIG2 decomposes the image into several regions (text, halftone and generic) and encodes each one using the most appropriate method. The creation of secure AWTs for compressed binary images is an important practical problem. However, it seems that no AWT for JBIG2 has ever been proposed. This work proposes some data-hiding techniques for JBIG2 files. Then, these techniques are used to design AWTs for JBIG2-encoded images. The resulting watermarked images are visually pleasant, without visible salt and pepper noise. This work also proposes a reversible data hiding technique for binary images. The proposed technique selects a set of low-visibility pixels and uses Golomb codes to compress the predictions of these pixels. Then, this compressed data and the net payload data are embedded into the image. Images watermarked by the proposed technique have excellent visual quality, because only low-visibility pixels are flipped. Then, the proposed data hiding is used to reversibly authenticate binary images and documents.

Keywords: Image processing. Data-hiding. Authentication watermark. Reversible watermark. Binary images. JBIG2.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE DIAGRAMAS

LISTA DE ABREVIATURAS E SIGLAS

Alice	Nome fictício da pessoa que assina um documento digital usando a sua chave privada.
AS	*Authentication Signature* (assinatura de autenticação).
AWCJH	*Authentication Watermarking by template ranking with symmetrical Central pixels for JBIG2 Halftone region* (marca d'água de autenticação pelo ranqueamento de modelos com pixels centrais simétricos para a região de meio-tom do JBIG2).
AWCJT	*Authentication Watermarking by template ranking with symmetrical Central pixels for JBIG2 Text region* (marca d'água de autenticação pelo ranqueamento de modelos com pixels centrais simétricos para a região de texto do JBIG2).
AWST	*Authentication Watermarking by Self Toggling* (marca d'água de autenticação por auto-inversão).
AWT	*Authentication Watermarking Technique* (técnica de marca d'água de autenticação).
AWTC	*Authentication Watermarking by Template ranking with symmetrical Central pixels* (marca d'água de autenticação pelo ranqueamento de modelos com pixels centrais simétricos).
AWTCJ	*Authentication Watermarking by Template ranking with symmetrical Central pixels for JBIG2* (marca d'água de autenticação pelo ranqueamento de modelos com pixels centrais simétricos para JBIG2).
AWTR	*Authentication Watermarking by Template Ranking* (marca d'água de autenticação pelo ranqueamento de modelos).
AWTRAL	*Authentication Watermarking by Template Ranking with Alteration Locating* (marca d'água de autenticação pelo ranqueamento de modelos com localização de alteração).
AWTRJ	*Authentication Watermarking by Template Ranking for JBIG2* (marca d'água de autenticação pelo ranqueamento de modelos para JBIG2).

BLS Esquema de assinatura de Boneh, Lynn e Shacham.

Bob Nome fictício da pessoa que verifica a assinatura de um documento digital usando a chave pública.

CALIC *Context-Based, Adaptive, Lossless Image Coder* (codificador adaptativo de imagens sem perda baseado em contexto).

CCITT *Comité Consultatif International Téléphonique et Télégraphique*. Organização que regulariza os padrões internacionais de comunicação.

Contone *Continuous tone* (tonalidade contínua). Imagens coloridas e em níveis de cinza.

DBP *Data Bearing Pixels* (pixels carregadores de informação).

DBS *Data Bearing Symbols* (símbolos carregadores de informação).

DH *Data Hiding* (ocultamento de informações, esteganografia).

DHCCJ *Data-Hiding by Coordinate Changing for JBIG2* (esteganografia por mudança de coordenada para JBIG2).

DHCJT *Data Hiding by template ranking with symmetrical Central pixels for JBIG2 Text region* (esteganografia pelo ranqueamento de modelos com pixels centrais simétricos para a região de texto do JBIG2).

DHPT *Data Hiding by Pair-Toggling* (embutimento de dados por inversão aos pares).

DHSPT *Data Hiding by Smart Pair Toggling* (embutimento de dados por inversão inteligente aos pares).

DHST *Data hiding by self toggling* (embutimento de dados por auto-inversão).

DHTC *Data Hiding by Template ranking with symmetrical Central pixels* (esteganografia pelo ranqueamento de modelos com pixels centrais simétricos).

DHTR *Data Hiding by Template Ranking* (esteganografia pelo ranqueamento de modelos).

DHTRJ *Data Hiding by Template Ranking for JBIG2* (esteganografia pelo ranqueamento de modelos para JBIG2).

DJGPP *DJ's GNU Programming Platform* (plataforma de programação GNU desenvolvida por D.J. Delorie).

DPI *Dots Per Inch* (pontos por polegada).

DS *Digital signature* (assinatura digital).

DSA *Digital Signature Algorithm* (algoritmo de assinatura digital).

FAX *Facsimile*.

FELICS *Fast Efficient & Lossless Image Compression System* (sistema de compressão de imagens sem perda rápido e eficiente).

G3 CCITT *Group* 3. Protocolo para transmissão de fax.

G4 CCITT *Group* 4. Protocolo para transmissão de fax.

GNU *GNU is Not UNIX* (GNU não é UNIX). Sistema operacional totalmente livre compatível com UNIX.

HBC *Hash Block Chaining* (encadeamento de blocos de *hash*).

IEC *International Electrotechnical Commission* (comissão eletrotécnica internacional).

IMG Biblioteca para Processamento de Imagens e Visão Computacional de autoria do Prof. Dr. Hae Yong Kim do Departamento de Engenharia de Sistemas Eletrônicos da Escola Politécnica da Universidade de São Paulo.

ISO *International Organization for Standardization* (organização internacional para padronização).

JBIG *Joint Bi-level Image experts Group* (grupo de especialistas em imagens de dois níveis). Formato de codificação de imagens.

JBIG1 Formato de codificação de imagens de dois níveis criado pelo JBIG. Também conhecido apenas como formato JBIG.

JBIG2 Formato de codificação de imagens de dois níveis criado pelo JBIG.

JPEG *Joint Photographic Experts Group* (grupo de especialistas em fotografia). Formato de codificação de imagens.

JPEG2000 Formato de codificação de imagens criado pelo JPEG.

JPEG-LS	Formato de codificação de imagens sem perda criado pelo JPEG.
LOCO-I	*Low Complexity lossless Compression for Images* (compressão de imagens sem perda de baixa complexidade).
LSB	*Least Significant Bit* (bit menos significativo).
LUT	*Look-Up-Table* (tabela de busca).
LZW	Algortimo de compressão de Lempel, Ziv e Welch.
MAC	*Message Authentication Code* (código de autenticação de mensagem).
Mallory	Nome fictício de uma invasora maliciosa.
Marca	Abreviação de “marca d’água” (*watermark*).
NPD	*Net Payload Data* (carga útil de dados, sem informações adicionais).
OD	*Ordered Dithering* (excitação ordenada).
PDF	*Portable Document Format* (formato portável de documento).
ProEikon	Biblioteca para Processamento de Imagens e Visão Computacional de autoria do Prof. Dr. Hae Yong Kim do Departamento de Engenharia de Sistemas Eletrônicos da Escola Politécnica da Universidade de São Paulo.
PWLC	*Pair-Wise Logical Computation* (computação lógica baseada em pares).
RATC	*Reversible Authentication watermarking by Template ranking with symmetrical Central pixels* (marca d’água de autenticação reversível pelo ranqueamento de modelos com pixels centrais simétricos).
RDTC	*Reversible Data hiding by Template ranking with symmetrical Central pixels* (esteganografia reversível pelo ranqueamento de modelos com pixels centrais simétricos).
Resumo	Abreviação de “resumo criptográfico”.
RGB	*Red, Green, Blue* (vermelho, verde, azul).
RLE	*Run Length Encoding* (codificação por comprimento de carreiras).
RSA	Esquema de criptografia de chave pública de Rivest, Shamir e Adleman.
TIFF	Tagged Image File Format. Formato de armazenamento de imagens.

UNIX	Sistema operacional.
VIS	*Visual Impact Score* (nota de impacto visual).
XOR	*Exclusive OR* (operador ou exclusivo).

SUMÁRIO

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

A quantidade de informações geradas pela humanidade vem crescendo muito nos últimos anos, devido às facilidades proporcionadas por novas tecnologias usadas para a criação, armazenamento e distribuição de informações (AFIF, 2004).

Com a evolução da informática, do hardware e dos meios de armazenamento de informação, a grande maioria das informações como documentos de texto, sons, imagens estáticas e vídeos, vem sendo armazenada no formato digital.

Este formato possui algumas vantagens em relação aos antigos formatos (papel, meios magnéticos e outros): pode ser facilmente distribuído (principalmente após o advento da *World Wide Web*, o que facilitou muito o acesso das pessoas à informação), mantém a qualidade do conteúdo após a cópia ou duplicação, tem um volume menor (facilitando o armazenamento), tem menor custo, entre outras.

Porém, todas essas facilidades proporcionadas pela tecnologia também trouxeram novos problemas a serem considerados na manutenção de alguns aspectos dos dados digitais. Um desses problemas é o reconhecimento de propriedade para estabelecimento de direitos autorais e *copyright* (por exemplo, para fotografias jornalísticas). Outro problema é a verificação da integridade da informação digital, pois a mesma pode ser facilmente corrompida. Uma preocupação importante é a autenticidade da informação, isto é, ter certeza de quem é a fonte de informação. Outra preocupação é a irretratabilidade, como é o caso de imagens legais, médicas e similares. Portanto, torna-se necessário desenvolver meios para a proteção da informação digital.

Alguns métodos de proteção de cópia (DCCA De-CSS, Adobe e-books, Macrovision FlexNet, etc.) são uma alternativa em direção a proteção de dados digitais. Estes métodos tipicamente se apóiam em mecanismos criptográficos para executar suas tarefas. Todavia, todo conteúdo digital, uma vez decriptografado, pode ser copiado, falsificado e distribuído facilmente (AFIF, 2004). Uma forma de complementar esses

mecanismos é utilizar o conceito de marca d'água digital (*digital watermark*). Uma marca d'água digital insere um dado imperceptível no próprio conteúdo digital a ser distribuído, de maneira análoga às marcas materiais há muito aplicadas a documentos impressos. Marcas d'água digitais não impedem a reprodução, adulteração, redistribuição e modificação dos dados em si, contudo, elas podem oferecer mecanismos para identificar a origem da distribuição ilegal dos dados ou detectar qualquer forma alteração não autorizada sobre os dados.

Apesar de muitas técnicas de marca d'água poderem ser aplicadas diretamente para diferentes tipos de dados digitais, este trabalho irá tratar somente das marcas para imagens digitais 2-D estáticas. Quando não houver perigo de confusão, será utilizada a palavra "marca" como sinônimo de "marca d'água digital".

Em geral as marcas d'água digitais são classificadas como "frágeis" ou "robustas". Um marca "robusta" é resistente à maioria dos procedimentos de manipulação de imagens, sendo ideal para o estabelecimento de direitos autorais e *copyright*. As marcas "frágeis" ou "de autenticação" (*authentication watermarks*), por sua vez, não resistem aos procedimentos de manipulação, de maneira que qualquer alteração, por menor que seja, irá corromper a marca. Este tipo de marca é ideal para checar a integridade das imagens. Este trabalho irá tratar apenas de marcas frágeis ou de autenticação.

Um exemplo de aplicação de uma marca de autenticação de chave pública seria a criação de uma câmera digital segura. Tal câmera poderia ser capaz de armazenar as imagens capturadas, já com uma marca d'água digital inserida nas mesmas. A assinatura digital da imagem seria calculada com a aplicação da chave privada da câmera (que deveria estar armazenada de forma segura num circuito integrado dentro da câmera) e armazenada na própria imagem. Desta forma, a integridade e a autenticidade da imagem poderiam ser verificadas posteriormente usando-se um programa, que poderia ser distribuído publicamente, para decodificar a assinatura digital usando a chave pública. Este exemplo está baseado no artigo de Friedman (1993) e nos comentários feitos por Kim (2004b).

Um outro exemplo de aplicação seria a autenticação de imagens distribuídas por uma rede. Uma pessoa, antes de transmitir uma imagem pela rede, poderia inserir uma marca digital utilizando sua chave privada. O receptor da imagem poderia verificar a autenticidade e a integridade da imagem utilizando sua chave pública.

Neste exemplo, não há como falsificar a imagem sem o conhecimento da chave privada. Também não é possível enviar uma imagem em nome de uma pessoa sem conhecer a sua chave privada.

Um exemplo que se aplica exclusivamente a imagens binárias seria a criação de uma “máquina de FAX confiável”, utilizando os mesmos conceitos da câmera digital segura de Friedman (1993). Esta máquina poderia conter internamente uma chave privada e inserir uma marca d’água em todos os documentos transmitidos por ela. O receptor de FAX, usando a chave pública da máquina transmissora, poderia verificar que o documento foi originado de uma máquina específica de FAX e que o documento não foi manipulado.

Em geral, as marcas d’água digitais utilizam-se de técnicas esteganográficas (*data hiding techniques*) para a inclusão da marca dentro da própria informação digital. Uma técnica esteganográfica se preocupa apenas em como esconder uma informação dentro de outra, não se preocupando com a finalidade do armazenamento ou com o conteúdo a ser armazenado.

A maioria das técnicas esteganográficas modifica e conseqüentemente distorce a imagem hospedeira para que a informação adicional possa ser inserida. Essa distorção geralmente é pequena, mas irreversível. Porém, em muitos campos como legal, médico, astronômico e militar, uma técnica esteganográfica reversível (*reversible data hiding*) é importante, pois permite a exata restauração (sem perda) da imagem original depois que a informação adicional é extraída.

Um exemplo de uso de uma marca de autenticação reversível para imagens binárias é o armazenamento seguro de fotocópias de cheques bancários. Considere um sistema onde cheques são escaneados, protegidos com um esquema de autenticação baseado em uma técnica esteganográfica reversível e enviados através da Internet. Em muitos casos, os documentos marcados são suficientes para se distinguir sem ambigüidade o seu conteúdo. Porém, no caso de dúvida, a recuperação do documento original (desmarcado) é muito interessante. Um estudo sobre armazenamento digital de cheques bancários pode ser visto em Dias (2006). Da mesma forma, a recuperação da imagem original pode ser importante quando estão envolvidas imagens médicas, pois a alteração de alguns pixels da imagem pode causar dúvidas no diagnóstico do paciente. O mesmo ocorre nos campos militar, legal e astronômico.

Atualmente, já foram desenvolvidas técnicas para a inserção/verificação de marca d'água em diversos tipos de imagem, porém ainda existem alguns formatos que precisam ser protegidos, como é o caso do formato JBIG2. Este formato tem se mostrado um dos mais eficientes (DIAS, 2006) em termos de compressão de imagens binárias atualmente. O formato JBIG2 é um padrão internacional para compressão de imagens binárias com ou sem perda de informação. Este formato decompõe a imagem em 3 tipos de região (texto, meio-tom e genérica) e codifica cada uma delas usando o método mais apropriado.

O formato JBIG2 tem se tornado popular devido ao formato PDF, criado pela *Adobe Systems Incorporated*, que incorporou um filtro para JBIG2 a partir da especificação 1.4 deste formato (ATALASOFT, 2006), o que reduziu em muito o tamanho dos arquivos codificados. O tamanho final de um arquivo codificado com JBIG2 chega a ser de 2 a 5 vezes menor do que se comprimido pelos métodos industriais tradicionais (TIFF CCITT G4). Este formato também possui um alto desempenho em termos de velocidade de descompressão (JPEG COMMITTEE, 2004).

1.2 OBJETIVOS

Parte dos documentos que trafegam através de redes de computadores e da Internet são imagens binárias que necessitam de proteção para garantir a sua autenticidade e a sua integridade. O formato JBIG2 tem se mostrado um dos mais eficientes para a compressão deste tipo de imagem, principalmente textos escaneados.

De acordo com nossas pesquisas, não existe atualmente nenhuma técnica esteganográfica ou marca d'água digital proposta para o formato JBIG2. Desta forma, um dos objetivos deste trabalho é desenvolver uma técnica esteganográfica para ocultar informações nas regiões de texto e de meio-tom de uma imagem codificada no formato JBIG2. Esta técnica deve gerar imagens marcadas de boa qualidade visual e deve poder ser aplicada a arquivos comprimidos com ou sem perda de informações. Um outro objetivo é utilizar esta técnica para criar uma marca de autenticação segura para imagens no formato JBIG2.

Em muitos campos como legal, médico, astronômico e militar, uma técnica esteganográfica reversível é importante, pois permite a exata restauração da imagem original depois que a informação adicional é extraída.

Também de acordo com nossas pesquisas, não existe nenhuma técnica esteganográfica reversível que possa ser aplicada de forma eficiente a imagens binárias. Portanto, outro objetivo deste trabalho é desenvolver uma técnica reversível para ocultar informações em imagens binárias. Esta técnica esteganográfica deve gerar imagens hospedeiras de boa qualidade visual e deve permitir a restauração da imagem original após a extração das informações ocultas. Um outro objetivo deste trabalho é utilizar esta técnica para criar uma marca d'água de autenticação reversível para imagens binárias.

Está fora do escopo deste trabalho a análise e desenvolvimento de marcas robustas, o estudo de técnicas criptográficas de chave secreta ou chave pública e a análise perceptual da qualidade visual das imagens marcadas.

1.3 CONTRIBUIÇÕES ORIGINAIS

Esta tese apresenta, como contribuição, um total de 9 técnicas originais. No capítulo 3 deste trabalho são propostas três técnicas esteganográficas para imagens binárias codificadas no formato JBIG2:

1) DHCCJ - *Data-Hiding by Coordinate Changing for JBIG2* (esteganografia por mudança de coordenada para JBIG2). Técnica publicada em (PAMBOUKIAN; KIM; DE QUEIROZ, 2005);
2) DHTRJ - *Data Hiding by Template Ranking for JBIG2* (esteganografia pelo ranqueamento de modelos para JBIG2). Técnica publicada em (PAMBOUKIAN; KIM; DE QUEIROZ, 2005);
3) DHCJT - *Data Hiding by template ranking with symmetrical Central pixels for JBIG2 Text region* (esteganografia pelo ranqueamento de modelos com pixels centrais simétricos para a região de texto do JBIG2). Técnica publicada em (PAMBOUKIAN; KIM, 2005).

Ainda no capítulo 3 são propostas quatro técnicas para a inserção/extração de marca d'água digital de autenticação em arquivos JBIG2:

4) AWTRJ - *Authentication Watermarking by Template Ranking for JBIG2* (marca d'água de autenticação pelo ranqueamento de modelos para JBIG2). Técnica publicada em (PAMBOUKIAN; KIM; DE QUEIROZ, 2005);

5) AWCJT - *Authentication Watermarking by template ranking with symmetrical Central pixels for JBIG2 Text region* (marca d'água de autenticação pelo ranqueamento de modelos com pixels centrais simétricos para a região de meio-tom do JBIG2). Técnica publicada em (PAMBOUKIAN; KIM, 2005);

6) AWCJH - *Authentication Watermarking by template ranking with symmetrical Central pixels for JBIG2 Halftone region* (marca d'água de autenticação pelo ranqueamento de modelos com pixels centrais simétricos para a região de meio-tom do JBIG2);

7) AWTCJ - *Authentication Watermarking by Template ranking with symmetrical Central pixels for JBIG2* (marca d'água de autenticação pelo ranqueamento de modelos com pixels centrais simétricos para JBIG2).

No capítulo 4 é proposta uma técnica esteganográfica reversível para imagens binárias:

8) RDTC - *Reversible Data hiding by Template ranking with symmetrical Central pixels* (esteganografia reversível pelo ranqueamento de modelos com pixels centrais simétricos). Técnica publicada em (PAMBOUKIAN; KIM, 2006).

Ainda no capítulo 4 é proposta uma técnica reversível para a inserção/extração de marca d'água digital de autenticação em imagens binárias:

9) RATC - *Reversible Authentication watermarking by Template ranking with symmetrical Central pixels* (marca d'água de autenticação reversível pelo ranqueamento de modelos com pixels centrais simétricos). Técnica publicada em (PAMBOUKIAN; KIM, 2006).

Vale ressaltar que o artigo (PAMBOUKIAN; KIM, 2006) que apresenta as duas técnicas reversíveis (RDTC e RATC) recebeu Menção Honrosa no 6º SBSEG - Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais em 2006.

1.4 IMPLEMENTAÇÃO DAS TÉCNICAS PROPOSTAS

Todas as técnicas propostas neste trabalho foram implementadas em C++ utilizando os compiladores DJGPP e Dev-C++ e as bibliotecas para Processamento de Imagens e Visão Computacional IMG (KIM, 2004a) e ProEikon (KIM, 2006) de autoria do Prof. Dr. Hae Yong Kim do Departamento de Engenharia de Sistemas Eletrônicos da Escola Politécnica da Universidade de São Paulo.

1.5 ORGANIZAÇÃO DA TESE

Esta tese foi escrita seguindo as normas da Escola Politécnica da Universidade de São Paulo (2006) e está organizada da seguinte forma:

- o capítulo 2 traz definições, conceitos e informações básicas necessárias para o fácil entendimento das propostas deste trabalho. Mostra algumas técnicas para o ocultamento de informações e a marcação de imagens contone (coloridas e em níveis de cinza) e binárias, apontando a vulnerabilidade de cada uma delas. Mostra também algumas técnicas esteganográficas reversíveis;
- o capítulo 3 faz uma introdução ao formato JBIG2, explicando detalhadamente sua estrutura, e apresenta propostas de técnicas esteganográficas e marcas d'água de autenticação para este formato de arquivo. Apresenta também resultados experimentais obtidos pela aplicação das técnicas propostas;
- o capítulo 4 mostra alguns dos principais algoritmos de compressão e apresenta uma proposta de técnica esteganográfica reversível para imagens binárias e uma proposta de marca d'água reversível para imagens binárias. Apresenta também resultados experimentais obtidos pela aplicação das técnicas propostas;
- o capítulo 5 apresenta as conclusões deste trabalho e sugestões para a continuidade do trabalho.

2 ESTEGANOGRAFIA E MARCAS D'ÁGUA

2.1 INTRODUÇÃO

Neste capítulo, serão definidos alguns termos importantes para a compreensão deste trabalho como esteganografia (*data hiding*), marca d'água digital (*digital watermark*), marcas frágeis ou de autenticação, marcas robustas, criptografia, função de *hash*, assinatura de autenticação (AS – *Authentication Signature*), assinatura digital (DS – *Digital Signature*), código de autenticação de mensagem (MAC – *Message Authentication Code*), entre outros.

Em seguida, serão descritas as principais marcas de autenticação propostas para imagens estáticas de tonalidade contínua (isto é, imagens em níveis de cinza e coloridas), analisando-se a vulnerabilidade de cada uma delas.

Depois, serão apresentadas algumas técnicas esteganográficas e marcas d'água propostas para imagens binárias.

Por último, serão analisadas algumas técnicas reversíveis propostas para imagens coloridas, em níveis de cinza e binárias.

2.2 ESTEGANOGRAFIA × MARCA D'ÁGUA

Esteganografia (em inglês, *steganography* ou *information hiding* ou *data hiding*) é o estudo das técnicas utilizadas para esconder uma informação secreta dentro de outro tipo de informação. As técnicas esteganográficas possuem as seguintes características:

- as informações são inseridas de maneira imperceptível;
- não há perda de qualidade da informação hospedeira;
- a informação secreta pode ser recuperada posteriormente.

Por exemplo, uma seqüência de bits pode ser inserida em uma imagem através da modificação de alguns de seus pixels, sem degradação perceptível da qualidade da imagem. A esteganografia não se preocupa com a utilidade da informação escondida ou com a facilidade de remoção da mesma.

As marcas d'água digitais (*digital watermarks*) utilizam técnicas esteganográficas para inserir, em uma informação digital, um sinal portador de informação que pode ser extraído posteriormente com a finalidade de se fazer uma asserção sobre a informação hospedeira, identificando alterações maliciosas ou acidentais. Geralmente, uma marca d'água digital é classificada como "robusta" ou "frágil" dependendo da dificuldade de remoção da mesma.

Marcas robustas são projetadas para resistir a procedimentos comuns de manipulação de imagens (rotação, redimensionamento, recorte, compressão com perda, impressão/escaneamento, etc.). Uma boa marca robusta deveria ser impossível de ser removida a não ser que a qualidade da imagem resultante deteriorasse a ponto de destruir o seu conteúdo visual. Isto é, a correlação entre uma imagem marcada e a marca robusta nela inserida deveria permanecer detectável mesmo após um processamento digital, enquanto a imagem resultante do processamento continuar visualmente reconhecível e identificável como a imagem original (KIM, 2004b). Este tipo de marca é tipicamente usado para verificação de direitos autorais (*copyright*) e impressão digital (*fingerprinting*), pois não pode ser removida com facilidade. Uma impressão digital pode ser definida como uma informação reduzida, produzida de forma segura e unívoca, capaz de representar as características fundamentais da informação completa. Apesar de muitas pesquisas terem sido realizadas na área de marcas robustas, ainda não se conseguiu obter uma marca robusta realmente segura.

Por outro lado, marcas frágeis ou de autenticação (*authentication watermarks*) são facilmente corrompidas por qualquer processo de manipulação de imagem. Portanto, marcas d'água para verificar a integridade e a autenticidade de uma imagem devem ser frágeis, pois se a marca for removida ou a imagem for adulterada, o algoritmo de detecção da marca identificará corretamente a corrupção da imagem.

Existem ainda as marcas semifrágeis, capazes de resistir a determinados processos de manipulação de imagens. Este trabalho irá tratar apenas de marcas frágeis ou de autenticação.

2.3 MARCAS FRÁGEIS OU DE AUTENTICAÇÃO

As marcas frágeis ou de autenticação podem ser classificadas em três tipos: sem chave, com chave secreta (cifra simétrica) e com chave pública (cifra assimétrica).

Uma marca de autenticação sem chave funciona como uma espécie de "*check-sum*", sendo utilizada para detectar alterações não intencionais na imagem, tais como erros de transmissão ou de armazenamento. Neste tipo de marca, se o algoritmo de inserção/verificação estiver disponível publicamente, qualquer pessoa pode marcar uma imagem ou verificar se uma imagem possui uma marca válida.

A marca de autenticação com chave secreta (cifra simétrica) pode ser usada para detectar alterações (intencionais ou não) feitas na imagem. O algoritmo para inserção/verificação da marca pode ser disponibilizado publicamente, porém é impossível inserir uma marca ou verificar se uma imagem possui uma marca válida sem o conhecimento da chave secreta. Neste tipo de marca, a mesma chave é utilizada tanto no processo de inserção quanto no de verificação. Este tipo de marca é similar aos códigos de autenticação de mensagem, sendo que a única diferença é que o código de autenticação é inserido na imagem em vez de ser armazenado separadamente.

As marcas de autenticação com chave pública (cifra assimétrica) utilizam a criptografia de chave pública para inserir uma assinatura digital na imagem. A marca é inserida utilizando-se uma chave privada e a verificação da validade da marca é feita utilizando-se uma chave pública. É impossível inserir uma marca válida conhecendo-se apenas a chave pública, porém, para verificar a validade de uma marca, basta conhecer a chave pública.

Entre os três tipos de marca de autenticação, a de chave pública é a que oferece mais recursos.

Os primeiros trabalhos de marca d'água de autenticação foram inspirados pelo artigo de Friedman (1993) que propõe a criação de uma câmera digital segura. Neste tipo de câmera, cada imagem capturada produziria dois arquivos de saída: o primeiro contendo a imagem propriamente dita e o segundo contendo uma assinatura digital produzida aplicando-se a chave privada da câmera (que deveria estar armazenada

de forma segura num circuito integrado dentro da câmera). A integridade e a autenticidade da imagem poderiam ser verificadas usando-se um programa para decodificar a assinatura digital. Este programa receberia como entrada a imagem digital, a assinatura digital e a chave pública da câmera. Então calcularia o *hash* da imagem digital, decriptografaria a assinatura digital e verificaria se os dois resumos criptográficos obtidos são iguais. O programa de verificação poderia ser distribuído livremente aos usuários, pois a segurança desse esquema está no segredo da chave. Segundo Kim (2004b), o conceito de marca d'água de autenticação de chave pública poderia ser incorporado ao esquema proposto por Friedman (1993), para melhorá-lo, inserindo a assinatura digital na própria imagem.

2.4 CRIPTOGRAFIA E ASSINATURA DIGITAL

2.4.1 **Criptografia**

A palavra criptografia é formada a partir da concatenação dos termos gregos *kryptós* (escondido, oculto) e *gráphein* (grafar, escrever). A criptografia é considerada a arte de escrever em código e pode ser utilizada para uma comunicação mais segura entre dois agentes através de um canal aberto. De modo geral, o agente transmissor codifica a informação original, convertendo-a em dados incompreensíveis, e os transmite através do canal aberto. Por outro lado, o agente receptor recebe esses dados, decodifica os mesmos e recupera a informação original.

O processo de transformar uma informação em dados incompreensíveis pode ser identificado por diferentes termos como: codificar, criptografar, encriptar e cifrar. O processo inverso pode ser identificado por: decodificar, decriptografar, decriptar ou decifrar.

Para que uma mensagem seja cifrada é necessário que o usuário forneça uma chave ao algoritmo criptográfico (Figura 1). Para cada chave, o algoritmo irá gerar uma mensagem cifrada diferente. Sem o conhecimento da chave correta não é possível decifrar a mensagem. Assim, para manter uma informação secreta, basta cifrar a informação e manter em sigilo a chave.

Figura 1 – Processo de cifragem (ARISP/IRIB, 2007).

Em criptografia simétrica ou de chave secreta, a mesma chave é utilizada tanto no processo de cifragem quanto no processo de decifragem. O transmissor utiliza a chave para transformar a mensagem original em texto cifrado e o receptor utiliza a mesma chave para executar a transformação inversa, recuperando a mensagem original. Um dos problemas desta técnica é que o transmissor e o receptor precisam combinar a chave secreta. A troca de chaves deve ser feita de forma segura, uma vez que todos que conhecem a chave podem decifrar a informação cifrada ou mesmo produzir uma informação cifrada.

Em criptografia assimétrica ou de chave pública, são utilizadas duas chaves: uma chave pública e outra privada. Conhecendo a chave privada, é fácil e rápido calcular a chave pública correspondente. Porém, o contrário é uma tarefa extremamente difícil computacionalmente (levaria talvez séculos utilizando os supercomputadores atuais).

Figura 2 – Confidencialidade utilizando criptografia assimétrica (ARISP/IRIB, 2007).

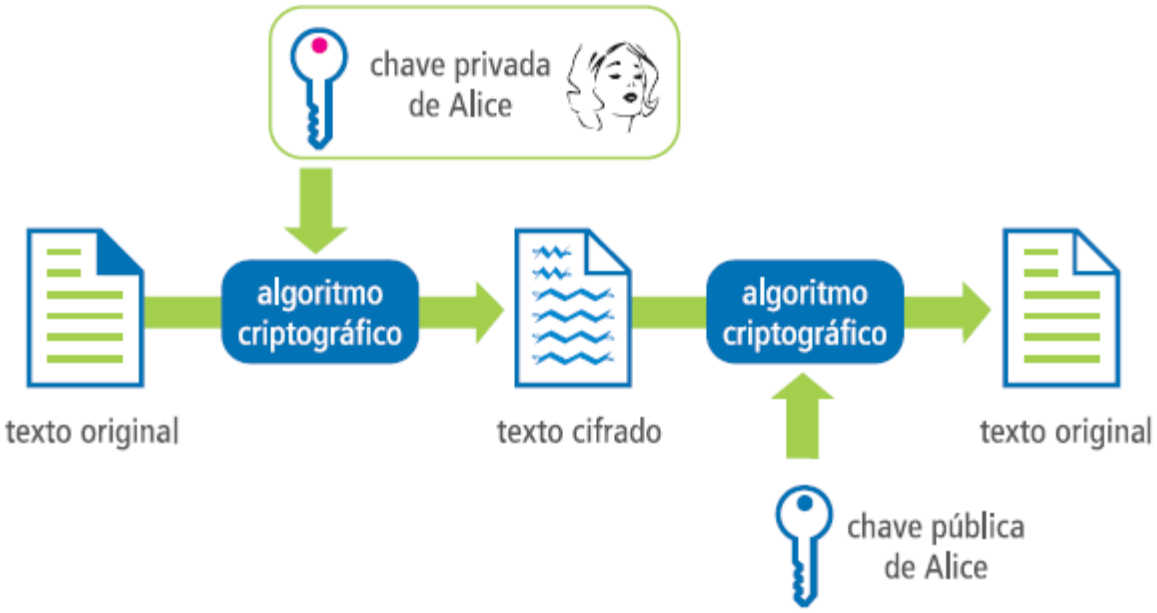


Figura 3 – Autenticidade utilizando criptografia assimétrica (ARISP/IRIB, 2007).

O objetivo tradicional da criptografia de chave pública é permitir uma comunicação confidencial entre as duas partes (Figura 2). Por exemplo, para enviar uma mensagem secreta que somente o receptor Bob possa ler, Bob primeiro torna a sua chave pública conhecida publicamente. Desta forma, qualquer pessoa que queira enviar uma mensagem secreta a Bob deve criptografar a mensagem usando esta chave pública e enviá-la a Bob. Bob, sendo único possuidor da chave privada, é a única pessoa capaz de decriptografar a mensagem. Note que a necessidade de se combinar uma chave secreta entre o transmissor e o receptor foi eliminada (KIM, 2004b).

Quando a criptografia de chave pública é utilizada na implementação de marcas d'água de autenticação, o processo que acaba de ser descrito precisa ser implementado "ao contrário" (Figura 3). Neste caso, por exemplo, a transmissora da mensagem Alice guarda uma chave privada, e a chave pública correspondente é disponibilizada publicamente a qualquer receptor que queira decriptografar. Este procedimento fornece um meio para assegurar que as mensagens não foram forjadas: somente Alice, que possui a chave privada, poderia ter codificado uma mensagem que é decifrável pela correspondente chave pública.

2.4.2 **Função de *Hash* e Resumo Criptográfico**

O conceito de função de *hash* é essencial à obtenção de assinaturas digitais compactas e eficientes. Intuitivamente, uma função de *hash* calcula, rápida, segura e univocamente, representantes adequadamente curtos (chamados “resumos criptográficos“, ou simplesmente “resumos“) para mensagens arbitrariamente longas (BARRETO, 2003).

Um resumo pode ser comparado a uma impressão digital, pois cada documento possui um valor único de resumo e até mesmo uma pequena alteração no documento resulta em um resumo completamente diferente (ARISP/IRIB, 2007).

Esses resumos são cifrados em lugar das próprias mensagens, mantendo em nível aceitável o esforço computacional comumente encontrado durante a operação de algoritmos assimétricos, que são muito lentos.

Uma diferença entre uma função de *hash* de via única e um algoritmo de cifragem é que na função de *hash* não existe nenhuma possibilidade de se recuperar a informação original, pois várias mensagens podem produzir um mesmo resumo. Já no algoritmo de cifragem, apenas uma mensagem pode originar uma determinada mensagem cifrada. Outra diferença é que uma função de *hash* de via única não possui chaves.

Normalmente o tamanho dos resumos gerados pela função de *hash* é pequeno, de 16 ou 20 bytes (PUTTINI; DE SOUZA JR., 2007). Portanto, um número muito grande de mensagens diferentes pode resultar em um mesmo resumo criptográfico. Porém, é importante salientar que a probabilidade de duas mensagens aleatórias resultarem em um mesmo resumo é desprezível.

2.4.3 **Assinatura Digital**

As assinaturas digitais são construídas sobre técnicas de criptografia de chave pública. Elas permitem autenticar o conteúdo da mensagem e identificar a identidade do emissor. Tais assinaturas são produzidas com o apoio de uma função de *hash*.

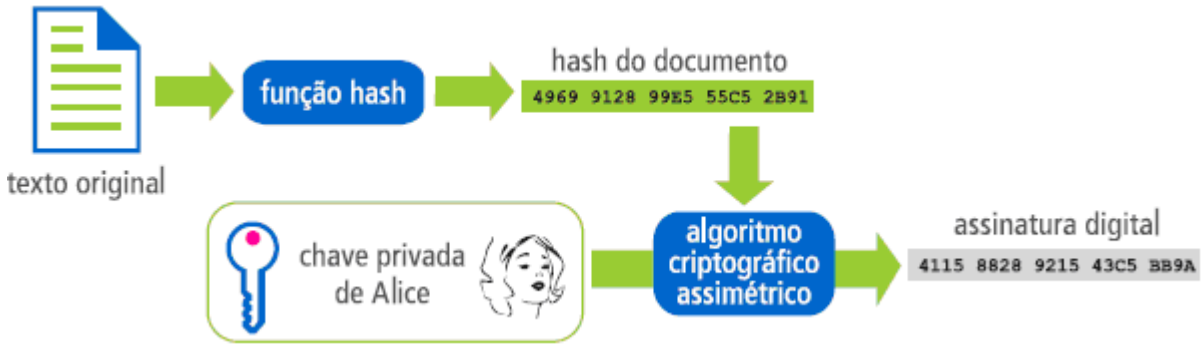


Figura 4 – Assinatura digital utilizando algoritmos de chave pública (ARISP/IRIB, 2007).

Os resumos criptográficos gerados pelo *hash* são criptografados utilizando a chave privada, em lugar das próprias mensagens, acelerando tanto o processo de criação da assinatura como o processo de verificação. O resultado é uma informação, chamada “assinatura digital“, que acompanha a mensagem original (Figura 4). Desta forma, por exemplo, a mensagem original pode ser lida por todos, porém se um receptor chamado Bob desejar autenticá-la, Bob pode decriptografar a assinatura digital da mensagem usando a chave pública de Alice, recuperando o resumo da mensagem. Este resumo deve ser idêntico ao resumo da mensagem original, se a mensagem não tiver sido adulterada (Figura 5).

Uma assinatura digital deve satisfazer os seguintes critérios (PUTTINI; DE SOUZA JR., 2007):

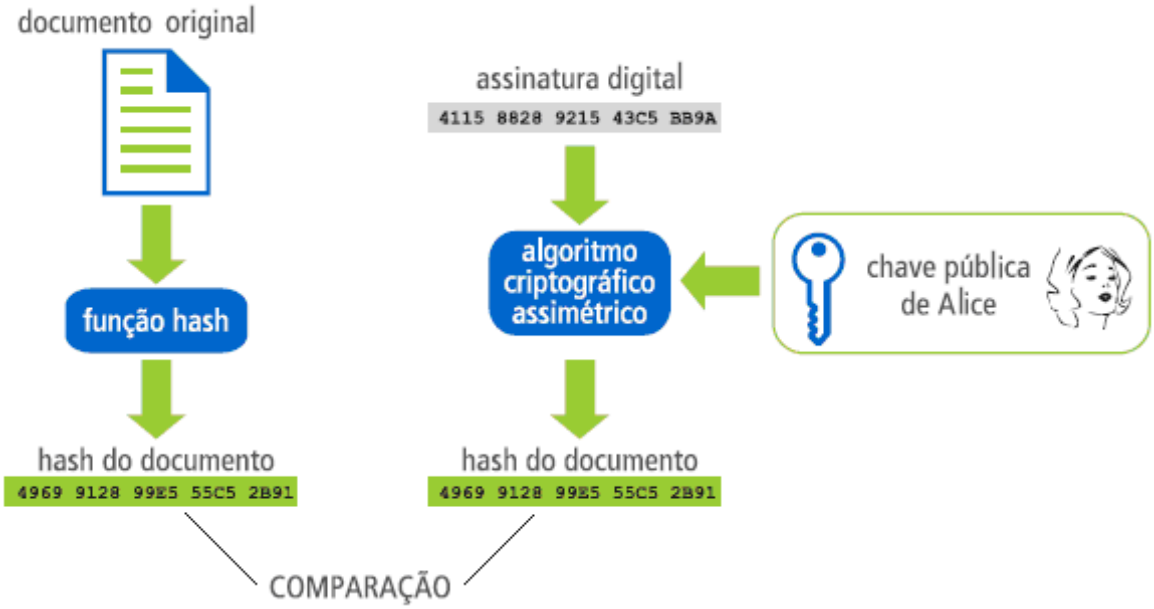


Figura 5 – Conferência de Assinatura digital (ARISP/IRIB, 2007).

1) a assinatura não pode ser falsificada. Só o emissor conhece a sua chave privada;
2) a assinatura é autêntica. Quando o receptor verifica o documento com a chave pública do emissor, ele comprova que quem assinou o documento foi o emissor;
3) a assinatura não é reutilizável. A assinatura em um documento não pode ser transferida para qualquer outro documento;
4) o documento assinado é inalterável. Qualquer alteração invalida a assinatura;
5) a assinatura não pode ser repudiada. O receptor não precisa do emissor para comprovar sua assinatura; o emissor não pode negar que assinou o documento.

O conceito de assinatura digital vem sendo aplicado atualmente em técnicas de marca d'água de autenticação (AWTs – *Authentication Watermarking Techniques*). Em uma AWT típica, baseada em criptografia, uma assinatura de autenticação (AS – *Authentication Signature*) é computada com base na imagem completa e inserida de alguma forma dentro da própria imagem. A AS contém informação sobre o conteúdo da imagem hospedeira que pode ser checado para verificar sua integridade.

Em criptografia, uma AS é chamada de Código de Autenticação de Mensagem (MAC – *Message Authentication Code*) se for usada uma cifra simétrica (chave secreta) e Assinatura Digital (DS – *Digital Signature*) se for usada uma cifra assimétrica (chave pública).

Um dos problemas encontrados na criação de AWTs é que a inserção da AS na imagem altera a própria imagem, modificando sua AS e invalidando a marca. Tipicamente, a imagem tem que ser dividida de alguma forma em pelo menos duas partes: uma porção que irá manter a integridade da imagem, pois não sofrerá mudanças durante a inserção da marca e uma outra porção que será utilizada para armazenar a AS. Porém, dividindo a imagem em duas partes, em alguns casos, torna-se possível a ocorrência de ataques de paridade. Um ataque de paridade permite que a imagem seja adulterada sem que a AWT possa detectar a alteração. Este tipo de ataque será descrito posteriormente neste capítulo.

2.4.4 **Algoritmos de criptografia**

A AS escolhida para autenticar uma imagem deve ter um comprimento suficiente para garantir a segurança. ASs muito curtas não são seguras contra ataques de aniversário (vistos posteriormente neste capítulo). Geralmente, um MAC com 128 bits é considerado seguro. A bem conhecida DS, RSA (RIVEST; SHAMIR; ADLEMAN, 1978), é considerada segura com 1024 bits. O novo esquema, DSA (*Digital Signature Algorithm*), é considerado seguro com 320 bits. O recente esquema, BLS (BONEH; LYNN; SHACHAM, 2002), é seguro com apenas 160 bits (KIM, 2004b). Como o estudo dos vários tipos de AS está além do escopo deste trabalho, para maiores detalhes, é sugerido ao leitor a consulta a livros de criptografia como, por exemplo, Schneier (1996) ou Menezes; Van Oorschot e Vanstone (1997).

2.5 TÉCNICAS EXISTENTES PARA IMAGENS COLORIDAS E EM NÍVEIS DE CINZA

Em imagens de tonalidade contínua (coloridas e em níveis de cinza) não comprimidas, existe uma maneira “natural” de embutir marcas de autenticação. Como a cor de um pixel é codificada por vários bits (tipicamente 8 bits para imagens em níveis de cinza e 24 bits para imagens coloridas no padrão RGB), pode-se inserir os dados nos bits menos significativos (LSBs – *Least Significant Bits*) de cada pixel. Alterar os LSBs afeta muito pouco a qualidade da imagem, ao mesmo tempo em que se conhece exatamente os bits que estão carregando a marca. O mesmo não ocorre com as imagens binárias. Numa imagem binária, cada pixel consiste de um único bit, de forma que não existem LSBs. Isto traz dificuldades especiais para projetar marcas de autenticação para este tipo de imagem.

Na literatura, existem muitas AWTs para imagens de tonalidade contínua (*contone*) como, por exemplo, Yeung e Mintzer (1997), Wong (1998), Barreto; Kim e Rijmen (2001, 2002), Li; Lou e Chen (2000), Zhao e Koch, (1995) e Celik et al. (2002b). Muitas destas técnicas estão sujeitas a ataques (técnicas de adulteração) que fazem

com que a alteração da imagem não seja detectada pela AWT. A seguir serão vistas algumas dessas AWTs e a vulnerabilidade de cada uma delas, com base nas análises feitas por Barreto (2003) e Kim (2004).

2.5.1 **Yeung-Mintzer**

Yeung e Mintzer (1997) propuseram uma das primeiras técnicas de marca d'água de autenticação. Esta marca é inserida pixel a pixel, com o auxílio de uma imagem logotipo binária e de uma Tabela de Busca (LUT – *Look-Up Table*). Nesta técnica, como há apenas 1 bit de marca para autenticar cada pixel, há 50% de chance de uma alteração de um único pixel passar despercebida. Porém, se uma região de tamanho razoável for alterada, muito dificilmente essa alteração passará despercebida. Este esquema funciona com chave secreta, isto é, a inserção e a detecção da marca devem ser feitas utilizando a mesma chave.

Demonstrou-se mais tarde (HOLLIMAN; MEMON, 2000) que este esquema é completamente inseguro e está sujeito a um ataque denominado ataque de falsificação (*counterfeiting attack*). Isto é, um falsificador poderia inserir uma marca válida em qualquer imagem dispondo apenas de um pequeno conjunto de imagens validamente marcadas.

Inserção da marca de Yeung-Mintzer

Seja *B* uma imagem logotipo binária a ser inserida na imagem original *I* para produzir a imagem marcada *I'*. *B* deve ser do mesmo tamanho que *I*. Se os tamanhos das duas imagens forem diferentes, *B* deve ser replicada ou redimensionada para que fique do mesmo tamanho de *I*.

Esta técnica pode ser aplicada tanto para imagens em níveis de cinza (por exemplo, 8 bits por pixel ou 256 níveis de cinza) como para imagens coloridas (por exemplo, 3 bytes por pixel no formato RGB). Para ilustrar os procedimentos de inserção e verificação da marca, será utilizada uma imagem em níveis de cinza (o procedimento para marcação de imagens coloridas é muito semelhante).

O primeiro passo para a inserção desta marca é a criação de uma Tabela de Busca (LUT – *Look-Up Table*). Uma LUT *k* aleatória pode ser gerada sorteando 256 valores booleanos: k: {0...255} $\rightarrow$ {0,1}. Esta LUT *k* funciona como uma chave secreta e deve ser mantida em segredo.

Cada pixel $I(i,j)$ da imagem original *I* deve ser analisado e, se necessário, um ou mais bits (os menos significativos) devem ser alterados de acordo com algoritmo que se segue. Em geral, a ordem de análise dos pixels é de cima para baixo e da esquerda para a direita (*raster*).

Algoritmo 1 - Inserção da marca de Yeung-Mintzer

1) Calcular $k(I(i,j))$;
2) Se $k(I(i,j)) = B(i,j)$, nenhuma mudança é necessária, isto é, $I'(i,j) = I(i,j)$.
3) Se $k(I(i,j)) \neq B(i,j)$, o valor de $I(i,j)$ deve ser alterado para um nível de cinza próximo $I'(i,j)$ de modo que $k(I'(i,j)) = B(i,j)$.
4) O erro cometido ao aproximar $I(i,j)$ para $I'(i,j)$, isto é, $I(i,j) – I'(i,j)$ deve ser espalhado para os pixels vizinhos que ainda não foram analisados.

A difusão de erro, descrita no passo 4, assegura que o nível de cinza médio não é alterado localmente, o que garante uma alta qualidade visual da imagem marcada. Os autores desta técnica sugerem a utilização dos seguintes pesos:

- $W(i+1,j) = 0,5$;
- $W(i+1,j+1) = 0,0$;
- $W(i,j+1) = 0,5$;

Desta forma, os novos valores dos pixels vizinhos seriam:

- $I(i+1,j) = I(i+1,j) + 0,5\ (I(i,j) - I'(i,j))$;
- $I(i,j+1) = I(i,j+1) + 0,5\ (I(i,j) - I'(i,j))$;

Algoritmo 2 - Verificação da marca de Yeung-Mintzer

1) Utilizando a imagem marcada *I'* e a LUT *k*, construir a imagem binária de verificação *C*, onde o valor de cada pixel pode ser obtido por $C(i,j) \leftarrow k(I'(i,j))$.

2) Se a imagem de verificação *C* for igual à imagem inserida *B*, a imagem marcada *I′* não foi alterada. Caso contrário, houve alteração na região onde as imagens *C* e *B* forem diferentes.

Ataque de falsificação

Holliman e Memon (2000) demonstraram que é possível subverter completamente a marca de Yeung-Mintzer através de um ataque de falsificação (*counterfeiting attack*). A idéia é que, tendo-se algumas poucas imagens marcadas utilizando uma LUT *k*, é possível marcar validamente uma imagem *J* qualquer, sem conhecer a tabela *k* ou a imagem logotipo *B*. Além disso, é possível obter a tabela secreta *k* a partir de algumas imagens marcadas, conhecendo a imagem logotipo *B*.

O exemplo a seguir mostra como um ataque de falsificação pode ser utilizado para marcar uma imagem em níveis de cinza (para imagens coloridas o ataque é semelhante).

Suponha que Mallory, uma invasora maliciosa, quer inserir uma marca válida em uma imagem *J* qualquer, sem conhecer a LUT *k*. Suponha também que Mallory, de alguma forma, conheça a imagem logotipo *B* e tenha à disposição uma imagem *I′* onde a imagem *B* foi embutida utilizando a LUT *k*.

Para marcar a imagem *J*, Mallory divide os pixels de *I′* em dois subconjuntos disjuntos: o primeiro subconjunto S_0 de pixels com valor zero na imagem logotipo *B* e o segundo S_1 de pixels com valor um em *B*. Como só existem 256 níveis de cinza, e uma imagem de tamanho usual possui centenas de milhares de pixels, provavelmente haverá exemplos de praticamente todos os níveis de cinza. Em cada pixel $J(i, j)$, deve ser embutido o bit $B(i, j)$. Para isso, Mallory procura, no subconjunto S_0 ou S_1 correspondente ao bit $B(i, j)$, o pixel com o nível de cinza mais próximo possível do $J(i, j)$. Então, coloca esse valor no pixel (i, j) da imagem falsificada *J′*. Basta repetir este processo para todos os pixels da imagem *J* para obter a imagem corretamente marcada *J′*. Aliás, se quisesse otimizar a qualidade visual da imagem forjada , seria até possível executar um algoritmo de difusão de erro semelhante ao utilizado no algoritmo de inserção da marca d'água.

Se o tamanho da imagem *I′* for suficientemente grande para conter um pixel exemplar para cada nível de cinza (o que costuma acontecer na prática), a LUT secreta *k* pode ser completamente descoberta a partir dos subconjuntos S_0 e S_1.

Basta associar a cada nível de cinza em S_0 o bit 0 e a cada nível de cinza em S_1 o bit 1.

2.5.2 **Wong**

Em (WONG, 1997) foi proposta uma marca d'água de autenticação baseada em criptografia simétrica. Essa técnica foi melhorada em (WONG, 1998) para utilizar a criptografia assimétrica, tornando-se o primeiro trabalho de marca de autenticação de chave pública.

O esquema de Wong divide a imagem em blocos e assina cada bloco individualmente. Desta forma, é possível localizar o bloco onde a imagem foi alterada. Quanto menor o tamanho dos blocos, melhor a resolução de localização da alteração.

A marca d'água é inserida nos bits menos significativos (LSBs – *Least Significant Bits*) da imagem. Assim, é possível inserir 1 bit em cada pixel em imagens em níveis de cinza e 3 bits em cada pixel em imagens coloridas (formato RGB). A alteração dos LSBs é visualmente imperceptível.

Assim como o trabalho de Yeung e Mintzer (1997), o trabalho de Wong (1997, 1998) possui defeitos sérios de segurança, como ataque recortar-e-colar, ataque de falsificação e ataque de aniversário simples.

Algoritmo 3 - Inserção da marca de Wong

1) Particionar a imagem original em níveis de cinza I, de tamanho $M \times N$ pixels, em n blocos I_t ($0 \leq t < n$) de 8×8 pixels.
2) Replicar ou redimensionar a imagem logotipo B, para que B e I tenham o mesmo tamanho. Cada bloco I_t terá um bloco correspondente B_t na imagem logotipo.
3) Seja I^*_t o bloco obtido de I_t zerando todos os LSBs. Usando uma função de *hash* H criptograficamente segura, calcular o resumo $H_t = H(M,N, I^*_t)$. M e N entram na função de *hash* para detectar cortes das bordas da imagem (*cropping*).

4) Calcular o ou-exclusivo de H_t com B_t, obtendo o resumo marcado $H^{\wedge}_t$. Note que H_t e B_t possuem o mesmo tamanho (64 bits).
5) Criptografar $H^{\wedge}_t$ com a chave privada, gerando a AS S_t para o bloco t.
6) Inserir S_t nos LSBs de I^*_t, obtendo o bloco marcado I'_t.

Algoritmo 4 - Verificação da marca de Wong

1) Particionar a imagem marcada I', de tamanho $M \times N$ pixels, em n blocos I'_t como na inserção.
2) Retirar os LSBs de I'_t e decriptografar o resultado utilizando a chave pública, obtendo o bloco decriptografado D_t.
3) Seja I^*_t o bloco obtido de I'_t zerando todos os LSBs. Usando a mesma função de *hash* H utilizada na inserção, calcular o resumo $H_t = H(M,N, I^*_t)$.
4) Calcular o ou-exclusivo de H_t com D_t, obtendo o bloco de verificação C_t.
5) Se C_t e B_t (o bloco t da imagem logotipo) forem iguais, a marca está verificada. Caso contrário, a imagem foi adulterada.

Como foi visto nos algoritmos de inserção e verificação, o operador * indica limpar os LSBs e a marca ′ indica um bloco ou uma imagem com a assinatura embutida. Esta notação também será utilizada no restante desta subseção e nas subseções seguintes.

No algoritmo de verificação, note que a imagem logotipo B não precisa estar disponível publicamente, pois a imagem B deve possuir algum sentido visual (logotipo da empresa, por exemplo) e qualquer alteração na imagem marcada irá gerar uma imagem C completamente irreconhecível (parecida com ruído aleatório).

O esquema de Wong possui algumas fraquezas e está sujeito a uma série de ataques. Uma das fraquezas é sugerir o uso de uma assinatura RSA de 64 bits, que é completamente insegura. Uma RSA com chave de 64 bits pode ser fatorada em segundos usando um computador pessoal atual (KIM, 2004). Outra fraqueza é marcar os blocos de maneira independente, o que facilita a ocorrência de ataques.

Técnica de Li, Lou e Chen

Li; Lou e Chen (2000) propuseram uma ligeira modificação no esquema de Wong:

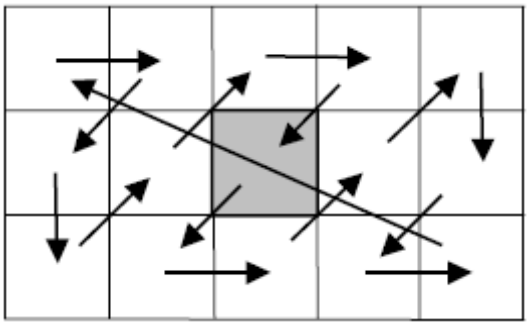


Figura 6 – Seqüência de varredura em *zig-zag* (BARRETO, 2003).

1) Dividir cada bloco em duas metades (direita e esquerda).
2) Trocar a metade direita de um bloco I^*_t com a metade direita do bloco seguinte $I^*_{(t+1) \bmod n}$, seguindo a ordem em *zig-zag* (Figura 6).
3) Assinar cada bloco combinado.
4) Inserir a assinatura nos LSBs do bloco I^*_t.

Ataque recortar-e-colar e ataque de falsificação

Um ataque muito simples, que pode ser utilizado com intenções maliciosas e não pode ser detectado pelo esquema de Wong, é o chamado ataque de “recortar-e-colar” (BARRETO; KIM, 1999; BARRETO, 2003; KIM, 2004).

Suponha que Mallory, uma invasora maliciosa, possui uma coleção de imagens legitimamente marcadas, todas elas do mesmo tamanho e contendo a mesma

(a) Imagem original. (b) Imagem adulterada.

Figura 7 – Ataque recortar-e-colar (BARRETO, 2003).

imagem embutida *B*. Como cada bloco é marcado separadamente, sem qualquer informação sobre a imagem hospedeira exceto as suas dimensões, é possível que Mallory selecione alguns blocos das imagens autênticas e construa com eles uma nova imagem cuja marca d'água será falsamente verificada como legítima. Pode até ser possível recortar um bloco de uma imagem e colá-lo dentro da mesma imagem mantendo a marca d'água inalterada. A Figura 7 mostra um exemplo deste ataque.

A técnica de Li, Lou e Chen também pode sofrer este tipo de ataque. Basta copiar os conteúdos (sem os LSBs) de dois semi-blocos de dois blocos vizinhos, por exemplo I^*_t e $I^*_{(t+1) \bmod n}$, e colá-los junto com a AS que se encontra nos LSBs do bloco I'_t.

Se o ataque recortar-e-colar for aplicado repetidamente, uma imagem inteira falsificada, mas com uma marca válida, pode ser construída. Esta é exatamente a idéia do ataque de falsificação (*counterfeiting attack*) de Holliman e Memon (2000).

Ataque de aniversário simples

O ataque de aniversário (MENEZES; VAN OORSCHOT; VANSTONE, 1997, seção 9.7) constitui um meio bem conhecido e poderoso para subverter assinaturas digitais.

O atacante procura por colisões, isto é, pares de blocos que possuem o mesmo resumo criptográfico. Segundo a análise de Nishimura e Sibuya (1990), se *H* é uma função de *hash* que pode gerar *n* resumos distintos e se dois conjuntos de resumos de *H*, com quantidade de elementos *r* e *s*, são gerados aleatoriamente, então o número esperado de colisões é $c \approx rs/n$.

Desta forma, um ataque bem sucedido pode ser montado obtendo-se uma base de dados com $s \approx n^{1/2}$ assinaturas válidas e gerando-se uma coleção de $r \approx n^{1/2}$ blocos forjados. Já que $c \approx 1$, uma assinatura válida provavelmente será encontrada com probabilidade superior a 50%.

O esquema de Wong utiliza uma função de *hash* de não mais que 64 bits. Então, espera-se que as colisões ocorram quando o atacante tiver coletado somente 2^{32} blocos.

Suponha que Mallory, uma invasora maliciosa, quer adulterar uma imagem *I'* (de tamanho $M \times N$) e que ela possui um grande banco de dados com imagens de tamanho $M \times N$, suficiente para se obter $s \approx 2^{32}$ assinaturas válidas.

Para efetuar a adulteração, Mallory irá substituir um bloco I'_t por um outro bloco J. Ela prepara $r \approx 2^{32}$ variantes de J visualmente equivalentes (J_1, J_2, ..., J_r), por exemplo alterando o segundo bit menos significativo de 32 pixels escolhidos de maneira aleatória dentro de J (os LSBs não podem ser utilizados, pois eles armazenam a marca). Mallory então procura um bloco D' em seu banco de dados que possua o mesmo resumo criptográfico de J^*_j, isto é, $H(M,N,J^*_j) = H(M,N,D^*)$.

Se este processo for repetido um número suficiente de vezes, uma imagem inteira falsificada pode ser gerada. Um ataque similar também pode ser executado contra a marca de Li, Lou e Chen.

Em geral, a única proteção contra o ataque de aniversário é aumentar o tamanho do resumo gerado pela função de *hash*. Porém, isto diminuiria a resolução de localização das alterações, pois os blocos teriam que ser maiores para hospedar mais dados inseridos.

2.5.3 **HBC – *Hash Block Chaining***

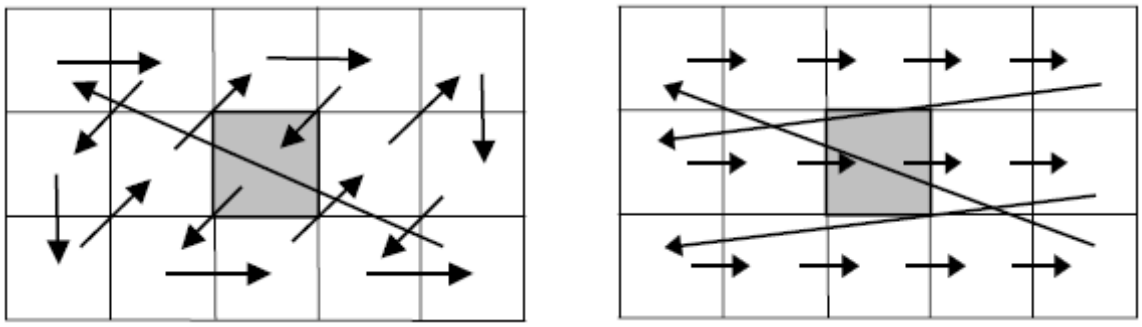
Esta subseção descreve soluções para impedir os ataques descritos anteriormente (ataque de falsificação, ataque recortar-e-colar e ataque de aniversário) e analisa novos tipos de ataque (ataque de transplante e ataque de aniversário melhorado).

Hash Block Chaining versão 1 (HBC1)

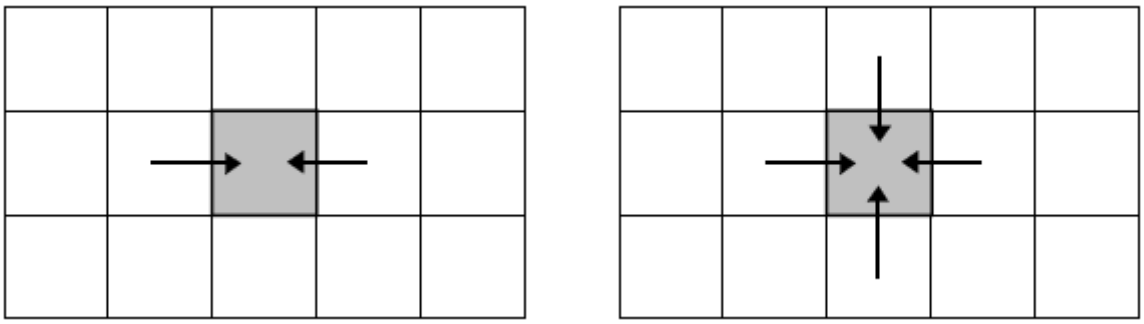
Conforme mostrado em (BARRETO; KIM, 1999; BARRETO; KIM; RIJMEN, 2000, 2001, 2002; HOLLIMAN; MEMON, 2000), a solução para evitar os ataques descritos nas subseções anteriores é utilizar uma informação contextual no cálculo do resumo criptográfico de cada bloco.

Barreto; Kim e Rijmen (2000) propuseram um esquema denominado *Hash Block Chaining*, versão 1 (HBC1) que calcula o resumo criptográfico H_t, alimentando a função de *hash* H com o conteúdo dos blocos vizinhos de I^*_t, além do próprio bloco I^*_t (Figura 8).

Neste caso, se um bloco I'_t for alterado, a verificação da assinatura irá falhar em todos aqueles blocos que dependem de I'_t, além do próprio bloco I'_t. Porém, um



(a) 1 dependência por bloco (*zig-zag*). (b) 1 dependência por bloco (*raster*).



(c) 2 dependências por bloco. (d) 4 dependências por bloco.

Figura 8 – Uso da informação contextual. Para calcular a assinatura de um bloco I_t (mostrado em cinza), o conteúdo do bloco I_t e de seus blocos vizinhos são levados em conta. A técnica HBC utiliza 1 dependência por bloco, em ordem *zig-zag* ou *raster* (BARRETO, 2003).

número tão pequeno quanto possível de dependências é desejável para uma localização acurada da alteração na imagem. O ideal é que haja apenas uma dependência por bloco. Esta idéia pode ser implementada como:

$$H_t \equiv H\,(\,M\,,\,N\,,\,I^*_t\,,\,I^*_{(t\text{-}1)\bmod n}\,,\,t\,).$$

O índice do bloco t foi inserido para detectar a rotação bloco a bloco. Assim como no esquema de Wong, os tamanhos M e N da imagem são inseridos para detectar cortes na imagem. Note que se um bloco I'_t for alterado, o HBC1 irá reportar que o bloco $I'_{(t+1)\bmod n}$ é inválido (além do próprio bloco I'_t).

Usando HBC1, os ataques recortar-e-colar simples, de falsificação e de aniversário simples não podem mais ser executados, pois qualquer alteração em um bloco I'_t tem grande probabilidade de invalidar a assinatura do bloco seguinte $I'_{(t+1)\bmod n}$. Se o bloco seguinte $I'_{(t+1)\bmod n}$ for alterado, para corrigir este problema, existe grande probabilidade de invalidar a assinatura do bloco $I'_{(t+2)\bmod n}$ e assim por diante, de forma cíclica.

Ataque de transplante

O ataque de transplante (BARRETO; KIM; RIJMEN, 2000, 2001, 2002) é uma forma melhorada do ataque recortar-e-colar que pode ser utilizada para adulterar imagens marcadas pelo HBC1.

Suponha que X' e Y' são duas imagens marcadas pelo HBC1. Para facilitar a explanação, a dependência entre blocos será denotada por $X'_A \rightarrow X'_B$, indicando que o resumo do bloco X'_B depende do conteúdo do bloco X'_A (isto é, X^*_A). Suponha que as imagens X' e Y' possuam as seguintes seqüências de blocos:

$$\ldots \rightarrow X'_A \rightarrow X'_D \rightarrow X'_B \rightarrow X'_C \rightarrow \ldots$$

$$\ldots \rightarrow Y'_A \rightarrow Y'_E \rightarrow Y'_B \rightarrow Y'_C \rightarrow \ldots$$

onde $X^*_A = Y^*_A$, $X^*_B = Y^*_B$, $X^*_C = Y^*_C$, mas $X^*_D \neq Y^*_E$.

Então, o par de blocos (X'_D, X'_B) pode ser trocado com o par (Y'_E, Y'_B), sem ser detectado pelo esquema HBC1:

$$\ldots \rightarrow X'_A \rightarrow Y'_E \rightarrow Y'_B \rightarrow X'_C \rightarrow \ldots$$

$$\ldots \rightarrow Y'_A \rightarrow X'_D \rightarrow X'_B \rightarrow Y'_C \rightarrow \ldots$$

Em geral, imagens de documentos apresentam amplas áreas brancas, o que as torna muito suscetíveis a ataques de transplante. Por exemplo, se X'_A, X'_B, X'_C, Y'_A, Y'_B e Y'_C fossem blocos brancos sem ruído, o ataque teria sucesso facilmente.

Ataques semelhantes também poderiam ser executados com 2, 4 ou 8 dependências.

Ataque de aniversário melhorado

O esquema HBC1 não consegue resistir ao ataque de aniversário mais sofisticado descrito a seguir (BARRETO; KIM; RIJMEN, 2001, 2002).

Este ataque substitui dois blocos consecutivos X'_t e X'_{t+1} pelos blocos forjados J_t e J_{t+1} (nos índices, o "mod n" será omitido para simplificar a notação). Três resumos criptográficos são afetados por estas substituições: H_t (que depende de X'_t), H_{t+1} (que depende de X'_t e de X'_{t+1}) e H_{t+2} (que depende de X'_{t+1}).

Suponha que o banco de dados possui s blocos assinados e que o atacante prepara p variantes visualmente equivalentes para J_t. Então, provavelmente $P \cong ps/m$

colisões para H_t serão encontradas (NISHIMURA; SIBUYA, 1990), onde m é a quantidade de resumos distintos que podem ser gerados pela função de *hash*. De forma mais clara, podemos dizer que P pares (J_t^1, D_t^1), ..., (J_t^P, D_t^P) serão encontrados, onde J_t^1, ..., J_t^P são as variantes visualmente equivalentes de J_t e D_t^1, ..., D_t^P são os blocos do banco de dados, tais que o resumo de D_t^i é o mesmo de J_t^i:

$$H(M, N, D^{*i}_t, X^*_{t-1}, t) = H(M, N, J^{*i}_t, X^*_{t-1}, t) \text{ para } 1 \le i \le P.$$

Desta forma, a assinatura do bloco t permanecerá válida se X_t for substituído por qualquer bloco J^{*i}_t junto com a assinatura obtida dos LSBs de D_t^i. Porém, quase certamente esta substituição irá tornar inválida a assinatura do bloco t+1.

Assim, o atacante também deve preparar q variantes de J_{t+1}, gerando provavelmente $Q \cong qs/m$ colisões para H_{t+2}. Sejam (J_{t+1}^1, D_{t+2}^1), ..., (J_{t+1}^Q, D_{t+2}^Q) os pares que colidem, isto é:

$$H(M, N, X^*_{t+2}, J^{*j}_{t+1}, t+2) = H(M, N, X^*_{t+2}, D^{*j}_{t+2}, t+2) \text{ para } 1 \le j \le Q.$$

A assinatura do bloco t+2 permanece válida se X_{t+1} for substituída por quaisquer J^{*j}_{t+1} juntamente com a assinatura obtida dos LSBs de D_{t+2}^j. Mas esta substituição provavelmente irá invalidar a assinatura do bloco t+1.

Combinando todas as variantes de J_t e J_{t+1} que colidem irá gerar aproximadamente $(ps/m)(qs/m) = pqs^2/m^2$ pares (J_t^i, J_{t+1}^j) visualmente equivalentes a (J_t, J_{t+1}). Agora, o atacante deve encontrar uma colisão para H_{t+1}, isto é, deve achar um par variante (J_t^i, J_{t+1}^j) e um bloco do banco de dados D_{t+1} tais que:

$$H(M, N, J^{*j}_{t+1}, J^{*i}_t, t+1) = H(M, N, D^*_{t+1}, J^{*i}_t, t+1).$$

Desta forma, se X_t e X_{t+1} forem substituídos pelos blocos falsificados J^{*i}_t e J^{*j}_{t+1} e, ao mesmo tempo, as assinaturas dos blocos t, t+1 e t+2 forem substituídas pelas assinaturas obtidas dos LSBs de D_t^i, D_{t+1} e D_{t+2}^j, a adulteração passará despercebida pelo HBC1.

Para que a chance de sucesso neste tipo de ataque seja superior a 50%, é necessária uma quantidade maior de variantes para os blocos forjados. Como existem pqs^2/m^2 pares de blocos e s blocos no banco de dados, uma colisão para H_{t+1} irá ocorrer provavelmente quando (pqs^2/m^2)s ≈ m, isto é, quando $pq \approx (m/s)^3$. Portanto, se o banco de dados possuir $s = m^{1/2}$ assinaturas válidas, provavelmente dois blocos falsificados podem substituir dois blocos consecutivos válidos quando p

$\approx q \approx m^{3/4}$ blocos variantes, visualmente equivalentes a cada bloco falsificado, são preparados.

Hash Block Chaining versão 2 (HBC2)

O esquema HBC1 foi melhorado para resistir aos ataques de transplante e de aniversário melhorado. O novo esquema (BARRETO; KIM; RIJMEN, 2001, 2002) foi denominado *Hash Block Chaining*, versão 2 (HBC2).

O HBC2 faz uso de assinatura não-determinística para prevenir os ataques de transplante. Uma assinatura não-determinística não depende apenas da função de *hash*, mas também de algum parâmetro escolhido aleatoriamente. Este parâmetro (arbitrário e estatisticamente único) é conhecido como "sal". O esquema HBC2 pode ser definido por:

$$H_t \equiv H\,(\,M\,,\,N\,,\,I^*_t\,,\,I^*_{(t\text{-}1)\bmod n}\,,\,t\,,S_{t\text{-}1}\,)$$

onde $S_{t\text{-}1}$ é a assinatura não-determinística do bloco $I_{t\text{-}1}$ e $S_{\text{-}1} \equiv \varnothing$. Note que não é possível utilizar $S_{(t\text{-}1)\bmod n}$, porque quando o resumo H_0 estiver sendo calculado, a assinatura $S_{\text{-}1}$ ainda não será conhecida.

O ataque de aniversário melhorado é completamente ineficaz contra o HBC2, pois no HBC2 a assinatura de um bloco depende não somente do conteúdo do bloco vizinho, mas também da sua assinatura não determinística. O HBC2 é capaz de detectar se algum bloco foi modificado, rearranjado, apagado, inserido, ou transplantado de uma imagem legitimamente assinada. Além disso, esta técnica também permite localizar espacialmente as alterações.

2.5.4 **Wong-Memon**

Wong e Memon (2001) propuseram um esquema de marca d'água muito semelhante ao HBC2, embora as duas técnicas tenham sido desenvolvidas de maneira independente.

A diferença básica é que a técnica de Wong e Memon utiliza um identificador externo para tornar a assinatura não-determinística. Este identificador (*ID*) deve ser único (um número seqüencial, por exemplo) e de alguma forma armazenado fora da

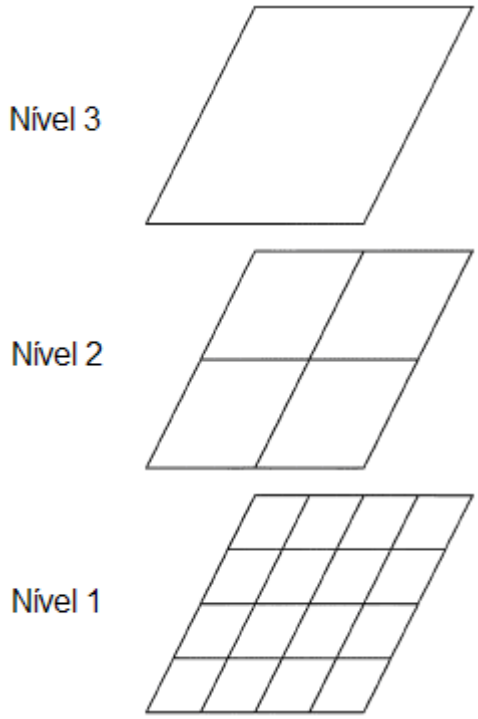


Figura 9 – Estrutura hierárquica (CELIK et al., 2002b).

imagem. O uso do identificador simplifica o algoritmo de marcação, mas trás o inconveniente do armazenamento externo. A função de *hash* de Wong e Memon pode ser descrita como:

$$H_t \equiv H\,(\,ID\,,\,M\,,\,N\,,\,I^*_t\,,\,t\,)\,.$$

Note que nesta técnica não é necessário alimentar a função *H* com o conteúdo dos blocos vizinhos.

2.5.5 **Estrutura hierárquica**

Como foi visto anteriormente, uma das maneiras de evitar ataques é aumentar o tamanho do resumo gerado pela função de *hash*. Porém, esta solução compromete o poder de localização das alterações, pois blocos maiores têm que ser utilizados para armazenar uma quantidade maior de bits.

CELIK et al. (2002b) propuseram uma modificação no esquema de Wong, onde a imagem é dividida em uma estrutura hierárquica de vários níveis e onde cada nível é dividido em blocos não sobrepostos (Figura 9). Neste esquema, cada bloco de cada

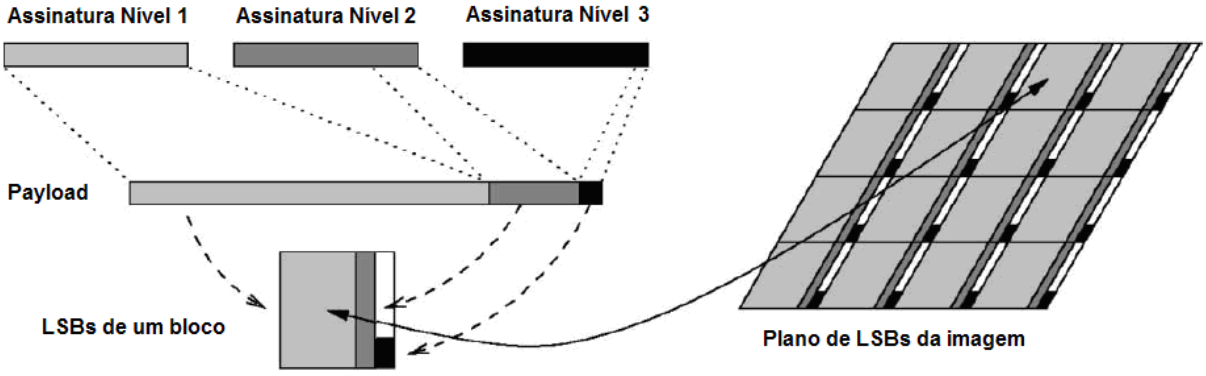


Figura 10 – Inserção das assinaturas em estrutura hierárquica (CELIK et al., 2002b).

nível recebe uma assinatura individual. Desta forma, as assinaturas dos blocos de nível mais baixo (blocos menores) servem para detectar com precisão a localização das alterações e as assinaturas dos blocos de nível mais alto (blocos maiores) servem para impedir ou dificultar a ocorrência de ataques. No nível mais alto, por exemplo, a assinatura é computada utilizando a imagem completa, o que impede completamente a ocorrência de um ataque de falsificação.

Todas as assinaturas (de todos os blocos pertencentes a todos os níveis) são armazenadas utilizando os LSBs de todos os pixels da imagem (Plano de LSBs). Para que não haja colisão entre as assinaturas, um esquema especial de armazenamento foi utilizado, como pode ser visto na Figura 10. Nesta figura, a assinatura do bloco de nível mais alto (Nível 3, em cor preta), por exemplo, fica espalhada por toda a imagem, enquanto a assinatura de um bloco de nível mais baixo (Nível 1, em cinza claro) é inserida completamente nos LSBs referentes ao seu bloco. Note que o tamanho do bloco do Nível 1 deve ser suficiente para conter a sua assinatura e parte das assinaturas dos níveis superiores.

2.6 AWTS SEGURAS PARA IMAGENS BINÁRIAS E ATAQUE DE PARIDADE

Muitas técnicas esteganográficas para imagens binárias podem ser transformadas em AWTs simplesmente dividindo a imagem hospedeira Z em duas regiões: Z_1 onde a AS será armazenada e Z_2 que será utilizada para computar a AS.

Esta idéia foi usada para projetar algumas técnicas que serão vistas mais adiante como a AWST (*Authentication Watermarking by Self Toggling* - marca d'água de autenticação por auto-inversão) utilizada para imagens meio-tom com pontos dispersos (KIM; AFIF, 2004) e AWTR (*Authentication Watermarking by Template Ranking* - marca d'água de autenticação pelo ranqueamento de modelos) para imagens binárias em geral (KIM; DE QUEIROZ, 2004a, 2004b).

Contudo, alguns cuidados devem ser tomados quando uma técnica esteganográfica é transformada em uma AWT, porque apesar da região Z_2 ser bem protegida (com a segurança garantida pela teoria da criptografia) a região Z_1 não é segura.

É chamada de AWT segura, um esquema que possui duas propriedades:

1) sua segurança não está no segredo do algoritmo, mas no segredo da chave;
2) é possível detectar qualquer alteração na imagem (acidental ou maliciosa), mesmo que seja um único pixel.

A propriedade (1) pode ser satisfeita pelo uso da criptografia.

A propriedade (2) pode ser satisfeita fazendo-se com que o comprimento da região Z_1 seja exatamente igual ao comprimento da AS adotada. Caso contrário, um invasor malicioso pode introduzir alterações indesejáveis não detectáveis usando uma técnica conhecida como ataque de paridade (KIM; DE QUEIROZ, 2004a, 2004b) que será descrita a seguir.

Para ilustrar este conceito, considere uma técnica esteganográfica baseada em componentes que insere um bit de informação em cada componente conexo, forçando-o a ter uma quantidade par ou ímpar de pixels pretos para representar os valores 0 ou 1, respectivamente. Um componente conexo pode ser forçado a ter a paridade desejada invertendo-se um de seus pixels de borda. Este esquema pode ser transformado em uma AWT dividindo-se a imagem em duas regiões: Z_1, composta pelos componentes conexos que carregarão a informação e Z_2, composta pelo restante da imagem. Z_2 será utilizada para computar a AS que será inserida em Z_1. Desta forma, um invasor malicioso pode alterar arbitrariamente a região Z_1 sem ser notado pela AWT, desde que as paridades de todos os componentes desta região permaneçam inalteradas. Por exemplo, um caractere "a" na região Z_1 pode ser transformado em um "e" (ou qualquer outro caractere formado por um único componente conexo) desde que sua paridade não seja alterada. Esta técnica de

adulteração é conhecida como ataque de paridade. Note que, neste exemplo, o tamanho da região Z_1 é muito maior do que o comprimento da AS. Por outro lado, se a quantidade de pixels de Z_1 é exatamente igual ao comprimento da AS adotada, qualquer alteração em Z_2 é detectada porque isto altera a AS da imagem marcada, e qualquer alteração em Z_1 também é detectada porque isto altera a AS armazenada, tornando impossível o ataque de paridade.

É importante ressaltar que, como pôde ser visto no exemplo anterior, não há uma divisão física entre as regiões Z_1 e Z_2. O elementos (pixels, componentes conexos) da região Z_1, em geral, estão espalhados por toda a imagem, intercalados com elementos da região Z_2. A sobreposição das regiões Z_1 e Z_2 resulta na imagem completa Z.

2.7 TÉCNICAS EXISTENTES PARA IMAGENS BINÁRIAS

Na literatura, existem muitas técnicas esteganográficas para imagens binárias como, por exemplo, Wu; Tang e Liu (2000), Fu e Au (2000, 2001, 2002a), Chen; Pan e Tseng (2000), Pan; Chen e Tseng (2000), Tseng; Chen e Pan (2002) e Pei e Guo (2003). Também existem várias AWTs seguras para imagens binárias como, por exemplo, Kim e Afif (2003, 2004), Kim e de Queiroz (2004a, 2004b) e Kim (2005).

Existem três maneiras básicas para inserir uma seqüência de bits em uma imagem binária: alterar os valores de pixels individuais, mudar as características de um grupo de pixels ou mudar as características de blocos da imagem (KIM, 2003, 2004):

1) Baseada em pixels (*pixel-wise*) – Mudança das cores de pixels individuais (geralmente escolhidos de maneira pseudo-aleatória) (CHUN; HA, 2003; FU; AU, 2000, 2002; TSENG; CHEN; PAN, 2002). Este enfoque é apropriado para imagens meio-tom com pontos dispersos (*dispersed-dot halftone images*). Porém, ruídos do tipo sal-e-pimenta (pontos pretos isolados em áreas brancas ou pontos brancos isolados em áreas pretas) podem aparecer quando este método é aplicado para outros tipos de imagem binária.
2) Baseada em componentes (*component-wise*) – Mudança das características de um grupo de pixels como, por exemplo, a posição ou a quantidade de

pixels pretos de um componente conexo (MAXEMCHUK; LOW, 1997). Infelizmente, o sucesso deste método depende do tipo de imagem hospedeira (sendo ideal para textos) e a quantidade de dados que pode ser inserida é limitada.

3) Baseada em blocos (*block-wise*) – Divisão da imagem hospedeira em blocos e mudança de algumas características de cada bloco. Por exemplo, pode-se dividir uma imagem binária em blocos (por exemplo, 8×8) e inserir um bit de informação em cada bloco forçando o número de pixels brancos do bloco a ser par ou ímpar. Se o número de pixels brancos do bloco for par, convenciona-se que o bit zero está embutido naquele bloco (WU; TANG; LIU, 2000). Se for ímpar, o bit um está embutido. Se um bloco já representar o bit que se deseja inserir, não há nada a fazer. Caso contrário, procura-se pelo pixel que causará a menor degradação visual segundo algum critério perceptual e troca-se o seu valor. Esta técnica pode ser aplicada a qualquer tipo de imagem, porém não permite a inserção de mais de um bit em cada bloco. Alguns trabalhos (BAHARAV; SHAKED, 1999; HEL-OR, 2001; PEI; GUO, 2003) sugerem a utilização de duas diferentes matrizes de peso (Floyd-Steinberg, Jarvis ou Stucki) para transformar uma imagem em meio-tom, de maneira que a matriz utilizada em cada bloco pode ser determinada no futuro analisando-se as propriedades estatísticas dos mesmos (usando transformada de Fourier). Outros trabalhos (TSENG; CHEN; PAN, 2002; WU; LIU, 2004) sugerem uma leve modificação no conteúdo do bloco para que os mesmos possam armazenar a seqüência de bits desejada.

Para as imagens meio-tom, pode ser citada ainda uma quarta abordagem: uma imagem é escondida em duas imagens meio-tom de forma que ela torna-se visível quando as duas são sobrepostas (WANG, 1998; FU; AU, 2001; PEI; GUO, 2003].

De acordo com o paradigma criptográfico amplamente aceito, a segurança de uma marca de autenticação deve estar apoiada somente no segredo da chave. O fato de que uma imagem foi marcada, assim como o algoritmo utilizado para marcar a imagem devem poder se tornar públicos sem comprometer a segurança do esquema.

Nesta seção, serão analisadas três marcas de autenticação para imagens binárias que satisfazem este requerimento. Estas técnicas foram denominadas AWST

(*Authentication Watermarking by Self Toggling* - marca d'água de autenticação por auto-inversão), AWTR (*Authentication Watermarking by Template Ranking* - marca d'água de autenticação pelo ranqueamento de modelos) e AWTC (*Authentication Watermarking by Template ranking with symmetrical Central pixels* - marca d'água de autenticação pelo ranqueamento de modelos com pixels centrais simétricos).

A marca AWST (KIM; AFIF, 2003, 2004) é apropriada para as imagens meio-tom pontos dispersos. As marcas AWTR (KIM; DE QUEIROZ, 2004a, 2004b) e AWTC (KIM, 2005) são apropriadas para imagens binárias em geral. As marcas AWTR e AWTC não são adequadas para as imagens meio-tom pontos dispersos, mas podem ser utilizadas em imagens meio-tom pontos aglutinados. Assim, estas duas técnicas podem ser usadas de forma complementar à AWST para proteger qualquer imagem binária. As marcas AWST, AWTR e AWTC podem ser usadas com criptografias de chave secreta ou pública. A marca AWTR de chave pública necessita de cuidados especiais para evitar ataques de paridade.

2.7.1 ***Data Hiding by Self Toggling*** **(DHST)**

DHST (*Data Hiding by Self Toggling* - embutimento de dados por auto-inversão) (FU; AU, 2000, 2002) é uma técnica esteganográfica baseada em pixels (*pixel-wise*) especialmente interessante devido à sua simplicidade.

Algoritmo 5 - Inserção da DHST

1) Gerar um conjunto de posições pseudo-aleatórias não repetidas dentro da imagem. Para isto, basta utilizar um gerador de número pseudo-aleatórios com semente conhecida.
2) Embutir um bit de informação em cada posição, forçando a cor do pixel a ser branca (0) ou preta (1).

Existe a probabilidade de 50% de um pixel escolhido na imagem original ter o mesmo valor que se deseja inserir. Neste caso, nenhuma mudança é necessária. Com a probabilidade de 50%, o pixel tem o valor oposto ao desejado, e o pixel deve ser alterado.

É importante garantir que não haja repetições de posições pseudo-aleatórias, pois neste caso o algoritmo tentaria inserir dois ou mais bits de informação num único pixel, o que evidentemente levaria a erro.

Algoritmo 6 - Extração da DHST

1) Gerar novamente o mesmo conjunto de posições pseudo-aleatórias não repetidas dentro da imagem, da mesma forma que no algoritmo de inserção.
2) Extrair de cada posição um bit de informação de acordo com a cor do pixel.

Esta técnica foi originalmente concebida para embutir informações em imagens meio-tom pontos dispersos. Evidentemente, DHST também pode ser usada para qualquer imagem binária. Porém, neste caso, um ruído do tipo sal-e-pimenta poderá ser notado.

Em imagens meio-tom pontos dispersos, como DHST muda os valores de pixels selecionados de maneira pseudo-aleatória, a intensidade local média pode ser severamente afetada. Uma solução para este problema, apresentada por Fu e Au (2000), é a técnica DHPT (*Data Hiding by Pair-Toggling* - embutimento de dados por inversão aos pares). Nesta técnica, a mudança de valor de um pixel deve ser acompanhada, sempre que possível, pela mudança complementar de um de seus vizinhos. Por exemplo, se um pixel mestre é forçado mudar de 0 para 1, então os pixels vizinhos (na vizinhança 3×3) com valor 1 são identificados e um deles é escolhido aleatoriamente para mudar o seu valor para 0. Este pixel é chamado de “pixel escravo”. No mesmo artigo, Fu e Au apresentam também a técnica DHSPT (*Data Hiding by Smart Pair Toggling* - embutimento de dados por inversão inteligente aos pares), que consiste basicamente em estabelecer algumas regras para escolher o pixel escravo, entre os candidatos, de forma a perturbar o menos possível a qualidade visual da imagem meio-tom.

2.7.2 ***Authentication Watermarking by Self Toggling*** **(AWST)**

A técnica AWST (*Authentication Watermarking by Self Toggling* - marca d'água de autenticação por auto-inversão) (KIM; AFIF, 2003, 2004) utiliza como base a técnica DHST para inserir uma marca d'água de autenticação em uma imagem binária.

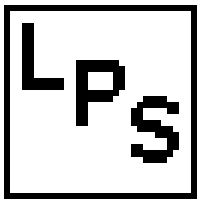
Algoritmo 7 - Inserção da AWST

1) Seja *B* a imagem binária a ser marcada e seja *A* uma imagem logotipo binária a ser inserida em *B*.
2) Usando um gerador de números pseudo-aleatórios com uma semente conhecida, gerar um conjunto de posições pseudo-aleatórias não-repetidas *L* dentro da imagem *B*.
3) Zerar todos os pixels de *B* que pertencem a *L*, obtendo a imagem *B**.
4) Calcular o resumo criptográfico da imagem *B**: $H = H(B^*)$.
5) Calcular o ou-exclusivo de *H* com *A*, obtendo o resumo marcado $H\hat{}$.
6) Criptografar $H\hat{}$ com a chave secreta (criptografia simétrica) ou privada (criptografia assimétrica), gerando a AS *S*.
7) Inserir *S* no conjunto de pixels *L*, gerando a imagem marcada *B'*.

Algoritmo 8 - Verificação da AWST

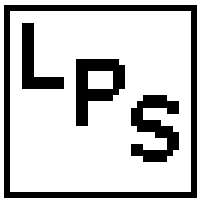
1) Seja *X'* uma imagem marcada pela AWST. Usando o mesmo gerador de números pseudo-aleatórios, gerar novamente o mesmo conjunto de posições pseudo-aleatórias não-repetidas *L* onde a marca foi inserida.
2) Extrair a AS inserida em *X'* lendo os pixels de *L* e decriptografando-os com a chave secreta (criptografia simétrica) ou pública (criptografia assimétrica), obtendo o resumo marcado *D*.
3) Seja *X** a imagem obtida de *X'* zerando todos os pixels de *L*. Usando a mesma função de *hash* da inserção, calcular o resumo criptográfico $H = H(X^*)$.
4) Calcular o ou-exclusivo de *H* com *D*, obtendo a imagem logotipo binária de checagem *C*.

(a) Parte da imagem meio-tom *B* (512×512 pixels) a ser marcada.



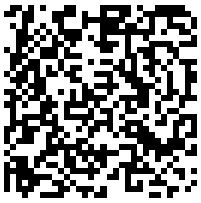
(b) Imagem logotipo binária *A* (32×32 pixels) a ser inserida em *B*.

(c) Parte da imagem *B'* marcada com 1024 bits.



(d) Imagem logotipo *C* extraída de *B'*.

(e) Parte da imagem modificada *X'*.



(f) Imagem logotipo *C* extraída de *X'*.

Figura 11 – Ilustração da AWST de chave pública (KIM, 2004). A imagem logo *A* (b) foi inserida na imagem *B* (a) usando uma cifra de chave pública. A figura (c) mostra a imagem marcada. Executando o algoritmo de extração da marca, a figura (d) foi obtida. Se a imagem marcada for modificada, mesmo que seja ligeiramente (e), uma imagem completamente aleatória é extraída (f)).

5) Se *C* e *A* são iguais, a marca está verificada. Caso contrário, a imagem marcada *X'* foi alterada.

Um exemplo de aplicação da técnica AWST pode ser visto na Figura 11. Para proteger a imagem binária *B* (Figura 11a), a imagem logotipo *A* (Figura 11b) foi inserida usando o algoritmo AWST. A imagem *B'* (Figura 11c) é a imagem marcada onde 1024 bits foram inseridos. Isto é suficiente para embutir uma assinatura digital RSA (SCHNEIER, 1996).

Quando o algoritmo de verificação é executado, obtém-se a imagem de checagem *C* (Figura 11d) exatamente igual à imagem logotipo *A*. Mesmo que um único pixel de *B'* seja alterado, a imagem extraída será completamente ruidosa (Figura 11f).

A qualidade visual de uma imagem meio-tom pontos dispersos marcada com AWST pode ser melhorada usando as técnicas para embutir dados DHPT ou DHSPT (FU; AU, 2000, 2002), em vez de DHST. Estes melhoramentos procuram manter inalterada a intensidade média local. Nas posições pseudo-aleatórias selecionadas, a alteração de um pixel é acompanhada pela modificação complementar de um pixel vizinho.

Entretanto, para implementar um desses esquemas, nenhum pixel vizinho dos pixels pseudo-aleatórios pode alimentar a função de *hash*. Conseqüentemente, esses pixels permanecerão desprotegidos, isto é, se uma alteração ocorrer num pixel vizinho de uma posição pseudo-aleatória, essa alteração não será detectada pela marca AWST.

2.7.3 ***Data Hiding by Template Ranking*** **(DHTR)**

Nesta subseção, será descrita de maneira sucinta uma técnica esteganográfica não reversível para imagens binárias chamada DHTR (*Data Hiding by Template Ranking* - esteganografia pelo ranqueamento de modelos) (WU; TANG; LIU, 2000; DE QUEIROZ; FLECKENSTEIN, 1999; WU; LIU, 2004), que servirá de base para a técnica DHTRJ (*Data Hiding by Template Ranking for JBIG2* - esteganografia pelo ranqueamento de modelos para JBIG2) proposta no Capítulo 3 deste trabalho.

Esta técnica pode ser aplicada à maioria das imagens binárias, pode ocultar uma quantidade moderada de dados e gera imagens marcadas de excelente qualidade visual, pois a escolha dos pixels a serem modificados é feita com base na nota de impacto visual dos mesmos.

Algoritmo 9 - Inserção da DHTR

1) Dividir a imagem em pequenos blocos (por exemplo, 8×8).
2) Analisar a vizinhança (geralmente 3×3) de cada pixel, para verificar a sua significância visual, através do cálculo da “nota de impacto visual” (VIS - *Visual Impact Score*) do pixel. A Figura 12 mostra um exemplo de ranqueamento de modelos (*template ranking*) com todas as combinações

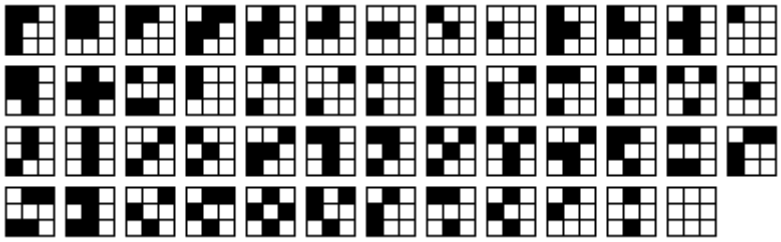


Figura 12 – Ranqueamento de modelos 3×3 em ordem crescente de nota de impacto visual. Espelhos, rotações e reversos de cada padrão possuem a mesma nota (KIM; DE QUEIROZ, 2004b).

possíveis em ordem crescente de significância visual da esquerda para a direita, de cima para baixo. Espelhos, rotações e reversos de cada padrão possuem a mesma nota. Parece não existir um único ranqueamento que seja o melhor de todos, pois o ranqueamento que proporciona o visual mais agradável depende da natureza da imagem a ser marcada. Outros possíveis ranqueamentos podem ser encontrados em Wu e Liu (2004) e Fujii et al. (2003).

3) Inserir um bit em cada bloco forçando-o a ter um número de pixels pretos que seja par ou impar, para inserir os bits 0 ou 1, respectivamente. Se o bloco já possui a paridade desejada, nenhuma mudança é feita. Caso contrário, o pixel de menor visibilidade (menor nota de impacto visual) é alterado.

4) Como diferentes blocos possuem diferentes quantidades de pixels de baixa visibilidade, é sugerido o embaralhamento da imagem antes da inserção da informação. Isto evita ruídos do tipo sal-e-pimenta em blocos que possuem apenas pixels de alta visibilidade (blocos totalmente brancos ou pretos, por exemplo).

Algoritmo 10 - Extração da DHTR

1) Dividir a imagem em blocos, do mesmo modo utilizado no processo de inserção.

2) Verificar a paridade de cada bloco, extraindo de cada um deles um bit de informação.

2.7.4 ***Authentication Watermarking by Template Ranking*** **(AWTR)**

A técnica DHTR descrita na subseção anterior por ser facilmente empregada na criação de uma AWT criptograficamente segura para imagens binárias. Tal idéia foi sugerida por Wu; Tang e Liu (2000) e implementada por Kim e de Queiroz (2004a, 2004b). Esta técnica, nomeada AWTR (*Authentication Watermarking by Template Ranking* - marca d'água de autenticação pelo ranqueamento de modelos), pode ser aplicada a todos os tipos de imagens binárias, apresentando excelente qualidade visual nas imagens marcadas, exceto em imagens meio-tom de pontos dispersos onde o conceito de pixel de baixa visibilidade não se aplica.

Como foi visto anteriormente, a idéia de computar a AS de toda a imagem e inseri-la na própria imagem é falha, pois a inserção altera o resumo criptográfico da imagem. A solução é dividir a imagem em duas regiões: a primeira região (pequena) é utilizada para o armazenamento da AS; a segunda região (grande) é utilizada para o cômputo da AS.

Algoritmo 11 - Inserção da AWTR

1) Seja dada uma imagem binária Z. Usando um gerador de números pseudo-aleatórios com uma determinada semente, construir uma estrutura de dados auxiliar, que pode ser chamada de vetor de embaralhamento V, de maneira que a imagem Z possa ser vista como uma seqüência embaralhada de pixels P. Na versão de chave secreta desta técnica, a chave pode ser usada como semente para a geração dos números pseudo-aleatórios. Na versão de chave pública, a semente deve ser pública.
2) Seja n o comprimento da AS adotada e m a quantidade de pixels de cada bloco. Dividir a seqüência P em duas regiões:
   - Região P_1: composta pelos primeiros $n \times m$ pixels da seqüência embaralhada P, onde a AS será armazenada. Esta região será subdividida em n blocos com m pixels. Em cada bloco, um bit de AS será inserido.
   - Região P_2: composta pelo restante da seqüência embaralhada P. Esta região será utilizada para o cômputo da AS.

3) Usando uma função de *hash* criptograficamente segura, computar o resumo criptográfico da região P_2: $H = H(P_2)$.

4) Criptografar o resumo H usando a chave secreta ou a chave privada, obtendo a AS: $S = K(H)$.

5) Inserir S na região P_1, usando DHTR, obtendo a imagem marcada Z'.

Algoritmo 12 - Verificação da AWTR

1) Dada uma imagem binária Z' marcada pela AWTR, computar o mesmo vetor de embaralhamento V usado na inserção, obtendo a seqüência embaralhada de pixels P'. Note que na versão de chave secreta, a chave também é utilizada como semente na geração de números pseudo-aleatórios e conseqüentemente apenas o proprietário da chave pode reconstruir V. Porém, na versão de chave pública, a semente é pública e conseqüentemente o vetor V também pode ser conhecido publicamente.

2) Dividir P' em duas regiões P'_1 e P'_2, como no processo de inserção.

3) Computar o resumo criptográfico H de P'_2.

4) Extrair a AS armazenada em P'_1 e decriptografá-la usando a chave secreta ou a chave pública, obtendo o resumo de verificação D.

5) Se $D = H$ a marca está verificada. Caso contrário, a imagem Z' foi modificada ou foi utilizada uma chave incorreta.

Nesta técnica, qualquer alteração visual significante (até mesmo um único pixel) na região P'_2 da imagem marcada pode ser detectada. A probabilidade de não detecção é de apenas 2^{-n} a qual pode ser negligenciada (n é o comprimento da AS).

Infelizmente, qualquer alteração que mantenha a paridade dos blocos na região P'_1 não pode ser detectada. É o problema do ataque de paridade já discutido anteriormente.

De fato, o cenário descrito acima só se aplica à versão de chave pública do algoritmo, onde a localização das regiões 1 e 2, bem como a subdivisão da região 1 em blocos são publicamente conhecidas. Na versão de chave secreta, o problema de ataques de paridade não existe.

Apesar da versão de chave secreta ser teoricamente segura contra este tipo de ataque, os invasores possuem várias maneiras de obter pistas sobre a localização das regiões e dos blocos. Por exemplo, se um invasor tiver acesso a um banco de dados de imagens binárias marcadas, com suas respectivas imagens originais, todas do mesmo tamanho e todas marcadas com a mesma chave, então ele pode concluir que todos os pixels que diferem nas imagens marcadas e originais pertencem à região 1.

Para minimizar a possibilidade de um ataque de paridade, Kim e de Queiroz (2004a, 2004b) sugerem uma modificação no passo 5 do algoritmo de inserção:

Modificação no passo 5 do algoritmo de inserção da AWTR

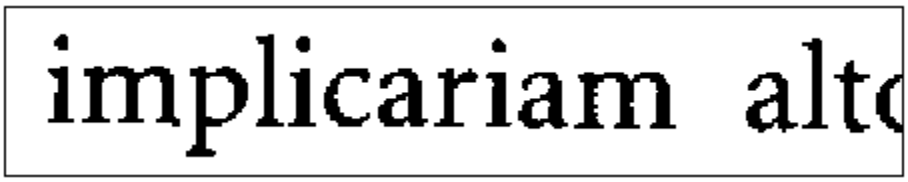
5) Inserir S na região P_1, usando o algoritmo a seguir, obtendo a imagem marcada Z':

```
para i = 0 até n-1
{
    inserir o bit i de S no bloco i, forçando-o a ter um número par ou ímpar
        de pixels pretos;
    computar a nova AS S, alimentando a função de hash com o conteúdo
        do bloco i e criptografando-a: S = K(H(S, pixels do bloco i));
}
```

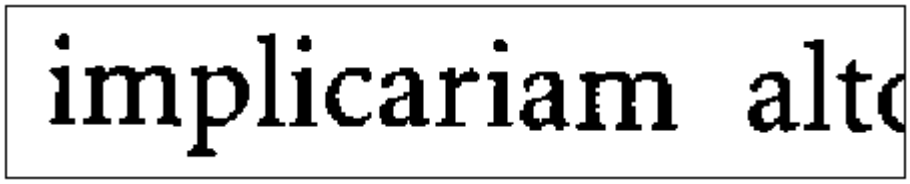
Desta forma, o último bloco ($n-1$) ainda pode sofrer um ataque de paridade. Porém, se o bloco $n-2$ for modificado sem mudança de paridade, existe 50% de chance da modificação ser detectada. Se o bloco $n-3$ for modificado sem mudança de paridade, existe 75% de chance da modificação ser detectada. Se o bloco 0 for modificado sem mudança de paridade, existe a probabilidade de $1-2^{-(n-1)}$ da mudança ser detectada. Esta modificação, com certeza, torna muito mais difícil a ocorrência de um ataque de paridade.

A Figura 13 ilustra uma aplicação da técnica AWTR.

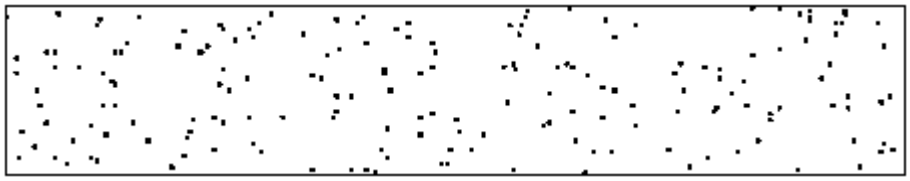
Em (KIM; DE QUEIROZ, 2004b) também é proposta uma evolução deste algoritmo que permite a localização das áreas da imagem que foram adulteradas. Esta técnica foi chamada de AWTRAL (*Authentication Watermarking by Template Ranking with Alteration Locating* - marca d'água de autenticação pelo ranqueamento de modelos com localização de alteração) e será descrita a seguir.



(a) Imagem original.



(b) Imagem marcada.



(c) Divisão de regiões.

Figura 13 – Exemplo de aplicação da técnica AWTR. (a) Parte de uma página de revista escaneada a 300 dpi. (b) Parte da imagem marcada com 1024 bits (suficiente para uma assinatura RSA). (c) Divisão de regiões: pixels pretos pertencem à região P_1 e pixels brancos pertencem à região P_2 (KIM; DE QUEIROZ, 2004a).

2.7.5 ***Authentication Watermarking by Template Ranking with Alteration Locating*** **(AWTRAL)**

A técnica AWTR pode ser transformada em uma marca d'água de autenticação capaz de localizar espacialmente as regiões que foram alteradas. Esta técnica (KIM; DE QUEIROZ, 2004b) foi chamada de AWTRAL (*Authentication Watermarking by Template Ranking with Alteration Locating*) e utiliza uma estrutura hierárquica (CELIK et al., 2002b) de duas camadas para inserir assinaturas que permitam a localização espacial das alterações.

A idéia da AWTRAL é dividir a imagem em blocos (primeira camada) e marcar cada bloco de maneira independente usando AWTR. Depois, marcar também a imagem completa (segunda camada) utilizando também AWTR.

As duas camadas podem utilizar tanto criptografia de chave publica quanto criptografia de chave secreta. Se ambas as camadas utilizarem chave pública, as alterações poderão ser detectadas e localizadas publicamente.

Nesta técnica, muitos bits têm que ser inseridos em cada bloco, o que reduz a resolução da localização de alterações, pois blocos maiores têm que ser utilizados. Uma possível solução para este problema é utilizar um MAC com uma chave secreta curta na primeira camada para localizar as alterações e uma DS com uma chave pública longa na segunda camada para autenticar a imagem.

Neste esquema, qualquer pessoa pode verificar se a imagem é autêntica usando a chave pública. Se uma imagem for adulterada, o proprietário da chave secreta pode localizar espacialmente as alterações para ajudar a descobrir as intenções do invasor.

Algoritmo 13 - Inserção da AWTRAL

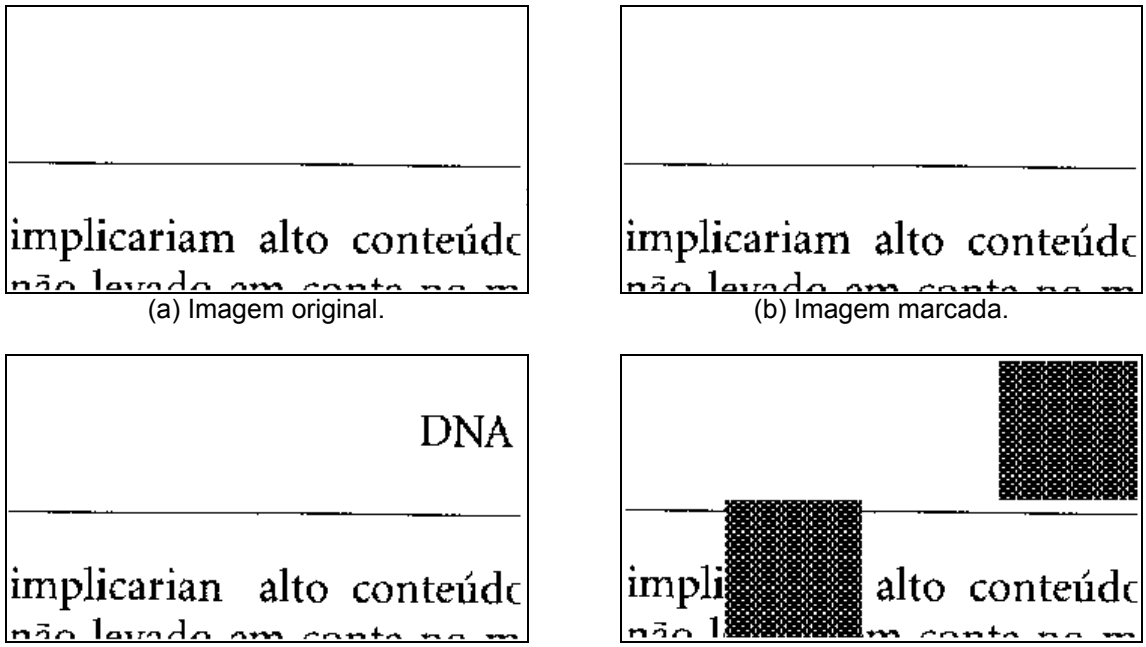
1) Dada uma imagem binária Z a ser marcada, embaralhar os pixels de Z de maneira pseudo-aleatória, obtendo a seqüência de pixels P.
2) Seja n_2 o comprimento da DS a ser utilizada na segunda camada e m a quantidade de pixels de cada bloco. Dividir a seqüência P em duas regiões:
   - Região P_A: composta pelos primeiros $n_2 \times m$ pixels da seqüência embaralhada P, onde a DS da segunda camada será armazenada. Esta região será subdividida em n_2 blocos com m pixels. Em cada bloco, um bit de DS será inserido.
   - Região P_{BC}: composta pelo restante da seqüência embaralhada P. Esta região será utilizada para o cômputo da DS. Esta região será dividida posteriormente em P_B e P_C.
3) Dividir a imagem original Z em blocos (por exemplo, 128×128 pixels): Z_1, Z_2, ..., Z_I.
4) Embaralhar os pixels de cada bloco Z_i de maneira pseudo-aleatória, obtendo as seqüências de pixels P_i. Dividir cada seqüência P_i em três regiões não sobrepostas:

- Região P_{iA}: composta pelos pixels do bloco P_i que pertencem à região P_A determinada no passo 2.
- Região P_{iB}: com $n_1 \times m$ pixels onde a MAC da primeira camada será armazenada (n_1 é o tamanho da MAC adotada e m é o tamanho do bloco).
- Região P_{iC}: composta pelo restante de P_i e utilizada para o cômputo da MAC.

5) Para cada bloco P_i, computar o resumo criptográfico da região P_{iC} e criptografar com a chave secreta, obtendo a MAC S_i. Inserir S_i na região P_{iB} usando DHTR.
6) Computar o resumo criptográfico da região P_{BC} e criptografar com a chave privada, obtendo a DS S. Inserir S na região P_A usando DHTR, obtendo a imagem marcada Z'.

Algoritmo 14 - Verificação da AWTRAL

1) Dada uma imagem binária Z' marcada pela AWTRAL, embaralhar os pixels de Z' de maneira pseudo-aleatória, obtendo a seqüência de pixels P'.
2) Dividir a seqüência P' em duas regiões, P'_A e P'_{BC}, como na inserção.
3) Computar o resumo criptográfico H de P'_{BC}.
4) Extrair a DS S armazenada em P'_A, decriptografar usando a chave pública, obtendo o resumo de verificação D.
5) Se $D = H$, a marca está verificada e o algoritmo de verificação pode ser encerrado. Caso contrário, a imagem foi adulterada.
6) Dividir a imagem original Z' em blocos $Z'_1, Z'_2, ..., Z'_l$, como na inserção.
7) Embaralhar os pixels de cada bloco Z'_i de maneira pseudo-aleatória, obtendo as seqüências de pixels P'_i. Dividir cada seqüência P'_i em três regiões não sobrepostas: P'_{iA}, P'_{iB} e P'_{iC}, como na inserção.
8) Para cada seqüência P'_i, computar o resumo H_i da região P'_{iC}. Extrair a MAC S_i armazenada na região P'_{iB} e decriptografá-la, obtendo o resumo de verificação D_i. Se $D_i = H_i$, a marca do bloco está verificada. Caso contrário, o bloco foi modificado.



(a) Imagem original. (b) Imagem marcada.

(c) Imagem adulterada. (d) Localização das alterações.

Figura 14 – Exemplo de aplicação da técnica AWTRAL. (a) Página de revista escaneada a 300 dpi. (b) Imagem marcada. (c) Imagem adulterada: letra “m” foi transformada em “n” e a palavra “DNA” foi adicionada. (d) Localização das alterações. (KIM; DE QUEIROZ, 2004b).

Os autores fazem três comentários interessantes sobre esta técnica:

1) Se uma linha ou coluna da imagem for removida, esta técnica perde a capacidade de localizar espacialmente as alterações. A sugestão para resolver este problema é utilizar uma “janela deslizante”, do mesmo tamanho de um bloco da primeira camada, para detectar todos os possíveis blocos Z_1, Z_2, ..., Z_l dentro de Z'.

2) Blocos que possuem quase a totalidade de pixels da mesma cor, não devem ser marcados ou verificados. Esses blocos não possuem pixels de baixa visibilidade e ruídos do tipo sal-e-pimenta podem aparecer. Qualquer modificação nestes blocos pode ser detectada visualmente.

3) O comprimento da MAC da primeira camada pode ser inferior aos usuais 128 bits, porque AWTRAL não pode ser subvertida pelo ataque de aniversário simples, pois a DS da segunda camada irá detectar a alteração. Porém, AWTRAL pode ser subvertida pelo ataque de aniversário melhorado, por isso o comprimento da MAC deve ser superior a 64 bits. Os autores sugerem MAC de 96 bits.

A Figura 14 mostra um exemplo de aplicação da técnica AWTRAL.

Figura 15 – Ranqueamento de modelos 3×3 com pixels centrais simétricos em ordem crescente de nota de impacto visual. Pixels hachurados podem ser pretos ou brancos (note que todos os pixels centrais são hachurados). A nota de um determinado padrão é aquela que combina com o modelo de menor nota. Espelhos, rotações e reversos de cada padrão possuem a mesma nota (KIM, 2005).

2.7.6 ***Data Hiding by Template ranking with symmetrical Central pixels*** **(DHTC)**

Nesta subseção, será descrita brevemente uma técnica esteganográfica não reversível para imagens binárias chamada DHTC (*Data Hiding by Template ranking with symmetrical Central pixels* - esteganografia pelo ranqueamento de modelos com pixels centrais simétricos) (KIM, 2005), que servirá de base para várias das técnicas propostas nos Capítulos 3 e 4 deste trabalho.

Algoritmo 15 - Inserção da DHTC

1) Dividir a imagem binária original *Z* em uma seqüência *v* de blocos não sobrepostos (por exemplo, 3×3). Nesta seqüência, apenas o pixel central de cada bloco poderá ter sua cor modificada para armazenar um bit de informação.

2) Ordenar a seqüência *v* de maneira crescente utilizando a nota de impacto visual como chave primária e um número pseudo-aleatório sem repetição como chave secundária. A chave primária classifica cada bloco de acordo com a “visibilidade” de seu pixel central (menor visibilidade, menor nota). A Figura 15 enumera todos os possíveis modelos 3×3, listados em ordem crescente de visibilidade de seus pixels centrais. Para assegurar a viabilidade de reconstrução de *v* no processo de extração, dois modelos que diferem apenas pela cor do pixel central devem ter a mesma nota. De acordo com o tipo de imagem a ser marcada, este ranqueamento de visibilidade pode ser modificado ou modelos maiores podem ser utilizados para minimizar alguma distorção perceptual específica. A chave secundária impede que a informação seja inserida apenas na parte superior da imagem.

3) Os pixels centrais dos *n* primeiros blocos da seqüência ordenada *v* são os pixels que levarão a informação embutida (DBPs – *Data Bearing Pixels*). Embutir *n* bits de informação modificando (se necessário) os DBPs. A cor do pixel central (preto ou branco) deve ser equivalente ao bit de informação que se quer armazenar (0 ou 1).

Algoritmo 16 - Extração da DHTC

1) Reconstruir exatamente a mesma seqüência *v* criada no processo de inserção.
2) Ordenar a seqüência *v* como no processo de inserção.
3) Os pixels centrais dos *n* primeiros blocos de v são os DBPs e seus valores são a informação embutida. Extrair os *n* bits de informação dos pixels centrais dos *n* primeiros blocos de *v*.

A técnica DHTC modifica apenas pixels de baixa visibilidade para embutir informações e conseqüentemente imagens marcadas pela DHTC possuem excelente qualidade visual e não apresentam ruídos do tipo sal-e-pimenta.

Na DHTC, a posição exata dos *n* DBPs é conhecida tanto no processo de inserção quanto no de extração. Esta propriedade torna possível transformar DHTC em uma técnica esteganográfica reversível. Para isto, os valores originais dos DBPs devem ser comprimidos, adicionados às informações que realmente se quer embutir e armazenados nos próprios DBPs. Esta técnica reversível será proposta no Capítulo 4.

2.7.7 ***Authentication Watermarking by Template ranking with symmetrical Central pixels* (AWTC)**

Como foi visto anteriormente, muitas AWTs para imagens binárias podem sofrer adulterações devido a uma técnica chamada ataque de paridade (KIM; DE QUEIROZ, 2004a, 2004b). Para AWTs de chave secreta, alguns métodos de prevenção a esses ataques têm sido propostos (KIM; DE QUEIROZ, 2004b). Para AWTs de chave pública, existe uma técnica totalmente imune aos ataques de

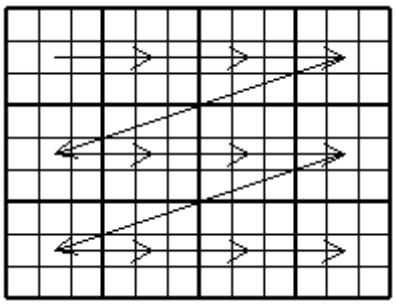


Figura 16 – Uma imagem 9×12 dividida em blocos 3×3 e escaneada da esquerda para a direita, de cima para baixo (seqüência *raster*) (KIM, 2005).

paridade chamada AWTC (*Authentication Watermarking by Template ranking with symmetrical Central pixels* - marca d'água de autenticação pelo ranqueamento de modelos com pixels centrais simétricos) que será descrita a seguir (KIM, 2005).

Como a AWTC é baseada na técnica esteganográfica DHTC, a imagem é dividida em blocos (por exemplo, 3×3) e apenas o pixel central de cada bloco é utilizado para armazenar a marca d'água.

Algoritmo 17 - Inserção da AWTC

1) Dividir a imagem *Z* a ser marcada em uma seqüência *v* de blocos (por exemplo, 3×3) não sobrepostos. O meio mais simples de obter esta divisão é percorrer a imagem *Z* da esquerda para a direita, de cima para baixo (seqüência *raster*) dividindo-a em blocos 3×3 como visto na Figura 16 (blocos incompletos nas bordas da imagem são descartados). Apenas os pixels centrais de cada bloco de *v* terão sua cor modificada pela inserção da marca.
2) Ordenar a seqüência *v* de maneira crescente usando a nota de impacto visual (Figura 15) como chave primária e números pseudo-aleatórios sem repetição como chave secundária. A chave secundária evita que as informações sejam inseridas apenas na parte superior da imagem.
3) Limpar os pixels centrais dos *n* primeiros blocos da seqüência ordenada *v*, onde *n* é o comprimento da AS.
4) Usando uma função de *hash* criptograficamente segura, computar o resumo criptográfico *H* da imagem *Z* limpa.
5) Criptografar o resumo *H* com a chave secreta ou a chave privada, obtendo a AS *S*.

6) Restaurar a imagem *Z* original.
7) Inserir *n* bits de *S* invertendo (se necessário) a cor do pixel central dos *n* primeiros blocos de *v*.

Algoritmo 18 - Verificação da AWTC

1) Reconstruir e ordenar novamente a seqüência *v* de blocos 3×3 não sobrepostos. O resultado é exatamente a mesma seqüência *v* usada na inserção. Os valores dos *n* primeiros pixels centrais contêm a informação escondida.
2) Extrair a AS *S* dos *n* primeiros pixels centrais da seqüência ordenada *v*.
3) Decriptografar *S* usando a chave secreta ou pública, obtendo o resumo criptográfico extraído *H*.
4) Limpar os *n* primeiros pixels centrais da seqüência ordenada *v* e computar o resumo criptográfico de verificação *C* da imagem *Z* limpa.
5) Se o resumo extraído *H* for igual ao resumo computado *C*, a marca está verificada. Caso contrário, a imagem foi modificada.

As imagens marcadas por esta técnica possuem excelente qualidade visual e não apresentam ruídos do tipo sal-e-pimenta, pois apenas pixels de baixa visibilidade são alterados. Esta técnica pode detectar qualquer alteração na imagem, até mesmo a mudança de um único pixel.

Por que os ataques de paridade não se aplicam ao AWTC? Porque a quantidade de pixels que armazenam informação na região Z_1 é exatamente igual ao comprimento da AS utilizada e também porque todos os pixels da imagem, exceto os *n* pixels que carregam os *n* bits da AS, são levados em conta para computar a AS (região Z_2). Conseqüentemente, qualquer alteração na região Z_2 pode ser detectada porque causa mudança no cômputo da AS da imagem marcada, e qualquer alteração na região Z_1 também pode ser detectada, pois muda a AS armazenada.

Como esta técnica é imune a ataques de paridade, as suas versões de chave secreta e de chave pública são consideradas seguras.

2.8 MARCAS D'ÁGUA REVERSÍVEIS

A maioria das técnicas esteganográficas modifica e conseqüentemente distorce a imagem hospedeira para que a informação adicional possa ser inserida. Essa distorção geralmente é pequena, porém irreversível.

Técnicas reversíveis de esteganografia (*Reversible Data Hidings*) inserem a informação na imagem hospedeira, modificando seus pixels, porém permitem a exata restauração (sem perda) da imagem original depois que a informação adicional é extraída.

Na literatura, podem ser encontrados vários termos utilizados para definir técnicas reversíveis: *reversible*, *distortion-free*, *invertible*, *lossless*, *erasable* e outros.

Na maioria das aplicações, a pequena distorção causada pela inserção da informação adicional é tolerável. Porém, a possibilidade de se recuperar a imagem original é uma propriedade desejável em muitos campos, como legal, médico, astronômico e militar.

Alguns autores (AWRANGJEB; KANKANHALLI, 2004; CELIK et al., 2002a; SHI, 2004) classificam as técnicas esteganográficas reversíveis em dois tipos:

1) baseadas em espectro de expansão aditivo (*additive spread spectrum*);
2) baseadas em compressão de características da imagem (marcas reversíveis de alta capacidade).

O primeiro tipo (FRIDRICH et al., 2001; HONSINGER et al., 2001) faz uso de técnicas de espectro de expansão aditivo. Nestas técnicas, um sinal de espectro de expansão (*spread spectrum*) correspondente aos dados a serem embutidos (adicionados) é sobreposto ao sinal hospedeiro. Na decodificação, a informação escondida é detectada e o sinal adicionado é removido (subtraído) para restabelecer o sinal hospedeiro original. Neste tipo, a extração da informação escondida é feita de forma robusta, de modo que a mesma pode ser extraída mesmo se a imagem for ligeiramente modificada. Porém, neste caso, a imagem original não pode ser recuperada. Essas técnicas utilizam aritmética de resto (*modulus arithmetic*) para evitar erros de *overflow*/*underflow*, o que pode causar ruídos do tipo sal-e-pimenta na imagem. Além disso, elas normalmente oferecem capacidade muito limitada para

esconder informações. Este primeiro tipo não será analisado neste trabalho, por oferecer baixa capacidade de armazenamento e por ter um comportamento robusto, não podendo ser utilizada para marcas de autenticação.

O segundo tipo (AWRANGJEB; KANKANHALLI, 2004; CELIK et al., 2002a, 2005; FRIDRICH et al., 2002; NI et al., 2004; TIAN, 2002, 2003) sobrescreve algumas porções do sinal hospedeiro com os dados a serem embutidos. Para permitir a reversibilidade, devem ser embutidos dois tipos de informação: os dados comprimidos referentes à porção sobrescrita (que permitirá recuperar o sinal original) e as informações que realmente se quer embutir (NPD – *Net Payload Data*). Durante a decodificação, a informação embutida é extraída, a NPD é recuperada, e os dados comprimidos são usados para restabelecer o sinal original. Estas técnicas não causam ruídos do tipo sal-e-pimenta, porque as porções modificadas normalmente são os bits menos significativos (LSBs) ou os coeficientes de *wavelet* de alta freqüência que não causam distorção perceptível. Estas técnicas normalmente oferecem maior capacidade de armazenamento do que as do primeiro tipo. A técnica RDTC (*Reversible Data hiding by Template ranking with symmetrical Central pixels* - esteganografia reversível pelo ranqueamento de modelos com pixels centrais simétricos) proposta no Capítulo 4 desta tese pertence a este segundo tipo.

Existem dois principais desafios para a criação de uma técnica esteganográfica do segundo tipo para imagens binárias:

1) O primeiro é encontrar uma técnica que permita localizar precisamente os pixels modificados tanto no processo de inserção quanto no processo de extração, para que a recuperação da imagem original seja possível. Considere, por exemplo, uma técnica onde a imagem hospedeira é dividida em blocos e um bit é inserido em cada bloco modificando-se (se necessário) o pixel de menor visibilidade. Os blocos com número par (ímpar) de pixels pretos possuem o bit zero (um) embutido. Nesta técnica, a imagem original não pode ser recuperada, mesmo que as paridades originais dos pixels pretos de cada bloco sejam conhecidas, porque o pixel modificado dentro de cada bloco não pode ser localizado.
2) O segundo desafio é encontrar um método eficiente para a compressão da porção da imagem que será sobrescrita pelos dados embutidos. Esta porção é tipicamente pequena (380 bits, por exemplo), não tem estrutura, possui

seus bits espalhados por toda a imagem e seus bits não possuem nenhuma relação entre si. A compressão direta desses dados resulta em uma baixa capacidade de armazenamento. Uma estratégia para minimizar este problema é utilizar o restante da imagem como informação adicional. Desta forma, significantes ganhos de compressão podem ser obtidos (CELIK et al., 2002a). Em técnicas esteganográficas reversíveis para imagens *contone* (coloridas e em níveis de cinza), a escolha do algoritmo de compressão parece não ser crítica, porque existe muito espaço para armazenar informação (bits menos significativos, por exemplo). Para este tipo de imagem, Awrangjeb e Kankanhalli (2004) usam Codificação Aritmética (*Arithmetic Coding*), LZW e JBIG para compressão sem perda e Celik et al. (2002a) usam uma versão adaptada do algoritmo CALIC (WU, 1997). Porém, em técnicas esteganográficas reversíveis para imagens binárias, muitos algoritmos de compressão baseados em redundância e dicionários não são efetivos (a informação comprimida fica maior do que a original). A técnica reversível RDTC, proposta no Capítulo 4, utiliza o algoritmo de Golomb para comprimir as previsões dos pixels de baixa visibilidade para obter espaço para embutir a informação desejada.

Existem também algumas técnicas que não se encaixam em nenhum dos tipos anteriores como a técnica de Tian (2003) descrita mais adiante neste capítulo.

2.9 TÉCNICAS REVERSÍVEIS PARA IMAGENS COLORIDAS E EM NÍVEIS DE CINZA

Existem várias técnicas esteganográficas reversíveis propostas para imagens de tonalidade contínua. Shi (2004) apresenta uma pesquisa sobre essas técnicas esteganográficas e Awrangjeb e Kankanhalli (2004) apresentam uma pesquisa sobre AWTs reversíveis. A seguir, a título de exemplo, serão detalhadas duas destas técnicas.

2.9.1 **Técnica reversível de Celik et al. (2005)**

Esta técnica, proposta por Celik et al. (2005) e discutida em Feng et al. (2006), utiliza a compressão de dados para reduzir o tamanho da informação a ser embutida. Nesta técnica, cada pixel é quantificado usando a função de quantização (*L-level scalar quantization*) a seguir:

$$Q_L(x) = L \times \left\lfloor \frac{x}{L} \right\rfloor$$

Na função acima, os símbolos $\lfloor$ e $\rfloor$ indicam um arredondamento para baixo.

Por exemplo, considerando o seguinte bloco 4×4 da imagem original:

$$H = \begin{pmatrix} 20 & 37 & 7 & 22 \\ 35 & 12 & 32 & 13 \\ 22 & 12 & 18 & 23 \\ 12 & 23 & 12 & 26 \end{pmatrix}$$

O bloco quantificado, usando L=5, é:

$$Q = \begin{pmatrix} 20 & 35 & 5 & 20 \\ 35 & 10 & 30 & 10 \\ 20 & 10 & 15 & 20 \\ 10 & 20 & 10 & 25 \end{pmatrix}$$

Depois da quantização, uma matriz de restos também é gerada. Neste exemplo, os restos são:

$$R = \begin{pmatrix} 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 3 \\ 2 & 2 & 3 & 3 \\ 2 & 3 & 2 & 1 \end{pmatrix}$$

O próximo passo é comprimir os restos utilizando uma adaptação do algoritmo para compressão sem perdas CALIC (WU, 1997). Segundo Feng et al. (2006), pode-se assumir que os 16 restos podem ser comprimidos de maneira reversível para 12 valores denotados como { x_0, x_1, …, x_{11} } e a marca d'água pode ser expressa em 4 valores na base 5, por exemplo, $\{4\ 2\ 1\ 0\}_5$, a imagem marcada *H'* pode ser gerada adicionando-se a informação comprimida e a marca d'água à imagem quantificada.

$$H' = \begin{pmatrix} 20+x_0 & 35+x_1 & 5+x_2 & 20+x_3 \\ 35+x_4 & 10+x_5 & 30+x_6 & 10+x_7 \\ 20+x_8 & 10+x_9 & 15+x_{10} & 20+x_{11} \\ 10+4 & 20+2 & 10+1 & 25+0 \end{pmatrix}$$

No processo de extração, a imagem é quantificada como no processo de inserção, e os valores de { x_0, x_1, …, x_{11} } e a marca d'água são recuperados. A marca d'água pode ser utilizada para autenticar a imagem e os outros valores podem ser utilizados para recuperar a imagem original.

2.9.2 **Técnica reversível de Tian (2003)**

Esta técnica, proposta por Tian (2003) e também discutida em Feng et al. (2006), utilize a expansão de diferenças (*difference expansion*) para embutir informação.

Algoritmo 19 - Inserção da técnica de Tian

1) Selecionar dois pixels adjacentes (*x* e *y*) para embutir um bit da marca d'água (*w*).
2) Computar os valore de *l*, *h* e *h´*:

$$l = \left\lfloor \frac{(x+y)}{2} \right\rfloor \qquad h = x - y \qquad h' = h \times 2 + w$$

3) Computar o valor dos pixels marcados (*x'* e *y'*):

$$x' = l + \left\lfloor \frac{(h'+1)}{2} \right\rfloor \qquad y' = l - \left\lfloor \frac{h'}{2} \right\rfloor$$

Exemplo: Considerando os pixels *x*=106 e *y*=100 e o bit de marca d'água *w*=1, os pixels marcados (*x´* e *y´*) podem ser calculados da seguinte forma:

$$l = \left\lfloor \frac{(x+y)}{2} \right\rfloor = \left\lfloor \frac{(106+100)}{2} \right\rfloor = 103 \qquad h = x - y = 106 - 100 = 6$$

$$h' = h \times 2 + w = 6 \times 2 + 1 = 13$$

$$x' = l + \left\lfloor \frac{(h'+1)}{2} \right\rfloor = 103 + \left\lfloor \frac{(13+1)}{2} \right\rfloor = 110 \qquad y' = l - \left\lfloor \frac{h'}{2} \right\rfloor = 103 - \left\lfloor \frac{13}{2} \right\rfloor = 97$$

Algoritmo 20 - Extração da técnica de Tian

1) Selecionar dois pixels adjacentes (*x'* e *y'*) que possuam um bit da marca d'água (*w*) inserido.
2) Computar os valores de *l* e *h´*:

$$l = \left\lfloor \frac{(x'+y')}{2} \right\rfloor \qquad h' = x'-y'$$

3) Computar o bit de marca d'água *w* extraindo o bit menos significativo de *h´*.
4) Computar o valor de *h* e o valor dos pixels originais (*x* e *y*):

$$h = \left\lfloor \frac{h'}{2} \right\rfloor \qquad x = l + \left\lfloor \frac{(h+1)}{2} \right\rfloor \qquad y = l - \left\lfloor \frac{h}{2} \right\rfloor$$

Exemplo: Considerando os pixels marcados *x´*=110 e *y´*=97, o bit de marca d'água *w* pode ser extraído da seguinte forma:

$$l = \left\lfloor \frac{(x'+y')}{2} \right\rfloor = \left\lfloor \frac{(110+97)}{2} \right\rfloor = 103 \qquad h' = x'-y' = 110-97 = 13$$

O valor de *h´* em binário é $(1101)_2$ e o bit menos significativo (1) é a informação embutida (*w*=1).

$$h = \left\lfloor \frac{h'}{2} \right\rfloor = \left\lfloor \frac{13}{2} \right\rfloor = 6 \qquad x = l + \left\lfloor \frac{(h+1)}{2} \right\rfloor = 103 + \left\lfloor \frac{(6+1)}{2} \right\rfloor = 106$$

$$y = l - \left\lfloor \frac{h}{2} \right\rfloor = 103 - \left\lfloor \frac{6}{2} \right\rfloor = 100$$

O valor original dos pixels utilizados para embutir informação são *x*=106 e *y*=100.

2.10 TÉCNICAS REVERSÍVEIS PARA IMAGENS BINÁRIAS

Muitas técnicas esteganográficas reversíveis e AWTs reversíveis têm sido desenvolvidas como, por exemplo, Awrangjeb e Kankanhalli (2004), Celik et al. (2002b), Fridrich; Goljan e Du (2001, 2002), Honsinger et al. (2001), Shi (2004), Tian (2002), Tsai et al. (2004a, 2004b). Porém, segundo nossas pesquisas, nenhum dos

métodos reversíveis existentes na literatura é adequado para autenticação de imagens binárias.

2.10.1 ***Pair-Wise Logical Computation*** **(PWLC)**

Segundo nossos estudos, a única técnica esteganográfica reversível para imagens binárias proposta é PWLC (*Pair-Wise Logical Computation*) (TSAI et al., 2004a, 2004b). Porém, esta técnica parece algumas vezes não conseguir extrair corretamente a informação embutida e falhar na recuperação perfeita da imagem original.

PWLC não utiliza técnicas de espectro de expansão aditivo ou de compressão. Utiliza operações binárias XOR para embutir informações na imagem hospedeira. Esta técnica percorre a imagem em uma determinada ordem (por exemplo, escaneando da esquerda para a direita, de cima para baixo) e apenas seqüências “000000” ou “111111” localizadas próximas aos contornos da imagem são escolhidas para armazenar informações. A seqüência “000000” torna-se “001000” se o bit 0 é inserido, e torna-se “001100” se o bit 1 é inserido. Similarmente, a seqüência “111111” torna-se “110111” se o bit 0 é inserido, e torna-se “110011” se o bit 1 é inserido.

Esta técnica constrói uma cadeia de bits (*bitstream*) com a informação a ser embutida na imagem. Então, insere um bit “1” (bit de referência) em frente a cada bit da cadeia, formando pares de bits. Esses pares são armazenados na imagem hospedeira, sobrepondo pares de mesmo valor (“00” ou “11”) usando a operação XOR. Desta forma, a informação original pode ser recuperada, pois sabe-se que a informação embutida sempre tem o primeiro bit igual a “1” e que a informação sobreposta sempre tem um par de bits de mesmo valor. A informação original e a informação embutida podem ser identificadas de forma única como pode ser visto na Tabela 1.

Tabela 1 – Tabela de Busca (*Look-Up Table*) da técnica PWLC (TSAI et al., 2004a, 2004b).

Par de bits sobrepostos na imagem hospedeira		Informação embutida			
1° pixel	2º pixel	Bit de referência "1"	Bit embutido "0"	Bit de referência "1"	Bit embutido "1"
0	0	1	0	1	1
1	1	0	1	0	0

Porém, os artigos de Tsai et al. (2004a, 2004b) não descrevem claramente como identificar os pixels modificados no processo de extração. Os contornos da imagem podem ser modificados com a inserção da informação. Além disso, suponha que uma seqüência "001000" (localizada próxima ao contorno da imagem) foi encontrada na imagem marcada. Os artigos não descrevem como diferenciar uma seqüência "001000" não marcada de uma seqüência "000000" original que se tornou "001000" devido à inserção de um bit 0.

3 MARCAS D'ÁGUA DE AUTENTICAÇÃO PARA IMAGENS JBIG2

3.1 INTRODUÇÃO

Neste capítulo serão propostas técnicas esteganográficas e AWTs para imagens no formato JBIG2. Inicialmente será descrita a codificação de imagens no padrão JBIG2, como é feita a segmentação dessas imagens e qual a estrutura de cada uma das regiões que compõem uma imagem JBIG2. Em seguida serão apresentadas propostas para três novas técnicas esteganográficas e quatro novas AWTs para imagem codificadas no padrão JBIG2. Também serão apresentados os resultados experimentais obtidos pela aplicação de cada uma das técnicas propostas.

3.2 FORMATO JBIG2

3.2.1 **Introdução**

O *Joint Bi-level Image Experts Group* (JBIG), uma "equipe colaborativa" estabelecida em 1988, foi responsável pela elaboração do padrão JBIG2 (ISO/IEC, 1999). Este padrão define um método de compressão para imagens em dois níveis (*bi-level*), que geralmente são o preto e o branco. Este formato está explicitamente preparado para compressões com perda ou sem perda e também permite que uma imagem codificada com perda possa ser recuperada totalmente (sem perda) através de informações adicionais que podem ser utilizadas para o refinamento da mesma (HOWARD et al., 1998).

JBIG2 foi projetado para permitir um melhor desempenho na compressão sem perda do que os padrões existentes e permitir uma compressão com perda com maiores taxas de compressão, sem uma visível degradação na qualidade da imagem.

Além disso, o padrão JBIG2 permite codificação progressiva de qualidade (com o progresso indo de uma baixa qualidade até uma alta qualidade, chegando até a uma

compressão sem perda) e também codificação progressiva de conteúdo (adicionando sucessivamente diferentes tipos de imagens, por exemplo, primeiro textos depois meio-tons).

Este formato também possui um alto desempenho em termos de velocidade de descompressão (JPEG COMMITTEE, 2004).

O padrão JBIG2, como a grande maioria dos padrões, não define explicitamente um codificador padrão (*encoder*); ele apenas define os requisitos para uma cadeia de bits (*bitstream*) compatível.

Segundo Atalasoft (2006), fabricante do codificador/decodificador para JBIG2 “DotImage”, o uso de dicionários de símbolos permite uma codificação eficiente de imagens que possuem símbolos recorrentes, o que faz o JBIG2 ideal para a compressão de documentos.

O tamanho final de um arquivo codificado com JBIG2 chega a ser de 2 a 5 vezes menor do que se comprimido pelos métodos industriais tradicionais (TIFF CCITT Group 4), como pode ser visto na Tabela 2.

Tabela 2 – Comparação JBIG2, G4 e PDF (ATALASOFT, 2006)

	Sem compressão	CCITT G4	JBIG2 sem perda	PDF com JBIG2 sem perda
Fax de 3 páginas 1728×2293	1.454 kb	121 kb	24 kb	26 kb
Diagrama escaneado 9760×6976	8.321 kb	1.088 kb	674 kb	676 kb
Desenho mecânico (CAD) 9259×6816	7.716 kb	209 kb	100 kb	105 kb

Segundo CVISION TECHNOLOGIES (2007), uma imagem comprimida com o padrão TIFF CCITT G4 é, em geral, 15 vezes menor do que a imagem não comprimida (supondo uma imagem binária de resolução 300 dpi). A mesma imagem, se compactada com JBIG2, será de 5 a 15 vezes menor que a imagem G4. A diferença é ainda maior quando o arquivo original possui uma origem eletrônica (não escaneado). Nestes casos, a taxa de compressão do JBIG2 pode ser de 20 a 30 vezes menor comparado com o G4.

Em (KNOLL, 2007) é feita uma comparação entre vários esquemas de compressão para uma página de revista no formato binário. Os resultados podem ser vistos na Tabela 3.

Tabela 3 – Tabela comparativa entre vários métodos de compressão (KNOLL, 2007)

Sem compressão	CCITT G3	CCITT G4	JBIG2 sem perda
3.314 kb	477 kb	192 kb	154 kb

O formato JBIG2 tem se tornado popular devido ao formato PDF, criado pela *Adobe Systems Incorporated*, que incorporou o JBIG2 a partir da especificação 1.4 deste formato, o que reduziu em muito o tamanho dos arquivos codificados.

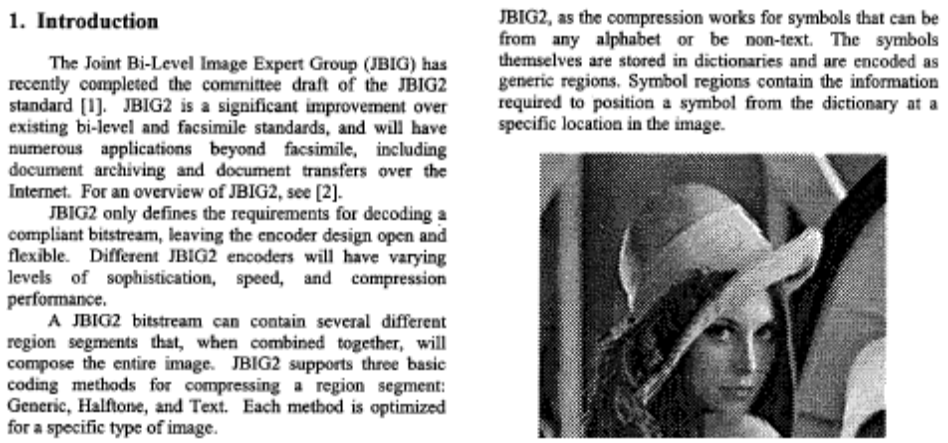
Antes da especificação 1.4 (ADOBE SYSTEMS INCORPORATED, 2001), o formato PDF suportava os seguintes filtros para compressão/descompressão:

- DCT (baseado em JPEG) para compressão de imagens coloridas e em níveis de cinza;
- CCITT (*Group* 3 e *Group* 4) e *RunLength* para compressão de imagens binárias;
- LZW (Lempel-Ziv-Welch) e *Flate* para compressão de textos, gráficos e imagens.

A partir de 2001 o formato passou a contar também com o filtro para JBIG2 para compressão de textos e imagens binárias.

Segundo *Adobe Systems Incorporated* (2001), usando a compressão JPEG, imagens coloridas e em níveis de cinza podem ser comprimidas a taxas superiores a 10:1. A compressão de imagens binárias depende do filtro utilizado e das propriedades da imagem, mas reduções de 2:1 a 8:1 são comuns (ou 20:1 a 50:1 para a compressão JBIG2 de uma página de texto). LZW e *Flate* são utilizados para a compressão dos demais textos e gráficos presentes no documento e resultam em taxas de compressão de aproximadamente 2:1.

A criação e implementação de AWTs seguras para o formato JBIG2 parece ser um problema prático importante. Documentos escaneados são grandes imagens binárias que precisam ser protegidas contra alterações fraudulentas. Além disso,



1. Introduction

The Joint Bi-Level Image Expert Group (JBIG) has recently completed the committee draft of the JBIG2 standard [1]. JBIG2 is a significant improvement over existing bi-level and facsimile standards, and will have numerous applications beyond facsimile, including document archiving and document transfers over the Internet. For an overview of JBIG2, see [2].

JBIG2 only defines the requirements for decoding a compliant bitstream, leaving the encoder design open and flexible. Different JBIG2 encoders will have varying levels of sophistication, speed, and compression performance.

A JBIG2 bitstream can contain several different region segments that, when combined together, will compose the entire image. JBIG2 supports three basic coding methods for compressing a region segment: Generic, Halftone, and Text. Each method is optimized for a specific type of image.

JBIG2, as the compression works for symbols that can be from any alphabet or be non-text. The symbols themselves are stored in dictionaries and are encoded as generic regions. Symbol regions contain the information required to position a symbol from the dictionary at a specific location in the image.

(a) Imagem original.

(b) Imagem reduzida.

Figura 17 – Exemplo de aplicação da técnica de segmentação. (a) Imagem original escaneada a 200 dpi. (b) Imagem reduzida utilizando blocos 8×8 (TOMPKINS; KOSSENTINI, 1999).

imagens binárias de documentos precisam ser salvas de maneira comprimida para diminuir o espaço de armazenamento necessário.

3.2.2 **Segmentação**

Uma imagem, para ser codificada no padrão JBIG2, precisa ser decomposta em 3 tipos de região: texto, meio-tom e genérica. Então, cada região é codificada do modo mais apropriado. Em Tompkins e Kossentini (1999) pode ser visto um método rápido e eficiente para a segmentação de imagens binárias, que separa as regiões de texto das demais regiões.

Este método requer que a análise seja feita em uma imagem de tamanho reduzido. Porém, ao invés de reduzir a imagem através das técnicas tradicionais de *downsampling*, uma técnica específica, baseada em blocos, foi desenvolvida.

Nesta técnica, cada pixel da imagem reduzida corresponde a um bloco *N*×*M* na imagem original. Um pixel na imagem reduzida é branco se todos os pixels do bloco correspondente forem brancos. Caso contrário, o pixel na imagem reduzida é preto. Uma imagem obtida por esta técnica pode ser vista na Figura 17.

Após a redução da imagem, os componentes 8-conexos são analisados para ver se possuem ou não características de uma região de texto. Em geral, regiões de meio-tom e genéricas aparecem como grandes blocos pretos e regiões de texto aparecem como blocos pequenos e largos.

3.2.3 **Região de Texto**

Cada região no formato JBIG2 é codificada em vários segmentos. A região de texto, por exemplo, possui dois tipos de segmento:

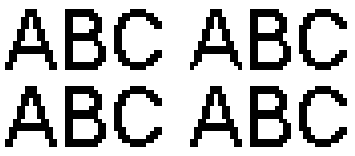
- *Symbol dictionary segment* – Dicionário de símbolos que contém os mapas de bits (*bitmaps*) correspondentes aos caracteres presentes na região de texto.
- *Text region segment* – Descreve a posição dos caracteres dentro da região de texto, através de referências ao dicionário de símbolos.

Na literatura podem ser encontradas algumas referências sobre a criação de dicionários de símbolos (YE et al., 2000; YE; COSMAN, 2001).

Em uma região de texto, vários caracteres podem se referir ao mesmo símbolo no dicionário, aumentando a taxa de compressão. Na compressão com perda, instâncias similares de um mesmo símbolo podem se referir ao mesmo símbolo no dicionário. Isto acontece, por exemplo, em documentos escaneados onde muitas instâncias de uma mesma letra podem ser ligeiramente diferentes. Se estas letras se referirem a um único símbolo no dicionário, a qualidade da imagem diminui, mas a taxa de compressão aumenta.

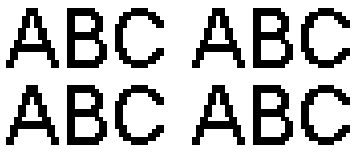
(a) Imagem original (ampliada).

(b) Dicionário da imagem original.



(c) Imagem comprimida com perda (1 pixel/símbolo).

(d) Dicionário da imagem comprimida com perda (1 pixel/símbolo).



(e) Imagem comprimida com perda (2 pixels/símbolo).

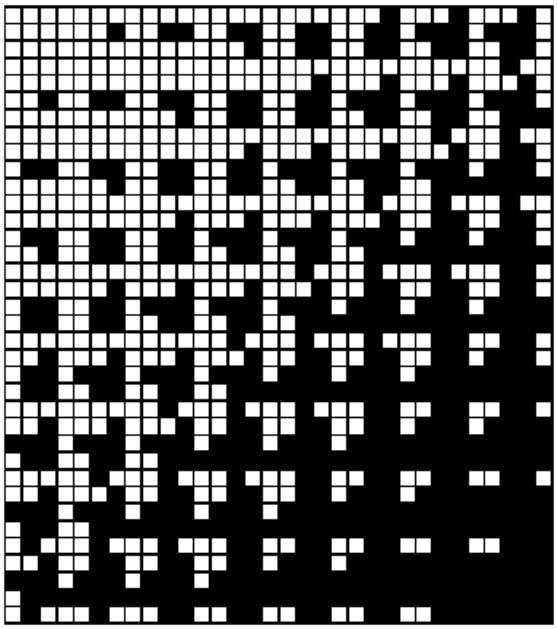
(f) Dicionário da imagem comprimida com perda (2 pixels/símbolo).

Figura 18 – Exemplo de compressão de uma região de texto de uma imagem JBIG2 com e sem perda de informação. (a) Imagem original ampliada. (b) Dicionário produzido pela codificação sem perda da imagem original. (c) Imagem codificada com perda (símbolos que diferem em apenas 1 pixel são considerados iguais). (d) Dicionário produzido pela codificação com perda de 1 pixel/símbolo. (e) Imagem codificada com perda (2 pixels/símbolo). (f) Dicionário produzido pela codificação com perda de 2 pixels/símbolo.

A Figura 18 mostra uma região de texto comprimida pelo padrão JBIG2, com e sem perda de informação. Note que, nas imagens 18a e 18b, as quatro instâncias da letra “A” são ligeiramente diferentes na parte inferior. As imagens 18a e 18b mostram a compressão sem perda e as imagens 18c a 18f mostram a compressão com perda Nas imagens 18c e 18d foram considerados similares símbolos que tinham apenas 1 pixel de diferença. Nas imagens 18e e 18f foram considerados similares símbolos que tinham 2 pixels de diferença. Esta quantidade de pixels que diferem entre um símbolo e outro é conhecida como “Distância de Hamming”.

3.2.4 **Região de Meio-Tom**

Para codificar uma região de meio-tom, o padrão JBIG2 quebra a região em pequenos blocos chamados de padrões (*patterns*), por exemplo, blocos 8×8, e cria um dicionário de padrões para armazenar uma cópia única de cada bloco (sem duplicação).



(a) Região de meio-tom de uma imagem JBIG2 (ampliada).

(b) Dicionário de padrões (ampliado).

Figura 19 – Exemplo de compressão de uma região de meio-tom de uma imagem JBIG2, sem perda de informação (ISO/IEC, 1999). (a) Imagem original ampliada. (b) Dicionário de padrões produzido pela codificação sem perda da imagem original usando blocos 4×4. O quadriculado exibido não faz parte da imagem original, apenas foi inserido para melhor visualização dos blocos.

A região é então codificada, de maneira semelhante à região de texto, utilizando referências ao dicionário de padrões (Figura 19). Uma região de meio-tom no padrão JBIG2 possui dois tipos de segmento:

- *Pattern dictionary segment* – Dicionário de padrões que contém os *bitmaps* dos padrões presentes na região de meio-tom.
- *Halftone region segment* – Descreve a localização dos padrões dentro da região de meio-tom, através de referências ao dicionário de padrões.

Na compressão sem perda, todas as instâncias idênticas de um padrão se referem a um único padrão no dicionário, aumentando a taxa de compressão. Na compressão com perda, instâncias similares de um padrão se referem a um único padrão no dicionário, aumentando ainda mais a taxa de compressão, em troca da introdução de alguma distorção na imagem codificada.

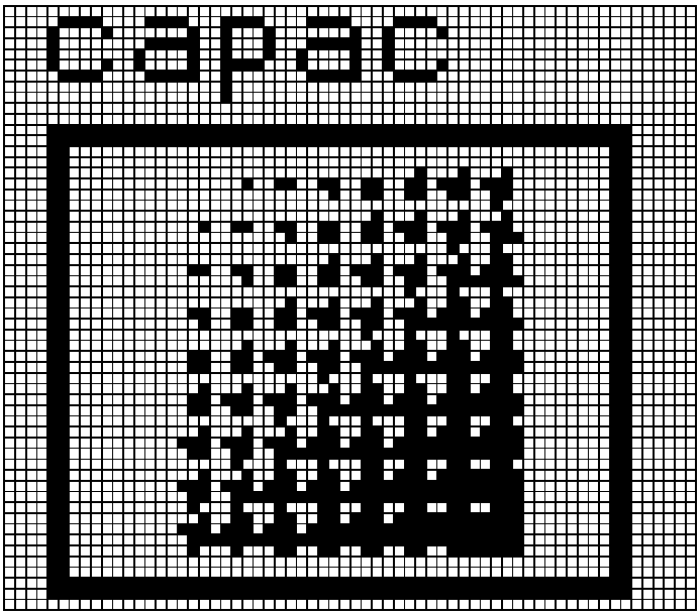


Figura 20 – Exemplo de uma imagem JBIG2 com regiões de texto, meio-tom e genérica (ISO/IEC, 1999). O quadriculado exibido não faz parte da imagem original, apenas foi inserido para melhor visualização dos blocos.

3.2.5 **Região Genérica**

As regiões que não puderem ser classificadas como texto ou meio-tom são consideradas regiões genéricas.

Na Figura 20, podemos ver uma imagem que possui claramente uma região de texto, uma região de meio-tom e uma região genérica (borda retangular).

3.3 *DATA-HIDING BY COORDINATE CHANGING FOR JBIG2* (DHCCJ)

A técnica DHCCJ (*Data-Hiding by Coordinate Changing for JBIG2* - esteganografia por mudança de coordenada para JBIG2) insere informações na região de texto da imagem JBIG2, mais precisamente no segmento *text region segment*. Cada referência neste segmento é composta basicamente por um número que identifica o símbolo no dicionário e a posição (coordenadas x e y) do símbolo na região de texto.

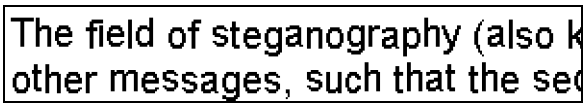


Figura 21 – Ampliação de parte de uma imagem hospedeira gerada pela técnica DHCCJ (PAMBOUKIAN; KIM; DE QUEIROZ, 2005).

Um bit pode ser inserido em cada referência, forçando uma de suas coordenadas a ser par ou ímpar, para inserir os bits 0 ou 1, respectivamente.

3.3.1 **Resultados experimentais da técnica DHCCJ**

A Figura 21 mostra o exemplo de uma imagem com resolução 81×81 dpi marcada usando a técnica DHCCJ. A informação foi embutida na coordenada y dos símbolos. O deslocamento dos símbolos é claramente visível, prejudicando a qualidade visual da imagem. De acordo com os testes realizados, apenas imagens com resolução maior ou igual a 600 dpi podem ser marcadas por esta técnica com qualidade visual aceitável. Como a resolução típica para documentos escaneados no formato binário é de 300 dpi, foram desenvolvidos outros métodos mais eficientes para imagens de baixa resolução, como será visto a seguir.

Como esta técnica mostrou-se ineficiente, não foi desenvolvida uma AWT baseada na mesma.

3.4 *DATA-HIDING BY TEMPLATE RANKING FOR JBIG2* (DHTRJ)

A técnica DHTRJ (*Data Hiding by Template Ranking for JBIG2* - esteganografia pelo ranqueamento de modelos para JBIG2) insere informações na região de texto do arquivo JBIG2, mais precisamente no segmento *symbol dictionary segment*. Imagens marcadas com esta técnica apresentam boa qualidade visual, até mesmo imagens de baixa resolução.

DHTRJ seleciona de maneira pseudo-aleatória (usando uma semente pré-definida) uma quantidade apropriada de instâncias de símbolos da região de texto para

armazenar a informação. A quantidade de instâncias selecionadas deve ser escolhida de tal forma que seja suficiente para armazenar toda a quantidade de informação desejada.

Como cada símbolo presente no dicionário pode ser referenciado por muitas instâncias, a alteração de um símbolo pode ter seu efeito multiplicado, prejudicando a qualidade visual da imagem marcada. A solução é duplicar os símbolos que receberão os dados no *symbol dictionary segment* e modificar o *text region segment* de modo que apenas uma instância do símbolo (aquela escolhida de maneira pseudo-aleatória) faça referência ao símbolo que irá carregar informação (todas as outras instâncias continuarão se referindo ao símbolo original).

Os símbolos que carregam informação (DBSs – *Data Bearing Symbols*) podem ser inseridos no final do *symbol dictionary segment* ou em um novo segmento especialmente criado para armazená-los.

Nesta técnica, um bit poderia ser inserido em cada DBS forçando o mesmo a ter uma quantidade de pixels pretos par ou ímpar. Porém, se apenas um bit for armazenado em cada DBS, o tamanho do dicionário crescerá de maneira significativa, afetando o poder de compressão do JBIG2. A solução é armazenar vários bits em cada DBS, usando uma técnica similar à DHTR descrita anteriormente.

Algoritmo 21 - Inserção da DHTRJ

1) Escolher de forma pseudo-aleatória (utilizando alguma semente), uma quantidade *n* de instâncias de símbolos que serão utilizadas para armazenar informação e identificar seus respectivos DBSs no dicionário de símbolos.
2) Embaralhar de forma pseudo-aleatória (utilizando alguma semente), o conjunto de todos os pixels de todos os DBSs.
3) Dividir o conjunto de pixels embaralhados dos DBSs em pequenos blocos (cada bloco com, por exemplo, 64 pixels).
4) Analisar a vizinhança (geralmente 3×3) de cada pixel para determinar sua significância visual.
5) Inserir um bit em cada bloco, forçando o mesmo a ter uma quantidade par ou ímpar de pixels pretos.

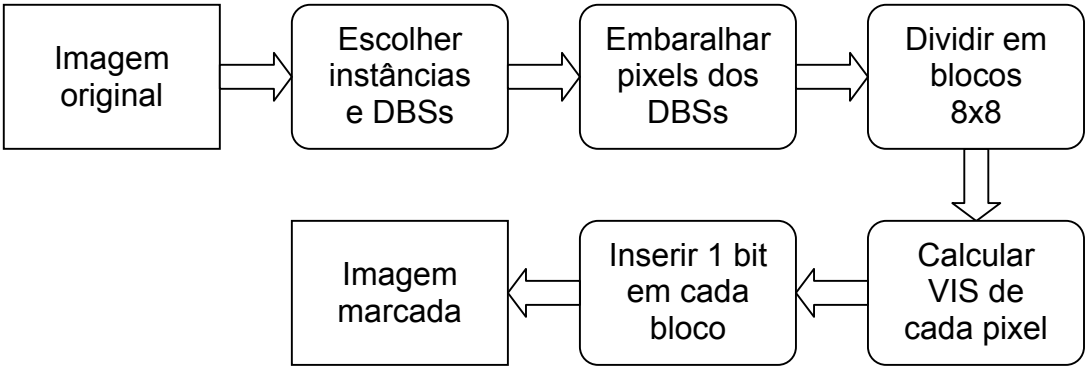


Diagrama 1 – Algoritmo de inserção da DHTRJ.

O embaralhamento do passo 2 faz com que todos os blocos tenham aproximadamente a mesma quantidade de pixels de baixa visibilidade. Sem este embaralhamento, o algoritmo de inserção/extração teria que distinguir símbolos pequenos (como pontos, vírgulas, etc.), que não podem carregar muitos bits sem ter seu aspecto visual prejudicado, de símbolos grandes. O embaralhamento também oferece proteção adicional contra os ataques de paridade descritos anteriormente.

Algoritmo 22 - Extração da DHTRJ

1) Dividir a imagem em blocos, do mesmo modo utilizado nos passos 1, 2 e 3 do processo de inserção.
2) Verificar a paridade de cada bloco, extraindo de cada um deles um bit de informação.

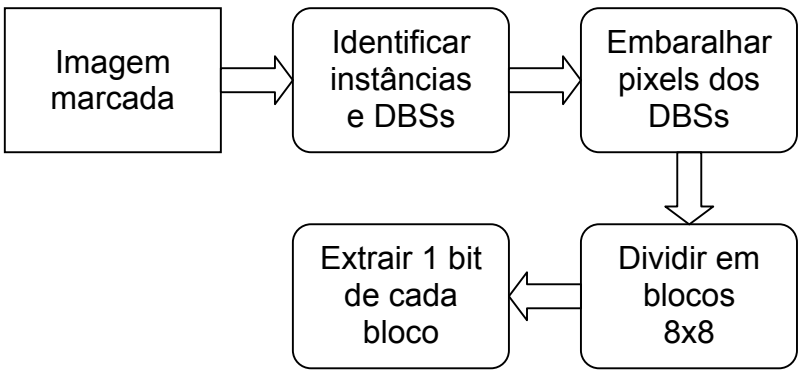


Diagrama 2 – Algoritmo de extração da DHTRJ.

(a) Original. (b) Desconexão. (c) Conexão.

Figura 22 – Conexão e desconexão de símbolos. Um símbolo pode se separar ou dois símbolos podem se juntar quando a informação é inserida (PAMBOUKIAN; KIM, 2005).

A seguir, serão descritos alguns cuidados que devem ser tomados na implementação da DHTRJ.

1) A seqüência de símbolos dentro da região de texto deve ser determinada usando um método específico (por exemplo, ordem *raster*, ou seja, da esquerda para a direita, de cima para baixo). A inserção de dados não pode modificar esta ordem; caso contrário a informação não pode ser extraída corretamente da imagem decodificada. Usando a ordem *raster*, a posição da primeira linha de cada símbolo deve permanecer inalterada.

2) O ranqueamento de modelos previamente descrito pode ser utilizado para determinar o pixel com a menor significância visual. Porém, na DHTRJ, o conjunto original de modelos (Figura 12) deve ser modificado de maneira a eliminar modelos que causem a conexão ou desconexão de símbolos. Considerando uma região de texto com fundo branco e letra preta, se um pixel preto se transforma em um branco, uma desconexão pode ocorrer separando o símbolo em duas partes (Figura 22b). Por outro lado, se um pixel branco se transforma em um preto, dois símbolos podem ser unidos (Figura 22c). A conexão/desconexão provoca uma alteração na seqüência de símbolos, de forma que os DBSs são identificados de maneira errada no processo de extração. O conjunto de modelos que não causam conexão ou desconexão pode ser vistos na Figura 23.

A técnica que acaba de ser descrita supõe um arquivo JBIG2 com uma única região de texto e um único dicionário de símbolos. A adaptação desta técnica para arquivos com mais de uma região de texto ou mais de um dicionário é semelhante.

Figura 23 – Conjunto de modelos 3×3 que não causam conexão ou desconexão de símbolos. Espelhos, rotações e reversos de cada padrão possuem a mesma nota (PAMBOUKIAN; KIM, 2005).

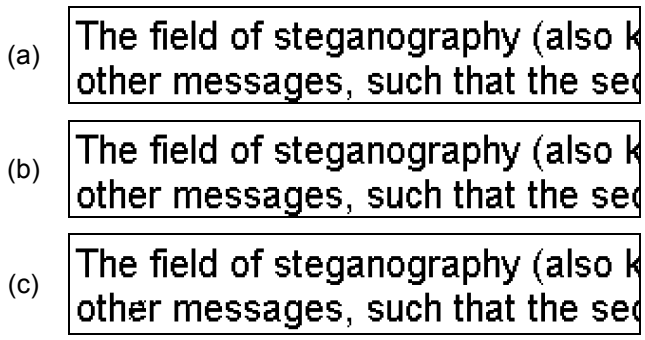


Figura 24 – Aplicação da técnica DHTRJ. (a) Parte da imagem original. (b) Imagem modificada com a inserção de 1 bit/símbolo. (c) Imagem modificada com a inserção de 12 bits/símbolos em média (PAMBOUKIAN; KIM; DE QUEIROZ, 2005).

DHTRJ pode ser aplicada a arquivos comprimidos com ou sem perda de informação, porque as informações inseridas no dicionário de símbolos sempre são comprimidas sem perda. Nesta técnica, a informação escondida pode ser extraída diretamente do arquivo JBIG2 ou também através da imagem binária gerada pela decodificação da região de texto do arquivo JBIG2.

3.4.1 **Resultados experimentais da técnica DHTRJ**

A análise do impacto psico-visual das imagens marcadas pela DHTRJ está além do escopo deste trabalho. Porém, pôde-se verificar que diversas imagens binárias geradas por software ou escaneadas em várias resoluções (81 a 600 dpi) foram marcadas por esta técnica, resultando em imagens com uma qualidade visual muito agradável. A imagem original, da qual se pode ver uma parte na Figura 24a, possui 942×306 pixels, resolução de 81×81 dpi e 1025 instâncias de símbolos. Nesta imagem foram inseridos 128 bits de informação, armazenados em 128 DBSs, resultando na imagem vista parcialmente na Figura 24b, que possui uma excelente qualidade visual. Armazenando a informação em apenas 11 símbolos, a degradação visual se torna bastante perceptível, como na letra “*e*” da palavra “*other*” (Figura 24c). Esta degradação é devida à baixa resolução da imagem original. Provavelmente, a mesma imagem em uma resolução típica (300 dpi, por exemplo) poderia armazenar a informação em apenas 11 símbolos sem que sua qualidade visual fosse prejudicada. Portanto, deve ser escolhida uma quantidade de bits/símbolo que não afete a qualidade visual da imagem hospedeira.

3.5 *AUTHENTICATION WATERMARKING BY TEMPLATE RANKING FOR JBIG2* (AWTRJ)

DHTRJ pode ser convertida em uma AWT segura chamada AWTRJ (*Authentication Watermarking by Template Ranking for JBIG2* - marca d'água de autenticação pelo ranqueamento de modelos para JBIG2). AWTRJ pode ser utilizada para autenticar qualquer imagem JBIG2 que possua uma região de texto grande o suficiente para armazenar um Código de Autenticação de Mensagem (MAC – *Message Authentication Code*). Geralmente, um MAC de comprimento 128 bits é considerado seguro.

Usando AWTRJ, apenas o proprietário da chave secreta pode inserir uma marca válida. Esta técnica seleciona de maneira pseudo-aleatória (usando a chave secreta como semente) uma quantidade de instâncias de símbolos da região de texto para armazenar o MAC e identifica os correspondentes DBSs. As instâncias selecionadas são removidas da imagem, eliminando-se suas referências, e a imagem resultante (que inclui não apenas a região de texto, mas também as regiões de meio-tom e genérica) e a chave secreta são utilizadas para computar o MAC da imagem. Então o MAC é inserido nos DBSs através da técnica DHTRJ, usando a chave secreta como semente para o embaralhamento pseudo-aleatório dos pixels dos DBSs. Em seguida, as referências aos DBSs na região de texto são restauradas.

Algoritmo 23 - Inserção da AWTRJ

1) Selecionar de forma pseudo-aleatória, usando a chave secreta como semente, uma quantidade apropriada de instâncias de símbolos da região de texto para armazenar os dados e identificar os DBSs correspondentes.
2) Remover as instâncias de símbolos da imagem e computar o MAC da imagem resultante (que inclui regiões de meio-tom e genéricas, além da região de texto) usando a chave secreta.
3) Como cada símbolo do dicionário pode ser referenciado por várias instâncias na região de texto, uma alteração de um símbolo terá seu efeito multiplicado. Para evitar este problema, duplicar no dicionário (*symbol dictionary segment*) os símbolos que serão utilizados para armazenar dados e modificar as

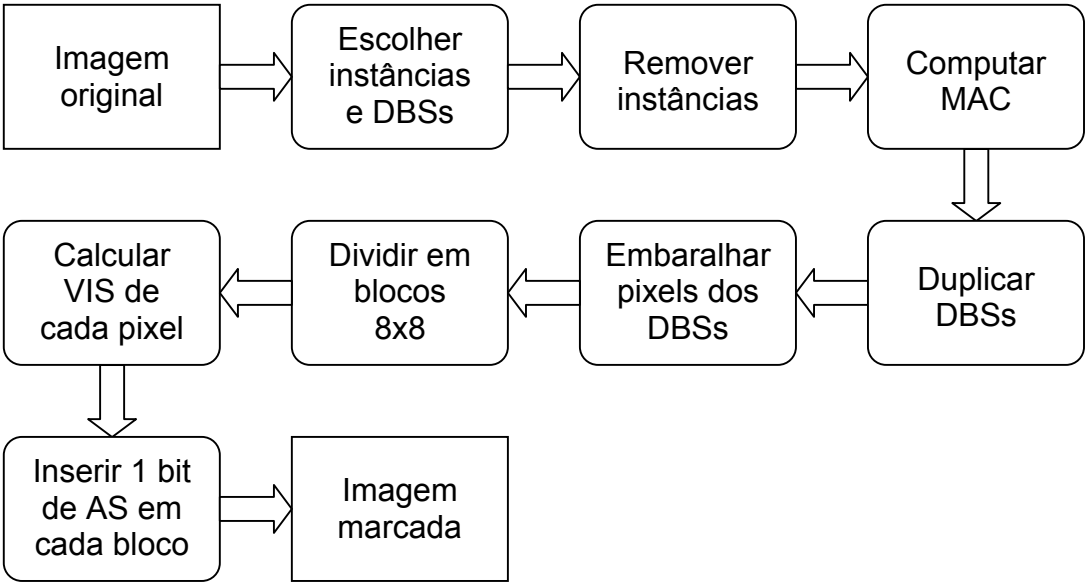


Diagrama 3 – Algoritmo de inserção da AWTRJ.

referências na região de texto (*text region segment*) de modo que apenas uma instância (selecionada de forma pseudo-aleatória) se refira ao símbolo que irá carregar informação (DBS). Todas as outras instâncias devem continuar se referindo ao símbolo original.

4) Embaralhar de maneira pseudo-aleatória o conjunto de todos os pixels de todos os DBSs do dicionário de símbolos.

5) Dividir o conjunto de pixels embaralhados dos DBSs em pequenos blocos (por exemplo, cada bloco com 64 pixels).

6) Analisar a vizinhança (geralmente 3×3) de cada pixel embaralhado para verificar sua significância visual.

7) Inserir um bit do MAC em cada bloco forçando-o a ter uma quantidade par ou impar de pixels pretos. Apenas símbolos que não conectam nem desconectam símbolos devem ser alterados (Figura 22).

Algoritmo 24 - Verificação da AWTRJ

1) Detectar as instâncias de símbolos que foram marcadas e seus respectivos DBSs utilizando o mesmo gerador de números pseudo-aleatórios, tendo a chave secreta como semente, como no processo de inserção.

2) Extrair o MAC original *S* desses símbolos.

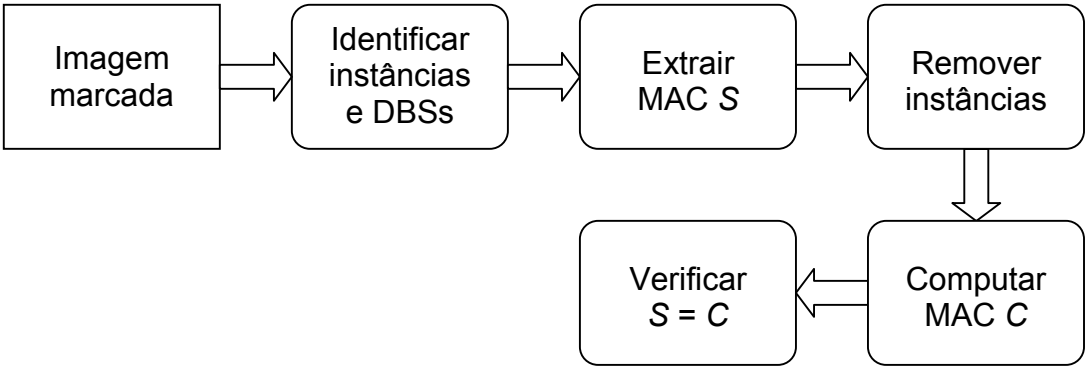


Diagrama 4 – Algoritmo de verificação da AWTRJ.

3) Remover da imagem as instâncias de símbolos referentes aos DBSs e computar o MAC de verificação *C* da imagem resultante (que inclui regiões de meio-tom e genéricas, além da região de texto) usando a chave secreta.

4) Se o MAC extraído *S* for igual ao MAC de verificação *C*, a marca está verificada. Caso contrário, a imagem foi modificada.

A versão da AWTRJ de chave secreta é protegida contra ataques de paridade porque utiliza a chave secreta como semente para a seleção pseudo-aleatória no passo 1, e também usa a chave secreta como semente para o embaralhamento pseudo-aleatório no passo 4. Infelizmente, a mesma idéia não pode ser aplicada a versão de chave pública, porque qualquer pessoa pode extrair completamente os bits escondidos sem o conhecimento da chave privada, pois é utilizada uma semente pública.

Nesta técnica, a marca d'água pode ser extraída diretamente do arquivo JBIG2 ou também através da imagem binária gerada pela decodificação da região de texto do arquivo JBIG2.

3.5.1 **Ataque de Paridade**

Da mesma forma que AWTR, a técnica AWTRJ pode detectar qualquer alteração feita na porção da imagem que é utilizada para calcular o MAC, mesmo se um único pixel for modificado. A probabilidade de não detectar uma modificação nesta porção

da imagem é de apenas 2^{-n} (onde *n* é o comprimento do MAC), a qual pode ser negligenciada.

Infelizmente, algumas alterações feitas nos DBSs não podem ser detectadas. Suponha, por exemplo, que apenas um bit é inserido em cada símbolo. Neste caso, os DBSs podem ser identificados facilmente (por exemplo, um DBS “a” é ligeiramente diferente do “a” padrão”). Qualquer DBS pode ser modificado, sem ser detectado pelo algoritmo de verificação da marca, se a paridade dos pixels pretos não for alterada. Um DBS “a” pode ser transformado em qualquer outra letra sem ser notado se a paridade não for modificada. Esta técnica de adulteração, conhecida como ataque de paridade (*parity attack*), já foi discutida anteriormente e pode ser vista em Kim e de Queiroz (2004a, 2004b).

Felizmente, o embaralhamento pseudo-aleatório de todos os pixels dos DBSs, usando a chave secreta como semente, reduz a possibilidade do ataque de paridade, pois o invasor não sabe como os DBSs são divididos em blocos.

Como foi visto anteriormente, uma modificação na técnica descrita em (KIM; DE QUEIROZ, 2004a, 2004b) reduz ainda mais a possibilidade de um ataque de paridade. Depois de inserir um bit do MAC em um bloco, um novo MAC é computado, alimentando a função que calcula o MAC com o MAC anterior e o símbolo modificado. Desta forma, o ultimo DBS (*n*-ésimo DBS) pode ainda sofrer um ataque sem ser detectado. Porém, se o penúltimo DBS é modificado mantendo sua paridade, a modificação pode ser detectada com 50% de chance. Se o primeiro DBS é modificado (mantendo sua paridade), existe a probabilidade de $1–2^{–(n–1)}$ de detecção desta mudança. A chance de detecção é maior quando os primeiros DBSs são modificados.

3.5.2 **Resultados experimentais da técnica AWTRJ**

Nos testes realizados com esta técnica, qualquer alteração visual significante (até mesmo um único pixel) na imagem marcada pôde ser detectada. As imagens marcadas apresentaram excelente qualidade visual, como se pode ver na Figura 24b, que mostra parte de uma imagem onde foi inserida uma AS de 128 bits.

3.6 *DATA-HIDING BY TEMPLATE RANKING WITH SYMMETRICAL CENTRAL PIXELS FOR JBIG2 TEXT REGION* (DHCJT)

A técnica DHCJT (*Data Hiding by template ranking with symmetrical Central pixels for JBIG2 Text region* - esteganografia pelo ranqueamento de modelos com pixels centrais simétricos para a região de texto do JBIG2), baseada na DHTC, também utiliza a região de texto do arquivo JBIG2 para a inserção da marca d'água, porém, ao contrário da DHTRJ, é totalmente imune a ataques de paridade. A imagem JBIG2 a ser marcada pode estar codificada com ou sem perda. Além disso, a qualidade visual da imagem marcada é excelente, sem ruídos do tipo sal-e-pimenta. A verificação da imagem pode ser feita tanto no arquivo JBIG2 quanto na imagem binária obtida pela decodificação da região de texto do arquivo JBIG2.

Algoritmo 25 - Inserção da DHCJT

1) Dada uma imagem *Z'* codificada no formato JBIG2 e *n* bits de dados a serem inseridos em *Z'*, decodificar a região de texto de *Z'*, obtendo a imagem binária não comprimida *Z*.
2) Dividir *Z* em uma seqüência *v* de blocos não sobrepostos e ordenar *v* como na técnica DHTC.
3) Identificar na região de texto (*text region segment*) as instâncias de símbolos que contêm os *n* primeiros pixels centrais da seqüência ordenada *v* e suas referências aos DBSs no dicionário de símbolos (*symbol dictionary segment*). Note que a quantidade de DBSs pode, em alguns casos, ser menor do que *n*, porque cada símbolo pode carregar mais de um bit de AS.
4) Nesta técnica, ao contrário da DHTRJ, é proposto um esquema para minimizar a duplicação de símbolos. Para isto, deve-se verificar quantas vezes cada DBS é referenciado na região de texto:
    - Se houver apenas uma referência, a informação pode ser armazenada no símbolo original. Para economizar espaço no dicionário, pode-se verificar se o símbolo gerado após a inserção da informação já existe no dicionário. Se existir, basta referenciar o símbolo que já existe e remover o símbolo original do dicionário.

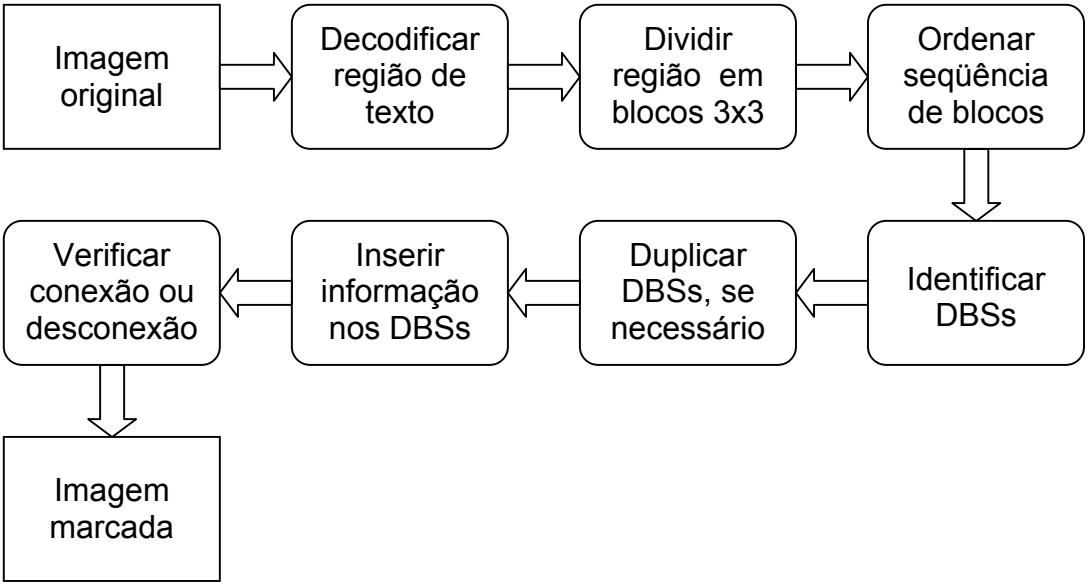


Diagrama 5 – Algoritmo de inserção da DHCJT.

- Se houver mais de uma referência, o símbolo deve ser duplicado e inserido no final do dicionário. A informação deve ser armazenada no símbolo duplicado. Neste caso, as referências dos símbolos também devem ser alteradas de modo que apenas uma instância faça referência ao símbolo duplicado, enquanto as demais continuam fazendo referência ao símbolo original. Também para economizar espaço no dicionário, pode-se verificar se o símbolo gerado após a inserção da informação já existe no dicionário. Se existir, basta referenciar o símbolo que já existe ao invés de duplicá-lo.

5) Inserir *n* bits de informação nos DBSs invertendo, se necessário, a cor dos *n* primeiros pixels centrais da seqüência ordenada *v*.

6) Verificar a possibilidade de conexão ou desconexão dos DBSs. Considerando uma região de texto com fundo branco e letra preta, se um pixel preto se transforma em um branco, uma desconexão pode ocorrer separando o símbolo em duas partes (Figura 22b). Por outro lado, se um pixel branco se transforma em um preto, dois símbolos podem ser unidos (Figura 22c). No primeiro caso, o DBS original deve ser eliminado do dicionário e os novos símbolos devem ser inseridos. No segundo caso, os DBSs originais devem

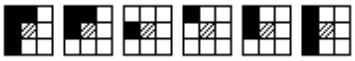


Figura 25 – Ranqueamento de modelos 3×3 projetado para ser usado no DHCJT em ordem crescente de impacto visual. Apenas modelos que não causam conexão ou desconexão de símbolos fazem parte do conjunto. Pixels hachurados podem ser pretos ou brancos (note que todos os pixels centrais são hachurados). A nota de um determinado padrão é aquele que combina com o modelo de menor nota. Espelhos, rotações e reversos de cada padrão possuem a mesma nota (PAMBOUKIAN; KIM, 2005).

ser eliminados do dicionário e o novo símbolo deve ser inserido. Em ambos os casos a(s) referência(s) na região de texto deve(m) ser modificada(s).

Para simplificar a implementação do algoritmo, é sugerida a utilização de um novo ranqueamento de modelos (Figura 25) ao invés do modelo original utilizado pela técnica DHTC (Figura 15). O novo conjunto contém apenas modelos que não causam a conexão ou desconexão de símbolos. Usando o novo conjunto de modelos, o último passo do algoritmo de inserção do DHCJT (que tem uma implementação trabalhosa) pode ser ignorado.

Quando se utiliza este novo conjunto de modelos, uma observação importante deve ser feita: todos os possíveis blocos 3×3 tem um modelo correspondente no conjunto original de modelos (Figura 15). Ao contrário, no novo conjunto de modelos, existem alguns blocos 3×3 que não possuem um correspondente. Isto significa que podem existir algumas imagens muito pequenas que podem armazenar uma certa quantidade de bits utilizando o conjunto antigo (provavelmente invertendo alguns pixels de alta visibilidade), mas não podem armazenar a mesma quantidade usando o novo conjunto.

Algoritmo 26 - Extração da DHCJT

1) Dada uma imagem *Z'* codificada no formato JBIG2 com *n* bits de informação inseridos pela DHCJT, decodificar a região de texto de *Z'*, obtendo a imagem binária não comprimida *Z*.
2) Dividir a imagem binária *Z* em uma seqüência *v* de blocos não sobrepostos e ordenar *v* como na inserção.
3) Extrair a informação escondida dos *n* primeiros pixels centrais da seqüência ordenada *v*.

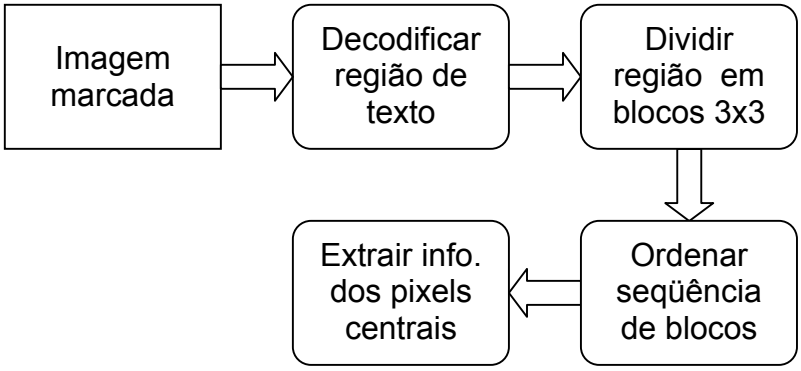


Diagrama 6 – Algoritmo de extração da DHCJT.

3.6.1 **Resultados experimentais da técnica DHCJT**

Não foi aplicada qualquer medida de distorção perceptual para quantificar a qualidade da imagem marcada, porque esta análise está além do escopo deste trabalho, mas pode-se perceber que a qualidade visual das imagens hospedeiras geradas por esta técnica é excelente. Se necessário, o ranqueamento de modelos sugerido pode ser modificado para minimizar a distorção de acordo com um modelo perceptual específico.

Esta técnica foi testada em diversos tipos de imagens binárias geradas por software e escaneadas em diferentes resoluções. As imagens hospedeiras, mesmo as de tamanho reduzido e baixa resolução, apresentaram excelente qualidade visual.

A imagem mostrada na Figura 26 possui 626×240 pixels, 93 instâncias de símbolos e foi escaneada a 300×300 dpi. Nesta imagem foram inseridos 128 bits de informação, armazenados em 61 DBSs. Para melhor visualização, a Figura 26 apresenta apenas uma parte da imagem original, de forma ampliada.

(a) Parte da imagem original. (b) Parte da imagem marcada. (c) Pixels modificados.

Figura 26 – Ampliação de uma imagem escaneada a 300 dpi, com 626×240 pixels, 93 instâncias de símbolos e 128 bits de informação inseridos pela DHCJT. (a) Parte da imagem original; (b) Imagem marcada; (c) Pixels modificados sobrepostos à imagem original (ampliação).

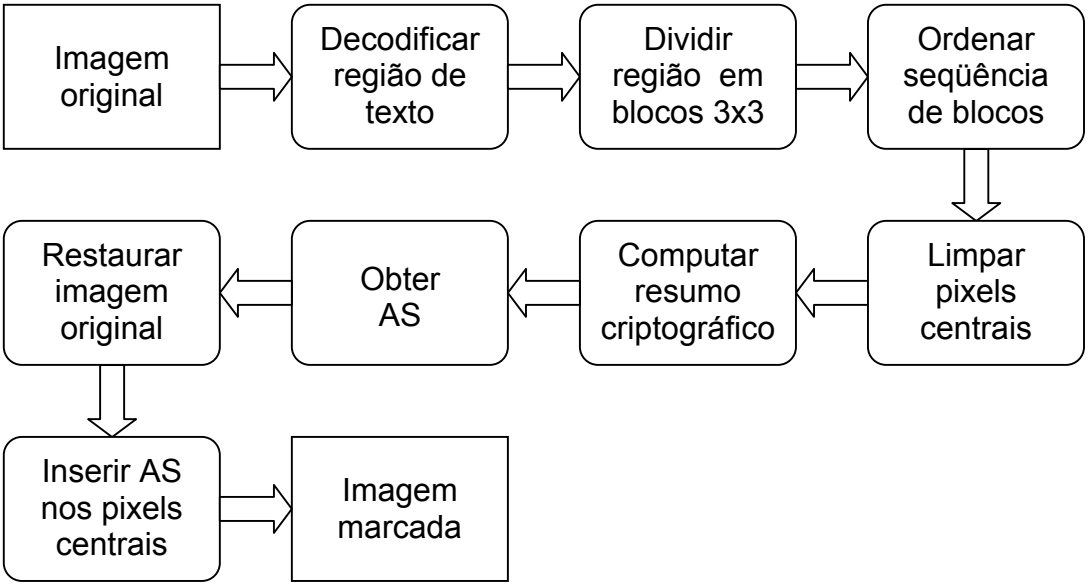


Diagrama 7 – Algoritmo de inserção da AWCJT.

3.7 *AUTHENTICATION WATERMARKING BY TEMPLATE RANKING WITH SYMMETRICAL CENTRAL PIXELS FOR JBIG2 TEXT REGION* (AWCJT)

A técnica AWCJT (*Authentication Watermarking by template ranking with symmetrical Central pixels for JBIG2 Text region* - marca d'água de autenticação pelo ranqueamento de modelos com pixels centrais simétricos para a região de texto do JBIG2), derivada da DHCJT, tanto em sua versão de chave secreta quanto na versão de chave pública, é totalmente imune a ataques de paridade. Esta técnica pode ser utilizada para proteger qualquer arquivo JBIG2 (codificado com ou sem perda) que possua uma região de texto grande o suficiente para armazenar a AS.

Algoritmo 27 - Inserção da AWCJT

1) Dada uma imagem *Z'* codificada no formato JBIG2, decodificar a região de texto de *Z'*, obtendo a imagem binária não comprimida *Z*.
2) Dividir *Z* em uma seqüência *v* de blocos não sobrepostos e ordenar *v* como na técnica DHCJT.
3) Limpar os pixels centrais dos *n* primeiros blocos da seqüência ordenada *v*, onde *n* é o comprimento da AS adotada.

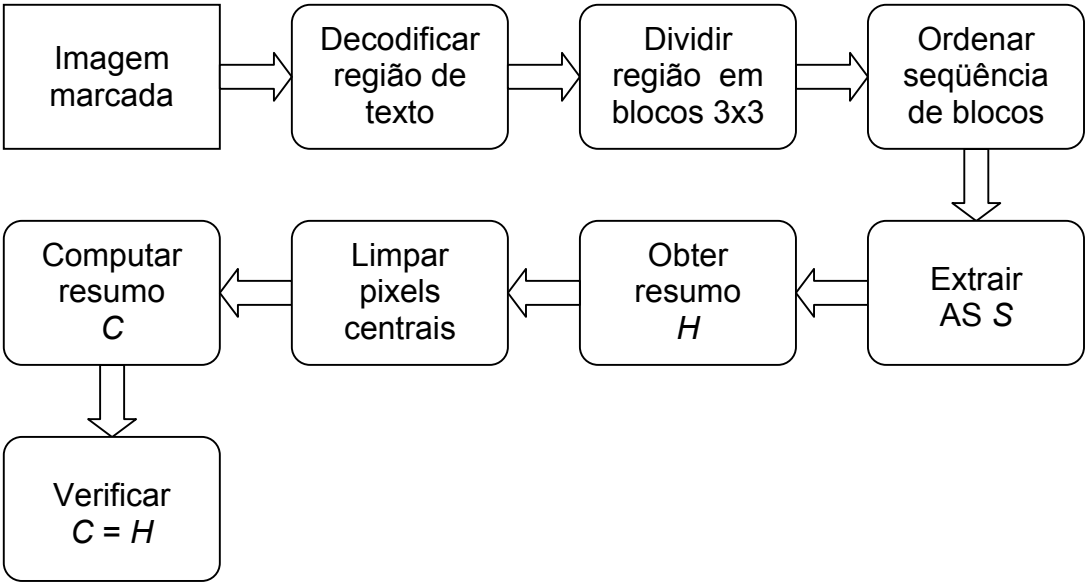


Diagrama 8 – Algoritmo de verificação da AWCJT.

4) Usando uma função de *hash* criptograficamente segura, computar o resumo criptográfico *H* da imagem *Z* limpa. Além da imagem *Z*, todas as outras regiões de *Z'* a serem protegidas (meio-tom e genérica) devem ser levadas em conta no cômputo de *H*.
5) Criptografar o resumo *H* com a chave secreta ou privada, obtendo a AS *S*.
6) Restaurar a imagem *Z* original.
7) Inserir *n* bits de *S* nos DBSs como explanado na DHCJT, obtendo a imagem marcada.

Algoritmo 28 - Verificação da AWCJT

1) Dada uma imagem *Z'* codificada no formato JBIG2 e marcada pela AWCJT, decodificar a região de texto de *Z'*, obtendo a imagem binária não comprimida *Z*.
2) Dividir a imagem binária *Z* em uma seqüência *v* de blocos não sobrepostos e ordenar *v* como na inserção.
3) Extrair a AS *S* dos *n* primeiros pixels centrais da seqüência ordenada *v*.
4) Decriptografar S com a chave secreta ou pública, obtendo o resumo criptográfico extraído H.

5) Limpar os pixels centrais dos *n* primeiros blocos da seqüência ordenada *v*.
6) Computar o resumo criptográfico de verificação *C* da imagem *Z* limpa, usando a mesma função de *hash* utilizada na inserção. Além da imagem *Z*, todas as outras regiões de *Z'* protegidas (meio-tom e genérica) devem ser levadas em conta no cômputo de *C*.
7) Se o resumo extraído *H* e o resumo computado *C* forem iguais, a marca d'água está verificada, indicando que a imagem não foi corrompida. Caso contrário, a imagem foi modificada.

Ataques de paridade não se aplicam ao AWCJT, pois a quantidade de pixels que armazenam informação é exatamente igual ao comprimento da AS adotada. Todos os pixels da imagem (com exceção daqueles utilizados para armazenar a AS) são usados no cômputo da AS. Desta forma, qualquer alteração nos pixels utilizados para o cômputo da AS será detectada, pois a mudança irá gerar uma AS diferente da armazenada, e qualquer alteração nos pixels utilizados para armazenar a informação também poderá ser detectada, pois irá alterar a AS armazenada.

3.7.1 **Resultados experimentais da técnica AWCJT**

Nos testes realizados com esta técnica, qualquer alteração visual significante (até mesmo um único pixel) na imagem marcada pôde ser detectada. As imagens marcadas apresentaram excelente qualidade visual, como se pode ver na Figura 26, onde foi inserida uma AS de 128 bits.

3.8 *AUTHENTICATION WATERMARKING BY TEMPLATE RANKING WITH SYMMETRICAL CENTRAL PIXELS FOR JBIG2 HALFTONE REGION* (AWCJH)

A técnica AWCJH (*Authentication Watermarking by template ranking with symmetrical Central pixels for JBIG2 Halftone region* - marca d'água de autenticação pelo ranqueamento de modelos com pixels centrais simétricos para a região de

meio-tom do JBIG2) é uma variação da AWCJT. A diferença é que a informação é inserida na região de meio-tom ao invés da região de texto. AWCJT pode ser utilizada para inserir informações em arquivos JBIG2 que não possuem uma região de texto, consistindo basicamente em imagens meio-tom, esboços (*sketches*), esquemas e caricaturas (*cartoons*). AWCJH pode ser utilizada para proteger qualquer arquivo JBIG2 (com ou sem perda) que possua uma região de meio-tom de tamanho suficiente para abrigar a AS.

Ambas as versões (de chave secreta e de chave pública) desta técnica são seguras, pois sua segurança depende apenas do segredo da chave e são imunes a ataques de paridade. As informações escondidas podem ser extraídas tanto do arquivo JBIG2 quanto da imagem binária obtida pela decodificação da região de meio-tom. As imagens marcadas por esta técnica apresentam excelente qualidade visual, sem ruídos visíveis do tipo sal-e-pimenta, pois apenas pixels de baixa visibilidade são alterados.

Embora o algoritmo de inserção da AWCJH seja muito parecido com o algoritmo de inserção da AWCJT, o mesmo será descrito a seguir, para maior clareza.

Algoritmo 29 - Inserção da AWCJH

1) Seja *Z'* uma imagem codificada no formato JBIG2 a ser marcada. Decodificar a região de meio-tom de *Z'*, obtendo a imagem binária não comprimida *Z*.
2) Dividir *Z* em uma seqüência *v* de blocos não sobrepostos e ordenar *v* como descrito na técnica AWTC.
3) Limpar os *n* primeiros pixels centrais da seqüência ordenada *v*, onde *n* é o comprimento da AS adotada.
4) Usando uma função de *hash* criptograficamente segura, computar o resumo criptográfico *H* da imagem *Z* limpa. Além da região de meio-tom *Z*, todas as demais regiões de *Z'* serem protegidas (regiões de texto e genéricas) devem ser levadas em conta no cômputo de *H*.
5) Criptografar o resumo *H* com a chave secreta ou privada, obtendo a AS *S*.
6) Restaurar a imagem *Z* original.

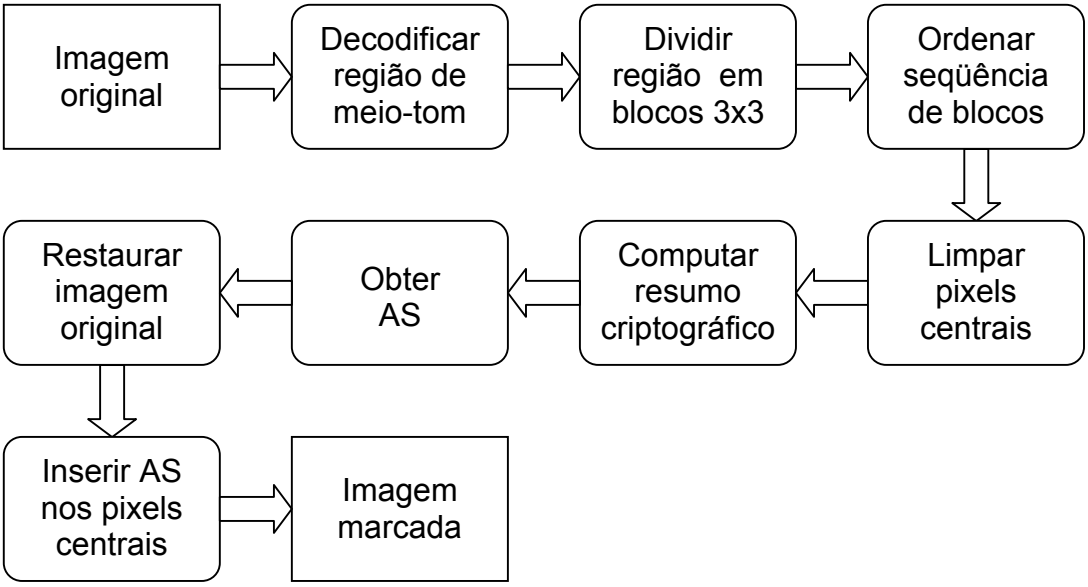


Diagrama 9 – Algoritmo de inserção da AWCJH.

7) Identificar na região de meio-tom (*halftone region segment*) as referências de padrões que contêm um ou mais pixels correspondentes aos *n* primeiros pixels centrais da seqüência ordenada *v*. Identificar também os padrões correspondentes referenciados no dicionário de padrões (*pattern dictionary segment*). Para cada referência a padrão *r*, o correspondente padrão *p* do dicionário deve ser transformado em um outro padrão *q* para esconder a AS *S*. Para isto deve-se verificar: (a) Existem mais referências a *p* na região de meio-tom, além de *r*? (b) O padrão *q* já está presente no dicionário de padrões? Existem quatro respostas possíveis:

   a) Apenas *r* se refere a *p* e *q* já está presente no dicionário. Neste caso, deve-se deletar *p* do dicionário e fazer com que *r* se refira a *q* no dicionário.

   b) Apenas *r* se refere a *p* e *q* não está presente no dicionário. Modificar os bits de *p* para convertê-lo em *q*.

   c) Existem mais referências a *p* além de *r* e *q* já está presente no dicionário. Fazer com que *r* se refira a *q* no dicionário.

   d) Existem mais referências a *p* além de *r* e *q* não está presente no dicionário. Inserir *q* no final do dicionário de padrões e fazer com que *r* se refira ao recém-inserido padrão *q*.

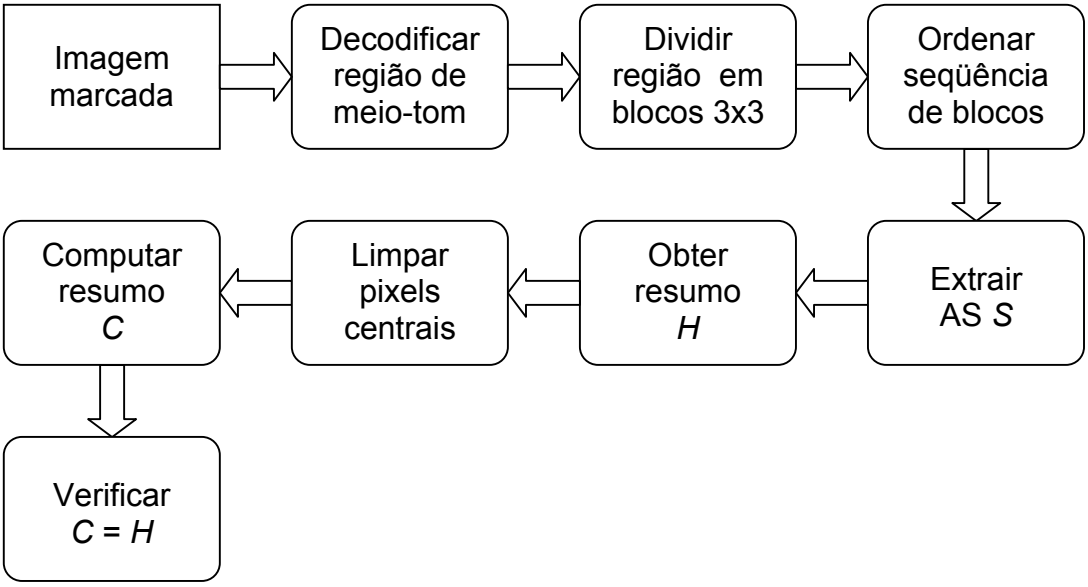
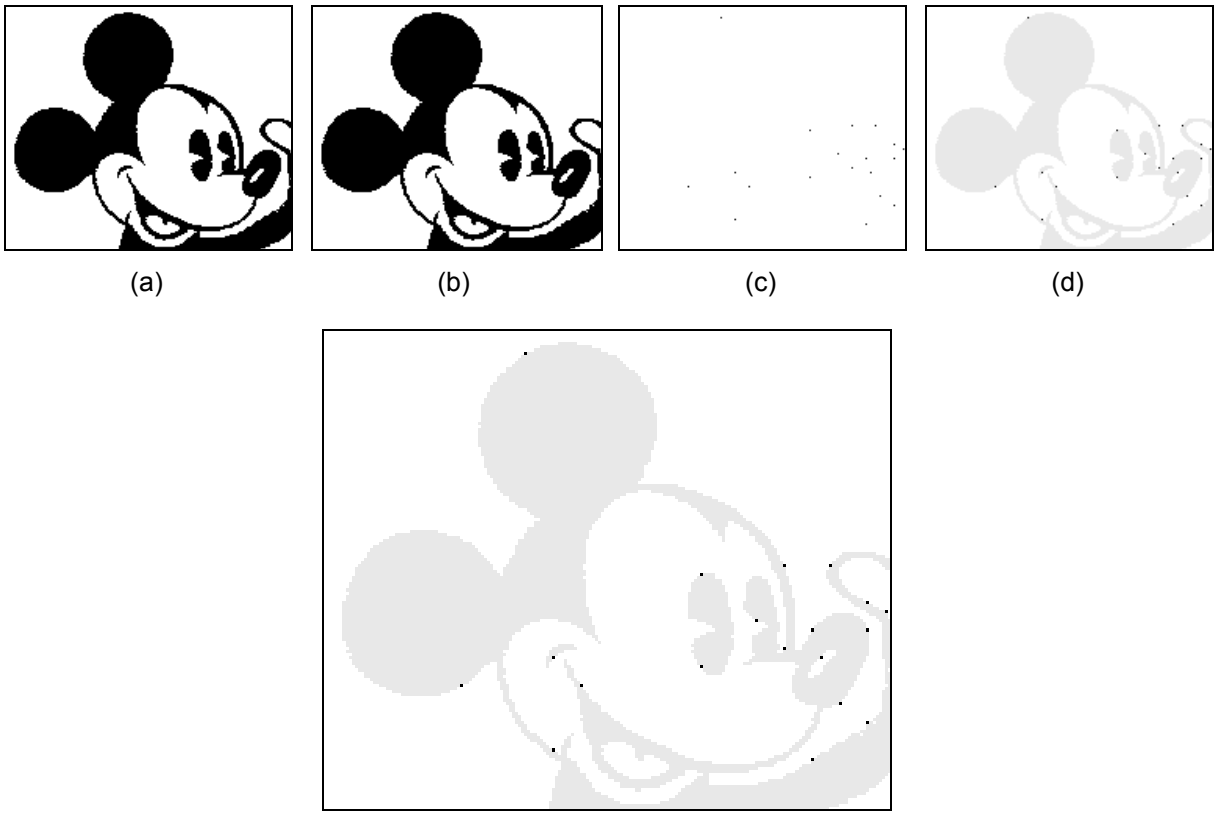


Diagrama 10 – Algoritmo de verificação da AWCJH.

Como a região de meio-tom é formada por padrões (blocos retangulares) justapostos, não há necessidade de preocupar-se com a conexão ou desconexão de padrões como no AWCJT. Portanto, o ranqueamento de modelos utilizado para a ordenação da seqüência *v* pode ser o ranqueamento original da técnica DHTC (Figura 15).

Algoritmo 30 - Verificação da AWCJH

1) Seja *Z'* uma imagem codificada em JBIG2 e marcada pela AWCJH. Decodificar a região de meio-tom de *Z'*, obtendo a imagem binária não comprimida *Z*.
2) Dividir *Z* em uma seqüência *v* de blocos não sobrepostos e ordenar *v* como no processo de inserção.
3) Extrair a AS *S* dos *n* primeiros pixels centrais da seqüência ordenada *v*.
4) Decriptografar *S* usando a chave secreta ou pública, obtendo o resumo criptográfico extraído *H*.
5) Limpar os *n* primeiros pixels centrais da seqüência ordenada *v*.
6) Computar o resumo criptográfico de verificação *C* da imagem *Z* limpa, usando a mesma função de *hash* da inserção. Além da imagem *Z*, todas as demais regiões protegidas de *Z* devem ser levadas em conta no cômputo de *C*.



(a) (b) (c) (d)

(e)

Figura 27 – (a) Parte (65×55 pixels) de uma imagem de tamanho 295×331 pixels com resolução 72×72 dpi codificada como uma região de meio-tom no padrão JBIG2 usando um dicionário de padrões 8×8. (b) Imagem marcada pela AWCJH com uma AS de comprimento 128 bits, armazenada em 111 padrões. (c) Pixels modificados. (d) Pixels modificados sobrepostos à imagem original. (e) Imagem sobreposta ampliada.

7) Se o resumo extraído *H* e o resumo de verificação *C* forem iguais, a marca está verificada, indicando que a imagem não foi corrompida. Caso contrário, a imagem foi modificada.

3.8.1 **Resultados experimentais da técnica AWCJH**

AWCJH foi aplicada em diversos tipos de imagens binárias: caricaturas (*cartoons*), esboços e esquemas. As imagens marcadas apresentaram excelente qualidade visual, mesmo quando uma imagem de dimensões reduzidas foi utilizada. O exemplo visto na Figura 27, apesar de não ser realmente uma imagem meio-tom, foi codificada como uma região de meio-tom do JBIG2 e marcada pela AWCJH.

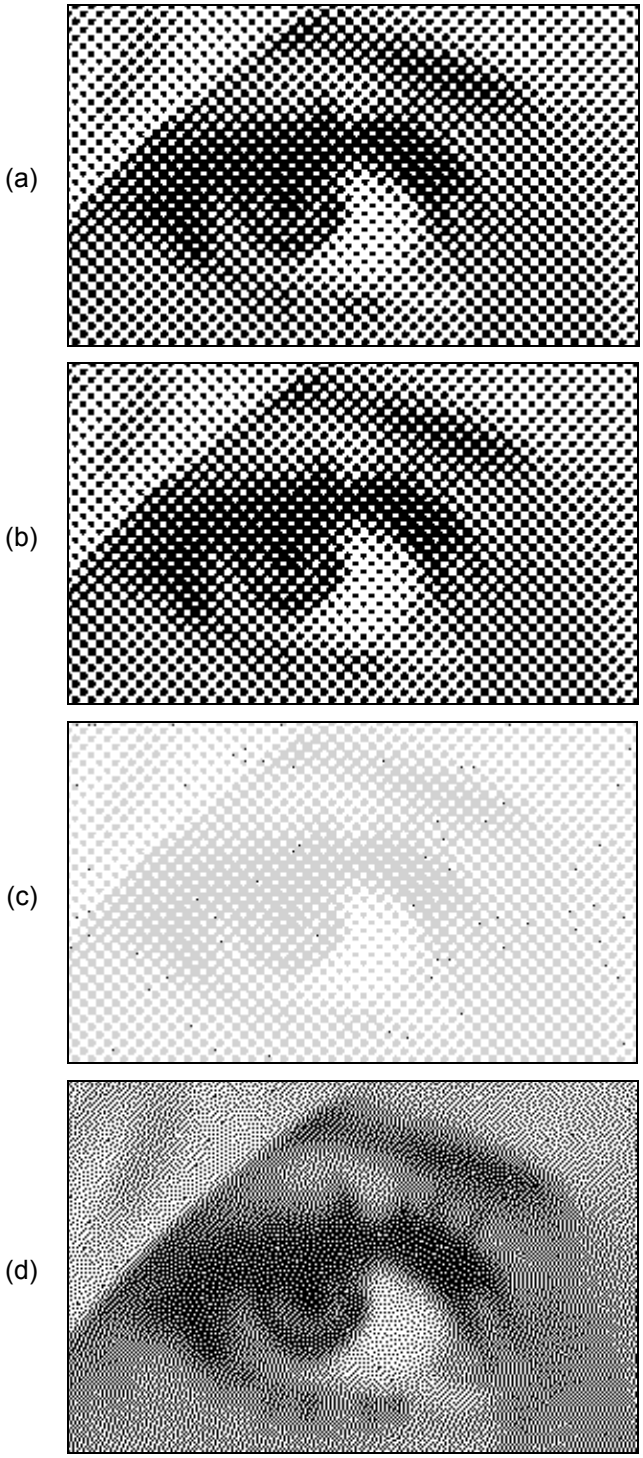


Figura 28 – (a) Imagem meio-tom com pontos aglutinados (gerada pelo *driver* da impressora HP LaserJet) de tamanho 280×170 pixels codificada como uma região de meio-tom no padrão JBIG2 usando um dicionário de padrões 8×8. (b) Imagem marcada pela AWCJH com uma AS de 128 bits, armazenada em 117 padrões. (c) Pixels modificados sobrepostos à imagem original. (d) Imagem meio-tom gerada por difusão de erro e marcada pela AWCJH.

AWCJH também foi aplicada a imagens meio-tom escaneadas e geradas por software utilizando diferentes técnicas de meio-tom em diferentes resoluções (Figura 28). AWCJH se mostrou muito boa para imagens de meio-tom com pontos aglutinados (*clustered-dot halftone*), como pode ser visto na Figura 28b. Porém, algum ruído pode ser percebido em imagens meio-tom com pontos dispersos (por exemplo, imagens geradas por técnicas como difusão de erro (*error-diffusion*) ou excitação ordenada de pontos dispersos (*dispersed-dot ordered dithering*)), pois o conceito de "baixa visibilidade" não se aplica a este tipo de imagem, como pode ser visto na Figura 28d. Apesar disto, a segurança permanece a mesma.

3.9 *AUTHENTICATION WATERMARKING BY TEMPLATE RANKING WITH SYMMETRICAL CENTRAL PIXELS FOR JBIG2* (AWTCJ)

É possível analisar uma imagem JBIG2 como um todo e distribuir as informações que se deseja esconder por várias regiões (texto, meio-tom e genérica), com a finalidade de se obter o melhor aspecto visual da imagem hospedeira. Isto é, parte da AS pode ser inserida na região de texto, parte na região de meio-tom e parte na região genérica, por exemplo.

Esta técnica que unifica as anteriores foi chamada de AWTCJ (*Authentication Watermarking by Template ranking with symmetrical Central pixels for JBIG2* - marca d'água de autenticação pelo ranqueamento de modelos com pixels centrais simétricos para JBIG2).

Apesar dos algoritmos de inserção e verificação serem extremamente parecidos com os das técnicas AWCJT e AWTCH, os mesmos serão descritos a seguir para maior clareza.

Algoritmo 31 - Inserção da AWTCJ

1) Seja *Z'* uma imagem codificada no formato JBIG2 a ser marcada. Decodificar todas as regiões de *Z'*, obtendo a imagem binária não comprimida *Z*.
2) Dividir *Z* em uma seqüência *v* de blocos não sobrepostos e ordenar *v* como descrito na técnica AWTC.

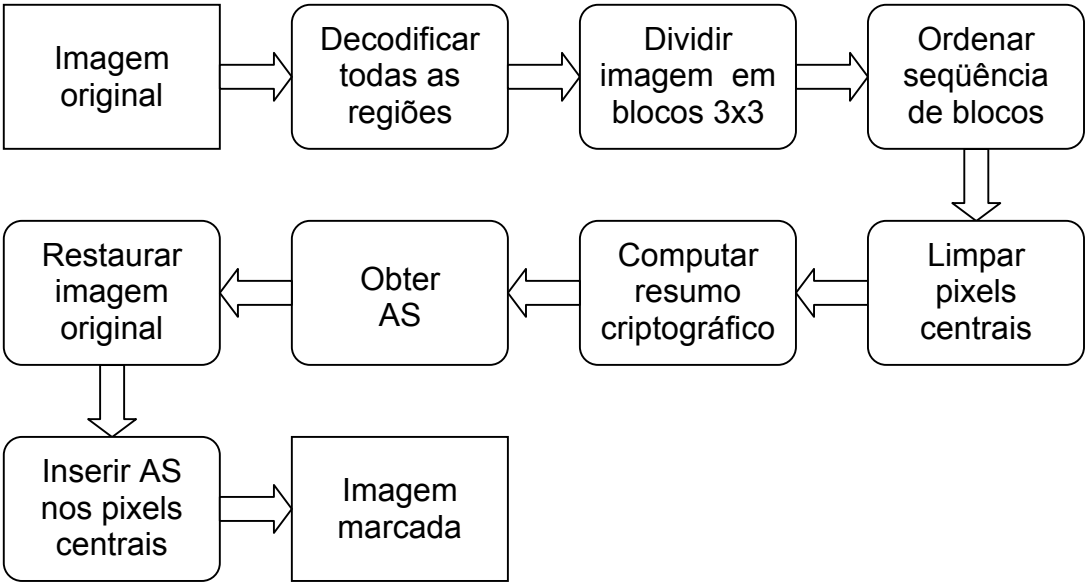


Diagrama 11 – Algoritmo de inserção da AWTCJ.

3) Limpar os *n* primeiros pixels centrais da seqüência ordenada *v*, onde *n* é o comprimento da AS adotada.

4) Usando uma função de *hash* criptograficamente segura, computar o resumo criptográfico *H* da imagem *Z* limpa.

5) Criptografar o resumo *H* com a chave secreta ou privada, obtendo a AS *S*.

6) Restaurar a imagem *Z* original.

7) Para cada um dos *n* primeiros pixels centrais da seqüência ordenada *v*, verificar a que região pertence o pixel (texto, meio-tom, genérica):

   a) região de texto – inserir um bit de AS no pixel como descrito na técnica DHCJT, alterando o respectivo símbolo no dicionário (duplicando, inserindo ou removendo símbolos e ajustando referências, se necessário);

   b) região de meio-tom – inserir um bit de AS no pixel como descrito na técnica DHCJH, alterando o respectivo padrão no dicionário (duplicando padrões e ajustando referências, se necessário);

   c) região genérica – simplesmente alterar a cor do pixel (se necessário) para armazenar um bit de AS.

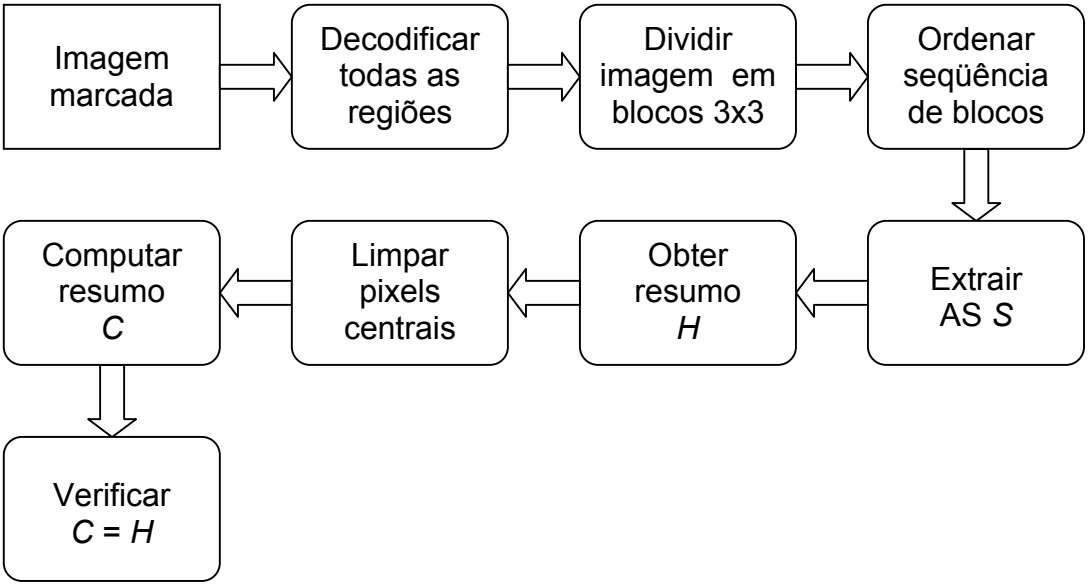


Diagrama 12 – Algoritmo de verificação da AWTCJ.

Algoritmo 32 - Verificação da AWTCJ

1) Seja *Z'* uma imagem codificada em JBIG2 e marcada pela AWTCJ. Decodificar todas as regiões de *Z'*, obtendo a imagem binária não comprimida *Z*.
2) Dividir *Z* em uma seqüência *v* de blocos não sobrepostos e ordenar *v* como no processo de inserção.
3) Extrair a AS *S* dos *n* primeiros pixels centrais da seqüência ordenada *v*.
4) Decriptografar *S* usando a chave secreta ou pública, obtendo o resumo criptográfico extraído *H*.
5) Limpar os *n* primeiros pixels centrais da seqüência ordenada *v*.
6) Computar o resumo criptográfico de verificação *C* da imagem *Z* limpa, usando a mesma função de *hash* usada na inserção.
7) Se o resumo extraído *H* e o resumo de verificação *C* forem iguais, a marca está verificada, indicando que a imagem não foi corrompida. Caso contrário, a imagem foi modificada.

FIGURA 1-1:
BLAISE
PASCAL.

Nos primórdios da humanidade, os homens utilizavam seus próprios dedos para contar e efetuar pequenos cálculos, o que deu origem ao sistema Decimal e aos termos Digital e Dígito.

Com o passar do tempo, os cálculos foram se complicando e aumentando de tamanho e surgiu a necessidade de instrumentos que pudessem auxiliar nesta tarefa. Então, por volta de 3.000 a.C. surgiram os primeiros ábacos. Estes instrumentos permitiam fazer operações simples como adição, subtração, multiplicação e divisão, porém seu manuseio era difícil e requeria certa técnica.

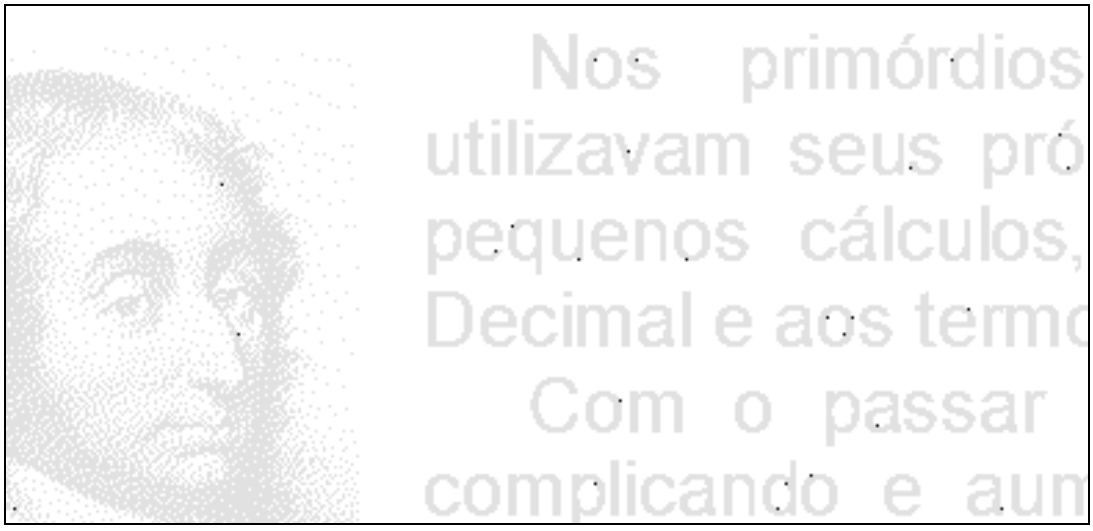
(a) Parte da imagem original.

FIGURA 1-1:
BLAISE
PASCAL.

Nos primórdios da humanidade, os homens utilizavam seus próprios dedos para contar e efetuar pequenos cálculos, o que deu origem ao sistema Decimal e aos termos Digital e Dígito.

Com o passar do tempo, os cálculos foram se complicando e aumentando de tamanho e surgiu a necessidade de instrumentos que pudessem auxiliar nesta tarefa. Então, por volta de 3.000 a.C. surgiram os primeiros ábacos. Estes instrumentos permitiam fazer operações simples como adição, subtração, multiplicação e divisão, porém seu manuseio era difícil e requeria certa técnica.

(b) Parte da imagem marcada.



(c) Pixels modificados sobrepostos ao original (imagem ampliada).

Figura 29 – Exemplo de aplicação da AWTCJ. Página de livro (ZAMBONI; PAMBOUKIAN; BARROS, 2006) com 901×1536 pixels, codificada como JBIG2 e marcada com AS de 1024 bits.

3.9.1 **Resultados experimentais da técnica AWTCJ**

A Figura 29 mostra uma imagem JBIG2 que possui uma região de texto e também uma região de meio-tom marcada com uma AS de 1024 bits. Utilizando esta técnica, qualquer alteração feita na imagem, mesmo que seja um único pixel, pode ser detectada. Além disso, as imagens marcadas apresentam excelente qualidade visual.

4 MARCAS D'ÁGUA DE AUTENTICAÇÃO REVERSÍVEIS PARA IMAGENS BINÁRIAS

4.1 INTRODUÇÃO

Neste capítulo serão analisados os principais algoritmos de compressão existentes na atualidade. Em seguida será proposta uma técnica esteganográfica reversível para imagens binárias, que será utilizada para criar uma AWT reversível para imagens binárias.

4.2 ALGORITMOS DE COMPRESSÃO

4.2.1 **Introdução**

É chamado de "compressão de dados" o processo de conversão de informações de um formato para outro com a finalidade de reduzir o tamanho da informação original. Esta redução muitas vezes é necessária para facilitar o armazenamento e para melhorar a velocidade de transmissão dessas informações.

Existem dezenas de métodos para compressão para vários tipos de dados (textos, imagens, etc.). Porém todos seguem um mesmo princípio: remover as redundâncias da informação original. Nesta seção, serão discutidos de forma sucinta alguns dos algoritmos de compressão mais conhecidos.

4.2.2 ***Run-Length Encoding*** **(RLE)**

A idéia básica desta técnica é substituir uma seqüência de símbolos repetidos por um único símbolo seguido por um número que identifique a quantidade de

repetições. Isto é, se uma informação *i* aparece *n* vezes consecutivas, deve-se substituir essas *n* ocorrências de *i* de por um par *ni* (SALOMON, 2004). Uma seqüência de *n* ocorrências consecutivas de uma informação é chamada de "*run-length*".

Esta idéia, apesar de parecer simples, encontra alguns problemas práticos na sua implementação. Considere a sentença: AAAAABBB2CCCCABC. Tal seqüência poderia ser comprimida para: 5A3B24CABC. Um dos problemas encontrados na sentença comprimida é a ambigüidade que aparece quando caracteres numéricos fazem parte da sentença original. Por exemplo, a seqüência "24C" indica que existem 24 caracteres "C" ou que existe o caractere "2" seguido de 4 caracteres "C"? Neste exemplo, a segunda opção é a verdadeira.

Para resolver este problema, pode-se pensar na utilização de um caractere de controle ("@", por exemplo) para indicar que o número que se segue é a quantidade de repetições de um determinado caractere. A sentença anterior então ficaria com o seguinte formato: @5A@3B2@4CABC. Agora temos outro problema, nossas sentenças não poderiam usar o caractere "@" em sua composição.

O método chamado MNP5 (*Microcom Network Protocol class* 5), comumente utilizado para compressão de dados em modems, utiliza o conceito do RLE e trás uma solução para não haver necessidade da utilização do caractere de controle. Este método comprime apenas "*run-lengths*" com mais de três elementos. Neste caso, quando o algoritmo de codificação encontra três ou mais ocorrências consecutivas de um mesmo caractere, ele gera uma seqüência de três caracteres seguida por um número que indica a quantidade de repetições. Quando o algoritmo de decodificação encontra uma seqüência de três valores idênticos, ele sabe que o próximo caractere indica a quantidade de repetições. Este método possui alguns inconvenientes: seqüências de 3 caracteres precisam de 4 caracteres para serem codificados; seqüências de 4 caracteres precisam dos mesmos 4 caracteres para serem codificados; apenas seqüências de 5 ou mais caracteres podem ser codificados em 4 caracteres. Usando MNP5, a seqüência analisada anteriormente ficaria: AAA2BBB02CCC1ABC.

A técnica RLE também pode ser utilizada (com muito sucesso) para a compressão de imagens. Considere uma imagem binária, por exemplo, que possui 17 pixels brancos, seguidos de 10 pixels pretos, seguidos de 55 pixels brancos e assim por

diante. Essa imagem pode ser codificada pela seqüência de números 17, 10, 55, ... Neste caso, o compressor e o descompressor assumem que a seqüência sempre começa com pixels brancos. Se a imagem começa realmente com pixels pretos, um valor 0 é inserido no início da seqüência.

Este processo, com algumas variações, também pode ser aplicado para a compressão de imagens em níveis de cinza e coloridas.

4.2.3 **Código de Huffman**

Um dos métodos mais utilizados para a compressão de dados é o Código de Huffman. O método começa construindo uma lista de todos os símbolos do alfabeto em ordem decrescente de suas probabilidades. Então, constrói uma árvore, com um símbolo em cada folha, de baixo para cima. Isto é feito em etapas, onde em cada etapa dois símbolos da lista (aqueles com as menores probabilidades) são selecionados e adicionados ao topo da árvore que está sendo construída. Estes símbolos são removidos da lista e substituídos por um símbolo auxiliar que representa ambos. Quando a lista é reduzida a um único símbolo auxiliar (representando o alfabeto inteiro), a árvore está completa. Em seguida, a árvore é analisada para determinar o código de cada símbolo (SALOMON, 2004).

O exemplo a seguir será utilizado para ilustrar melhor este conceito. Considere um alfabeto com os símbolos a_1, a_2, a_3, a_4 e a_5 que possuem respectivamente as seguintes probabilidades 0,4; 0,2; 0,2; 0,1 e 0,1. A árvore vista na Figura 30 pode ser construída seguindo os seguintes passos:

1) a_4 é combinado com a_5 e ambos são substituídos pelo símbolo a_{45}, cuja probabilidade é 0,1+0,1 = 0,2.
2) Restam agora 4 símbolos na lista: a_1 com probabilidade 0,4 e a_2, a_3 e a_{45} com probabilidade 0,2. São selecionados arbitrariamente a_3 e a_{45}, combinados e substituídos pelo símbolo a_{345} cuja probabilidade é 0,4.
3) Dos três símbolos restantes, são selecionados a_2 e a_{345}, combinados e substituídos por a_{2345} cuja probabilidade é 0,6.

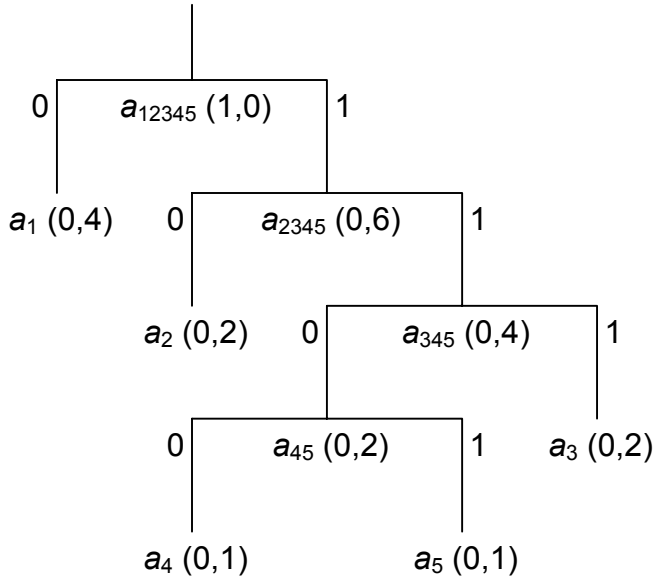


Figura 30 – Árvore construída para a obtenção dos códigos de Huffman.

4) Finalmente, os símbolos a_{2345} e a_1 são combinados e substituídos pelo símbolo a_{12345} que tem probabilidade 1.

O código de Huffman para cada símbolo, pode ser encontrado analisando-se a árvore da Figura 30. Basta seguir o caminho a partir da raiz até chegar à folha que contém o símbolo desejado. Os resultados obtidos podem ser vistos na Tabela 4. Note que um símbolo pode ser codificado, em média, por 2,2 bits (0,4×1 + 0,2×2 + 0,2×3 + 0,1×4 + 0,1×4). Note também que os símbolos que aparecem com maior freqüência possuem códigos menores.

Tabela 4 – Códigos de Huffman obtidos para cada símbolo.

Símbolo	**Probabilidade**	**Código**
a_1	0,4	0
a_2	0,2	10
a_3	0,2	111
a_4	0,1	1101
a_5	0,1	1100

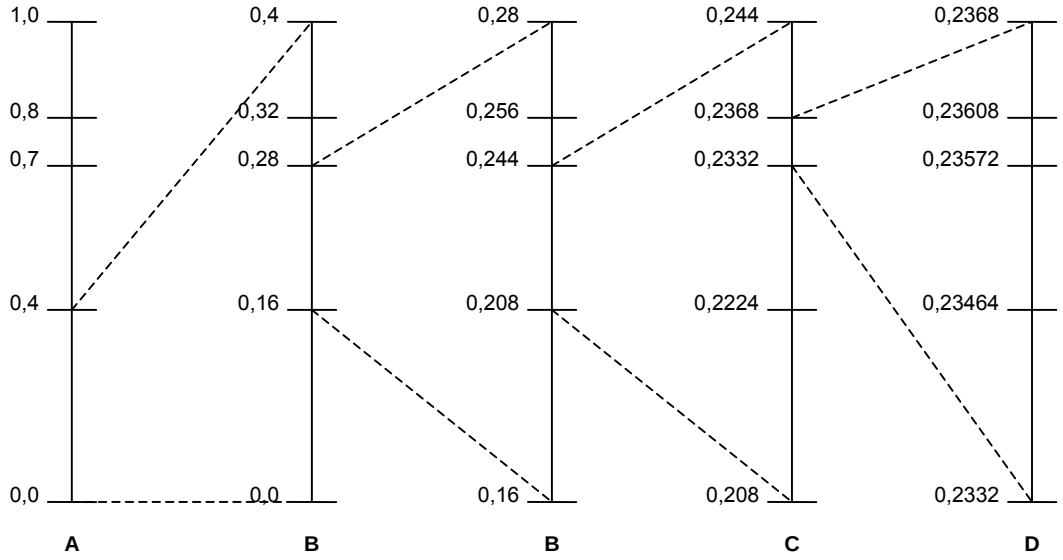


Figura 31 – Exemplo de codificação aritmética.

Como existem várias árvores diferentes para um mesmo alfabeto (e conseqüentemente várias tabelas de códigos), as informações necessárias para a reconstrução dos códigos de Huffman devem estar presentes no código comprimido.

4.2.4 **Codificação Aritmética**

A codificação aritmética permite representar uma seqüência qualquer de símbolos utilizando um único número real. Neste método, cada símbolo é codificado dentro de um intervalo específico que depende das probabilidades de ocorrência de cada símbolo e também dos símbolos da seqüência que já foram codificados. O modo mais fácil de compreender está técnica é através de um exemplo, como o que se segue.

Considere um alfabeto com os símbolos A, B, C, e D que possuem respectivamente as seguintes probabilidades 0,4; 0,3; 0,1 e 0,2. Considere também que a seqüência de símbolos ABBCD deve ser codificada. Acompanhe através da Figura 31 a seqüência de codificação descrita a seguir.

1) Dividir o intervalo [0; 1,0) em 4 subintervalos de acordo com as probabilidades de ocorrência dos símbolos. A notação [*a*;*b*) indica a faixa de números reais entre *a* e *b*, incluindo *a* e não incluindo *b*. Os intervalos obtidos são: [0; 0,4) correspondente à probabilidade 0,4 de ocorrência do símbolo A, [0,4; 0,7) correspondente à probabilidade 0,3 de ocorrência do símbolo B, [0,7; 0,8) correspondente à probabilidade 0,1 de ocorrência do símbolo C e [0,8; 1,0) correspondente à probabilidade 0,2 de ocorrência do símbolo D.

2) Como o primeiro símbolo a ser codificado é A, apenas o intervalo [0; 0,4) será utilizado de agora em diante para codificar os demais símbolos. Dividir o intervalo [0; 0,4) em outros 4 subintervalos como feito no passo 1. Os intervalos obtidos são: [0; 0,16) correspondente à probabilidade 0,4 de ocorrência do símbolo A, [0,16; 0,28) correspondente à probabilidade 0,3 de ocorrência do símbolo B, [0,28; 0,32) correspondente à probabilidade 0,1 de ocorrência do símbolo C e [0,32; 0,4) correspondente à probabilidade 0,2 de ocorrência do símbolo D.

3) Como o segundo símbolo a ser codificado é B, apenas o intervalo [0,16; 0,28) será utilizado de agora em diante para codificar os demais símbolos. Dividir o intervalo [0,16; 0,28) obtendo os subintervalos: [0,16; 0,208), [0,208; 0,244), [0,244; 0,256) e [0,256; 0,28).

4) Como o terceiro símbolo a ser codificado também é B, apenas o intervalo [0,208; 0,244) será utilizado de agora em diante para codificar os demais símbolos. Dividir o intervalo [0,208; 0,244) obtendo os subintervalos: [0,208; 0,2224), [0,2224; 0,2332), [0,2332; 0,2368) e [0,2368; 0,244).

5) Como o quarto símbolo a ser codificado é C, apenas o intervalo [0,2332; 0,2368) será utilizado de agora em diante para codificar os demais símbolos. Dividir o intervalo [0,2332; 0,2368) obtendo os subintervalos: [0,2332; 0,23464), [0,23464; 0,23572), [0,23572; 0,23608) e [0,23608; 0,2368).

6) Como o quinto e último símbolo a ser codificado é D, o intervalo [0,23608; 0,2368) será utilizado para representar a seqüência de símbolos. Em geral, o primeiro elemento do intervalo (0,23608) é utilizado para representar a seqüência, porém qualquer elemento deste intervalo (como o médio, por

exemplo) poderia ser utilizado. Na prática, apenas o valor 23608 é armazenado (não há necessidade de se armazenar o “0,”).

O processo de decodificação é semelhante:

1) Dividir o intervalo [0; 1,0) em 4 subintervalos: [0; 0,4), [0,4; 0,7), [0,7; 0,8) e [0,8; 1,0). Como o valor a ser decodificado (0,23608) está no primeiro intervalo, o primeiro símbolo extraído é A.
2) Dividir o intervalo [0; 0,4) entre 4 subintervalos: [0; 0,16), [0,16; 0,28), [0,28; 0,32) e [0,32; 0,4). Como o valor a ser decodificado (0,23608) está no segundo intervalo, o segundo símbolo extraído é B.
3) Dividir o intervalo [0,16; 0,28) obtendo os subintervalos: [0,16; 0,208), [0,208; 0,244), [0,244; 0,256) e [0,256; 0,28). Como o valor a ser decodificado (0,23608) está no segundo intervalo, o terceiro símbolo extraído também é B.
4) Dividir o intervalo [0,208; 0,244) obtendo os subintervalos: [0,208; 0,2224), [0,2224; 0,2332), [0,2332; 0,2368) e [0,2368; 0,244). Como o valor a ser decodificado (0,23608) está no terceiro intervalo, o quarto símbolo extraído é C.
5) Dividir o intervalo [0,2332; 0,2368) obtendo os subintervalos: [0,2332; 0,23464), [0,23464; 0,23572), [0,23572; 0,23608) e [0,23608; 0,2368). Como o valor a ser decodificado (0,23608) está no quarto intervalo, o quinto símbolo extraído é D. Assim, a seqüência extraída é ABBCD.

4.2.5 **Algoritmo de Golomb**

O algoritmo de Golomb é utilizado para codificar seqüências de zeros e uns, onde um “zero” ocorre com (alta) probabilidade p e um “um” ocorre com (baixa) probabilidade $1–p$. A probabilidade de uma subseqüência de n “zeros” é p^n e a probabilidade de uma subseqüência de n “zeros” seguida por um “um” é $p^n(1–p)$, indicando que as subseqüências são distribuídas geometricamente (SALOMON, 2004).

Segundo Salomon (2004), o algoritmo de Golomb é o melhor método para a compressão de dados que são distribuídos geometricamente. Segundo Gallager e Voorhis (1975), o algoritmo de Golomb é ótimo para distribuições com decaimento exponencial (geométrico) da forma Q(n) = $p^n(1–p)$, onde $0 < p < 1$.

A codificação usando o algoritmo de Golomb depende da escolha de um parâmetro inteiro $m \geq 2$, cujo melhor valor é:

$$m = \left\lceil -\frac{\log_2(1+p)}{\log_2 p} \right\rceil .$$

Os símbolos ⌈ e ⌉ indicam arredondamento para cima (*ceiling*) e os símbolos ⌊ e ⌋ indicam arredondamento para baixo (*floor*). Nesta subseção estes símbolos serão utilizados várias vezes.

Para valores pequenos de m, o código gerado é pequeno para seqüências curtas, mas cresce rapidamente em comprimento para seqüências maiores. Para valores grandes de m, o código gerado começa longo, mas seu comprimento cresce devagar.

Na distribuição geométrica, sempre haverá um valor de m que irá proporcionar o menor comprimento médio possível de código para representar de maneira exclusiva um número inteiro não negativo (WEINBERGER; SEROUSSI e SAPIRO, 1996).

4.2.5.1 Codificação

De maneira simplificada, pode-se dizer que esta técnica de compressão, que combina conceitos de RLE e Huffman, divide o valor a ser comprimido n pelo parâmetro m, armazenando o quociente (q) e o resto (r) desta divisão.

Para computar o código referente a um número inteiro (não negativo) n, deve-se utilizar o algoritmo a seguir.

Algoritmo 33 - Codificação de Golomb

1) Computar os valores *q*, *r* e *c*:

$$q = \left\lfloor \frac{n}{m} \right\rfloor, \quad r = n - qm, \quad \text{and} \quad c = \lceil \log_2 m \rceil.$$

2) Construir a primeira parte do código, codificado o valor de *q* em unário. Um número *k* codificado em unário consiste de *k* bits 1 seguidos de um bit 0. Por exemplo, o número 4 em unário é representado por 11110.

3) Construir a segunda parte do código, codificando o valor de *r* em binário da seguinte maneira:

   - se $r < 2^c - m$, *r* é codificado como um inteiro sem sinal em *c*–1 bits.
   - se $r \geq 2^c - m$, *r* é representado como o inteiro sem sinal $r + 2^c - m$ em *c* bits.

Exemplo 1: Codificar *n*=17 usando *m*=14:

$$q = \left\lfloor \frac{17}{14} \right\rfloor = 1, r = 17 - 1 \times 14 = 3 \text{ e } c = \lceil \log_2 14 \rceil = 4.$$

1) Codificando *q*=1 em unário resulta 10.

2) Como $r \geq 2^c - m$, o resto *r*=3 é codificado como o inteiro sem sinal $r + 2^c - m$=5 usando *c*=4 bits, que resulta $(0101)_2$.

3) Portanto, *n*=17 é codificado como 100101, resultado da concatenação de 10 e 0101.

Exemplo 2: Codificar a seguinte seqüência de bits:

0000010011000101000000111010001000001000100100011010000 1001

Esta seqüência tem 19 subseqüências de zeros:

5, 2, 0, 3, 1, 6, 0, 0, 1, 3, 5, 3, 2, 3, 0, 1, 4, 2 e 0.

O último zero indica que a seqüência termina com um 1. Como esta seqüência tem 41 zeros e 18 uns, a probabilidade de um 0 é 41/(41+18) ≅ 0.7, resultando:

$$m = \lceil -\log 1.7 / \log 0.7 \rceil = \lceil 1.487 \rceil = 2.$$

Então, codificando a seqüência com *m*=2, obtém-se uma seqüência de 19 códigos:

1101|100|00|101|01|11100|00|00|01|101|1101|101|100|101|00|01|1100|100|00

O resultado é uma seqüência de 52 bits que representa a seqüência original de 59 bits. A compressão não é maior porque *p* não é grande o suficiente.

Valores muito pequenos de *p*, tais como 0.1, resultam em uma seqüência com mais uns do que zeros. Neste caso, o algoritmo de Golomb pode ser usado para comprimir subseqüências de 1s. Para valores de *p* em torno de 0.5, o algoritmo de Golomb não é uma boa escolha e outros métodos devem ser considerados.

4.2.5.2 Decodificação

No algoritmo de decodificação de Golomb, os valores de *q* e *r* são usados para reconstruir o valor original *n* (*n* = *r* + *qm*).

Algoritmo 34 - Decodificação de Golomb

1) Computar o valor de *c*:

$$c = \lceil \log_2 m \rceil$$

2) Assumindo que o código começa com *a* 1s, começar removendo esses 1s e o zero seguinte.
3) Chamar de *r* os *c* − 1 bits seguintes.
4) Se $r < 2^c - m$, então o comprimento total do código é *a* + 1 + (*c* − 1) (os *a* 1s, o zero seguinte e os *c* − 1 bits seguintes) e seu valor é $m \times a + r$.
5) Se $r \geq 2^c - m$, então o comprimento total do código é *a* + 1 + *c* e seu valor é $m \times a + r' - (2^c - m)$, onde *r'* é o inteiro de *c* bits formado por *r* e pelo bit seguinte de *r*.

Exemplo: Decodificar 100101 usando *m* = 14:

1) Calcular $c = \lceil \log_2 14 \rceil = 4$.
2) Calcular a quantidade de 1s antes do primeiro 0 (*a* = 1).
3) Chamar de *r* os próximos *c* – 1 bits (*r* = $(010)_2$ = 2).
4) Como $r \geq 2^c - m$, então o comprimento do código é *a* + 1 + *c* = 6 bits.

5) Calcular r' = $(0101)_2$ = 5 (r e o bit seguinte de r).

6) Calcular $n = m \times a + r' - (2^c - m) = 14 \times 1 + 5 - 2 = 17$.

4.2.5.3 Códigos de Rice

O caso em que m é uma potência de 2 é especial, pois possui uma codificação muito mais simples, não necessitando da codificação em c–1 bits. Esses códigos são chamados de Códigos de Rice.

Neste caso, $2^c - m$ é sempre igual a zero e o valor r pode ser codificado diretamente em c bits.

Exemplo 1: Codificar n=17 usando m=16:

$$q = \left\lfloor \frac{17}{16} \right\rfloor = 1,\ r = 17 - 1 \times 16 = 1 \text{ e } c = \lceil \log_2 16 \rceil = 4 .$$

1) Codificando q=1 em unário resulta 10.

2) Codificando r=1 usando c=4 bits, que resulta $(0001)_2$.

3) Portanto, n=17 é codificado como 100001, resultado da concatenação de 10 e 0001.

Exemplo 2: Decodificar 100001 usando m=16:

1) Calcular $c = \lceil \log_2 16 \rceil = 4$.

2) Calcular a quantidade de 1s antes do primeiro 0 (a = 1).

3) Chamar de r os próximos c bits (r = $(0001)_2$ = 1).

4) Calcular $n = m \times a + r = 16 \times 1 + 1 = 17$.

4.2.6 **Outros**

Além dos algoritmos de compressão descritos nesta seção, existem mais alguns, como JPEG, JPEG2000, JBIG1, JBIG2 e outros. Porém, a maioria deles também

não pode ser utilizada de maneira eficiente para a compressão de uma quantidade pequena de bits.

4.3 *REVERSIBLE DATA-HIDING BY TEMPLATE RANKING WITH SYMMETRICAL CENTRAL PIXELS* (RDTC)

Esta seção propõe uma técnica esteganográfica reversível para imagens binárias chamada RDTC (*Reversible Data hiding by Template ranking with symmetrical Central pixels* - esteganografia reversível pelo ranqueamento de modelos com pixels centrais simétricos), baseada na técnica DHTC descrita previamente.

Na RDTC, três tipos de informação devem ser embutidas na imagem hospedeira: a informação comprimida que permite recuperar a imagem original, a informação que realmente se quer embutir (NPD – *Net Payload Data*) e informações de controle (quantidade de DBPs, por exemplo). Desta forma, os valores originais dos *n* primeiros DBPs devem ser comprimidos para criar espaço para o armazenamento do NPD e das informações de controle.

Como foi visto anteriormente, existem algumas dificuldades para comprimir os valores originais dos DBPs. A maioria dos algoritmos de compressão baseados em redundância e dicionários não funcionam, porque a quantidade de bits a serem comprimidos é muito pequena. Não funcionam RLE, Huffman, Codificação Aritmética, LZW, JBIG, JBIG2, etc. Além disso, não é possível prever o valor do próximo bit baseando-se no anterior, pois estes bits correspondem a pixels dispersos por toda a imagem.

Previsão

A solução encontrada foi comprimir as previsões dos valores dos DBPs (usando os pixels vizinhos como informação adicional) ao invés de comprimir diretamente os valores dos DBPs. Um pixel pode ser da mesma cor ou de uma cor diferente da maioria de seus vizinhos. Será assumido que a primeira hipótese (o pixel é da mesma cor que a maioria de seus vizinhos) é mais provável que a segunda. Seja *b* o número de pixels vizinhos de cor preta de um DBP (usando modelos 3×3, um DBP possui 8 pixels vizinhos). A previsão é correta (representada por 0) se o DBP original

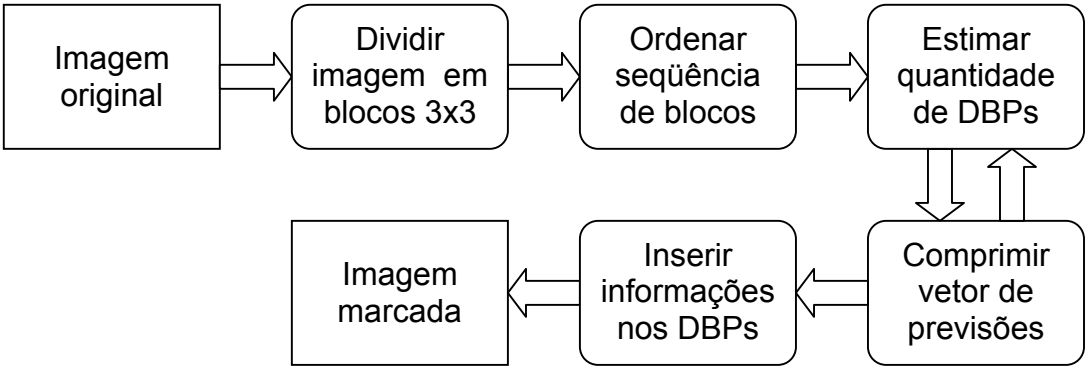


Diagrama 13 – Algoritmo de inserção da RDTC.

é preto e $b>4$, ou se é branco e $b\leq4$. Caso contrário, a previsão é incorreta (representada por 1).

Se a previsão for boa, o valor previsto e o valor verdadeiro devem ser iguais com probabilidade superior a 50%. Como o valor zero é armazenado quando a previsão é correta e o valor um é armazenado quando a previsão é incorreta, subseqüências de zeros devem ser mais longas (na maioria das vezes) do que subseqüências de uns, o que torna a compressão possível.

Um método eficiente para comprimir segmentos (geralmente longos) de zeros separados por segmentos (geralmente curtos) de uns é o algoritmo de Golomb (GALLAGER; VOORHIS, 1975; GOLOMB, 1966; SALOMON, 2004). Alguns outros métodos baseados no algoritmo de Golomb como LOCO-I (WEINBERGER et al., 1996), FELICS (HOWARD; VITTER, 1993) e JPEG-LS (ISO/IEC, 1997) também parecem ser eficientes, porém não foram testados por nós.

Como a vizinhança dos DBPs não é modificada durante a inserção, as previsões podem ser reconstruídas no processo de extração. O vetor de erros de previsão (0s e 1s), junto com a vizinhança dos DBPs, permite a recuperação dos valores originais dos DBPs. Se necessário, outros métodos de previsão pode ser utilizados dependendo do tipo de imagem que está sendo utilizada.

Algoritmo 35 - Inserção do RDTC

1) Dividir a imagem original Z em uma seqüência v de blocos não sobrepostos (por exemplo, 3×3).

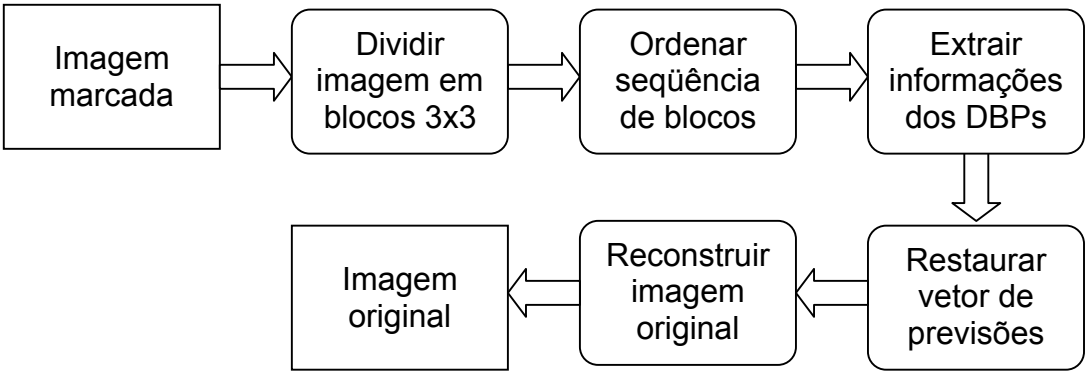


Diagrama 14 – Algoritmo de extração da RDTC.

2) Ordenar a seqüência *v* de maneira crescente utilizando a nota de impacto visual como chave primária, a quantidade de vizinhos pretos em torno do pixel central como chave secundária e um número pseudo-aleatório sem repetição como chave terciária.

3) Estimar a quantidade mínima *n* de DBPs capaz de armazenar um cabeçalho (de tamanho *h* bits), o vetor de erros de previsão comprimido (tamanho *w*) e a informação que realmente se deseja embutir (NPD – *Net Payload Data*, tamanho *p*), por exemplo, que satisfaça a condição $n \geq h + w + p$. Tentar iterativamente diferentes valores de *n*, até obter o menor valor de *n* que satisfaça a condição acima.

4) Inserir o cabeçalho (valores de *n*, *w*, *p* e o parâmetro *m* do algoritmo de Golomb), o vetor de erros de previsão comprimido e o NPD modificando (se necessário) os pixels centrais dos *n* primeiros blocos da seqüência ordenada *v*.

Algoritmo 36 - Extração do RDTC

1) Reconstruir e ordenar a seqüência *v* de blocos 3×3 como no processo de inserção.

2) Extrair os valores de *w*, *m*, *n* e *p* (cabeçalho) dos pixels centrais dos *h* primeiros blocos da seqüência ordenada *v*.

3) Extrair o NPD dos pixels centrais dos *p* blocos seguintes da seqüência ordenada *v*.

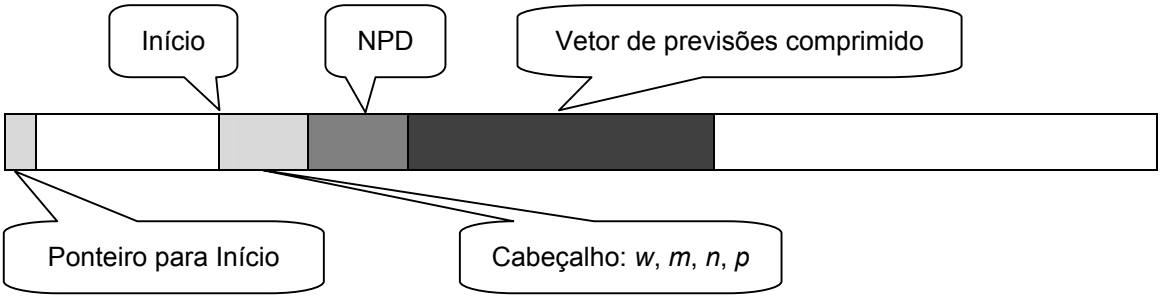


Figura 32 – Organização da seqüência *v* de blocos não sobrepostos, indicando a informação armazenada no pixel central dos blocos.

4) Extrair o vetor de erros de previsão comprimido dos *w* blocos seguintes da seqüência ordenada *v*.
5) Descomprimir o vetor de erros de previsão utilizando algoritmo de Golomb com o parâmetro *m* extraído do cabeçalho.
6) Utilizar o vetor de erros de previsão para reconstruir a imagem original.

No algoritmo proposto, a informação foi embutida no começo da seqüência *v*, porque esta parte possui os pixels centrais de visibilidade mais baixa, resultando em imagens de melhor qualidade visual. Porém, para obter uma capacidade de armazenamento maior, a seqüência *v* pode ser escaneada em busca de um segmento que permita uma melhor compressão. Os pixels do começo de *v* são menos visíveis, mas não podem ser previstos com muita acuidade, pois são pixels que estão no contorno da imagem e possuem quantidades semelhantes de vizinhos pretos e brancos. Ao avançar em *v*, são encontrados pixels que possuem uma visibilidade maior, porém podem ser previstos com maior acuidade e proporcionam uma melhor compressão. Neste caso, um apontador deve ser posicionado no começo de *v* indicando a posição onde o restante da informação foi embutida (Figura 32). Pelo exposto, pode-se notar que existe uma relação inversa entre a capacidade de armazenamento e a qualidade da imagem.

Imagens marcadas pela técnica RDTC possuem excelente qualidade visual, pois apenas os pixels de baixa visibilidade são modificados. RDTC é adequada para marcar muitos tipos de imagens binárias, como textos escaneados ou gerados por computador, diagramas e gráficos, imagens do tipo caricatura (*cartoon*) e do tipo

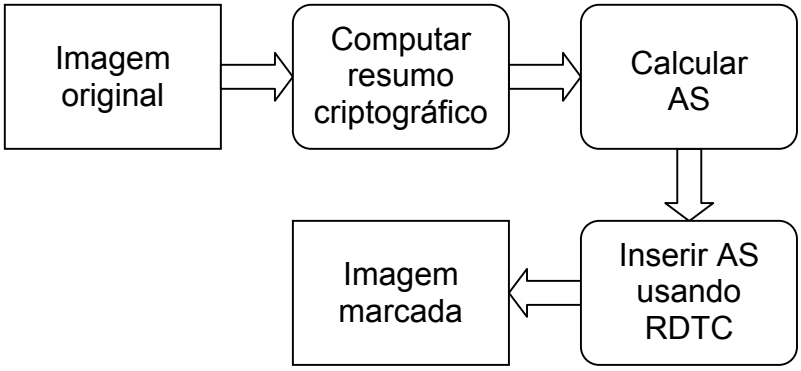


Diagrama 15 – Algoritmo de inserção da RATC.

meio-tom com pontos aglutinados (*clustered-dot halftone*). RDTC também pode ser utilizada para marcar imagens do tipo meio-tom com pontos dispersos (*dispersed-dot halftone*), como imagens geradas por difusão de erro (*error diffusion*), porém a qualidade visual da imagem marcada pode não ser tão boa, pois o conceito de pixel de baixa visibilidade não se aplica a este tipo de imagem.

4.4 *REVERSIBLE AUTHENTICATION WATERMARKING BY TEMPLATE RANKING WITH SYMMETRICAL CENTRAL PIXELS* (RATC)

Nesta seção, será mostrado como uma marca d'água de autenticação reversível pode ser facilmente criada usando os conceitos da técnica RDTC. A técnica proposta, chamada de RATC (*Reversible Authentication watermarking by Template ranking with symmetrical Central pixels* - marca d'água de autenticação reversível pelo ranqueamento de modelos com pixels centrais simétricos), pode detectar qualquer alteração na imagem marcada, até mesmo um único pixel, e pode ser utilizada com criptografia de chave secreta ou de chave pública.

Algoritmo 37 - Inserção da RATC

1) Dada uma imagem binária Z a ser autenticada, computar o resumo criptográfico de Z utilizando uma função de *hash* criptograficamente segura $H = H(Z)$.
2) Criptografar o resumo H usando a chave privada ou a chave secreta, obtendo a AS S.

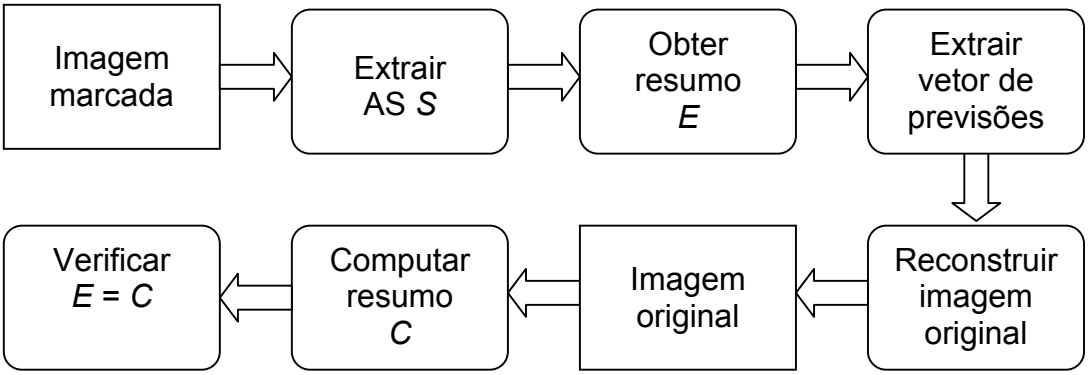


Diagrama 16 – Algoritmo de verificação da RATC.

3) Inserir *S* em *Z* usando RDTC, obtendo a imagem marcada *Z'*.

Algoritmo 38 - Verificação da RATC

1) Dada uma imagem marcada *Z'*, extrair a assinatura de autenticação *S*.
2) Decriptografar *S* usando a chave pública ou a chave secreta, obtendo o resumo criptográfico extraído *E*.
3) Extrair o vetor de erros de previsão, descomprimi-lo e restaurar a imagem original *Z*.
4) Recalcular o *hash* da imagem, obtendo o resumo criptográfico de verificação $C = H(Z)$.
5) Se o resumo extraído *E* e o resumo de verificação *C* forem iguais, a marca está verificada. Caso contrário, a imagem foi adulterada.

Esta técnica possui as mesmas características da AWTC, sendo completamente imune aos ataques de paridade discutidos anteriormente.

4.4.1 **Resultados experimentais das técnicas RDTC e RATC**

A técnica RDTC foi testada para embutir de maneira reversível 128 bits de informação em imagens binárias de diferentes tipos e tamanhos (textos escaneados, textos gerados por computador, *cartoons*, meio-tons, ruídos, etc.). Como é sabido, 128 bits são suficientes para armazenar um MAC (*message authentication code*) usado na autenticação de imagens com chave secreta.

Nos testes realizados, foi necessário comprimir em média (excluindo-se as imagens com ruídos aleatórios) apenas 437 pixels de baixa visibilidade para obter espaço suficiente para armazenar os 128 bits da NPD, 37 bits de cabeçalho e o vetor de erros de previsão comprimido (Tabela 5).

As imagens marcadas apresentaram excelente qualidade visual (Figuras 33 a 36), pois apenas pixels de baixa visibilidade foram modificados. As imagens recuperadas após a extração da NPD são idênticas às originais.

A Figura 33 mostra uma página de revista, composta apenas por texto, escaneada a 400 dpi. Este é o tipo ideal de imagem para a aplicação desta técnica, pois é formada por componentes conexos pequenos, os quais se adaptam muito bem à teoria de pixels de baixa visibilidade. Note que a imagem foi marcada com um MAC dez vezes maior do que o tamanho usual (1280 bits) e ainda assim, a qualidade apresentada é excelente.

A Figura 34 apresenta uma imagem meio-tom com pontos aglutinados. Após as imagens compostas apenas por texto, este é o tipo de imagem meio-tom mais apropriada para a aplicação desta técnica. Como pode ser observado, a qualidade da imagem marcada é excelente.

A Figura 35 apresenta uma imagem meio-tom com pontos dispersos. Este tipo de imagem não é apropriado para a aplicação desta técnica, pois não existe, neste tipo de imagem, o conceito de pixel de baixa visibilidade. Apesar da segurança da marca e a reversibilidade da imagem serem preservadas, é possível enxergar na imagem ampliada alguns pixels alterados com a inserção da marca. A qualidade da imagem marcada não é excelente, mas continua muito boa.

A Figura 36 apresenta um tipo de imagem que, apesar de não ser formado por elementos conexos pequenos, também se adapta bem a esta técnica. Porém, se a imagem marcada for observada com atenção, alguns pontos podem ser vistos, pois o tamanho desta figura é muito pequeno e a resolução da mesma é extremamente baixa. A qualidade da imagem marcada não é excelente, mas pode ser considerada muito boa.

Como se pode ver na Tabela 5, apenas dois tipos de imagem não puderam ser marcados: (1) imagens muito pequenas, pois não há espaço suficiente para armazenar as informações necessárias e (2) imagens com ruídos aleatórios com

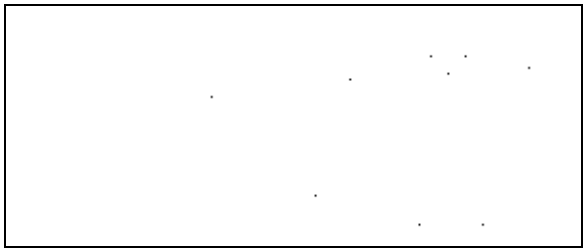
quantidades similares de pixels pretos e brancos, pois a previsão é muito difícil. Por outro lado, estes dois tipos de imagem são extremamente raros e a técnica proposta pode ser utilizada praticamente em todas as imagens.

Tabela 5 – Inserção de 128 bits de informação (*p*) e 37 bits de cabeçalho (*h*) em diferentes imagens, onde *n* é o número de DBPs e *w* é o tamanho do vetor comprimido de DBPs. “Não” significa que a inserção não foi possível.

Imagem	**Descrição**	**Tamanho**	***n***	***w***	***n*-*w***
lena0	Difusão de Erro	512×512	432	264	168
lena2	Pontos aglutinados	512×512	272	98	174
fides	Texto gerado por computador	1275×1650	432	259	173
persuas	Texto escaneado a 150 dpi	1275×1650	560	395	165
toip300	Texto escaneado a 300 dpi	2384×3194	496	325	171
toip400	Texto escaneado a 400 dpi	3179×4259	464	293	171
abc	Texto gerado por computador (pequeno)	91×58	400	219	181
pag1	Texto gerado por computador (muito pequeno)	64×56	Não	Não	Não
noise10	10% de pixels pretos aleatórios	300×300	400	227	173
noise20	20% de pixels pretos aleatórios	300×300	880	711	169
noise25	25% de pixels pretos aleatórios	300×300	1264	1098	166
noise30	30% de pixels pretos aleatórios	300×300	3824	3654	170
noise35	35% de pixels pretos aleatórios	300×300	Não	Não	Não
noise65	65% de pixels pretos aleatórios	300×300	Não	Não	Não
noise70	70% de pixels pretos aleatórios	300×300	1424	1256	168
noise75	75% de pixels pretos aleatórios	300×300	880	712	168
noise80	80% de pixels pretos aleatórios	300×300	592	423	169
noise90	90% de pixels pretos aleatórios	300×300	368	194	174

(a) Imagem original.

(b) Imagem marcada.

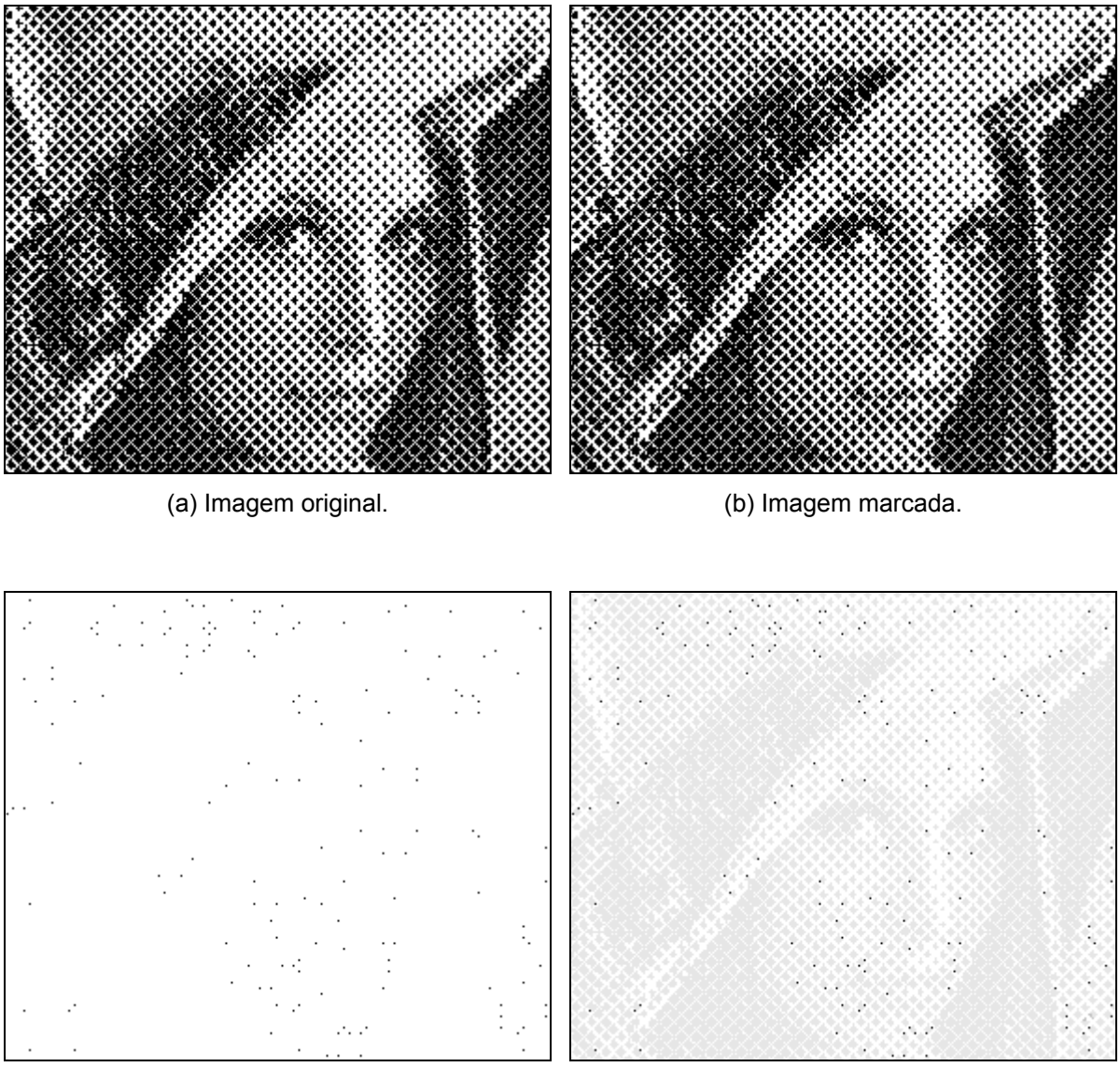


(c) Pixels modificados.

(d) Pixels modificados sobrepostos ao original.

(e) Imagem sobreposta ampliada

Figura 33 – Ampliação de parte de uma página de revista escaneada a 400 dpi, com 3179×4259 pixels e marcada de maneira reversível pela RATC com um MAC de 1280 bits (dez vezes maior do que o tamanho usual de um MAC) usando 3440 pixels para armazenar informações.



(a) Imagem original. (b) Imagem marcada.

(c) Pixels modificados. (d) Pixels modificados sobrepostos ao original.

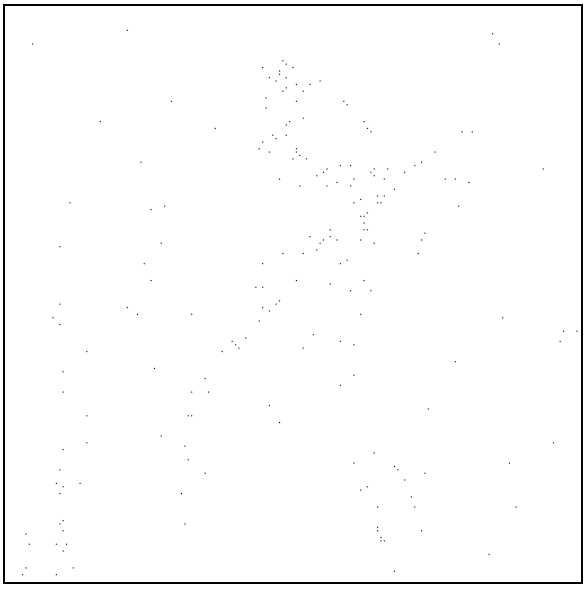
Figura 34 – Imagem meio-tom pontos aglutinados (512×512 pixels) marcada de maneira reversível com MAC de 128 bits, usando 272 pixels para armazenar informação.

(a) Imagem original.

(b) Imagem marcada.

(c) Imagem original ampliada.

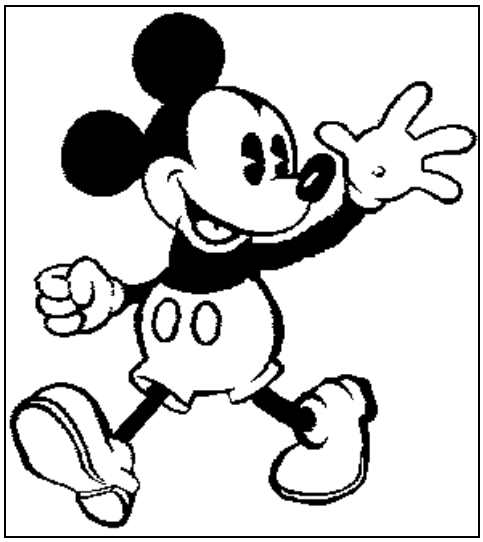
(d) Imagem marcada ampliada.



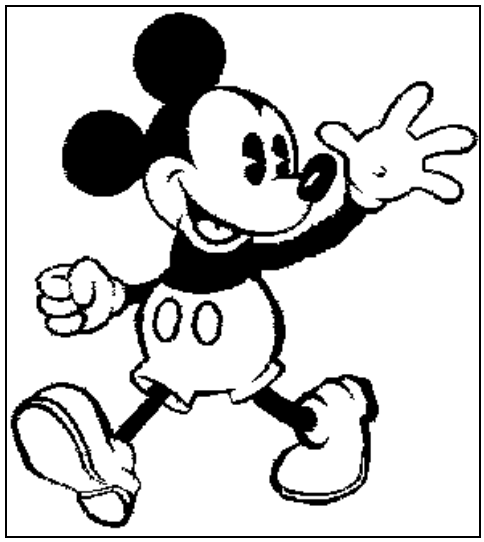
(e) Pixels modificados.

(f) Pixels modificados sobrepostos ao original.

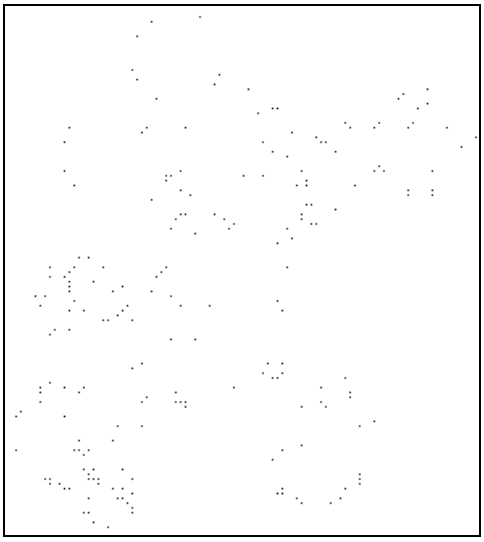
Figura 35 – Imagem meio-tom pontos dispersos (512×512 pixels) marcada de maneira reversível com MAC de 128 bits, usando 432 pixels para armazenar informação.



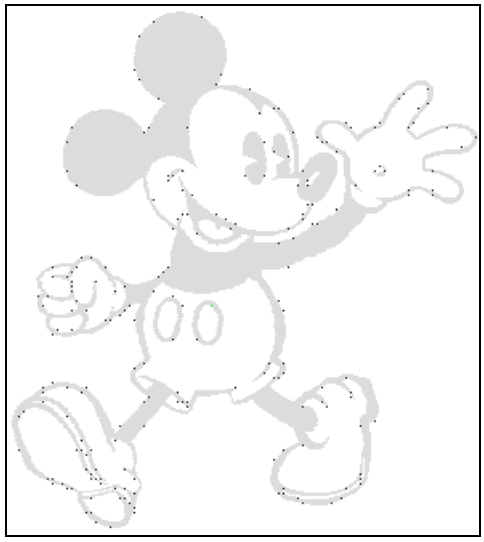
(a) Imagem original (resolução muito baixa).



(b) Imagem marcada.



(c) Pixels modificados.

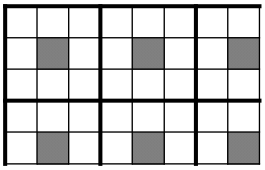


(d) Pixels modificados sobrepostos ao original.

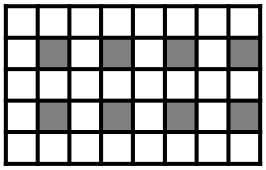
Figura 36 – Imagem de resolução muito baixa com apenas 295×331 pixels, marcada de maneira reversível com MAC de 128 bits, usando 496 pixels para armazenar informação.

4.4.2 **Utilização de blocos parcialmente sobrepostos**

Após o desenvolvimento de várias técnicas que utilizavam blocos 3×3 não sobrepostos (DHTC, AWTC, AWCJT, RDTC, RATC e outras), notou-se que não haveria nenhum problema se os mesmos fossem parcialmente sobrepostos desde que nenhum vizinho do pixel central fosse candidato a DBP (KIM et al., 2007), conforme ilustra a Figura 37.



(a) Blocos não sobrepostos.



(b) Blocos parcialmente sobrepostos.

Figura 37 – Divisão da imagem em blocos (KIM et al., 2007). (a) Blocos não sobrepostos. (b) Blocos parcialmente sobrepostos. Pixels centrais (candidatos a DBP) aparecem hachurados.

Com isto, obtém-se um aumento de cerca de 125% na capacidade de armazenamento dessas técnicas. Utilizando blocos não sobrepostos, apenas 1 pixel em cada 9 poderia ser candidato a DBP. Utilizando blocos parcialmente sobrepostos, 1 em cada 4 pixels pode ser candidato a DBP.

Tabela 6 – Inserção de 128 bits de informação (*p*) e 37 bits de cabeçalho (*h*) em diferentes imagens, onde *n* é o número de DBPs e *w* é o tamanho do vetor comprimido de DBPs. Nestes testes, foram utilizados como candidatos a DBP os pixels centrais de blocos 3×3 parcialmente sobrepostos.

Imagem	**Descrição**	**Tamanho**	***n***	***w***	***n-w***
lena0	Difusão de Erro	512×512	400	233	167
lena2	Pontos aglutinados	512×512	208	41	167
fides	Texto gerado por computador	1275×1650	336	146	190
persuas	Texto escaneado a 150 dpi	1275×1650	368	203	165
toip300	Texto escaneado a 300 dpi	2384×3194	528	355	173
toip400	Texto escaneado a 400 dpi	3179×4259	464	290	174
abc	Texto gerado por computador (pequeno)	91×58	368	200	168
pag1	Texto gerado por computador (muito pequeno)	64×56	368	193	175

Como foi visto na Tabela 5, é necessário comprimir em média 437 pixels de baixa visibilidade para obter espaço suficiente para armazenar informações utilizando-se blocos não sobrepostos. Porém, com a utilização de blocos parcialmente sobrepostos, esta quantidade cai para 380 pixels, como pode ser visto na Tabela 6.

Note também que a imagem “pag1.bmp”, de dimensões reduzidas, que não podia ser marcada (Tabela 5) porque a quantidade de DBPs não era suficiente, pôde ser marcada quando foram utilizados blocos parcialmente sobrepostos (Tabela 6).

4.4.3 **Estudo de capacidade de armazenamento**

Como foi dito anteriormente, nas técnicas RDTC e RATC a informação é embutida no começo da seqüência *v*, porque esta parte possui os pixels centrais de visibilidade mais baixa, resultando em imagens de melhor qualidade visual. Porém, para obter uma capacidade de armazenamento maior, a seqüência *v* pode ser escaneada em busca de um segmento que permita uma melhor compressão.

Tabela 7 – Inserção de 128 bits de informação (*p*) e 37 bits de cabeçalho (*h*) em diferentes imagens, onde *n* é o número de DBPs e *w* é o tamanho do vetor comprimido de DBPs. Nestes testes, foram utilizados blocos 3×3 parcialmente sobrepostos e foi feita uma busca pelo segmento que permite melhor compressão.

Imagem	**Descrição**	**Tamanho**	***n***	***w***	***n-w***
lena0	Difusão de Erro	512×512	336	171	165
lena2	Pontos aglutinados	512×512	176	11	165
fides	Texto gerado por computador	1275×1650	176	11	165
persuas	Texto escaneado a 150 dpi	1275×1650	176	11	165
toip300	Texto escaneado a 300 dpi	2384×3194	176	11	165
toip400	Texto escaneado a 400 dpi	3179×4259	176	11	165
abc	Texto gerado por computador (pequeno)	91×58	176	11	165
pag1	Texto gerado por computador (muito pequeno)	64×56	176	11	165

As Tabelas 5 e 6 ilustram testes realizados com a inserção de informação no início da seqüência *v*. A Tabela 7 ilustra testes realizados com as mesmas imagens, porém fazendo-se a busca pelo segmento que proporciona melhor compressão.

Note que a quantidade média de pixels necessária para armazenar informação é de apenas 196 pixels.

Também foi realizado um outro estudo para verificar a capacidade máxima de armazenamento de algumas imagens. Os resultados obtidos são apresentados na Tabela 8.

Tabela 8 – Quantidade máxima de bits de informação que podem ser armazenados de maneira reversível em uma imagem utilizando RDTC e RATC.

Imagem	Descrição	Tamanho (pixels)	Máximo (pixels e %)
lena0	Difusão de Erro	512×512 = 262144	6.773 (2,6%)
lena2	Pontos aglutinados	512×512 = 262144	4.287 (1,6%)
fides	Texto gerado por computador	1.275×1.650 = 2.103.750	401.600 (19,1%)
persuas	Texto escaneado a 150 dpi	1.275×1.650 = 2.103.750	214.032 (10,2%)
toip300	Texto escaneado a 300 dpi	2.384×3.194 = 7.614.496	1.543.680 (20,3%)

Analisando estes resultados, pode-se afirmar que as técnicas reversíveis propostas neste trabalho se mostraram muito boas quanto à capacidade de armazenamento, levando-se em conta que uma imagem utiliza no máximo 25% de seus pixels para armazenar informações (1 pixel em cada grupo de 4 pixels, se forem utilizados blocos parcialmente sobrepostos), sobretudo para documentos de texto. A figura “toip300.bmp”, por exemplo pode utilizar até 20,3% de seus pixels para o armazenamento de informações (1.543.680 pixels).

É certo que o armazenamento de uma quantidade muito grande informação prejudica a qualidade visual da imagem hospedeira, pois forçadamente pixels de visibilidade alta têm que ser utilizados. Porém, se a intenção é inserir uma marca d’água de autenticação, a quantidade de pixels necessária é muito mais baixa e as imagens resultantes apresentam excelente qualidade visual.

5 CONCLUSÕES

Neste trabalho foram propostas várias técnicas esteganográficas e AWTs para imagens codificadas no padrão JBIG2. Também foi proposta uma técnica reversível para ocultação de informações em imagens binárias e uma AWT reversível também para imagens binárias.

Nas técnicas propostas para imagens JBIG2, a imagem a ser marcada pode estar codificada com ou sem perda e a verificação da imagem pode ser feita tanto no arquivo JBIG2 quanto na imagem binária obtida pela decodificação do arquivo JBIG2.

A primeira técnica esteganográfica (DHCCJ) proposta para imagens JBIG2 não se mostrou eficiente, pois as imagens hospedeiras (principalmente as de baixa resolução) não apresentavam qualidade visual satisfatória e, por isso, esta técnica foi abandonada. Esta técnica foi publicada em (PAMBOUKIAN; KIM; DE QUEIROZ, 2005).

A segunda técnica esteganográfica (DHTRJ) proposta para imagens JBIG2 insere informações na região de texto da imagem JBIG2 e gera imagens hospedeiras de excelente qualidade visual, sem ruídos do tipo sal-e-pimenta, até mesmo em imagens pequenas e de baixa resolução. Esta técnica foi publicada em (PAMBOUKIAN; KIM; DE QUEIROZ, 2005).

Esta segunda técnica foi utilizada para criar uma marca d'água de autenticação (AWTRJ) para imagens JBIG2. Esta AWT pode ser utilizada para autenticar qualquer imagem JBIG2 que possua uma região de texto grande o suficiente para armazenar um Código de Autenticação de Mensagem (MAC – *Message Authentication Code*) e é capaz de detectar qualquer alteração feita na imagem (mesmo um único pixel). Apenas a versão de chave secreta desta técnica é protegida contra ataques de paridade. Esta técnica foi publicada em (PAMBOUKIAN; KIM; DE QUEIROZ, 2005).

A terceira técnica esteganográfica (DHCJT) proposta para imagens JBIG2 foi testada em diversos tipos de imagens binárias geradas por software e escaneadas em diferentes resoluções. As imagens hospedeiras, mesmo as de tamanho reduzido

e baixa resolução, apresentaram excelente qualidade visual. Esta técnica foi publicada em (PAMBOUKIAN; KIM, 2005).

Esta técnica foi utilizada para gerar uma marca d'água de autenticação (AWCJT) totalmente imune a ataques de paridade, tanto nas versões de chave secreta quanto nas versões de chave pública. Nos testes realizados com esta técnica, qualquer alteração visual significante (até mesmo um único pixel) na imagem marcada pôde ser detectada. Esta técnica foi publicada em (PAMBOUKIAN; KIM, 2005).

Foi desenvolvida também uma técnica para marcar imagens JBIG2 inserindo a AS na região de meio-tom (AWCJH) e uma técnica que unifica as anteriores (AWTCJ) permitindo que as informações armazenadas sejam distribuídas pelas diversas regiões que compõem a imagem JBIG2.

Além das técnicas para JBIG2 também foram apresentadas duas técnicas reversíveis para imagens binárias (uma técnica esteganográfica e uma marca d'água de autenticação).

A técnica esteganográfica reversível proposta para imagens binárias (RDTC) utiliza os conceitos da técnica DHTC para a escolha dos pixels que devem carregar informação e o algoritmo de Golomb para comprimir as informações necessárias à recuperação da imagem original. A qualidade da imagem hospedeira é excelente, pois apenas pixels de baixa visibilidade são alterados e a imagem original pode ser restaurada sem perda após a extração das informações embutidas. Nos testes realizados em vários tipos de imagens binárias, em média, apenas 380 pixels precisaram ser comprimidos para armazenar uma informação de 128 bits, além das informações necessárias para recuperar a imagem original. Esta técnica foi publicada em (PAMBOUKIAN; KIM, 2006).

Esta técnica reversível foi utilizada para a criação de uma marca d'água de autenticação reversível (RATC) para imagens binárias. Esta técnica permite detectar qualquer alteração feita na imagem e também permite a restauração da imagem original após a extração da informação armazenada. Esta técnica foi publicada em (PAMBOUKIAN; KIM, 2006).

O trabalho que apresenta as duas técnicas reversíveis (RDTC e RATC) recebeu Menção Honrosa no 6º SBSEG - Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais.

Vale a pena ressaltar que, segundo nossos estudos, não existem técnicas esteganográficas e AWTs propostas para o formato JBIG2 e também não existem técnicas esteganográficas reversíveis ou AWT reversíveis que funcionem de maneira correta para imagens binárias.

5.1 CONTINUIDADE DO TRABALHO

A seguir são apresentadas algumas idéias e sugestões para pesquisas futuras e continuidade do presente trabalho.

- Apesar do algoritmo de Golomb ter se mostrado satisfatório para a compressão de informações nas técnicas reversíveis apresentadas, acreditamos que alguns outros métodos derivados de Golomb (LOCO-I, FELICS e outros) podem apresentar resultados talvez mais eficientes do que Golomb. Seria interessante fazer um estudo comparativo entre essas técnicas para determinar qual a mais adequada para a compressão de dados nas técnicas reversíveis.
- A previsão da cor do pixel central baseada na cor dos pixels vizinhos talvez possa ser melhorada. Seria interessante estudar algumas técnicas que utilizam previsão e contexto como JBIG1 (ISO/IEC, 1992), por exemplo, e tentar fazer uma previsão mais acurada da cor do pixel central. Isto ajudaria muito na compressão das informações utilizadas para recuperar a imagem original.
- Desenvolvimento de técnicas para inserção de marcas d’água em imagens digitais resistentes à manipulação: rotação, translação, escala, recorte, impressão e escaneamento.

REFERÊNCIAS

ADOBE SYSTEMS INCORPORATED. **PDF reference version 1.4**. 3rd ed. Addison-Wesley, 2001. 978 p. Disponível em: <http://www.adobe.com/devnet/pdf/pdfs/PDFReference.pdf>. Acesso em: mar. 2007.

AFIF, A. **Estudo de esteganografia e marcas d'água de autenticação para imagens binárias e meio-tom**. 2004. 115 p. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2004.

ARISP/IRIB. Associação dos Registradores Imobiliários de São Paulo / Instituto de Registro Imobiliário do Brasil. **Cartilha de certificação digital**. Disponível em: <https://www.oficioeletronico.com.br/Downloads/CartilhaCertificacaoDigital.pdf>. Acesso em: mar. 2007.

ATALASOFT. **DotImage JBIG2 codec.** Disponível em: <http://www.atalasoft.com/products/dotimage/jbig2/>. Acesso em: dez. 2006.

AWRANGJEB, M; KANKANHALLI, M. S. Lossless watermarking considering the human visual system. **Lecture Notes in Computer Science**, Springer-Verlag, v. 2939, p. 581-592, 2004. Apresentado ao International Workshop on Digital Watermarking, Seoul, Korea, 2003.

BAHARAV, Z.; SHAKED, D. Watermarking of dither halftone images. In: SPIE SECURITY AND WATERMARKING OF MULTIMEDIA CONTENTS, 1999, San Jose, California, USA. **Proceedings…** v. 3657, p. 307-316. Também disponível como Tech. Rep. HPL-98-32, Hewlett-Packard Labs, 1998.

BARRETO, P. S. L. M. **Criptografia robusta e marcas d'água frágeis: construção e análise de algoritmos para localizar alterações em imagens digitais**. 2003. 150 p. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2003.

BARRETO, P. S. L. M.; KIM, H. Y. Pitfalls in public key watermarking. In: BRAZILIAN SYMP. ON COMP. GRAPH. AND IMAGE PROCESSING, 12., 1999, Campinas. **SIBGRAPI Proceedings**. Campinas: Unicamp, 1999. p. 241–242.

BARRETO, P. S. L. M.; KIM, H. Y.; RIJMEN, V. Um modo de operação de funções de hashing para localizar alterações em dados digitalmente assinados. In:

SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES, 18., 2000, Gramado. **Anais do SBrT**. Gramado: SBT, 2000. Paper 5150124.

BARRETO, P. S. L. M.; KIM, H. Y.; RIJMEN, V. Toward a secure public-key blockwise fragile authentication watermarking. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 2001, Thessaloniki, Greece. **ICIP Proceedings**. Piscataway: IEEE, 2001. v. 2, p. 494-497.

BARRETO, P. S. L. M.; KIM, H. Y.; RIJMEN, V. Toward a secure public-key blockwise fragile authentication watermarking. **Vision, image and signal processing**, v. 149, n. 2, p. 57-62, 2002.

BONEH, D.; LYNN, B.; SHACHAM, H. Short signatures from the Weil pairing. **Lecture Notes in Computer Science**, Springer-Verlag, v. 2248, p. 514-532, 2002. Apresentado a Advances in cryptology - Asiacrypt'2001 - International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, 2001.

CELIK, M. U.; SHARMA, G.; TEKALP, A. M.; SABER E. Reversible data hiding. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 2002, Rochester, New York, USA. **ICIP Proceedings**. Piscataway: IEEE, 2002a. v. 2, p. 157-160.

CELIK, M. U.; SHARMA, G.; SABER, E.; TEKALP, A. M. Hierarchical watermarking for secure image authentication with localization. **IEEE Trans. Image Processing**, v. 11, n. 6, p. 585-595, 2002b.

CELIK, M. U.; SHARMA, G.; TEKALP, A. M.; SABER, E. Lossless generalized-LSB data embedding. **IEEE Transactions on Image Processing**, v. 14, n. 2, p. 253-266, 2005.

CHEN, Y.-Y.; PAN, H.-K.; TSENG, Y.-C. A secure data hiding scheme for binary images. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, 2000, Antibes, France. **ISCC Proceedings**. Piscataway: IEEE, 2000. p. 750-755.

CHUN, I. G.; HA, S. A Robust Printed Image Watermarking Based on Iterative Halftoning Method. **Lecture Notes in Computer Science**, Springer-Verlag, v. 2939, p. 200-211, 2003. Apresentado ao Int. Workshop on Digital Watermarking, Seoul, Korea, 2003.

CVISION TECHNOLOGIES. **A primer on JBIG2 bitonal compression**. Disponível em: <http://www.cvisiontech.com/pdfs/jb2_primer.pdf>. Acesso em: mar. 2007.

DE QUEIROZ, R. L.; FLECKENSTEIN, P. **Object modification for data embedding through template ranking**. Xerox Invention Proposal, 1999.

DIAS, D. **Modelo para representação eletrônica de cheques**. 2006. 97 p. Dissertação (Mestrado) – Faculdade de Tecnologia, Universidade de Brasília, Brasília, 2006.

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO. Divisão de Biblioteca. **Diretrizes para apresentação de dissertações e teses**. 3. ed. São Paulo: 2006. 103p. Disponível em: <http://www.poli.usp.br/Bibliotecas/PublicacoesOnLine/Diretrizes3.pdf>. Acesso em: 19 set. 2006.

FENG, J.-B.; LIN, I.-C.; TSAI, C.-S.; CHU, Y.-P. Reversible watermarking: current status and key issues. **International Journal of Network Security**, v. 2, n. 3, p. 161-171, 2006.

FRIDRICH, J., GOLJAN, M. and DU, R. Invertible authentication. In: SPIE SECURITY AND WATERMARKING OF MULTIMEDIA CONTENTS III, 2001, San Jose, California, USA. **Proceedings…** v. 3971, p. 197-208.

FRIDRICH, J.; GOLJAN, M.; DU, R. Lossless data embedding – new paradigm in digital watermarking. **EURASIP Journ. Appl. Sig. Proc.**, v. 2002, n. 2, p. 185-196, 2002.

FRIEDMAN, G. L. The trustworthy digital camera: restoring credibility to the photographic image. **IEEE Trans. on Consumer Electronics**, v. 39, p. 905-910, 1993.

FU, M. S.; AU, O. C. Data hiding by smart pair toggling for halftone images. In: IEEE INT. CONF. ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 2000, Istanbul, Turkey. **ICASSP Proceedings**. Piscataway: IEEE, 2000. v. 4, p. 2318-2321.

FU, M. S.; AU, O. C. Data hiding in halftone images by stochastic error diffusion. In: IEEE INT. CONF. ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 2001, Salt Lake City, Utah, USA. **ICASSP Proceedings**. Piscataway: IEEE, 2001. v. 3, p. 1965-1968.

FU, M. S.; AU, O. C. Data hiding watermarking for halftone images. **IEEE Trans. Image Processing**, v. 11, n. 4, p. 477-484, 2002.

FUJII, Y.; NAKANO, K.; ECHIZEN, I.; YOSHIURA, H.; TEZUKA, S. A method of maintaining the image quality of digital watermarking for binary images using local measures. **IPSJ Journal (Information Processing Society of Japan)**, v. 44, n. 08, p. 1872-1883, 2003. Em japonês.

GALLAGER, R.; VOORHIS, D. V. Optimal source codes for geometrically distributed integer alphabets. **IEEE Trans. Inform. Theory**, v. 21, p. 228-230, 1975.

GOLOMB, S. W. Run-length encodings. **IEEE Trans. Inform. Theory**, v. 12, p. 399-401, 1966.

HEL-OR, H. Z. Watermarking and copyright labeling of printed images. **Journal of Electronic Imaging**, v. 10, n. 3, p. 794-803, 2001.

HOLLIMAN, M.; MEMON, N. Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes. **IEEE Trans. Image Processing**, v. 9. n. 3, p. 432-441, 2000.

HONSINGER, C. W.; JONES, P. W.; RABBANI, M.; STOFFEL, J. C. **Lossless recovery of an original image containing embedded data**. United States Patent Number 6.278.791, August, 2001.

HOWARD, P. G.; KOSSENTINI, F.; MARTINS, B.; FORCHHAMMER, S.; RUCKLIDGE, W. J. The emerging JBIG2 standard. **IEEE Trans. Circ. Syst. Video Tech.**, v. 8, n. 7, p. 838-848, 1998.

HOWARD, P. G.; VITTER, J. S. Fast and efficient lossless image compression. In: IEEE DATA COMPRESSION CONFERENCE, 1993, Snowbird, Utah, USA. **DCC Proceedings**. Piscataway: IEEE, 1993. p. 351-360.

ISO/IEC. FCD14495. **Information technology – lossless and near-lossless compression of continuous-tone still images**. (Final committee draft for ISO/IEC international standard 14495-1). ISO/IEC, 1997. 79p. Disponível em: <http://www.jpeg.org/public/fcd14495p.pdf>. Acesso em: mai. 2005.

ISO/IEC. FCD11544. **Coded representation of picture and audio information — progressive bi-level image compression**. (Final committee draft for ISO/IEC

international standard 11544). ISO/IEC, 1992. 84p. Disponível em: <http://www.jpeg.org/public/fcd11544.pdf>. Acesso em: jun. 2004.

ISO/IEC. FCD14492. **Information technology — coded representation of picture and audio information — lossy/lossless coding of bi-level images**. (Final committee draft for ISO/IEC international standard 14492). ISO/IEC, 1999. 189p. Disponível em: <http://www.jpeg.org/public/fcd14492.pdf>. Acesso em: jun. 2004.

JPEG COMMITTEE. **JBIG2 development**. Disponível em: <http://www.jpeg.org/jbig/jbigpt2.html>. Acesso em: jun. 2004.

KIM, H. Y. Marcas d'água frágeis de autenticação para imagens em tonalidade contínua e esteganografia para imagens binárias e meio-tom. **Revista de Informática Teórica e Aplicada**, Instituto de Informática da UFRGS, v. 10, n. 1, p. 97-125, 2003.

KIM, H. Y. **IMG - Rotinas e programas em C++ para processamento de imagens e visão computacional**. São Paulo, 2004a. Disponível em <http://www.lps.usp.br/~hae/software>. Acesso entre abr. 2004 e set. 2006.

KIM, H. Y. **Projeto de operadores pela aprendizagem, difusão anisotrópica e marca d'água de autenticação**. 2004. 203p. Tese (Livre-Docência) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2004b.

KIM, H. Y. A new public-key authentication watermarking for binary document images resistant to parity attacks. In: IEEE INT. CONF. ON IMAGE PROCESSING, 2005, Genoa, Italy. **ICIP Proceedings.** Piscataway: IEEE, 2005. v. 2, p. 1074-1077.

KIM, H. Y. **ProEikon - Rotinas e programas em C++ para processamento de imagens e visão computacional**. São Paulo, 2006. Disponível em <http://www.lps.usp.br/~hae/software>. Acesso em 31 out. 2006.

KIM, H. Y.; AFIF, A. Secure authentication watermarking for binary images. In: BRAZILIAN SYMP. ON COMP. GRAPH. AND IMAGE PROCESSING, 16., 2003, São Carlos. **SIBGRAPI Proceedings**. São Carlos: USP, 2003. p. 199-206.

KIM, H. Y.; AFIF, A. Secure authentication watermarking for halftone and binary images. **Int. J. Imaging Systems and Technology**, v. 14, n. 4, p. 147-152, 2004.

KIM, H. Y.; DE QUEIROZ, R. L. A public-key authentication watermarking for binary images. In: IEEE INT. CONF. ON IMAGE PROCESSING, 2004, Singapore. **ICIP Proceedings.** Piscataway: IEEE, 2004a. p. 3459-3462.

KIM, H. Y.; DE QUEIROZ, R. L. Alteration-locating authentication watermarking for binary images. **Lecture Notes in Computer Science**, Springer-Verlag, v. 3304, p. 125-136, 2004. Apresentado ao International Workshop on Digital Watermarking, Seoul, Korea, 2004b.

KIM, H. Y.; PAMBOUKIAN, S. V. D.; BARRETO; P. S. L. M.; LI, C.-T. (Org.). **Multimedia forensics and security**. Coventry, UK: Idea Group Publishing, Information Science Reference – ISR, 2007. Capítulo de livro aguardando aprovação para publicação.

KNOLL, A. **Compression of bi-level images**. National Library of the Czech Republic, Prague, 2007. (Compressor Performance Report). Disponível em: <http://digit.nkp.cz/knihcin/digit/vav/bi-level/Compression_Bi-level_Images.html>. Acesso em: mar. 2007.

LI, C. T.; LOU, D. C.; CHEN, T. H. Image authentication and integrity verification via content-based watermarks and a public key cryptosystem. IEEE INT. CONF. IMAGE PROCESSING, 2000, Vancouver, Canada. **ICIP Proceedings**. Piscataway: IEEE, 2000. v. 3, p. 694-697.

MAXEMCHUK, N. F.; LOW, S. Marking text documents. IEEE INT. CONF. IMAGE PROCESSING, 1997, Santa Barbara, USA. **ICIP Proceedings**. Piscataway: IEEE, 1997. v. 3, p. 13-17.

MENEZES, A. J.; VAN OORSCHOT, P. C.; VANSTONE, S. A. **Handbook of applied cryptography**. CRC Press, 1997. Disponível em: <http://www.cacr.math.uwaterloo.ca/hac/>. Acesso em: mar. 2007.

NI, Z. C.; SHI, Y. Q.; ANSARI, N.; SU, W.; SUN, Q. B.; LIN, X. Robust lossless image data hiding. IEEE INT. CONF. MULTIMEDIA AND EXPO, 2004, Taipei, Taiwan. **ICME Proceedings**. Piscataway: IEEE, 2004. p. 2199-2202.

NISHIMURA, K.; SIBUYA, M. Probability to meet in the middle. **Journal of Cryptology**, v. 2, n. 1, p. 13-22, 1990.

PAMBOUKIAN, S. V. D.; KIM, H. Y. New public-key authentication watermarking for JBIG2 resistant to parity attacks. **Lecture Notes in Computer Science**, Springer-

Verlag, v. 3710, p. 286-298, 2005. Apresentado ao Int. Workshop on Digital Watermarking, Siena, Italy, 2005.

PAMBOUKIAN, S. V. D.; KIM, H. Y. Reversible data hiding and reversible authentication watermarking for binary images. In: SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS, 6., 2006 Santos. **Anais do SBSeg**. Santos: SBC, 2006.

PAMBOUKIAN, S. V. D.; KIM, H. Y.; DE QUEIROZ, R. L. Watermarking JBIG2 text region for image authentication. In: IEEE INT. CONF. ON IMAGE PROCESSING, 2005, Genoa, Italy. **ICIP Proceedings**. Piscataway: IEEE, 2005. v. 2, p. 1078-1081.

PAN, H.-K.; CHEN, Y.-Y.; TSENG, Y.-C. A secure data hiding scheme for two-color images. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, 2000, Antibes, France. **ISCC Proceedings**. Piscataway: IEEE, 2000. p. 750-755.

PEI, S. C.; GUO, J. M. Hybrid pixel-based data hiding and block-based watermarking for error-diffused halftone images. **IEEE Trans. on Circuits and Systems for Video Technology**, v. 13, n. 8, p. 867-884, 2003.

PUTTINI, R. S.; DE SOUZA JR., R. T. **Criptografia, autenticação e assinatura digital**. Material de apoio à disciplina “Segurança de Redes” do Departamento de Engenharia Elétrica da Universidade de Brasília. Disponível em: <http://www.redes.unb.br/security/cripto.pdf>. Acesso em: mar. 2007.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. M. A method for obtaining digital signatures and public-key cryptosystems. **Communications of the ACM**, v. 21, p. 120-126, 1978.

SALOMON, D. **Data compression: the complete reference**. 3rd ed. New York: Springer-Verlag, 2004. 920p.

SCHNEIER, B. **Applied cryptography**. 2nd ed. John Wiley & Sons, 1996.

SHI, Y. Q. Reversible data hiding. **Lecture Notes in Computer Science**, Springer-Verlag, v. 3304, p. 1-13, 2004. Apresentado ao Int. Workshop on Digital Watermarking, Seoul, Korea, 2004.

TIAN, J. Wavelet-based reversible watermarking for authentication. In: SPIE SECURITY AND WATERMARKING OF MULTIMEDIA CONTENTS IV, 2002, San Jose, California, USA. **Proceedings…** v. 4675, p. 679-690.

TIAN, J. Reversible data embedding using difference expansion. **IEEE Transactions on Circuits Systems and Video Technology**, v. 13, n. 8, p. 890-896, 2003.

TOMPKINS, D. A. D.; KOSSENTINI, F. A fast segmentation algorithm for bi-level image compression using JBIG2. In: IEEE INT. CONF. ON IMAGE PROC., 1999, Kobe, Japan. **ICIP Proceedings**. Piscataway: IEEE, 1999. p. 224-228.

TSAI, C. L.; FAN, K. C.; CHUNG, C. D.; CHUANG, T. C. Data hiding of binary images using pair-wise logical computation mechanism. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO, 2004, Taipei, Taiwan. **ICME Proceedings**. Piscataway: IEEE, 2004a. v. 2, p. 951-954.

TSAI, C. L.; FAN, K. C.; CHUNG, C. D.; CHUANG, T. C. Reversible and lossless data hiding with application in digital library. In: IEEE INTERNATIONAL CARNAHAN CONFERENCE ON SECURITY TECHNOLOGY, 2004, Albuquerque, New Mexico, USA. **ICCST Proceedings**. Piscataway: IEEE, 2004b. p. 226-232.

TSENG, Y.-C.; CHEN, Y.-Y.; PAN, H.-K. A secure data hiding scheme for binary images. **IEEE Trans. on Communications**, v. 50, n. 8, p. 1227-1231, 2002.

WEINBERGER, M. J.; SEROUSSI, G.; SAPIRO, G. LOCO-I: A low complexity, context based, lossless image compression algorithm. In: DATA COMPRESSION CONFERENCE, 1996, Snowbird, Utah, USA. **DCC Proceedings**. Piscataway: IEEE, 1996. p. 140-149.

WONG, P. W. A watermark for image integrity and ownership verification. In: IS&T PIC CONFERENCE, 1998, Portland, Oregon, USA. **Proceedings…** Também disponível como Tech. Rep. HPL-97-72, Hewlett-Packard Labs, May 1997.

WONG, P. W. A public key watermark for image verification and authentication. In: IEEE INT. CONF. IMAGE PROCESSING, 1998, Chicago, Illinois, USA. **ICIP Proceedings**. Piscataway: IEEE, 1998. v. 1, p. 455-459.

WONG, P. W.; MEMON, N. Secret and public key image watermarking schemes for image authentication and ownership verification. **IEEE Trans. Image Processing**, v. 10, n. 10, p. 1593-1601, 2001.

WU, M.; LIU, B. Data hiding in binary image for authentication and annotation. **IEEE Trans. on Multimedia**, v. 6, n. 4, p. 528-538, 2004.

WU, M.; TANG, E.; LIU, B. Data hiding in digital binary image. In: IEEE INT. CONF. MULTIMEDIA AND EXPO, 2000, New York, USA. **ICME Proceedings**. Piscataway: IEEE, 2000.

WU, X. Lossless compression of continuous-tone images via context selection, quantization, and modeling. **IEEE Transactions on Image Processing**, v. 6, n. 5, p. 656-664, 1997.

YE, Y.; COSMAN, P. Dictionary design for text image compression with JBIG2. **IEEE Trans. Image Processing**, v. 10, n. 6, p. 818-828, 2001.

YE, Y.; SCHILLING, D.; COSMAN, P.; KO, H. H. Symbol dictionary design for the JBIG2 standard. In: DATA COMPRESSION CONFERENCE, 2000, Snowbird, Utah, USA. **DCC Proceedings**. Piscataway: IEEE, 2000. p. 33-42.

YEUNG, M. M.; MINTZER, F. An invisible watermarking technique for image verification. In: IEEE INT. CONF. IMAGE PROCESSING, 1997, Washington, DC, USA. **ICIP Proceedings**. Piscataway: IEEE, 1997. v. 1, p. 680-683.

ZAMBONI, L.; PAMBOUKIAN, S. V. D.; BARROS, E. A. R. **C++ para universitários**. São Paulo: Páginas & Letras, 2006. 402p.

ZHAO, J.; KOCH, E. Embedding robust labels into images for copyright protection. In: INT. CONG. INTELLECTUAL PROPERTY RIGHTS FOR SPECIALIZED INFORMATION, KNOWLEDGE AND NEW TECHNOLOGIES, 1995, Vienna, Austria. **Proceedings…** p. 242-251.

APÊNDICE A – PUBLICAÇÕES DO AUTOR RELACIONADAS COM A TESE

KIM, H. Y.; PAMBOUKIAN, S. V. D.; BARRETO; P. S. L. M.; LI, C.-T. (Org.). **Multimedia forensics and security**. Coventry, UK: Idea Group Publishing, Information Science Reference – ISR, 2007. Capítulo de livro aguardando aprovação para publicação.

PAMBOUKIAN, S. V. D.; KIM, H. Y. New public-key authentication watermarking for JBIG2 resistant to parity attacks. **Lecture Notes in Computer Science**, Springer-Verlag, v. 3710, p. 286-298, 2005. Apresentado ao Int. Workshop on Digital Watermarking, Siena, Italy, 2005.

PAMBOUKIAN, S. V. D.; KIM, H. Y.; DE QUEIROZ, R. L. Watermarking JBIG2 text region for image authentication. In: IEEE INT. CONF. ON IMAGE PROCESSING, 2005, Genoa, Italy. **ICIP Proceedings**. Piscataway: IEEE, 2005. v. 2, p. 1078-1081.

PAMBOUKIAN, S. V. D.; KIM, H. Y. Reversible data hiding and reversible authentication watermarking for binary images. In: SIMPÓSIO BRASILEIRO EM SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS, 6., 2006 Santos. **Anais do SBSeg**. Santos: SBC, 2006.